Proceedings of the ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference
IDETC/CIE 2016
August 21-24, 2016, Charlotte, North Carolina

DETC2016-59570

DESIGN OF COMPLEX ENGINEERING SYSTEMS USING MULTIAGENT COORDINATION

Nicolás F. Soria
School of Mechanical, Industrial and Manufacturing Engineering
Oregon State University
Corvallis, Oregon, 97331
Email: soriazun@onid.oregonstate.edu
Colegio de Ciencias e Ingeniería
Universidad San Francisco de Quito
Quito, Ecuador

Mitchell K. Colby
School of Mechanical, Industrial and Manufacturing Engineering
Oregon State University
Corvallis, Oregon, 97331
Email: colbym@engr.orst.edu

Irem Y. Tumer *
School of Mechanical, Industrial and Manufacturing Engineering
Oregon State University
Corvallis, Oregon, 97331
Email: irem.tumer@oregonstate.edu

Christopher Hoyle
School of Mechanical, Industrial and Manufacturing Engineering
Oregon State University
Corvallis, Oregon, 97331
Email: chris.hoyle@oregonstate.edu

Kagan Tumer
School of Mechanical, Industrial and Manufacturing Engineering
Oregon State University
Corvallis, Oregon, 97331
Email: kagan.tumer@oregonstate.edu

ABSTRACT

In complex engineering systems, complexity may arise by design, or as a by-product of the system's operation. In either case, the root cause of complexity is the same: the unpredictable manner in which interactions among components modify system behavior. Traditionally, two different approaches are used to handle such complexity: (i) a centralized design approach where the impacts of all potential system states and behaviors resulting from design decisions must be accurately modeled; and (ii) an approach based on externally legislating design decisions, which avoid such difficulties, but at the cost of expensive external mechanisms to determine trade-offs among competing design decisions. Our approach is a hybrid of the two approaches, providing a method in which decisions can be reconciled without the need for either detailed interaction models or external mechanisms. A key insight of this approach is that complex system design, undertaken with respect to a variety of design objectives, is fundamentally similar to the multiagent coordination problem, where component decisions and their interactions lead to global behavior. The design of a race car is used as the case study. The results of this paper demonstrate that a team of autonomous agents using a cooperative coevolutionary algorithm can effectively design a Formula racing vehicle.

INTRODUCTION

Complex engineering systems, such as state-of-the-art aircraft, advanced power systems, unmanned aerial vehicles, and autonomous automobiles, are required to operate dependably in an ever widening variety of environmental conditions, over a wide range of missions. Such systems must be cost-effective

*Address all correspondence to this author.

while being dependable in potentially extreme conditions and adaptable to a given environment. When a large system is designed, multiple design teams are involved. These teams often are formed according to disciplinary lines, and each team is responsible for the design of a subsystem. Each team aims to maximize the performance of their subsystem, but must be aware of interactions between subsystems and system-level constraints in order to result in high overall system performance. In some occasions the goal of one team can be in conflict with the interests of another team. In many design problems, design engineering teams share design variables or constraints, which is also controlled by a systems engineering team. Different tradeoffs are required between many design teams before all the subsystems can be implemented in the final systems.

As the complexity of the system increases, it becomes exceedingly difficult to model such interactions and explore the design space in a manner that allows system level certification goals to be met. A systematic method that explores this space, while providing the necessary adaptability to meet mission needs and dependability with respect to mission requirements is needed. The key insight of this paper is that complex system design, undertaken with respect to a variety of design objectives, is fundamentally similar to the multiagent coordination problem. In both instances, the decisions at the component level (subsystems or agents), and the interactions among those components, lead to global behavior (complex system or multiagent system.)

In multiagent coordination, a key research challenge is to determine what each agent needs to do so that the system as a whole achieves a predetermined objective. This does not in itself "solve" the design problem; rather, it shifts the focus from modeling interactions to determining how to evaluate/incentivize components so that their collective behavior achieves the system design goals. This shift in focus is critical to enabling a new paradigm to emerge: multiagent coordination approaches can now be used to determine how to distribute credit (or blame) in a design process to the components/stages in the design that are critical to success (or failure).

The overall goal of this research is to formulate design agents that will explore all the possible design solutions for a complex engineering system. To be able to achieve more complex solutions, it is necessary to coordinate the actions of all the design teams. Figure 1 ilustrates the design process envisioned in this paper. The approach we explore is to implement a team of autonomous agents responsible for selecting the best concept using multiagent coordination. After the customer and engineering requirements are defined, engineers will create a team of agents suitable for the problem related to the system level objectives. A cooperative coevolutionary algorithm will perform the design exploration and multiagent coordination. The algorithm will autonomously evaluate, select and refine the design solution that results from the best tradeoffs between all the subsystems.

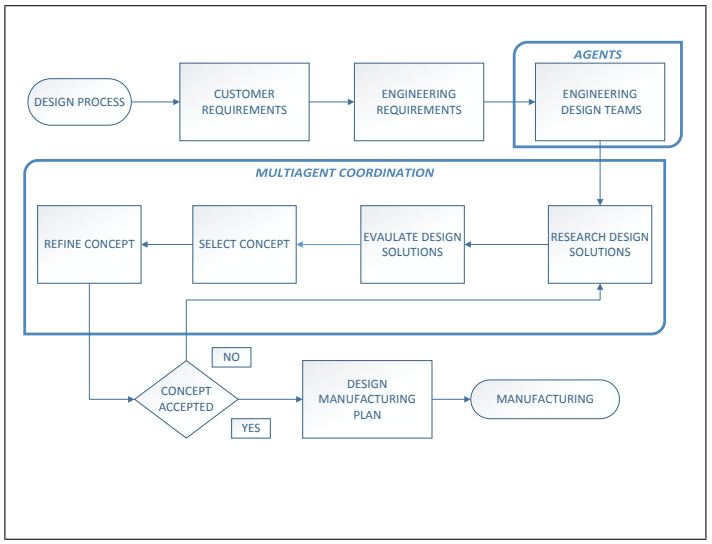

FIGURE 1: Design Process

BACKGROUND

Complex System Design

Selection of design architecture while considering various design criteria and sources of uncertainty is a fundamental research problem in designing complex systems. Explicitly computing quantitative and qualitative objectives of a complex system is generally viewed as the preferred method for formalizing the design process; however, one of the key problems in typical large-scale engineering system design is the over-emphasis on requirement satisfaction for evaluating design alternatives [17]. This focus is primarily the result of the acquisition process, but is exacerbated by overly simplistic design objectives, such as minimizing weight or cost, that do not reflect the true value of the designed system. As an example, rather than making design decisions based primarily upon requirement (i.e., constraint) satisfaction, Value-Centric Design (or Value-Driven Design) offers an alternative approach with the formulation of a system-level design objective that reflects the true value of the system, which can be subsequently *optimized* [11]. This is a dramatic change in perspective for system design, promising a reduction (or elimination) of cost and schedule overruns [7, 10] by identifying high value designs for development. Value-Centric Design can be considered part of the larger field of Decision-Based Design (DBD) [15, 25]. DBD has been specifically developed in the system design community as a decision-theoretic approach to selecting a preferred system design from among the alternatives. DBD takes an enterprise-level view of the design problem, considering not only typical engineering concerns but also broader objectives that comprise the total value of the system to the enterprise.

In this research, we seek an alternative process that enables distributed design of complex systems based on multiagent coordination, described next.

Multiagent Coordination

Multiagent coordination is a key research area in agent-based approaches to automation [24]. One of the biggest challenges in such an approach is decentralization of control, and in particular the question of how to incentivize the individual agents such that they work together [9] to achieve the system objective. The key challenge is that a system designer needs to address two major credit assignment problems: *structural* and *temporal* [9, 24] credit. The first addresses who should get credit (or blame) for system performance, and the second addresses which key action (at which key time step) is responsible for fulfilling the objective [2, 23].

The temporal credit assignment problem has been extensively studied through single-agent reinforcement learning [9, 21]. The structural credit assignment problem has also received attention, and has been addressed by two broad approaches: *feedback shaping* and *organizational structures*. Feedback shaping aims to shape the system objective such that the action of agents optimizing local objectives results in desirable system-level performance [6, 8]. Organizational structures decompose the agents themselves into roles that enable coordinated behavior [1, 29].

One particular research area in the credit assignment problem focuses upon ensuring that agents' objectives are aligned with the system objective (i.e., what is good for the agent is good for the system), and that the system objective is sensitive to agents' actions [27]. Providing agents with objectives that satisfy these two properties (formalized in [4, 27]) leads to a solution where key interactions among the agents are implicitly accounted for. A particular set of agent objectives that achieves these goals are the *difference objectives*, which are based on the difference between the actual performance of the system and the performance of a counter-factual system in which certain agents have been removed. Difference objectives have been extensively studied and applied to real world applications including air traffic control, multi-robot coordination, and resource allocation [3, 5, 26, 27].

The success of the difference objective approach in developing appropriate agent learning objectives suggests that the approach is applicable to complex system design where a structural credit assignment problem exists when designing individual components.

One implementation of this approach is based on coevolutionary algorithms, descibed next.

Coevolutionary Algorithms

Evolutionary Algorithms (EAs) are a class of stochastic population-based search algorithms which can often outperform classical optimization techniques, particularly in complex domains where gradient information is not available [13]. An evolutionary algorithm typically contains three basic mechanisms: solution generation, a mutation operator, and a selection operator. These mechanisms are used on an initial set of candidate solutions, or a population to generate new solutions and retain solutions that show improvement. Simple EAs are excellent tools, but need to be modified to be applicable to large multiagent search problems for distributed optimization. One such modification is *coevolution*, where multiple populations evolve simultaneously in order to develop policies for interacting agents.

Coevolution: Coevolutionary Algorithms (CEAs) are an extension of evolutionary algorithms and are often well-suited for multiagent coordination domains [12]. In a CEA, the fitness of an individual is based on its interactions with other agents it collaborates with. Thus, assessing the fitness of each agent is context-sensitive and subjective [19]. In *cooperative* coevolution, individuals succeed or fail as a team. This paper is focused on Cooperative Coevolutionary Algorithms (CCEAs) for designing optimized complex systems.

One of the key advantages to coevolution is that the algorithm only needs to search subspaces of the overall solution space, rather than the entire solution space. This reduced state space often makes the learning process simpler for the cooperating agents, because as each agent is only optimizing a portion of the overall system, they can focus on a projection of the overall solution space which is typically of lower dimensionality than the original solution space.

However, these simpler subspaces represent a large loss in information; the consequence of this is that the policies obtained by using these state projections are strongly influenced by other populations. The result is that agents evolve to partner well with a broad range of other agents, rather than evolving to form optimal partnerships [20]. Thus, in addition to trying to decrease the complexity of the learning process, research in coevolution aims to achieve optimal policies rather than stable ones.

Cooperative Coevolutionary Algorithms: Cooperative Coevolutionary Algorithms (CCEAs) are a natural approach in domains where agents need to develop local solutions (such as subsystem design), but the metric for success or failure is related to overall system performance [22]. In CCEAs, distinct populations evolve simultaneously, and agents from these populations collaborate to reach good system solutions. One issue with CCEAs is that they tend to favor stable solutions, rather than optimal solutions [28]. This phenomena occurs because the different evolving populations adapt to each other, rather than adapting to form an optimal policy. Another issue that arises with CCEAs is the problem of credit assignment. Since the agents succeed or fail as a team, the fitness of each agent becomes subjective and context-dependent (e.g. an agent might be a "good" agent,

but the agents it collaborates with are "bad," and the objective isn't reached. In this case, the "good" agent may be perceived as "bad") [28].

Difference Evaluation Function Theory: The agent-specific difference evaluation function is defined as:

$$D_i(z) = G(z) - G(z_{-i} + c_i) \tag{1}$$

where z is the overall system state, $G(z)$ is the system evaluation function, z_{-i} is the system state without the effects of agent i, and c_i is the *counterfactual* term used to replace agent i. Intuitively, the difference evaluation compares system performance with and without agent i, to approximate the agent's impact on overall system performance. Note that:

$$\frac{\partial G(z)}{\partial a_i} = \frac{\partial D_i(z)}{\partial a_i} \tag{2}$$

where a_i is the action taken by agent i. This means that any action an agent takes which increases the value of the difference evaluation also increases the value of the overall system performance. This property is termed *alignment*. Also note that the second term in Equation 23 removes the portions of the system evaluation which are not affected by agent i. This reduces noise in the feedback signal, meaning that difference evaluations are highly *sensitive* to the actions of an individual agent.

In addition to the theoretical properties of alignment and sensitivity, difference evaluations have been proven to increase the probability of finding optimal solutions in cases where the optimal Nash equilibrium is *deceptive*. In these cases, one agent deviating from the optimal strategy results in a large decrease in the overall system payoff, meaning that finding these Nash equilibria is typically extremely difficult.

METHODOLOGY

This paper demonstrates that, in a design problem, a team of autonomous agents can replicate or outperform a team of engineers. The first step is to define the design process for the agents as shown in Figure 2. To begin with, it is necessary to define the system level objectives and the system constraints. Then we select the team of agents, where each agent will be responsible for optimizing a specific subsytem. Secondly, using the different system-level objectives, it is necessary to define the overall system objective. The overall system objective will be used by the algorithm to measure the impact of the design concept for each agent team. Using CCEAs (cooperative coevolutionary algorithm), the agents will evaluate all the possible combinations of solutions and choose the best one. In this paper we will compare the final design of a system designed by a team of engineers against the design reached by a team of autonomous agents.

Formula SAE design problem

This paper will illustrate the proof-of-concept of the approach using the design of a formula SAE racing vehicle. Formula SAE is a collegiate design competition that requires students to design, build, test and, compete with an racing automobile [16]. Formula SAE works as a fictional company, where teams of students are contracted to create and build a functional small formula racing vehicle. The final design is tested based on a series of rules which ensure safety of all operations, and promote a design challenge for engineers.

The objective is to design a racing vehicle, which will win the acceleration event of a Formula SAE race. The acceleration event evaluates the car acceleration in a straight line on flat pavement. The course layout has a length of 75 m and 4.9 m wide from starting to finish line [16]. In this paper, we compare the design process of a Formula SAE engineering team against a team of autonomous design agents. The design process for the autonomous agents will follow the same design principles as the one followed by the team of engineers, using the parameters in Figure 1. We will set some customer requirements for the vehicle performance. Secondly, we will define the system-level objectives and constraints. Finally, the autonomous design teams (agents) will be defined and an algorithm will be implemented. The key factor of the presented design process will be the design agent's selection.

For the purpose of this project the system to be analyzed is going to be simplified. The design of the suspension system and steering system will not be analyzed in this document. The selection of components such as the differential, clutch and transmission will be ignored for this first part of the project. All the subsystems and components mentioned will be implemented as part of future work.

Formulating the System level objective

The first step is to investigate the form of a system-level design objective that ensures an intrinsically dependable and adaptable system directly from the design process. The key principle here is that the objective function should capture the designer's underlying preferences for the system while ensuring the design is both adaptable and dependable. To win the competition the system needs to maximize the vehicle acceleration and aerodynamic grip, and minimize weight, drag and the location of the center of gravity. Engineering teams are responsible for designing the system as shown in Figure 3. Consistent with the philosophy of engineering design, the goal is to make the decisions as accurately as possible without having to build prototypes or conduct costly testing.

Formulating the Agents as Design Teams

There are eight design teams (agents) each responsible for a subsystem in the overall design problem. These teams, as well as the design parameters they are responsible for, are given in

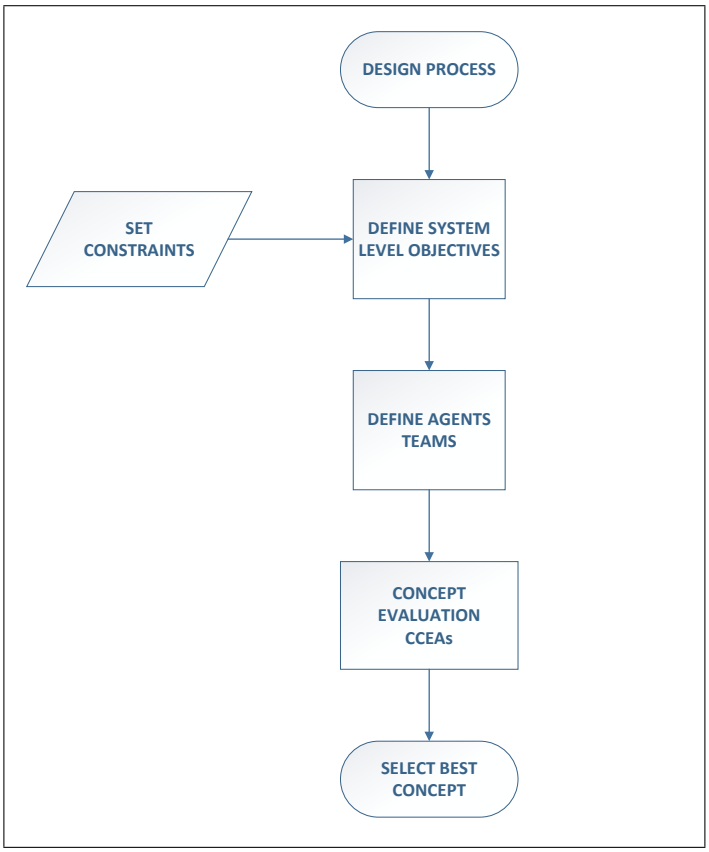

FIGURE 2: Design Process for Agents

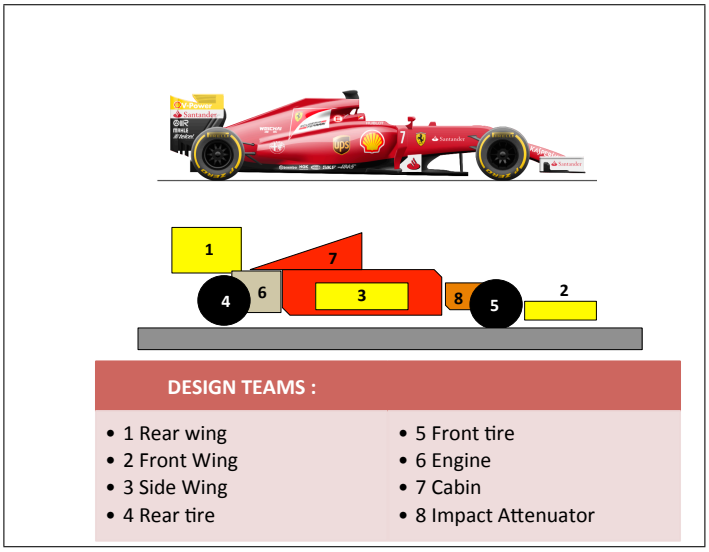

FIGURE 3: Racing vehicle Model

Table 1. Note that for each component, h corresponds to height, l corresponds to length, w corresponds to width, α corresponds to angle of attack, x corresponds to an x-position, y corresponds to a y-position, ρ corresponds to density, P corresponds to pressure, r corresponds to radius, m corresponds to mass, Φ corresponds to power, τ corresponds to torque, t corresponds to thickness, and E corresponds to a material's modulus of elasticity. Further, note that each team name has an abbreviation given in Table 1 to define variable naming conventions. So, for example, the height of the rear wing is denoted h_{rw}.

Continuous variables such as height are chosen from a constrained portion of $\mathbb{R}$ (constraints based on SAE competition rules), while discrete variables such as engine power are determined by choosing from a discrete list of available engines.

TABLE 1: Description of Design Teams (Agents)

Team Name	Continuous Parameters	Discrete Parameters
Rear Wing (rw)	h, l, w, α, x, y	ρ
Front Wing (fw)	h, l, w, α, x, y	ρ
Side Wings (sw)	h, l, w, α, x, y	ρ
Rear Tires (rt)	P, x	r, m
Front Tires (ft)	P, x	r, m
Engine (e)	x, y	Φ, l, h, τ
Cabin (c)	h, l, w, t, x, y	ρ
Impact Attenuator (ia)	h, l, w, x, y	ρ, E

For the purpose of this analysis, the customer requirement for the designed vehicle is to win the acceleration event of a Formula SAE race. The following assumptions will be used as the requirements for the system levels objectives and the environment:

1. The car's top velocity v_{car} is $26.8 m/s$ (60mph)
2. The car's engine speed ω_e is 3600 rpm
3. The density of air ρ_{air} during the race is $1.225 kg/m^3$

System level objectives

We now discuss the objectives to be optimized for the entire system in the following sections.

Mass: The first design objective is to minimize the mass of the car. The rear wing, front wings, side wings, and impact attenuator are all modeled as cuboids, and their mass is given by:

$$m = l \cdot w \cdot h \cdot \rho$$

The cabin is modeled as a cuboid shell with thickness t, and its mass is given by:

$$m_c = 2(h_c \cdot l_c \cdot t_c + h_c \cdot w_c \cdot t_c + l_c \cdot h_c \cdot t_c)\rho_c$$

The mass of the rear tires, front tires, and engine are defined once a set of tires and an engine are chosen. The total mass of the vehicle is thus:

$$m_{total} = m_{rw} + m_{fw} + 2 \cdot m_{sw} + 2 \cdot m_{rt} + 2 \cdot m_{ft} + m_e + m_c + m_{ia} \tag{3}$$

Note that as there are two side wings, two rear tires, and two front tires, these mass values are doubled in the overall mass calculation.

Center of Gravity Height: The second objective is to minimize the center of gravity height (CG_y) of the car, or to keep the center of gravity as low as possible. The y-position of the center of gravity is defined as:

$$CG_y = \frac{m_{rw}y_{rw} + m_{fw}y_{fw} + m_e y_e + m_c y_c + m_{ia}y_{ia}}{m_{total}} + \ldots + \frac{2(m_{sw}y_{sw} + m_{rt}r_{rt} + m_{ft}r_{ft})}{m_{total}} \tag{4}$$

Drag and Downforce: The third and fourth objectives are to minimize the overall drag of the vehicle and to maximize the downforce of the vehicle. We assume that the components which influence drag are the rear wing, front wing, side wings, and cabin. We also assume that only the wings influence vehicle downforce. We will first analyze the wings, and then the cabin.
The aspect ratio AR of a wing is defined as:

$$AR = \frac{w\cos\alpha}{l}$$

The lift coefficient C_l of a wing is defined as:

$$C_l = 2\pi\frac{AR}{AR+2}\alpha \tag{5}$$

The drag coefficient C_d of a wing is defined as:

$$C_d = \frac{C_l^2}{\pi AR} \tag{6}$$

The overall downforce F_d of a wing is given by:

$$F_d = \frac{1}{2}\alpha h w \rho_{air} v_{car}^2 C_l \tag{7}$$

The overall drag F_R of a wing is given by:

$$F_R = \frac{1}{2}\rho_{air} v_{car}^2 C_d w h \tag{8}$$

For the cabin, we assume a drag coefficient $C_{d,c}$ of 0.04 for a streamlined body [18].
The overall drag of the vehicle is thus:

$$F_{R,total} = F_{d,rw} + F_{d,fw} + 2F_{d,sw} + F_{d,c} \tag{9}$$

and the overall downforce of the vehicle is:

$$F_{d,total} = F_{R,rw} + F_{R,fw} + 2F_{R,sw} \tag{10}$$

Acceleration: The fifth objective of the design process is to maximize the acceleration of the car in the x-direction. The rolling resistance coefficient C of the car is given by:

$$C = 0.005 + \frac{1}{p}(0.01 + 0.0095 v_{car}^2) \tag{11}$$

where p is the tire pressure. The overall rolling resistance R_{roll} of the car is given by:

$$R_{roll} = \frac{C m_{tot} g}{r} \tag{12}$$

where g is the gravitational constant and r is the tire radius. Thus, the total resistance of the car R_{tot} is given by the sum of drag and rolling resistance:

$$R_{tot} = F_{d,total} + R_{roll} \tag{13}$$

The efficiency η of the engine is given by:

$$\eta = \frac{R_{tot} v_{car}}{\Phi_e} \tag{14}$$

The wheel force F_{wheels} at the rear tires is given by:

$$F_{wheels} = \frac{\tau_e \eta \omega_e}{r_{rt}\omega_{wheels}} \tag{15}$$

where ω_{wheels} is the rotational speed of the rear wheels, given by:

$$\omega_{wheels} = \frac{v_{car}}{r_{rt}} \tag{16}$$

We can thus find the acceleration of the car a_{car} as follows:

$$\sum F = m_{total} a_{car}$$
$$F_{wheels} - R_{total} = m_{total} a_{car}$$
$$a_{car} = \frac{F_{wheels} - R_{total}}{m_{total}} \tag{17}$$

Crash Force: The sixth objective of the design problem is to minimize the crash force of the car. The axial deformation δ of the impact attenuator is given by:

$$\delta = \frac{F_{crash} l_{ia}}{w_{ia} h_{ia} E_{ia}} \tag{18}$$

The crash force F_{crash} is defined as:

$$F_{crash} = \frac{m_{total} v_{car}^2}{2\delta} \tag{19}$$

Combining Equations 18 and 19 yields:

$$\begin{aligned} F_{crash} &= \frac{m_{total} v_{car}^2}{2\frac{F_{crash} l_{ia}}{w_{ia} h_{ia} E_{ia}}} \\ \Rightarrow F_{crash}(2F_{crash} l_{ia}) &= m_{total} v_{car}^2 w_{ia} h_{ia} E_{ia} \\ \Rightarrow F_{crash}^2 &= \frac{m_{total} v_{car}^2 w_{ia} h_{ia} E_{ia}}{2 l_{ia}} \\ \Rightarrow F_{crash} &= \sqrt{\frac{m_{total} v_{car}^2 w_{ia} h_{ia} E_{ia}}{2 l_{ia}}} \end{aligned} \tag{20}$$

Impact Attenuator Volume: The seventh and final objective of the design problem is to minimize the impact attenuator volume V_{ia}, given by:

$$V_{ia} = l_{ia} w_{ia} h_{ia} \tag{21}$$

Overall System Objective

In a Formula SAE competition the car prototype is judged in a number of different events. In this paper we are not replicating a Formula SAE competition, however, it is necessary to judge the design of the vehicle. A weighted linear sum is our approximation on how to judge the design with respect to its performance.

The overall system objective is given by a weighted linear combination of the individual objectives. Given a candidate design solution z, the system evaluation function $G(z)$ is defined as:

$$\begin{aligned} G(z) = &-w_m m_{total} - w_{CG} CG_y - w_R F_{R,total} + w_d F_{d,total} + \ldots \\ &+ w_a a_{car} - w_{crash} F_{crash} - w_{iav} V_{ia} \end{aligned} \tag{22}$$

where w_i is a weight corresponding to objective i.

Recall that the **agent-specific difference evaluation function** is defined as:

$$D_i(z) = G(z) - G(z_{-i} + c_i) \tag{23}$$

where z is the overall system state, $G(z)$ is the system evaluation function, z_{-i} is the system state without the effects of agent i, and c_i is the *counterfactual* term used to replace agent i. Intuitively, the difference evaluation compares system performance with and without agent i, to approximate the agent's impact on overall system performance.

Constraints

The constraints used for the vehicle were set according to the Rules of the 2016 Formula SAE Rules [16]. The SAE rules present the competition regulations technical and design requirements. The SAE rules were use to define the minimal dimensions and the areas where the structural components are allowed to allocated.

CCEAs Implementation

The approach to optimizing the vehicle design using cooperative coevolution is shown in Algorithm 1. Initially, N populations are seeded with k random solutions. In this case, there are 8 populations, one for each subsystem agent (rear wings, front wings, etc.). In each generation, each population creates mutated solutions, and then the solutions are used to create teams, where a team consists of an entire vehicle design. The solution presented by the team is then evaluated, and each member of the team is assigned a fitness score. Once each member of each population has been assigned a fitness, solutions in each population are selected to survive to the next generation. Each of these evolutionary mechanisms are explained in the following paragraphs.

A "solution" in the Cooperative Coevolutionary Algorithms consists of two elements: a continuous element and a discrete element. For any given team (optimized by a single population), the continuous element of the solution contains an array of values corresponding to that team's subsystem. For example, for the rear wing team, the continuous portion of the solution is an array of the form $\{h, l, w, \alpha, x, y\}$. The discrete element of the solution contains an array of choices corresponding to that team's subsystem. For the rear wing team, the discrete portion of the solution is of the form $\{\rho\}$, where the density ρ was chosen from a list of available materials for wing construction. A random solution is chosen by drawing the continuous variables from a uniform probability distribution, and drawing discrete variables where each discrete choice has an equal probability of selection. Each population (corresponding to different design subsystems) is initially populated with random solutions.

For mutation, each population of size k is doubled in size to $2k$ solutions. Each solution in the original population is copied, and then mutated to create a child solution. For the continuous portion of the solution, mutation is carried out by first adding a value drawn from a Gaussian distribution $N(\mu = 0, \sigma = 0.001)$ to each element, and then ensuring the resulting value is still within the allowable constraints. For example, if a mutated parameter is $a + \varepsilon$, but the maximum value (based on vehicle constraints) that the parameter may take is a, then the value is changed to a.

For team formation, one solution is drawn from each population to form a complete vehicle design. Each solution in a population has an equal probability of being selected for a team, and each solution is used only once (i.e, if each population has a size of $2k$, then $2k$ teams are tested).

For fitness assignment (line 11 in Algorithm 1), we test two fitness assignment operators. First, we use the global evaluation $G(z)$ to assign fitness to each agent in a team. This means that the performance of each subsystem design is assigned using the overall system performance. Next, we use the difference evaluation $D_i(z)$ to assign fitness to each agent. In this case, we assign a counterfactual of 0, meaning we analyze performance of the car if a component was removed from the design. This provides an estimate of the impact on the overall system performance provided by a single component.

For selection, we use binary tournament selection, which reduces a population of size $2k$ to size k using the following procedure. Two solutions are drawn from the population (each solution may be drawn only once). The solution with the higher fitness value is returned to the population, and the solution with the lower fitness value is discarded. Binary tournament selection ensures that the best solution in the population is retained and that the worst solution in the population is discarded. For all other solutions in the population, their probability of survival increases with their fitness value.

```
1  Initialize N populations of k subsystem solutions
2  foreach Generation do
3      foreach Population do
4          produce k cloned solutions
5          mutate cloned solutions
6      end
7      for i = 1 → 2k do
8          randomly select one agent (previously
           unselected) from each population
9          add selected agents to team T_i
10         simulate T_i in domain
11         assign fitness to each agent in T_i
12     end
13     foreach Population do
14         select k solutions using binary tournament
           selection
15     end
16 end
```

Algorithm 1: Cooperative Coevolutionary Algorithm

RESULTS

The results from the Cooperative Coevolutionary Algorithm were validated by comparing them to a real Formula SAE vehicle. The current formula SAE Michigan champion since 2010 is the Global Formula Racing GFR [14]. GFR is a Formula SAE team formed by an international cooperation between the BA Racing Team from Duale Hochschule Baden-Württemberg-Ravensburg (DHBW), Germany, and the Beaver Racing Team from Oregon State University (OSU), USA. GFR team has proven to be the best student engineering team, winning more than 15 competitions worldwide since 2010. The results from the coevolutionary algorithm will be compared to the GFR 2013 combustion car.

TABLE 2: Weights for Overall System Objective

Objective	Weight
Mass	15
CG_y	5
Drag	3
Downforce	2.5
Acceleration	10
Crash Force	10
IA Volume	1

For the simulation run the population size was set to 50 and the number of generations for evolution 10,000. The weights for the Overall System Objective (Eqn 22) are defined in Table 2.

Since the primary objective is to reduce the overall mass of the vehicle while maximizing the car's acceleration, these objectives have the higher weight. Future work will analyze the effects of weight variation between systems objectives, and modify the Overall System Objective to enclose the entire complexity of the system. For example, if the weight value for the downforce objective is increased, the design agents will create larger wings. However, the use of large wings will increase the drag forces on the vehicle, thus causing a lower overall system performance. The current model does not include the analysis of the dynamics behind suspension or lateral accelerations because the vehicle is only moving on a smooth, straight track. In a real model, where the vehicle would be turning at high speeds, downforce plays an important role and would be considered in future work.

Figure 4 shows how the system performance increases for $G(z)$ and $D_i(z)$ as the number of generations increases. The system performance has negative values because five of the seven objectives need to be minimized.

Difference Reward $D_i(z)$ usually provides better performance than system evaluation function $G(z)$. This is due to the fact that each team member is given better feedback on their performance. $G(z)$ outperforms $D_i(z)$ when weights are chosen that significantly favor acceleration and crash forces being optimized. This is because these parameters are the most coupled as they depend on the mass of each component as well as engine specs. So basically every team member affects these parameters; and

$D_i(z)$ has problems when there is extremely high coupling between agents.

More importantly, in these cases where $D_i(z)$ struggles, we see the effect of one agent being very good with one team but very bad with another team. This type of behavior is also difficult for $D_i(z)$ to handle, because it will only provide good feedback to an agent if its teammates are reasonably well-suited to work with that agent. For example, an agent that works extremely well with engine *A* but extremely poorly with engine *B* will get a bad difference evaluation if it is paired with a teammate which selects engine *B*, even if it is actually a decent solution.

The result from the design process is shown in Table 3, where we compare a solution found using the CCEA with the GFR solution. The design of the agents does better on all but one objective, which is drag.

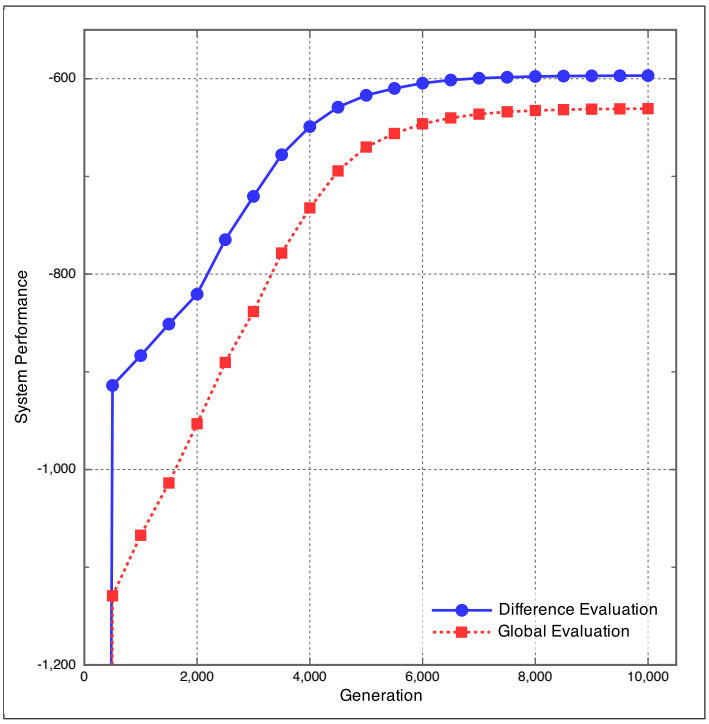

FIGURE 4: Preliminary results: difference evaluations provide better designs than global evaluatons.

CONCLUSIONS

The long-term goal of this project is to enable new design paradigms for complex systems to ensure that design space exploration, system architecture selection, and system integration are conducted in a way to produce a certifiably *dependable* and *adaptable* system meeting high-level design objectives.

The work done in this paper is primary evidence that distributed artificial intelligence can be used in design processes by splitting up the overall system into specific teams.

TABLE 3: Comparison of design found using CCEA vs. the SAE solution.

Objective	CCEA Solution	SAE Solution
Mass	**34.4210**	34.8772
CG_y	**0.1919**	0.2778
Drag	281.7162	**217.1085**
Downforce	**390.5887**	318.7476
Acceleration	**0.3336**	0.2497
Crash Force	**18.0697**	40.9255
IA Volume	**0.0135**	0.00864

More specifically, the results from Table 3 ilustrate that a team of autonomous agents using a cooperative coevolutionary algorithm (CCEA) can effectively design a Formula racing vehicle. The CCEA results in better performance than the Global Furmula Racing (GFR) design on 6 objectives, and is worse on 1 objective (drag). One of the reasons why the autonomous agents have a worse performance on Drag could be the selected weight for the system objectives. Drag and Downforce objectives are highly related, if the weight value for the downforce objective is increased, the design agents will create larger wings. However, the use of large wings will increase the drag forces on the vehicle, thus causing a lower overall system performance. In a higher fidelity model, where the vehicle would be turning at high speeds, downforce and drag will play an important role.

FUTURE WORK

The next step in this research is to obtain results increasing the complexity of the system. In this paper the case study was simple, but a real Formula racing vehicle needs to consider all the dynamics between the system and the environment. The system needs to simulate the lateral accelerations caused while the vehicle is turning at a high speed. Different types of mechanical and electrical components most be included using a large number agents. Suspension and brake systems needs to be included as part of the engineering requirements. The system objective must also consider the cost of manufacturing operations and the cost of the components. The weights used and the linear form of Overall System Objective will be analyzed in more detail. As the complexity of the system increases, it will be necessary to analyze how the team of agents behave with multiobjectives.

ACKNOWLEDGMENT

This research is supported by the National Science Foundation award number CMMI-1363411. Any opinions or findings of this work are the responsibility of the authors, and do not necessarily reflect the views of the sponsors or collaborators. Special thanks to Universidad San Francisco de Quito for supporting one of the authors' graduate studies; and to the GFR team for the constant support.

REFERENCES

[1] S. Abdallah and V. Lesser. Multiagent reinforcement learning and self-organization in a network of agents. *In Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multi-Agent Systems*.

[2] A. Agogino and K. Tumer. Unifying temporal and structural credit assignment problems. *In Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems*.

[3] A. K. Agogino and K. Tumer. Distributed agent-based air traffic flow management. *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multi-Agent Systems*.

[4] A. K. Agogino and K. Tumer. Efficient evaluation functions for evolving coordination. *Evolutionary Computation*.

[5] A. K. Agogino and K. Tumer. A multiagent approach to managing air traffic flow. *Autonomous Agents and MultiAgent Systems*.

[6] M. Babes, E. M. de Cote, and M. L. Littman. Social reward shaping in the prisoner's dilemma. *In Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, 3(8):1389—-1392, 2008.

[7] O. Brown and P.Erenko. Application of value-centric design to space architectures: the case of fractionated spacecraft. *In Proc. of AIAA Space 2008 Conference and Exposition, San Diego, CA, USA* .

[8] O. Buffet, A. Dutech, and F. Charpillet. Shaping multiagent systems with gradient rein- forcement learning. *Autonomous Agents and Multi-Agent Systems*.

[9] L. Busoniu, R. Babuska, and B. D. Schutter. A comprehensive survey of multiagent rein- forcement learning. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions*, 38(2):156.

[10] P. Collopy. Economic-based distributed optimal design. *AIAA Paper* .

[11] P. Collopy and R. Horton. Value Modeling for technology evaluation. *AIAA paper*.

[12] S. G. Ficici, O. Melnik, and J. Pollack. A game-theoretic and dynamical-systems analysis of selection methods in coevolution, 2005.

[13] D. Fogel. An introduction to simulated evolutionary optimization. *Neural Networks, IEEE Transactions on*, 5(1):3 –14, jan 1994.

[14] Global Formula Racing. GFR, 2016. URL `http://www.global-formula-racing.com`.

[15] G. A. Hazelrigg. A framework for decision-based engineering design. *Journal of Mechanical Design*, 4(120):653–658.

[16] S. International. Formula SAE. Rules, 2016. URL `http://www.fsaeonline.com`.

[17] B. Mullins. Defense Acquisitions: Assessments of selected weapon programs. Technical Report GAO-08-467 SP, US Government General Accountability Office, 2008.

[18] NASA. Shape Effects on Drag. Shape Effects on Drag, May 2015. URL `http://www.grc.nasa.gov/airplane.htmlm`.

[19] L. Panait and S. Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11(3):387–434, 2005.

[20] L. Panait, S. Luke, and R. P. Wiegand. Biasing coevolutionary search for optimal multiagent behaviors. *IEEE Transactions on Evolutionary Computation*, 10(6):629–645, 2006.

[21] S. Paquet and L. Tobin. An online pomdp algorithm for complex multiagent environments. *In Proceedings of the Autonomous Agents and Multiagent Systems Conference*.

[22] M. A. Potter and K. A. De Jong. Evolving Neural Networks with Collaborative Species. *Computer Simulation Conference*, 1995.

[23] P. Stone. Layered learning in multi-agent systems:a winning approach to robotic soccer. *MIT Press*.

[24] P. Stone and M. Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345.

[25] K. Tumer and A. K. Agogino. Multiagent learning for black box system reward functions. *Advances in Complex Systems*.

[26] K. Tumer and N. Khani. Learning from actions not taken in multiagent systems. *Advances in Complex Systems*.

[27] K. Tumer and D. Wolpert. Collectives and the design of complex systems. *Springer*.

[28] R. P. Wiegand, K. A. D. Jong, and W. C. Liles. Modeling variation in cooperative coevolution using evolutionary game theory, 2002.

[29] C. Zhang, S. Abdallah, and V. Lesser. Integrating organizational control into multi-agent learning. *In Proceedings of the Conference on Autonomous Agents and Multiagent Systems*.