Original paper

Research in Engineering Design 14 (2003) 12–24
DOI 10.1007/s00163-002-0025-x

The A-Design approach to managing automated design synthesis

Matthew I. Campbell, Jonathan Cagan, Kenneth Kotovsky

Abstract This paper presents an approach to managing the complexities of design synthesis techniques. Often computational approaches model design as a search process or as an optimization problem. The nature of such techniques, however, does not allow the user to interact with the search process once it has begun. Furthermore, traditional computational search lacks the ability to learn from experience. 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 method introduced here improves the capabilities of design synthesis methods by allowing for user input and by making decisions based on previous experience. In studying past candidate designs the process learns to be more effective in searching for solutions. Results show how such a technique improves the quality of designs and efficiency of an existing search process.

Received: 12 May 2001 / Revised: 15 October 2002 /
Accepted: 15 October 2002 / Published online: 28 November 2002

M.I. Campbell (✉)
Department of Mechanical Engineering,
The University of Texas at Austin, 1 University Station,
C2200 Austin, TX 78712-0292, USA
E-mail: mc1@mail.utexas.edu
Tel.: +1-512-2329122
Fax: +1-512-4717682

J. Cagan
Department of Mechanical Engineering,
Carnegie Mellon University, Pittsburgh, PA 15213, USA

K. Kotovsky
Department of Psychology, Carnegie Mellon University,
Pittsburgh, PA 15213, USA

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. The authors would also like to acknowledge the efforts of Jarrod Moss in the testing and development of the learning algorithm and the financial support of the National Science Foundation under grant EID-9256665.

Keywords Learning, Agents, Multi-objective optimization, Design synthesis, Stochastic optimization

Introduction

As design synthesis techniques find their way into engineering practice, the designer has at his/her disposal computational processes to alleviate the need to perform tedious or time-intensive design tasks by hand. While the computer has become an indispensable companion to the engineering designer because of its speed and accuracy, the most significant decisions of the design process remain a product of human decision-making. Often several challenges within a larger engineering design problem can be restated as a computational search problem. The computer exhaustively, or by following heuristics, searches for a solution that best meets the designer's goal. From this model, numerous techniques have developed to enhance engineering design quality and efficiency. While the model of design as a search process has proven to be useful, it appears that challenges still exist in integrating the human and computer as a symbiotic design team. In order for the computer to tackle more ambitious and conceptual design tasks several theoretical and computational hurdles must be overcome.

In Fig. 1, a general model of computational synthesis shows four basic aspects of modeling design as a search process. The process initiates with some starting point description of the design problem and some qualities for making better designs. Prior to the search process, a language or representation needs to be developed for the domain of the problem. Based on the representation, the search process is performed as an iterative loop of generating alternatives, evaluating their worth, and determining how to perform redesign in subsequent iterations. While each of these tasks could be the impetus for future research, the method described here focuses on improving the guidance task of the iterative process. Guidance in human design processes results from learning from past experiences and understanding how to balance the tradeoffs in a design problem.

This paper establishes a method to overcome two shortcomings of current computational synthesis techniques. These are (1) the lack of information exchanged during the search process between man and machine, and (2) the lack of learning that occurs throughout computational search. The strategies developed here to overcome these challenges are together enveloped in a software agent that interacts with other software agents responsible for separate aspects of the search process. Altogether, the

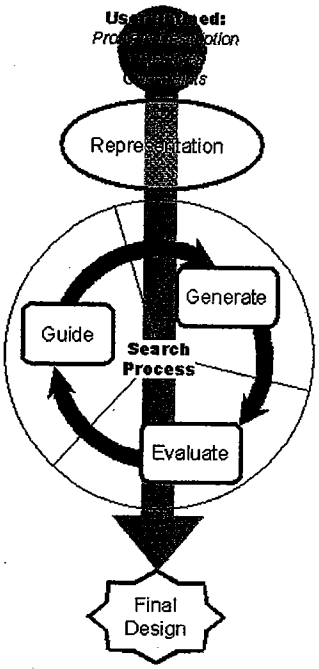


Fig. 1. The computational search process has four basic divisions: (1) a representation of the design space, (2) a method for generating new solutions, (3) a method for evaluating solutions, and (4) a method for guiding the search process

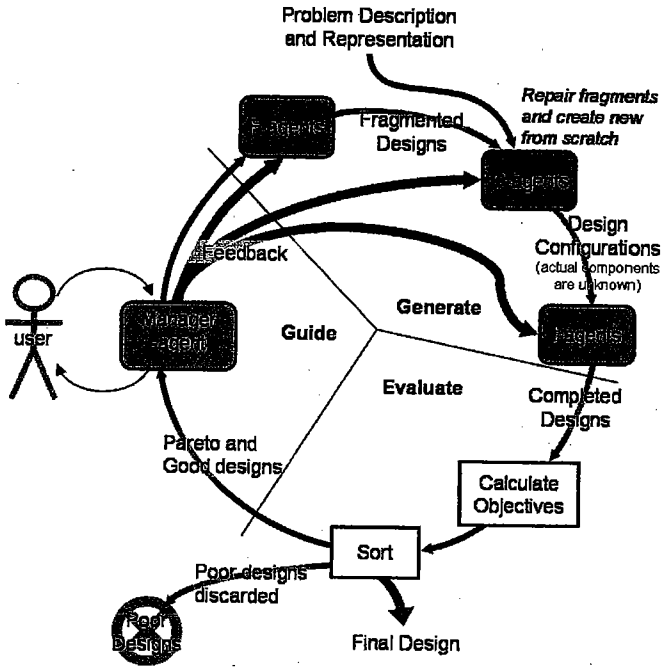


Fig. 2. A-Design accomplishes the search process by bringing together several types of agents and the user's input

agents are combined within an existing design methodology known as A-Design (Campbell 2000). The A-Design approach combines aspects of several different research endeavors in order achieve a synthesis technique capable of solving design problems at a more conceptual level than has previously been investigated.

Prior to the work shown here, A-Design had three basic components: (1) a set of interacting agents that generate possible design alternatives, (2) a multi-objective evaluation method capable of handling changes in objectives, (3) a representation for describing electromechanical designs of varying topologies. In previous publications, the representation (Campbell et al. 2000), generation, and evaluation (Campbell et al. 1999) procedures have been described in detail. In Fig. 2, the A-Design approach is detailed as a specific instance of the general flowchart in Fig. 1. Three types of agents are responsible for generating solutions to the design problem. The organization and operations of these agent teams along with the representation and evaluation methods are summarized in Sect. 2.1. The techniques described here are combined with the manager-agent in the guidance section of the search process. In order to improve design knowledge, a dialog with the user provides further guidance to improve designs.

The development of the manager-agent has shown to both improve the quality of the final designs and the time needed to arrive at those solutions. This enhanced performance is due to the fact that the new system can learn from its mistakes as well as its successes. It can also learn what the user feels is important in the process. Furthermore, the user interaction and the learning increase the power of the computational design process by making it more similar to how human engineers solve design problems.

After the section on related work (Sect. 2), Sect. 3 describes the details of the various subprocesses of the manager-agent. These include (a) a method of communicating with the user and adapting to the changes that he/she proposes (Sect. 3.1), (b) a method of detecting commonalities in past designs and storing them as either good design examples or bad design examples (Sect. 3.2), (c) a method of using these commonalities to adjust how the other agents are organized (Sect. 3.3), and (d) a method by which the other agents learn from the commonalities found by the manager-agent (Sect. 3.4). While each of these components have been combined in the development of the manager-agent, it is possible that they can be used independently outside of A-Design to aid other computational design synthesis methods. The results of this paper are divided into two parts in Sect. 4. The first subsection illustrates the effect of the manager-agent on an electromechanical design problem: the design of a body-weight scale. In the second subsection, statistical results are gathered to verify the effect of the manager-agent guidance. Throughout the descriptions in Sect. 3, the body-weight scale is used as an example to illustrate the operations of the manager-agent.

Related work

A-Design framework

A-Design developed as a combination of several distinct computational techniques with the goal of exploring automated design synthesis as a tool for aiding or accomplishing the 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. At the foundation of the A-Design approach 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.

As seen in Fig. 1, the first aspect of computational design is the development of 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 for conceptual problems 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 1996; Ulrich and Seering 1989; Schmidt and Cagan 1998), linguistic approaches of 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. (2000), 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 conceptual design of complex problems can be accomplished.

Figure 1 shows that the next three tasks of a computational design process occur within an iterative loop where numerous candidate designs are visited in searching the space for the most successful design. The first of these steps is to generate candidate solutions. A-Design creates design alternatives through a hierarchy of software agents as seen in the detailed A-Design approach to search in Fig. 2. The multi-agent collaboration in A-Design is inspired by work from artificial life (Langton 1988; Holland 1992). Artificial life or A-Life puts computation in the hands of naïve agents who, through following simple reflexive operations, generate a complicated emergent behavior. The idea of exploring multi-agent systems in engineering applications has proven useful in asynchronous teams research (Talukdar et al. 1996). Asynchronous teams (A-Teams) is a method of encapsulating various computation strategies such as optimization techniques into autonomous agents that are capable of parallel and unstructured interaction, resulting in a more thorough and efficient approach to solving design problems. It has often been found that multi-agent systems succeed when the demands for a more flexible and knowledge-based approach requires the interaction of various software components (Lander 1997).

The A-Design agents build, update, and modify solutions through use of the design representation. Based on the various responsibilities and preferences, agents are divided into different classes. As seen in the figure, there are three classes of agents. Configuration-agents (C-agents) choose how components should be connected in a design but do not choose actual component values. The next set of agents, the instantiation-agents (I-agents), finds specific components from a catalog to fulfill the design configurations. At the completion of an iteration, designs preserved from the previous iteration are fragmented by the fragmentation-agents (F-agents) so that they can be improved through reconstruction in the next iteration. All of these agents encapsulate knowledge-driven strategies for solving open-ended problems that, through their collaboration, achieve a specified design goal. As a result, each design alternative results from the contributions of several agents interacting together. By cooperatively combining these agent 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. To overcome this, 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), has 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. As seen in Figure 2, first objectives are calculated so as to determine which designs are more successful at meeting the specified goals. From these calculations, A-Design employs a unique strategy whereby candidates are divided into three separate populations labeled Pareto designs, good designs and poor designs. From this division, the Pareto and good candidates are preserved for use in the next iteration, while the remaining poor designs are discarded (see Fig. 2).

In Fig. 3, the method of sorting designs is shown. 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 is a conservative preservation of designs depicting a diversity of relative strengths in the objectives. Outside of Pareto-optimal designs, the system further 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

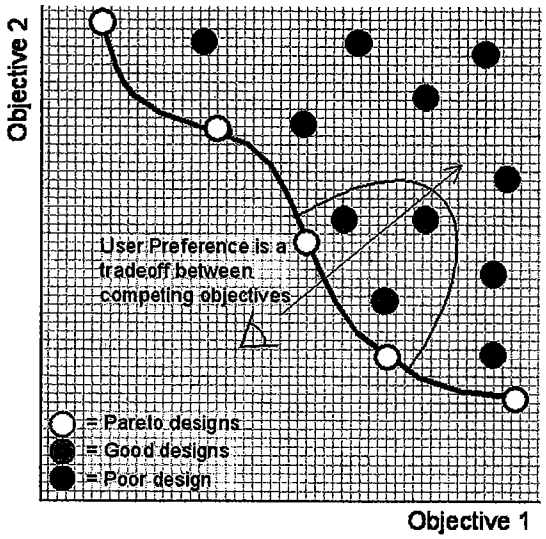


Fig. 3. Design selection separates a population into three sets: Pareto designs, good designs, and poor designs

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 3). 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 to adapt to changes in the process. Experiments that reveal how this division of designs makes A-Design adaptive can be found in Campbell et al. (1999).

Learning

The final task of the search process is to guide the iteration toward better solutions. Based on information gained from past iterations, the process is steered towards productive areas of the search space. The approach developed here is inspired by previous search techniques such as simulated annealing (Kirkpatrick et al. 1983), and genetic algorithms (e.g., Goldberg 1989; Queipo et al. 1994; Koza et al. 1996), which contain a stochastic guidance approach. The guidance method in these algorithms determines which solutions are retained for future modification in subsequent iterations of the search process. In addition to preserving solutions, statistics can be used in these methods to adjust the design decisions in the generation of future designs. Examples of this include proportional selection in genetic algorithms (Bäck and Hoffmeister 1991) and move set probabilities in simulated annealing (Hustin and Sangiovanni-Vincentelli 1989), both of which have influenced the manager-agent innovation in A-Design. Perhaps the most similar stochastic optimization technique to the manager-agent is the Tabu search technique (Glover 1989; Moscato 1993). This technique keeps track of previously visited candidate solutions in order prevent the search from repeating itself. Past solutions are stored on a Tabu list as memory to prevent the process from backtracking, thus forcing the process into new areas of the search space. Within the manager-agent model developed for A-Design, a similar approach is developed which stores good and bad design paradigms in order to influence future generation of new designs. The resulting method is called TODO/TABOO learning and is discussed in more detail in Sects. 3.2, 3.3, and 3.4.

In addition, the guidance of the manager-agent is similar to several other machine learning techniques (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).

Manager-agent guidance

The iterative search process of A-Design can be seen as analogous to the iteration occurring in the human design process. However, in human problem solving, learning plays a key role in determining 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. Similarly, the philosophy behind the manager-agent guidance is to observe past design activity and make decisions about how to improve designs in the future. This guidance will mainly effect how the other agents in the system generate new candidate designs. However, 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 learns what the user's preferences are for successful designs and as a result, tailors the search to meet these preferences.

This section is divided into four subsections to describe in detail the overall behavior of the manager-agent that is shown in Fig. 2. After designs are sorted into the three populations of Pareto, good, and poor, the manager-agent queries the user about the current state of the search process. Sect. 3.1 details how this dialog works and how the process learns from the user's responses. Next, in Sect. 3.2, the manager-agent examines the past designs to find common sets of agents and common configurations of components referred to as "trends". 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. After the TODO and TABOO lists are constructed, the manager-agent invokes the next generation cycle by controlling how agents are selected through a stochastic method based on their prior performance (see Sect. 3.3). Finally, the TODO and TABOO trends are provided as feedback to the agents. The configuration-, instantiation-, and fragmentation-agents incorporate these trends with their individual preferences to make informed decisions on how to construct better designs. This method of feedback is similar to reinforcement learning, and is discussed in Sect. 3.4.

Adapting to user preference

One of the unique characteristics of the A-Design system is the ability to adapt to changes in the objectives of the design problem. This adaptation affords the process the ability to interact with the user so that design activity can be monitored and directed throughout the synthesis of new designs. In each iteration, the manager-agent displays various statistics on the current state of the process and prompts the user for input. If the user decides not to interact, the process continues. If the user sees that a change is required, he/she can temporarily halt the process and enter into a dialog with the manager-agent. The manager-agent then prompts the user with questions about the current set of designs to determine what changes can be made to adapt to the user's needs.

Figure 4 shows a typical dialog from the body-weight scale test problem that includes four objectives to be minimized: cost, weight, dial error, and input displacement. In this example, the process is initiated with linear weights on the objectives of [1 10 1,000,000 1,000,000]. At each iteration data on the current state of the process are presented to the user (population sizes, the amount of change in the Pareto set, etc.), as well as the current best design for the user preference. In this figure, the user hits return at the end of the sixth iteration initiating a brief dialog with the manager-agent.

Having presented the attribute values of the best design at this juncture of the process, the manager-agent asks the user to rate three random Pareto-optimal designs on a scale of one to ten. In order to provide a reference for the user, the current best solution is assigned a value of 5 so that the user may rank the remaining three designs in proportion to this value. This ranking supplies the system with data used to approximate a user's new utility function. It has been shown that a user can accurately estimate meaningful values for his/her utility for such designs due to experiments in human perception. Most notably, Stevens (1975) has shown that people are capable of accurate prediction through experiments that correlate perception and cardinal rankings in a technique known as magnitude estimation.

As a result of this dialog, the manager-agent determines a new form of the utility function in an approach similar to the learning of "static evaluators" through linear regression (Abramson 1990). From these cardinal rankings, the manager-agent performs a best fit to the system of linear equations that result from the weighted sum of objectives. Figure 5 shows how these equations are constructed for the dialog in Fig. 4. Based on this best fit, the new best Pareto design is presented to the user. To further refine the accuracy of the newly adapted weights, the manager-agent asks the user if the new preference is somewhere between the new best design and the prior best design or tending more towards the new design. This dialog is used to determine if the user preference can be adjusted by interpolated between new and old or to extrapolate a new target beyond the newly determined preference. In the dialog of Fig. 4, the user opts for an interpolation of the new and prior preferences, and as a result a new approximation for the weights is established, in this example the new weights change from "1 10 1,000,000 1,000,000" to "23.5 28.0 500,023.0 500,000.0".

The design selection process described above now has a means to determine where the good population of designs should lie – at the intersection of this new preference and the Pareto surface. Further, the result of changing this preference leads the A-Design process and the manager-agent to present new solutions in the subsequent iterations of Fig. 4. While this demonstration is descriptive of the adaptive nature of A-Design, further experimental results demonstrate its effectiveness in Campbell et al. (1999).

Finding TODO and TABOO 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 frequently 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 accomplished through simple mathematical set intersection operations, however, the first type of trends (design fragments) is found through a specialized graph intersection operation.

In order to determine trends, all designs Pareto, good and poor are sorted from best to worst by collapsing the objectives to a single metric as seen for two objectives in the example of Fig. 6a. In this figure, the eye represents a user preference approximated via the "manager-agent to user" dialog shown in Sect. 3.1. The designs, A–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 Fig. 6b).

Next the designs are exhaustively compared to find commonalities in the selected designs. In Fig. 6c, the top five designs separated for TODO comparison are examined to find commonalities 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 mem-

```
Iteration 6.
Population = 44.
Pareto has 13 members.
Pareto changed by 7.
The ave. Pareto values are
 (46.295383 0.570433 7.8543525
  0.0028573587).
The best Pareto values are
 (18.5 0.14699998 0.0012533892
  9.648292e-4).
Good has 16 members.
 The radius is 47.80917.
Poor has 15 members.
Todo has 0 members.
Taboo has 0 members.
Weights are (1.0 10.0 1000000.0 1000000.0)
Entering evaluate....done
Pareto last changed 0 iterations ago.
Top Design last changed 5 iterations ago.
Currently the best design for your preference
is:
     COST = 72.84
     MASS = 0.23563002
     DIAL-ERROR = 0.0012533892
     INPUT-DX = 0.002038508
----------------------------------------
Talk to M-agent (hit return) ?

Given that the best design presented is 5,
how would you rate each of these designs
(1-10)?

1:   COST = 26.43
     MASS = 0.175
     DIAL-ERROR = 0.066433564
     INPUT-DX = 9.648292e-4
----------------------------------------
2:   COST = 84.13
     MASS = 2.9145
     DIAL-ERROR = 4.7503047
     INPUT-DX = 9.648292e-4
----------------------------------------
3:   COST = 85.13999
     MASS = 2.232
     DIAL-ERROR = 6.769965
     INPUT-DX = 9.648292e-4
----------------------------------------
1: 10
2: 2
3: 2

Is this design as good as the best shown above
[y/n]?
     COST = 20.99
     MASS = 0.164
     DIAL-ERROR = 0.31395644
     INPUT-DX = 0.002038508
----------------------------------------
y
Interpolate or Extrapolate new preference[i/e]
Is your preference between these two designs
or more along the lines of the second? i
new weights = 23.53806 28.02538 500023.03
              500000.0
The design process is 0.15 complete.
```

```
Iteration 7.
Population = 46.
Pareto has 17 members.
Pareto changed by 4.
The ave. Pareto values are
 (45.26059 0.52912116 6.0937095
  0.003011668)
The best Pareto values are
 (18.11 0.14699998 2.5082377e-5
  9.648292e-4).
Good has 15 members.
   The radius is 47.212936.
Poor has 14 members.
Todo has 0 members.
Taboo has 0 members.
Weights are (23.5 28.0 500023.0 500000.0)
Entering evaluate....done
Pareto last changed 0 iterations ago.
Top Design last changed 5 iterations ago.
Currently the best design for your preference
is:
     COST = 22.3
     MASS = 0.169
     DIAL-ERROR = 0.31395644
     INPUT-DX = 0.002038508
----------------------------------------
Talk to M-agent (hit return) ?


Iteration 8.
Population = 49.
Pareto has 22 members.
Pareto changed by 9.
The ave. Pareto values are
 (48.8 0.46536332 2.2392752 0.002516745)
The best Pareto values are
 (18.11 0.14199999 2.5082377e-5
  9.648292e-4).
Good has 14 members.
 The radius is 48.007374.
Poor has 13 members.
Todo has 0 members.
Taboo has 0 members.
Weights are (23.5 28.0 500023.0 500000.0)
Entering evaluate....done
Pareto last changed 0 iterations ago.
Top Design last changed 1 iteration ago.
Currently the best design for your preference
is:
     COST = 19.71
     MASS = 0.172
     DIAL-ERROR = 0.31395644
     INPUT-DX = 0.002038508
----------------------------------------
Talk to M-agent (hit return) ?
```

Fig. 4. A display of the dialog that occurs between the user and the system in adapting to user preference

bers 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

Given: the best design dialog of Figure 4 has attribute values of:

$$f_{01} = 72.84$$
$$f_{02} = 0.23563002$$
$$f_{03} = 0.0012533892$$
$$f_{04} = 0.002038508$$

and has a ranking $r_0 = 5$

Find: new values for w_1, w_2, w_3, w_4.

- Manager presents three new designs.
- Users ranks these 10, 2, 2
  - $r_1 = 10, r_2 = 2, r_3 = 2.$
- Establish as a system of linear equations:

$$w_1 \bullet f_{01} + w_2 \bullet f_{02} + w_3 \bullet f_{03} + w_4 \bullet f_{04} = r_0$$
$$w_1 \bullet f_{11} + w_2 \bullet f_{12} + w_3 \bullet f_{13} + w_4 \bullet f_{14} = r_1$$
$$w_1 \bullet f_{21} + w_2 \bullet f_{22} + w_3 \bullet f_{23} + w_4 \bullet f_{24} = r_2$$
$$w_1 \bullet f_{31} + w_2 \bullet f_{32} + w_3 \bullet f_{33} + w_4 \bullet f_{34} = r_3.$$

- Solve for w_i.
- Interpolate/Extrapolate with previous w_i.

Fig. 5. Determining the new weights is a process of solving the linear system of equations that is created by the cardinal ranking of various Pareto design states

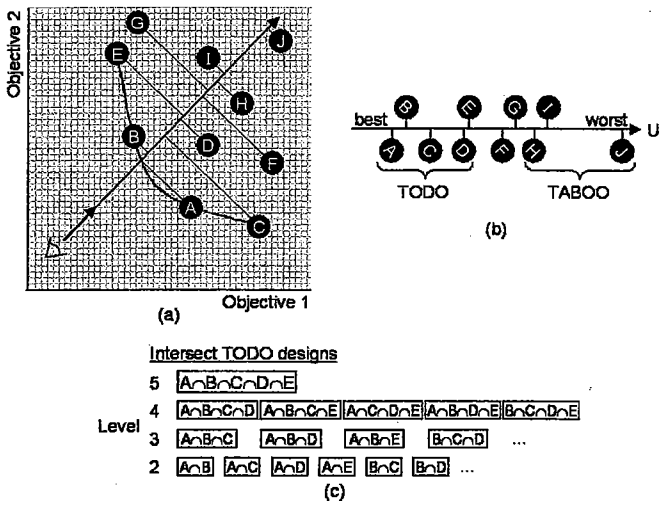


Fig. 6. An example of detecting TODO and TABOO trends from a set of 10 designs

between two designs. However, many connections and repeated components make finding common design fragments difficult.

In the A-Design implementation, the intersection procedure is not particularly time-consuming, but because of the number of intersections (as in Fig. 6c), 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.

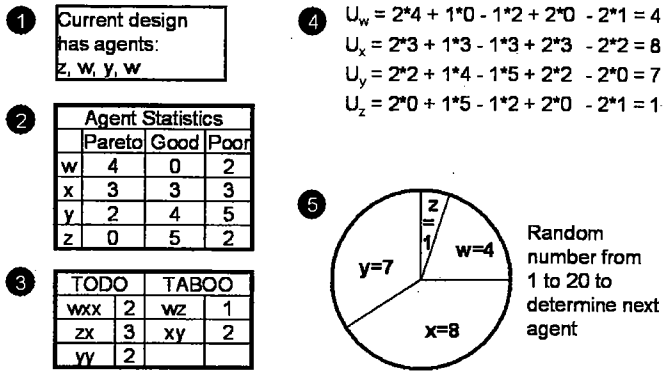


Fig. 7. Manager-agent proportions probabilities based on evaluation function

Using trends to invoke agents, and agent-teams

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 in generating designs is completely random. However, with the information provided in the trends, the manager-agents can advocate specific agents at certain times in design generation. Although the agents are still invoked randomly, this manager-agent can weight certain agents higher than others based on their prior performance, thus making the process stochastically guided.

Before every agent call, the manager-agent rates each agent based on prior statistics and prior collaborations with other agents as determined by the TODO and TABOO trends. In addition to these 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 7 illustrates an example of how this stochastically guided proportional agent selection is accomplished. During the construction of one particular design, the previous agents *z*, *w*, *y*, *w* have been invoked (as seen in Fig. 7, element 1). The manager-agent then sets up probabilities for the next agent to call based on information on the agent statistics on performance quality from previous iterations. For example, as seen in Fig. 7, element 2, the statistics for agent A 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:

$$\begin{aligned} U = {} & \text{wM1} \times \text{num\_of\_pareto} + \text{wM2} \times \text{num\_of\_good} \\ & - \text{wM3} \times \text{num\_of\_poor} \\ & + \text{wM4} \times \text{TODO\_team} \\ & - \text{wM5xTABOO\_team}. \end{aligned} \quad (1)$$

The number of past Pareto designs the agent contributed to (num_of_pareto), the number of past good designs (num_of_good), and the number of past poor designs

(num_of_poor) weigh in to the manager-agent's evaluation. Design teams from the TODO and TABOO lists (Fig. 7, element 3) also influence how each agent is considered. If 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 Fig. 7, element 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 (Fig. 7, element 4). As a result of these values, the manager-agent divides the probabilities as shown in Fig. 7, element 5. The process then picks a random number to determine which agent to invoke based on the manager-agent's division. The weights in Eq. (1) are prescribed to be wM1=wM4=wM5=2, wM2=wM3=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 to various strengths and weaknesses, and encourages certain agents to collaborate on specific or diverse designs. The process of continually recommending agents based on the state of a design is a unique guidance mechanism of A-Design.

How agents learn from manager-agent feedback

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. Similar to the manager-agent functionality, the C- I-, F-agents have unique evaluation functions that combine terms from individual agent preferences and TODO/TABOO learning to determine what components to add or remove from a design.

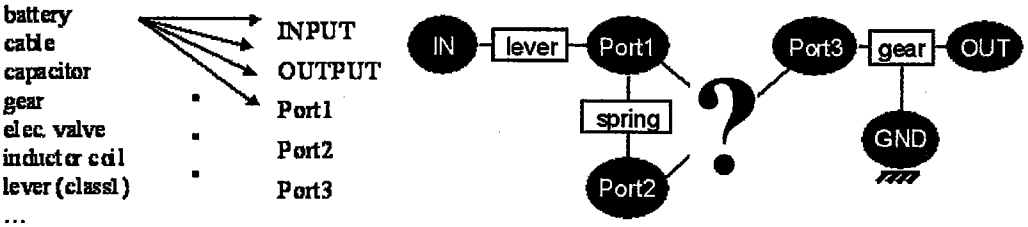
To illustrate the agents' learning we will examine in detail how the configuration-agents choose components. The utility functions, or evaluation functions, of the configuration-agents include five terms which quantify (1) what component is chosen (VCOMP), (2) how a new component is connected (VCONNECT), (3) the components it connects to (VOTHER), (4) the adherence to TODO trends (VTODO), and (5) the avoidance of TABOO trends (VTABOO). 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, VTODO and VTABOO, 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).

In Fig. 8, pseudo-code is presented to clarify this agent's operation and 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 the effective outcomes of adding each component. As can be seen in points b, c, d of Fig. 8, extra points are awarded for choices that fulfill goal states of the design problem. 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 (wC1, wC2, wC3, wC4, wC5) 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 implementation for each agent. As opposed to the manager-agent weights (wMi), 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 wC1) more than how it is connected in a design (lower values for wC2, wC3). Also, some agents can strongly consider learning influences (higher wC4, wC5), while others ignore learning (wC4=0, wC5=0). With this approach, a variety of agents can be constructed to produce more variety in the design process. In the current A-Design system, four agents, each with different weights, are formed for each agent type. This results in a total of 192 C-agents in the A-Design process (48 types multiplied by four of each type). Future work with this utility agent model will investigate alternative ways of choosing weights either automatically or through experimentation.

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

$$V = w_1 \times V_{COMP} + w_2 \times V_{CONNECT} + w_3 \times V_{OTHER} + w_4 \times V_{TODO} - w_5 \times V_{TABOO}$$

2. **Choose new component and connection producing the highest value of V**

Output = design state updated with new component and connections.

Fig. 8. Pseudo-code for configuration-agent

Results

In this section, the manager-agent's inclusion into the A-Design process is evaluated. Simulations of A-Design are performed with and without the guidance mechanism. A design problem is posed, which has several design objectives, and an unstructured representation that allows for a limitless space of possible solutions. Because of these issues and the lack of a known global optimum, developing measures to compare the success of different runs is not trivial. In order for A-Design to synthesize design solutions, three things must be specified for a design problem: the set of objectives, a specification of the inputs and outputs, and a catalog of components. Each of these specifications to the system provides bounds to the synthesis capabilities of A-Design. For example, designs are only constructed from the components provided in the catalog; there is currently no means of finding components not represented in the catalog. The first section below examines the quality of the final designs produced with and without manager-agent guidance, while Sect. 4.2 examines statistics of different TODO/TABOO learning strategies gathered over the iterations.

Body-weight scales

Figure 9 provides a visualization of the specifications for describing the body-weight scale 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.

Designs resulting from running the process for 30 iterations with a population of 100 designs per iteration are compared. Designs A and B in Fig. 10 do not include the manager-agent guidance and are to be compared to designs C and D that have the guidance mechanism present. One interesting aspect of the resulting designs is

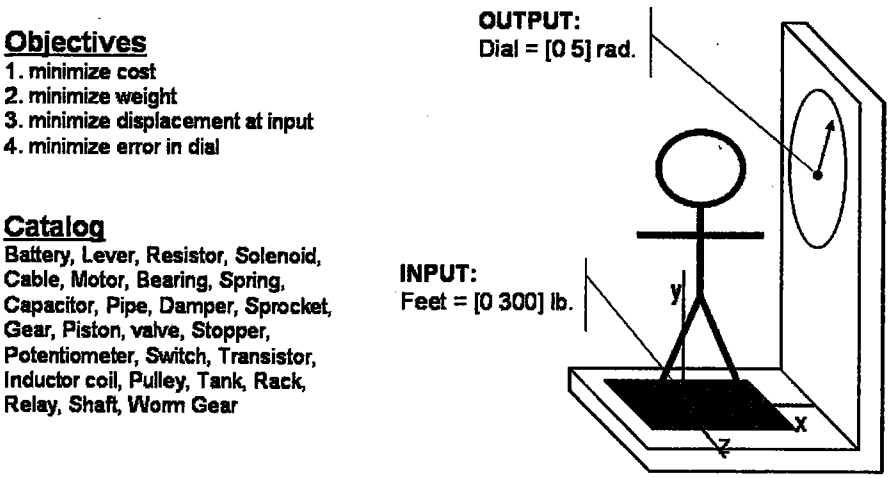


Fig. 9. Description of body-weight scale design problem as posed to A-Design

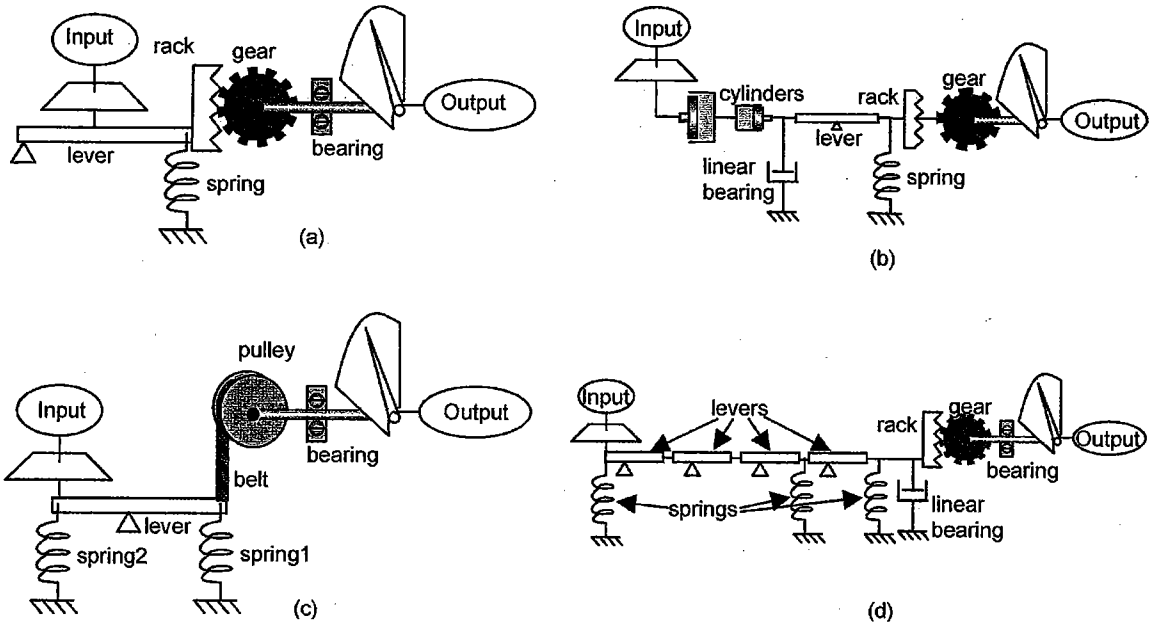


Fig. 10. Four final designs created by A-Design. Designs A and B are created without TODO/TABOO learning. Designs C and D are created with TODO/TABOO learning

the diversity in their configurations and the different degrees to which they meet the user objectives. These solutions are nearly constructible artifacts. They are comprised of actual components and indicate how these components connect to one another. However, there are some specific design issues that A-Design has not yet been able to address. In examining the results in Fig. 10, one may perceive possible secondary problems with these designs, for example the lack of bearings to properly support the shaft as well as other bracketing issues. Currently, we are exploring ways that the design process can become aware of these inconsistencies by improving the representation of individual components (Campbell and Limaye 2002).

Figure 11 shows how each of the four designs are evaluated on the four objectives. 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.

Statistical results for TODO/TABOO learning

In this section, statistics are gathered on the body-weight scale test problem to better compare how design quality is improved with TODO/TABOO learning. Figure 12 shows three 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.