AN ABSTRACT OF THE THESIS OF

Nicolás Francisco Soria Zurita for the degree of Master of Science in Mechanical Engineering presented on June 9, 2016.

Title: Design of Complex Engineered Systems Using Multiagent Coordination

Abstract approved: _____

Irem Y. Tumer

This thesis is the combination of two research publications working toward a unified strategy in which the design of complex engineered systems can be completed using a multiagent coordination approach. Current engineered system modeling techniques segment large complex models into multiple groups to be simulated independently. These methods restrict the evaluations of such complex systems, as their failure properties are typically unknown until they are experienced in operation. In an effort to help engineers to design complex engineered systems, this research proposes that a distributed yet non-legislated approach can be used in the design processes by splitting up the overall system into specific teams. The approach specifically hypothesizes that multiagent credit assignment can be used to effectively determine how to properly incentivize subsystem designers so that the global set of system-level objectives can be achieved.

The first publication presents a multiagent systems based approach for designing a self-replicating robotic manufacturing factory in space. The simulation in this work is able to present the coordination of the agents during the construction of the factory as the parameters of the learning algorithm are changed. The results show the advantage of using a learning algorithm to design a large system. The second publication presents a hybrid approach to design complex engineered systems, providing a method in which design decisions can be reconciled without the need for either detailed interaction models or external

legislating mechanisms. The results of this paper demonstrate that a team of autonomous agents using a cooperative coevolutionary algorithm can effectively design a complex engineered system.

Each publication utilized a system model to illustrate and simulate the methods and potential results. By designing complex systems with a multiagent coordination approach, a design methodology can be developed in an effort to reduce design uncertainty and provide mechanisms through which the system level impact of decisions can be estimated without explicitly modeling such interactions.

Design of Complex Engineered Systems Using Multiagent Coordination

by

Nicolás Francisco Soria Zurita

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented June 9, 2016
Commencement June 2017

Master of Science thesis of Nicolás Francisco Soria Zurita presented on June 9, 2016.

APPROVED:

Major Professor, representing Mechanical Engineering

Head of the School of Mechanical, Industrial, and Manufacturing Engineering

---

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Nicolás Francisco Soria Zurita , Author

PUBLICATION THESIS OPTION

This thesis is presented in accordance with the Manuscript Document Format option. Two manuscripts are provided. The first was accepted for publication in the ASME 2015 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference (IDETC/CIE 2015), and the second was accepted for publication in the ASME 2016 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference (IDETC/CIE 2016). Journal versions of each publication are in preparation for submission in Summer, 2016.

ACKNOWLEDGEMENTS

I want to express my gratitude and appreciation to my advisor Dr. Irem Y. Tumer. Her continued support has made it possible to expand my knowledge and expertise. She challenged and developed my critical thinking, while guiding me on the development of this research project.

I would also like to thank Dr. Mitchell Colby, Dr. Chris Hoyle and Dr. Kagan Tumer. Like Dr. Tumer, they inspired many of the thoughts and ideas that have defined my research, while providing constructive criticism and critical insights. I would like to thank all the professors at Oregon State University, my colleagues at Design Engineering Lab. Thanks to their help, support and motivation I was able to complete all the analysis of this project. I want to give a special thanks to my dear friend Jenny Urbina. Jenny provided me support, persisting enthusiasm, and motivation during the development of the presented project.

Last but not least, I would like to thank my parents, Diego y Kathya; to my sister Nia and the rest of my family. Your support and guidance has been truly immeasurable and I appreciate all that you have done for me.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

Page

LIST OF FIGURES

LIST OF TABLES

Chapter 1: Introduction

A growing class of complex systems, such as state-of-the-art aircraft, advanced power systems, unmanned aerial vehicles, and autonomous automobiles, is required to operate dependably in an ever widening variety of environmental conditions, over a wide range of missions. Such systems must be cost-effective while being dependable in potentially extreme conditions and adaptable to a given environment.

In the early stages of conceptual design, engineers tasked with designing complex engineered systems do not have detailed models that ensure that design space exploration, system architecture selection, and system integration are conducted in a way to produce systems that meet high-level design objectives. This research seeks to address this issue, creating innovation at the intersection of multiple disciplines, requiring expertise in complex system design, multiagent coordination, uncertainty quantification, and multi-objective optimization.

1.1 Motivation

In complex engineered systems, complexity may arise by design. For example, during the design of an aircraft the complexity is triggered by software & hardware integration and the use of new materials. Complexity can also be caused by the systems operation. This is the case of autonomous vehicles, where the vehicle must drive/operate on different type of environments. In either case, the root cause of complexity is the same: the unpredictable manner in which interactions among components (or between components and the environment) modifies system behavior.

Traditionally, two broadly different approaches are used to handle such complexity: (i) Approaches based on a centralized design approach where the impacts of all potential system states and behaviors resulting from design decisions must be accurately modeled. (ii) Approaches based on externally legislating design decisions, which avoid such difficulties,

but at the cost of expensive (in time and money) external mechanisms to determine the trade-offs among competing design decisions.

The work presented in this research 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 work 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).

1.2 Definitions

Many of the terms utilized within this research have alternate meanings when used in different domains. Therefore, it is necessary to define several terms which are utilized prominently. Table 1.1 provides a list of the definitions as they are used in this work. Each definition is supported within the engineering literature.

Table 1.1: Terminology Utilized Throughout This Work

Term	**Definition**
Flow	A *Flow* is an energy, material, or signal acted on by a function.
Function	A *Function* is the transformation of a flow.
Behvaior	*Behavior* is how a function is achieved.
Performance	*Performance* is a specific manifestation of behavior.
Failure	A *failure* is the degradation of performance.
Component	A *component* is the lowest level of a system with behavior.
System	A *system* is a collection of elements, components or otherwise, which combine to perform a function.
Complex system	A *complex system* is a system displaying emergent interactions that are not separable without altering system architecture.
Complex engineered system	A *complex engineered system* (CES) is an artificial complex system comprised of interdependent engineered systems performing a function.
Algortithm	An *algortithm* is a self-contained step by step set of operations to be performed.
Agent	*Agent* is an autonomous entity which observes and acts upon an environment and directs its activity towards achieving goals.
Multiagent system	*Multiagent system* is a system composed of multiple interacting agents within an environment.

1.3 Contributions

This thesis consists of two manuscripts that describe a process for implementing a methodology for making design decisions for complex systems. The presented methodology uses a multiagent coordination approach to design complex engineered systems. The contributions to the field are as follows:

1. *The implementation of a learning algorithm within the design optimization process guarantee the identification of the preeminent solution.*

   The first contribution of this research specifically addresses the performance of autonomous agents with pre-programmed vs. learned behavior. In the first paper the implementation of a learning algorithm was shown to work much better than a pre-programmed behavior. The second paper demonstrates that the implementation of a cooperative coevolutionary algorithm can complete the design of complex system using the collaborative work of a team of autonomous agents. It was corroborated that a complex engineered system can effectively be designed using agents with learned behavior.

2. *The application of a cooperative coevolutionary algorithm and difference evaluation in the design methodology guarantees the collaboration between agents inside a design team, and approximate the agent's impact on the overall system performance.*
   The second contribution addresses the multiagent coordination problem translated to the complex engineered system domain. The results from the case studies provide preliminary evidence that multiagent coordination can be used in design processes by splitting up the overall system into specific teams. Specifically, cooperative coevolutionary agents from distinct populations collaborate to reach good system solutions. However, one issue that arises is the problem of credit assignment. One good agent can be collaborating with a poor team, as a result the desired objective will not be satisfied. Implementing a difference evaluation function allowed autonomous agents to measure the impacts of their individual design decisions (choices) in the team performance for system.

3. *The application of multiagent coordination in the design methodology certifies the collaboration of the teams of autonomous agents as designers, and guarantees the success of the system objective.*
   The third contribution addresses the translation of the multiagent coordination problem to the design of complex engineered systems. In this research teams of autonomous agents were responsible for the design of different subsystems inside a complex system. Using the cooperative coevolutionary algorithm, the team of agents demonstrated that all agents are able to find an appropriate level of tradeoffs between the team members (subsystems) to satisfy the general system

This research provides evidence demonstrating that a multiagent coordination approach can be used to design a complex engineered system. It was corroborated that a team of autonomous agents can design a system through the use of difference evaluation functions combined with cooperative coevolutionary algorithms. The preliminary results demonstrate that a team of autonomous agents using a cooperative coevolutionary algorithm (CCEA) can effectively optimize a multiobjective design problem. By demonstrating compatibility and performance improvements, providing a theoretical analysis, and providing a methodology for implementation, we demonstrate the potential practicality of designing complex engineered systems using multiagent coordination.

Chapter 2: Background

Before proceeding, a discussion on what is a *complex engineered systems design* and how teams of engineers design such systems is included.

The following sections present information related to the design of complex engineered systems. Also included is a discussion on *Integrated Concurrent Engineering (ICE)* and an example of how *Team X* uses *ICE* to design complex engineered systems. The concepts of *Multiobjective Optimization*, *Multidisciplinary Design Optimization*, and *Problem Formulation*, which are important methods and concepts to this research are also presented within this section.

2.1 Complex Systems

Complex systems represent group of systems with a set of important features. The distinction between a *complex* [8] and *complicated* [30] system frequently leads to misperception about the nature of being complex. A *complicated* system is one that may be difficult to build, have a large number of components, or may be enormous [30]. While in a *complex* system, complexity arises as result of the system interactions between sub-functions. The system interactions are not separable so long as system behavior and function is retained [8]. Numerous aspects of biology and biological processes could be considered complex systems. Ecosystem interactions, brain chemistry and structure, and immune responses, to name a few, are complex systems [43]. Similarly, the interaction of many social groups could be considered a complex system as well. The existence of a group structure that is dependent on individuals belonging to and interacting with multiple groups may make these types of interactions complex.

The main problem dealing with complex engineered systems is the difficulty with modeling all the interactions within the system. Complex engineered systems share properties, interactions, and even emergent behaviors between components that cannot be fully un-

derstood or modeled [65, 42]. However, some authors argue that even very complicated systems such as aircraft carriers are not actually complex [49]. This research assumes that complex engineered systems are indeed complex systems.

Furthermore, as an engineered system is an organized combination of hardware, software, people, policies, and procedures, the unpredictable interacting nature of all the adove mentioned elements creates unexpected interactions within the system architecture, thereby making it *complex*. The terminology, 'complex engineered systems', is used to make a differentiation from other complex systems (such as biological ones) to emphasize that they are artificial, intended to perform specific functions, and comprised of interdependent engineered systems. Various examples include: aircraft, process plants, and satellites [4].

Selection of a 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 is the over-emphasis on requirement satisfaction for evaluating design alternatives [53]. 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.

The key concept is that many designs meet requirements; however, the design which maximizes design value is *preferred*. This approach to design can be applied to complex systems by decomposing the system-value model to individual components. This allows concurrent design that carries guarantees that optimizing an individual design goal will optimize the system-level goal simultaneously. Currently, the impact of component behavior upon system performance is quantified using sensitivity analysis [17, 61]. In this approach, components are broken into their attributes which are then perturbed to analyze their impact on the system as a whole; however, this analysis is performed after the system is composed and not concurrently with system design. This approach is typically not feasible in complex system design, where component interactions are difficult to quantify and model.

2.2 Integrated Concurrent Engineering

Integrated Concurrent Engineering (ICE) represents a collection of practices that attempts to eradicate inefficiencies in conceptual design and reorganize the process of sharing information inside a design team. ICE uses: a combination of expert designers; advanced modeling, visualization and analysis tools; social processes, and a specialized design facility; to create preliminary designs for complex engineered systems.

When compared with a traditional sequential engineering method, ICE users reduce project schedule by several orders of magnitude, while substantially improving design cost and maintaining quality standards. Within a very short period of time, ICE allows engineers to consider, implement, evaluate, accept and/or reject numerous ideas, with a relatively high level of fidelity [74]. Traditional design approaches constrain interdisciplinary trades because of a deficiency of communication among team members. Information is often scattered throughout the project team, meaning those seeking data on a particular subject have no central location to search [63]. As a result, engineers spend a significant amount of time searching or recreating information that already exists; rather than developing new information or data.

Primarily, ICE methodology addresses this problem by:

- Boosting communication between subsystem teams
- Centralizing information storage
- Providing a universal interface for parameter trading
- Stimulating multidisciplinary trades

The ICE structure allows teams to work independently on problems local to a subsystem and to coordinate effectively on issues that affect other teams. Projects using ICE methodology are more flexible and can quickly adapt to changes in top-level requirements. As a result of the combination of all this factors, engineering teams that work with ICE can successfully complete rapid trades among complex multidisciplinary subsystems.

The format of an ICE Team vastly increases the efficiency of the design process. Figure 2.1 highlights the advantage of a concurrent engineering process when compared with the traditional sequential approach.

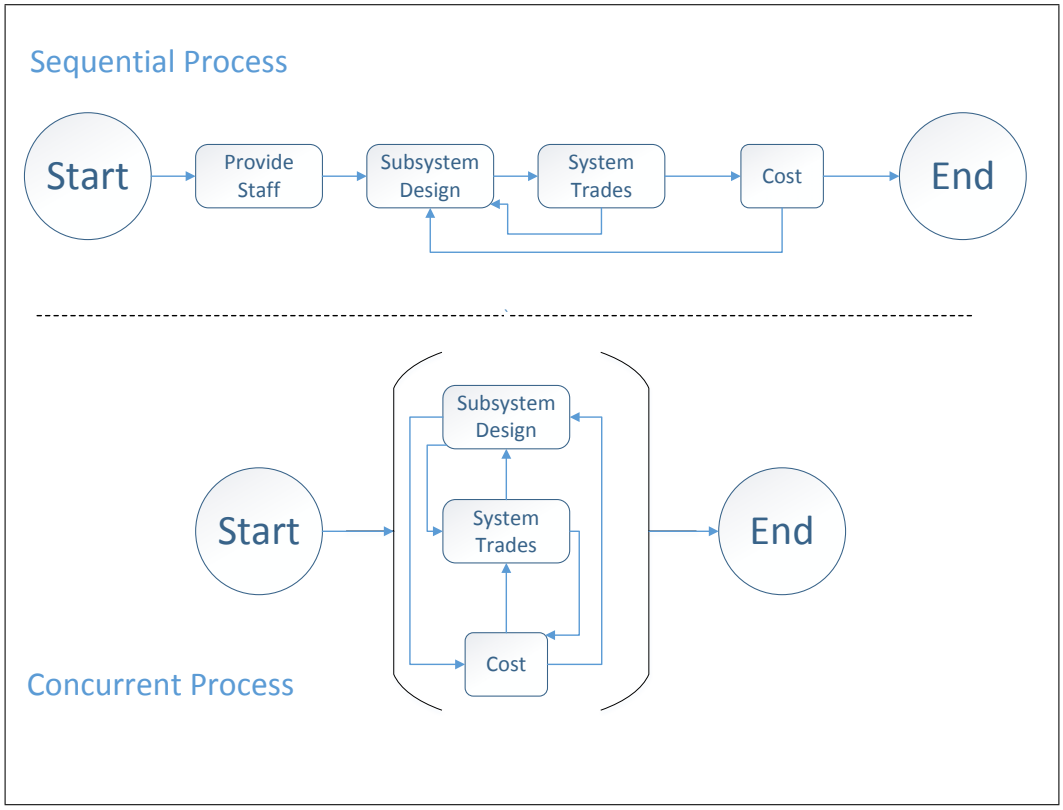

Figure 2.1: Difference between Sequential & Concurrent Design Process

Innovators in ICE methodology are primarily in the aerospace industry such as Boeing, where several closely related methods are termed ICE, Extreme Collaboration, Concurrent Design Engineering, or Radical Collocation [44]. Whereas traditional engineering superficially resembles a government bureaucracy, ICE performs the same work in an environment similar to NASA's Shuttle Mission Control operations.

2.2.1 Advanced Project Development Team

One of the most experienced team of engineers working with ICE can be found at NASA Jet Propulsion Laboratory (JPL). JPL is the headquarters of the *Advanced Project Development Team*, conventionally known as *Team - X*.

Team - X completes early-phase design projects in less than one-tenth the time of the previous process at JPL , and for less than one third of the variable cost [10]. Team - X begun operations in April of 1995 at the Jet Propulsion Laboratory (JPL).

The team has built a reputation behind his name because they were able to revolutionize the design of Complex Systems. They improved the speed and quality of JPL's new mission concepts; additionally it created a reusable study process with dedicated facilities, equipment, procedures, and tools. They were also responsible for developing a database for the initial mission requirements, which can be easily updated and electronically transferred for use in subsequent project phases.

Although there are continuous efforts to increase and improve the quality of designs and the generality of their method, Team - X product results is good enough that outside investigators choose to purchase their services about fifty times a year [10]. Team - X is in heavy demand in the competitive market for mission design services, and their successful concept designs have brought hundreds of millions of dollars in business to JPL and its suppliers [64].

Team - X is formed by a group of experienced engineers who work in parallel to develop and evaluate a system-level design such as spacecraft's. The team includes about eighteen domain experts, a facilitator, and a customer representative. The team leader divides the project in several groups that are responsible of the design decisions for each subsystem such as: cost estimation, telecommunications, hardware, mission design, programmatics, etc. Each subsystem leader is an *expert* in his or her dedicated field, and brings considerable technical proficiency to the team. Each Team - X chair designs a component of mission function, design form, or anticipated behavior as Figure: 2.2 illustrates. They coordinate using four independent processes: Facilitator- mediated tracking of design conformance to goals; Sidebar agreements on design trades; functional review of goals feasibility; and automatic data sharing of networked spreadsheets.

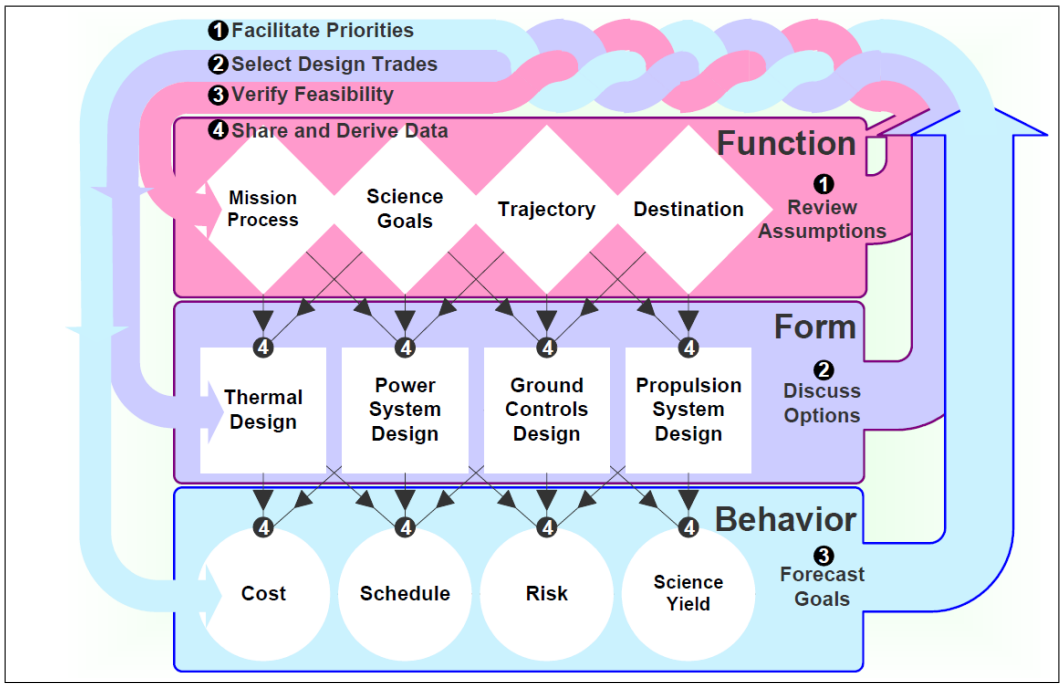

Figure 2.2: Team - X Schematic [10]

While the majority of engineering teams of this size employ a multilevel management hierarchy, Team - X is a much more flat and broad organization [10]. The team's facilitator focuses group attention on particular problems, lateral conversations are generated in which some discipline specialists resolve the problem of shared interest, and directs attention of individuals and the group to newly emerging information. A customer representative has the final authority on decisions that impact the achievement of the project's scientific goals.

Team - X members are selected for their technical skills, experience, and their independent ability to work effectively in the informal, superficially chaotic, high-pressure conditions. Partly because they are so psychologically demanding, Team-X limits design sessions to three hours [28].

Figure 2.3 illustrates an example on how Team - X is organized inside the war room. In war rooms, team member's work together synchronously in all phases of a project, engineers work closely together using a variety of computer technologies, including networked computers and public displays. The objective of the war room is to maximize communica-

tion and information flow between team members.

The implementation of Team -X in the of War Room helps achieve the design complex systems with a balance between electronic and social networks. However, how to optimize the flow of information between them is an open question [44]. The implementation of auxiliary autonomous tools for sharing information might be helpful and beneficial for the team. It may relieve the personal stress reported by team members, and free up time for more human networking and design decision making.

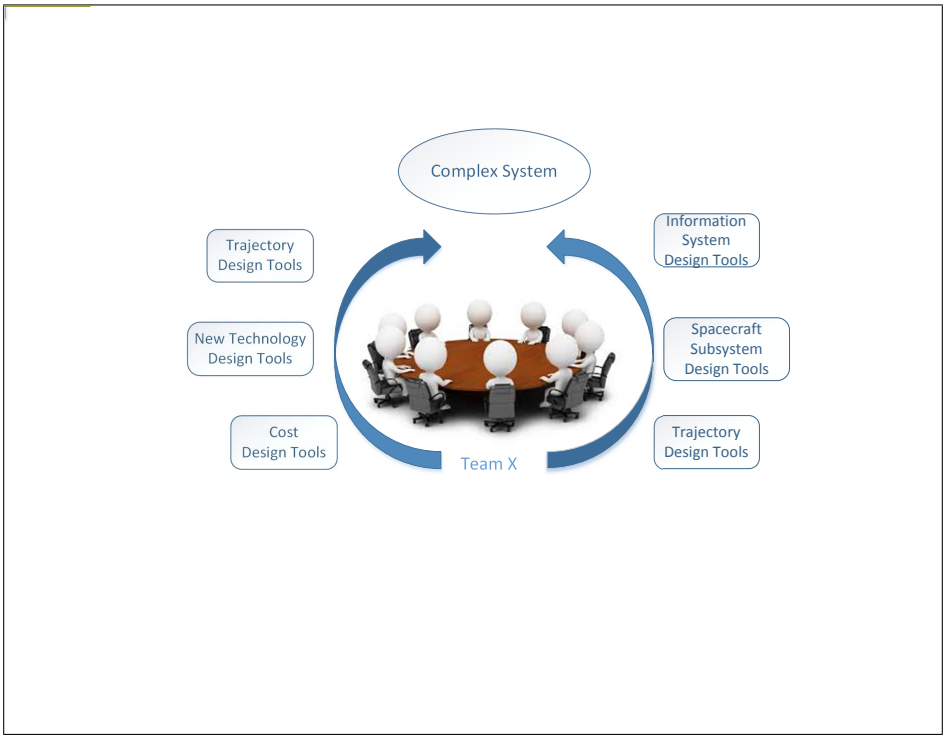

Figure 2.3: Team - X War Room [74]

The purpose of this research is to assist and help Team - X with the design decision process. This research will set the preliminary work in the implementation of a multiagent coordination approach to design of complex engineered systems. Team - X would be able to use this new methodology as an auxiliary tool for their design decision process.

2.3 Multiobjective Optimization

In system design, multiple criteria, such as cost, safety, and performance, are generally considered in the process. This represents the multi-objective design optimization (MOO) problem [12, 34, 73, 85]. General methodologies exist for solving MOO problems [62, 70]; the distinguishing feature of a MOO problem is that, in general, one cannot identify a single solution that simultaneously optimizes each objective. In the design objective space, the point obtained by solving the k single-criteria optimization problems individually is called the *utopia* (ideal) point. The utopia point is generally not achievable due to conflicts among the multiple design criteria. A design point is said to be a Pareto solution [58] if there exists no other feasible design that would improve some design objectives without sacrifice in at least one other criterion/objective. The Pareto frontier consists of Pareto solutions in the objective space such that a design criterion cannot be further improved without sacrifice to another criterion. Various methods for finding the Pareto frontier have been developed, such as the weighted-sum method [15], compromise programming [12], and the Normal Boundary Intersection (NBI) method [19]. Work exists in identifying the Pareto frontier when uncertainty exists in the multiple objectives, either by treating the mean and variance of each objective as separate objectives [46] or by computing an expected single attribute utility for each objective [33].

In this research the ultimate goal is to optimize a set of multiple objectives, which are the one of the distinguish characteristic of a complex system. This research will enable the identification of the Pareto frontier and allow meaningful trade-offs in problems with non-convex objective functions, utilizing agent-level information.

2.4 Multidisciplinary Design Optimization

Multidisciplinary Design Optimization is an optimization method used in engineering problems that involve multiple subsystems [63]. The advances in computing power have increased to popularity of this type of optimization methodologies. A number of techniques have emerged in an attempt to integrate both system decomposition and optimization such as Collaborative Optimization and Bi-level Integrated System Synthesis.

Collaborative Optimization (CO) is a multidisciplinary design optimization technique, developed at Stanford University, that divides a problem along disciplinary lines into sub-problems. Each sub-problem is then optimized so that the difference between the achievable subsystem response and target variables established by the system optimizer is minimized [5, 38].This methodology is powerful and effective for problems with well-defined disciplinary boundaries, a large number of shared variables and calculations, and a minimum of interdisciplinary coupling. However, on the downside CO leads to setups with high dimensionality, which requires high processing power. CO has been successfully applied to a number of different engineering problems typically in the area of vehicle design.

Bi-level Integrated System Synthesis (BLISS) is a multidisciplinary design optimization technique, developed at NASA Langley Research Center [67, 69, 68], that uses hierarchical decomposition. BLISS works similar to CO, where the goal is to optimize distributed engineering systems developed by specialty groups who work concurrently to solve a design problem. The methodology differs from CO because preference weights are used for multi-objective optimization at the subsystem level. The Constraints and coupling variables are also controlled fairly differently. The design parameters in BLISS are divided into three groupings. In the first group the variables are optimized at the local level and are found only within each of the subsystems. The second group contains variables which are outputs by one subsystem and are used as inputs for a different subsystem. Finally the third group contains the system-level design variables, which are shared by at least two subsystems.

One of the main problems in modern design results from the conflict between the problem decomposition and multidisciplinary optimization [63]. Decomposing a problem into smaller sub-problems makes the overall problem more controllable, but as a result it is more

challenging for system-level optimization to make a important contribution. Connecting a different number of subsystems into one is not simple. As the complexity of the system increases, so does the complexity of the model needed to complete the system level optimization. Solving optimization problems with a computer can take long periods of time, which can result in a delay of time between members on a design team. While waiting for the optimization results to return, the design team continues on with their work, often updating models and reacting to changes in the requirements. When an optimization does finally produce data, the results are often obsolete by these changes. This is the principal weakness of MDO, because it prohibits a full integration of the parts, subsystems and teams in the design.

Team of engineers working on the design of complex system cannot afford to wait for weeks for optimization data when performing a trade analysis. They also require integrating their work with a computer-based optimizer without distress of overriding each other's actions. Thus, is necessary to have a methodology that can relieve the fundamental conflict between these two approaches throughout the design cycle.

The topics presented in this chapter briefly introduce the different sets of techniques and tools that engineers use to design complex system. Even though the implementation of such methodologies generates good results, these methodologies have some problems and limitations.

This research presents an innovative approach to help engineers with the design decision process following complex engineered system. The design of complex engineered systems undertaken with respect to a variety of design objectives is essentially comparable to the multiagent coordination problem. It both cases, the decisions at the component level (subsystems and agents), and the interaction between those components, lead to global behavior. The following chapter introduces the subject of multiagent coordination.

Chapter 3: Related Research

Engineers put a lot of effort to create accurate models for complex engineered systems. The desired objective is to use a model that allows engineers to perform accurate failure analysis and evaluations of the system before any manufacturing process. The modeling of complex engineered systems, and the evaluation of failures are highly researched areas of interest within the engineering literature. This includes different areas that cover topics from performance evaluations to sensitivity analyses with input variables under uncertainty.

The main objective associated with this research work is to develop a methodology where a team of engineers can design complex engineered systems with the implementation of a multiagent coordination approach. The design of complex engineered system undertaken with respect to a variety of design objectives is fundamentally similar to the multiagent coordination problem. In both cases, the decisions at the component level (subsystems or agents), and the interactions among those components, lead to global behavior (complex system or multiagent system).

Multiple areas of engineering design have been reviewed as part of the elaboration of the two manuscripts included in this document. The following sections describe the concepts behind: *Complex Engineered System Modeling*, *Multiagent Coordination*, and *Coevulutionary Algorithms*.

3.1 Complex Engineered System Modeling

Model-based design techniques are often needed when designing complex engineered systems [75]. Model-based design builds a model of the system that is used to simulate the functions and evaluate the performance. These models are at times segmented and combined to form a single model, although they do not need to be [75]. The simulation of the models helps designers to understand the behavior and performance of the system prior to any physical testing.

Most of the times, models such as nuclear power plants are so large and complex that they are too computationally expensive to create, let alone simulate. For this type of problems, the system is divided in smaller sub-systems that can be simulated separately to collect information. On the down side, this approach outcome with high levels of uncertainty because of the missing information of the complete model simulation. Complex engineered systems present emergent interactions, that are not necessarily modeled or expected couple a set of components or sub-systems together. The degree of this *coupling*, or *complexity* as it is sometimes called, is difficult to determine because of the associated ambiguity related to the unknown component links. Coupling is described by the relationship between system variables and functions [72].

Nevertheless, it could be possible to shift the focus from modeling interactions to evaluate and incentivize subsystems so that their collective behavior achieves the system design goals. This shift in focus could be translated into a new methodology to model complex engineered systems. A multiagent coordination approach could be used to determine how to distribute responsibilities in a design process to the components in the design that are crucial to the success of the system.

3.2 Multiagent Coordination

Multiagent coordination is a key research area in agent-based approaches to automation [71]. 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 acheive the system objective. The key challenge is that a system designer needs to address two major credit assignment problems: *structural* and *temporal* [9, 71] 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, 84].

The temporal credit assignment problem has been extensively studied through single-agent reinforcement learning [9, 57]. The structural credit assignment problem has also received attention, and has been addressed by two broad approaches: *reward shaping* and *organizational structures*. Reward shaping aims to shape the system objective such that

the action of agents optimizing local objectives results in desirable system-level performance [7, 31].Organizational structures decompose the agents themselves into roles that enable coordinated behavior [1, 66].

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 [78, 83] Providing agents with objectives that satisfy these two properties (formalized in [77, 83]) 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, 37]

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.

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 problem can be translated to the design of complex engineered systems. In both cases, the decisions at the component level (subsystems or agents), and the interactions among those components, lead to global behavior (complex system or multiagent system.)

The final objective of this research is to demonstrate that a complex engineered system can be design using a multiagent coordination approach. However, significant challenges exist in adapting this approach to designing complex systems to meet design goals, while operating in stochastic and often unpredictable environments.

The first problem that a complex engineered system will confront is the accomplishment of the system global objective with the interaction of a large number of agents (subsystems). The complexity of the task resides on accomplishing the correct coordination between the responsibilities of the different agents. The agents need to receive high $factoredness$ and $learnability$ objective alignment [20]. $Factoredness$ defines how well two rewards are

matched in terms of their assessments of the desirability of particular actions. $Learnability$ defines how discernible the impact of an action is on an agent's reward function [20].

3.2.1 Reinforcement Learning for Multiagent Systems

The learning algorithm used in this research is $Q - learning$. When the system under study is using a large number of autonomous agents , *Potential-Based Reward Shaping* (PBRS) is used for a significant improvement in the coordination of agents. Reward Shaping is used to help agents to learn faster.

Reward Shaping includes modifying local rewards, difference rewards and global rewards such that agents can learn faster; this also helps to understand the agent's behavior[82]. Reinforcement learning is a paradigm that allows agents to learn by reward and punishment from interactions with the environment. The numeric feedback received from the environment is used to improve the agent's actions. The majority of work in the area of reinforcement learning applies a Markov Decision Process (MDP) as a mathematical model.

An MDP consists of state, action, action reward pair, where s is the state space, A is the action space, $T(s, a, s') = Pr(S0|S, A)$ is the probability that action a in state s will lead to state s', and $R(s, a, s')$ is the immediate reward r received when action a taken in state s results in a transition to state s'. MDP deals with finding a policy to maximize the reward. When we know about the environment we can approach this problem through policy and value iteration.

Most real life problems will not have any information regarding system dynamics, so value iteration cannot be used. But the concept of the iterative approach remains the same. Transferring information about values of states, $V(s)$, or state action pairs, $Q(s, a)$ pairs falls under the category of Temporal-Difference learning. These updates are based on the difference of the two temporally different estimates of a particular state or state-action value. After each transition, $(s, a) \rightarrow (s', a')$, in the system, the state-action values updates by the Eqn. (3.1) [80]:

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma maxQ'(s', a') - Q(s, a)] \tag{3.1}$$

where α is the rate ($\%$) of learning and γ;] is the discount factor.

The discount factor modifies the value of taking action a in state s, when after executing this action the environment returned reward r, and moved to a new state s'. The variable α, a value between 0 and 1, determines the relevance of future rewards in the update. A value of $\alpha = 0$ will optimized the immediate reward. Whereas, values of α closer to 1 will increase the contributions of future rewards in the future. The immediate reward r, which is in the update rule given by in above equation, represents the feedback from the environment. The idea of reward shaping is to provide an additional reward, which will improve the convergence of the learning agent with regard to the learning speed. Reward shaping in $Q - learning$ can be represented by Eqn. (3.2):

$$Q(s,a) \leftarrow Q(s,a) + \alpha * [r + F(s,s') + \gamma * maxQ'(s',a') - Q(s,a)] \tag{3.2}$$

where $F(s,s')$ is the general form of the shaping reward.

3.2.2 Difference Evaluation Function Theory:

The agent-specific difference evaluation function is defined as:

$$D_i(z) = G(z) - G(z_{-i} + c_i) \tag{3.3}$$

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:

$$\frac{\partial G(z)}{\partial a_i} = \frac{\partial D_i(z)}{\partial a_i} \tag{3.4}$$

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 3.3 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.

In this research the concepts of *Reinforcement Learning for Multiagent Systems* and *Difference Evaluation Function Theory* will be implemented in the complex engineered systems field of study. These concepts are used to address some of the challenges within multiagent coordination, how to properly select the agents inside the complex system, and how to define a global objective for the agents in the system that accurate captures the final objective of the system.

3.3 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 [23]. 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.

3.3.1 Coevolution

Coevolutionary Algorithms (CEAs) are an extension of evolutionary algorithms and are often well-suited for multiagent coordination domains [22]. 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 [55]. In *cooperative* coevolution, individuals succeed or fail as a team. This research 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 [56]. 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.

3.3.2 Cooperative Coevolutionary Algorithms

Cooperative coevolutionary clgorithms (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 [59]. 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 [81]. 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") [81].

Coevolutionary algorithms are implemented in this research to give a fitness evaluation

to the each design agent based on its collaboration with other agents. The cooperation between the agents will be used to optimize the design of a subsystem or subsystems inside a complex engineered system.

3.4 Summary of Manuscripts

The first manuscript presented in this thesis is the *Design of a Self Replicating Robotic Manufacturing Factory*, presented at IDETC 2015. The objective of the first manuscript was the implementation of a multiagent coordination to create a robotic manufacturing factory that would not require any human intervention. The robots would be responsible of keeping the system working and expanding. The design of a self-replicating robotic system was studied using a multiagent coordination based design approach. This paper specifically compared pre-programmed vs. learned behavior. The learning algorithm used in this case study was Qlearning. We investigated three different scenarios: two where agents used Q-learning, and one where agents used pre-programmed behavior. The rewards for tor the two learning scenarios were a local reward and the difference reward.

The second manuscript presented in this thesis is the *Design of Complex Engineered System Using Multiagent Coordination*, presented at IDETC 2016.

The objective of this manuscript was the design of a system using the collaborative work between a team of autonomous agents. The system we selected is a formula racing vehicle. The system is evaluated based on a set of objectives. We assigned an agent to design a particular subsystem of the vehicle. The design task of each agent is to make design choices that impact the performance of the entire vehicle. This paper specifically compared the performance of the system using global objectives vs. difference objectives. The agents used a cooperative coevolutionary algorithm to optimize their objectives.

A discussion about what have we learned from the application on the different learning algorithms is included in chapter 6.

Chapter 4: Designing a Self-Replicating Robotic Manufacturing in Space Lands

Authors:

Charlie Manion
102 Dearborn Hall
Email: manionc@oregonstate.edu

Nicolas F. Soria Zurita
102 Dearborn Hall
Email: soriazun@oregonstate.edu

Kagan Tumer
Rogers Hall 426
Email: kagan.tumer@oregonstate.edu

Chris Hoyle
Roger Hall 418
Email: chris.hoyle@oregonstate.edu

Irem Y. Tumer
Covell 116
Email: irem.tumer@oregonstate.edu

Proceedings of the 2015 ASME International Design & Engineering Technical Conferences
35th Computers and Information in Engineering Conference (CIE-19-2) IDETC 2015
August 2-5, 2015, Boston, MA, United States of America
Journal version in preparation to be submitted Summer of 2016

4.1 Abstract

This paper presents a Multiagent Systems based design approach for designing a self-replicating robotic manufacturing factory in space. Self-replicating systems are complex and require the coordination of many tasks which are difficult to control. This paper presents an innovative concept using multiagent Systems to design a robotic factory for space exploration. Specifically presented is an approach for coordinating a conceptual model of a self-replicating system. The arrival of a set of agents on an unknown planet is simulated, whereby these simple agents will expand into a self-replicating factory using the regolith gathered from the surface of the planet. NASA is currently investing in space exploration missions that consider using the resources on the surface of other planets, asteroids or satellites. The challenge of the project is in the implementation of a learning algorithm that allows a large number of different agents to complete simultaneous tasks in order to maximize productivity. The simulation in this work is able to present the coordination of the agents during the construction of the factory as the parameters of the learning algorithm are changed. System performance is measured with a pre-programmed method, using local and difference rewards. The results show the advantage of using a learning algorithm to better build the robotic factory.

4.2 Introduction

One of the means of obtaining clean and sustainable power is a solar power satellite; that is putting large arrays of solar panels into geosynchronous orbit and beaming power down to where it is needed using microwaves. In orbit, there is no night, and cloudy days are non-existent, so clean power can be produced around the clock. Unfortunately, launching everything needed for this solar power satellite from earth is too expensive at the current time [27].

Utilizing resources found in space can reduce the cost of space missions, allowing the development of large-scale planetary engineering projects. Projects such as terraforming via machine self-replicating systems on Mars or Venus are proposed, where a large factory will create modifications of the environment that will allow human colonies on those planets[24].

Most proposed self-replicating systems rely on centralized control to coordinate activities for replication. These systems are difficult to design, and are not very robust. Failure of the central controller leads to failure of the entire system, so the centralized controller must be designed to be very robust which leads to increased design costs[13]. However, if factory robots and machines could coordinate themselves using only local actions then the system could be made much more robust and easier to design. If one robot fails, the rest of them will still function.

This paper provides a proof-of-concept to coordinate the different activities necessary to operate a self-replicating factory so that it replicates rapidly using Multiagent Systems. The autonomous robotic manufacturing factory will work with two type of agents using a learning algorithm. The two types of agents, called producers and workers, will interact together to construct the factory using the resources on the surface. *Methodology* section describes the learning algorithm, agents and processes implemented on this paper.

Table (4.1) presents the four possible case scenarios for the implementation of a learning algorithm in the system. Agents have only two possibilities, $Learning$ or $NoLearning$ after the implementation of the algorithm. For the present work, cases 1 and 2 were simulated and compared in sections *Simulator* $\&$ *Results*. The agents workers will be first simulated using a pre-programmed behavior ($Case\ 1$), then using a learning algorithm ($Case$

2). The paper shows how the implementation of a learning algorithm allows the factory to maintain an increasing performance. Cases 3 & 4, were not simulated at this time. However, the producers agents use a set of reward functions on the simulations which allow the system to behave as a self-replicating robotic manufacturing factory.

Table 4.1: Implementation of learning algorithm.

		Producer	
		No Learning	Learning
Worker	No Learning	Case 1	Case 3
	Learning	Case 2	Case 4

As the factory increases the number of agents, the complexity of the system will increase. It will become difficult to control all the interactions between the agents. In this work, the robotic manufacturing factory is viewed as a large complex engineering system. The hypothesis in this paper is that a multiagent coordination problem is fundamentally similar to complex system design, undertaken with respect to a variety of design objectives. In complex engineering systems, complexity will arise in an unpredictable manner in which interactions between components and environment modifies system behavior. The self-replicating manufacturing factory is shown to account for unanticipated events and extreme variation in the system conditions over time. As such, this paper presents a proof-of-concept for the design of the factory using a candidate complex system design approach based on a multiagent coordination.

4.3 Background

In the 1980s NASA conducted studies on building a self-replicating factory on the moon in Advanced Automation for Space Missions.[25] In this study, it was proposed to land a small seed factory that was capable of expanding itself and replicating by mining lunar regolith and processing it using solar power. However, the system required a complicated centralized controller with machine vision, pattern recognition, inference, and reasoning capabilities to coordinate the factory and troubleshoot faults. This made the system more

complicated to design and not very robust. This study also outlined a test to determine the feasibility for this system[48].

More recently, Lackner and Wendt [39] proposed making a self-replicating system in a desert on earth that was much simpler. This system was to consist of simple small robots they called auxons running along electrified tracks. To control the system, they proposed that the auxons in the system follow simple local rules for interacting with each other to coordinate manufacturing activities. To avoid the need for machine vision and complicated control schemes, they restricted the auxons to only move on electrified tracks and scrap auxons that are broken instead of attempting to repair them. This is potentially easier to design, more robust, and requires less resources to replicate.

However, even though Lackner and Wendt proposed to use local rules to coordinate self-replicating robot factories, they did not go into detail of what local rules should be nor did they outline the system beyond the conceptual level [50].Chirikjian and Sukathorn have demonstrated simple self-replicating robots that pass the basic feasibility test outlined in the advanced automation for space missions study[26], and did so following simple local rules in a structured environment. In work done by Eno et al., self-replicating robots capable of replication in a structured environment were demonstrated that did not require microcontrollers for control [21]. Lee and Chirikjian have demonstrated a self-replicating robot that can replicate in a minimally structured environment without using microcontrollers [41]. Moses et al. recently presented a modular robot capable of assembling copies of itself from components it could potentially make from raw materials [52].

Much of this work demonstrates that self-replicating robots are feasible and that progress is being made in the area of processing raw materials into usable components. However, many of the self-replicating robots mentioned here are not robust enough to handle failure, do not carry out resource gathering operations, and require centralized control. Recent research on robotics tries to emulate systems from nature. Inspiration is taken from insect colonies such as: ant and wasp colonies to emulate or derive new coordination algorithms [14].

Reinforcement learning allows agents to learn by reward and punishment from interactions with the environment. One common algorithm from this field is $Q - Learning$.

$Q-Learning$ can be used to find an optimal action-selection policy for any given Markov decision process (MDP). It works by learning an action-value function that ultimately gives the expected utility of taking a given action in a given state and following the optimal policy thereafter. The use of $Q-Learning$ for exploration robots is popular [45]. However, while $Q-Learning$ has been extensively used for exploration robots, it has not been widely used in the self-replicating robot domain.

Swarm Logic typically consists of a population of simple agents interacting locally with one another and with their environment. The inspiration often comes from nature, especially biological systems like ants or wasps. The agents follow very simple rules, and although there is no centralized control structure dictating how individual agents should behave, local, and to a certain degree random, interactions between such agents lead to the emergence of "intelligent" global behavior, unknown to the individual agents [20].

This paper combines $Q-Learning$ with $SwarmLogic$ to produce simple agents (ants) with simple tasks that worked in the factory (colony), with limited communication. Monekosso and Remagnino have used a $Q-Learning$ algorithm inspired by the natural behavior of ants. They use a belief factor that measures the confidence of the agent in the detected pheromone[51]. Chia-Feng Juang used a Reinforcement Q-Learning algorithm called ACO-FQ to optimize the behavior of an ant colony[36]. The algorithm creates a list of candidate consequent action rules, and the best combination of actions is selected according to the pheromone levels and $Q\ values$.

4.3.1 Complex System Design

Selection of a 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 [53].

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* [18]. This is a dramatic change in perspective for system design, promising a reduction (or elimination) of cost and schedule overruns [6, 16] by identifying high value designs for development. Value-Centric Design can be considered part of the larger field of Decision-Based Design (DBD) [32, 79]. 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.

The general approach to formulating a system-level value function is to quantify the balance between benefits and cost [16, 18]; the model can be developed either from economic measures, such as surplus value (benefits of a system minus all costs) or Net Present Value (NPV) [32], or using design metrics such as complexity or system connectivity [6, 11, 47, 17, 61, 40].

4.3.2 Multiagent Coordination

Multiagent coordination is a key research area in agent-based approaches to automation [71]. 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 acheive the system objective. The key challenge is that a system designer needs to address two major credit assignment problems: *structural* and *temporal* [9, 71] 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, 84].

The temporal credit assignment problem has been extensively studied through single-agent reinforcement learning [9, 57]. The structural credit assignment problem has also received attention, and has been addressed by two broad approaches: *reward shaping* and *organizational structures*. Reward shaping aims to shape the system objective such that the action of agents optimizing local objectives results in desirable system-level performance [7, 31]Organizational structures decompose the agents themselves into roles that enable coordinated behavior [1, 66].

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 [78, 83] Providing agents with objectives that satisfy these two properties (formalized in [77, 83]) 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, 37]

4.4 Methodology: Robotic Manufacturing Base

This paper proposes the design of a simplified version of the manufacturing factory with a minimum number of tasks to perform in a $2D$ grid environment.

4.4.1 Environment and Agents

The environment consists of a $2D$ grid of cells that can be either regolith or factory elements on which mobile agents can move and where actions occur in discrete time steps or turns. The simple factory consists of resource gathering and product placing workers, power producing solar cells, power distributing pavers, resource processing producers. This simple representation of a self-replicating factory necessitates coordination of resource acquisition, power management, production management, and factory layout that are also necessary with more complex models.

Pavers are power distributing elements on which other factory elements can be placed and on which workers can charge. The task of placing pavers represents the task of laying the foundation for the factory. The factory elements of solar cells and producers are required to be placed on pavers. Connected sets of pavers share the same amount of power that is available for use by factory elements on top of them.

Factory elements that use power subtract from the amount of power available. On top of pavers we can place solar cells, which serve as power production components. Solar cells

add to this amount of power available each turn. Power cannot be stored and any unused power is lost at the end of the turn.

The next component which goes on the pavers is the producer which represents the materials processing and manufacturing system in our factory. The producer can make anything in the factory from x regolith in n turns as long as it has enough electrical power in each turn to do so. Producers can only store so much regolith and so many finished products and cannot receive any more regolith or make more products if this capacity is full. Producers are also agents and can communicate with worker robots and other producers to coordinate resource gathering and production activities.

Worker robots, or workers, represent mobile multi-purpose mining and construction robots. The worker's main purpose is to collect regolith from cells that do not have pavers on them and deposit it into producers for processing. The other purpose of the worker is to pick up pavers and factory elements and put them where they need to be to expand the factory. Each of these tasks uses up a certain amount of charge from the worker's battery and the worker must periodically recharge on paver cells. If a worker runs out of charge and it isn't on a paver with $power\ available > 0$, it becomes non-functional.

The producer is a non-learning agent and produces the product with the highest utility determined by a set of utility functions.

4.4.2 Multiagent coordination problem

The first problem that the robotic manufacturing system will confront is the accomplishment of the global objective with the interaction of a large number of agents and processes inside the factory. The complexity of the project resides on accomplishing the correct coordination between the tasks of the different agents. The agents need to receive high $factoredness$ and $learnability$ objective alignment [20]. $Factoredness$ defines how well two rewards are matched in terms of their assessments of the desirability of particular actions. $Learnability$ defines how discernible the impact of an action is on an agent's reward function [20]. These objective alignments will allow the systems to expand rapidly on the planet with maximized productivity.

The producers need to manage the production and prioritize the construction of work-

ers, solar cells, and more producers according to the needs of the system. The producer makes decisions to produce a certain product as a function of the available energy and regolith. For example, if the system needs more energy, the production of solar cells will be prioritized until the energy requirements are satisfied. The same logic will be used on the production of the other elements and agents. Workers have different tasks, they are responsible for the regolith collection and placement/retrieval of factory elements. The workers will have to coordinate not only the interactions with the other workers but also the activities to meet the needs of the producer. Workers and producers are responsible for the correct storage of resources and elements in the factory. If this coordination between agents fails, the factory will stop the manufacturing process and the system will collapse.

4.4.3 Global Objective

The global objective is to maximize its productivity, which is defined for this system as the number of products produced at each time step divided by the amount of products that are already placed. (How much is produced divided to how much it took to produce it). This is similar to the productivity measure defined in [39], which is the amount of mass processed per time step divided by the mass of the entire system doing the processing.

Since the system does not maintain a measure of product mass, so the number of products is used instead. The problem here is that, during the initial time steps, the productivity will be equal zero. When the system starts no new products will be produced for a couple time steps, so the productivity will be $zero$. Therefore, the productivity of the factory will be poor. As the factory starts working and growing the value of productivity will increase. Unfortunately, productivity will vary by a large amount. To solve this, the productivity measure also counts incomplete products when taking into account number of products.

A product has a value proportional to how many turns the producer has carried out producing the product over the number of turns to complete the product. A product is a quarter of the fractional amount that is complete if it is being produced, a half of a product if it is stored in the producer, three quarters of a product if it is being carried by a worker, and a full product if it is placed. The regolith carried by a worker is counted as being half as valuable and is assigned to be the full value if it is stored in the producer. Productivity

is modeled in Eqn. (4.1).

$$P = \frac{abs(d_{regolith}) + d_{pavers} + d_{solarcells} + d_{workers} + d_{producers}}{P_{pavers} + P_{solarcells} + n_{workers} + P_{producers}} \tag{4.1}$$

where:

$d_{regolith}$:	Difference in regolith
d_{pavers}:	Diffe rence # pavers
$d_{solarcells}$:	Difference # solar cells
$d_{workers}$:	Difference # workers
$d_{producers}$:	Difference # producers
P_{pavers}:	Number of pavers placed
$P_{solarcells}$:	Number of solar cells placed
$n_{workers}$:	Number of workers present
$P_{producers}$:	Number of producers placed

This global objective has high factoredness for the workers and producers. Both the workers and the producers have actions that can increase the global objective. We take the absolute value of the difference in regolith to prevent the system from being penalized for using up regolith. Using up regolith to make products is also considered productive. This objective function has high factoredness for both producers and workers, as each action is aligned to the global objective. It is worth noting that this global objective function has low factoredness for the workers, but high factoredness for the producers. The actions of the worker do not directly affect the production number of paver, solar cells, workers, or producers. However, the actions of a producer immediately affect the productivity.

4.5 Simulator

NetLogo is used for the simulation environment. NetLogo is an agent-based programming language with a modelling environment. NetLogo implements a grid environment, agents, and useful functions for agents to interact with the grid environment.

For the simulation a two dimensional $20x20$ grid is created with a torus topology so

that the world is continuous. As the worker do not know where they are, the size of the grid does not affect the simulations or the results. The size of the grid can be changed at any time. However a small grid size allows a faster run time and easy visualization of the simulation. Each grid cell is created with a random regolith value between: 0 to 10. To simplify things for the simulation, the environment does not have any obstacles, and workers will be allowed to go through grid cells containing other agents or factory elements. In future work the simulation will consider a large and complex world. The world will start with this set of elements on the first iteration:

Producer: The producer is a fixed agent that can create more elements such as: *workers, solar cells, pavers and producers* (self-replicating agent) from the regolith. The construction of each element needs a defined number of resources, so each producer will evaluate which element is necessary to complete the global objective. Producers can only store a defined amount of finished products proportional to the size of the product. Whereas a producer can store four pavers, it can only store one producer. In addition, workers are not stored in the producer and drive off after they have been produced. Producers are no-learning agents and produce the most valuable product as long as they have enough regolith and power to do so. Producers will store the products in a stack and workers can only remove the product at the top of the stack.

To communicate with workers, producers also have beacons on them to coordinate workers. The producers emit three different types of beacons, location beacons, regolith beacons, and product-done beacons. Location beacons are intended to help the workers determine where to place things and where pavers are. The location beacons are always on. Regolith beacons allow the producer to call workers to obtain regolith. Meanwhile product-done beacons will call for free workers to take the manufactured item out of the producer. The signal strength to a location beacon decays as an inverse square of distance to the beacon and the intensity of the signal at the beacon. All location beacons have the same intensity and are on all the time. The regolith and product-done beacons have intensities that depend on how full regolith or products are. Product-done and regolith beacons do not decay with distance and the worker can sense the producer with the maximum beacon intensity.

Workers: The workers are mobile agents that can collect regolith from grid cells that are not covered by pavers and transport products from the producer and the world. Workers are allowed to install producers and solar cells on empty pavers. Workers can sense the properties of the cell that they are on and the cells around them. The workers have different sensors that allow them to detect: factory elements and the amount of regolith on the grid cell. Workers also have a beacon receiver that receive the different beacon signals emitted from the producers so that they can coordinate their actions with respect to requirements of the producers. In addition, the worker has actions to move in the direction of the strongest signal for each of the beacon types.

Pavers: The pavers work as a network array that transmits energy and information across continuous pavers. The pavers are the link between the producer's, solar cells and worker's interaction. The eighteen pavers are located at the center of the world. The elements (producer, workers and solar cell)are placed over the pavers on the center of the world.

Solar cells: The solar cells produce a constant amount of energy at each turn and distribute it everything on the pavers network. Solar cells add to the energy available on each set of connected pavers, likewise, producers and charging workers decrease the amount of energy available on connected pavers. For the proper development of the factory the energy flow needs to work constantly, and each process will require energy.

Each worker automatically recharge whenever they are on a paver that has power. Producers on the other hand, have different energy consumption according to the product it is producing; for example manufacturing a worker requires more energy than a paver. Each product that the producer can create requires a defined amount of resources (energy and regolith) and time (turns). Currently the code is structured so that the individual energy, time, and regolith costs for each product can be easily changed.

4.5.1 States and Actions of Agents

In the robotic system, workers are dynamic agents. Workers move around the environment and collect regolith and drop it at the producer. They learn actions from Q-learning

which moves them to get regolith. Workers also collects pavers from the produces and place them on the surface forming a grid-like structure. Workers picks up the finished products from the producer and place it on the paver. A worker can move to any of the surrounding cells, mine, transfer regolith to a producer, pick up a product from a producer, place a factory element, move to the cell with the lowest beacon value, move to the cell with the highest beacon value, move toward the producer with the highest regolith beacon intensity, or move toward the producer with the highest product done beacon intensity. The last couple of actions are high-level actions in that they specify a behavior, in this case driving toward or away from a producer, instead of a low level task such as moving up, down, left, or right. This enables the Q-matrix to be made smaller and learning to be carried out more efficiently, because the worker does not need to learn the behavior or keep track of additional states so the behavior can be implemented.

The $state$ of the agent $worker$ is what is on the cell the $worker$ is currently on, plus what is on the surrounding cells, and the charge level of the worker's battery. What is on a cell is expected to have a discrete value. For a cell containing only regolith, a value from 0 to 1 is assigned in increments of 0.25. For a cell that contains a paver, solar cell, or producer a value from 2 to 4 is assigned respectively. The state of the battery is discretized to be either 25%, 50%, 75%, or 100%. The NetLogo interface allows the user to modify the initial settings of the world. The user can change the number of: resources in the world, agents (producers, workers), and resources needed for the construction of the different elements in the factory. As the initial conditions and settings changed, the simulation will present the different results from the behavior of the system for our analysis. To keep things simple, workers have a $Q - matrix$ where all the state and actions will be updated, this list will be use on the implementation of the learning algorithm. The utility functions, local rewards, and global objective function are calculated using the respective estimate $Q - matrix$ the agents. If all workers are not charged and located off pavers and if all producers cannot make another worker, then the factory has failed and the simulation is reset. Additionally, if all regolith cells are covered by pavers then the simulation is also reset. The software interface allows the user to stop the simulation at any time step.

4.5.2 Learning Algorithm

The learning algorithm for the model is $Q - learning$. As, multiple agents are need, *Potential-Based Reward Shaping* (PBRS) is used for a significant improvement in the coordination of agents. With multiple agents and every agent working optimally Reward Shaping is used to help agents to learn faster. Reward Shaping includes modifying local rewards, difference rewards and global rewards such that agents can learn faster; this also helps to understand the agent's behavior[82]. Reinforcement learning is a paradigm which allows agents to learn by reward and punishment from interactions with the environment. The numeric feedback received from the environment is used to improve the agent's actions. The majority of work in the area of reinforcement learning applies a Markov Decision Process (MDP) as a mathematical model.

An MDP consists of state, action, action reward pair, where s is the state space, A is the action space, $T(s, a, s') = Pr(S0|S, A)$ is the probability that action a in state s will lead to state s', and $R(s, a, s')$ is the immediate reward r received when action a taken in state s results in a transition to state s'. MDP deals with finding a policy to maximize the reward. When we know about the environment we can approach this problem through policy and value iteration.

Most real life problems, will not have any information regarding system dynamics, so value iteration cannot be used. But the concept of the iterative approach remains the same. Transferring information about values of states, $V(s)$, or state action pairs ,$Q(s, a)$ pairs falls under the category of Temporal-Difference learning. These updates are based on the difference of the two temporally different estimates of a particular state or state-action value. After each transition, $(s, a) \rightarrow (s', a')$, in the system, the state-action values updates by the Eqn. (4.2) [80]:

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma max Q'(s', a') - Q(s, a)] \tag{4.2}$$

where α is the rate ($\%$) of learning and γ;] is the discount factor.

The discount factor modifies the value of taking action a in state s, when after executing this action the environment returned reward r, and moved to a new state s'. The variable

α, a value between 0 and 1, determines the relevance of future rewards in the update. A value of $\alpha = 0$ will optimized the immediate reward. Whereas, values of α closer to 1 will increase the contributions of future rewards in the future. The immediate reward r, which is in the update rule given by in above equation, represents the feedback from the environment. The idea of reward shaping is to provide an additional reward which will improve the convergence of the learning agent with regard to the learning speed. Reward shaping in $Q - learning$ can be represented by Eqn. (4.3):

$$Q(s,a) \leftarrow Q(s,a) + \alpha * [r + F(s,s') + \gamma * maxQ'(s',a') - Q(s,a)] \tag{4.3}$$

where $F(s,s')$ is the general form of the shaping reward.

4.5.3 Agent Rewards

Local reward for worker

The worker receives a reward for mining regolith proportional to the regolith mined, a reward for delivering regolith proportional to the amount of regolith delivered, and a reward for placing a factory element. The reward for delivering regolith is higher than the reward for mining regolith, as delivered regolith is more valuable than mined regolith. It is worth noting that this local reward is aligned with the global, the actions of mining, delivering regolith, picking up elements, and placing elements all directly increase the global objective. The total reward given in one time step will be Eqn. (4.4) :

$$R_{worker} = Reg_{mined} * w_1 + Reg_{deliver} * w_2 + R_{pickup} + R_{place} \tag{4.4}$$

where:

Reg_{mined}:	Amount of regolith mined
$Reg_{deliver}$:	Amount of regolith delivered
R_{pickup}:	Reward for picking up a producer, solarcell, or paver
R_{place}:	Reward for placing a producer, solarcell, or paver
w_1:	Reward per regolith mined
w_2:	Reward per regolith delivered

Difference Reward for Worker

A difference reward can be implemented for the worker to attain a reward with high factoredness and learnability. The difference reward is the difference between the group utility with the agent, and without it Eqn. (4.5)

$$D_i = G(z) - G(z_{-i}) \tag{4.5}$$

where: $G(z)$ is the World with agent i and $G(z_{-i})$ is the World without agent i.

4.5.4 Worker Potential Functions

To help the workers learn faster four different potential functions are used. The first potential function is designed to give the worker an incentive to recharge, the second gives the worker an incentive to return to a producer if it is full of regolith, the third and fourth potential functions encourage the worker to drop off or pick up products from a producer.

The first potential function,called the "stay charged" potential function Eqn. (4.6), gives the worker a reward proportional to how close the battery is to being drained if the worker moves in the direction of increasing location-beacon values. Moving in the direction of increasing location beacon values means that the worker is moving toward a location where it can recharge.

The second potential function, called the "go home" potential function Eqn. (4.7), gives the worker a reward proportional to how close regolith storage is to being full. If the worker moves in the direction of increasing location beacon values it is likely to find a producer to drop regolith at. To encourage the workers to drop off regolith and pick up products from the producers, the Q values are initialized for these associated actions to a high value. It is worth noting that the first two potential functions use information that can be determined entirely locally.

$$Pot_{staycharged} = \frac{W_{bc} - W_c}{W_{bc}} \tag{4.6}$$

$$Pot_{gohome} = \frac{W_{regolith}}{W_{regolithcapacity}} \tag{4.7}$$

where:

$W_{regolith}$:	Worker regolith
$W_{regolithcapacity}$:	Worker charge
$Reg_{delivered}$:	Amount of regolith delivered
$R_{placepaver}$:	Reward for placing paver
$R_{placeelement}$:	Reward for placing element
w_2:	Reward per regolith delivered

4.5.5 Worker Q-Matrix Initialization

One problem with the system is how to initialize a newly produced worker's Q-matrix. Initializing a worker's Q-matrix to empty is inefficient and keeping a population of Q-matrices from previous runs is difficult due to the variability in population size. In order to solve this problem, when a worker picks up a product from a producer it copies the Q-matrix to the producer and any workers produced by said producer will be initialized with a copy of the Q-matrix. Workers that pick up a product from a producer are a good candidate to copy Q-matrices from, because they have likely already learned to stay charged, mine, and transfer regolith to a producer. On the initial time step, no producers will have any products to pick up, so the workers won't be able to copy their Q-matrix until regolith has been mined, transferred, and a product has been produced.

4.5.6 Producer Reward Functions

In the current simulation environment, producers are non-learning agents that manufacture the product that is most valuable to the system. If the system is running low on power it is more valuable to manufacture a solar cell than a producer because if power available goes to zero then productivity will decrease. As the power available goes up, solar cells are not as valuable because productivity is not as constrained by power available. How valuable a given product is to the system is formally expressed with a set of functions describing the 'benefit' of manufacturing that product. This enables the producer to select the 'best' product to manufacture at each time step.

Alternatively, since the producers are aware of what other products are being produced

a rough estimate of the future reward can be made. To prevent division to 0, a factor of 0.1 is added to the denominator on the function.

Reward of a solar cell Eqn. (4.8) is inversely proportional to the amount of power available Pa for reasons that have already been elaborated above.

$$U_{sc} = \frac{1}{P_a + 0.1} \tag{4.8}$$

Reward of a paver cell Eqn. (4.9) is inversely proportional to the number of pavers available P_{va}, because if the number of pavers goes to zero then the factory can no longer expand.

$$U_{pav} = \frac{1}{P_{va} + 0.1} \tag{4.9}$$

Reward of producers Eqn. (4.10) is proportional to the power available and inversely proportional to the regolith capacity C_r. If regolith and power are not being used, then the system is harvesting more energy and material than it can process and production should be expanded to compensate. C_r is the system regolith capacity remaining or how much regolith can be stored minus how much regolith is held.

$$U_{pro} = \frac{Pa}{C_r + 0.1} \tag{4.10}$$

Reward of a worker Eqn. (4.11) is proportional to the global idle time. A high idle time indicates that there are not enough workers to remove products from producers and more workers should be added. In addition, while the producer is currently a non-learning agent, these utility functions might be used as local rewards or potential functions for learning producers. $Idletime$, is the amount of time the producers spend with $productcapacity = 0$. For each time step that a producer has zero capacity it increments a counter, when the capacity goes up, the counter resets to zero. The global idle time is the sum of all the these counters divided by the number of producers. This represents the amount of time the producer is wasting waiting for a worker to pick up products. $w5$ is a weighting factor for tuning the importance of global idle time.

$$U_w = globalidletime * w_5 \tag{4.11}$$

However, at the current time it is difficult to determine the factoredness of the functions. All of these functions encourage a producer to start making a product that will immediately increase the global objective, but the long term impact of that product on the global objective is difficult to quantify. The functions for the paver and solar cells are well factored, because producing a solar cell or paver when power available or pavers available is low could prevent the system from crashing. It is much more difficult to determine if the utilities of workers and producers increase the objective function in the long term. A worker or producer made could end up decreasing the productivity if said worker or producer does nothing for the system.

4.6 Results

Three scenarios were considered: local rewards, difference rewards, and pre-programmed behavior. Five simulations were run of each scenario for 2085 time steps, and each of the five runs were averaged together. As this data ended up being very noisy despite averaging five different runs, a running average was used. These results are plotted on Figure 4.1. It is worth noting that if the factory starts from an initially large ”seed” with a large number of workers, the factory expands very slowly and almost half of the population of workers is in the uncharged state at every time step. For the Pre-programmed behavior, the workers are controlled with a simple state machine and switch between states of mining and placing elements. The worker also drives back to base if the battery charge drops below a certain value. In the mining state, the worker drives off the pavers and moves towards cells with high regolith values and mines. If it can’t find any regolith, it drives in the direction of decreasing location beacon values. Once it is full of regolith it drives in the direction of the producer which needs regolith the most determined by regolith beacon values, once it reaches the producer it drops regolith off and switches to the ”place element” state. In the ”place element” state it finds a producer, picks up a factory element from it, finds a place for said element and returns to the mining state.

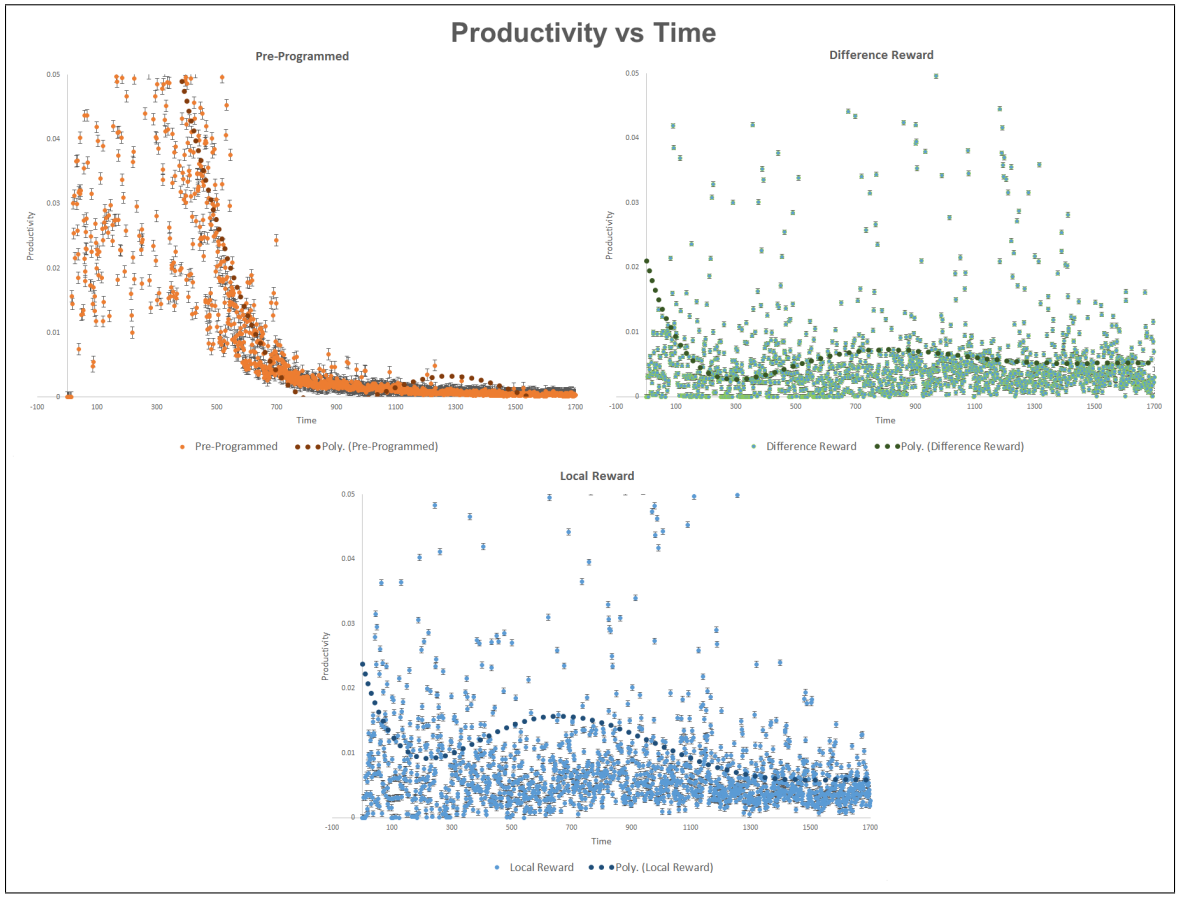

Figure 4.1: Averaged productivity of the system.

The Pre-programmed behavior works very well and has high productivity for about 500 time steps, then the productivity approaches zero and the factory stops expanding. This is because the system mines out all the regolith close to the factory until the regolith is too far away for the workers to reach without needing to head back and recharge. An example of this state is shown in Figure 4.2. The Pre-programmed behavior also leads to overproduction of workers. Workers get stuck in the mining state because they can't find any regolith, so they don't remove products from the producers leading to producer idle times to be increased.

As a result, more workers are been produced, which get stuck in the mining state and

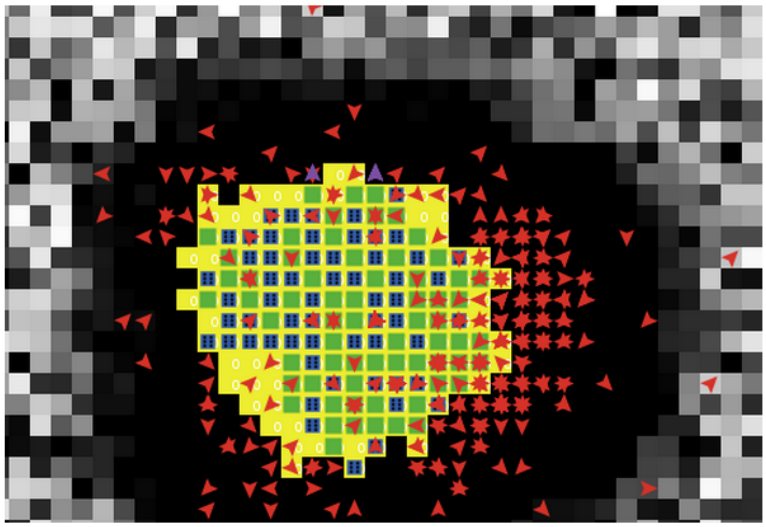

Figure 4.2: Result of pre-programmed behavior. The large ring of black cells contain zero regolith, around factory.

furthers the problem. The learning agents with difference rewards and local rewards perform better than the pre-programmed behavior and the agents are able to keep expanding the factory. The policy the workers tend to learn seems to be to expand one side of the factory. The paths taken by the workers indicate that they tend to stay in one side of the factory and this is illustrated in the Figure 4.3. In addition the workers tend to keep the distance to regolith from pavers as short as possible. As the workers expand one side of the factory, the producers on the far side will stop the manufacturing process because no worker will deliver regolith.

These inactive producers have indirect negative effect on the Productivity of the systems, but they are not at capacity so they don't increase the system idle time and increase the utility of workers. In addition, because these producers are not using power, the power available stays almost constant so the utility of solar cells stays low and few new solar cells are produced. This allows more resources to be used on making a continuously expanding edge of pavers and producers.

Although the productivity is really low and still decreases for both the difference and

local rewards, it does not crash to zero like the pre-programmed behavior. Local rewards perform slightly better than difference rewards. This could be due to the fact that the productivity without a worker can be higher than the productivity with a worker if said worker is not mining, transferring regolith, picking up or placing elements. This could be penalizing the worker for going through the intermediate actions necessary to accomplish tasks that improve the global objective. The difference in performance between difference rewards and local rewards is not very large and could just be noise. Another possibility is that the potential functions have not been weighted correctly in comparison to the difference reward.

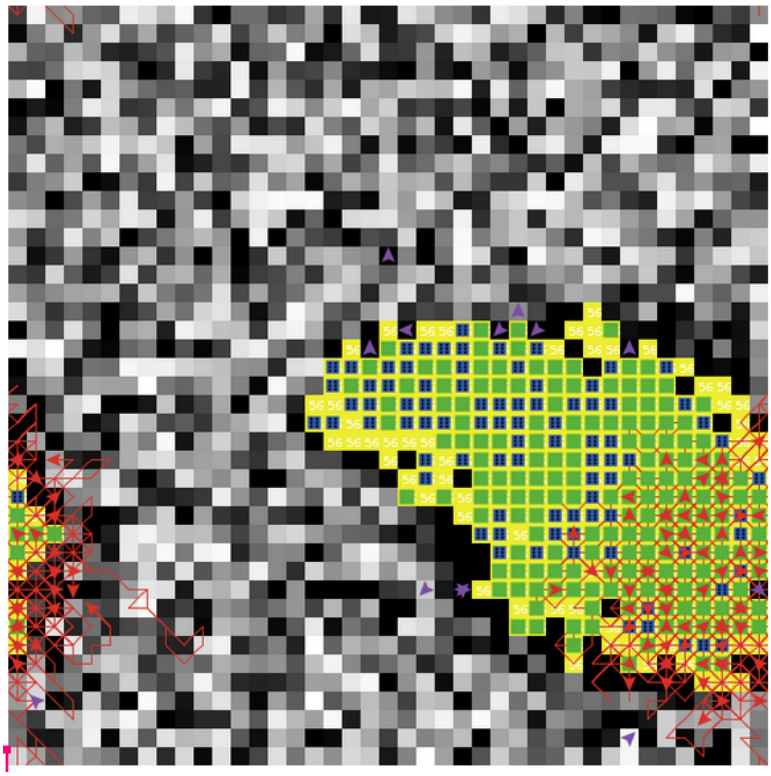

Figure 4.3: Learned behavior with paths.

4.7 Conclusions

In this paper the design of a self-replicating robotic system was studied using a multiagent coordination based design approach. This paper specifically compared pre-programmed vs learned behavior. A pre-programmed behavior was shown not to be the best approach to solve the multiagent-based design problem in this domain, as the results showed how the system can collapse. The pre-programmed behavior was used to present the difference between using a learning algorithm and a programmed behavior.

The implementation of a learning algorithm was shown to work much better than a pre-programmed behavior. After running the code for several simulations the workers were shown to learn how to continuously expand the edges of the factory as shown in Figure 4.3. As workers expand on one side of the factory, the producers on the opposite side will become inactive. As a result the productivity of the systems decreases because the number of inactive agents on the system increases as the factory expands towards one side, but the productivity of the system does not completely crash as is the case with the pre-programmed behavior.

Even though the local rewards have high factoredness, from the simulation results, there appears to be quite a bit of room for improvement as the productivities achieved were very low. The low productivities could be caused by shortcomings of using non-learning producers and might be remedied by using producers that learn. The workers might be purposely keeping productivity low as this may be the only way for them to prevent the producers from overproducing a product and crashing the system.

The utility functions could be used as a local reward for the producers, wherein the reward a producer receives is proportional to the utility of the product it made at the time of that product's completion. Shaping difference rewards by potential-based reward shaping $DRiP$, is a very good candidate for a producer reward method and the utility functions presented above could be used as potentials to determine which product a producer should start making. If the producers are to be kept non-learning, at the very least it is necessary to implement a function that makes all producers aware of what other products are being produced. This way overproduction of the unneeded products does not end up crashing the system.

Although the difference reward is well suited for many multiagent Systems domains, it may not be the best approach for this complex domain. It is necessary to explore other learning algorithms that consider agent congestion and small task distribution such as Assignment-Based Decomposition [60]. The method consists of decomposing the problem of action selection into an upper assignment level and a lower task execution level. The Assignment-Based Decomposition was used to solve a problem where a large number of collaborative agents are trying to complete a set of actions assigned determined beforehand with search. The self-replicating factory could use the same method to separate the different tasks that workers and producers need to complete in order to coordinate the performance of the system.

4.8 Future Work

The next objective in this research is to obtain results after modifying the interactions between the world and the agents. In this paper the world was simple; but a real self-replicating robotic factory needs to consider all the dynamics between gathering different type of resources and producing different type of mechanical and electrical components, using a large number of processes and agents. The next challenge in this research is the implementation of different simultaneous processes that will allow the manufacturing of several specific components and resources for the assembly of different agents. This will show how the system performs when failure modes are introduced inside the agents and see how the system resolves the failure. The goal is to challenge the agents as the environment and world represent a real hazard to the system.

Furthermore, how the failure of a single agent can cause other agents to fail will be explored as well as whether the system can adapt to failure. There has been a large amount of work dedicated to failure analysis of complex systems at the conceptual design phase. However, much of this work has been focused on what are arguably 'single agent systems'. The multiagent System should facilitate the failure analysis design approach and obtain new results.

Failure analysis is important for making a self-replicating robotic factory a reality, because it can determine whether such a system is viable. In order for a self-replicating sys-

tem to be viable it needs to be able to replace components faster than said components fail. However, in the real world determining this is complicated by the fact that failures can cascade into ever larger failures through the interactions of the components. This work could open a new frontier in the multiagent System research field. Of particular interest, research will explore the field of complex system design and the analysis of failure propagation.

The design of the robotic factory required the coordinate work between different agents and process; next steps will consider the agents as the designers of the factory, with the objective to minimize the cost of construction and maximize the production of an engineering system. Parallel to this work, the research will explore how the propagation of failures inside the agent's components will affect the performance of the system.

Chapter 5: Design of Complex Engineering Systems Using Multiagent Coordination

Authors

Nicolas F. Soria Zurita
102 Dearborn Hall
Email: soriazun@oregonstate.edu

Mitchell K. Colby
Autonomous Systems Group. Scientific Systems Company, Inc.
Email: colby.mk@gmail.com

Irem Y. Tumer
Covell 116
Email: irem.tumer@oregonstate.edu

Chris Hoyle
Roger Hall 418
Email: chris.hoyle@oregonstate.edu

Kagan Tumer
Rogers Hall 426
Email: kagan.tumer@oregonstate.edu

Proceedings of the 2016 ASME International Design & Engineering Technical Conferences
42st Design Automation Conference (DAC)
IDETC 2016
August 21-24, 2016, Charlotte, NC, United States of America
Journal version in preparation to be submitted Summer of 2016

5.1 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.

5.2 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 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 dis-

tribute 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 5.1 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.

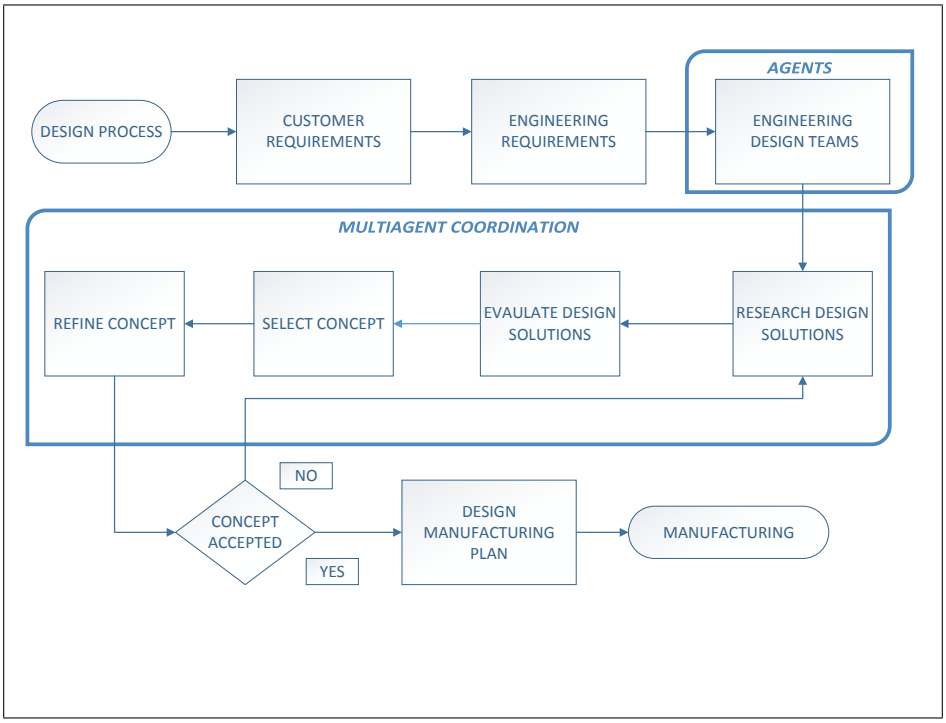

Figure 5.1: Design Process

5.3 Background

5.3.1 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 [53]. 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* [18]. This is a dramatic change in perspective for system design, promising a reduction (or elimination) of cost and schedule overruns [6, 16] by identifying high value designs for development. Value-Centric Design can be considered part of the larger field of Decision-Based Design (DBD) [32, 76]. 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.

5.3.2 Multiagent Coordination

Multiagent coordination is a key research area in agent-based approaches to automation [71]. 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 acheive the system objective. The key challenge is that a system designer needs to address two major credit assignment problems: *structural* and *temporal* [9, 71] 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, 84].

The temporal credit assignment problem has been extensively studied through single-agent reinforcement learning [9, 57]. 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 [7, 31]. Organizational structures decompose the agents themselves into roles that enable coordinated behavior [1, 86].

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 [78, 83]. Providing agents with objectives that satisfy these two properties (formalized in[77, 83]) 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, 37].

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.

5.3.3 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 [23]. 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.

5.3.3.1 Coevolution:

Coevolutionary Algorithms (CEAs) are an extension of evolutionary algorithms and are often well-suited for multiagent coordination domains [22]. 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 [55]. 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 [56]. 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.

5.3.3.2 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 [59]. 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 [81]. 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") [81].

5.3.3.3 Difference Evaluation Function Theory:

The agent-specific difference evaluation function is defined as:

$$D_i(z) = G(z) - G(z_{-i} + c_i) \tag{5.1}$$

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:

$$\frac{\partial G(z)}{\partial a_i} = \frac{\partial D_i(z)}{\partial a_i} \tag{5.2}$$

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 5.23 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.

5.4 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 5.2. 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.

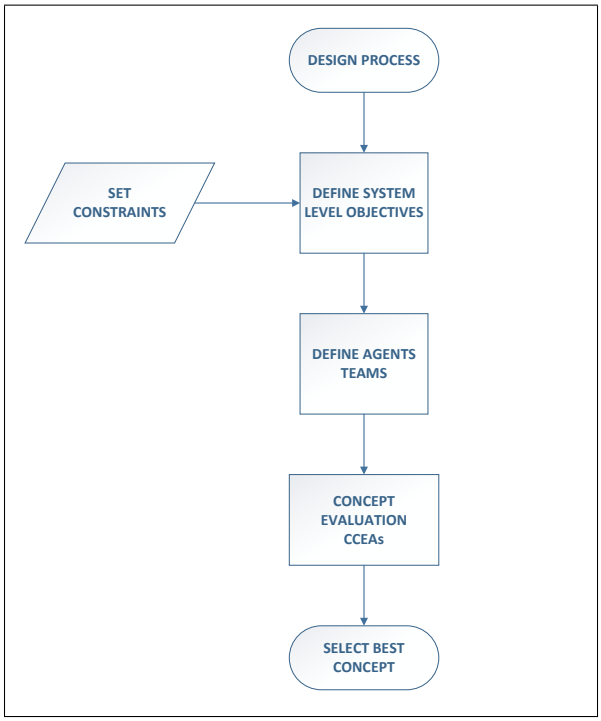

Figure 5.2: Design Process for Agents

5.4.1 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 [35]. 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 [35]. 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 5.1. 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.

5.4.2 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 5.3. 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.

5.4.3 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 Table 5.1. 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,

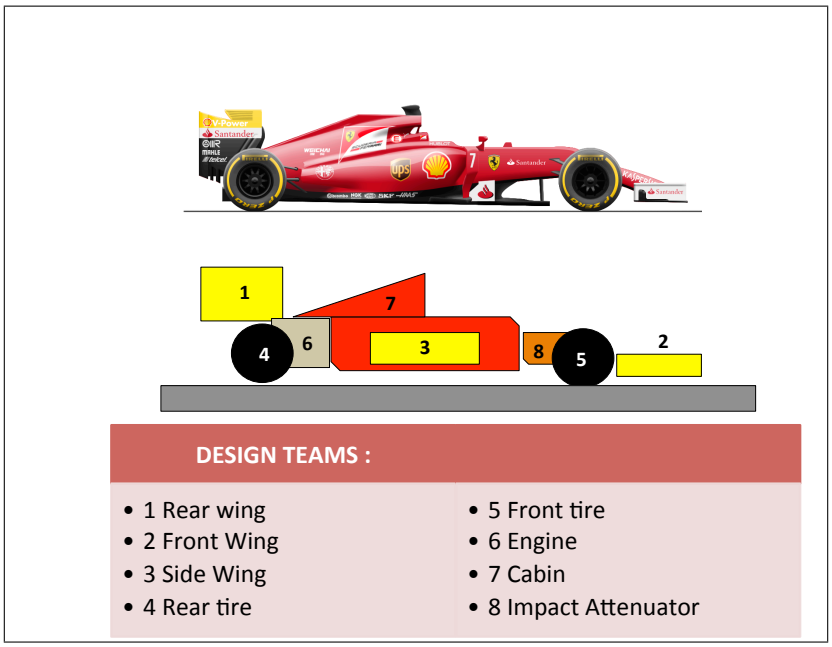

Figure 5.3: Racing vehicle Model

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 5.1 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.

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

Table 5.1: 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

5.4.4 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:

$$m_{total} = m_{rw} + m_{fw} + 2 \cdot m_{sw} + 2 \cdot m_{rt} + 2 \cdot m_{ft} + m_e + m_c + m_{ia} \tag{5.3}$$

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:

$$\begin{aligned} CG_y = \frac{m_{rw}y_{rw} + m_{fw}y_{fw} + m_e y_e + m_c y_c + m_{ia}y_{ia}}{m_{total}} + ... \\ + \frac{2(m_{sw}y_{sw} + m_{rt}r_{rt} + m_{ft}r_{ft})}{m_{total}} \end{aligned} \tag{5.4}$$

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:

$$C_l = 2\pi \frac{AR}{AR + 2}\alpha \tag{5.5}$$

The drag coefficient C_d of a wing is defined as:

$$C_d = \frac{C_l^2}{\pi AR} \tag{5.6}$$

The overall downforce F_d of a wing is given by:

$$F_d = \frac{1}{2}\alpha hw\rho_{air}v_{car}^2 C_l \tag{5.7}$$

The overall drag F_R of a wing is given by:

$$F_R = \frac{1}{2}\rho_{air}v_{car}^2 C_d wh \tag{5.8}$$

For the cabin, we assume a drag coefficient $C_{d,c}$ of 0.04 for a streamlined body [54].
The overall drag of the vehicle is thus:

$$F_{R,total} = F_{d,rw} + F_{d,fw} + 2F_{d,sw} + F_{d,c} \tag{5.9}$$

and the overall downforce of the vehicle is:

$$F_{d,total} = F_{R,rw} + F_{R,fw} + 2F_{R,sw} \tag{5.10}$$

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:

$$C = 0.005 + \frac{1}{p}(0.01 + 0.0095v_{car}^2) \tag{5.11}$$

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

$$R_{roll} = \frac{Cm_{tot}g}{r} \tag{5.12}$$

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:

$$R_{tot} = F_{d,total} + R_{roll} \tag{5.13}$$

The efficiency η of the engine is given by:

$$\eta = \frac{R_{tot} v_{car}}{\Phi_e} \tag{5.14}$$

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

$$F_{wheels} = \frac{\tau_e \eta \omega_e}{r_{rt} \omega_{wheels}} \tag{5.15}$$

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

$$\omega_{wheels} = \frac{v_{car}}{r_{rt}} \tag{5.16}$$

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

$$\begin{aligned} \sum F &= m_{total} a_{car} \\ F_{wheels} - R_{total} &= m_{total} a_{car} \\ a_{car} &= \frac{F_{wheels} - R_{total}}{m_{total}} \end{aligned} \tag{5.17}$$

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:

$$\delta = \frac{F_{crash} l_{ia}}{w_{ia} h_{ia} E_{ia}} \tag{5.18}$$

The crash force F_{crash} is defined as:

$$F_{crash} = \frac{m_{total} v_{car}^2}{2\delta} \tag{5.19}$$

Combining Equations 5.18 and 5.19 yields:

$$\begin{aligned} F_{crash} &= \frac{m_{total} v_{car}^2}{2\frac{F_{crash} l_{ia}}{w_{ia} h_{ia} E_{ia}}} \\ \Rightarrow F_{crash}(2F_{crash} l_{ia}) &= m_{total} v_{car}^2 w_{ia} h_{ia} E_{ia} \\ \Rightarrow F_{crash}^2 &= \frac{m_{total} v_{car}^2 w_{ia} h_{ia} E_{ia}}{2 l_{ia}} \\ \Rightarrow F_{crash} &= \sqrt{\frac{m_{total} v_{car}^2 w_{ia} h_{ia} E_{ia}}{2 l_{ia}}} \end{aligned} \tag{5.20}$$

Impact Attenuator Volume:

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

$$V_{ia} = l_{ia} w_{ia} h_{ia} \tag{5.21}$$

5.4.5 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:

$$G(z) = -w_m m_{total} - w_{CG} CG_y - w_R F_{R,total} + w_d F_{d,total} + ... \\ + w_a a_{car} - w_{crash} F_{crash} - w_{iav} V_{ia} \tag{5.22}$$

where w_i is a weight corresponding to objective i.

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

$$D_i(z) = G(z) - G(z_{-i} + c_i) \tag{5.23}$$

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.

5.4.6 Constraints

The constraints used for the vehicle were set according to the Rules of the 2016 Formula SAE Rules [35]. 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.

5.4.7 CCEAs Implementation

The approach to optimizing the vehicle design using cooperative coevolution is shown in Figure 5.4. 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 $a + \epsilon$, 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 Figure 5.4), 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
```

Figure 5.4: Cooperative Coevolutionary Algorithm

5.5 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 [29]. GFR is a Formula SAE team formed by an international cooperation between the BA Racing Team from Duale Hochschule Baden-Wrttemberg-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.

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 5.22) are defined in Table 5.2.

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 5.5 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.

Table 5.2: 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

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 5.3, 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.

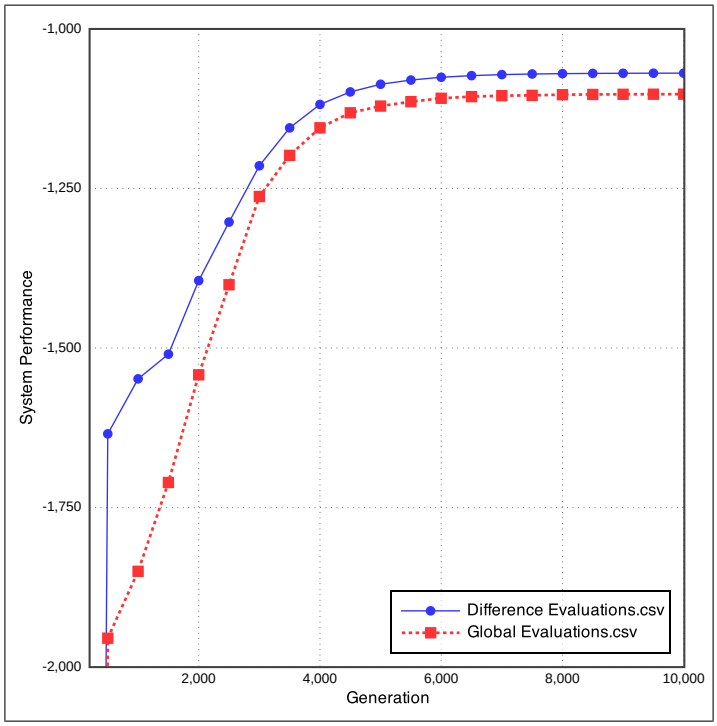

Figure 5.5: Preliminary results: difference evaluations provide better designs than global evaluatons.

Table 5.3: Comparison of the SAE solution vs. the design found using CCEA.

Objective	SAE Solution	CCEA Solution
Mass	34.88	**34.45**
CG_y	0.277	**0.260**
Drag	**217.108**	281.720
Downforce	318.748	**390.593**
Acceleration	0.250	**0.333**
Crash Force	40.92	**18.08**
IA Volume	0.009	**0.013**

5.6 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.

More specifically, the results from Table 5.3 illustrate 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 Formula 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.

5.7 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.

Chapter 6: Concluding Remarks

The preliminary results presented in the two manuscripts show how the implementation of a learning algorithm within autonomous agents increases the performance of the system. The global objective presented in the first manuscript was to maximize productivity. Productivity is defined for this system as the number of products produced at each time step divided by the amount of products that are already placed. (How much is produced divided by how much it took to produce it). The levels of productivity achieved by the agents using learning algorithm were low. This could be caused by the defined global objective. The system objective as is currently defined is just gathering resources and expanding the factory elements without any specific target. The journal version of this manuscript would redefine the global objective of the system, and incorporate the learning algorithm in the all the agents of the system (workers and producers) case 4 from Table 4.1.

The multiple objectives presented in the second manuscript were a weighted sum of the different objectives. Each weight was adjusted according to the preferences of the client, and the objectives were built using the simplified version of the racing vehicle and racing event. The results were compared with the Global Formula Racing Team in Table 5.3. The team of agent's design achieved better performance than the Global Formula Racing (GFR) design on six of seven objectives. However, the design created by the team of agents cannot be completely compared to the design of the GFR because the global objectives of each vehicle model are different. To properly compare the two systems it would be necessary to update the model fidelity and the multiple objectives of the system used in this project. The journal version of this manuscript will incorporate more detail to the model of the system and to the racing event, which will create a new set of objectives to maximize and minimize, and incorporate new subsystem and agents to the case study.

It is fundamental to notice the importance on defining an accurate global objective function for the system while implementing a learning algorithm.

Two different learning algorithms were presented in this document. The first manuscript

used a $Q - learning$ with one type of agent (worker). Three scenarios were considered: local rewards, difference rewards, and pre-programmed behavior. Local rewards and difference rewards were the scenarios where the learning algorithm ($Q - learning$) was incorporated. The second manuscript used a *cooperative coevolutionary algorithm* with eight different types of agents. Two scenarios were considered: cooperative coevolutionary algorithm to evaluate the team performance, and cooperative coevolutionary algorithm with difference evaluation to evaluate the team and agents' performance.

From the first manuscript we learned that $Q - learning$ was suitable to the system as the defined agents have clear actions and states. Each worker will take an action (deliver resource, gather resource, etc.) related to the state of the agent. From the results obtained it was hard to see a distinction between the use local reward and difference reward. In both scenarios the productivity of the system was low but the system was able to keep expanding and working. With the current global objective it is hard to see an impact on the system performance when the difference reward is implemented. Additionally, the implementation of $Q - learning$ is mostly used in discrete domain, which was suitable for this case study. However, to implement the multiagent coordination within the design of complex engineered systems, it would be necessary to explore other types of learning algorithms that work in discrete and continuous domains.

In the second manuscript we incorporated a cooperative coevolutionary algorithm as the learning algorithm. From the results obtained we were able to observe how the incorporation of cooperative coevolutionary algorithm with difference evaluations evaluated the performance of the team and each agent member of the team.

Figure 6.1 illustrates the process completed by the cooperative coevolutionary algorithm to evaluate the system performance. Each agent responsible for a specific subsystem will generate a random population of possible solutions. One selection from each agent is selected and grouped in one team. The team is then evaluated using the global objective defined to the system. According to the system performance the team will get a fitness assignment. The fitness assignment is used to evaluate the performance of the team. The agent member of the evaluated team used the fitness information to generate new popu-

lation for each agent to refine future solutions. The implementation of the cooperative coevolutionary algorithm allowed us to evaluate the performance of the team. However, if we know how well a team of agents performs when they are all collaborating, we next need to explore how we can evaluate the performance of a particular agent in that team. The implementation of the difference evaluation function allows the system to solve the credit assignment problem. Using the difference evaluation agents can measure the impact of their actions in the performance of the system.

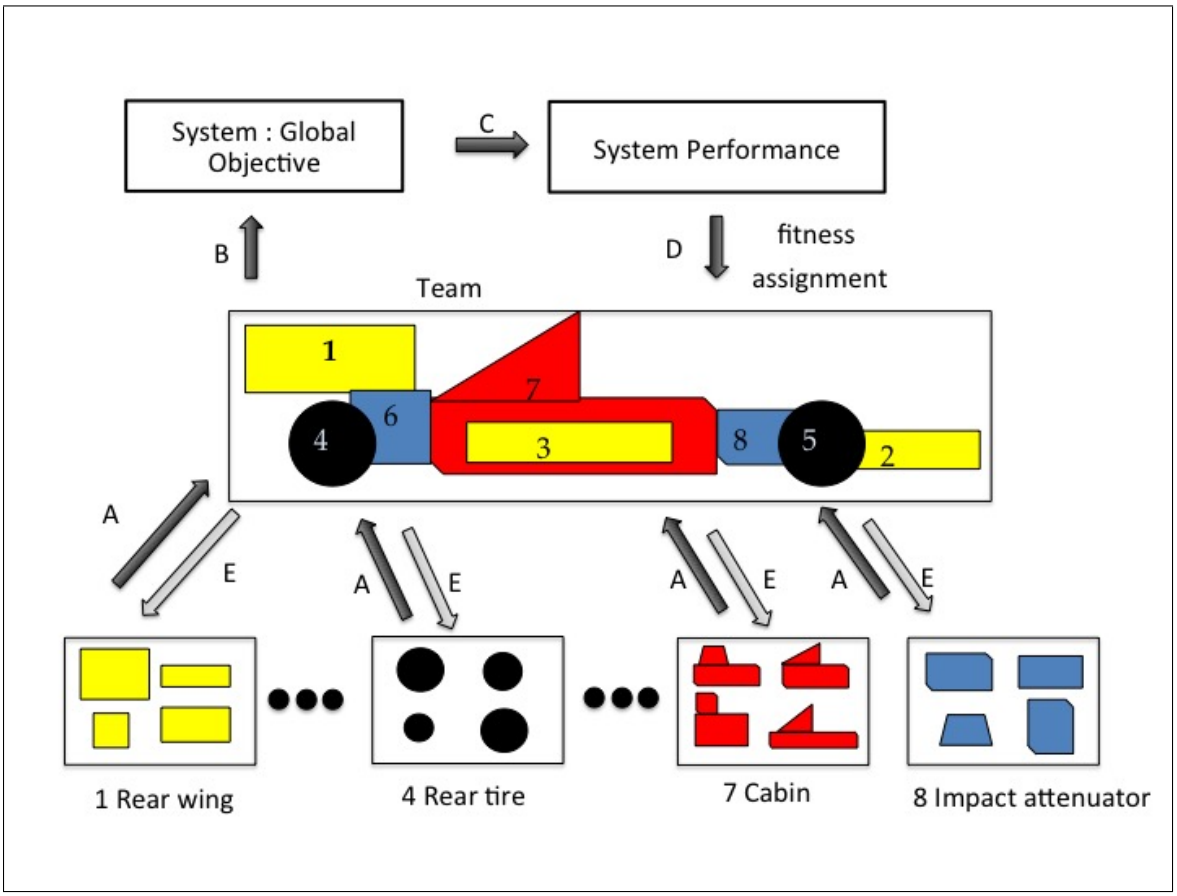

Figure 6.1: Cooperative Coevolutionary Algorithm

The results from the second manuscript are very promising. The implemented learning algorithm was shown to effectively evaluated the performance of the team and their indi-

vidual members. These preliminary results open the door to new research questions that are part of the future work of this research. But one important question remain: how can we start implementing the cooperative evolutionary algorithm with difference evaluation with an integrated concurrent engineering team such as Team X or the Global Formula Racing Team (GFR) at OSU?

The preliminary results of this research are going to be presented to the leaders of GFR. And we can discuss the potential benefits of incorporating this methodology into their design process. Future work will study designing the racing vehicle for the Formula SAE 2017 competition in parallel between GFR team and the presented CCEA methodology. The collaboration will provide knowledge and experience to increase the fidelity of the current model. The results from the applied methodology can be used to improve the design decision process inside the GFR team. Testing the final design in a competition event would validate the implementation of a cooperative coevolutionary algorithm with difference evaluation as an optimization tool with an integrated concurrent engineering.

Chapter 7: Conclusions

As complex engineered systems continue to grow in size or complexity, it is important to develop new methodologies that allow engineers to model and evaluate such systems faster and more accurately. The long-term goal of this research 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 system meeting high-level design objectives.

The implementation of the multiagent coordination approach provides one avenue for such exploration. The results from the case study prove that complex system design, undertaken with respect to a variety of design objectives, is in fact similar to the multiagent coordination problem. This research translates different techniques used to solve the multiagent coordination problem to the design of complex engineered systems.

The problem with multiagent coordination, as many researchers have pointed out, is the credit assignment between agents. As the number of agents increases in the system, it becomes difficult to track the contribution and collaboration of each agent to achieve the predetermined objective. This is especially true when agents need to interact with a large number of agents performing a similar task; or when different agents need to collaborate performing different sub-objectives. When translated to to designing complex engineered systems, the presented approach 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. Multiagent coordination approaches, can 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).

However, significant challenges exist in adapting this concept to designing complex systems to meet design goals; while operating in stochastic and often unpredictable environments. More research, experiments, and simulations using systems containing large number of agents and interaction with the environment are required to directly address this

challenge.

Within the first manuscript, methodologies for using and understanding multiagent coordination as designers were introduced. The manuscript presents an approach for coordinating a conceptual model of a self-replicating system. The arrival of a set of agents on an unknown planet is simulated, whereby these simple agents will expand into a self-replicating factory using the regolith gathered from the surface of the planet. The results on this work demonstrated that the implementation of a learning algorithm allows a large number of different agents to complete simultaneous and different tasks in order to complete a desired objective. However, there appears to be quite a bit of room for improvement as the productivities achieved by the system of agents were very low. The low productivities were thought to be caused by shortcomings of using non-learning on one set of agents (producers), and might be remedied by using producers that learn.

Within the second manuscript, a hybrid approach for the design of complex engineered systems was introduced. With this method engineering design decisions can be reconciled without the need for either detailed interaction models or external legislating 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 presented approach demonstrates that multiagent coordination can be used to design a complex system. A team of autonomous agents was able to design a Formula racing vehicle, with the implementation of a cooperative coevolutionary algorithm.

Multiagent coordination offers an intriguing tool to assist the design of complex engineered systems. The work presented in this thesis provides a first look into the potential of using multiagent coordination for the engineering design process. Nevertheless, more research is necessary to demonstrate the accuracy and benefits of such methodology. Is it also necessary to incorporate the presented methodology within different design approaches such as Integrated Concurrent Engineering (ICE), Multi-Objective Optimization tools (MOO), and Decision Based Design (DBD).

VITA

Nicolás Soria earned his Bachelor of Science degree in Mechanical Engineering from Universidad San Francisco de Quito, in Ecuador. While finishing his Master of Science in Mechanical Engineering, he developed research skills and critical thinking. His interests are in investigating and improving the design of complex engineered systems. His next objective is to complete the qualifying exams to become a PhD candidate.

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.

Bibliography

[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*, pages 172–179, May 2007.

[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*, New York, NY, July 2004.

[3] A. K. Agogino and K. Tumer. Handling communication restrictions and team formation in congestion games. *Journal of Autonomous Agents and Multi Agent Systems*, 13(1):97–115, 2006.

[4] C. Bloebaum and A. McGowan. Design of complex engineered systems. *Journal of Mechanical Design*, 132(12):120301–120301, 2010. 10.1115/1.4003033.

[5] R. Braun, P. Gage, and I.M. Implementation and performance issues in collaborative optimization. *Sixth AIAA/USAF MDO Symposium*, 94(4325), Sept 1996.

[6] O. Brown and P. Eremenko. 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*, 2008.

[7] O. Buffet, A. Dutech, and F. Charpillet. Shaping multi-agent systems with gradient reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 15:197–220, 2007.

[8] M. Bugallo and P. Djurić. Complex systems and particle filtering. In *Signals, Systems and Computers, 2008 42nd Asilomar Conference on*, pages 1183–1187, 2008.

[9] L. Busoniu, R. Babuska, and B. De Schutter. A comprehensive survey of multiagent reinforcement learning. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 38(2):156–172, 2008.

[10] J. Chachere, J. Kunz, and R. Levitt. Observation, theory, and simulation of integrated concurrent engineering: Grounded theoretical factors that enable radical project acceleration. *Center for Integrated Facility Engineering CIFE*.

[11] W. Chen and K. Lewis. A Robust Design Approach for Achieving Flexibility in Multidisciplinary Design. *AIAA Journal*, 37(8):982–989, 1999.

[12] W. Chen, M. Wiecek, and J. Zhang. Quality utilitya compromise programming approach to robust design. *Journal of Mechanical Design*, 121(2):179–187, 1999.

[13] G. S. Chirikjian. An architecture for self-replicating lunar factories. niac phase, 1. *Autonomous Robots*, 2004.

[14] V. Cicirello and S. Smith. Wasp-like agents for distributed factory coordination. *splike agents for distributed factory coordination.*, 2004.

[15] C. Coello. A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information systems*, 1(3):129–156, 1999.

[16] P. Collopy. Economic-based distributed optimal design. *AIAA Paper*, 4675, 2001.

[17] P. Collopy and D. Consulting. Value-driven design and the global positioning system. *AIAA paper*, 7213, 2006.

[18] P. Collopy and R. Horton. Value modeling for technology evaluation. *AIAA Paper*, 3622, 2002.

[19] I. Das and J. Dennis. Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems. *SIAM Journal on Optimization*, 8(3):631–657, 1998.

[20] S. Devlin, L. Yliniemi, D. Kudenko, and K. Tumer. Potential-based difference rewards for multiagent reinforcement learning. *International Foundation for Autonomous Agents and Multiagent Systems*, 2014.

[21] S. Eno, L. Mace, J. Liu, B. Benson, K. Raman, and K. Lee. Robotic self-replication in a structured environment without computer control. *Computational Intelligence in Robotics and Automation*, 2007.

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

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

[24] J. R. A. Freitas. Terraforming mars and venus using machine self-replicating systems (srs).

[25] R. Freitas and W. Gilbreath. Advanced automation for space missions. *Journal of the Astronautical Sciences*, 1982.

[26] R. A. Freitas and R. C. Merkle. Kinematic self-replicating machines. *Robotics and Autonomous Systems*, 2004.

[27] A. Freundlich, A. .Ignatiev, C. Horton, M. Duke, P. Curreri, and L. Sibille. Manufacture of solar cells on the moon. *Conference Record of the Thirty-first IEEE*, 2005.

[28] A. Garcia, J. Kunz, M. Ekstrom, and A. Kiviniemi. Building a project ontology with extreme collaboration and virtual design and construction. *Center for Integrated Facility Engineering Technical Report. Stanford University, Stanford, CA*.

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

[30] F. Grabowski and D. Strzalka. Simple, complicated and complex systems – the brief introduction. In *Human System Interactions, 2008 Conference on*, pages 570–573, 2008.

[31] M. Grzes and D. Kudenko. Plan-based reward shaping for reinforcement learning. In *Intelligent Systems, 2008. IS '08. 4th International IEEE Conference*, volume 2, pages 1022–1029, 2008.

[32] G. A. Hazelrigg. A framework for decision-based engineering design. *Journal of Mechanical Design*, 120(4):653–658, 1998.

[33] J. Horn, N. Nafpliotis, and D. Goldberg. Multiobjective optimization using the niched pareto genetic algorithm. *IlliGAL report 93005*, pages 1–32, 1993.

[34] H. Huang, W. Wu, and C. Liu. A coordination method for fuzzy multi-objective optimization of system reliability. *Journal of Intelligent & Fuzzy Systems*, 16(3):213–220, 2005.

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

[36] C. Juang and C. Lu. Ant colony optimization incorporated with fuzzy q-learning for reinforcement fuzzy control. *Systems, Man and Cybernetics", year = 2009*.

[37] M. Knudson and K. Tumer. Coevolution of heterogeneous multi-robot teams. In *Proceedings of the Genetic and Evolutionary Computation Conference*, Portland, OR, July 2010.

[38] I. M. Kroo and I. Sobieski. Collaborative optimization using response surface estimation. *AIAA*, 98(0915), 1993.

[39] K. Lackner and C. Wendt. Exponential growth of large self-reproducing machine systems. *Mathematical and Computer Modelling*, 1985.

[40] I. Lee, K. Choi, L. Du, and D. Gorsich. Dimension reduction method for reliability-based robust design optimization. *Computers and Structures*, 86(13):1550–1562, 2008.

[41] K. Lee and G. S. Chirikjian. An autonomous robot that duplicates itself from low-complexity components. *Robotics and Automation (ICRA)*, 2010.

[42] K. Lewis. Making sense of elegant complexity in design. *Journal of Mechanical Design*, 134(12):120801–120801, 2012.

[43] P. Liu, Q. Zhang, X. Yang, and L. Yang. Passivity and optimal control of descriptor biological complex systems. *IEEE Trans Autom Control*, 53:122–125, January 2008.

[44] G. Mark. Extreme collaboration. *Communications of the ACM*, 45(6):89–93, june 2002.

[45] M. J. Mataric. Reinforcement learning in the multi - robot domain. *AAMAS - 8th International Conference on Autonomous Agents and Multiagent Systems*, 2009.

[46] C. Mattson and A. Messac. Pareto frontier based concept selection under uncertainty, with visualization. *Optimization and Engineering*, 6(1):85–115, 2005.

[47] R. Melchers. *Structural reliability: analysis and prediction*. John Wiley & Sons, Chichester, England, 1987.

[48] P. T. Metzger. Nature's way of making audacious space projects viable. *100 Year Starship Symposium*, 2011.

[49] A. A. Minai, D. Braha, and Y. Bar-Yam. Complex engineered systems: A new paradigm. In D. Braha, A. A. Minai, and Y. Bar-Yam, editors, *Complex Engineered Systems : Science Meets Technology*, chapter 1, pages 1–21. Springer, Berlin, 2006.

[50] R. A. Mole. Terraforming mars with (largely) self reproducing robots. *The Mars Society*, 2003.

[51] N. Monekosso, P. Remagnino, and A. Szarowicz. An improved q-learning algorithm using synthetic pheromones. *From Theory to Practice in Multi-Agent Systems*, 2002.

[52] M. Moses, H. Ma, K. C. Wolfe, and G. S. Chirikjian. An architecture for universal construction via modular robotic components. *Robotics and Autonomous Systems*, 2014.

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

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

[55] 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.

[56] 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.

[57] S. Paquet and L. Tobin. An online pomdp algorithm for complex multiagent environments. In *Proceedings of the Autonomous Agents and Multiagent Systems Conference*, pages 970–977, 2005.

[58] V. Pareto. *Manual of political economy*. ed. Schwier, A. S., Page, A. N.,Kelley, A.M., Augustus M. Kelley Publishers, New York, 1971.

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

[60] S. Proper and P. Tadepalli. Solving multiagent assignment markov decision processes. *AAMAS - 8th International Conference on Autonomous Agents and Multiagent Systems*, 2009.

[61] A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, and S. Tarantola. *Global sensitivity analysis: the primer*. John Wiley & Sons, Ltd., Chichester, UK, 2008.

[62] Y. Sawaragi, H. Nakayama, and T. Tanino. *Theory of multiobjective optimization*, volume 176. Elsevier Science, Orlando, FL, 1985.

[63] T. Schuman, O. L. de Weck A. Author, and J. Sobieski. Integrated system-level optimization for concurrent engineering with parametric subsystem modeling. *46th AIAA ASME ASCE AHS ASC Structures, Structural Dynamics and Materials Conference*, 46(2199):1–20, April 2005.

[64] J. Sercel. Ice heats up design. *Aerospace America*.

[65] T. W. Simpson and J. R. R. A. Martins. Multidisciplinary design optimization for complex engineered systems: Report from a National Science Foundation workshop. *Journal of Mechanical Design*, 133(10):101002–101002, 2011. 10.1115/1.4004465.

[66] M. Sims, D. Corkill, and V. Lesser. Automated Organization Design for Multi-agent Systems. *Autonomous Agents and Multi-Agent Systems*, 16(2):151–185, 2008.

[67] J. Sobieszczanski-Sobieski, J. Agte, and R. Sandusky. Bilevel integrated system synthesis (bliss). *NASA/TM*.

[68] J. Sobieszczanski-Sobieski, T. Altus, M. Phillips, and R. Sandusky. Bilevel integrated system synthesis for concurrent and distributed processing. *AIAA*, 41(0915):1996 –2003, Oct 2003.

[69] J. Sobieszczanski-Sobieski, M. Emiley, J. Agte, and R. J. Sandusky. Advancement of bi-level integrated system synthesis (bliss). *AIAA*, 38(0421), Jan 2000.

[70] R. Steuer. *Multiple criteria optimization: Theory, computation, and application*. Krieger, Malabar, FL, 1989.

[71] P. Stone and M. Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, July 2000.

[72] J. Summers and J. Shah. Mechanical Engineering Design Complexity Metrics: Size, Coupling, and Solvability. *Journal of Mechanical Design*, 132(2), 2010.

[73] R. Tappeta and J. Renaud. Interactive multiobjective optimization design strategy for decision based design. *Journal of Mechanical Design*, 123:205, 2001.

[74] A. Team. Advanced projects design teams assessment of esas world space observatory proposal. *Assessment report - National Aeronautics and Space Administration NASA*, 1.

[75] M. Tiller. *Introduction to Physical Modeling with Modelica*, volume 615. Springer Science+Business Media, LLC, 2001.

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

[77] K. Tumer and D. Wolpert, editors. *Collectives and the Design of Complex Systems*. Springer, New York, 2004.

[78] K. Tumer and D. Wolpert. A survey of collectives. In *Collectives and the Design of Complex Systems*, pages 1,42. Springer, 2004.

[79] H. J. Wassenaar and W. Chen. An Approach to Decision-Based Design with Discrete Choice Analysis for Demand Modeling. *Transactions of the ASME: Journal of Mechanical Design*, 125(3):490–497, 2003.

[80] G. Weis. *Multiagent Systems*. MIT Press, 2013.

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

[82] E. Wiewiora. Potential-based shaping and q-value initialization are equivalen. *JAIR*, 2003.

[83] D. H. Wolpert and K. Tumer. Optimal payoff functions for members of collectives. *Advances in Complex Systems*, 4(2/3):265–279, 2001.

[84] D. H. Wolpert, K. Tumer, and J. Frank. Using collective intelligence to route internet traffic. In *Advances in Neural Information Processing Systems - 11*, pages 952–958. MIT Press, 1999.

[85] J. Wu and S. Azarm. Metrics for quality assessment of a multiobjective design optimization solution set. *Journal of Mechanical Design (Transactions of the ASME)*, 123(1):18–25, 2001.

[86] 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*.