Agent-Based Synthesis of Electromechanical Design Configurations

Matthew I. Campbell
Jonathan Cagan
Department of Mechanical Engineering
Carnegie Mellon University
Pittsburgh, PA 15213

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

A new automated approach to engineering design known as A-design is presented that creates design configurations through the interaction of software agents. By combining unique problem solving strategies, these agents are able to generate solutions to open-ended design problems. The A-design methodology makes several theoretical claims through its combination of multiagent systems, multiobjective design selection, and stochastic optimization, and is currently implemented to solve general electromechanical design problems. While this paper presents an overview of the theoretical basis for A-design, it primarily focuses on the method for representing electromechanical design configurations and the reasoning of the agents that construct these configurations. Results from an electromechanical test problem show the generality of the functional representation. [S1050-0472(00)00701-7]

1 Introduction

The bulk of computational design tools are focused on refining the details of an established design concept. However, prior to the creation of an established conceptual design, the designer is faced with a variety of difficult issues such as choice of the basic configuration of components within the design and the selection of components used in the configuration. In creating successful designs, one must also consider how the product will be positioned among competitor products in order to find a profitable market niche. Within this preliminary stage of design, computational aids could assist the designer in reducing the search space of possible solutions and in establishing the specifics of a design artifact. However, it is difficult to envision what the basis for such a computational tool would be since conceptual design has an open-ended formulation. Also, constraints and objectives at this stage of design are rather transitory; new technologies and market niches often cause a designer to change the focus of the design. The intent of *A*-design is to create a design tool capable of creating design solutions while adapting to market changes. The automated design approach of *A*-design establishes an agent-based computational environment to assist a designer in the earlier parts of the design cycle.

The *A*-design approach captures characteristics of human designers while taking advantage of a broad range of computational processes. The development of this new design generation method builds on innovations from artificial life, stochastic optimization, multiobjective optimization, qualitative physics, asynchronous teams, and human cognition. The name, *A*-design, is based on the agent-based and adaptive nature of the process. All design states are created through the collaboration of various design agents. By providing feedback to agents and storing designs with a variety of strengths and weaknesses, the process is able to retain flexibility while improving the search for solutions to the problem at hand.

The current focus of *A*-design is on electromechanical conceptual design. The implemented system differs from previous work in that (1) designs are not only generated but iteratively improved upon to meet objectives specified by a designer, (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 changes in the problem description made by the designer throughout the design process. This paper focuses on the second contribution above by describing in detail the representation used to describe a general class of electromechanical designs and the software agents that create an array of design alternatives within this representation.

2 Related Work

The multiagent subsystem of *A*-design is inspired by work in artificial life and asynchronous teams. Artificial life [1,2] or *A*-life puts computation in the hands of naive agents who often, through following simple reflexive operations, generate a complicated emergent behavior. The idea of exploring multiagent systems in engineering applications has proven useful in asynchronous teams [3] research. 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 multiagent systems succeed when the demands for a more flexible and knowledge-based approach requires the interaction of various software components [4]. However, the usefulness of interacting agents does not require very complex agent types as is shown in single function agents [5]. Single function agents are capable of accomplishing design tasks by interacting to negotiate the various design decisions.

Few optimization techniques are able to address the highly nonlinear, discontinuous, and multimodal problems found in engineering design. However, stochastic optimization techniques such as simulated annealing [6] and Tabu search [7] have been able to solve these problems with some success. Also, genetic algorithms are powerful problem solving tools that have been used in a variety of engineering design problems [8–10]. Solutions within genetic algorithms are synthesized by combining the configurations of other alternatives. These algorithms store many design states simultaneously in order to compare, propagate, and modify alternatives in a manner similar to that used in natural evolution. *A*-design adopts this storage of multiple solutions and "evolves" both populations of designs and agents, the design creators. Rather than random mutation and crossover, the alternatives in the *A*-design process are modified by goal-directed agents.

Multiobjective optimization has supplied a foundation for *A*-design's adaptive nature. Specifically, Pareto optimality provides a method for determining which solutions are most useful under different evaluation criteria [11]. Since its development, many algorithms, especially genetic algorithms, have incorporated

Contributed by the Design Theory and Methodology Committee for publication in the JOURNAL OF MECHANICAL DESIGN. Manuscript received June 1998; revised Jan. 2000. Associate Technical Editor: D. L. Thurston.

Pareto optimality as a method of removing inferior designs (see overview in Fonseca and Fleming [12]). Also, multiobjective decision making has been combined with multiagent systems as seen in Petrie et al. [13] where existing software tools are controlled by a single governing agent that makes decisions based on Pareto optimality.

In applying optimization to electromechanical design, a representation is required to qualitatively reason about how systems are to be constructed to arrive at a desired behavior, and quantitatively reason about how energy, material, and signal state variables effect one another. Computational representations of design function can be accomplished through linguistic approaches such as that seen in Stone and Wood [14], or through descriptions of how discrete components influence one another in a given system [15–19]. While there are advantages to both approaches, the *A*-design representation that is described here builds on the latter, which has been previously used in other design generative methods such as that of Welch and Dixon [20], Ulrich and Seering [21], and Schmidt and Cagan [22,23]. As will be discussed in Section 4, the *A*-design representation combines aspects of several of these approaches into a new representation that can process real world components, their configurations, and interactions.

3 Theoretical Basis and Implementation of *A*-Design

The basic subsystems of *A*-design are (1) an agent architecture that, through the interaction of a multitude of agents, creates viable solutions, (2) a representation used by the agents to produce candidate alternatives, (3) a scheme for multiobjective decision making that retains recessive solutions while focusing on a user specified weighted sum of objectives, and (4) an evaluation-based iterative algorithm for improving basic design concepts to near-optimal designs. The latter two subsystems provide the basic framework for *A*-design and can be generalized to a variety of optimization and problem solving situations. The theory mandates that for a given problem, a method of representing the problem space and agents that manipulate such a representation be defined. In the conceptual electromechanical design problem presented here, the representation of designs and the agents that manipulate designs must be general enough to address a variety of functional specifications. These systems are explained in detail in Sections 4 and 5, respectively.

A-design's search for optimal designs is accomplished through intelligent modification of alternatives by a hierarchy of agents. In the *A*-design methodology, an agent is defined as a knowledge-based strategy 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. By having agents with different abilities contributing to designs, the process gains robustness and variety in solving conceptual design problems. Based on the various responsibilities and preferences, agents are first divided into maker-agents, modification-agents, and manager-agents. The agents interact with designs and other agents based on their perception of the design problem, the preferences presented by the designer, and an agent's individual strategy for solving the design problem. Agents affect their environment by adding or subtracting elements to designs or by altering other agents. With many agents sharing the same responsibilities, a sense of parallel execution is achieved that produces a system more capable of solving complex design problems than the interaction of simple functions in an algorithm. The deterministic nature of simpler algorithms often leads to stagnation at local minima which is prevented in *A*-design by randomly choosing which goal-directed agents to invoke.

Figure 1 shows the flowchart of the overall process. Initially, the system accepts a starting point or description of the problem from the designer. In electromechanical design, the initial specification is a functional description of the design which states the expected inputs and outputs of the system. Maker agents work directly with the input specifications and produce various

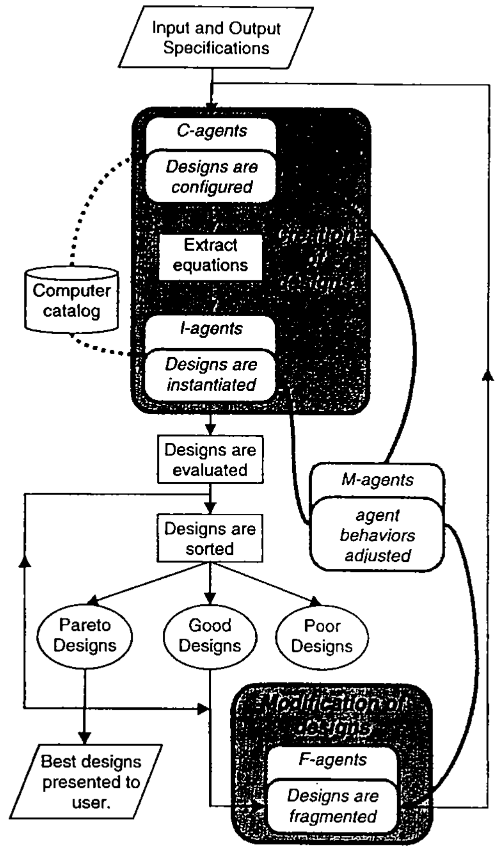

Fig. 1 Electromechanical *A*-design flowchart

design states. The maker-agents are divided into two groups: configuration-agents (*C*-agents) and instantiation-agents (*I*-agents). The configuration-agents reason about how the inputs and outputs can be successfully connected to make conceptual design alternatives. These designs are configurations consisting of simple component descriptions such as gear, spring, and resistor, but are not complete descriptions that specify the exact values of component data.

After design configurations are created by the *C*-agents, 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 referencing these equations, instantiation-agents (*I*-agents) choose actual components from a computer catalog to instantiate the conceptual components. After outfitting designs with real components, the process is now able to evaluate the alternatives created in this maker-agent phase.

Next, designs are evaluated on the various objectives specified by the designer and sorted via a design selection strategy that accommodates both an in-depth search of the design space and an adaptability to user preference. Due to the size of the design space, an exhaustive search of alternatives is not feasible. Therefore, solutions to the design problem are explored through successively iterating through the flowchart in Fig. 1 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 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 near optimal designs. The selection pressure employed in *A*-design conservatively prunes the design space to not only preserve designs that meet current user preference but also designs that represent possible changes in preference. The designs within the current set of alternatives that are candidate optimal choices under any user preference are Pareto-optimal designs. These designs are saved to the next iteration where they will be compared with new and modified alternatives. This set of designs includes the equivalent "recessive" design traits that are useful in meeting changes in preference. Good designs are non-Pareto designs that contain beneficial characteristics under the current preference. These designs are passed to the modification phase where the fragmentation-agents (F-agents) alter concepts to hopefully form better solutions. The rest of the alternatives and poor designs are discarded to make room for new design concepts. This selection of designs is the process' main method of searching the space and is described in more detail in Campbell et al. [24]

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 sorted, how agents are given feedback, and when the process terminates. Feedback increases the number of agents in the agent pool that produce better designs and decreases the number of agents that contribute to poor designs. The process then iterates evolving design populations and agent populations until final Pareto-optimal designs are returned including a subset that best meets the user's current preference.

4 Functional Representation

4.1 Basic Components. In order for agents to generate electromechanical designs, they must be able to reason about the functionality of the devices at a level complex enough to configure components that satisfy a functional specification. We argue that this level of sufficiency focuses on the transformation of energy, material, and signal (as seen in Pahl and Beitz [25]) by components throughout the device and the mode of transformation of those quantities (by amplification, integration, etc.). We present a representation based on designs being describable by the points of connection between components in a design. By describing these points of connectivity, a formal syntax is established which models the input–output functionality of a design as well as internal states of interacting components. The representation describes connection points by their position, orientation, and state variables, and the behavior of components as transformations of energy, signal, or material. In this sense, our representation builds on that of Welch and Dixon [20].

The basic function description focuses on connection points between components represented by a structure called a functional parameter (FP). In describing a specific design problem to the A-design process, the user must classify the desired function of the design as a set of input and output behaviors within the function parameter formulation. C-agents then choose components to connect the input and output functional parameters in order to make basic design configurations. With each C-agent having distinctly different strategies for adding components to a design (as is seen in Section 5), alternatives are constructed in a variety of configurations. From these basic configurations, instantiation agents replace all simple component descriptions in the design with real components found in the computer catalog. This method of using functional parameters was originally introduced in Welch and Dixon [20] and is expanded in the A-design representation to allow for more general component connectivity and a more complete component description.

The FP structure contains several variables for describing electromechanical behavior. As mentioned above, the user must describe the design problem in terms of the input and output behavior that is desired for the design. For example, in Fig. 2, a design for a weighing machine is specified by an input weight and an output dial displacement. The FP_{input} and FP_{output} structures detail the specific behavior desired at these points.

Figure 3 shows the contents of the expanded functional parameter structure and the possible values variables can have. The most crucial slots in this structure are the **through** and **across** variables; these slots provide a general description of the state variables at a given point in a design. Through variables include concepts like

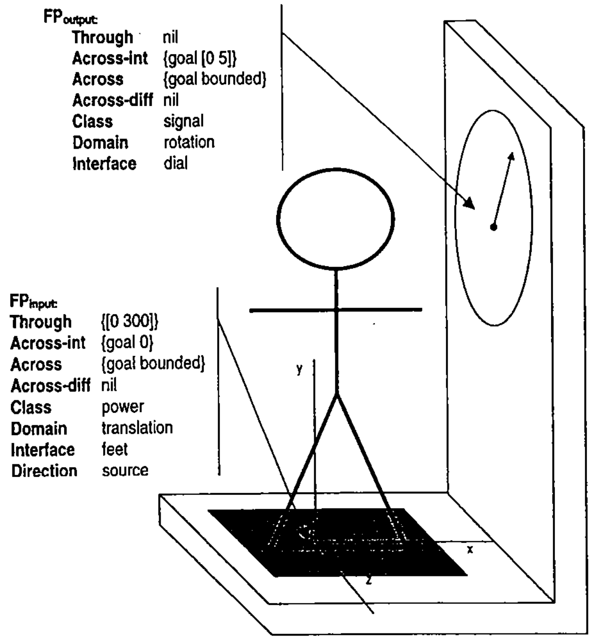

Fig. 2 Weighing machine design problem describe as input and output FP's

force, torque, current, and flow rate and derive their name by affecting a system's behavior as it passes "through" elements in the system. This is opposed to across variables such as velocity, voltage, and pressure where system elements behave according to the difference in the variable "across" an element. Often, through and across variables also have a physical meaning when integrated or differentiated with respect to time. For example, velocity is integrated to find displacement and differentiated to find acceleration. Since these variables have physical meaning in many domains (i.e., electrical, mechanical, or hydraulic), they are a useful classification for modeling both familiar and unfamiliar domains and are used here as a framework for the agents to reason about functionality. The **domain** variable in the FP structure describes what domain the functional parameter is currently operating in. Table 1 depicts the physical definition of the through and across variables for each domain. The table also depicts how components within each domain proportionally relate across and through variables. Often, domains have physical components for each transformation of the through variable to the across variables.

The values for **through** and **across** slots can be either quantitative or qualitative in nature. While a numerical value or range of values describes a specific amount, values such as *bounded* or

FP - Functional Parameter:

Through	{any real number, *bounded, unbound*}
Across-integral	{any real number, *bounded, unbound*}
Across-none	{any real number, *bounded, unbound*}
Across-differential	{any real number, *bounded, unbound*}
Class	{*power, signal, material*}
Domain	{*trans, rotate, electric, hydraulic*}
Position	{x, y, z}
Orientation	{θ, φ, ψ}
Interface	{standard size, e.g. *9/16" bolt*}
Direction	{*nil, source, sink*}.

Fig. 3 The contents of the FP structure

Table 1 Through and across variables for each domain

	Translational	Electrical	Rotational	Hydraulic
Through Variable	Force (f [Newtons])	Current (I [amps])	Torque (T [N-m])	flow rate (m [kg/s])
Across Variable	Velocity (v [m/s])	Voltage (v [volts])	angular speed (Ω [rad./s])	Pressure (P [N/m²])
Through ∝ Across	Damper friction	resistor	Damper friction	Valve viscous drag
Through ∝ d(Across)/dt	mass	capacitor	Rotational inertia	tank
Through ∝ ∫(Across) dt	spring	inductor coil	Rotational spring	long piping

unbound further qualify the more complicated behavior of a functional parameter. To classify a variable as *bounded* means that the true numerical value will eventually converge to a single point while *unbound* variables describe a numerical value that diverges. Throughout the construction of a design, the *bounded* and *unbound* values act as placeholders to inform agents of the behavior at a functional parameter. Also, a *goal* prefix can be added to FP slots to keep track of user specified functionality as opposed to the true physical value caused by connections.

Along with the **through**, **across**, and **domain** variables, the FP structure represents other important characteristics of connection points. The **class** characteristic is a description of what is physically transferred through the connection. Although not implemented in Welch and Dixon [20], the hope is that other than describing energy flow (**class**=power) is shown in these electromechanical devices, one could also describe flow of information (**class**=signal), or possibly even flow of a material (**class** =material). Also, the functional parameter includes slots for describing the location of a connection point in space with the **position** and **orientation** variables as well as a description of the connection's **interface** with other components and the **direction** of flow at that point.

Besides describing connecting points, the components in this representation are describable by the embodiment (EB) structure that contains the basic description of how inputs and outputs on a device relate to one another. This structure builds on the EB structure of Welch and Dixon as well, but includes enhancements such as *n*-port components, nonlinear through and across transformations, and constraints on FP connectivity. These descriptions are used to model how a class of components operates, but not the specifics of a particular component. For example, a spring embodiment has equations and constraints describing the general behavior of springs, but does not specify numerical values for variables such as spring stiffness. A catalog of real components is used by *I*-agents to instantiate the embodiments in a design after the basic configurations have been established.

Figure 4 shows the makeup of the EB structure. The **variables** slot lists the relevant data for distinguishing component instantiations of the embodiment. The **constraint parameter** (CP) establishes how components are constrained at their outputs thus preventing infeasible connections with other components in the system. A constraint parameter has the same variables as the functional parameter and can thus prevent improper matching of domain, interface, direction of flow, etc. The magnitude-change (**MG-change**) slot includes executable code to describe the transformation between the energy flow state variables into and out of ports on the components. These include such formulas as "$F=k\cdot x$" for springs, or possibly more complicated nonlinear functions such as $F=K(x)\cdot x$. Finally, the position-change (**PO-change**) slot is a matrix for transforming the position and orientation variables within the FP's connected to the embodiment.

EB - Embodiment:

variables	(length, flux density, number of gear teeth, etc.)
CP's	constraint parameters - constrain which FP's can be connected.
MG-Change	magnitude change functions relating the through and across variables at each port to those of other ports.
PO-Change	position change matrix relating position of each port to the position of other ports.

Fig. 4 The contents of the EB structure

The description of a design state, either complete or incomplete, is a collection of functional parameters and embodiments. When the configuration-agents connect new EB's to a partial design, the FP's throughout the system are updated. The notion of updating functionality in a design is based on Schmidt and Cagan [23] where designs are created through a functional grammar. The *A*-design representation of FP's and EB's includes a framework for allowing designs to be constructed one embodiment at a time, thus allowing for agents to reason about how previously added components effect a design and how a system behaves even when it is not fully connected. This allows for an interactive construction of designs where *C*-agents add components based on their individual design strategies. In Welch and Dixon [20], design alternatives were conceived through an immediate expansion of the system's input and output behavior. As a result only simple design states were constructed. The framework here for allowing agents to interact in partial design states allows for an infinite number of possible design configurations. The following example sums up the construction of basic design configurations.

Figure 2 introduced the weighing machine example, where the user specified the input and output FP's that describe the functionality of a device. Note that at the output, the functionality of dial displacement is classified as a goal range in the across-integrated slot of 0 to 5 radians. This range represents an angle of displacement since the integrated across variable in the rotational domain is an angle. The goal slots in the input and output FP's direct agents in achieving the desired functionality. At the input FP, the domain variable is translational and therefore specifying the across-integrated slot to {goal 0} relates to zero displacement at this point. Also, the through variable has a range of {[0 300]} which specifies it as an input force from zero to 300 pounds. Note that in the across variable slots of the input and output the user has specified their values as {goal bounded} since damped motion is a desired functionality required in the weighing machine design problem.

C-agents begin attaching various EB's to fulfill the *goal* slots in the initial FP_{input} and FP_{output}. Figure 5(*a*) shows a partial design state with functional parameters represented as ovals and embodiments as rectangles. The configuration presently contains seven EB's as a result of various *C*-agent contributions. As a result of connecting these EB's, FP_{input} and FP_{output} have changed slightly. The **through** variable range in the input has caused the output FP to include a *bounded* value for its **through** variable. Due to fric-

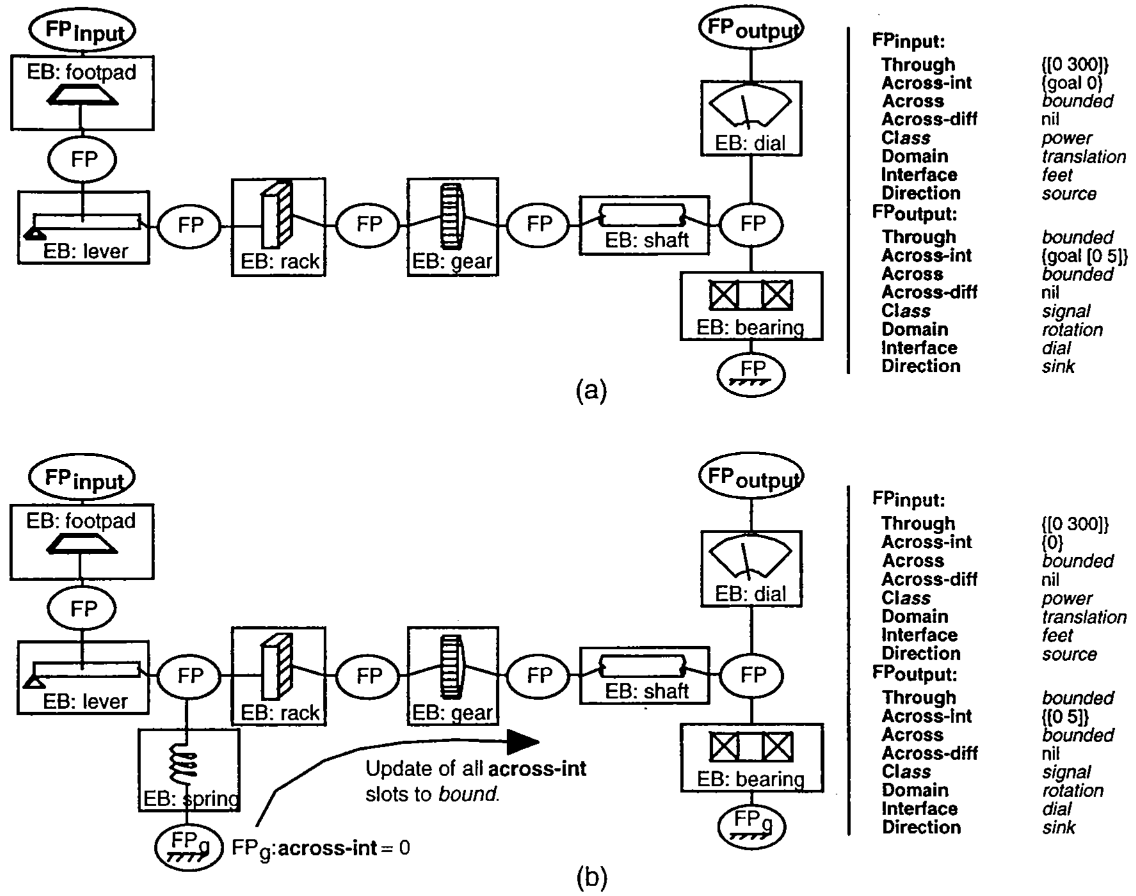

Fig. 5 (*a*) **Partial design state with update FP's;** (*b*) **design is completed with addition of spring**

tion in the system, especially in the bearing, the across variable in the input and output FP has changed from {*goal bounded*} to *bounded.*

The update mechanism performed between agent operations is an important mechanism in constructing unique design configurations. Subsequent agents reason about the updated design state, allowing for a progression towards a complete design problem. Agents are not constrained to add components in series, parallel or any particular order, but instead follow their individual preferences for design; these preferences are addressed in detail in Section 5. These seven EB's have completed all *goal* slots in the input and output FP's except for {*goal* [0 5]}, and {*goal* 0}. When a new spring EB is added to the design, as seen in Fig. 5(*b*), a complete weighing machine is created. The update mechanism transfers knowledge from the ground FP on the spring throughout the design to note that **across-integrated** is now *bounded* and any goal ranges in **across-integrated** slots have also been met. Following this update, all the goals in the input and output FP's have been met and therefore the design is viewed as complete and brought to closure by extracting the behavioral equations, as discussed below.

4.2 Extracting Equations. After a design configuration is completed, symbolic equations are extracted to determine the sum effect of the embodiments on the behavior of the design. These behavioral equations describe how the goal ranges in the system are formulated. For example, the equation for the *goal* dial displacement (θ_{dial}) and *goal* input displacement (x_{input}) in Fig. 5(*b*) are found to be:

$$\theta_{\text{dial}}=\frac{1}{r_{\text{gear}}k_{\text{spring}}}\frac{d_1}{(d_1+d_2)}F_{\text{weight}}, \tag{1}$$

$$x_{\text{input}}=\frac{1}{k_{\text{spring}}}\frac{d_1^2}{(d_1+d_2)^2}F_{\text{weight}}, \tag{2}$$

where $\theta_{\text{dial}}=[0\ 5]$, $x_{\text{input}}=0$, and $F_{\text{weight}}=[0\ 300]$.

These equations are formed by extracting key information about the connections in the system: r_{gear} is the radius for the gear EB, k_{spring} is the stiffness term from the spring EB, and d_1 and d_2 are length terms extracted from the second class lever EB. This equation is symbolically determined by a computational recursive algorithm which starts at each goal range of the system and works back to given data, such as F_{weight} or zero displacement at ground FP's. The equation extractor derives information stored in the embodiments' magnitude-change (**MG-change**) functions which describe how the embodiments interact with neighboring functional parameters. By performing a depth first search through the graph of connecting FP nodes, the extractor also determines **across** and **through** variable transformations of series and parallel connections. Through this manner, the process finds equations for a wide variety of design configurations including cycles in the graph and nonlinear FP-transformations that occur in the EB.

These goal range equations are the basis for determining the values for causal-based objectives. The closer that the left-hand side of Eqs. (1) and (2) are to the behavior predicted on the right-hand side, the better the system conforms with the user specifications. The accuracy of the θ_{dial} and x_{input} equations yields values for the dial error and input displacement subobjectives. The iterative nature of the algorithm is then responsible for improving the accuracy of these equations through the various design instances that are created and evaluated.

5 Agent Strategies

Within the electromechanical *A*-design methodology, agents are divided into three main categories: maker agents, modification agents, and manager agents. Within these categories, further divisions can be made; for example, maker agents are divided into configuration-agents (*C*-agents) and instantiation-agents (*I*-agents). All agents are implemented as independent functions that accept completed or partial design states and return their modifications or additions. Throughout the iterative process, the manager agents make decisions about which designs are better and how the agents should be grouped, penalized, or rewarded for their past experience.

5.1 Maker Agents

5.1.1 Configuration-Agents (C-Agents). *C*-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 that achieve the functionality specified in the design problem. In doing so, *C*-agents focus on specific areas of the complete design problem to determine what portion of the system they will address and what EB's best fulfill this functionality. For this reason, their operation is closely linked to the functional representation described above. The embodiments present within a given design are the result of contributions from a variety of *C*-agent behaviors.

Initially, *C*-agents examine a partial design state to determine where to attach a new embodiment. In an incomplete design state like that found in Fig. 5(*a*) there are many opportunities to connect embodiments; theoretically a new embodiment can be connected at any functional parameter present in the system. Therefore, *C*-agents are created that encompass the various tactics that can be used in adding new embodiments to a design. Table 2 shows various characteristics that distinguish agents within each type. In the *C*-agent type, agents exist for each combination of characteristics in the first column. For example, some agents prefer electrical EB's while others rotational, translational, or hydraulic EB's. As a result of this preference, agents can rule out some functional parameters in the design state as possible candidates for attaching embodiments. For example, an agent preferring electrical components can safely ignore functional parameters in the hydraulic domain in favor of electrical FP's; however, this same agent might opt for rotational FP's if no electrical connections are present, as a motor might provide a means of producing electrical FP's from rotational connections. Upon limiting the embodiments to apply and the possible functional parameters to connect to, *C*-agents then use their preference for direction of flow connections. As seen in Table 2, agents can prefer connecting to ports that are supplying energy (sources) or receiving energy (sinks). This is followed by another preference that chooses to connect embodiments in parallel or series. Again, these different agent preferences can be seen in Table 2 where different combinations of these preferences can uniquely create a wealth of different agent types, for example, an agent who prefers connecting electrical EB's in a serial manner to sink nodes in the partial design state. Other than specific preferences for where to connect components in a design, agents also have different strategies for fulfilling the functionality of the design determined by the manner in which they connect other ports of an embodiment to other functional parameters in the design.

The specifics explained here are for the current set of *C*-agents. They are by no means the only way to implement such *C*-agents and are shown here to portray 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 also equally important that they are not so directed as to 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 better searched by the set of *C*-agents producing extravagant alternatives at first and then progressively becoming focused on improving the better solutions over time. The balance of random search and goal-directed agents is a major factor in allowing the *A*-design process to solve difficult conceptual design problems.

5.1.2 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. The catalog contains information on the values of variables used in the connection and behavior of embodiments in a configuration as well as other information used for evaluating alternatives. For example, in Fig. 6, a gear EB is instantiated by gear components that contain data on the radius of the gear (r_{gear}) which is used in solving the equations, the pitch of gear teeth which is used to determine the exact interface of a functional parameter, and the cost of the gear which is used in calculating the total cost of a design.

Similar to *C*-agent preferences, *I*-agents can also exhibit different preferences in instantiating embodiments with actual components. The *I*-agent preferences are shown in Table 2 and often

Table 2 Different preferences that exist in the current set of *C*-, *I*-, and *F*-agents

C-Agents (48 agents)	I-Agents (6 agents)	F-Agents (36 agents)
Domain Preference electrical EB's / translational EB's / rotational EB's / hydraulic EB's	**Objective Preference** prefer inexpensive components / prefer lightweight components / prefer efficient components	**Design Preference** modify expensive designs / modify inexpensive designs / modify heavy designs / modify light designs / modify efficient designs / modify inefficient designs
Parallel vs. Serial connect EB's in parallel with other FP's / connect in series	**Variable Preference** select component based on variables present in behavioral equations / select component based on variables present in FP interfaces	**Objective Preference** remove expensive components / remove heavy components / remove inefficient components
Source vs. Sink connect to source FP's / connect to sink FP's		**Degree of Fragmentation** remove component from design / remove EB from design
Other EB connections link new EB to existing FP's / link new EB to new FP / link new EB to new ground FP		

Embodiment	**EB** - Gear: **variables** Rgear, Dshaft, Pitch **CP's** 1: **class**=power **domain**=rotation **interface**=shaft-hole(Dshaft) 2: **class**=power **domain**=translation **interface**=teeth(Pitch) **MG-Change** **through1** = (**through2** * D)/2 **across-int1** = (**across-int2** *2)/D **across1** = (**across-int2** *2)/D **across-diff1** = (**across-diff2** *2)/D **through2** = (**through1** * 2)/D **across-int2** = (**across-int1** *D)/2 **across2** = (**across-int1** *D)/2 **across-diff2** = (**across-diff1** *D)/2 PO-Change (90.0 0.0 0.0 (D / 2))
Components	;;; Gear.comps ;;; This file contains information on gear components. ;;; It contains two lists: one instantiating the variables ;;; in the gear EB, and another with evaluatable criteria ;;; Components have the format: ;;; (name (Dia [m], shaft-Dia [m], teeth, Pitch [teeth/in.]) ;;; (cost [$], mass [kg.], efficiency)) ;;; The following gears are from the Nordex catalog p. 286 (LAS-F7-28 (11.1e-3 6.35e-3 28 64) (5.75 5e-3 0.98)) (LAS-F7-36 (14.2e-3 6.35e-3 36 64) (6.29 10e-3 0.98)) (LAS-F7-46 (18.3e-3 6.35e-3 46 64) (6.66 14e-3 0.98)) (LAS-F7-60 (23.8e-3 6.35e-3 60 64) (7.54 65e-3 0.98)) (LAS-F7-75 (29.8e-3 6.35e-3 75 64) (8.18 17e-3 0.98)) (LAS-F7-90 (35.7e-3 6.35e-3 90 64) (9.40 22e-3 0.98)) (LAS-F7-104 (41.3e-3 6.35e-3 104 64) (10.26 25e-3 0.98)) (LAS-F7-128 (50.8e-3 6.35e-3 128 64) (12.03 33e-3 0.98)) (LAS-F7-168 (66.7e-3 6.35e-3 168 64) (15.69 35e-3 0.98)) (LAS-F7-208 (82.6e-3 6.35e-3 208 64) (21.99 45e-3 0.98)) (LAS-F7-248 (98.4e-3 6.35e-3 248 64) (24.74 50e-3 0.98))

Fig. 6 Gear embodiment with sample of gear component file: gear.comps

include specific design objectives that the user wishes to optimize. For example, if cost and mass of a design artifact are to be minimized, then specific *I*-agents will have preferences for choosing inexpensive or lightweight components. Also, *I*-agents can have preference for which variables in the embodiment are the deciding factor in choosing a component, for example, choice of a gear component can be based on radius or gear pitch. Again, the strategies shown here are not the only possible implementations; they merely reflect a combination of goal-directed selection of components and the stochastic interaction of agents.

5.2 Modification Agents

5.2.1 Fragmentation-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 fragmentation-agents are crucial in driving the system towards optimal design states. *F*-agents individually choose a design to be modified and remove EB's 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 as is seen in Table 2. Similar to how *I*-agents are differentiated, these agents differ from each other in their choice of objectives to address. For example, an *F*-agent might remove a motor from a system based on the fact that the motor is the most expensive component within that design. *F*-agents also differ in selecting which designs to modify and in choosing to fragment the configuration by removing EB's or just to remove costly component instantiations. Note that in fragmenting configurations, some *F*-agents might prefer to remove repeated EB's or dangling FP's that have no effect on the behavior of a design. 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 to this work 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 Manager-Agents. The populations of both designs and agents are maintained by the *M*-agents. *M*-agents are responsible for providing feedback to the other agents as well as controlling basic parameters within the algorithm. By examining Pareto and good design populations, *M*-agents adjust agent populations and design populations to accommodate changes in user preference. As currently implemented, *M*-agents reinforce the number of agents of a given agent-type that have contributed to past Pareto and good designs, while deemphasizing agent-types that have produced poor solutions. Future *M*-agent strategies include grouping agents that have produced radically different alternatives together in hopes of combining beneficial ideas as well as grouping similar agents together in attempts to concentrate on achieving particular design characteristics. In addition to grouping agents, *M*-agents will also identify successful combinations of components through "chunking" methods such as Laird et al. [26] for use in constructing new alternatives in future iterations or design problems. The range of possible *M*-agent behaviors is large and intricate and lends itself to future development, however, current simplified strategies of providing feedback have produced promising results.

6 Results

The electromechanical design test problem of fabricating weighing machine alternatives has produced some innovative designs. The implemented electromechanical *A*-design system includes the FP and EB 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 subprocesses. The system is written in LISP and runs on a Silicon Graphics Indigo 2. The weighing machine problem posed in this example (see Fig. 2) includes four objectives to be optimized: minimize cost, minimize mass, minimize dial error, and minimize input displacement. The design specifications are split between both the functional description of input and output FP's as well as metrics for other constraints and objectives in the problem. In performing conceptual design, one often poses both high-level objectives such as "reduce cost of design artifact" and low-level objectives such as "minimize settling time at a given point."

The catalog of components for the weighing machine example is a preliminary list of possible components that can be represented within the methodology. Table 3 shows the embodiments that are currently being represented in the process. In addition, component library files exist for each embodiment in Table 3. These files are based on data from components manufactured or distributed by Allied Electronics, Nordex Inc., and Mc-Master Carr Supply Company. The library has over 300 component instantiations for the 32 embodiments in Table 3.

Figure 7(*a*) is a final design instantiation of the example shown in Fig. 5. This design is returned as the best solution from one particular run of the *A*-design process. It is found after 25 iterations through the *A*-design flowchart (Fig. 1) with an average of 100 designs per iteration. In this particular example, the user weights all four objectives equally. As a result of this preference, the design has a relatively low cost and weight, but to some extent sacrifices input displacement. In the second example (Fig. 7(*b*)), the user preference puts more emphasis on minimizing input displacement. As a result the design uses more components and a slightly higher cost to produce a higher quality artifact. The generality of the functional representation and differing agent strategies leads to some varied and unique solutions in accomplishing the user's specifications as further seen in the results of Campbell et al. [24].

7 Conclusions

This paper has focused on the representation and creation of electromechanical design configurations within *A*-design. By itself, *A*-design is a general methodology for searching unstructured design spaces based on the collaboration of many goal-directed agents. The methodology has four distinct subsystems in all: an

Table 3 Current embodiments implemented in the system

Battery	Lever (class 3)	Resistor	Solenoid
Cable	Motor	Rotational Bearing	Spring
Capacitor	Pipe	Rotational Damper	Sprocket
Gear	Piston	Rotational valve	Stopper
Electrical valve	Potentiometer	Switch	Transistor
Inductor coil	Pulley	Tank	Translational Bearing
Lever (class 1)	Rack	Torsional Spring	Translational Damper
Lever (class 2)	Relay	Shaft	Worm gear

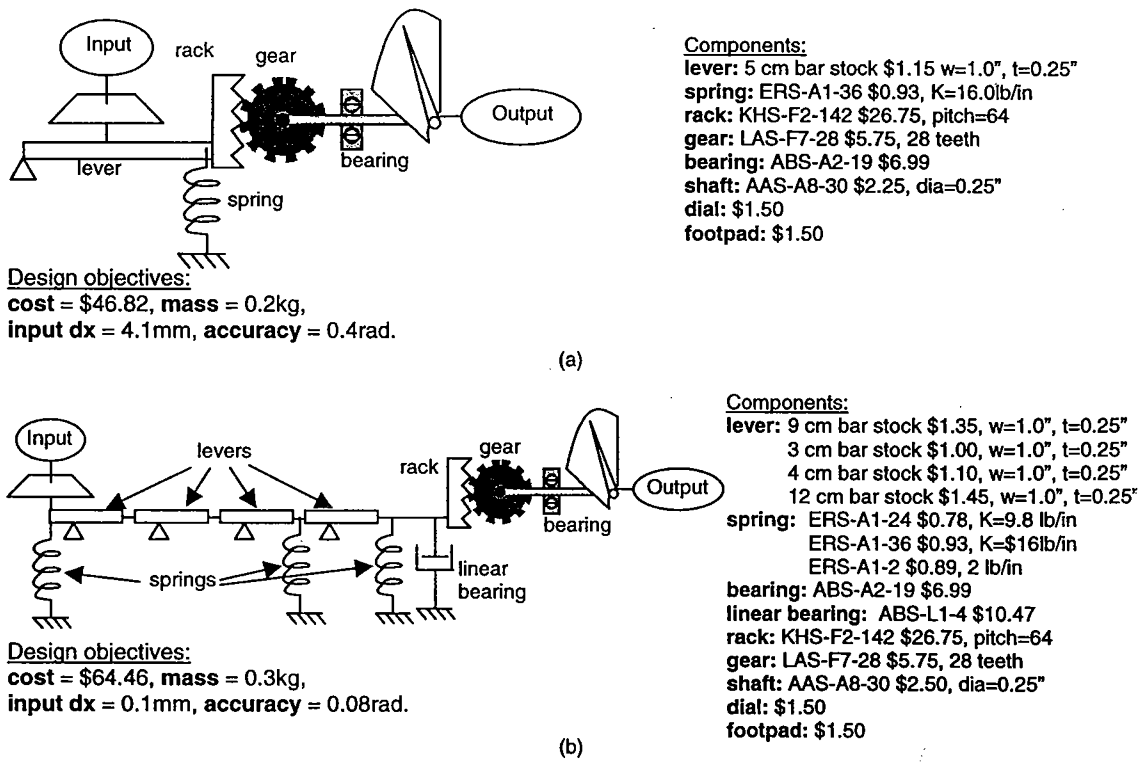

Fig. 7 Three different alternatives created by the *A*-design process. Design (*a*) is found by an equal preference for the four design objectives, whereas design (*b*) is found by placing more importance on minimizing input displacement

agent architecture, a multiobjective design selection scheme, a functional representation for describing solutions, and an iterative-based algorithm for evolving optimally directed design states.

In order to create electromechanical design configurations, A-design must be supplied with a description of the design problem in the form of inputs and outputs, a set of objectives to be optimized, a library of electromechanical embodiments, and a catalog of real components. The agents of the A-design system do not contain any information on the specifics of the electromechanical design problem at hand. Instead, they contain preferences for specific objectives or strategies for particular ways of creating design topologies. In order to keep agents as general as possible for a wide variety of design problems, the knowledge of how components behave in a design is placed in the EB representations. This generality can accommodate mechanical, electrical, or even hydraulic inputs and outputs as well as multiinput/multioutput electromechanical systems.

The power of A-design's generality and adaptability is also due to the iterative approach to design generation. Since functionality of a given design problem is specified in part by the objectives that define the problem, generated solutions can be compared on a common metric in accomplishing design specifications. This iterative process also reduces the burden on the agent interaction since difficulties that arise amongst the cooperation of interdisciplinary teams of agents are simply pruned from the process as the iterations ensue. Initially, agents create 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 a 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 produced.

The results from the weighing machine test problem display a diverse set of possible design alternatives that can be created. These design configurations depict a successful combination of the agent architecture and functional representation subsystem which, although currently integrated within the A-design methodology, could be used independently by other problem solving methods in the generation of topologies for electromechanical design problems.

Acknowledgments

The research effort was partially sponsored by the Defense Advanced Research Projects Agency (DARPA) and Rome Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-96-2-0304. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA, Rome Laboratory, or the U.S. Government. The authors would also like to acknowledge the financial support of the National Science Foundation under grant EID-9256665.

References

[1] Langton, C. G. (ed.), 1988, *Artificial Life*, Addison Wesley, Reading, MA.

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

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

[4] Lander, S. E., 1997, "Issues in Multiagent Design Systems," IEEE Expert, **12**, No. 4, pp. 18–26.

[5] Grecu, D. L., and Brown, D. C., 1996, "Learning by Single Function Agents During Spring Design," *Artificial Intelligence in Design*, Gero, J. S., and Sudweeks, F., eds., Kluwer Academic, Dordrecht, Netherlands, pp. 409–428.

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

[7] Glover, F., 1989, "Tabu Search—Part I," ORSA J. Comput. **1**, No. 3, pp. 190–206.

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

[9] Queipo, N., Devarakonda, R., and Humphrey, J. A. C., 1994, "Genetic Algorithms for Thermosciences Research: Application to the Optimized Cooling of Electronic Components," Int. J. Heat Mass Transf., **37**, pp. 893–908.

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

[11] Balachandran, M., and Gero, J. S., 1984, "A Comparison of Three Methods for Generating the Pareto Optimal Set," Eng. Opt. **7**, pp. 319–336.

[12] Fonseca, C. M., and Fleming, P. J., 1995, "An Overview of Evolutionary Algorithms in Multiobjective Optimization," Evol. Comput. **3**, pp. 1–16.

[13] Petrie, C. P., Webster, T. A., and Cutkosky, M. P., 1995, "Using Pareto Optimality to Coordinate Distributed Agents," AIEDAM, **9**, pp. 269–281.

[14] Stone, R. B., and Wood, K. L., 1999, "Development of a Functional Basis for Design," Proc. Design Engineering Technical Conferences, DETC99/DTM8765, Sept. 12–15, 1999, Las Vegas, NV.

[15] Bracewell, R. H., and Sharpe, J. E. E., 1996, "Functional Description Used in Computer Support for Qualitative Scheme Generation- 'Schemebuilder'," Artif. Intel. Eng. Des. Anal. Manufac. **10**, pp. 333–345.

[16] Kota, S., 1990, "A Qualitative Matrix Representation Scheme for the Conceptual Design of Mechanisms," *Proceedings of the ASME Design Automation Conference*.

[17] Navinchandra, D., Sycara, K. P., and Narasimhan, S., 1991, "A Transformational Approach to Case-Based Synthesis," AIEDAM, **5**, pp. 31–45.

[18] Palmer, R. S., and Shapiro, V., 1993, "Chain Models of Physical Behavior for Engineering Analysis and Design," Res. Eng. Des., **5**, pp. 161–184.

[19] Chakrabarti, A., and Bligh, T. P., 1996, "An Approach to Functional Synthesis of Mechanical Design concepts: Theory, Applications, and Merging Research Issues," Artif. Intel. Eng. Des. Anal. Manufact. **10**, pp. 313–331.

[20] Welch, R. V., and Dixon, J., 1994, "Guiding Conceptual Design Through Behavioral Reasoning," Res. Eng. Des., **6**, pp. 169–188.

[21] Ulrich, K., and Seering, W., 1989, "Synthesis of Schematic Descriptions in Mechanical Design," Res. Eng. Des., **1**, pp. 3–18.

[22] Schmidt, L. C., and Cagan, J., 1995, "Recursive Annealing: A Computational Model for Machine Design," Res. Eng. Des., **7**, pp. 102–125.

[23] Schmidt, L. C., and Cagan, J., 1998, "Optimal Configuration Design: An Integrated Approach Using Grammars," ASME J. Mech. Des., **120**, pp. 2–9.

[24] Campbell, M. I., Cagan, J., and Kotovsky, K., 1999, "A-Design: An Agent-Based Approach to Conceptual Design in a Dynamic Environment," Res. Eng. Des. **11**, No. 3, pp. 172–192.

[25] Pahl, G., and Beitz, W., 1988, *Engineering Design—A Systematic Approach*, Springer-Verlag, New York.

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