Research in Engineering Design (1999)11:172–192

Research in
Engineering
Design

A-Design: An Agent-Based Approach to Conceptual Design in a Dynamic Environment

Matthew I. Campbell[1], Jonathan Cagan[1] and Kenneth Kotovsky[2]

[1]Computational Design Laboratory, Department of Mechanical Engineering; [2]Department of Psychology, Carnegie Mellon University, Pittsburgh, PA, USA

Abstract. *This paper provides an introduction to a new design methodology known as A-Design, which combines aspects of multi-objective optimization, multi-agent systems, and automated design synthesis. The A-Design theory is founded on the notion that engineering design occurs in interaction with an ever-changing environment, and therefore computer tools developed to aid in the design process should be adaptive to these changes. In this paper, A-Design is introduced along with some simple test problems to demonstrate the capabilities of different aspects of the theory. The theory of A-Design is then shown as the basis for a design tool that adaptively creates electro-mechanical configuration designs for changing user preferences.*

Keywords: Adaptive search; Agents; Design synthesis; Multi-objective optimization

1. Introduction

Traditionally, the engineering design process is a cooperative effort among engineers and management, where virtual and physical prototypes are constructed, tested and evaluated. Based on constantly changing design goals, available technologies, and design constraints, managers refine the design specifications and appropriate design teams, while engineers within those teams collaborate to affect changes to the design to meet target specifications.

The goals of a design problem are established to address market-driven issues such as product cost or performance, and engineering objectives such as target lifecycle or efficiency. One company's product can be distinguished from others in the manner in which it achieves the various product design goals. While some companies produce inexpensive designs in order to find a profitable market niche, others may produce high-end products; to position one's product amongst competing products a constant awareness of market demands is required. However, in the conceptual design phase, it is not always possible to set the relative importance of the various objectives and constraints prior to generating solutions, as is required in most automated design systems. Current computer tools tend to be static in, and unresponsive to, such changes in the problem description and the solution strategy. These limitations motivate the conceptual design methodology introduced in this paper, which is capable of adapting to changes in design problem specifications.

This paper introduces A-Design – a new search strategy for the conceptual stages of engineering design that incorporates **a**gent collaboration with an **a**daptive selection of designs. Having a multitude of agents that are responsible for the same task yet present different strategies and differing knowledge for solving the same design problem generates a greater variety of design alternatives. The system gains robustness due to the collaboration of these varying agents, thus affording the design process the ability to change focus when the user's preference changes. These agents are analogous to individual specialists within a design firm. As with a company, there is usually not a single optimal design, but rather designs evolve and, as they do so, the company selects designs at certain states and markets them as products. This dynamic nature of true engineering design is the foundation for creating A-Design in a manner that is adaptive in both the design process and design specification.

The basic subsystems of the A-Design theory are (1) an agent architecture that is responsible for creating and improving design alternatives, (2) a representation of the conceptual design problem that is comprehended by the agents in order to create design concepts, (3) a scheme for multi-objective

Correspondence and offprint requests to: Dr. J. Cagan, Computational Design Laboratory, Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA

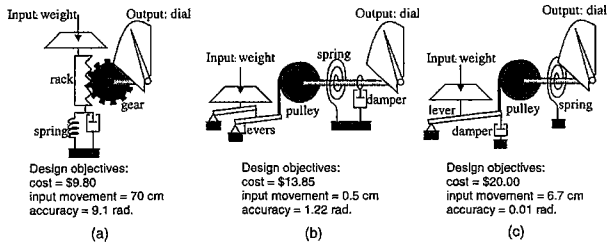


Fig. 1. Initial weighing machine results: (a) Low cost design, (b) minimum input displacement design, (c) accurate dial design.

decision making that retains solutions exhibiting different patterns of strengths and weaknesses in order to accommodate change in user preferences, and (4) an evaluation-based iterative algorithm for improving basic design concepts towards successful solutions. This paper introduces the A-Design theory for conceptual engineering design, with some examples to illustrate its general search technique and adaptability. The theory is then implemented on a problem where the goal is to produce configurations for a specified electro-mechanical function. These configurations are composed of actual components chosen from a catalog of various manufacturers, and the topologic connection of those components within the design. This electro-mechanical A-Design process differs from previous work in that (1) designs are not only generated but iteratively improved upon to best meet current objectives specified by a user, (2) synthesis of designs is performed within a rich descriptive representation of components and configurations that models real-world component interactions, and (3) key design alternatives are retained to allow the system the flexibility to adjust to future changes in the problem specifications.

The flexibility in the electro-mechanical A-Design method allows for the production of a variety of solutions for a given design problem. For example in Fig. 1 the user has defined a design problem to create weighing machines (fully described in Section 6) for which the process creates a variety of results with different strengths and weaknesses. In this problem, the user poses three objectives to be optimized within the A-Design process: minimize cost, minimize displacement at the input, and minimize the error in the dial. This figure shows three different designs that were generated by the system and exhibit different strengths in each of the objectives. Through choosing the relative importance of the design goals, the designer can guide the A-Design process as it generates practical and useful alternatives for a given set of needs. The designer's preference for the objectives can be changed throughout the process, in response to the types of designs that are being produced and to changes in the market forces that are driving the process.

In the remainder of this paper, we first describe related work and introduce the A-Design theory. Next we present two examples that illustrate the search strategy and adaptability of the approach. Implementation of A-Design as an electro-mechanical configuration design tool is then presented, and the resulting system is tested on a practical design problem, the design of a weighing machine. Finally, the implications of the technique are discussed.

2. Related Work

The A-Design methodology combines innovations from artificial life, genetic algorithms, stochastic optimization, multi-objective optimization, qualitative physics and asynchronous teams. This section places A-Design within the perspective of these approaches. Table 1 summarizes the four basic subsystems of A-Design, and shows how they relate

Table 1. Derivation and Innovations of the A-Design theory and the electro-mechanical implementation

A-Design subsystems	Agent archecture	Iterative-based search strategy	Design selection	Representation of electro-mechanical systems
Related work	Artificial Life, MADS, A-Teams	Stochastic Optomization, Genetic Algorithms	Pareto Optimality	Qualitative Physics, Functional Grammars
A-Design innovations	Agents are goal-directed and work within a specific A-Design hierarchy where feedback to agents provides for an ever-changing environment	A-Design iteratively improves upon existing design alternatives and creates new ideas through management of maker- and modification-agents	Designs are divided into three unique populations that are used to add adaptability and diversity to design states and to provide a mechanism of feedback for agents.	Two tier representation allows system to both describe behavior of components and instantiate behavior with real-world components.

to each of the different approaches. The third row describes how A-Design has expanded upon or diverged from these research ideas to create the unique constituents of the theory. The following subsections discuss each of these areas as they relate to A-Design.

2.1. Artificial Life

Recent insights on how nature performs complicated problem solving have led to the field of Artificial Life (Langton 1988, Holland 1992). Artificial Life puts computation in the hands of naïve agents who often, through following simple reflexive operations, achieve a complicated emergent behavior. The complex adaptive systems created by the interaction of agents has the benefit of a highly distributed and parallel process that can lead to enhanced robustness and computational speed. Similarly, A-Design's combination of different agent behaviors occurring together within an engineering design setting ideally leads to more creative solutions than would be obtained from a system having one intelligent agent performing all design tasks.

2.2. Multi-Agent Design Systems (MADS) and Asynchronous Teams

Multi-agent systems have previously been applied in a number of engineering design applications (see the overview by Lander 1997). The approach in these multi-agent systems is to use agents to handle the pre- and post-processing of various computational analysis tools such as spreadsheets or CAD systems in order establish a common communication among them. The agents, which communicate through a common framework, act as 'experts' in that they represent the results produced by their encapsulated applications and present it to the design process (see the example in Goldstein 1994). Currently, these approaches offer an unobtrusive approach to communicating between the large computer tools used in concurrent engineering design. Other research projects in this area are incorporating reasoning and learning (Grecu and Brown 1996) into these agents in order to more closely simulate strategies used by human designers.

Asynchronous Teams, or A-Teams (Talukdar 1996), is a computational methodology which combines design utilities such as optimization techniques with autonomous agents. These autonomous agents perform computations independently from other agents, and contribute their results in a parallel and distributed fashion. It is believed that combining programs in a cooperative yet anarchistic manner will produce better results than if programs were executed individually. Talukdar (1993) describes this as synergy: "When the effectiveness of cooperation is so great that a super-object is, in some sense, greater than the sum of its parts, the cooperation is synergistic". The philosophy of the agent-based approach of A-Design is similar to that of A-Teams. Note that A-Teams have also explored the use of a multi-objective selection of designs, as seen in Murthy (1992) and Quadrel et al (1993). These projects were similar to A-Design both in the division of labor in agents, and in the agent preferences for design objectives. What differentiates our work (as will be discussed below) is our functional representation used to model and evaluate design alternatives, our hierarchical and specialized agents that operate upon various subtasks within the design problem, and our management and dynamic modification of agent populations.

2.3. Genetic Algorithms

Genetic algorithms are a popular form of problem solving (see the overviews in Mitchell 1996 and Goldberg 1989), and have been applied to various engineering design problems (e.g. Queipo et al. 1994, Koza et al. 1996, Gage and Kroo 1995). Often, they are used to find solutions in highly constrained situations where the user is more concerned about finding feasible solutions and less about optimizing.

Genetic algorithms synthesize solutions by combining the configurations of other alternatives. These algorithms store many design states simultaneously, as is done in A-Design, in order to compare, propagate and modify alternatives in a similar way to that used in natural evolution. A-Design differs most from these algorithms in that it maintains and evolves both populations of designs and populations of the design creators, the agents. Further, modification of designs in A-Design is determined by goal-directed agents rather than only random mutation and crossover, allowing for a more efficient search of feasible design states by preventing undirected design creation.

2.4. Multi-Objective Optimization

Pareto optimality has been frequently used to address multi-objective optimization problems, for example in Balachandran and Gero (1984) and in Eschenauer et al. (1990). Many algorithms, especially genetic

algorithms, have incorporated this notion of comparing designs (see the overview in Fonseca and Fleming 1995). For example, Schaffer (1985) developed VEGA, a genetic algorithm that incorporates both Pareto optimality and computational search. The combination of Pareto optimality and agents has been explored by Petrie et al. (1995), where existing software tools are controlled by a single governing agent that makes decisions to keep designs based on Pareto optimality. Other than Pareto optimality, multi-objective conceptual design is also tackled through multi-attribute selection (Thurston 1991, D'Ambrosio and Birmingham 1995), and through spectral optimization (Dong and Agogino 1995). Our work uses Pareto optimality as the basis for dividing designs into the three populations of Pareto, good and poor, which is done to give the A-Design system both flexibility and focus on the changing problem specifications as shown in Section 7.

2.5. Stochastic Optimization

Few optimization techniques are able to address the highly nonlinear, discontinuous and multi-modal problems found in engineering design. However, stochastic optimization techniques such as simulated annealing (Kirkpatrick et al. 1983), Tabu search (Glover 1989) and hybrid methods (Fox 1992) have been able to solve these problems with some success. These approaches make numerous perturbations to solutions while evaluating the objective function with each change, thereby allowing the system to move through the design space in search of optimal states. Since these algorithms rely heavily on random moves and statistical behavior, they need to perform numerous iterations to arrive at good results. Optimization of this sort can be costly if the evaluation of each iteration is time-consuming, as is often the case in design problems. A-Design's use of agents attempts to reduce the inefficient moves in the design space by having agents make intelligent directed decisions about how to improve designs. A stochastic influence is still present in the random selection of appropriate agents, and components.

2.6. Conceptual Design Generation and Representation

To implement a conceptual design generation tool for electro-mechanical design, comprehension of functional behavior is important. Much of the work in qualitative physics (see the overview in Forbus 1988) is devoted to producing computer models that understand how the physical world operates on a symbolic or qualitative level. While a few research projects have addressed functionality as a consequence of the geometry of interacting components (Stahovich et al. 1998), most describe components as having non-dimensional influence on one another (Hoover and Rinderle 1989, Gero 1990, Navinchandra et al. 1991, Chandrasekaran et al. 1993). The representation aspect of our work builds primarily on the work of Welch and Dixon (1994) and Schmidt and Cagan (1995), with extensions to allow for real-world components and complex component configurations.

3. A-Design Theory

The A-Design theory presented here is an iterative process that incorporates both populations of designs as well as populations of agents, the creators of the designs. In each iteration, the agents modify the population of designs, and are themselves modified based on the quality of designs they produce. In this manner, A-design captures the evolutionary nature of design, and overcomes multi-modal, non-linear, discontinuous design spaces that are typical of conceptual electro-mechanical design problems. This section describes each of the four parts of the theory. First, we present an overview of the entire iterative process (Section 3.1), followed by a detailed description of the adaptive design selection method (Section 3.2), then the representation of designs (Section 3.3), and finally the creators of the designs, the agents (Section 3.4).

3.1. Iterative Process

Figure 2 shows a flowchart of the overall process. Initially, the system accepts a starting point or description of the problem to be solved. For example, the initial specification for electro-mechanical design is a formal description of input and output behaviors. One class of agents, known as maker-agents, works directly with these input specifications to produce a population of design alternatives. These design states are then evaluated on the various objectives specified by the user. The design sorting method is capable of retaining flexibility in the design process, and allowing for robust selection of designs regardless of the number of objectives. The strategy divides the designs into three populations – Pareto optimal, good and poor – via the design selection process described in the following section. As seen in the figure, the Pareto optimal designs are copied to the next iteration, as well as sent with the good designs to

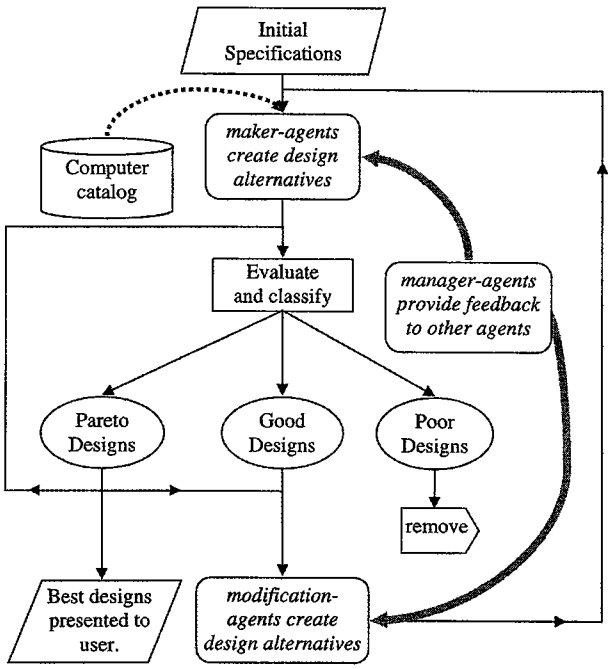


Fig. 2. General flowchart for A-Design theory.

modification-agents. This modification phase of the process involves agents that attempt to improve the solutions and effectively refine the current set of best designs. Poor designs are discarded since they have the least potential to improve future designs, and to make room for additional modified or new designs in future iterations. Throughout the process, manager-agents provide design sorting criteria, feedback to agents, and convergence criteria for the process. 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.

This methodology is intended for open-ended design problems where there is not a well-defined solution; hence, an in-depth search of the design space and adaptability to user preferences is required. Solutions to the design problem are constantly being improved upon via the selection of designs in a manner similar to that done in genetic algorithms. From a population of design alternatives, the best ones are selected to propagate into the next iteration, while the remaining ones are discarded to make room for new solutions. In genetic algorithms, this selection pressure or 'survival of the fittest' is the primary motivating factor in finding optimally directed designs. A-Design's selection pressure conservatively eliminates only the designs that would never be desired under any user preference. The user's preference, however, is used to guide the process towards solutions that best exemplify the desired trade-offs by propagating a higher concentration of designs that meet the current user preference. This selection of designs is the process' main method of searching the space, and is described in more detail in the next section.

Along with selection pressure, A-Design is also able to search for improved designs through intelligent modification of past alternatives and through feedback provided to agents. The modification-agents do not perform simple mutation and crossover operations as in genetic algorithms, but instead perform intelligent fragmenting or chunking of portions of a design based on the evaluations of the design. For example, a heavy design might have its more massive components removed in an attempt to improve the design when it is reconstructed by the maker-agents.

3.2. Design Selection

In a conceptual design problem where one is considering various component choices and configurations, there are often many goals and constraints that need to be addressed. These can include objectives of various levels of specificity, from the more general such as minimizing weight, cost and size to more specific goals such as minimizing the amount of movement at a given point, or minimizing overshoot in a transient response. Constraints present in a design description can include, for example, preventing excessive pressures in fluid lines or not exceeding maximum temperature requirements. Further, the problem description in conceptual design often changes during the design process, thereby necessitating that a conceptual design algorithm be able to handle multiple objectives in a flexible manner. To do so in A-Design, the process focuses on current user preferences to determine which objectives and constraints are most important, while also maintaining designs that exemplify other relative strengths. These alternatives act as recessive design traits in that they preserve combinations not optimal under the current user preference, but potentially so under other preferences, thus allowing A-Design to quickly respond to changes in the design environment.

The algorithm thus maintains a population of design candidates that is divided into three separate populations labeled poor designs, good designs and Pareto optimal designs, as described below. By using the idea of Pareto optimality, we can mathematically

determine which design alternatives are clearly better than others without simplifying the objectives to only a single scalar, either by a weighted sum or other methods used in utility-theory. The best design for any user preference can be found in a set of designs called the Pareto optimal set, defined as (Eschenauer et al. 1990):

$$x^* \in X \text{ is a Pareto optimal point if and only if there is no other vector } x \in X \text{ such that}$$
$$f_j(x) \leqslant f_j(x^*) \quad \forall j \in \{1, \ldots, m\} \quad \text{and} \quad f_j(x) < f_j(x^*) \; \exists j \in \{1, \ldots, m\} \tag{1}$$

In this equation, f is an objective, m is the total number of objectives, and X is the set of designs. By requiring all objectives to be optimized through minimization, it is necessary to note that no other design should exist that has more minimal values for the total of the objectives presented. This can be visually determined through plotting alternatives on a graph, with the axes representing different objectives, as in Fig. 3(a). If only two objectives are to be optimized, one can view design alternatives on a two-dimensional grid, and the Pareto optimal designs can easily be determined. These designs form a 'front' of solutions that are close to the coordinate axes. Despite the difficulty in viewing the Pareto optimal set for higher dimensions, the designs can still be found by applying the mathematical criteria stated above.

The essence of the A-Design process is to propagate this Pareto front to constantly find better solutions, as seen in Fig. 3(b). However, at any given time in the process, the user might change the relative importance of objectives. As mentioned above, the multi-objective selection is useful in handling large numbers of objectives or constraints in a more robust manner, but it is also useful in tailoring designs for specific designer needs or consumer markets. Imagine a situation in which performance and cost are two competing objectives in a conceptual design problem. A-Design might be initiated to conceptualize designs, with more importance given to minimizing cost than maximizing performance. The process concentrates its effort on this user preference, but by maintaining the complete Pareto optimal set, the system can accommodate changes to this preference. The user preference can be implemented in a variety of ways. In the current implementation, the user defines a linear weighting; however, multi-attribute decision-making (Thurston 1991), for example, could also be used.

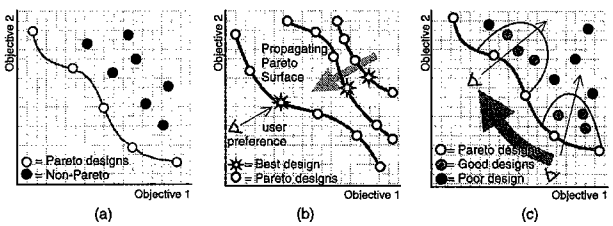


Fig. 3. (a) Two-dimensional plot of designs depicting pareto optimal set, (b) propagating Pareto surface while considering user preference, (c) bulb of good populations move with user preference.

Outside of Pareto optimal designs, the system further divides solutions into good and poor designs. At a given time, A-Design concentrates its effort on improving designs that best meet the user preference. Some designs, while not Pareto optimal, might better meet user preference than some of the outlying Pareto optimal alternatives that are preserved for extreme changes in user preference. This set of preferred but non-Pareto designs makes up the good population which, for example, can be visualized as a set of solutions located within a given radius of the intersection of current user preference and the Pareto set (Fig. 3(c)). In general, the good designs are a user-defined percentage (25% in the current implementation) of the non-Pareto designs that best meet the user preference. By concentrating on improving the set of good designs, A-Design focuses on the desired user preference. If the preference changes, then the location of the good population with respect to the Pareto optimal set will also change. The purpose of the good designs is to propagate alternatives which might not currently be optimal, but which through some modification might prove to be a successful design alternative. The good population serves to add diversity in design configurations as opposed to diversity in objective values, as is done by the Pareto optimal designs. The remaining poor designs are discarded by the system to make room for new design states in subsequent iterations.

Another important feature of the division of designs is to track the agents that created them. We increase the prevalence of agents that create Pareto and good designs so that, in future iterations design activity will produce more designs like the current Pareto and good set. Conversely, agents that produce poor designs are penalized in order to eliminate design activity that leads to inferior design states. Manager-agents control this feedback mechanism, which modifies the maker-agent and modification-agent populations; in future, more sophisticated feedback will be explored.

This division of designs into Pareto, good and poor is thus the process' method of keeping track of useful alternatives, as well as providing feedback to agents. It is this division and the propagation of the Pareto

front that provides the foundation for our approach to maintaining recessive characteristics, providing a pool of designs for possible evolution, and influencing agent selection.

3.3. Design Representation

It is necessary in any computational process to have a formal description of the artifact being manipulated, created or optimized. Traditionally, in engineering optimization, the representation is a simple vector of a fixed length, set prior to execution, containing variables representing different physical attributes of a design. The perturbation of this vector, $\mathbf{x}$, is often done either randomly or through knowledge of previous evaluations of $\mathbf{x}$. In the case of genetic algorithms, for example, the $\mathbf{x}$ is often expanded to a bit string to represent a genotype encoding. The manner in which the string is manipulated can have unpredictable or far-reaching effects in the phenotype or physical construction of the design. The unsystematic changes that can result by manipulating the design encoding make logical modification of the design state difficult.

It is for this reason that we develop a description of the design state that is most natural to modeling the actual device, and not a description which is merely beneficial for the manipulation and creation of designs through computational methods. As will be seen in Section 5.2, an electro-mechanical representation is developed to model a range of electro-mechanical devices, and does not depend upon the manner of constructing or altering designs. Allowing more freedom in constructing computational representations, be it through data structures or variable length vectors, allows design synthesis to be more general without the restrictions imposed by the construction process. The drawback of having a more general design description, however, is the need for more involved techniques for configuring designs, hence agents are used to act as a buffer between the iterative process and the representation.

3.4. Agents

The creation of any one design in A-design is due to collaboration among several different agents. These agents contain knowledge of how to design based on their individual strategies and preferences. They are constructed to understand the representation of a design state, be it complete or incomplete, and contribute in a manner that leads to successful solutions. The strategies used by these agents are based on deterministic algorithms, such as tree-searching or pattern matching. In the current implementation, agents are not autonomous, but are triggered by the system or by other agents. A simple hierarchy exists where one set of agents, known as manager-agents, invokes the operations of the other agent classes: the maker- and modification-agents. Within each agent class subclasses can exist, as well as different agent types within each subclass, and populations of agents of the same agent type. For example, the electro-mechanical A-Design approach shown in Section 5.3 has two subclasses of maker-agents, Configuration-agents (C-agents) and Instantiation-agents (I-agents); there are 32 different C-agent types and six different I-agent types, and within a given iteration of A-Design there is an average population of 3000 maker-agents in all.

The maker-agents have two responsibilities: create design alternatives based on the problem description; and re-build designs returned by the modification-agents. Construction of a complete design state is accomplished by several maker-agents, which each add their distinctive parts to make a complete state as prescribed by the design representation. The modification-agents are active at the end of the evaluation phase of the process (see Fig. 2) to take design states from the Pareto and good populations and improve them based on how they were evaluated. These agents allow the process to move from current design states to possibly better ones. As seen in Fig. 3(b), the modifications performed by these agents can significantly advance the Pareto optimal front.

While the maker- and modification-agents perform directed strategies to construct or improve designs, the manager-agents add both a stochastic and learning aspect to the A-Design system. Initially, all maker- and modification-agents have the same population, and the manager-agents randomly invoke these agents until all design tasks for the given iteration are accomplished. At the end of the iteration, the manager-agents analyze the designs created by these agents and provide feedback about how to improve the design process according to the user's preferences. Currently, a single manager agent is implemented which provides feedback in the form of increasing or decreasing the population of particular agent types, thereby forcing the agents to compete for survival by contributing to better designs.

There is some ambiguity in computational research over the proper use of the word 'agent'. Our use of agent is consistent with the definition of Russell and Norvig (1995), where agents are viewed as perceiving their environment (here, the design state) through sensors (i.e. functional inputs) and acting upon their

environment through effectors (i.e. modifications to the design state). While some might argue that without displaying specific behaviors of autonomy, mobility or sociability (Sycara 1998, Franklin and Graesser 1996), the agents in A-Design do not fully conform to the definition of 'agency'. However, within the Artificial Life community, agents often are defined as simple strategies that when combined lead to a more complex emergent behavior (Langton 1988). Similarly, our agents are defined as knowledge-based strategies for solving open-ended problems that, when cooperatively combined with other similar strategies, leads to a more complex and often emergent behavior for achieving the design goal. For example, in designing travel paths between two set destinations, one can imagine different agents to accomplish the goal. One agent might try to determine the best path for driving while another might look at possible flight paths for connecting the start and end points. The various agents would accomplish the same goal but in different manners. Because, in design, there is no single or clear-cut answer, different agents working on the same design problem can generate completely different solutions. By having agents with different abilities contributing to designs, the process gains robustness and variety in solving various conceptual design problems.

4. A-Design as Search Strategy

The following two test examples explore the effectiveness of the A-Design theory as a computational search tool. They were created to test individual sections of the theory, and are presented here to illustrate the capabilities of the methodology. The first demonstrates the operation of the multi-objective design selection and interaction of agent subsystems, while the second challenges A-Design with a numerical optimization problem. In each of these examples, the design representation and agents are quite simple. After this exploration of the operation of the general components of A-Design, a richer representation and set of agents is introduced in Section 5 for the conceptual design problem of electro-mechanical configuration design.

4.1. Manhattan Transfer

This example presents a problem for which it is relatively easy to find solutions, but difficult to find solutions that maximize the satisfaction of the traveler's preference. The object of the Manhattan Transfer problem is to get from one location to another in a grid-based city in the minimum amount of time, cost and effort. A user specifies the start and end locations of a trip as the initial specification to the algorithm. It is the algorithm's duty to find solutions that connect the start and end locations via various transportation media. A simulated two-dimensional grid of squares represents city blocks, while the transportation devices consist of bike, walk, run, bus, taxi and subway; each having different values for the cost, time and effort required. The problem specification requires A-Design to create alternatives using combinations of the six transportation devices in order to best satisfy the user's weighted criteria of minimizing cost, time and effort.

The process starts as in Fig. 2 with the maker-agents contributing partial travel paths along the way to the creation of complete trips. Rather than each agent solving the complete problem from start to end, the maker-agents subgoal on shorter trips that combine to make a complete solution. Therefore, all design states are the result of the combined effort of several maker-agents. These maker-agents differ in which transportation device they add to a trip and how they handle constraints in the system such as bus and subway stops, and maximum distances one can walk or run. After maker-agents complete designs, the designs are evaluated on their cost, time and effort, and solutions are sorted into Pareto, good and poor populations. Next, modification-agents remove elements from solutions that are believed to be preventing designs from reaching optimal states and return the fragmented designs back to the maker-agents for reconstruction. After modification, another iteration ensues until the process is finished and returns to the user a solution represented as a list of transportation devices with their start and stop locations making up the complete travel plan.

The Manhattan Transfer results are generated by specifying the beginning (0, 0) and end (20, 20) locations, as well as the relative importance of each objective. At the end of the process, A-Design returns several solutions that best meet the user preference. For example, if the user feels that minimizing cost is more important than minimizing time or effort, a solution such as the one shown in Fig. 4 is suggested. This solution was generated for a user whose preference for cost is five times more important than time and two times more important than effort. The design was found in 62 iterations with a maximum design population of 160. In addition, the user is free to change preference throughout the process, allowing the system to adapt appropriately. By examining the results produced, the user can adjust the preference weighting to achieve the desired trip. In this example,

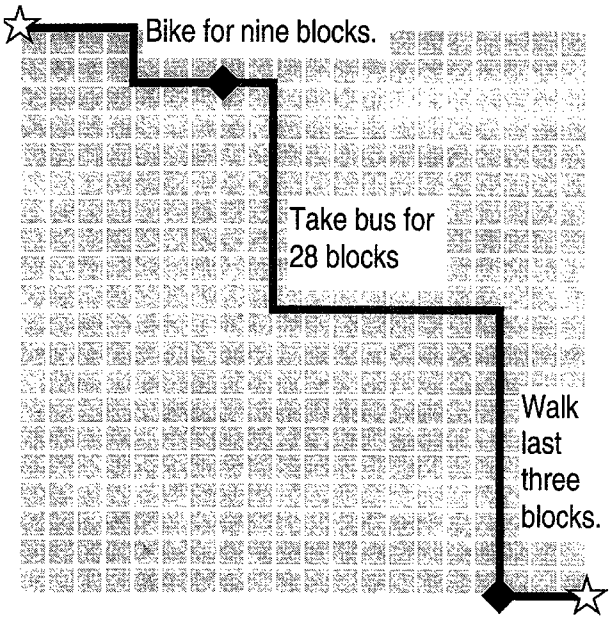


Fig. 4. Possible solution for Manhattan Transfer problem.

user preference was changed midway through the process to prefer minimizing time twice as much as effort, and five times as much as cost, thus producing the result of 'take taxi from start to finish'. By using the population of designs created under the old preference, which includes a variety of designs, the process is easily adapted to this new preference. As a result, the algorithm took only 11 iterations to converge, thus illustrating the power of the recessive traits stored in the Pareto set. It was determined through a separate experiment that the complete space of uniquely evaluatable design states numbers approximately 1.2 million, and that of these, only 99 are Pareto optimal. In comparison, A-Design found 40 Pareto points of the 99 while searching only 0.6% (approximately 7000 design states) of the design space.

Although the Manhattan Transfer example deals with a highly abstracted situation, it illustrates that agents with different characteristics are able to work together to find optimal design configurations, and their interaction produces a flexibility in adapting to changes. Alone, the agents consider minimizing only single objectives. For example, low-cost designs are usually created by agents that prefer low-cost devices. Through the collaboration of different agent types, solutions are constructed that exemplify the preferences suggested by the user. The combination of agents that prefer low-cost devices with agents that prefer low-effort devices will ideally lead to designs that are both low in cost and low in effort. In the example, the initial user preference was for low-cost designs, but when the preference changed to emphasize quick trips, A-Design was able to accommodate this alteration and switch focus to agents with a preference for quick designs. The example also demonstrates the algorithm's effectiveness in advancing the Pareto optimal front through the various iterations. Early iterations produce a set of alternatives that through modification lead to better states, which in turn add new solutions to the Pareto surface and remove previous ones that are shown to be suboptimal.

4.2. Numerical Optimization

In addition to A-Design's ability to adapt to changes in user preference is the need for A-Design to optimize the various objectives specified by the user. This example, therefore, tests the optimizing power of A-Design apart from its use as a conceptual design generation tool. In its final form, A-Design creates designs to best meet the objectives and constraints specified by a user, and is thus driven toward optimally directed configurations.

This example sets up two numerical functions to be optimized in order to maintain a multi-objective problem. The two functions shown in Fig. 5, are expressed by

$$f_1(x,y) = 300(1-x)^2 e^{-x^2-(y+1)^2} - 1000\left(\frac{x}{5} - x^3 - y^5\right)e^{-x^2-y^2} - \frac{100}{3}e^{-(x+1)^2-y^2} - |x| - |y| \quad (2)$$

where the minimum is -59.2 and is found at ($x = 0.023$, $y = -2.105$), and

$$f_2(x,y) = x^2 + y^2 - 100\sin(x) - 100\sin(y) \quad (3)$$

where the minimum is -234.6 and is found at ($x = -1.57$, $y = -1.57$).

These functions are both highly multi-modal, and therefore difficult to solve by traditional methods. A-Design was compared with a robust SQP algorithm (Lawrence et al. 1993) in finding an optimum for a weighted sum of the two objectives. Both systems were first tested by weighting f_1 ten-to-one over f_2, and then tested again with a weighting of f_2 ten-to-one over f_1 using the results of the first run as the initial specifications for the second run. The 10:1 and 1:10 preferences are used to establish quite different optimal points for the lumped objective functions. Agents within this problem are simple functions that increase or decrease parameters within the equations to reduce objective values or avoid local minima.

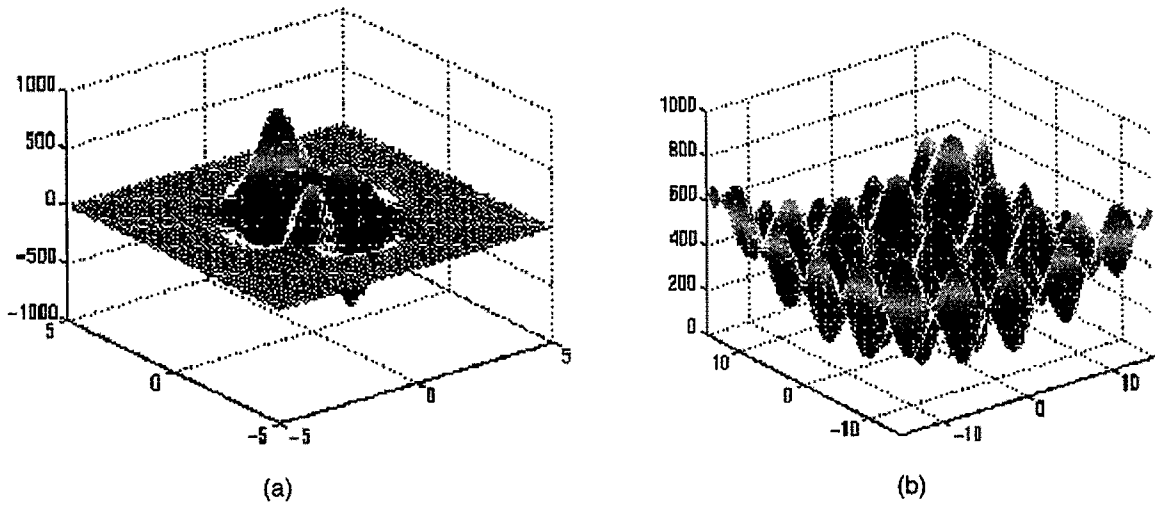


Fig. 5. Numerical optimization test functions: (a) f_1: peaks function, (b) f_2: parabola plus sine function.

A single run of A-Design proved to be a time-consuming process, partly due to the number of design state evaluations and partly due to agent and design management. Although SQP found the optimal solution quicker for the 10:1 weighting, it successfully found the global minimum only one out of seven times. The A-Design algorithm, due to its agent-based search and storing of design alternatives, found the solution in every run with a population size of 100 and an average of 88 iterations. When the weightings changed, A-Design only needed to perform a single iteration to arrive at the new optimum, while SQP had to be rerun 13 times before finding the new optimum. By retaining the Pareto optimal set from the first run, the A-Design algorithm quickly adapts to changes in weighting of the two objectives. Certainly, SQP is a more efficient optimization strategy if it starts in the neighborhood of the optimum. However, A-Design is, in general, more robust in its ability to find the optimum and the time required in arriving at Pareto optimal solutions for the first user preference results in a large savings when the process is reinitiated with different weights.

The two examples discussed above demonstrate the versatility of the A-Design methodology. Both examples illustrate the methodology's potential outside the intended use as a conceptual design tool. These examples show that A-Design, through its unique design selection scheme of preserving the Pareto front and iterative-based agent operations, can successfully produce results for a wide variety of problems.

5. Electro-Mechanical Configuration Design

5.1. Overview

Since the philosophy behind A-Design is to aid or even automate portions of the conceptual design process, it is important that we construct an example which captures the open-endedness of true design. The previous two examples simplify the design process to optimizing search, but the real design process is much more complicated than searching for a single global optimal state. Design is an ever-changing process where not only does the space of relevant designs change, but also the means in which those designs are evaluated.

This section describes a completed implementation of A-Design that creates electro-mechanical configuration designs for a general class of user-defined problems. In Fig. 6, a detailed flowchart for the electro-mechanical A-Design process is presented. Note that it is an augmentation of the general A-Design flowchart presented in Fig. 2. Initially, the user defines a design problem by the functional description of the inputs and outputs of an electro-mechanical device as the starting point for the algorithm; it is assumed that the device to be designed is fully describable by the inputs and outputs. Maker-agents are divided into two groups: configuration-agents (C-agents) and instantiation-agents (I-agents). The configuration-agents reason about how these inputs and outputs can be successfully connected to make conceptual design alternatives. These config-

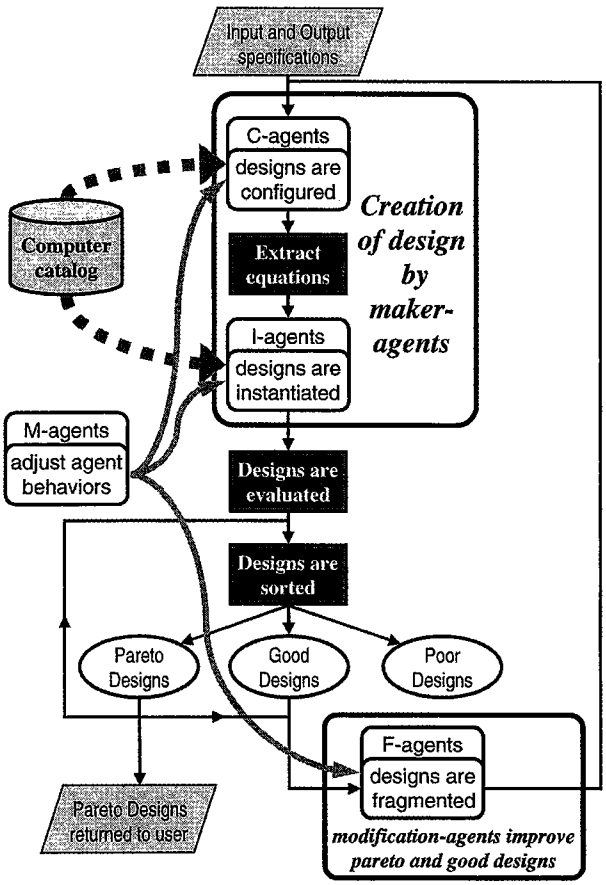


Fig. 6. Electro-mechanical A-Design flowchart.

urations consist of abstract component descriptions such as gear, spring and resistor, but do not specify exact values for parameters within the component; this method of representing designs is described in Section 5.2.

After C-agents construct designs on an abstract level, behavioral equations are constructed for each design. These equations, which describe how inputs relate to outputs, set up a method for choosing the exact values of variables within the design. By referring to these equations, instantiation-agents (I-agents) choose actual components from a computer catalog to instantiate the conceptual components. After outfitting conceptual designs with real components, the maker-agent phase is finished and alternatives are ready for evaluation.

The evaluation phase requires the calculation of all the objectives specified by the user in order to sort designs into Pareto optimal, good and poor populations, as described in Section 3.2. All Pareto optimal designs are copied to the next iteration to preserve the set of currently best alternatives in the process. Pareto designs along with the good designs are copied to the modification-agents phase of the process. This part of the process improves upon the better ideas created thus far, and attempts to use them to propagate the Pareto surface. Poor designs are thought to hold little or no design value, and are discarded to make room for new alternatives. Note that, while these designs are discarded, their potentially promising descendents in future iterations are not necessarily lost. It is possible to reach these good alternatives from other parts of the space of designs with the help of the modification process. The elimination of the poor designs simply acts to allow the process to search the most promising areas of the problem space – an important strategy on problems such as creative design tasks that are too large to allow an exhaustive search.

At this point, the process is repeated. The manager-agents (M-agents) make crucial decisions about the process. They determine how the designs are to be separated, how agents are given feedback, and when the process terminates. The process then iterates, maintaining design populations and agent populations until termination at which point solutions are returned, including a subset which best meets the user's preference.

5.2. Functional Representation

For A-Design to create electro-mechanical devices, a thorough functional representation for describing components and designs is required. This representation is tailored to describing functionality in electro-mechanical systems so that agents can produce real-world design configurations. The representation we have developed is based on qualitative physics (Forbus 1988), bond graphs (Paynter 1961, Ulrich and Seering 1989) and functional block diagrams (Pahl and Beitz 1988), and more specifically on work done by Welch and Dixon (1994) and Schmidt and Cagan (1995). A summary of the representation is given in this section.

The fundamental issue in creating a representation for describing a design configuration is developing a formal method for how individual components behave and, further, how the connected sum of components behave. In our representation, components are described by their ports, or points of connectivity with other components. Information about how components are constrained at their ports, how energy and signals are transformed between ports, and how energy variables within the system relate to others throughout the design is found in the component descriptions. These descriptions, known as embodiments, also contain information about the

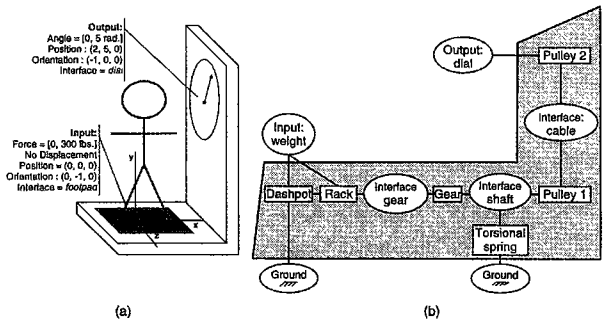


Fig. 7. Weighing machine: (a) User-defined functional specifications, (b) possible configuration of embodiments that meet the weighing machine specifications.

kinds of parameters that describe the component such as length, spring constant, cost and efficiency. Although these descriptions embody the functionality of components, they do not contain actual values for the variables within a component. These variables are replaced with real values when a real-world component from the catalog is used to instantiate an embodiment.

The user describes the design problem to the A-Design system by supplying information about the desired inputs and outputs of the device to be constructed. In the test example shown in Fig. 7(a), the input is described as a downward force supplied by the user (a specimen's weight) and the output is an angular displacement of a dial. With the knowledge of the system's inputs and outputs, the configuration-agents (C-agents) are able to construct design configurations by individually adding one embodiment at a time to the configuration to create completed conceptual design alternatives like that shown in Fig. 7(b).

The basic design configurations are then further processed to determine the equations that describe how inputs in the system relate to outputs. These equations are derived from information stored in the embodiments that describe how the components operate and interact with one another. The equations are then used to determine the best values for variables within the system. A computer catalog of actual components includes data on the possible values that an embodiment variable can possess. For example, the gear embodiment in Fig. 7(b) contains a variable radius (r_{gear}) which appears in the equation describing the relation of input and output in the design:

$$\theta_{dial} = \frac{r_{gear}\, r_{pulley1}}{k_{spring}\, r_{pulley2}} F_{weight} \tag{4}$$

In this equation, the r variables represent the radii of the gear and pulleys, k_{spring} is the stiffness of the spring, F_{weight} is the system input, and θ_{dial} is the system output. With this equation and the descriptions of gears, pulleys and torsional springs in the component catalog, a range can be found to determine what gear sizes can fulfil the design requirement of 0 to 300 lb. input, and 0 to 5 radians output. At this stage, instantiation-agents (I-agents) can instantiate designs with these components from the component catalog, thus producing feasible design alternatives. Further details on the representation and approach to equation extraction can be found in Campbell et al (1998b).

5.3. Agent Framework

Within electro-mechanical design, agents are divided into four categories: configuration-agents (C-agents), instantiation-agents (I-agents); fragment-agents (F-agents); and manager-agents (M-agents). While future agent types are planned, the maker-agents are currently composed of the configuration-agents and instantiation-agents while the modification-agents are comprised of the fragment-agents. All agents are implemented as independent functions that accept designs or partial design states, and return their modifications, additions or possible recommendations. Throughout the iterative process and interaction of agent types, the manager agents make decisions about which designs are better, and how the agents should be grouped, penalized or rewarded for their design contributions.

5.3.1. Maker-Agents

Configuration-Agents (C-AGENTS): configuration-agents are by far the most involved and interesting of the agent types. They have the ability to take the user-defined inputs and outputs and choose components to fulfil the functionality of the design problem. In doing so, C-agents subgoal on the large problem, determine what portion of the system they will address, and find an embodiment that most fulfils this functionality. For this reason, their operation is closely linked to the functional representation described in Section 5.2. The embodiments present within a given design are the result of the various C-agent behaviors.

Initially, a C-agent examines the partial design state it is given and determines where to attach a new embodiment. In doing so, a C-agent can add new embodiments in series or parallel, and from input towards output or from output towards input within a configuration. The C-agents subgoal on the desired inputs and outputs specified by the user and break

Table 2. Representation of current set of agents

C-Agents (32 agents)	I-Agents (6 agents)	F-Agents (32 agents)
Domain preference	**Objective preference**	**Design preference**
electrical EB's	prefer inexpensive components	modify expensive designs
translational EB's	prefer lightweight components	modify inexpensive designs
rotational EB's	prefer efficient components	modify heavy designs
hydraulic EB's		modify light designs
	Variable preference	
Source vs. sink	select component based on *variables* present in behavioral equations	**Objective preference**
connect to source FP's		remove expensive components
connect to sink FP's	select component based on *constraints* present in FP	remove heavy components
Parallel vs. serial		**Degree of fragmentation**
connect EB's in parallel		remove component from design
connect in series		remove doubled components from design
		remove EB from design
Other EB connections		remove dangling EB's from design
link new EB to existing FP's		
link new EB to new ground FP		

them down into simpler problems in order to make a decision about which embodiment to add to the design. For example, C-agent will start by trying to find an embodiment that satisfies all the goal requirements of the input and output ports; when this cannot be accomplished, they try to find intermediate embodiments by diminishing the requirement until a satisfactory EB is found. With different methods of placing the embodiment within a design, choosing embodiments from a catalog, and subgoaling on the larger design problem, 32 different C-agent types exist in the electro-mechanical A-Design system. As seen in Table 2, where each box has several different strategies or preferences for a specific decision, the selection of one item from each box produces 32 distinct C-agent types.

The specifics explained here are for the current set of C-agents. They are by no means the only way to implement such configuration-agents, and are shown to exemplify the amount of programmed intelligence that can successfully lead to interesting design states. It is important that agents are directed enough to prevent infeasible design alternatives, but it is equally important that they are not so directed as to unduly limit the exploration of the design space. The iterative nature of A-Design relieves C-agents from having to produce viable alternatives at the start. The space of possible solutions is searched by C-agents producing possibly extravagant alternatives at first, and then improving the more successful alternatives that are carried over in the next iteration. The interplay of goal-directed agents and random search is a major factor in allowing the A-Design process to solve difficult conceptual design problems.

Instantiation-Agents (I-AGENTS): the I-agents have a simpler job than the C-agents. They take the equations describing the behavior of a design and determine which actual components best meet the design specifications. Component selection is performed by referencing a catalog of real components that exists for each embodiment in the process. For example, in Fig. 7(b), a gear embodiment is instantiated by gear components that contain data on the radius of the gear (r_{gear}) which is used in solving the behavioral equation (Eq. (4)), as well as data on other relevant details such as the pitch of gear teeth, and the cost and weight of the gear.

I-agents rely on their various preferences to choose a component. For example, some I-agents might prefer cheaper components, while others might prefer lightweight components. This propensity for a certain objective, along with the selection of embodiments to instantiate, results in a variety of I-agent types. Through the different I-agent characteristics, different instantiated designs can result from a single conceptual design produced by the C-agents. Again, the strategies shown in Table 2 are not the only possible implementations; they were created in this manner to reflect the theory's desired combination of goal-directed knowledge-based agent strategies for selecting components and a stochastic search process.

5.3.2. Modification-Agents

Fragment-Agents (F-AGENTS): the designs that are brought to the modification stage of the process are fragmented by agents attempting to improve the current design states. The fragment agents are crucial in driving the system towards optimal design states. F-agents individually choose a design to be modified, and remove embodiments and/or components from the design that are believed to be reducing the design's worth. Various fragmenting tactics differentiate F-agent types from one another. Predominantly, these agents differ from each other in their choice of objective to address; for example, some F-agents remove costly components, while others remove inefficient or heavy components. As seen in Table 2, F-agents also differ in selecting which design or designs to modify, and the degree of modification that is performed. The fragmented designs created by the F-agents are returned to the C-agents and I-agents in the next iteration for reconstruction.

Future extensions will include other types of modification-agents, such as agents that exchange components, agents that merge designs, or agents that attempt to create function sharing in designs.

5.3.3. Manager-Agents (M-AGENTS)

The populations of both designs and agents are maintained by the manager agents. M-agents are responsible for providing feedback to the other agents, as well as controlling basic parameters within the system. By examining Pareto and good design populations, M-agents adjust agent populations and design populations to accommodate changes in user preference.

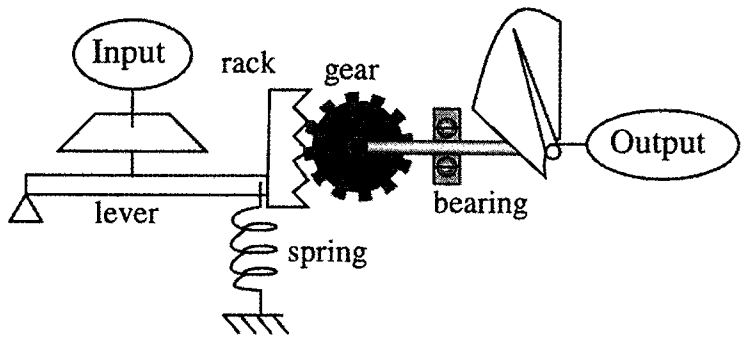
The range of possible M-agent behaviors is large and intricate, and lends itself to future research, however current simplified strategies of providing feedback have produced promising results. As currently formulated, the single M-agent type reinforces the number of agents of a given agent-type that have contributed to past Pareto and good designs, while de-emphasizing agent-types that have produced poor solutions. This reinforcement makes the process more efficient in subsequent iterations by reducing the random and unproductive search produced by ineffective agents, and focuses on agent strategies that have proven useful on the given design problem. Future work on manager-agents include agents that suggest useful combinations of components via chunking (Laird et al. 1986), or more sophisticated feedback mechanisms such as grouping agents that have produced radically different alternatives together in hopes of combining beneficial ideas.

6. Test Results for Electro-Mechanical Design Example

We now turn to an example of A-Design designing a complete electro-mechanical configuration. The electro-mechanical design test problem shown in Fig. 7(a) of fabricating weighing machine alternatives has produced some interesting results. The implemented electro-mechanical A-Design system includes the functional representational system, the C-, I-, F-agents, one M-agent strategy, the equation extractor, the design evaluation mechanism, design sorter and a general framework for transferring designs to the various sub-processes. The system is written in LISP and runs on a Silicon Graphics Indigo 2. To pose the design problem, the user supplies the desired inputs and outputs, and objectives. In this case, we have chosen to optimize four objectives: minimize cost, minimize mass, minimize dial error, and minimize input displacement. Note that, while the first two objectives (minimize cost, and minimize mass) are calculated by summing data provided within the catalog on each of the components used in the design, the latter two (minimize dial error, and minimize input displacement) are results of the values of the components used and the behavior predicted by the behavioral equations. The problem is therefore described to the process by the functional description, as well as the metrics for the objectives in the problem. The catalog of components for this test example consists of the various embodiments shown in Table 3. For each of the 32 embodiments shown, there currently exist actual components drawn from

Table 3. Current embodiments implemented in the system

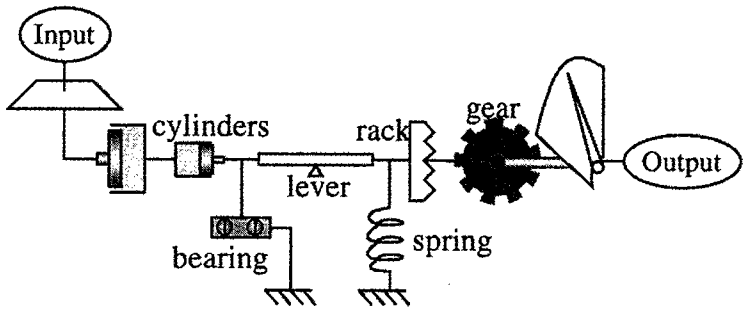
battery	cable	capacitor	electrical valve
gear	inductor coil	lever (class 1)	lever (class 2)
lever (class 3)	motor	pipe	piston
potentiometer	pulley	rack	relay
resistor	rotational bearing	rotational damper	rotational valve
shaft	solenoid	spring	sprocket
stopper	switch	tank	torsional spring
transistor	translational bearing	translational damper	worm gear



(a)

Components:
lever: 5 cm bar stock w=1.0", t=0.25"
spring: ERS-A1-36 $0.93, K=16.0lb/in
rack: KHS-F2-142 $26.75, pitch=64
gear: LAS-F7-28 $5.75, 28 teeth
shaft: AAS-A8-20
bearing: ABS-A2-19

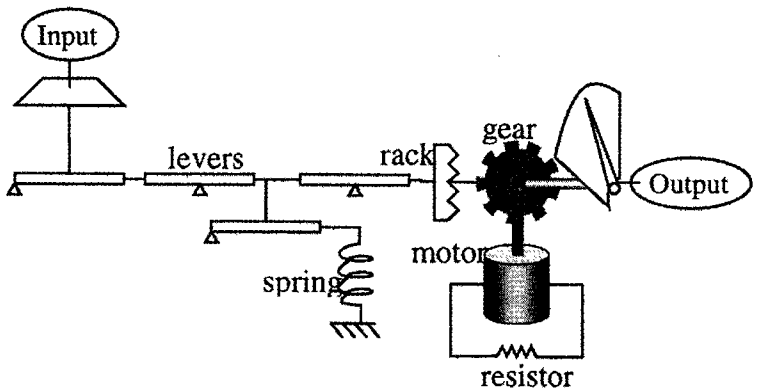
Design objectives:
cost = $46.82, **mass** = 0.2kg,
input dx = 4.1mm, **accuracy** = 0.4 rad.



(b)

Components:
cylinder-1: 62205K77 $368.49,Dia=3.25"
cylinder-2: 62205K71 $198.63, Dia=1.5"
linear-bearing: ABS-L1-4 $10.47, Dia=0.25"
lever: 10 cm bar stock w=1.0", t=0.25"
spring: ERS-A1-2 $0.89, K=2.0lb/in
rack: KHS-F2-142 $26.75, pitch=64
gear: LAS-F7-28 $5.75, 28 teeth

Design objectives:
cost = $616.18, **mass** = 1.3kg,
input dx = 0.5mm, **accuracy** = 0.4 rad.



(c)

Components:
lever-1: 4 cm bar stock w=1.0", t=0.25"
lever-2: 4 cm bar stock w=1.0", t=0.25"
lever-3: 13 cm bar stock w=1.0", t=0.25"
lever-4: 7 cm bar stock w=1.0", t=0.25"
spring: ERS-A1-7 $0.78, K=14.6lb/in
rack: KHS-F2-142 $26.75, pitch=64
gear: LAS-F7-128 $12.03, 128 teeth
motor: 542-0130 $34.19, 300prm
resistor:297-7751 $0.01, 180Kž

Design objectives:
cost = $90.20, **mass** = 0.5kg,
input dx = 0.7mm, **accuracy** = 0.2 rad.

Fig. 8. Three different alternatives created by the A-Design process. Design (a) is found by an equal preference for the four design objectives, whereas designs (b) and (c) are found by placing more importance on minimizing input displacement.

Allied Electronics, Nordex Inc., and Mc-Master Carr Supply catalogs totaling just over 300 components available for constructing designs. The computer representation for each element in the catalog contains both embodiments and the actual components used to instantiate the embodiments. The catalog is expandable to any number of embodiments and components allowing the user to introduce new components.

Figure 8 shows three weighing machines created by the process over 30 iterations with a maximum population of 100 designs.[1] The first solution (Fig. 8(a)) was created when the user preference was for low-cost low-weight designs, and as a result the solution is relatively inexpensive and lightweight compared to the other two solutions, but it contains a bit too much movement at the input (4.1 cm). These deficiencies are due to the simple configuration established by the C-agents, as well as the components chosen by the I-agents that had emphasized low-cost and low-weight over parameter selection like proper gear radius and spring stiffness. The system was then instructed to place more emphasis on minimizing movement at the input. As a result of this preference shift, designs were created that although more costly and more massive, addressed the issue of minimizing input displacement. The second and third designs (Figs 8(b), (c)) show two different ways in which the system was able to solve this problem. Figure 8(b) uses two hydraulic cylinders with different diameters, while Fig. 8(c) utilizes a series of levers to minimize input. These examples demonstrate the richness of alternatives that are created by the A-Design algorithm in solving a non-trivial design problem. The challenge of having an accurate dial with little displacement at the input forces agents to constantly find better configurations as well as the combinations of components to fulfil those configurations. Earlier results for the same problem can be found in Fig. 1, and are discussed in Campbell et al (1998a).

7. Experimental Testing

The claim is made that A-Design is an effective mechanism for searching design spaces and for adapting to changing market demands or user preferences. To test these claims, a series of tests were performed to analyze various performance aspects of the A-Design system applied to the electro-mechanical design problem. These experiments, which consisted of running the algorithm multiple times and gathering statistics, tested the importance of the Pareto population, the importance of the good population, and the adaptability of the process.

One concern is whether the stochastic nature of A-Design would allow too much randomness in the results to prevent a proper comparison of different runs. Thus, the first test that was performed involves determining what effect the starting points or initial set of designs has on the final results of the process. To test this, a comparison was made of 20 runs all with the same initial population to 20 runs with random initial populations. Each run consisted of 60 iterations of the A-Design process using a population of 100 designs. The use of 20 runs is intended to ensure that no idiosyncratic or outlying runs are presented. As shown in Fig. 9(a), these two sets of runs are compared by plotting the best design at each iteration for a given user preference. The dark solid lines all contain the same initial population, and therefore all have the same best design at the first iteration, while the gray dashed lines have very different best designs in the first iterations. Notice that although there is much fluctuation in the beginning of the processes, all runs converge to a fairly uniform value. Figure 9(b) clarifies the results shown in Fig. 9(a) by averaging the runs to a single line. Here the solid black line and the dashed gray line are very similar at the end of the process, demonstrating that the there is little difference in the data produced through random starting points as opposed to set starting points. The pruning of designs and the execution of guided agents quickly eliminates the effect of starting at different points in the design space. Therefore, in the following tests it is assumed that the use of random starting points has little effect on the statistical results.

The theory is based strongly on the design selection methodology described in Section 3.2, where designs are divided into Pareto, good and poor populations. To observe the effects of this division, we isolated the good and Pareto populations in separate runs and compared their ability to optimize the user's objectives. Figure 10 shows the best designs at each iteration averaged over 20 runs for each case, as was done in Fig. 9(b), where the best design's evaluation is plotted with respect to a given user preference. Note that with just the good population in operation, the best design clearly has a higher, and therefore inferior, lumped evaluation value than the runs with a Pareto population. However, when comparing the final design from the 'Pareto population only' run with the 'both good and Pareto' run, only small differences are found; both processes find a similarly minimal value for their best design. From this experiment we can draw the conclusion that the Pareto population contributes significantly to the process' ability to find optimally directed solutions. This is a surprisingly useful conclusion to draw since we preserved the Pareto population to model adaptability of changing user preferences. However, this conclusion shows the importance of saving

[1] Figure 8 illustrations are the authors' renditions of the appearance of the embodiments and instantiations generated by the system.

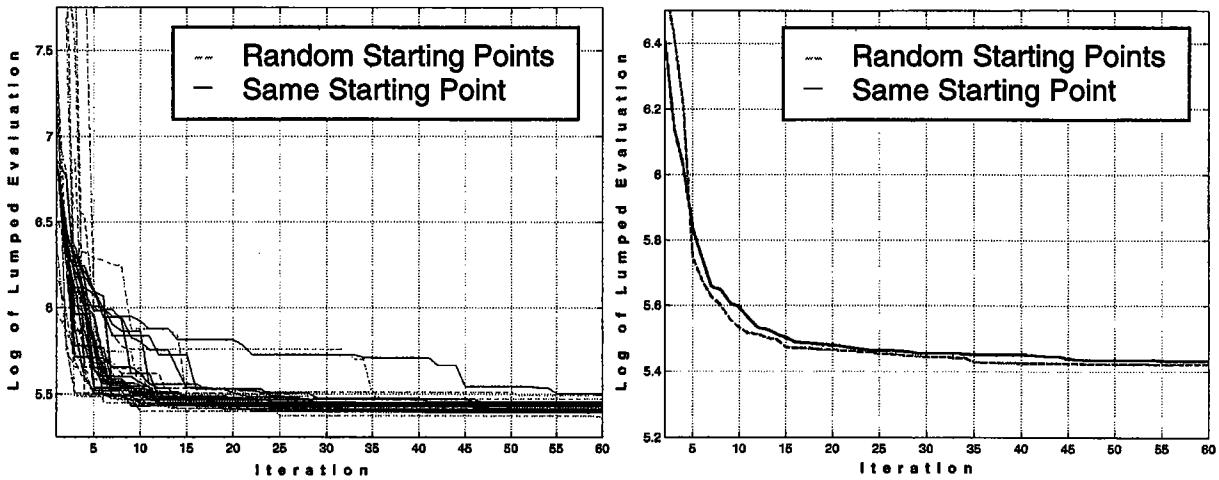


Fig. 9. Comparison of random vs. set initial design populations (a) Best design at each iteration for 20 runs, (b) average of best design from 20 runs.

diverse solutions, even in the exploration of a predetermined design problem. It is believed that preserving diverse design states allows the system better coverage of the search space of possible designs.

From the experiment shown in Fig. 10, one could question the importance of the good population in finding improved design states. Its inclusion in the 'both Pareto and good' run leads to slightly better solutions over the Pareto only run, but not a substantial enough amount to validate the theory. There is, however, a substantially quicker convergence of the best design in the 'both Pareto and good' compared to the 'Pareto population only' run seen in the earlier part of the process. Since the algorithm is run for an arbitrary number of iterations, as decided by the user, quicker convergence may produce better

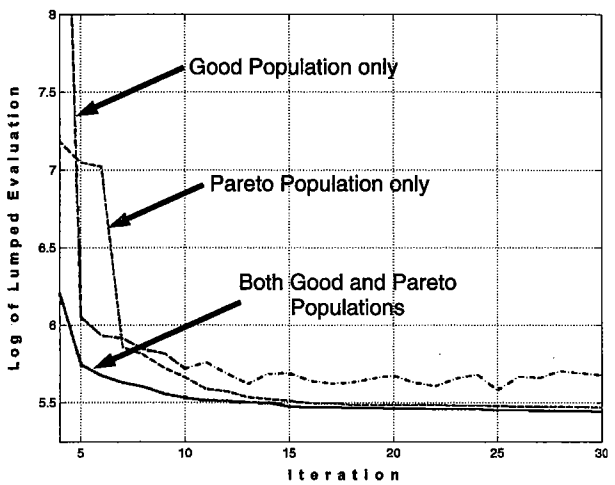


Fig. 10. Comparison of best designs from Pareto only, good only and both.

design states if the user stops the algorithm prematurely. As will be seen in the next experiment, the good population is responsible for this quick convergence, as well as playing a role in adapting to a change in user preference.

The next experiment tests the adaptive capabilities of A-Design by observing the effect of switching the user preference at the 30th iteration (of 60 iterations). Both Pareto and good populations are in operation, and when the user preference changes, a new location for the good population is defined with respect to the Pareto surface as is seen in Fig. 3(c). Again, the results shown Fig. 11 are created through averaging over 20 separate runs of the A-Design process and plotting the best design for a given user preference. In this experiment, three separate runs are compared. Figure 11(a) graphically depicts the various runs plotted in Figs 11(b) and 11(c). These plots are depicting how the process reacts to a switch in preference from W1, which is a preference favoring low-cost and low-weight designs (the same preference used to produce the weighing machine in Fig. 8(a) to W2, which favors minimizing input displacement (Figs 8(b) and 8(c)). Figure 11(a) shows five sub-procedures within the experiment; sub-procedure **E** is the focus of this experiment while **A, B, C** and **D** are the control for the experiment. Sub-procedures **A** and **B** are created under a constant user preference, W2, while sub-procedures **C** and **D** together represent the system under constant user preference of W1. After 30 iterations under preference W1 (**C**), the process is stopped and sub-procedures **D** and **E** are initiated with the agent and design data last achieved in sub-procedure **C**.

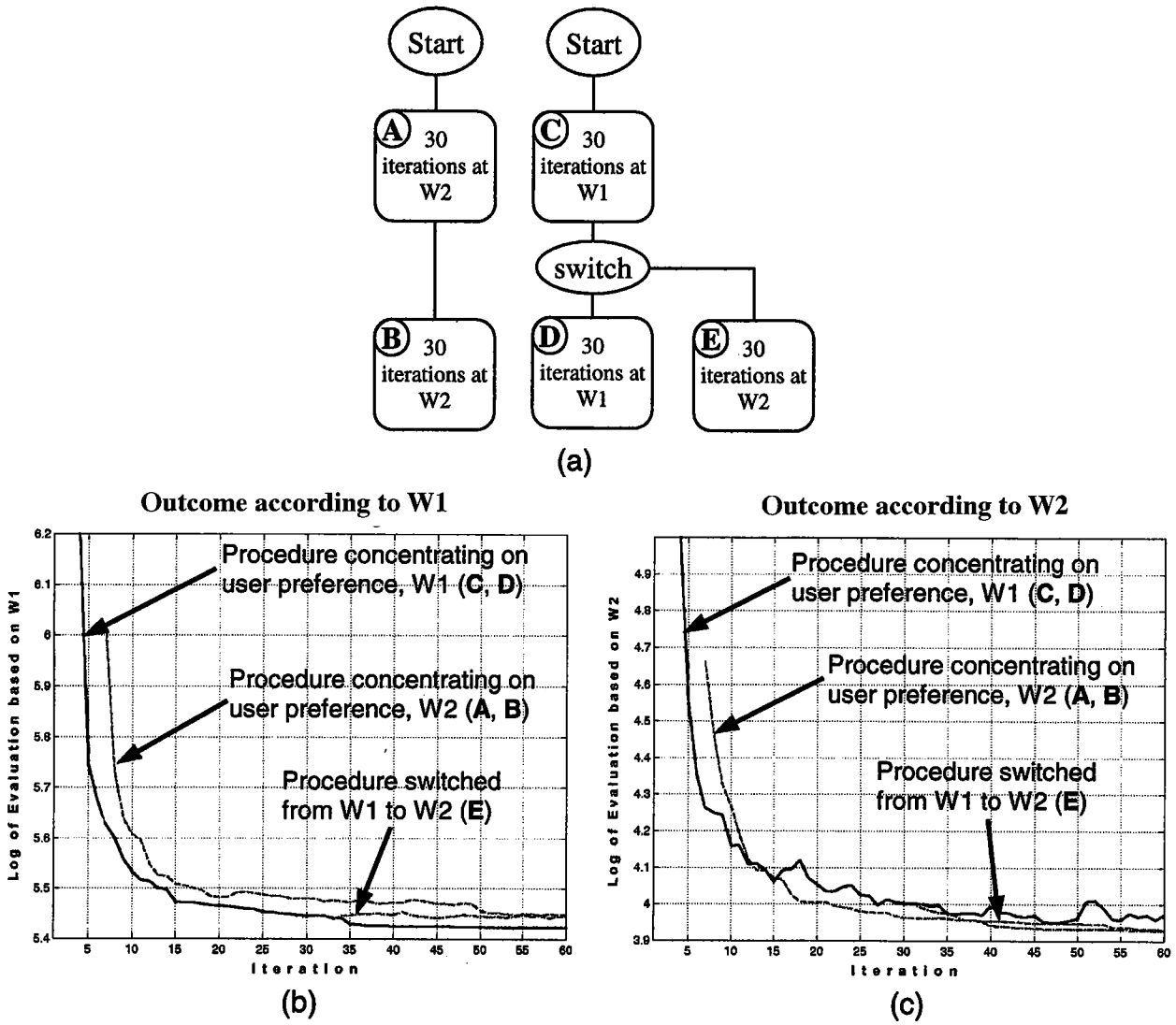


Fig. 11. Adaptability of A-Design: (a) Flowchart of adaptability experiment, (b) best designs viewed by W1, (c) best designs viewed by W2.

To view the results, all five sub-procedures are plotted as viewed under W1 preference (Fig. 11(b)) and under W2 preference (Fig. 11(c)). Therefore, by comparing the best lumped evaluation under these preferences, we can determine how the process responds to optimizing the objectives under these preferences. It is important to note that while all five sub-procedures are plotted in both Figs 11(b) and 11(c), each procedure concentrates on only one user preference. Therefore, the plots are showing some procedures in a different preference than that governing the creation of the designs. For example, **A** is created under user preference W2, despite the fact that in Fig. 11(b) the best designs under W1 are plotted. It is for this reason that **C** and **D** have better values in Fig. 11(b), while **A** and **B** have better values in Fig. 11(c).

The switch in preference (**C** to **E**) occurs at the thirtieth iteration, at which point the process stops optimizing designs for W1 and the good population shifts focus to user preference W2. Although viewing sub-procedure **E** in Fig. 11(b) appears to offer no improvement after the switch, there is a significant gain shown when viewed from the perspective of W2 (Fig. 11(c)), as is expected, since the process is now optimizing under W2. The switched run, **E**, almost completely recovers from the time spent developing designs for W1 as seen when comparing **E** with **A** and **B**, where the concentration was constant on W2 for a full 60 iterations.

Due to the Pareto-based conservation of diverse designs, the process was able to accommodate the switch to the W2 preference after spending half of the iterations concentrating on a completely different user preference. While it is believed that the recessive characteristics in the Pareto set allow the system to start at an advanced stage in the search process, instead of starting from undeveloped design states, this experiment has shown that the good population allows the process to concentrate on what is currently desired by the user. In this experiment, the use of the good population in conjunction with the Pareto population has shown to be a successful way to handle changing user preference. The good population provides focus for the process in searching under the current user preference, while the Pareto population allows flexibility if the preferences should change.

8. Discussion and Concluding Remarks

This paper has introduced a new design generation theory known as A-Design that, through its implementation, has shown that computer tools can adapt to change as well as play a part in the conceptual phase of the design process. A-Design is an agent-based and adaptive strategy for performing conceptual engineering design. The methodology has four distinct subsystems: an agent architecture; a multi-objective design selection scheme; a functional representation for electro-mechanical systems; and an iterative-based algorithm for evolving optimally directed design states. According to the theory, each of these subsystems plays an important role in enabling the production of creative and adaptive designs. The A-Design theory presented here provides a foundation for computer tools that can be developed to aid industries in meeting fast design cycles, in choosing ideal components from the overwhelming number of suppliers, and in meeting the variety of consumer demands.

Other search strategies or optimization techniques that solve problems through numerous permutations often require the existence of a well-defined set of variables prior to execution. However, defining these variables is often part of the conceptual design problem at hand. Knowledge-based techniques can often overcome problems in ill-defined conceptual design, but these approaches often do not explore the vast space of conceptual designs, as is done in stochastic search strategies. In fact, the conceptual design space is larger than the somewhat constrained parameter optimization space addressed by stochastic strategies, suggesting that a mechanism for generating conceptual design explores as many alternatives as possible to find successful design alternatives. It is for this reason that the A-Design approach presented here combines aspects of both stochastic optimization and knowledge-based design strategies. The knowledge-based strategy is contained within software agents which interact within an iterative algorithm to search the design space in a stochastically guided manner for solutions that best meet user specifications.

Initial test examples help to demonstrate the effectiveness of the algorithm's adaptability and search success. The Manhattan Transfer test example finds solutions for a specific user preference, and can quickly adjust when a change in preference occurs. The numerical optimization test example pits the A-Design theory against a traditional SQP algorithm, and although the SQP method runs faster than A-Design, it often gets stuck in local optima, while A-design is able to more consistently find the global optimum. Agent interaction and Pareto design selection in A-Design allow solutions to be searched in the non-monotonic and multi-modal spaces of the test examples. For a conceptual design problem, the A-Design system has also been experimentally shown to produce successful designs, as demonstrated by the weighing machine example. Furthermore, experiments with this problem have validated the theory's separation of designs into Pareto, good and poor subsets as an effective way to both optimize objectives and retain flexibility. The 'recessive' characteristics preserved in the array of alternative designs provide adaptiveness to changing conditions.

The electro-mechanical design system is capable of not only producing real designs, but of producing a diversity of solutions that are representable and configurable by the functional representation scheme and agent architecture. The agents of the A-Design system contain very little information specific to electro-mechanical design. Agents are based on simple reasoning strategies such as tree search and pattern matching. To keep agents as general as possible for a wide variety of design problems, the knowledge of how components influence one another in a design has been placed in the embodiment representations. This generality allows for a variety of embodiments such as multi-input/multi-output components, electrical components, as well as characteristically different mechanical components from the rotational, hydraulic and translational domains.

The power of A-Design's generality is due to the iterative approach to design generation. Since functionality of a given device is specified in part by the objectives that define the design problem,

generated solutions can be compared on a common metric of how well they satisfy the design specifications. Initially, agents design relatively poor solutions, but as the design selection scheme isolates better alternatives and feedback is provided to the agents, the process improves designs to best meet the user's defined functionality. For example in the weighing machine problem, accuracy of dial is an objective that partially defines the functionality of the device. Initially, agents may create solutions that do not even cause the output dial to rotate. As the process unfolds, solutions appear that better meet the accuracy objective, and feasible weighing machines are created. Future work with the electro-mechanical A-Design approach will accommodate dynamic analyses to allow the user to better describe a design problem through desired system dynamics, such as the settling time of the dial in the weighing machine example.

The premise of this work is that design takes place within a dynamic environment. Thus far, we have focused on changing user preferences for the various objectives describing a design problem. In current work, A-Design's adaptability is being tested in cases where objectives not only change in importance, but are also added or removed from the design problem. This enables the addition and deletion of design constraints when modeled as penalty functions and treated as objectives in the optimization process. Finally, our model can also readily incorporate changing component embodiments; as new technologies emerge or older ones are retired, components can be readily added and removed from the catalog. Future work will explore these and other means of modeling the changing environment in which A-Design is able to create useful and innovative design concepts.

Acknowledgements

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 financial support of the National Science Foundation under grant EID-9256665.

References

Balachandran M, Gero JS (1984) A comparison of three nethods for generating the Pareto optimal set. Engineering Optimization, 7:319–336.

Campbell M, Cagan J, Kotovsky K (1998a) A-Design: Theory and implementation of an Adaptive, Agent-based method of conceptual design. In JS Gero, F Sudweeks (eds), Artificial Intelligence in Design '98, Lisbon, July 20–23, 579–598.

Campbell M, Cagan J, Kotovsky K (1998b) Agent-based synthesis of electro-mechanical design configurations. Proceedings 1998 ASME Design Engineering Technical Conferences and Computers in Engineering Conference: Design Theory and Methodology Conference, DETC98/ DTM-5673, Atlanta, GA, September 13–16

Chandrasekaran B, Goel AK, Iwasaki Y (1993), Functional representation as design rationale. Computer 26:48–56.

D'Ambrosio JG, Birmingham WP (1995) Preference-directed design. AI EDAM, 9:219–230

Dong A, Agogino AM (1995) A spectral optimization algorithm for multi-objective prototype selection. DE-Vol. 83, Proceedings of the ASME Design Engineering Technical Conferences 2:447–453

Eschenauer H, Koski J, Osyczka A (eds) (1990) Multicriteria Design Optimization. Springer-Verlag, Berlin, Germany

Fonseca CM, Fleming PJ (1995) An overview of evolutionary algorithms in multiobjective optimization Evolutionary Computation 3:1–16

Forbus KD (1988) Qualitative physics: Past, present, and future. In: H. Shrobe (ed), Exploring Artificial Intelligence. Morgan Kaufmann, 239–296

Fox BL (1992) Uniting probabilistic methods for optimization. In JJ Swain, D Goldsman, RC Crain, JR Wilson (eds), Proceedings 1992 Winter Simulation Conference, 500–505

Franklin S, Graesser A (1997) Is it an agent, or just a program?: A taxonomy for autonomous agents. Proceedings Third International Workshop on Agent Theories, Architectures, and Languages: Intelligent Agents III. Springer-Verlag, 21–35

Gage PJ, Kroo IM (1995) Representation issues for design topological optimization by genetic methods. In: GF Forsyth, MAI: (eds), Industrial and Engineering Applications of Artificial Intelligence and Expert Systems: Proceedings Eighth International Conference, Melbourne, Australia. June 6–8, 383–388

Glover F (1989) Tabu Search-Part 1. ORSA Journal on Computing 1(3):190–206

Goldberg DE (1989) Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading, MA

Goldstein D. (1994) An agent-based architecture for concurrent engineering. Concurrent Engineering: Research and Applications 2:117–123.

Grecu DL, Brown DC (1996) Design agents that learn. AI EDAM 10:149–150

Holland JH (1992) Adaptation in Natural and Artificial Systems. MIT Press, Cambridge, MA

Hoover SP, Rinderle JR (1989) A synthesis strategy for

mechanical devices. Research in Engineering Design 1:87–103

Kirkpatrick S, Gelatt Jr. CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220:671-679.

Koza JR, Bennett III FH, Andre D (1996) Automated design of both the topology and sizing of analog electrical circuits using genetic programming. In: JS Gero, F Sudweeks (eds). Artificial Intelligence in Design 151–170.

Lande SE (1997) Issues in multiagent design systems. IEEE Expert 12:18–26

Langton CG (ed) (1988) Artificial Life. Addison-Wesley, Reading, MA

Lawrence CT, Zhou JL, Tits AL (1993) CFSQP: C-implemented Functional Sequential Quadratic Programming, University of Maryland, Institute for Systems Research, College Park, MD

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

Murthy SS (1992) Synergy in Cooperating Agents: Design Manipulators from Task Specifications, PhD Dissertation, Carnegie Mellon University

Navinchandra D, Sycara KP, Narasimhan S (1991) A transformational approach to case-based synthesis. AI EDAM 5:31–45

Pahl G, Beitz W (1988) Engineering Design – A Systematic Approach. Springer-Verlag, New York

Paynter HM (1961) Analysis and Design of Engineering Systems. MIT Press, Cambridge, MA

Petrie CP, Webster TA, Cutkosky MP (1995) Using Pareto optimality to coordinate distributed agents. AI EDAM 9:269–281

Quadrel R, Woodbury R, Fenves S, Talukdar S (1993) Controlling asynchronous team design environments with simulated annealing. Research in Engineering Design 5(2):88–104

Queipo N, Devarakonda R, Humphrey JAC (1994) Genetic algorithms for thermosciences research: application to the optimized cooling of electronic components. International Journal Heat and Mass Transfer 37:893–908

Russell S, Norvig P (1995) Artificial Intelligence: A Modern Approach. Prentice-Hall, Englewood Cliffs, NJ

Schaffer JD (1985) Multiple objective optimization with vector evaluated genetic algorithms. In: JJ Grefensette (ed), Genetic Algorithms and their Applications: Proceedings First International Conference on Genetic Algorithms, Lawrence Erlbaum, 93–100

Schmidt LC, Cagan J (1995) Recursive annealing: a computational model for machine design. Research in Engineering Design 7:102–125

Stahovich TF, Davis R, Shrobe H (1998) Generating multiple new designs from a sketch. Artificial Intelligence 104:211–264

Talukdar S (1993) Asynchronous teams. Fourth International Symposium on Expert Systems Applications to Power Systems, La Trobe University, Australia, January 4–8

Talukdar S, Baerentzen L, Gove A, de Souza P (1996) Asynchronous teams: organizations for algorithmic computation. EDRC Tech-Report 18-56-96, EDRC Carnegie Mellon University, Pittsburgh, PA

Thurston DL (1991) A formal method for subjective design evaluations with multiple attributes. Research in Engineering Design 3:105–122

Ulrich K, Seering W (1989) Synthesis of schematic descriptions in mechanical design. Research in Engineering Design 1:3–18

Welch RV, Dixon J (1994) Guiding conceptual design through behavioral reasoning. Research in Engineering Design 6:169–188