Proceedings of DETC'01:
ASME 2000 Design Engineering Technical Conferences
and Computers and Information in Engineering Conference
Pittsburgh, Pennsylvania, September 9-12, 2001

DETC2001/DTM-21687

LEARNING FROM DESIGN EXPERIENCE: TODO/TABOO GUIDANCE

Matthew I. Campbell[1]
Department of Mechanical Engineering
University of Texas at Austin
Austin, TX 78712
mc1@mail.utexas.edu

Jonathan Cagan
Department of Mechanical Engineering
Carnegie Mellon University
Pittsburgh, PA 15213

Kenneth Kotovsky
Department of Psychology
Carnegie Mellon University
Pittsburgh, PA 15213

ABSTRACT

This paper introduces a new learning algorithm to be used within a larger automated design synthesis technique. This learning method, known as TODO/TABOO learning, is based on the idea that experience gained from past design activities greatly improves the efficiency of the design process and the value of future designs. While computational search techniques have the ability to search many design alternatives quickly, the human engineer can often arrive at a more elegant and robust solution by applying heuristics learned from past experiences. The TODO/TABOO learning method extracts common "trends" from previous designs and sorts them as either positive or negative learning criteria for use in future design activity. Results show how such a technique improves the quality of designs and efficiency of an existing automated search process.

1 INTRODUCTION

While many strides have been made towards automating parts of the engineering design process through computational methods, it is difficult to replace the intuitive decisions made by experienced engineering designers. Despite the fact that optimization techniques and expert systems have proven to be useful in guiding engineering design procedures, the decisions made based on previous experiences are often the most valued. Often design solutions produced by an automated computational process appear to be naïve as they lack the understanding to predict deficiencies in a design that humans can predict through simple experiential learning.

This paper reports on a new context based learning method for use in complex automated design problems called TODO/TABOO learning. It is integrated with a design search engine known as A-Design (Campbell, et al., 1999, 2000a, 2000b), but could be used in separate search processes or applications. The A-Design approach captures characteristics of human designers while taking advantage of a broad range of computational processes. Currently, the focus of A-Design is on the conceptual design problem of establishing the configuration of components for electro-mechanical devices given only a functional description of the artifact. In general, A-Design and the new learning method are useful to engineering problems that do not have a fixed set of variables or a fixed evaluation functions. The potential to handle variability in both design parameters and performance parameters makes A-Design uniquely suited to more conceptual engineering design problems than traditional optimization techniques

Within the scope of the A-Design process, the TODO/TABOO learning algorithm guides the iterative synthesis technique by learning from past successes and failures, thereby building an understanding of the successful and unsuccessful regions of the search space. The learned information is provided to a set of computational agents responsible for developing new solutions to the design problem. As in human problem solving, "trial and error" is not a simple random search for solutions; human designers absorb an abundance of information from studying failed attempts and from observing competing products. The experience gained through "trial and error" can lead to an efficient search downstream in the design process or in future design problems.

One of the significant conclusions of this research is that it provides an important step in overcoming the naïveté of automated search processes. As computational techniques address more and more ambitious design problems, the space of possible design solutions becomes immensely complex. Stochastic techniques provide a good foundation in overcoming these complex design spaces, but heuristic approaches used by

[1] Corresponding author: (512-232-9122) ETC 5.160-C2200, University of Texas, Austin, TX 78712-1063

humans in solving these problems can often result in more elegant and robust solutions. It is believed that the experience gained from past design activities greatly improves the efficiency and effectiveness of future design attempts.

The main goal of this paper is to present the details of TODO/TABOO learning and how it improves the A-Design automated design synthesis technique. First, the A-Design method is briefly described followed by an in depth presentation of the TODO/TABOO learning mechanism. Next, experimental results are detailed to show how keeping track of past design activity allows the process to converge more rapidly and to achieve better quality solutions.

2 A-DESIGN FRAMEWORK

A-Design developed as a combination of several computational design synthesis techniques to automate conceptual stages of the design process. In order to do this, various approaches were brought together to model aspects of how people solve conceptual design problems. A-Design's foundation is a search process similar to an optimization routine where designs are created and modified in an iterative process to arrive at a solution that best meets a set of criteria. In the most general case, the search process contains four basic tasks as shown in Figure 1.

The first task is to develop a *representation* for the design problem. The *representation* is crucial because it establishes the limits of the search space and the scope of design complexity. Conceptual design as accomplished by human designers operates on a level not limited by number of components, types of components, or types of component configurations. This establishes a very open-ended set of possible design solutions. In developing a computational *representation* of design function in A-Design, a language is developed to mimic the open-endedness of real design problems. In doing so, the *representation* draws upon various techniques including other design generative methods (Welch and Dixon, 1994; Ulrich and Seering, 1989; Schmidt and Cagan, 1998), linguistic approaches to representing design function (Stone and Wood, 1999), and descriptions of how components influence one another (Bracewell and Sharpe, 1996; Navinchandra, et al., 1991; Palmer and Shapiro, 1993; Chakrabarti and Bligh, 1996). The method of the *representation* developed for A-Design, which is described in detail in Campbell et al., (2000a), contains building block descriptions of individual components. There is no preconceived way in which the building block components can be configured, rather designs are constructed through the successive linking of components either in series or in parallel. In this manner, the *representation* remains unstructured so that complex and conceptual design problems can be accomplished.

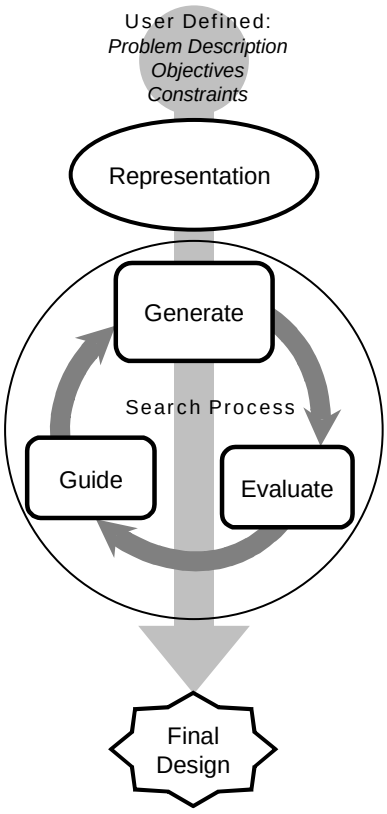


Figure 1: Conceptual Design as a Search Process has four distinct parts.

Figure 1 shows that the next three tasks in a search process occur within an iterative loop whereby many design states are visited in searching the space for the most successful design. The first of these steps is to *generate* alternatives. A-Design creates design alternatives through a hierarchy of software agents. The multi-agent subsystem of A-Design is inspired by work in artificial life (Langton, 1988; Holland, 1992), and Asynchronous Teams research (Talukdar, 1996). Artificial Life and Asynchronous Teams combine numerous software agents in a parallel and unstructured manner to produce an emergent behavior capable of thorough and efficient problem solving. It has often been found that multi-agent systems succeed when the demands for a more flexible and knowledge-based approach require the interaction of various software components (Lander, 1997).

The A-Design agents build, update and modify solutions through use of the design *representation* discussed above. Based on the various responsibilities and preferences, agents are divided into maker-, modification-, and manager-agent classes. Within these classes further classification can be made. For example, there are two types of maker-agents: those that choose how components should be connected and configured in a design (Configuration-agents), and those that find a specific component from a catalog to use in the design configuration (Instantiation-agents). These agents are knowledge-driven strategies for solving open-ended problems that collaborate with other similar agents to achieve a specified design goal. By cooperatively combining these similar strategies, a sense of parallel execution is achieved that produces a system with more robustness and variety in solving conceptual design problems. While all agents are driven to improve designs, invoking their

cooperation in a random manner provides A-Design with the means of overcoming stagnation in local optima.

After *generating* design alternatives, the solutions are *evaluated* to determine their individual worth. *Evaluation* is a complex and ever changing issue in conceptual design. Design goals can be difficult to determine and decisions about competing goals are often based on unknown or qualitative factors. As a result, A-Design has the ability to adapt to changes in the design goals throughout the search process. This adaptive approach builds on traditional multi-objective methods like Pareto optimality (Balachandran and Gero, 1984; Fonseca and Fleming, 1995). Also, multi-attribute utility theory as established by Keeney and Raiffa (1976) and utilized for engineering design by Thurston (1991) have influenced the manner in which A-Design handles multiple design objectives.

After the agents *generate* a set of designs, it is necessary to determine which alternatives are most useful for advancing toward successful designs and which alternatives have little or no design worth. In order to decide which designs to save and which to discard, A-Design employs a unique strategy whereby candidates are divided into three separate populations labeled Pareto designs, Good designs and Poor designs as shown in Figure 2. In this visualization, the orthogonal axes represent independent objectives that are to be minimized. The dots represent designs that are plotted by their objective values. In this division, Pareto designs are a mathematically determined set, which includes the designs that are clearly better than others without using a user preference to determine the proper tradeoff of the design objectives. This constantly updated front of Pareto designs preserves designs depicting a diversity of relative strengths in the objectives.

Outside of Pareto-optimal designs, the system further

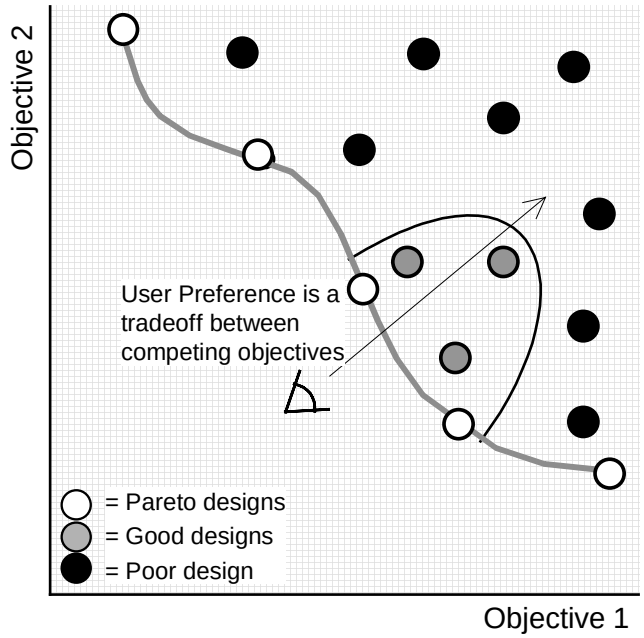


Figure 2: Design Selection separates a population into three sets: Pareto Designs, Good Designs, and Poor Designs

divides solutions into Good and Poor designs. Some designs, while not Pareto-optimal, might better meet user preference or better serve as a basis for modification than some of the outlying Pareto-optimal alternatives. These preferred but non-Pareto designs comprise the Good population visualized as a set of solutions located within a given radius of the intersection of current user preference and the Pareto set (Figure 2). In general, the Good designs are the best designs (top 50% in the current implementation) of the non-Pareto designs that meet the current approximation of the user's utility function. If the user's preference should change then the location of the Good population also changes. By preserving this set of Good designs, the design process concentrates on the current desired tradeoff in objectives. By preserving the Pareto designs, it maintains a diversity of design ideas so that the process can improve any one of these ideas to adapt to the changes that might occur to the user's preference. Experiments that reveal how this division of designs makes A-Design adaptive can be found in Campbell et al. (2000a).

The final task of the search process is to *guide* the iteration towards better solutions. Based on information gained from past iterations, the process is steered towards productive areas of the search space. This part of the process is most similar to stochastic optimization techniques such as simulated annealing (Kirkpatrick et al., 1983), and genetic algorithms (e.g., Goldberg, 1989; Queipo et al., 1994; Koza et al., 1996) where statistical data from numerous design states aids the search for better design states in future iterations. In A-Design, the *guidance* is handled by a manager-agent that observes the process and makes informed decisions about how to best improve the process in future iterations. The *guidance* method used by the manager-agent is the focus of this paper and is discussed in detail in Section 3.

In summarizing the A-Design system, Figure 3 is provided to show the details of how the general search process cycle in Figure 1 is accomplished. At the top of the figure, the process is initiated with some seed specified by the user that encapsulates the description of the design problem. Various design alternatives are generated by the Maker-agents that work directly with these specifications to produce a population of possible solutions. This *generation* phase happens in two steps: first the Configuration-agents (C-agents) choose how components will be connected, and second, the Instantiation-agents (I-agents) substitute in real components into the configurations. Each design alternative results from the contributions of several agents interacting together.

Next, the population of solutions passes to the *evaluation* process, which can be viewed as the engineering analysis that happens as a subset of the larger engineering design task. Based on the evaluations, the designs are sorted into the three populations of Pareto, Good, and Poor. The Good and Pareto designs are saved for modification and passed on to the Modification-agents. These agents choose designs from the preserved set of best solutions and attempt to refine them based on the quality of their evaluation. Agents in this category add,

remove, or otherwise alter elements of design alternatives to create new states that are returned to the process by way of the Maker-agents.

After the modification of designs, the process repeats, evolving the better solutions to the design problem. As the process unfolds, design states cycle through the exchange between Maker-agents and Modification-agents until the system converges or resource and time constraints require the acceptance of the current best design. Throughout the process, the manager-agent observes the design activity and provides *guidance* to improve designs in the future. In addition to guiding the process, the manager-agent also communicates with the user to determine how well the search process is satisfying the design requirement. Based on the dialog with the user, the manager-agent may make further changes in the process.

3 TODO/TABOO LEARNING

The innovations discussed in this paper build upon the A-Design algorithm described above. This learning mechanism, hereafter referred to as TODO/TABOO learning, enhances the *guidance* aspect of the process. The iterative operation of A-Design can be viewed as a simplification of the iteration occurring in human design. However, in human problem solving, learning plays a key role in what subsequent decisions are made. The learning that results from design activity can lead to an efficient search downstream in the process by building intuition about successful and unsuccessful regions of the search space. The philosophy behind this learning technique is to glean information from the extensive data obtained from past iterations to both narrow search to particular areas of the space of design solutions or broaden search to find new fruitful areas of the search space.

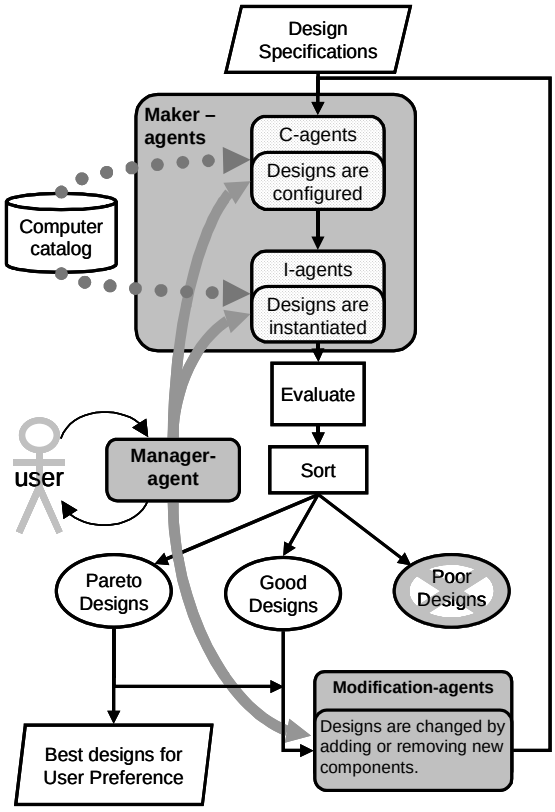


Figure 3: Detailed Flowchart of A-Design iterative process.

TODO/TABOO learning is built upon several distinct research endeavors. Perhaps the most similar research is that on the Tabu search technique (Glover, 1989; Moscato, 1993), which is a complete stochastic search process based on a similar principle of storing past visited design states. This approach to optimization stores past solutions on a Tabu list as a memory to prevent the process from backtracking, thus forcing the process into new areas of the search space. The technique developed here, however, also is based on several different machine learning research areas (see overview in Mitchell, 1997). For example, learning in electromechanical design has been shown in the LearnIT system (Stahovich, 1999) where a computational system is able to learn how devices behave and devise rules to perform certain design tasks. The detection of good and bad design trends in TODO/TABOO guidance is similar to learning by analyzing differences, as explored by Winston (1982), which classifies instances into distinct categories by observing differences and commonalities in test data. The grouping of good and bad design characteristics is also inspired by the SOAR system (Laird et al., 1986) for "chunking" similar entities together to produce a desired effect. The manner in which this learned data affects the decision-making process of the agents is similar to other approaches that have combined reinforcement learning with multi-agent systems (see Tan, 1993; Sandholm and Crites, 1995). Finally, the stochastic adjustment of design decisions as discussed below in Section 3.2.1 is similar to techniques such as proportional selection in genetic algorithms (see Bäck and Hoffmeister, 1991) and move set probabilities in simulated annealing (Hustin and Sangiovanni-Vincentelli, 1989). These approaches can allow stochastic optimization to be more directed and efficient in the search for successful design alternatives.

The learning process is initiated after a set of designs has been created by the Configuration-agents, instantiated by the Instantiation-agents, evaluated, and sorted in Pareto, Good, and Poor sets. The algorithm is a two-step process. First, common sets of agents and common configurations of components, which are both referred to as "trends", are detected in the data of previous iterations. These trends are found in both good and bad designs. The TODO list is constructed by examining the best designs to find trends of positive design activity. Conversely, the worst designs are compared to find elements of the TABOO list representing poor design trends. This detection of trends is described in depth in Section 3.1. Second, the trends are provided as feedback to the agents in the process. The manager-, Maker- and Modification-agents incorporate their individual preferences with these trends to make informed decisions based on past design experiences. This method of feedback is similar to reinforcement learning, and is discussed in Section 3.2.

3.1 FINDING TRENDS

In the current implementation of A-Design, three types of trends are currently found amongst the previous created designs. These are 1) design fragments consisting of small functional blocks of components and their connections, for example a "rack-and-pinion", 2) sets of agents, or teams of agents, that have collaborated on past design instances, and 3) groups of components that commonly occur to instantiate a design configuration. These trends are found by intersecting past created designs. The intersection operations for the latter two types of trends are found through simple mathematical set intersection operations, however, the first type of trend (design fragments) are found through a specialized graph intersection operation.

As shown in Figure 2, designs are sorted into three populations in order to handle design problems with more than one objective. However in determining trends, an approximated linear weighted sum of objectives is developed so that the designs can be sorted from best to worst by collapsing the various objectives to a single metric as seen for two objectives in the example of Figure 4a. In this figure, the eye represents a user preference approximated by a linear weighted sum of the two orthogonal objectives. The designs, A through J, evaluated by this weighted sum, collapse to a point on the line so that they can be sorted from best to worst. From this sorting, a number of the best designs are selected for constructing the TODO list and a number of the worst designs are selected to determine the TABOO list (see Figure 4b).

Next the designs are exhaustively compared to find commonalities in the selected designs. In Figure 4c, the top five designs separated for TODO comparison are examined to find common intersections in the designs. First, the process looks for an intersection in all five designs. Sometimes intersecting all of the designs does not yield any commonalities. Therefore, the process also checks all possible combinations within four out of the five designs to find trends. This intersection continues for all possible combinations of designs down to comparing only two members at a time. The intersections or trends found from this comparison are labeled with the number of designs in the intersection. This quantity is used in the reinforcement learning of the agents to judge the significance of the trends.

Finding the intersection of sets (such as common design teams of agents) is easier than finding the intersection of graphs (such as design fragments). While set intersection routines are rather quick and fundamental to computation, the intersection of graphs can be an intricate procedure. To find a design fragment, the sets of components are first intersected to find what the common components are. This set is used to reduce the amount of comparisons needed to find a common design fragment between two designs. However, many connections and repeated components make finding common design fragments difficult.

The resulting intersections of both sets and graphs are used in the following iteration for reference by the agents. These intersections only last one iteration, and new intersections are found for the subsequent designs generated by the agents. In the A-Design implementation, the intersection procedure is not particularly time-consuming, but because of the number of intersections (as in Figure 4c), the size of the TODO/TABOO lists can be limited by resource constraints. Increasing the membership of the TODO and TABOO sets drastically increases the number of required intersections. Also, if the designs that are created contain many components or a large number of repeated components, then the individual intersection times would greatly effect the time required to find the TODO and TABOO trends. Currently, the implementation relies only on the few constants specified by the user to operate effectively. These include setting the maximum number of designs to compare and the maximum number of trends to return. Besides these constants the detection of trends and how the information is used occurs completely computationally, and has produced promising results as seen in Section 4.

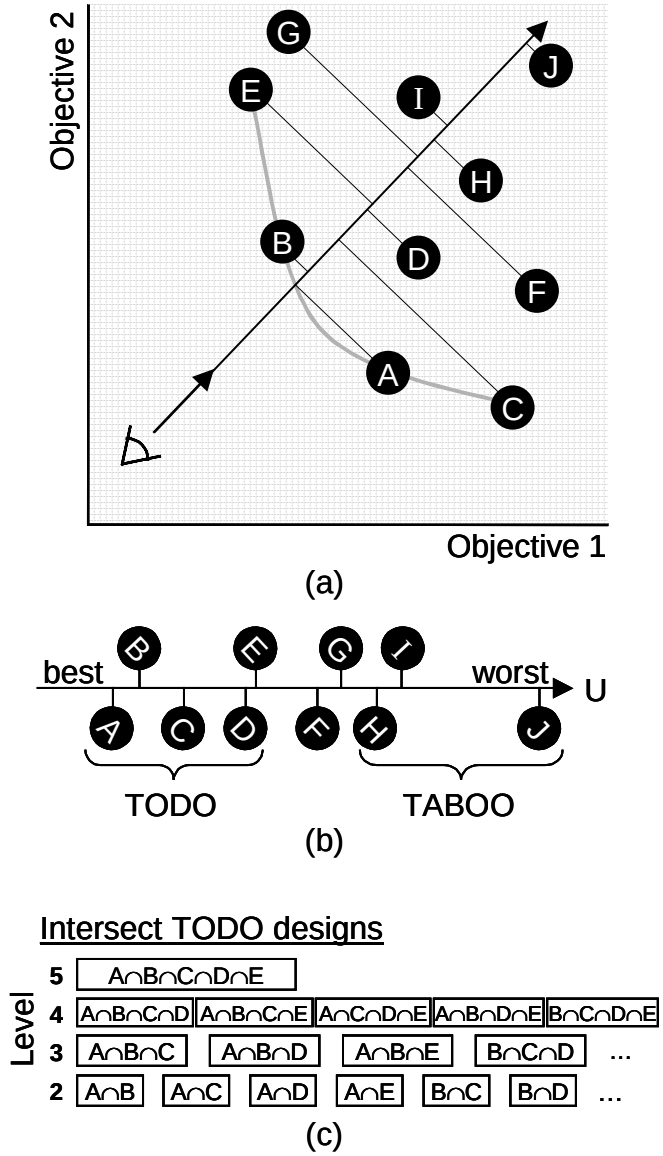


Figure 4: An example of detecting TODO and TABOO trends from a set of 10 designs.

3.2 LEARNING AND DECISION-MAKING IN THE AGENTS

After intersecting designs and separating them into the TODO and TABOO lists, the next iteration ensues. The design process in each iteration is an unstructured interaction between the populations of agents and the populations of designs. Initially, the process of choosing which agents to invoke and which components the agents choose to add or remove from a design is quite random. However, with the information provided in the trends, the process becomes more directed, or to be more exact, stochastically guided. All of the agents in A-Design follow a similar model for deciding what actions to take in effecting designs. The strategy involves performing a local search over all possible actions that the agent can carry out. Each action is tested by the agent and a choice is made based on the action that maximizes the agent's evaluation function.

3.2.1 Using Trends to Invoke Agents, and Agent-Teams

Throughout the modification and creation of designs, the basic operation of the process is to invoke Maker- and Modification-agents until all design tasks for a given iteration have been completed. Before every agent call, the manager-agent is queried by the process to generate a list of how to proportion the probabilities for selecting an agent. The manager-agent rates each agent based on prior statistics and prior collaborations with other agents as determined by the TODO and TABOO trends. While these manager-agent decisions *guide* the process, they do not completely override the randomness of the design activity thus producing a stochastic search for new design states. In addition to the TODO and TABOO trends, the manager-agent also keeps track of the designs each agent produces. This data is then used to adjust the probabilities of invoking an agent in future iterations

Figure 5 illustrates an example of how this stochastically-guided proportional agent selection is accomplished. In a particular design state, the previous agents z, w, y, w have been invoked (as seen in Figure 5, element 1). The manager-agent then sets up probabilities for the next agent to call based on information on the agent statistics from previous iterations. For example, as seen in Figure 5-2, the statistics for agent w are that it has contributed to four Pareto designs, no Good designs, and two Poor designs.

The agent statistics and the TODO/TABOO list information are filtered through an evaluation function. This function is a weighted sum of five terms that determines the probabilities of the next agent call:

$$U = \mathbf{w_{M1}}^{x} pareto\_num + \mathbf{w_{M2}}^{x} good\_num - \mathbf{w_{M3}}^{x} poor\_num + \mathbf{w_{M4}}^{x} TODO\_team - \mathbf{w_{M5}}^{x} TABOO\_team. \quad (1)$$

The number of past Pareto designs the agent contributed to (pareto_num), the number of past Good designs (good_num), and the number of past poor designs (poor_num) weigh in to the manager-agent's evaluation. Design teams from the TODO and TABOO lists (Figure 5-3) also influence how each agent is considered. If the addition of a particular agent completes an agent team from the TODO/TABOO list of trends, then the degree to which that trend is significant also weighs into the manager-agent's evaluation. As an example, agent x contributes to a TODO team (zx = 3) and a TABOO team (xy = 2) as seen in Figure 5-3. Given that agents z, w, y, and w have already been called, the effect of choosing agent x next would fulfill these two agent teams. As a result, the evaluation function for invoking x next includes *TODO_team* = 3, and *TABOO_team* = 2.

The manager-agent considers all possible agents and gives each a score from the evaluation function (Figure 5-4). As a result of these values, the manager-agent divides the probabilities as shown in Figure 5-5. The process then picks a random number to determine which agent to invoke based on the manager-agent's division. The weights in Equation 1 are prescribed to be $\mathbf{w_{M1}} = \mathbf{w_{M4}} = \mathbf{w_{M5}} = 2$, $\mathbf{w_{M2}} = \mathbf{w_{M3}} = 1$. These values were chosen to produce a balanced yet effective division of agent probabilities. It is possible that these coefficients could be adjusted for different applications or agents. The means of optimally or automatically weighing the factors of this stochastic guidance is subject to future experimentation and development.

While this strategy is similar to the dynamic selection mechanism of proportional selection in genetic algorithms, the approach is augmented by the fact that the manager-agent updates the selection probabilities for each agent call. The case-by-case updating of probabilities allows designs to be tailored

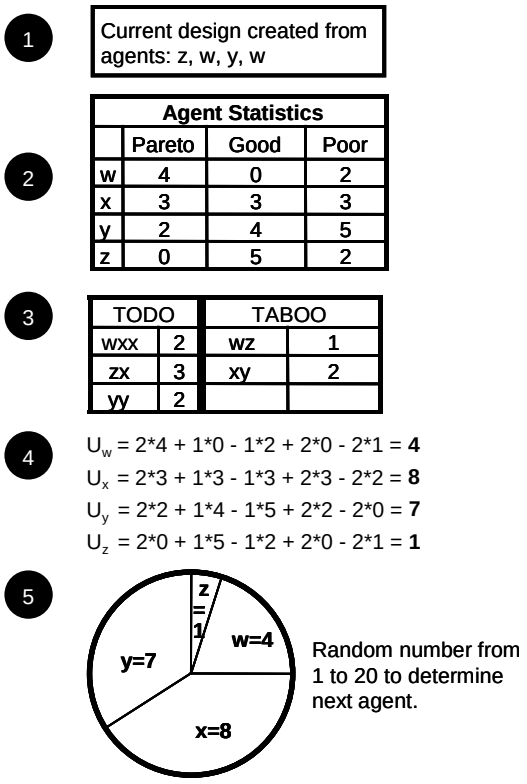


Figure 5: Manager-agent proportions probabilities based on evaluation function.

to various strengths and weaknesses, and encourages certain agents to collaborate on specific or diverse designs. The context of continually recommending agents based on the state of a design is a unique *guidance* mechanism of A-Design.

3.2.2 Learning from Past Trends to Build Future Designs

In addition to the *guidance* provided by the manager-agent, the Maker- and Modification-agents are augmented by a learning strategy to hone their understanding of how to best add or remove components from a design. Similar to the manager-agent functionality, these agents have unique evaluation functions that combine terms from individual agent preferences and TODO/TABOO learning.

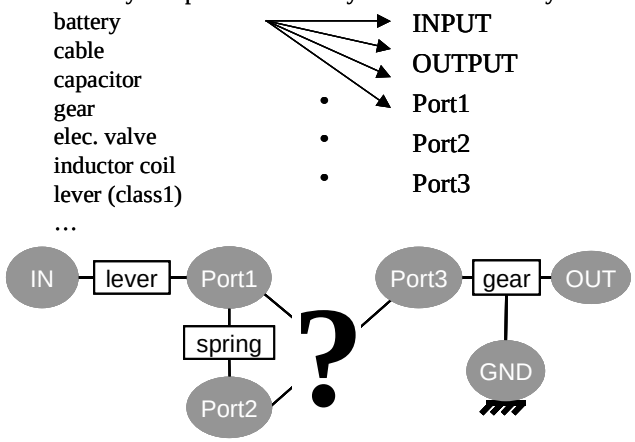
To accomplish this, the utility functions, or evaluation functions, of the configuration-agents include five terms which quantify 1) what component is chosen (V_{COMP}), 2) how a new component is connected ($V_{CONNECT}$), 3) the components it connects to (V_{OTHER}), 4) the adherence to TODO trends (V_{TODO}), and 5) the avoidance of TABOO trends (V_{TABOO}). The first three quantities are based on an agent's particular preferences. Different agents can have different preferences for several aspects of the design building process. While some agents prefer to connect components in parallel, others prefer to connect components in series. Some prefer using mechanical components to electrical components, and others have the opposite preference. As an example, a particular configuration-agent known as *C-agent-electrical-parallel-input-existing* has a preference for *electrical* components, connected in *parallel* with other components, working from *input* connections towards output connections (as opposed to vice-versa), with the remaining ports of a component connected to other *existing* connections (as opposed to connecting to *ground* or left *dangling* for future component connections). The last two terms, V_{TODO} and V_{TABOO}, represent how experience gained from the TODO and TABOO trends influence an agent's decisions. If the addition of a particular component coincides with a trend found in past iterations, then the agent's evaluation of that component is rewarded (in the case of TODO) or penalized (in the case of TABOO) accordingly.

In Figure 6, pseudo-code is presented to clarify this agent's learning. By observing a partially completed design state, and the TODO and TABOO lists, the agent proceeds to exhaustively test each component with each connection point in the configuration. With each test, the agent evaluates the five quantities that weigh into the decision for best component choice. In the calculation of these values, the agent tallies its unique preferences with decisions that complete the design states. As can be seen in points b, c, d of Figure 6, extra points are awarded for fulfilling goal states. This directs the agent to perform an action that leads to the completion of the design as opposed to simply adding components to only meet a personal preference. The result of testing each action is a scalar value, **V**, which is a weighted sum of the five values specified earlier. After the testing the agent chooses the component and connection that leads to the highest value of **V** and updates the partial configuration.

The weights ($\mathbf{w_{C1}}$, $\mathbf{w_{C2}}$, $\mathbf{w_{C3}}$, $\mathbf{w_{C4}}$, $\mathbf{w_{C5}}$) in the agents' decision-making process represent a nontrivial judgment between the crucial factors of the design decisions. Currently, these weights are predetermined in the A-Design

Input = incomplete design state, TODO, TABOO

1. Check every component with every connection in the system.



a. Initialize values to zero
(V_{COMP}= $V_{CONNECT}$ = V_{OTHER} = V_{TODO} = V_{TABOO} = 0).
b. What is the value of each component? (V_{COMP})
V_{COMP}:= V_{COMP} + 1 for meeting domain preference (electrical vs. rotational vs. translational vs. hydraulic)
V_{COMP}:= V_{COMP} + 1 for fulfilling goal state
c. What connection to connect to? What is the value of each port? ($V_{CONNECT}$)
$V_{CONNECT}$:= $V_{CONNECT}$ + 1 for meeting series or parallel preference
$V_{CONNECT}$:= $V_{CONNECT}$ + 1 for meeting direction preference (source vs. sink)
$V_{CONNECT}$:= $V_{CONNECT}$ + 1 for meeting domain preference (electrical vs. …)
$V_{CONNECT}$:= $V_{CONNECT}$ + 1 for containing a goal state
d. What to do with remaining ports? What is value of connections (V_{OTHER})?
V_{OTHER}:= V_{OTHER} + 1 for proper connection type (existing vs. ground vs. dangling)
V_{OTHER}:= V_{OTHER} + 1 for meeting series or parallel preference
V_{OTHER}:= V_{OTHER} + 1 for fulfilling goal state
e. Does this connection match a TODO or TABOO trend?
V_{TODO} := sum of the TODO trend values corresponding with resulting design (if none then = 0).
V_{TABOO} := sum of the TABOO trend values corresponding with resulting design (if none then = 0).
f. Calculate evaluation function
$\boldsymbol{V} = \mathbf{w_1} \times V_{COMP} + \mathbf{w_2} \times V_{CONNECT} + \mathbf{w_3} \times V_{OTHER} + \mathbf{w_4} \times V_{TODO} - \mathbf{w_5} \times V_{TABOO}$
2. Choose new component and connection producing the highest value of $\boldsymbol{V}$

Output = design state updated with new component and connections.

Figure 6: Pseudo-code for Configuration-agent.

implementation for each agent. As opposed to the manager-agent weights ($\mathbf{w_{Mi}}$), various agents of each agent-type are created with different weightings. For example, for the agent-type *C-agent-electrical-parallel-input-existing*, various agent instances can be implemented to emphasize different terms of the evaluation function. One agent might prefer the component choice (high value for $\mathbf{w_{C1}}$) more than how it is connected in a design (lower values for $\mathbf{w_{C2}}$, $\mathbf{w_{C3}}$). Also, some agents can strongly consider learning influences (higher $\mathbf{w_{C4}}$, $\mathbf{w_{C5}}$), while others ignore learning ($\mathbf{w_{C4}} = 0$, $\mathbf{w_{C5}} = 0$). With this approach, a variety of agents can be constructed to produce more variety in the design process. Future work with this utility agent model will investigate alternative ways of choosing weights either automatically or through experimentation.

4 RESULTS AND DISCUSSION

In this section, the proposed TODO/TABOO learning method is evaluated by comparing A-Design simulations with and without the learning mechanism. In this design problem, a multitude of objectives and an unstructured representation allow for a limitless space of possible solutions. Because of these issues and the lack of a known global optimum, developing measures to compare different simulations is not simple. In these experiments, comparison is performed by determining the quality of the final designs produced after an equal number of iterations, and by observing the designs found throughout the iterations. The first section below examines the quality of the final designs produced with and without learning, while Section 4.2 examines statistics gathered over the iterations.

4.1 DESIGN PROBLEM RESULTS

This section present results from the design of a weighing machine (scale) that has a downward force as an input and a dial as an output. In order for A-Design to synthesize design solutions, three things must be specified: a set of objectives, a specification of the inputs and outputs, and a catalog of components. Figure 7 provides a visualization of these specifications for describing the weighing machine design problem. The four objectives chosen to guide the system to successful design states are minimize cost, minimize mass, minimize dial error and minimize input displacement. The components inputted to A-Design comprise a computational catalog of just over 300 components drawn from Allied Electronics, Nordex Inc. and Mc-Master Carr Supply catalogs.

In this section, designs resulting from running the process for 30 iterations with a population of 100 designs per iteration are compared. Designs A and B in Figure 8 do not include TODO/TABOO learning and are to be compared to Design C and D that have the learning mechanism present[2]. One interesting aspect of these designs is the diversity in their configurations and the different degrees to which they meet the user objectives. Figure 9 shows how each of the four designs are evaluated on the four objectives. A more in depth study of the diversity of results from A-Design and how they meet different user preferences can be found in Campbell et al. (1999).

In Figure 9, the four objective values are plotted. In these plots, designs with lower values indicate a lower minimum was reached and, therefore, the resulting design has a higher quality. It can be seen that the two designs with learning have lower and therefore superior values for nearly all objectives compared to the designs without learning. Specifically, Design C is similar to Design A, but shows an improved reduction in cost by using a belt and pulley as opposed to a rack and pinion. Also the challenge in reducing the displacement in the input leads to some interesting differences. The expensive approach of using two different sized hydraulic cylinders, as in Design B, is avoided in Designs C and D by favoring more components. The learning affords the system the ability to create more complex designs to overcome the deficiencies of simple configurations. In addition to the weighing machine design problem shown here, other promising results have been performed in applying this technique to the design of micro-electromechanical accelerometers.

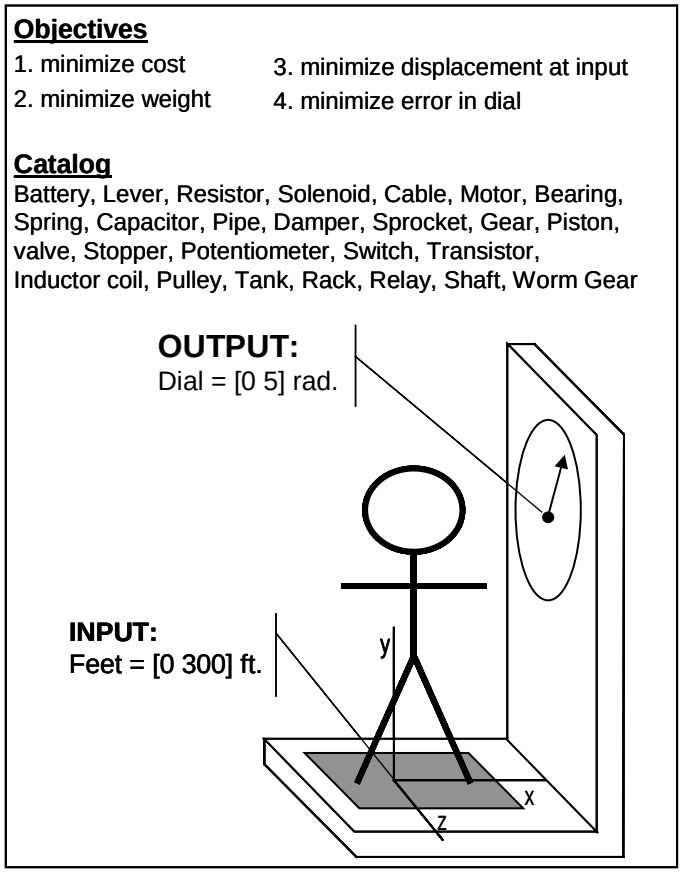


Figure 7: Description of weighing machine design problem as posed to A-Design.

[2]Figure 8 is the authors' renditions of the configurations. The system outputs a text file, which gives the specifics of which components are used and how they are connected.

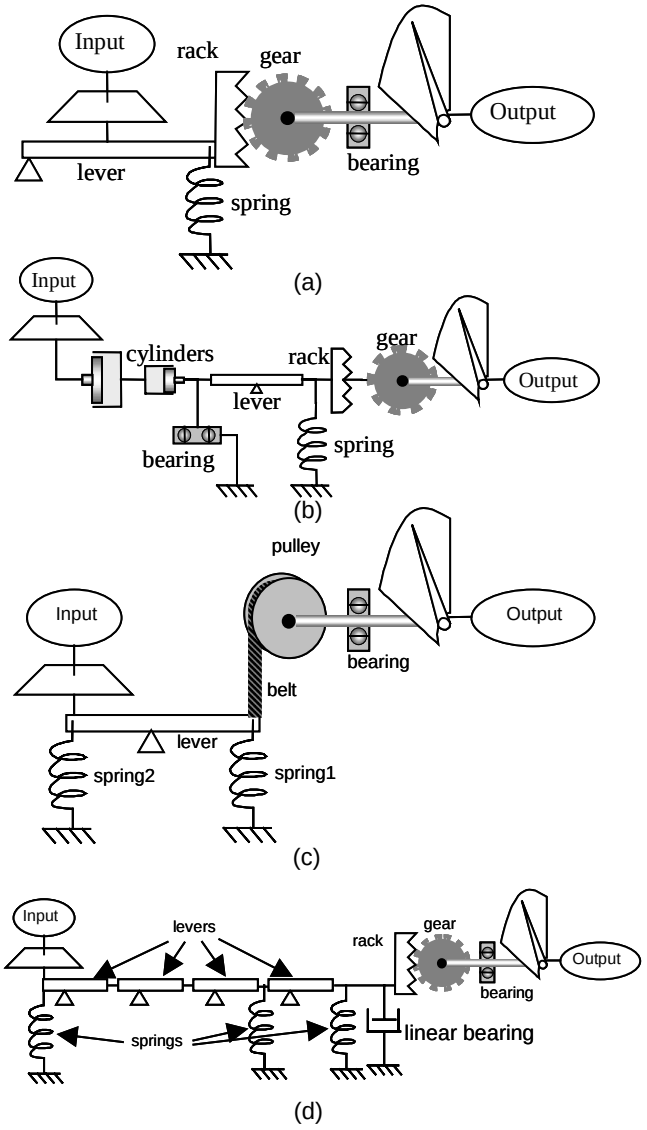
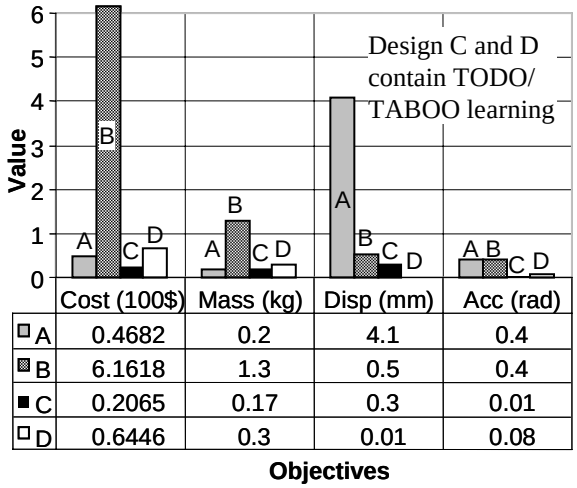


Figure 8: Four final designs created by A-Design.



	Cost (100$)	Mass (kg)	Disp (mm)	Acc (rad)
A	0.4682	0.2	4.1	0.4
B	6.1618	1.3	0.5	0.4
C	0.2065	0.17	0.3	0.01
D	0.6446	0.3	0.01	0.08

Figure 9:The four objectives plotted for the four designs.

4.2 STATISTICAL RESULTS

In this section, statistics are gathered on the weighing machine test problem to better compare how design quality is improved with TODO/TABOO learning. Figure 10 shows curves whose points represent an averaging of 20 separate runs of the A-Design system. Each run contains 40 iterations with a population of 100 designs. The TODO and TABOO lists are limited to 12 trends apiece. When new trends are found, the oldest trends are removed from the set in order to keep the learning examples current with the newly generated candidates.

In comparing the three curves with no learning (NONE), with just TODO learning (TODO), and with just TABOO learning (TABOO), the effect of the TABOO trends leads to a much improved design state over the other two approaches. Interestingly enough, the TODO learning, although the quickest process in the first 15 iterations, begins to level off and experiences a slower rate of improvement than the TABOO case. However, the end effect of the TODO learning still offers a significant improvement over the run with no learning.

Figure 10 also compares some hybrid learning techniques that involve combinations of both TODO and TABOO learning. BOTH contains a fixed TODO and TABOO list size, and this list size is maintained for all iterations. The KICK and UPDOWN tests contain dynamic TODO and TABOO list sizes. In KICK, a constant TODO list size is maintained, and the TABOO list is zero at all iterations except at every tenth iteration where it contains up to 15 members. The motivation for this strategy is to "kick" the process intermittently so as to avoid getting trapped in local optima. The UPDOWN procedure attempts to take advantage of the quick learning shown in the TODO and TABOO comparison. In the first 15 iterations, TODO has a population size of 15 members then diminishes to zero, while TABOO learning is increased from 0 to 15 members at the fifteenth iteration.

Upon examining the results in Figure 10, the system with simply TABOO learning performs better than any of the hybrid approaches. Interestingly enough, the hybrid approaches all fall somewhere between TODO and TABOO. The unadulterated TABOO learning appears to be the most successful feedback for these problems. This type of avoidance feedback is also the basis for Tabu search (Glover, 1989), which operates under a similar principle. The advantage in remembering bad design trends is twofold. First, it provides a means of tracking common mistakes that occur during design construction so that these can be avoided in future moves to make the process more efficient. Second, the avoidance of past moves can produce atypical designs or innovative search that allows the process to escape local optima.

In addition to the comparison shown in Figure 10, a plot shown in Figure 11 compares the best final design found from the 20 runs in each learning technique. In this plot, the best design without learning appears to be deficient in nearly all

comparisons. The three best designs from BOTH, KICK and TABOO are nearly equivalent, while the best solutions for the TODO and UPDOWN algorithms actually appear to find better values for the "minimize input displacement" objective. The reason for these better values is not well understood. It is possible that the early TODO learning in these two approaches provides insight into the most challenging design issues. TODO learning exploits the benefits of successful past solutions. This might not lead to design diversity but it does allow A-Design to concentrate on improving known successes.

Future experiments with TODO and TABOO learning within various design domains might clarify the advantages of each learning technique. However, it is safe to say that the learning mechanism incorporated in the A-Design process better guides the process to successful designs, and makes strides towards incorporating the kind of human learning that is used in conceptual design.

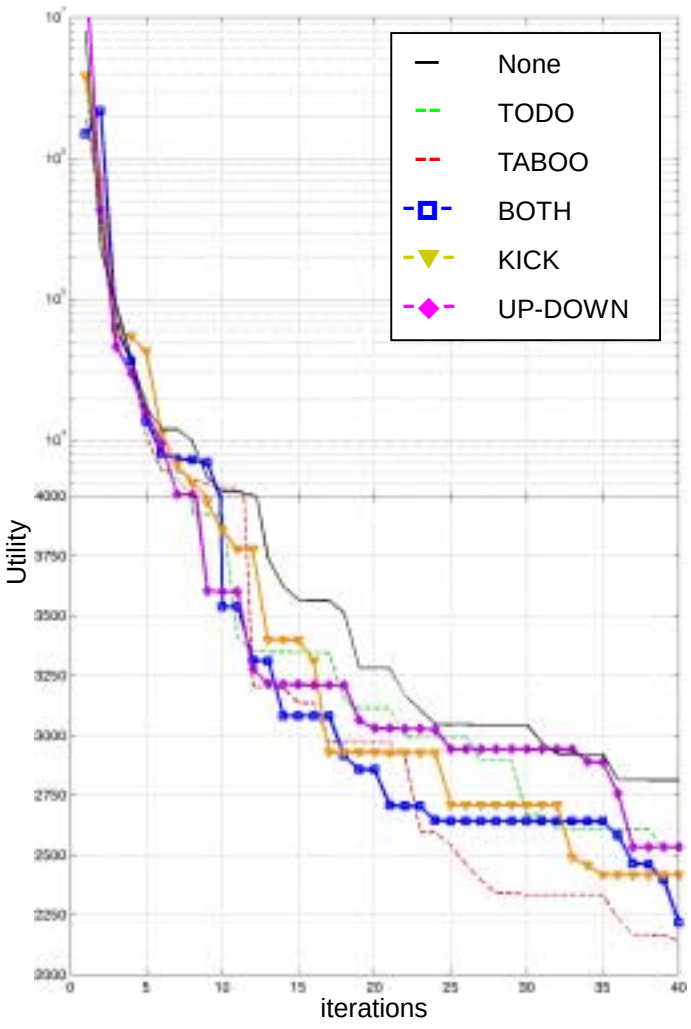


Figure 10: A comparison of simulations with different combinations of TODO and TABOO learning.

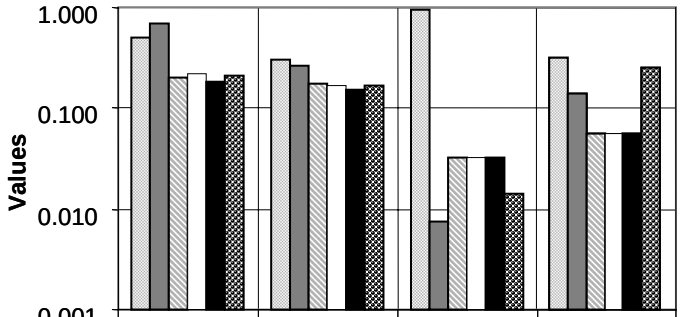
5 DISCUSSION

The A-Design methodology has been used to automate configuration design by innovating the *representation*, *generation*, *evaluation*, and *guidance* of traditional computational search specifically for electromechanical design problems. The ability of the process to overcome restrictions on fixed configurations and adapt to changing user preferences has been seen before in Campbell et al. (1999, 2000a, 2000b). This paper presents a new method for *guiding* automated design processes. The method known as TODO/TABOO learning keeps track of previous design commonalities. In this manner, the process can learn from its past mistakes and successes.

In Section 4, results from a test design problem were observed with and without the presence of TODO/TABOO learning. In the weighing machine results, learning led to better designs by finding ways around barriers in the search space. The TODO and TABOO trends both drive the search to focus on favorable areas of the search space, and avoid the unfavorable areas. This appears to make the process more efficient as well as produce better design quality.

The results in Section 4.2 reveal the unique benefits the TODO and TABOO trends have on the computational search process. Recalling TABOO design states can both make the process more efficient by avoiding common design errors, and force the search into new and possibly fruitful areas of the search space. Further, the experiments showed that although, TABOO learning was more effective than TODO learning, difficult design issues (as in minimizing input displacement of the weighing machine) benefited from the recollection of past design successes.

Learned instances are a large part of our design process, and this method of adding knowledge similarly provides a



	Cost (100$)	Mass (kg)	Disp (mm)	Acc (deg)
NONE	0.512	0.303	0.965	0.321
TODO	0.692	0.263	0.008	0.138
TABOO	0.202	0.173	0.033	0.056
BOTH	0.223	0.166	0.033	0.056
KICK	0.181	0.153	0.033	0.056
UPDOWN	0.207	0.169	0.014	0.251

Objectives

Figure 11: Table comparing the objective values of the best designs in each of the hybrid learning methods.

computational system with the head start that human designers experience. Unlike many expert system approaches to design automation, this design experience does not restrict solutions to conform to previous instances. Rather, the design knowledge can guide innovation to build on past experiences to create entirely new concepts.

TODO and TABOO learning is a fertile area of future research. Currently, performance has been improved by recalling both agent teams and design fragment trends. In the future, other design characteristics such as dynamic behaviors, or geometric similarities could greatly increase the amount of design experience that can be gained from this technique. While, the learning method has shown to produce better quality designs, the extent to which TODO and TABOO lists effect design activity and are dependent on the characteristics of the design space are not yet known. Further testing of this technique along with application to new design problems will increase our understanding of this approach to experiential learning in computational design synthesis.

ACKNOWLEDGMENTS

The research effort was partially sponsored by the Defense Advanced Research Projects Agency (DARPA) and Rome Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-96-2-0304. 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 DARPA, Rome Laboratory, or the U.S. Government.

REFERENCES

Bäck, T, and Hoffmeister, F., 1991, “Extended Selection Mechanisms in Genetic Algorithms”, *Proceedings of the Fourth International Conference on Genetic Algorithms*, eds. Belew, R. and Booker, L., Morgan Kauffman Publishers, San Mateo, CA, pp. 92-99.

Balachandran, M., J. S. Gero, 1984, “A Comparison of Three Methods for Generating the Pareto Optimal Set”, *Engineering Optimization*, Vol. 7, pp. 319-336.

Bracewell, R. H., and Sharpe, J. E. E., 1996, “Functional Description Used in Computer Support for Qualitative Scheme Generation- ‘Schemebuilder’”, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, Vol. 10, pp. 333-345.

Campbell, M., J. Cagan and K. Kotovsky, 1999, "A-Design: An Agent-Based Approach to Conceptual Design in a Dynamic Environment," *Research in Engineering Design*, Vol. 11, No. 3.

Campbell, M., J. Cagan and K. Kotovsky, 2000a, "Agent-based Synthesis of Electro-Mechanical Design Configurations," *Journal of Mechanical Design*, Vol. 122, No. 1, pp. 61-69.

Campbell, M., 2000b, The A-Design Invention Machine: A Means of Automating and Investigating Conceptual Design, *Ph.D. Dissertation*, Carnegie Mellon University.

Chakrabarti, A., Bligh, T. P., 1996, "An Approach to Functional Synthesis of Mechanical Design concepts: Theory, Applications, and Merging Research Issues," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, Vol. 10, pp. 313-331.

Fonseca, C. M., P. J. Fleming, 1995, “An Overview of Evolutionary Algorithms in Multiobjective Optimization”, *Evolutionary Computation*, Vol. 3, pp. 1-16.

Glover, F., 1989, "Tabu Search-Part 1," *ORSA Journal on Computing*, Vol. 1, No. 3, pp. 190-206.

Goldberg, D. E., 1989, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison Wesley, Reading, MA.

Holland, J. H., 1992, *Adaptation in Natural and Artificial Systems*, The MIT Press, Cambridge, MA, 2nd edition.

Hustin, S., and Sangiovanni-Vincentelli, A., 1987, “TIM, a New Standard Cell Placement Program Based on the Simulated Annealing Algorithm,” *IEEE Physical Design Workshop on Placement and Floorplanning*, Hilton Head, SC.

Keeney, R. L., and Raiffa, H., 1976. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*, John Wiley & Sons, New York.

Kirkpatrick, S., C. D. Gelatt Jr., and M. P. Vecchi, 1983, "Optimization by Simulated Annealing," *Science*, Vol. 220, pp. 671-679.

Koza, J. R., F. H. Bennett III, D. Andre, 1996, "Automated Design of Both the Topology and Sizing of Analog Electrical Circuits using Genetic Programming," *Artificial Intelligence in Design*, eds. J. S. Gero and F. Sudweeks, pp. 151-170.

Laird, J. E., Newell, A., and Rosenbloom, P. S., 1986, "Soar: An Architecture for General Intelligence," *Technical Report* CMU-CS-86-171, Carnegie Mellon Univ., Pgh., PA.

Lander, S. E., 1997, "Issues in Multiagent Design Systems", *IEEE Expert*, Vol.12, pp. 18-26.

Langton, C. G. (ed.), 1988, *Artificial Life: SFI Studies in the Sciences of Complexity*, Addison Wesley, Reading, MA.

Mitchell, T. M., 1997, *Machine Learning*, McGraw-Hill, New York.

Moscato, P., 1993, “An Introduction to Population Approaches for Optimization and Hierarchical Objective Functions: A Discussion on the Role of Tabu Search,” *Annals of Operations Research*, Vol. 41, pp. 85-121.

Navinchandra, D., K. P. Sycara, and S. Narasimhan, 1991, "A Transformational Approach to Case-Based Synthesis,” *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, Vol. 5, pp. 31-45.

Palmer, R. S., and Shapiro, V., 1993, “Chain Models of Physical Behavior for Engineering Analysis and Design,” *Research in Engineering Design*, Vol. 5, pp. 161-184.

Queipo, N., R. Devarakonda, and J. A. C. Humphrey, 1994, "Genetic Algorithms for Thermosciences Research: Application to the Optimized Cooling of Electronic Components," *International Journal Heat and Mass Transfer*, Vol. 37, pp. 893-908.

Sandholm, T.W.; Crites, R.H., 1995, "On Multiagent Q-learning in a Semi-Competitive Domain", *Adaptation and Learning in Multi-Agent Systems. IJCAI 95 Workshop*. Proceedings, eds. G. Weiss, S. Sen., Vol. 6, 191-205

Schmidt, L.C., and Cagan, J., 1998, "Optimal Configuration Design: An Integrated Approach Using Grammars", *Journal of Mechanical Design*, Vol. 120, pp. 2-9.

Stahovich, T.F., 1999, "LearnIt: A system that can learn and reuse design strategies" *Proceedings ASME Design Engineering Technical Conferences, DETC99/DTM-8779*, September 12-15, Las Vegas, NV.

Stone, R. and Wood, K., 1999, "Development of a Function Basis for Design", *Proceedings ASME Design Engineering Technical Conferences, DETC99/DTM-8765*, September 12-15, Las Vegas, NV.

Talukdar, S., L. Baerentzen, A. Gove, and P. de Souza, 1996, "Asynchronous Teams: Organizations for Algorithmic Computation", *EDRC Tech-Report* 18-56-96, EDRC Carnegie Mellon University, Pittsburgh, PA.

Tan, M., 1993, "Multi-agent Reinforcement Learning: Independent vs. Cooperative Agents", *Tenth International Conference on Machine Learning*, Amherst, MA, pp. 330-337.

Thurston, D. L., 1991, "A Formal Method for Subjective Design Evaluation with Multiple Attributes" *Research in Engineering Design*, Vol. 3, pp. 105-122.

Ulrich, K., and Seering, W., 1989, "Synthesis of Schematic Descriptions in Mechanical Design," *Research in Engineering Design*, Vol. 1, pp. 3-18.

Ulrich, K., T., 1989, Computation and Pre-Parametric Design, *Ph.D. Dissertation*, Massachusetts Institute of Technology, Tech. Report AI-TR-1043.

Welch, R. V., and Dixon, J., 1994, "Guiding Conceptual Design Through Behavioral Reasoning," Res*earch in Engineering Design*, Vol. 6 pp. 169-188.

Winston, P. H., 1982, "Learning New Principles from Precedents and Exercises," *Artificial Intelligence*, vol.19, no.3, p. 321-350.