Learning from Design Experience in an Agent-Based Design System

Jarrod Moss, Jonathan Cagan[1], Kenneth Kotovsky
Carnegie Mellon University, USA

Abstract. A simple learning mechanism was added to an agent-based computational design system to see if it could then transfer knowledge across problems. An existing system, A-Design, was enhanced by giving it the ability to store useful design knowledge in a memory store so that this knowledge could be used in new design problems. Three electromechanical design problems were used to evaluate this new learning mechanism, and results indicate that this simple learning mechanism is successful at transferring design knowledge to new problems with some limitations.

1. Introduction

Human designers typically become better at designing devices in a domain as their experience in that domain increases. This learning process is gradual and has been documented in a number of other domains (Chase and Simon 1973, Larkin et al. 1980, Reitman 1976, Richman et al. 1995). The reason for this improvement in performance with experience is due in part to the ability of the designer to transfer knowledge learned in one problem to the problem that is currently being pursued. The ability to transfer knowledge between problems is an important process which has been implemented in only a few computational design systems.

Case-based design systems such as CADET (Sycara et al. 1991), ARCHIE-II (Domeshek and Kolodner 1991) , and Kritik2 (Goel et al. 1997) all have the capability of transferring the knowledge they have stored as cases to new problems. However, only some of the systems have the capability of indexing new cases into memory so that the design experience can accumulate over time (Goel et al. 1997). While these systems are able to implement a form of knowledge transfer, there are other forms in which the knowledge being transferred is not an exact part of a previously encountered design. There are some cases where transfer of more abstract knowledge

[1] Author of contact, Dept. of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, E-mail: cagan@cmu.edu

may be beneficial. In these cases, another transfer process may be required such as analogy.

Analogy is a powerful method that may be used to transfer knowledge both within and across domains. Analogies could be used to increase understanding of a novel design by allowing the designer to map previous experience onto the new device. There are many other uses for analogy in design, and there has been some work on models of design by analogy (Bhatta and Goel 1996, Howe et al. 1986, Huhns and Acosta 1988). Other methods of transfer besides case-based reasoning and analogy have also been explored.

However, there have been fewer attempts to incorporate knowledge transfer into more search oriented design systems. These systems usually employ a form of search that takes advantage of computational power to search a large number of possible designs, and they have been based on traditional AI search techniques (Ulrich 1988, Welch and Dixon 1994), genetic algorithms (Brown and Hwang 1993), simulated annealing (Szykman and Cagan 1995), and a number of other techniques. This paper describes an attempt to modify such a design system in order to incorporate some knowledge transfer processes.

A-Design is a multi-agent design system based on an iterative stochastic algorithm which in many ways resembles a genetic algorithm (Campbell et al. 1999, Campbell et al. 2000, Campbell et al. 2001). The fact that the system is agent-based provides a natural location for the new learning processes since they can be embodied in the agents. The work presented here is an attempt to augment A-Design with a simple learning mechanism that allows it to transfer knowledge across design problems. The idea is to give the system the capability to extract knowledge as it solves design problems which can then be used to improve performance when solving novel problems. This process allows A-Design to work with design knowledge at an abstract level which is not tied to a specific design, but using this knowledge does not require a complex analogical process. An introduction to A-Design is presented first followed by a description of the changes to the system that allow it to learn from its design experience.

2. Background: A-Design

An introduction to A-Design is required in order to explain how the agents in the system produce designs, and how the system was modified in order to include the new learning processes. A detailed description of A-Design can be found in Campbell (Campbell 2000). A-Design is an agent-based design system that produces an array of conceptual design solutions in response to a set of input and output constraints for a design problem.

A design problem is specified to A-Design by specifying the input and output constraints of the desired device in A-Design's representation. For example, a punch press could be specified to the system by indicating that the input 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 (Figure 1). Along with these general input and output specifications, a number of other input/output constraints are specified such as the desire to minimize handle displacement in the punch press (shown as a goal of zero meters of displacement in Figure 1). This problem also specifies that the punch should only be displaced by .25m. In addition to these constraints a number of other objectives can be specified such as minimizing the cost and weight of the device. Once a problem and all of the associated objectives have been specified, A-Design's iterative design process attempts to produce a design that optimizes these design goals.

Punch Press problem

Input-device: Handle	Output-device: Punch
Input-force: 6 N	Output-force goal: 100 N
Input-displacement goal: 0 m	Output-displacement goal: .25 m

Objectives: minimize cost, minimize weight

Figure 1. The specification for the punch press problem.

2.1. ITERATIVE DESIGN PROCESS

A-Design's iterative design process is similar to a genetic algorithm because in each iteration a number of new design candidates are produced by the system, and then the best of these designs are chosen to be the basis for the production of new designs in the next iteration. The basic structure of A-Design's iterative design process is shown in Figure 2 along with the set of agents associated with each part of the process.

The design process begins with a set of configurations agents (C-agents) who construct candidate designs using a library of embodiments. Each C-agent adds one embodiment to an incomplete design until either the design is complete or a maximum number of embodiments have been added to the design. A candidate design starts off as just a set of input and output constraints to which components can be added. These input and output constraints are represented in structures called functional parameters (FP's). An FP represents the characteristics of an interface between components, and a C-agent utilizes the qualitative information in an FP to determine which embodiments in the catalog can be connected to the incomplete design. Once an embodiment is added to one of the FP's, the free ports of the embodiment are included as new FP's where future embodiments can be attached. A form of qualitative reasoning is used to update the constraints in the incomplete design as each embodiment is added. A candidate is complete once it has connected the input and output FP's and has qualitatively satisfied the input and output constraints. A C-agent chooses which embodiment to add to an incomplete design based on a set of preferences built into the agent, the current state of the incomplete design, and other influences originating from the manager agents in the system. For example, some C-agents may prefer hydraulic components while others prefer electrical ones or components connected in series over ones connected in parallel. The current state of the design can also influence which embodiment a C-agent selects. If a device should have a bounded displacement at the output such as in the punch press example in Figure 1, then an agent would prefer components which accomplished this goal when added to the system. The C-agents' choices are also influenced by feedback they receive from manager agents in the system as discussed below. Once the set of C-agents have constructed a set number of designs these designs are passed to a group of instantiation agents (I-agents).

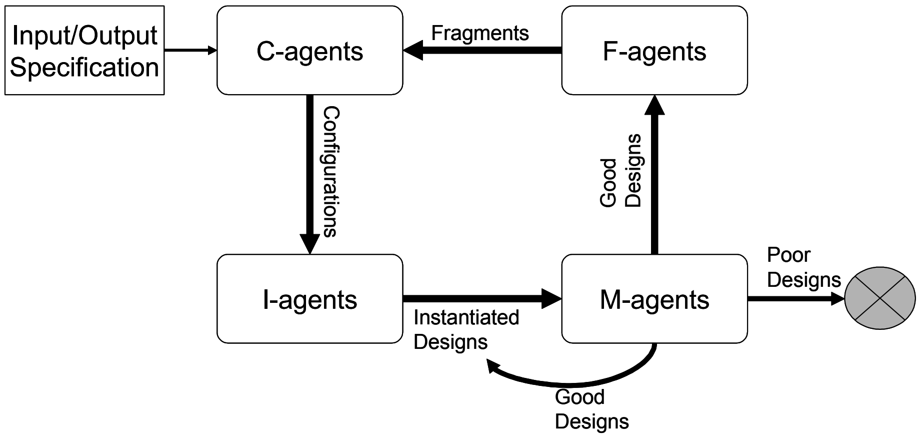

Figure 2. A-Design's iterative design process.

I-agents take the configuration designs from C-agents and instantiate the parameters in the system with values obtained from a catalog of components. Each embodiment in the configuration has a set of parameters such as length, weight, or resistance. These agents also have a set of built in preferences. For example, some I-agents may prefer using components that are low cost while others prefer those that are low weight. Once all of the components in a design have been instantiated, equations that describe the behavior of the design can be extracted. These equations allow the design to be evaluated on how well it satisfies the constraints of the given problem. For example, the equations extracted from a punch press design are then used to evaluate how much the handle is displaced since that is one of the specified constraints. Each device is evaluated along the dimensions specified in the problem, and these evaluations are combined into a linearly weighted sum. All of the devices are then sorted by this sum and good and poor devices are separated based on this ordering. These design partitions are then passed to a set of manager agents (M-agents).

M-agents take the current design population and produce feedback that controls how other agents in the system operate. First, the agents which contribute to good and poor designs are examined and the probabilities controlling how often those agents contribute to designs are adjusted. A C-agent that contributed to a number of good designs will be called more frequently than one who contributed to many poor designs. 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. These statistics are used for all of the agents except for the M-agents. Managers also look for trends in the design population. Trends are groups of agents or connected embodiments that appear together in a number of designs (see example in Figure 3). Good trends are found by examining the top six designs, and bad trends are found in the worst six designs. Good trends are placed on a todo list and bad trends on a taboo list. These lists work by encouraging agents to reproduce combinations of agents or embodiments that are on the todo list and discouraging agents from reproducing the groups on the taboo list. In this manner, the todo/taboo lists allow the M-agent to influence the designs that are generated in the next iteration of the design process, but they exert this influence only within a given run on a single problem.

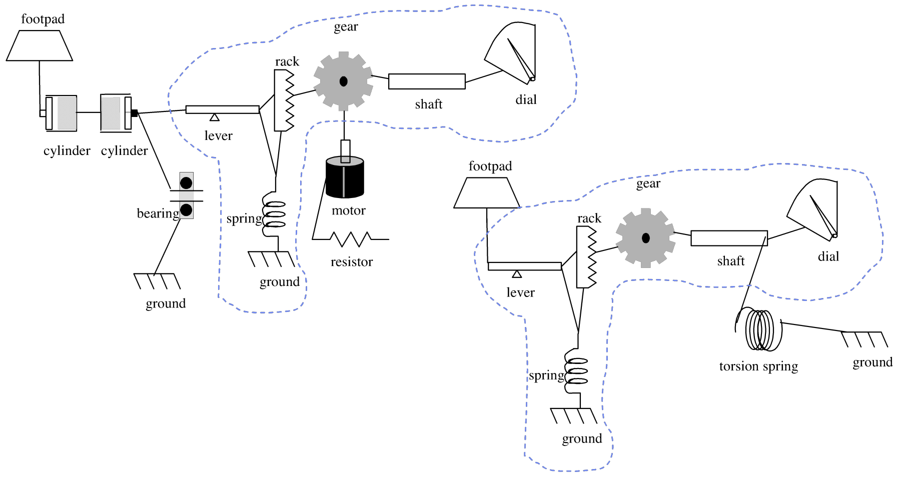

Figure 3. A trend is found by multiple designs with the result circled

In addition to passing the good designs from the current iteration into the next iteration, all good 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. The design population now consists of the good designs plus the newly reconstructed designs. In the next iteration the C-agents produce the number of new designs necessary to bring the design population back to the original number of designs, and this design population level is a parameter of the system. This iterative process is the basis of the A-Design system.

3. Learning Across Problems in A-Design

We extended A-Design so that it could learn from its design experiences. In order to do this A-Design needs to be able to extract knowledge from its problem solving activities, and once it has this knowledge there has to be some way of applying it to new design problems. One type of information that A-Design already knows how to extract is the good trends in a set of designs. The sets of interconnected embodiments that appear on the todo list are subsystems that appear in a number of good designs, and so it is likely then that these subsystems perform well in the current problem and may be worth remembering for future use.

A-Design's design process is series of iterations in which the current designs perform the same as or better than those in the previous iteration, and so it seemed to make sense that the subsystems found on the todo list in the final iteration were the ones to be remembered in some kind of memory store. Each subsystem appearing on this list is extracted and placed into memory as a chunk of knowledge. Each chunk in memory is indexed by its input and output constraints since this is the only information that is needed to determine if a component can be added to an incomplete design by the C-agents in the system. An example chunk consisting of a belt and pulley connected together is shown in Figure 4. This chunk extraction process is only half of the knowledge transfer process, and so a new set of C-agents were added to the system and given the ability to add chunks from memory to designs during the configuration part of the design process.

These memory agents (Mem-agents) add components to incomplete designs just as the C-agents do except that they are adding chunks from memory instead of embodiments from the embodiment catalog. There is a three step process by which a chunk is added to a design (Figure 5). First, a memory agent is called to work on an incomplete design. The memory agent then examines the open FP's in the system to determine where chunks may be added. The agent then looks in memory for a chunk that is compatible with one of more of the FP's in the design and adds this chunk. As mentioned before, all chunks are indexed in memory by their input and output constraints which are essentially input and output FP's for the chunk. Chunks can then be retrieved from memory either based on their input, their output, or both. Each of these retrieval methods is embodied in one Mem-agent. For example, the Mem-agent implementing the input retrieval strategy will search for a chunk in memory whose input constraints match one of the open FP's in the design.

Design Chunk

Belt-pulley-chunk

Isa: design-chunk

Input-domain: translation

Input-interface: bolt

Output-domain: rotation

Output-interface: shaft-hole

Components: (belt pulley)

Connectivity: port-1 of component-1 is connected to port-1 of component-2

Figure 4. An example design chunk

There are some cases in which multiple chunks in memory are retrieved as possible chunks to be added to a design. In this case, the agent needs a way of determining which chunk to add. The chunk could just be randomly selected from all possible chunks, or there could be some form of learning that takes place within the memory agent that allows it to pick the chunk most likely to produce a good design. The second option was implemented and is based on learning mechanisms found in the ACT-R model of human memory (Anderson and Lebiere 1998). Each memory agent is informed about the consequences of its actions by the manager agent, and the memory agent can then use this feedback to choose among multiple chunks when it has to. When a Mem-agent adds a chunk to a design, the design can either end up being a good design or a poor design. The proportion of good designs that result from using a particular chunk is calculated. When multiple chunks are retrieved, the probability that any particular one is chosen is just the proportion of good designs for that chunk divided by the sum of the proportions for all of the chunks retrieved from memory. If the proportion of good designs for a particular chunk is below some minimum level then this minimum proportion is used instead of the actual value so that every chunk has at least a small probability of being chosen regardless of its past performance. This mechanism is also similar to the strategy for move selection in the simulated annealing algorithm found in (Hustin and Sangiovanni-Vincentelli 1987). These new additions to A-Design were then tested to see if they allowed the system to successfully transfer knowledge to new problems.

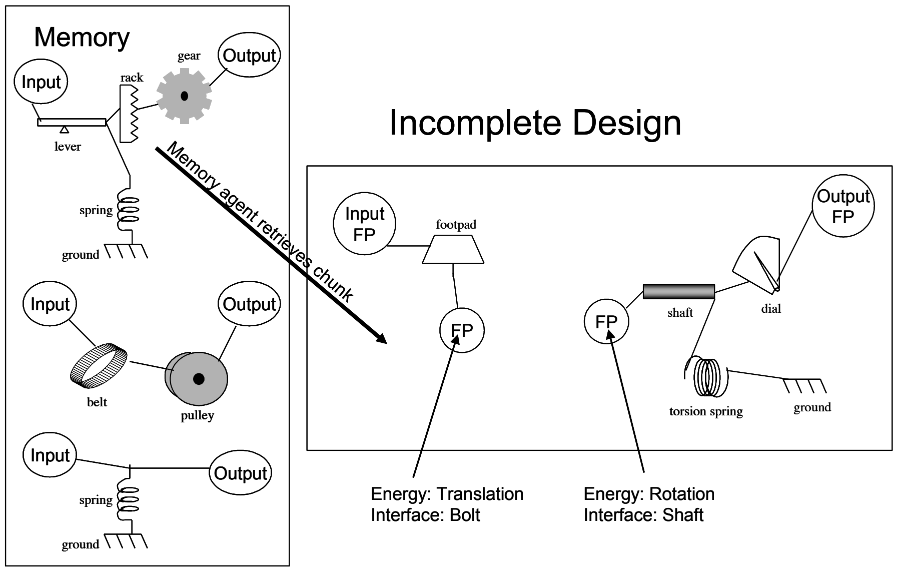

Figure 5. Memory agents retrieve chunks from memory

4. Testing Learning

The chunk learning mechanism of A-Design was tested to see if it allowed the system to demonstrate learning both within and between problems. Learning within the same problem is simply the case where A-Design works on a design problem, learns chunks, and then works on the same design problem again with the new chunks. Within problem learning should indicate whether the chunking mechanism is allowing the system to learn and apply useful knowledge about the design problem. Between problem learning occurs when A-Design applies chunks learned in one problem to a new design problem, and it is this type of learning that the chunking mechanism was designed to accomplish.

Within problem learning was evaluated by running A-Design 20 times on one problem, and in each of these runs, a set of chunks was generated from the final design population. A-Design was then run again on the same problem 20 times, once with each set of learned chunks. The number of iterations in each run was 60, and the design population was set to 120 designs. A graph of A-Design's performance on a design problem can be constructed by taking the evaluation score for the best design at each iteration and averaging this evaluation across all twenty runs of the problem.

This leads to a graph similar to Figure 6 in which each line in the figure shows A-Design's performance on one design problem. The performance difference between the initial problem and the second attempt at the problem was measured in three ways (see Figure 6): the evaluations of the initial and final designs for each condition can be compared and the number of iterations until a specified evaluation level is reached can be compared. The comparison of the number of iterations can be looked at as the number of iterations that were saved by the presence of chunks and it will be referred to as a measure of savings. The criterion evaluation level for each problem presented here is just the average evaluation score obtained in the final best design without using a set of stored chunks, i.e. just the performance of the A-Design system without the new learning mechanism. Any differences in performance from the first attempt to the second can then be attributed to the chunks that were used in the second attempt.

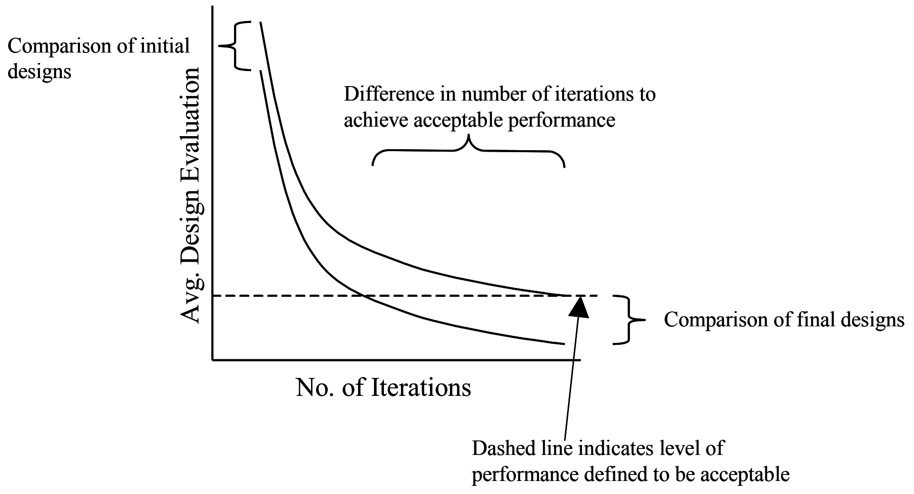

Figure 6. An illustration of the three measures of performance

4.1. DESIGN PROBLEMS

Three electromechanical design problems were used to assess the benefits of the chunking mechanism: 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.

These specific problems were utilized 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. However, 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. It was thought that this problem shares little with the weighing machine so it was used 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.

5. Results

5.1. WITHIN PROBLEM RESULTS

Within problem results for the three design problems can be seen in Figures 7-9 and Tables 1-3. There appears to be within problem learning in each of the three problems (lower evaluation scores are better). A series of paired t-tests was run to assess the statistical significance of any differences in the three performance measures discussed above.

5.1.1. Weighing Machine

In the weighing machine problem only the comparison of the initial designs was not significantly different (t(19) = 1.19, p = .12), but both the comparison of the last designs (t(19) = 2.47, p = .01), and the amount of savings were significant (t(19) = 3.73, p < .001). On average, the final design produced without any chunks had an evaluation of 21.7 as compared to 19.3 with chunks, and the number of iterations required to reach an evaluation of 773 was 46 without chunks and 31 with chunks.

TABLE 1. Within problem performance measures for the weighing machine.

	Initial evaluations		Final evaluations		Iterations to criterion	
	μ	σ	μ	σ	μ	σ
No chunks	773	1001	21.7	4.90	45.6	16.0
Within problem chunks	465	632	19.3	3.29	30.7	20.3

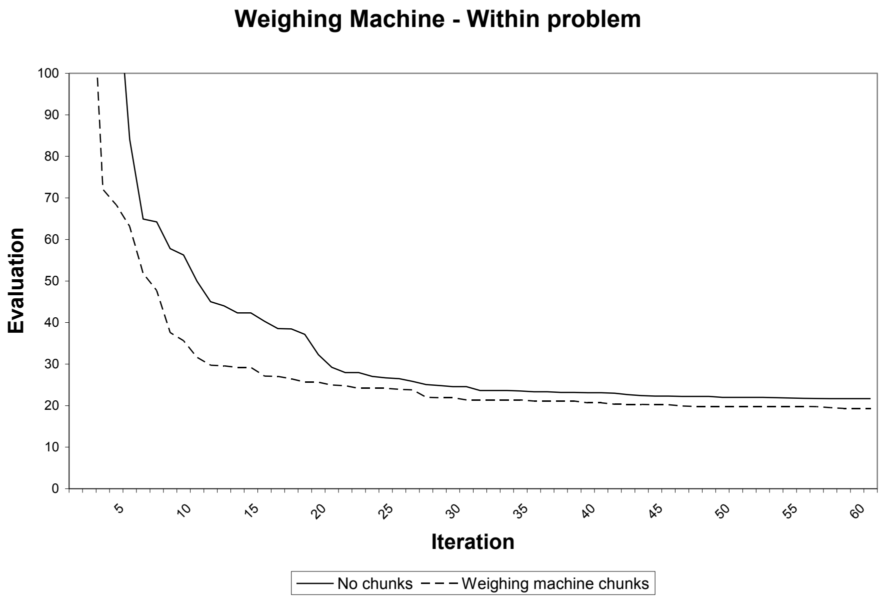

Figure 7. Within problem results for the weighing machine problem

5.1.2. Pressure Gauge

In the pressure gauge problem the comparison of initial designs (t(19) = 2.14, p = .02) and savings (t(19) = 1.82, p = .04) were significant, but the comparison of final design evaluations was not (t(19) = 1.19, p = .12). On average, the initial design produced without any chunks was 50,302 as compared to 35,358 with chunks, and the number of iterations required to reach an evaluation of 50,302 was 44 without chunks and 32.7 with chunks.

TABLE 2. Within problem performance measures for the pressure gauge.

	Initial evaluations		Final evaluations		Iterations to criterion	
	μ	σ	μ	σ	μ	σ
No chunks	50302	25775	3258	807	44.0	18.0
Within problem chunks	35358	26275	3005	180	32.7	23.5

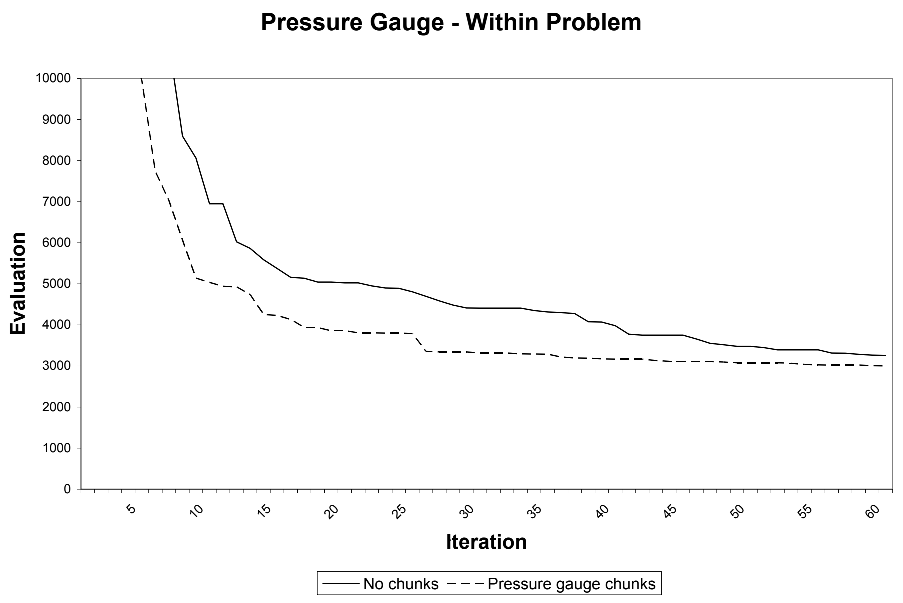

Figure 8. Within problem results for the pressure gauge problem.

5.1.3. Punch Press

In the punch press problem the comparison of initial designs (t(19) = 3.32, p = .002), final designs (t(19) = 3.34, p = .002), and savings (t(19) = 2.73, p = .007) were significant. On average, the initial design produced without any chunks was 9,046 as compared to 6,675 with chunks, the final design produced without any chunks was 1,658 compared to 1,016 with chunks, and the number of iterations required to reach an evaluation of 1,658 was 38.9 without chunks and 23 with chunks.

TABLE 3. Within problem performance measures for the punch press.

	Initial evaluations		Final evaluations		Iterations to criterion	
	μ	σ	μ	σ	μ	σ
No chunks	9046	1327	1658	524	38.9	21.4
Within problem chunks	6675	2762	1016	517	23.0	18.7

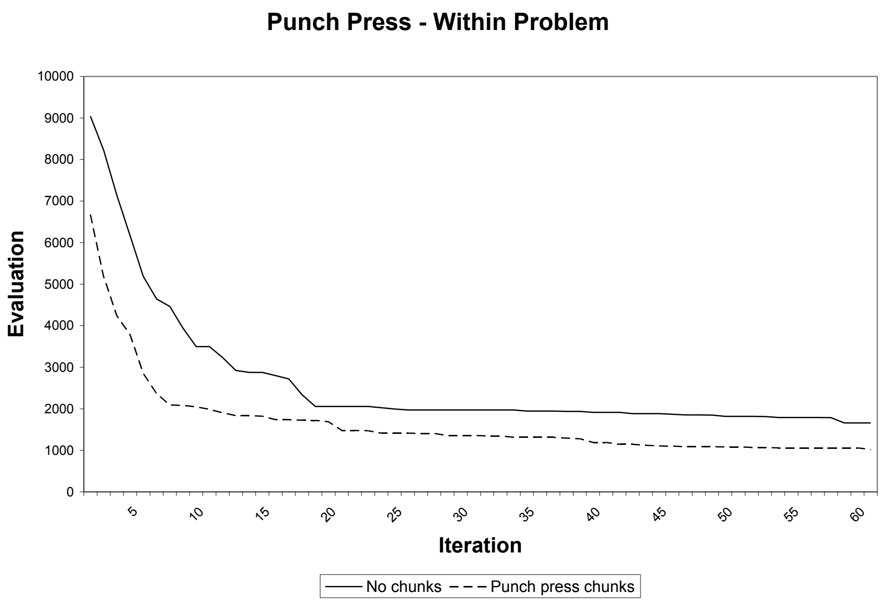

Figure 9. Within problem results for the punch press problem.

5.2. BETWEEN PROBLEM RESULTS

The results for between problem transfer can be seen in Figures 10-12 and Tables 4-6. For each problem, a one way ANOVA was run for each comparison followed by a series of planned contrasts if the ANOVA indicated any significant differences.

5.2.1. Weighing Machine

Figure 10 indicates that the three conditions perform about the same, and there were no significant differences in initial evaluations (F(2,57) = .74, p = .24), final evaluations (F(2,57) = .06, p = .47), or savings (F(2,57) = .804, p = .23) for the weighing machine problem.

TABLE 4. Between problem performance measures for the weighing machine.

	Initial evaluations		Final evaluations		Iterations to criterion	
	μ	σ	μ	σ	μ	σ
No chunks	773	1001	21.7	4.90	45.6	16.0
Pressure gauge chunks	700	1397	21.2	4.21	49.5	12.6
Punch press chunks	405	352	21.4	4.73	43.2	18.4

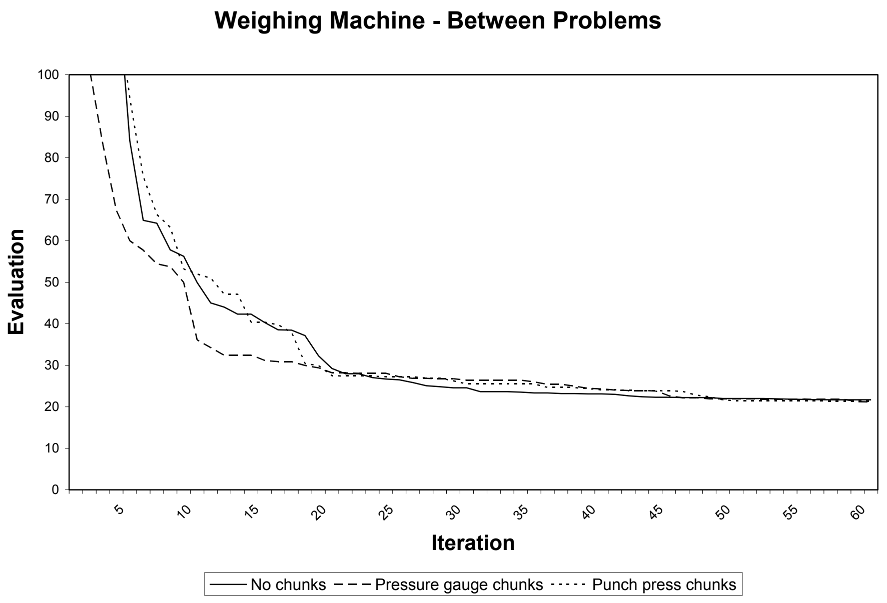

Figure 10. Between problem results for the weighing machine problem

5.2.2. Pressure Gauge

Figure 11 indicates that the two conditions with chunks perform better than solving the problem with no previous knowledge. There were significant differences in final evaluations (F(2,57) = 2.94, p = .03) and savings (F(2,57) = 4.53, p = .008) but not for the initial evaluations (F(2,57) = .9, p = .21). A series of contrasts reveal that chunks from the weighing machine problem allow the system to produce better designs than the no chunk condition, and both of the chunk conditions produce significant savings when compared to the no chunk condition. Final design evaluations in the weighing machine chunk condition had an average of 2,766 while the average in the no chunk condition was 3,258. Chunks from the weighing machine problem help the system to reach an evaluation of 3,258 in 28.8 iterations as compared to 44 iterations for the no chunk condition, and chunks from the punch press problem also allow the system to reach the criterion evaluation in 29 iterations. Neither of the two between problem transfer cases performed better than the within problem transfer in this problem.

TABLE 5. Between problem performance measures for the pressure gauge.

	Initial evaluations		Final evaluations		Iterations to criterion	
	μ	σ	μ	σ	μ	σ
No chunks	50302	25775	3258	807	44.0	18.0
Weigh. machine chunks	42190	23483	2766	441	28.8	18.7
Punch press chunks	52670	28337	2889	705	28.9	18.4

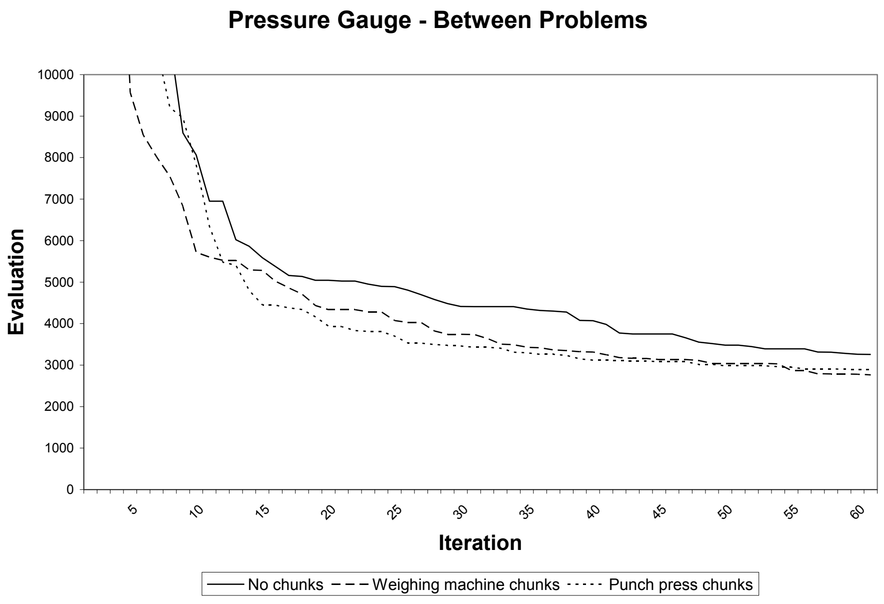

Figure 11. Between problem results for the pressure gauge problem.

5.2.3 Punch Press

Figure 12 indicates that the two conditions with chunks perform better than solving the problem with no previous knowledge. There were significant differences in final evaluations (F(2,57) = 4.09, p = .01) but not for the initial evaluations (F(2,57) = .65, p = .26) or savings (F(2,57) = 1.37, p = .13). A series of contrasts reveal that both of the chunk conditions produce significantly better final designs when compared to the no chunk condition. Final design evaluations in the weighing machine chunk condition had an average of 1,298 and in the pressure gauge chunk condition the average was 1,271, while the average in the no chunk condition was 1,658. Again neither of the between problem transfer cases performed better than the within problem transfer.

TABLE 6. Between problem performance measures for the punch press.

	Initial evaluations		Final evaluations		Iterations to criterion	
	μ	σ	μ	σ	μ	σ
No chunks	9046	1327	1658	524	38.9	21.4
Weigh. machine chunks	8583	1493	1298	413	28.7	20.0
Pressure gauge chunks	8472	2139	1271	491	30.7	20.9

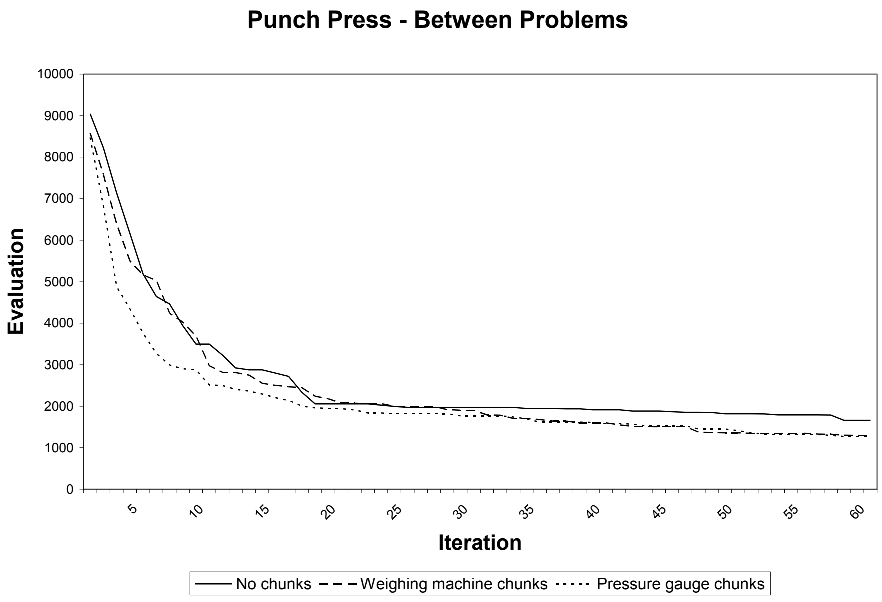

Figure 12. Between problem results for the punch press problem.

6. Discussion

The chunking mechanism did result in successful knowledge transfer in some cases. In particular there was no between problem transfer when working on the weighing machine problem, but the other two problems did show transfer on at least one of the three measures. In general, within problem transfer was the most effective type of transfer as none of the between problem cases outperformed the within problem case in any of the problems. In cases where between problem transfer did occur, the performance measures that improved were the final design evaluations and the savings measure. So while the knowledge from another problem did not produce better initial designs in any of the three problems, it was able to improve performance throughout the iterative design process. On the other hand, knowledge transfer within the same problem produced improvements in all three performance measures including the initial design evaluations in some cases. This could indicate that the chunks learned by the system could only be applied to a new problem at some later stage of the design process. Another explanation is that only some of the chunks learned in another problem are beneficial to the current problem, and it takes many attempts to find which of the chunks are beneficial when applied to the current design problem.

It was hypothesized that most of the between problem transfer would occur between the weighing machine and pressure gauge problems due to their similarity. In the pressure gauge problem, the within problem chunks lead to the most improvement followed by the chunks from the weighing machine problem and finally the punch press chunks. This graded transfer makes sense it is easiest for the system to apply knowledge from the same problem and somewhat more difficult to apply knowledge from a similar problem. However, in the weighing machine problem there was no between problem transfer. This behavior appears to be caused by the embodiments that occur in the chunks learned from the pressure gauge problem. Every pressure gauge takes some pressure source as its input, and, given the catalog that was supplied, the only way A-Design has of transforming this pressure into a translational motion is by using a hydraulic cylinder. So all pressure gauges have cylinders in them, and since chunks are found by extracting commonalities, a large portion of the chunks learned have the cylinder embodiment in them. None of the best weighing machine designs produced had cylinders in them because of the high cost of this embodiment, and so these pressure gauge chunks do not appear to be very useful in the weighing machine problem. This seems like the most likely cause of the asymmetric transfer.

Overall, there was some success in producing knowledge transfer with a simple chunking mechanism. The mechanism takes advantage of A-Design's preexisting trend finding function to extract chunks, and all that was required to use these chunks was a new type of C-agent that looks in a memory store of chunks instead of the embodiment library. This method of knowledge transfer still relies on the power of computational search in A-Design's iterative design process. So while this mechanism was able to produce transfer, it produced modest results when transferring chunks between problems. The system operates on embodiment level similarities between problems, and it lacks any more powerful method of transfer. A more powerful transfer mechanism would probably rely on more knowledge based methods in addition to or instead of brute force computational search. For example, cognitive processes such as abstraction, functional reasoning, and analogy can produce knowledge transfer that operates across vastly different problems and domains. Research into these and other cognitive processes has produced a basic understanding of knowledge transfer, and studying these processes in the domain of design should enable the construction of more knowledgeable design automation and assistance tools.

Future work on this topic will focus on the organization of knowledge in memory as this is one difference between experts and novices in many domains including chess (Chase and Simon 1973), physics (Larkin et al. 1980), game playing (Reitman 1976) and electrical diagrams (Egan and Schwartz 1979). Organization of information in memory is likely to be a difference between experts and novices in the domain of design as well. Perhaps providing the agents in A-Design with a better way of organizing and retrieving chunks would enable better transfer. This is one area in which the cognitive basis for expertise has been fairly well studied in a number of domains, and it would be beneficial to extend this understanding into the design domain.

The use of agents in computational design systems may also have implications for cognitive models of the design process. For example, each agent could be a model of an individual designer, and this type of multi-agent model could be used to study design team interactions and other group design phenomena. Alternatively, a multi-agent system could be constructed that was intended to model the cognitive processing of an individual. This may provide an alternative type of model in which to study the cognitive processes underlying design as opposed to traditional cognitive models (Anderson and Lebiere 1998). A-Design has been the basis for this initial exploration into methods of acquiring and transferring knowledge to new design problems, and it is apparent that further research in this area should yield benefits for computational design systems and further our understanding of the cognitive processes underlying the design process.

Acknowledgments

This research effort was partially sponsored by a National Defense Science and Engineering Fellowship and by the Air Force Office of Scientific Research, Air Force Material Command, USAF, under grant numbers F49620-98-1-0172 and F49620-01-1-0050. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of AFOSR or the U.S. Government.

The authors would also like to acknowledge the assistance and support provided by Matt Campbell while learning the intricacies of the A-Design system and while generating some initial ideas on how the system could be extended.

References

Anderson, J R and Lebiere, C: 1998, *The atomic components of thought,* Lawrence Erlbaum, Mahwah, NJ.

Bhatta, S R and Goel, A K: 1996, From design experiences to generic mechanisms: Model-based learning in analogical design, *Artificial Intelligence for Engineering Design Analysis and Manufacturing,* **10**(2)**:** 131-136.

Brown, D R and Hwang, K Y: 1993, Solving fixed configuration problems with genetic search, *Research in Engineering Design,* **5**(2)**:** 80-87.

Campbell, M I: 2000, The A-Design invention machine: A means of automating and investigating conceptual design, Ph.D. Dissertation, Carnegie Mellon University, Pittsburgh, PA.

Campbell, M I, Cagan, J and Kotovsky, K: 1999, A-design: An agent-based approach to conceptual design in a dynamic environment, *Research in Engineering Design,* **11**(3)**:** 172-192.

Campbell, M I, Cagan, J and Kotovsky, K: 2000, Agent-based synthesis of electromechanical design configurations, *Journal of Mechanical Design,* **122**(1)**:** 61-69.

Campbell, M I, Cagan, J and Kotovsky, K: 2001, Learning from design experience: Todo/Taboo guidance, *2001 ASME Design Engineering Technical Conferences and Computers in Engineering Conference: Design Theory and Methodology Conference,* Pittsburgh, PA, DETC01/DTM-21687.

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

Domeshek, E A and Kolodner, J L: 1991, Toward a Case-Based Aid for Conceptual Design, *International journal of expert systems,* **4**(2)**:** 201-220.

Egan, D E and Schwartz, B J: 1979, Chunking in Recall of Symbolic Drawings, *Memory & Cognition,* **7**(2)**:** 149-158.

Goel, A K, Bhatta, S R and Stroulia, E: 1997, Kritik: An early case-based design system *in* Maher, M L and Pu, P (Eds), *Issues and Applications of Case-Based Reasoning in Design,* Lawrence Erlbaum, Mahwah, NJ, pp. 87-132.

Howe, A E, Cohen, P R, Dixon, J R and Simmons, M K: 1986, Dominic: A domain-independent program for mechanical engineering design, *Artificial Intelligence in Engineering,* **1**(1)**:** 289-299.

Huhns, M N and Acosta, R D: 1988, Argo - a System for Design by Analogy, *IEEE Expert-Intelligent Systems & Their Applications,* **3**(3)**:** 53-68.

Hustin, S and Sangiovanni-Vincentelli, A: 1987, TIM, a new standard and cell placement program based on the simulated annealing algorithm, *IEEE Physical Design Workshop on Placement and Floorplanning,* Hilton Head, SC.

Larkin, J H, McDermott, J, Simon, D P and Simon, H A: 1980, Models of Competence in Solving Physics Problems, *Cognitive Science,* **4**(4)**:** 317-345.

Reitman, J S: 1976, Skilled Perception in Go - Deducing Memory Structures from Inter-Response Times, *Cognitive Psychology,* **8**(3)**:** 336-356.

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

Sycara, K, Chandra, D N, Guttal, R, Koning, J and Narasimhan, S: 1991, CADET: A Case-Based Synthesis Tool for Engineering Design, *International journal of expert systems,* **4**(2)**:** 157-188.

Szykman, S and Cagan, J: 1995, A Simulated Annealing-Based Approach to 3-Dimensional Component Packing, *Journal of Mechanical Design,* **117**(2)**:** 308-314.

Ulrich, K T: 1988, Computation and pre-parametric design, Technical Report 1043, AI Lab, MIT, Cambridge, MA.

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