AN ABSTRACT OF THE THESIS OF

Daniel E. Hulse for the degree of Master of Science in Mechanical Engineering
presented on June 4, 2018.

Title: Optimization Across Decomposition: A Multiagent Model of Collaboration in Decomposed System Design

Abstract approved: _____

Irem Y. Tumer Christopher Hoyle

Complex engineered systems design is a collaborative activity. To design a system, experts from the relevant disciplines must work together to create the best overall system from their individual components. This situation is analogous to a multiagent system in which agents solve individual parts of a larger problem. Current multiagent models of design teams, however, do not capture this distributed aspect of design teams–instead representing designers as agents which control all variables of the problem. This paper presents a new model of design which captures the distributed nature of complex systems design by decomposing the ability to control variables of the design to individual computational designers acting on a design problem with shared constraints. These designers are represented as a multiagent learning system which is shown in this paper to perform similarly to a centralized optimization algorithm on the same domain. When used as a model, this multia-

gent system is shown to perform better when the level of designer exploration is not decayed but is instead controlled based on the increase of design knowledge, suggesting that designers in multidisciplinary teams should not simply reduce the scope of design exploration over time, but should adapt based on changes in their collective knowledge of the design space to achieve the best design outcome. This multiagent system is further shown to produce better-performing designs when computational designers design collaboratively as opposed to independently, confirming the importance of collaboration across disciplinary boundaries in complex systems design.

Optimization Across Decomposition: A Multiagent Model of Collaboration in Decomposed System Design

by

Daniel E. Hulse

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented June 4, 2018
Commencement June 2018

Master of Science thesis of Daniel E. Hulse presented on June 4, 2018.

APPROVED:

Co-Major Professor, representing Mechanical Engineering

Co-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.

Daniel E. Hulse, Author

ACKNOWLEDGEMENTS

Here I would like to express gratitude to the many people who have helped me through the journey of developing this work. First, I would like to thank my co-advisors, Dr. Irem Tumer and Dr. Chris Hoyle. I would especially like to thank Dr. Tumer for giving me the initial opportunity to perform research in what I find to be an immensely interesting field of study, and for helping me to become a better researcher. I would to thank Dr. Hoyle for providing support, feedback, ideas and direction that helped me throughout the development of this work. Second, I would like to thank my co-authors, Dr. Kagan Tumer for their feedback and contributions which helped develop this thesis into a "research-worthy" paper and I am appreciative of his ability to critique and provide ideas for further development. Thirdly, I would like to thank the other OSU students that have helped me in the work by providing ideas, critiques, help with prototypes, and all sorts of guidance, including Brandon Gigous, Hannah Walsh, and Nico Soria. Finally, I would like to thank all of my friends, family, and former teachers who have inspired me and supported my pursuit of graduate education.

CONTRIBUTION OF AUTHORS

This thesis was developed in conjunction with Dr. Kagan Tumer, Dr. Chris Hoyle, and Dr. Irem Tumer. Early versions were developed with Brandon Gigous, resulting in these papers:

1. D. Hulse, B. Gigous, K. Tumer, C. Hoyle, I.Y. Tumer, *Towards a Distributed Multiagent Learning-Based Design Optimization Method.* in *ASME IDETC/CIE 2017 Design Automation Conference.* Cleveland, OH. 2017.

2. D. Hulse, K. Tumer, C. Hoyle, I.Y. Tumer, *Modelling Collaboration in Parameter Design Using Multiagent Learning.* in *Design Computing and Cognition, DCC 2018.* Milan, Italy. 2018. Accepted.

3. D. Hulse, K. Tumer, C. Hoyle, I.Y. Tumer, *Modelling Multidisciplinary Design with Multiagent Learning.* 2018. Accepted.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

Optimization Across Decomposition: A Multiagent Model of Collaboration in Decomposed System Design

1 Introduction

As the profession has advanced, engineering has sought to solve increasingly complex problems. This complexity inevitably creates greater challenges for designers, and for engineering organizations as a whole. For example, each new generation of fighter aircraft acquired by the United States Department of Defense has roughly doubled in cost over the previous aircraft [41], with plurality of increased cost of being attributable not to labor or material, but to complexity [5].

As a result, more attention must be paid in the engineering design process that may have been easy to neglect in low-complexity systems. As systems increase in complexity, engineers must focus not just on accurate analysis, but design decision-making (e.g., what requirements or features to develop), as the majority of project cost increase is incurred through these decisions [8]. Furthermore, the management of the design process becomes a challenge in its own right, both in terms of coordinating the designers in an integrated way to make a system that works as a whole [21] and in achieving a desired level of concurrency of engineering work so as to not increase project cost and re-work [6]. These problems can be thought of as a part of two distinct but related aspects of the engineering design of complex systems: decision-making and the decomposition of a system architecture across disciplinary, organizational, or functional lines.

Additionally, coordinating the design of complex engineered systems provides a significant challenge to engineering organizations. When designing a complex engineered system with a number of different analyses and concerns, engineers must take into account the many interactions between subsystems in order to optimize the performance of the design. Often this involves coordinating specialists from the relevant disciplines. In the design of a spacecraft, for example, experts in propulsion, computer science, power systems, and many other areas must work together to create a working design [42]. As a result, coordinating these design processes effectively can provide significant increases to the performance of the design organization, increasing the throughput of designs and lowering costs [69].

Several frameworks have been used to approach coordination complex engineered systems design, such as integrated concurrent engineering [69] [49], multidisciplinary design optimization [50], and systems engineering [61]. In integrated concurrent engineering, a small team of disciplinary experts, guided by a facilitator, design a complex engineered system over the course of a few three-hour meetings [69] [49]. In multidisciplinary design optimization, an optimization is broken up between disciplines or components to co-optimize different parts of a complex system autonomously [50]. In the broader field of systems engineering, different disciplinary areas encompassing all parts or concerns with respect to a design must be brought together to design a functioning system [61]. These approaches rely on delegating the design of different subsystems, components, or disciplinary areas to experts or disciplinary groups, which provides significant challenges to coordinating design activities, as no individual expert is able to comprehensively know how

a design change will affect the design as a whole.

To study this coordination in complex systems design, this thesis was prompted by the following research question:

Given a system has been decomposed across disciplinary lines, how should designers as agents act to achieve optimal design outcomes, provided the decision problem of which metrics to optimize has been solved?

This question was approached by developing a multiagent model of decomposed design to study the effect of agent behaviors on design outcomes.

1.1 Motivation and Previous Work

Clearly, the complexity of engineered systems has lead to increased challenges for design organizations. Particularly, as engineered systems increase in scale, one person alone cannot develop the system: there is simply too much work to do and too much expertise and knowledge to incorporate for the system to be developed in a reasonable amount of time. As a result, systems are decomposed and assigned to different people who design different parts. To manage these decomposed processes, frameworks have been developed such as systems engineering [61] and integrated concurrent engineering [69] [49] to coordinate the disciplines in a design process and develop systems that operate optimally both at the system and part-levels. However, while these design processes have been implemented and studied, there is little underlying theory about how they solve the coordination problems intrinsic to system decomposition, and little ability to perform non-qualitative tests about how they perform.

Towards this end, the development of multiagent models which capture aspects of the complex systems design process is compelling. The designers (or disciplines, in the case of multidisciplinary design optimization) in each of these organizations are analogous to the agents in a multiagent system in that they act autonomously, but must work together to achieve a desirable design outcome. Additionally, complex engineered systems and the complex systems design process may be represented as networks [9] [70], a typical multiagent systems domain. Multiagent systems have also had success designing policies for other complex systems domains, such as power grids [60] [16], air traffic control [2], and multi-robot coordination [86]. Consequentially, a multiagent approach has been used as a framework both for designing complex engineered systems [48] [87] [32] and for modelling design organizations [35] [36].

Multiagent models can enable us to gain insight into the complexities of the design process. The effect of designer actions and team-wide behaviors result on overall team performance would be difficult or expensive to test in practice, due to the amount of resources required in and possible noise factors inherent to conducting such a test. These models in turn enable design research to inform and generate new theories about how designer behaviors and team structures affect team performance based on how the designers behave as agents.

These multiagent models have been used to study negotiation in design settings [34] [40], mental-model formation in teams [17] [65], social learning in teams [66], the effect of design problems on optimal team structure [53], and the effect of designer search behaviors [51]. For the purposes of this paper, these models of

teams may be placed in two categories: those which represent the full problem to agents and those in which agents act on partial solutions. Those in which agents act on full solutions include the CISAT framework [51], the mental model framework presented in [17], commonly used optimization methods such as ant [18] and bee colony optimization [38], and protocol-based multiagent systems [45]. Since these multiagent methods and models of design teams assume that each designer can propose any change in any part of the system, they likely do not represent the coordination challenges present in multi-disciplinary organizations. As a result, while some studies using these frameworks do produce results showing collaboration improves the design process [46] (which would be expected from engineering research showing that removing barriers to collaboration produces a more effective design process [69] [49]), others do notinstead finding that the best team structures maximize autonomy [53]. It is likely that this is because these models do not account for the ability of designers to exclusively act on a local area of the problem and not the entire design.

Multiagent representations of design in which agents act on partial solutions show more promise towards modelling the effect of collaboration across disciplines on design outcomes, however current work using these models has not approached this specific problem. While early work studied the learning of partial solutions [55], this model partitioned the roles of agents such that some agents generate solution fragments while others integrate those fragments, which intrinsically coordinates the solutions of agents, making it inappropriate to study coordination across disciplines. Additionally, the models presented to study social learning in

teams focus on studying the formation of transactional memory and other organizationally desirable behaviors, rather than design outcomes [66] [68] [67]. Finally, while models studying negotiating represent the problems inherent in collaborative design well, with agents acting on different subsystems, these models are somewhat different from the problem of collaborative design represented in this paper in that those agents receive different rewardsit is assumed that an overall objective cannot be used to align agents individual local rewards with the overall purpose of the design [34] [40]. In this paper, however, the coordination challenge presented to designers comes not from differing designer objectives, but from the nature of individual designers acting on different parts of the design problem.

To summarize, current multiagent models of design do not encode many of the the coordination challenges in complex engineered systems design work. This is because they either do not decompose design authority across disciplines as is done in engineering practice, instead giving agents access to the entire design (as in [51] [17] [45]), or they do not study the design outcome, instead focusing on some external organizational metric (as in [66] [68] [67]). A multiagent model of multidisciplinary design gives the same strengths as previous multiagent models while better representing the coordination problems intrinsic to decomposition. This enables the ability to gain insights which are more applicable to design practice in complex, large scale, or multidisciplinary systems.

1.2 Definitions

Listed below are definitions to words that are used throughout this work which have a precise meaning.

Nomenclature

Agent: An entity that acts autonomously within an environment.

Complex Engineered System: An engineered system which displays a level of complexity through some combination of: a dynamic and stochastic environment, inter-component interactions, and multiple sources of value.

Constraint: A function denoting a relationship which must be followed within an optimization problem.

Decomposition: The process by which a system design problem is separated into different interacting sub-problems.

Global Minimum: The absolute lowest location on an objective function.

Learning: A process that stores information from the environment in an agent.

Local Minimum: A location on an objective function which has a lower value than all adjacent locations but may not be the lowest location on the function.

Multiagent Learning: A field of study which studies multiagent systems composed of agents which learn to take good actions based on feedback with the environment.

Multiagent System: A system of multiple agents which act autonomously within an environment.

Nash Equilibrium: A stable minimum reached by multiple agents which cannot be left without both agents moving.

Objective: A function denoting a value which must be minimized (or maximized) within an optimization problem.

Process Minimum: A low point on an objective function after which an optimizing process cannot proceed further. In this work this term is used to describe how the multiagent optimizer reaches minima depending on the process used by agents without using the term "local minimum," which may not always correspond to a process minimum.

System Architecture: The overall structure of a system and its interactions.

1.3 Contributions

This work adds to the body of research by informing complex systems design research about coordination in decomposed design using a multiagent model, as outlined below:

1. *The development of a multiagent learning-based optimizer which mimics the structure of a multidisciplinary design team.*

   An optimization method is developed based on multiagent learning which decomposes the design of each variable to an agent. As a multiagent learning system, this method optimizes by having each agent interact with its environment (in this case, an overall design model) to determine the best action (design for its component). This mimics the structure of a decomposed design team, in which each person is given a different part in the overall system to design.

2. *The use of this multiagent optimizer to model and gain insights about the nature of coordination in multidisciplinary design teams.*

   Using this optimization-based model, the level of coordination between agents is varied to determine how coordination affects the overall merit of the design. Furthermore, the exploratory behaviors of the designers are varied to determine how differing the level of exploration throughout the design process influences the final design.

In doing so, this work introduces a new multiagent model of design teams

specifically targeting complex engineered systems design. This representation allows modelling of independence and collaboration between designers that takes into account the core feature of these processesthe decomposition of the ability to design across the topology of the problem. In the multiagent system introduced in this work, this distribution of design agency is represented by having computational designers control different variables of the model which they iteratively propose to optimize the design. Studying behaviors of designers in this model is then used to provide insights into the complex systems design process. The main challenges in developing this model include:

- devising a multiagent learning approach which approximates the collaborative design process by acting as an effective optimization process in which agents act on different variables of the problem
- extending this approach to be compatible with constraints and mixed integer and continuous variablescommon features of engineering design problems
- adapting and applying this method to an optimization problem where shared constraints between subsystems must be coordinated

The following chapters are organized as follows: Chapter 2 provides background into multiagent systems, related work in decomposition, and previous approaches. Chapter 3 presents the multiagent design model, as well as the quadrotor problem model used to judge performance. Chapter 4 presents results of using this method and discusses related insights. Finally, Chapter 5 summarizes the findings of using this model, and presents limitations and directions for future work.

2 Background

The following chapter provides necessary background into understanding the context of decomposition in design, multiagent optimization approaches and other approaches used in this work, as well as previous work in developing multiagent approaches in design.

2.1 Decomposition in Design

Decomposition is a common strategy for designing systems of any complexity which involves decomposing a single problem into multiple sub-problems that can be solved with a degree of independence. These problems are then often delegated to different designers to solve. This presents two main challenges: decomposing the system appropriately to develop a viable design and coordinating design activities after the system has been decomposed. Research which approaches these challenges (shown in Figure 2.1) is discussed in the following sections.

2.1.1 System Decomposition

As discussed in [85], two main approaches exist for decomposition: the flow-based approach discussed in [56] and [74], and the hierarchical tree-based approach used in SysML.

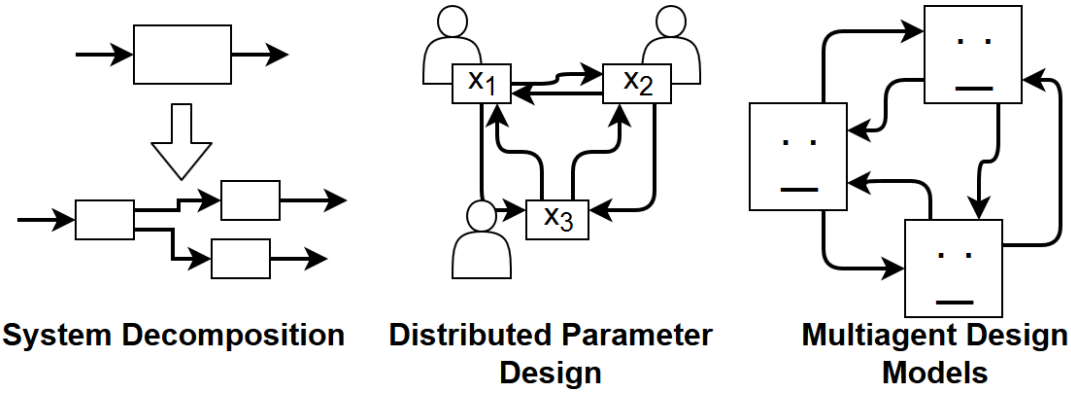

Figure 2.1: Types of decomposition discussed in Section 2.1

Functional decomposition has long been an established part of the engineering design process as a way of breaking down an overall system function into sub-functions [78] [56] [79]. A variety of functional modeling approaches exist across the disciplines of engineering, each with different representations (or lack of representations) of the states, effects, transformation processes, interaction processes, use cases, technical allocation, and stakeholder allocation in the model [19]. Within the design engineering literature, the Functional Basis [74] has been promoted and adopted as a standardized type of functional description which can be used to represent the effects of engineered systems [84] [33]. Within this modelling language, functions are represented as verb-noun pairs which act on the various flows of material, energy, and signal which proceed through the model.

SysML is a widely adopted language for within systems engineering modelling engineered systems which incorporates a variety of diagrams to develop the various perspectives of the system [31]. Functional architectures can be developed in SysML using block definition diagrams from the various requirements and use-

cases of the system [44]. While approaches have been used to attempt to reconcile SysML and the flow-based approaches [43], in general SysML has not been integrated well with engineering design methodology, while the flow-based approaches have [77].

While no explicit modelling language is used to decompose systems in Chapter 3, this background is provided to illustrate how this is decomposition is organized in practice.

2.1.2 Distributed Parameter Design

To study how design teams act on a problem that has been decomposed to different designers, work has been done in distributed parameter design. These studies give control over different variables of an engineered system to different designers and study the effects on the design and design process. In these studies, it has been found that, for individuals acting on coupled tasks, completion time increases geometrically with the number of variables [30]. This was confirmed in [23], which also studied the effect of distributing design problems throughout teams, showing that collaboration took up a large percent of the total team effort which increased with team size. While the authors were able to identify this increased effort as a cost to increasing the number of designers, they did not identify whether collaboration was necessary or beneficialonly that it occurred. More recent work has shown that collaboration and synchrony between designers correlates with better design outcomes in teams [24]. Modeling designer behaviors using markov processes has been

presented as a path for future research [3]. The publication presented in Chapter 3 seeks to approach this path by modeling designers as (markov) reinforcement learning agents.

2.1.3 Multiagent Design Models

Multiagent Design Models have additionally been developed to study how teams perform in a controlled computational setting. Existing models are used to study a variety of behaviors and effects, including the effect of negotiation protocols on design outcome [34] [40], the effect of mental-model formation on design outcome [17] [65], the effect of social learning in teams on the development of transactional memory [66], the effect of design problems on optimal team structure [53], and the effect of designer search behaviors on design outcome [51].

However, the use of these models to study design across a decomposed product depends on the representation used. Many multiagent models, such as the CISAT framework [51], the mental model framework presented in [17], commonly used optimization methods such as ant [18] and bee colony optimization [38], and protocol-based multiagent systems [45] do not represent decomposition very well because each agent in these models is effectively given access to the entire design. This assumption is not appropriate for decomposed design because by definition in a decomposed design process individual designers are only given the ability to design over their own part of the overall problem. Better representations for decomposed design, such as ([55], [66], [68], [67], [34], and [40]) only give agents the

ability to create partial solutions, an assumption which is far more representative of actual practice.

The work presented in Chapter 3 builds on and further informs this research area, by presenting a model in which designers act as agents which learn the best design through interaction with their environment.

2.2 Multiagent Optimization

Developing a multiagent learning-based optimization method requires knowledge of the principles of multiagent learning, optimization, and the problem at hand. This section provides an outline of multiagent systems, multiagent principles which are used in the optimization method, and previous multiagent approaches in engineering design and optimization.

2.2.1 Multiagent Learning

Multiagent learning is a research area of Multiagent Systems, which is a field of artificial intelligence studying multiple agents which interact with each other and the environment with a degree of autonomy [73] [83]. Multiagent systems in general covers a broad range of topics such as communication protocols, coordination, and distributed cognition [83] and a number of architectures exist for designing the cognition of the agents, such as belief-desire-intention, logic-based architectures, and layered architectures. This work relies on the multiagent learning which is

based on using rewards to determine which actions to take in a given state. Each of these processes are outlined in the following sections.

Reinforcement Learning

Reinforcement learning is a method of predicting or determining the value of actions in an uncertain or stochastic environment based on the previous rewards returned for those actions. It is popularly known for its use in the multi-armed bandit problem [75], in which a gambler hopes to maximize his or her returns on a series of slot machines. Reinforcement learning has also been successfully applied to other problems, such as air traffic control [2] and stock price prediction [47] in which it is useful to determine returns over time. The simplest case of reinforcement learning is action-value learning, in which an agent keeps a table of values for its actions, and updates the table at each time step according to:

$$V(a) \leftarrow V(a) + \alpha(r - V(a)) \tag{2.1}$$

where $V(a)$ is the learned value of the action, r is the reward given for the action taken, and α is the learning rate, which controls the learning step size. More sophisticated methods of reinforcement learning, such as Q-learning [82], encode state information in order to allow the agent to learn the values of its actions in different situations, or states, and comparisons with predictions of future rewards.

The Q-learning table assignment is:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha * (r_t + \gamma * max(Q(s_{t+1}, a_{t+1})) - Q(s_t, a_t)) \tag{2.2}$$

where $Q(s_t, a_t)$ is the value given to the action in the current state, α is a learning rate, r_t is the reward at the current state, γ is the preference for future rewards, and $max(Q(s_{t+1}, a))$ is the best value of the next state reached by taking the current action in the current state.

Action Selection

Action selection refers to the way agents take actions based on their learned value in a particular situation. Two common approaches are softmax and epsilon-greedy [83] action selection. For epsilon-greedy action selection, the agent chooses its most highly valued action with probability $1 - \epsilon$ and a random action with probability ϵ. This random action is taken in order to keep the values associated with each action up-to-date with the current conditions of the environment. In softmax action selection, the agent chooses actions with probabilities based on their relative values. The probability p that an agent takes an action a is based on a temperature parameter τ and value $V(a)$, as shown in the following equation:

$$p(a) = \frac{\exp(V(a)/\tau)}{\sum_{i=1}^{n} \exp(V(i)/\tau)} \tag{2.3}$$

where n is the number of actions available to the agent. This temperature parameter τ, similar to the temperature parameter used in simulated annealing, acts as a tolerance for sub-optimal values [64]. For $\tau \to 0$, the agent picks actions greedily, exclusively choosing high-valued actions; for $\tau \to \infty$, the actions become equally probable. Softmax action selection is not to be confused with the softmax normalization (also used in this paper), which is used in neural networks to minimize the influence of outliers in a data-set [62].

Reward Structure

Reward structure refers to how agents are incentivized based on the results of their actions [83]. In a local reward, agents are incentivized based on the directly observable results of their actions. In a global reward, agents are incentivized based on the total result of the actions of all of the agents. In a difference reward, agents are rewarded based on their individual contribution to global reward. This is calculated by finding the difference between the current global reward, and a hypothetical world in which the agent took a different "counterfactual" action [2] [86]. These rewards have attributes that affect the performance of the system as a whole–local rewards are typically most learn-able, global rewards are most aligned with the global behavior, and difference rewards capture the good attributes of both [1].

2.2.2 Multiagent Approaches in Engineering Design

Multiagent systems have been applied to engineering design problems in a variety of forms, as optimization methods, real and proposed frameworks for design, computational synthesis systems, models of design organizations, and in previous research. These applications are summarized below.

Multiagent Optimization Methods

A variety of optimization methods have been developed in the past using a multiagent paradigm, including particle swarm optimization, ant colony optimization, and bee colony optimization. In particle swarm optimization, candidate solutions travel through the design space based on information about their own best design and the best design found by all of the agents [39]. In ant colony optimization, ants leave "pheromones" communicating the fitness of each solution found and stochastically follow previous trails based on the strength of pheromone [18]. In bee colony optimization, bees communicate the fitness of their route to the next population of bees, which stochastically follow new routes based on the communicated fitness [38]. In each of these optimization methods, agents control candidate solutions rather than design parameters. As a result, these optimization methods are not as analogous to the complex systems design problem as the method presented in this work, making them unfit (as with the models referenced in the introduction) to model complex system design, where agency is instead distributed by components

and disciplines.

Multiagent Design Systems

A variety of multiagent approaches to design have been proposed and applied to perform a variety of design roles. Some multiagent design systems have been proposed to manage the design process, including information flow and design conflicts to enable collaboration or support design activities [45]. Multiagent design systems have also been applied as computational design synthesis, with agents given different operations to perform on the design, such instantiation, modification, and management [11] [14]. Finally, multiagent systems have been used to model the engineering design process [35] [36] [63] and support engineering design [37] [34]. While these systems typically use rule-based agent architectures and not multiagent learning, learning has been proposed as a possible way for agents to discover how their internal conditions affect external conditions [22].

2.3 Other Used Approaches

This section discusses decision-based and value-centric design, as these approaches are relevant to how the multiagent model works to allow comparison between designs: by developing a single profit-based metric.

2.3.1 Decision-based Design

As discussed in Chapter 1, as systems become more complex, engineering decision-making becomes more difficult. To resolve these difficulties decision-based design seeks to incorporate existing work in decision theory within engineering design to inform design decision-making [28]. These frameworks rely on the axiomatic definition of utility presented by Von Neumann and Morgenstern in [80]–seeking to maximize the statistical expectation of the utility of a design [76] [28] [29]. It should be noted that while this utility often translates directly into a profit value (as in [27]), it is really a function of a profit value, since profit may have differing marginal utility depending on the amount [80].

The way decision-based design approaches risk can be illustrated by considering design as a simple lottery. If a design x could lead to only two possible outcomes, 1 and 2, with utilities $u_1(x)$ and $u_2(x)$, and outcome 1 has probability $p(x)$, the expected utility $U(x)$ of that design is:

$$U(x) = p(x) * u_1(x) + (1 - p(x)) * u_2(x) \tag{2.4}$$

Considering this, a design where $p(x) = 0.5$, $u_1 = 200$, and $u_2 = 0$ is equivalent to a design where $p(x) = 1.0$, $u_1 = 100$, and $u_2 = 0$, since both have the same expected utility of $U = 100$. Alternatively, a design where $p(x) = 1.0$, $u_1 = 100$, and $u_2 = 0$ would be preferred to a design where $p(x) = 0.9$, $u_1 = 100$, and $u_2 = 10$, since it has a higher total utility ($100 > 91$).

Because decision-based design is ultimately a quantitative framework, it lends itself to optimization [27]. For industry applications, this is generally approached by balancing a cost model against a consumer-choice-based demand model to determine how profitable a design will be, and, consequentially, how much utility it provides [81]. Within the systems engineering community, it has developed into the value-driven design framework, which seeks to quantify the cost and value of attributes in order to truly optimize a design, rather than impose requirements [15].

The model presented in this thesis uses this value-driven approach (ignoring uncertainty) to combine objectives into a cost value to provide a single metric of how good the design is to judge the performance of designer behaviors.

2.4 Previous Work

Previous work by the authors using multiagent systems to design complex engineered systems involved applying a coevolutionary algorithm to racecar design [87] [71] and applying multiagent learning to the design of a self-replicating robotic manufacturing factory [48]. For the racecar application, agents designed sets of component solutions, which were combined into designs. The best performing racecar was selected, and the contribution of each agent was judged by comparing those components with a counterfactual using a difference reward. The team of agents was shown to generate designs which perform better with respect to a set of objectives as real formula SAE designs. For the self-replicating factory appli-

cation, each agent was a robot, with roles defining which agents could perform which operations, such as mining regolith or installing solar cells. It was shown that using Q-learning to find the best policies for the agents created a factory which remained productive over the long term, unlike preprogrammed behavior, which was unable to sustain productivity. While these approaches demonstrated the general applicability of multiagent systems to engineering design, they do not address the problems approached in this paper with respect to problem representation. In the self-replicating factory application, the problem was more focused on designing the policies of the robots that made up the factory than the factory itself. This problem was more comparable to problems commonly encountered in multiagent systems than engineering design, since the design solution was a control policy, rather than a set of optimal design parameters. In the racecar application, the problem at hand was very much an engineering design problem, but the brunt of the optimization problem was not handled by the agents themselves, but the cooperative coevolution coordination algorithm. That is, agents in this approach merely represented individual design solutions for each component which were then externally optimized, not the designers of each component which work together to produce an optimal design. The research presented in this paper, on the other hand, is interested in using the agents themselves as the optimization mechanism, rather than an external algorithm, since that is more analogous to complex engineered systems design teams, where the "agents," or designers, design or optimize based on their own actions, rather than an external algorithm acting on them.

Finally, work by the authors has been shown already towards developing the

method presented in this paper using a distributed agent-based optimization method to optimize a quadrotor. This work used an agent-based approach as an optimization method, resulting in the learning assignment used in this work, but without the theoretical justification shown in Section 4.1. Additionally, it used an external entityannealingto control exploration and exploitation [32], which is tested against other methods of control in Section 5.2. This paper further expands on that paper by extending the case study to a more comprehensive formulation which takes into account more components and operational characteristics, extending the multiagent-based optimization method to act on mixed integer-continuous domains and more explicitly take constraints into account, and using the method as a model to study design teams.

3 Methods

The multiagent model of design used in this paper is the combination of two parts: a design problem and a multiagent system. The design problem is presented in Section 3.1 as quadrotor design problem with a number of interconnected component variables (delegated to agents in our model) and objectives which are combined using a value-driven design approach. The multiagent system is described in Section 3.2, with a justification for the learning heuristic used.

3.1 Quadrotor Design Model

The method presented in this paper is used to model design teams in conjunction with a quadrotor design problem. This problem was chosen because it embodies the core attributes of a complex engineered systems design problem: constraints which represent the interactions between subsystems and the various requirements which must be met by those interactions. Additionally, as a constraint integer-continuous problem, it provides a significant challenge for our method, as some variables must be acted on differentlywhile some variables may accept small changes which are not very coupled with others (such as the twist or taper of the propeller), others are highly coupled with each other, and require an optimal configuration (such as the number of batteries to use in series or parallel). It should be stated that

problems with inter-disciplinary constraints create considerable challenge towards a decomposed learning-based optimization (as is discussed in Section 3.2.1), since violations of those constraints (for example, caused by an agent choosing a random variable value) necessarily yield extremely bad design outcomes and objective function values. Nevertheless, they are required to properly represent the type of multidisciplinary design that this paper studies, as designers in multidisciplinary design teams must necessarily ensure that their subsystem designs are compatible.

The basic synopsis of the design problem is as follows (a full description is shown in Appendix 3.1.1). The quadrotor is commissioned to perform ten missions in which it must climb to a height, fly towards, and hover around a number of points of interest, and then descend to its point of origin, visiting the maximum number of points-of-interest possible with its available energy. The objective of the design is to maximize profit based on the revenue of these missions (generated by viewing points of interest) and the cost of the quadrotor. This single-objective approach to an otherwise multiobjective problem (e.g. with mass, cost, and other performances as objectives) is inspired by decision-based design, in which a multi-attribute design problem is transformed into a single-attribute design problem of maximizing profit by using a demand model [13] instead of weighting the various objectives, the importance of each objective is modelled based on the value generated. The variables and constraints of the problem are shown in Table 3.1 and 3.2, respectively. Since the mission has a number of operational characteristics (hovering, climbing, and steady flight), these constraints are considered for each characteristic. Note that for our purposes the model is considered a black box, with all equality con-

straints (as well as an external model) considered a part of the model physics rather than a constraint given to the optimizer. This multidiscipline-feasible optimization framework was used because the coupling variables of each subsystem have a higher dimensionality than the subsystem design variables (making it a preferred framework [4]), and because this research is studying the coordination of designersnot the model structure. Additionally, this structure makes the one-variable-per-agent multiagent system a more realistic model of design, since the impact of each design variable is large and in several cases (such as the motor or battery cell) directly sets several other parameters of the design. The full description of these models is shown in Appendix 3.1.1, however, for the purposes of this paper it merely provides a multidisciplinary mixed-integer problem domain that encompasses challenges that multidisciplinary teams encounter: inter-disciplinary requirements that cause the optimal selection of a variable of one discipline to be coupled with the variable values in other disciplines.

3.1.1 Quadrotor Design Model Details

This section covers the optimization of a quadrotor's structural and propulsive platform with respect to flight performance and cost. The main components considered were the battery, motors, propellers, and support rods. These components were chosen because they are the primary drivers of the potential flight performance of the craft and the primary intersecting structural consideration.

To simplify quadrotor design to the conceptual level, many interrelated design

problems were not considered. The effect of several components (the controller, ESCs, housing, wiring, landing skid, etc.) were simply estimated under the assumption that different choices would have similar impact, or that those parts could be designed in conjunction with the rest of the craft without a large impact on the performance of the platform with respect to the objectives considered. Notably, the most often studied problem in quadrotor design has to do with the control system of the craft. While an optimal control system may have a large impact on the overall performance above or below the estimated performance, it is still treated as a separate problem from our design application.

Motor Design

For the motor, only one choice is made–a choice of several motors from a catalog (taken from the T-Motor Professional Series). This choice, A_m, gives the defining parameters of the motor model (k_v, R_0, I_0) as well as the performance constraints ($P_{m_{max}}$ and $I_{m_{max}}$), mass, area, and cost of the motor (M_m, A_m, and C_m).

$$A_m \rightarrow k_v, R_0, I_0, P_{m_{max}}, I_{m_{max}}, M_m, A_m, C_m \tag{3.1}$$

These parameters are then entered into the performance and constraint calculations in Section 3.1.1.

Battery Design

For the battery, we assume several choices are made: the cells to be used and the number of cells to be used in parallel and series. From these choices, the total performance characteristics of the battery can be constructed. The cell choice A_c determines the capacity mAh, discharge rate C, mass M_c, and cost C_c of each cell used in the battery.

$$A_c \rightarrow mAh, C, M_c, C_c \tag{3.2}$$

Since each cell is designed to give (nominally) 3.7 volts of voltage, the total voltage V of the battery can be calculated from the number of cells in series N_s. The total mass and cost of the battery M_b and C_b can be calculated from the mass M_c and C_c of each cell multiplied by the number of cells $N_s * N_p$. The current available I_{bmax} can be calculated from the discharge rate of each cell mAh, the C rating of each cell C, and the number of cells used in parallel N_s. Finally, the energy E available may be calculated from the voltage V available, the capacity of each cell mAh, and the number of cells in parallel N_p. Each of these performance characteristics

are then entered into the performance and constraint calculations in Section 3.1.1.

$$[h]V = 3.7v * N_s \tag{3.3}$$

$$M_b = M_c * N_s * N_p \tag{3.4}$$

$$C_b = C_c * N_s * N_p \tag{3.5}$$

$$I_{bmax} = C * 1000 * mAh * N_p \tag{3.6}$$

$$E = V * 1000 * 3600 * mAh * N_p \tag{3.7}$$

Propeller Design

For the propeller, the choice of airfoil A_a, diameter D_p, blade angles α_r and α_t and chords C_r and C_t are used to construct the properties of the propeller. The choice of airfoil A_a determines the lift and drag characteristics of the blade at each angle, as shown in equation 3.8. The coefficients for these relationships (obtained from xfoil), are then used in QProp. It also specifies the airfoil's ratio of thickness to chord tc. Additionally, the radius vector $\vec{R}$ and the chord $\vec{c}$ and angle vectors $\vec{\alpha}$ needed by qprop are constructed from the diameter of the propeller D_p and the dimensions of each at the root (C_r and α_r) and tip of the propeller blade (C_t and

α_t), assuming they change linearly across the blade.

$$A_a \rightarrow Cl_0, Cl_a, Cl_{min}, Cl_{max}, Cd_0, Cd_2, Clcd_0, tc \tag{3.8}$$

$$\vec{R} = [0.02, ..., D_p/2] \tag{3.9}$$

$$\vec{\alpha} = \alpha_r + \vec{R} * (\alpha_t - \alpha_r)/(.5D_p) \tag{3.10}$$

$$\vec{c} = c_r + \vec{R} * (c_t - c_r)/(.5D_p) \tag{3.11}$$

These vectors are entered into Qprop at each iteration to calculate the flight characteristics of the propulsion system.

The mass M_p and cost C_p may also be calculated by estimating the volume V_p of the propeller. To find this, the average chord c_{avg} and thickness t_{avg} of the propeller must be calculated from the root and tip dimensions, as well as the thickness ot chord ratio tc. Then the cross-sectional area A_{xs} may be estimated (assuming the airfoil is a roughly triangular section), as well as the volume V_p, cost C_p, and mass M_p:

$$c_{avg} = (c_r - c_t)/2 \tag{3.12}$$

$$t_{avg} = tc * c_{avg} \tag{3.13}$$

$$A_{xs} = 0.5 * t_{avg} * c_{avg} \tag{3.14}$$

$$V_p = A_{xs} * D_p \tag{3.15}$$

$$C_p = cd * V_p \tag{3.16}$$

$$M_p = \rho * V_p \tag{3.17}$$

where cd is the cost density of the propeller assumed to be $17700\$/m^3$ (or $.29\$/in^3$), and ρ is the density of the propeller assumed to be $1190kg/m^3$. This is based on the assumption that the propeller is made out of polycarbonate.

Support Rod Design

For the support rod, the design is defined by the choice of material, wall thickness, and diameter. Similar to the rest of the "black box" model used in the formulation (where equality constraints between components are considered a part of the model, rather than constraints which must be enforced) the length L_r is determined to the value required to keep the propellers a safe distance from each other. It can then be calculated using the propeller diameter D_p and frame width W_{res}.

$$L_r = max(\frac{1.25 * D_p}{\sqrt{2}} - \frac{W_{res}}{2}, 0.01) \tag{3.18}$$

The material choice A_{mat} (Aluminum, Titanium, Polycarbonate, or Nylon) determines the Young's modulus E_y, density ρ, ultimate strength S_{ut}, and cost density cd of the material used in the rod. Note that while carbon fiber is a commonly used material in quadrotor design, it was was not considered in our application because the strength of a carbon fiber rod is not necessarily dictated by the wall thickness as it is with other materials. Then the cross-sectional area A_r, area moment of inertia I, and bending stiffness k may be calculated. Each of these quantities help define the structural performance of the rod, and are entered into the constraint

calculation. Additionally, the cost C_r and mass M_r may be calculated from the length L, cross-sectional area A_r, and mass/cost density (ρ or cd) of the material.

$$A_{mat} \rightarrow E_y, \rho, S_{ut}, cd \tag{3.19}$$

$$A_r = \pi * 0.5 * (d^2 - (d-t)^2) \tag{3.20}$$

$$I = \pi * (d^4 - (d-t)^4)/64 \tag{3.21}$$

$$k = (3 * I * E_y)/L^3 \tag{3.22}$$

$$M_r = A_r * L * \rho \tag{3.23}$$

$$C_r = A_r * L * cd \tag{3.24}$$

Residual Quantities

To capture the impact of the components not being designed, a series of residuals have been added to simulate the impact of the rest of the system, including those listed in Table 3.3.

System Modelling

These component characteristics may then be used to find the overall system characteristics. The system mass M_{sys}, planform area A_{sys}, and cost C_{sys} are all the

sums of the components and residuals mentioned in the previous sections.

$$M_{sys} = 4 * (M_m + M_p + M_r) + M_b + M_{res} \tag{3.25}$$

$$A_{sys} = 4 * (A_m + A_r) + A_{res} \tag{3.26}$$

$$C_{sys} = 4 * (C_m + C_p + C_r) + C_b + C_{res} \tag{3.27}$$

Qprop is used to model the performance of the propulsion system. For the purposes of this paper, it will be used to find the performance of the propulsion system at different velocities vel and thrust requirements T_{req}. Using these parameters, as well as the characteristics of the motor and propeller, QProp finds the power use P, actual thrust T, current I, and rotational speed RPM.

$$\begin{matrix} T_{req}, vel, \\ k_v, R_0, I_0, \\ Cl_0, Cl_a, Cl_{min}, Cl_{max}, \\ Cd_0, Cd_2, Clcd_0, \\ \vec{R}, \vec{\alpha}, \vec{c} \end{matrix} \quad \rightarrow P, T, I, RPM \tag{3.28}$$

Hovering Characteristics

The hovering characteristic is modelled in order to determine the value of the "maximize hover time" objective. This characteristic is defined by a flight velocity of $vel_h = 0$ and a thrust requirement $T_{req,h}$ defined by the fraction of the total

mass of the system the propeller must hold up.

$$T_{req,h} = M_{sys} * 9.81/4 \tag{3.29}$$

Note that if the thrust requirement cannot be met, this method reports a failure which corresponds to a specific low reward value less than or equal to Q_{init}, the lowest possible learned value.

Climbing Characteristics

The climbing characteristic is modelled in order to determine the value of the "minimize power use in a climb" objective. This characteristic is defined by $vel_c = 10m/s$. Since the quadrotor is a blunt object, it generates drag D_c estimated by the equation:

$$D_c = \rho * C_d * A_{sys} * vel_c^2 \tag{3.30}$$

where ρ is the density of air ($1.225kg/m^3$) and $C_d = 1.5$ is the assumed coefficient of drag. The thrust requirement may then be calculated as the combination of the mass M_{sys} and drag D_c held by each individual motor:

$$T_{req,h} = (D_c + M_{sys} * 9.81)/4 \tag{3.31}$$

Structural Characteristics

Using the data calculated previously, the performance of the support rod may be modelled. First, the RPM value may be used to calculate the frequency f_{motor} of the motor. Then the natural frequency f_n of the rod-mass system (assuming the rod acts as a cantilever beam with stiffness k and masses $0.5 * M_r + M_m$) may be calculated, as well as the deflection of the rod δ when thrust T_{oper} is applied on the beam with stiffness k.

$$f_{motor} = RPM_h/60 \tag{3.32}$$

$$f_n = \frac{1}{2 * \pi} * \sqrt{\frac{k}{0.5 * M_r + M_m}} \tag{3.33}$$

$$\delta = T_{oper}/k \tag{3.34}$$

3.2 Multiagent Learning for Multidisciplinary Design

This section presents a multiagent learning-based optimization method which this paper uses to study decomposed, multidisciplinary design. The core representation of learning from the environment when a problem is decomposed between individual designers is presented in Section 3.2.1, along with a theoretical justification of the custom learning heuristic presented in this paper. Then the stochastic optimization method based on this representation is presented in 3.2.2, along with a summary of important parameters in 3.2.3.

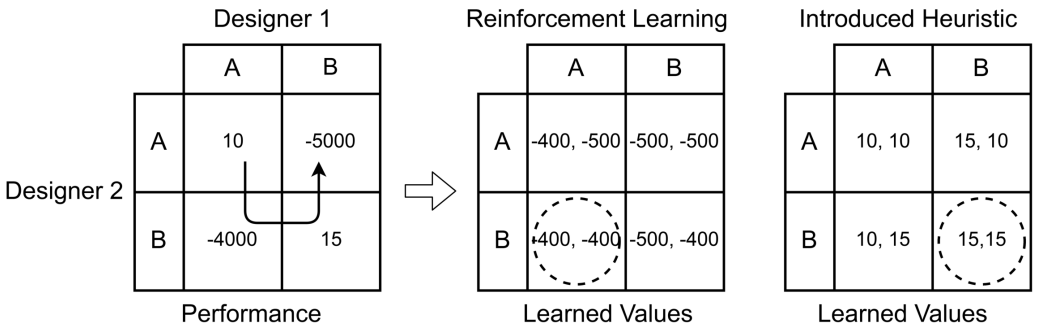

Figure 3.1: Learned values of reinforcement learning and this paper's introduced heuristic on a simple example problem meant to capture the coupling between different designers choices. Each agent has two designs which may be picked–A and B. Traversing the full space of designs, reinforcement learning designers do not encode the optimal design, while designers using the introduced heuristic do

3.2.1 Learning Design Merit in Distributed Design

This method presented in this work is based on a custom learning heuristic specifically designed to represent optimization on a problem which has been decomposed across disciplinary lines to different designers. While multiagent learning systems have been shown to optimize the time-based actions of agents on distributed problem domains, their application to a general design optimization context is hindered because the merit of actions in a design context is highly coupled with the other actions taken. In distributed design, interdisciplinary constraints mean the design of one subsystem and another subsystem have a significant impact on the overall performance of the system. As such, it can be thought of as a partially observable Markov decision process for both agents (described in [54] and [72]), since neither agent has full state information about the model, and designing without full knowledge of how its variables will interact with others.

To illustrate how this property of distributed design hinders using multiagent learning, consider the simple design problem shown in Figure 3.1, using the notation of game theory to describe the problem. In this problem, each designer must pick a design (A or B) for their subsystem, which, when put together with the other designers subsystem, has the best performance. While design A-A and B-B are compatible, A-B and B-A are not, such that they fundamentally have no merit as designs. However, while Design B-B has better performance than design A-A, design A-B is less infeasible than design B-A. If the designers traverse the entire space of designs A-A, A-B, B-B, and B-A in four time-steps, the learned value of each design for each designer using the standard reinforcement learning heuristic $V(a) \rightarrow V(a) + \alpha(r - V(a))$ with $\alpha = 0.1$ is shown in the middle of Figure 3.1. Based on their learned values, the designers would conclude that design A-B was the best, despite being an infeasible (and third-worst) design. This is because this learning heuristic captures the "average" value of an action, which is overly skewed by infeasible reward values.

In the learning heuristic presented in this paper, this problem is solved by having designers simply learn the best value for their action entered so far. This approach can be thought of as a simplification and adaptation of leniency in multiagent learning, which is discussed in [57] [58] [7]. With leniency, agents ignore poor individually rewards due to poorly-matched actions with their team-mates in order to focus on higher rewards from good joint actions, and help agents proceed from poor Nash equilibria to optimal Nash equilibria [58].

Using the heuristic introduced here, the designers traverse the entire design

space using $V(a) \leftarrow max(V(a), r)$, and their values for each variable are shown on the right side of Figure 3.1 1. As can be seen, while the learned values are inaccurate for the infeasible designs, they are accurate for the feasible designs. Most importantly, the heuristic allows the designers to identify the optimal design, as it has the highest learned values for both designers. It should also be noted that this heuristic does not change values (as would reinforcement learning) depending on how the designers explore the space (e.g. designer 2 choosing design A more than B giving it a lower value over time), but gives a constant value based on the actual performance given so far by the problem. This shows how this introduced heuristic better represents the collaborative design process compared to reinforcement learning.

Derivation from Q-learning

The multiagent learning-based representation used in this paper can also be thought of as an adaptation of Q-learning in a deterministic setting. Q-learning is a common learning method which has been shown to converge to the optimal policy in both a deterministic environment and certain stochastic environments, making it a natural choice of learning heuristic for use in optimization [82]. When applied to distributed optimization, an agent must choose a value (encoded as actions a_t) in its current state s_t for each variable to find the objective (encoded as a reward r_f), which is calculated at a final states s_f. The standard Q-learning assignment

is:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha * (r_t + \gamma * max(Q(s_{t+1}, a_{t+1})) - Q(s_t, a_t)) \tag{3.35}$$

where $Q(s_t, a_t)$ is the value given to the action in the current state, α is a learning rate, r_t is the reward at the current state, γ is the preference for future rewards, and $max(Q(s_{t+1}, a))$ is the best value of the next state reached by taking the current action in the current state. This assignment is used until the the agent reaches a final state s_f, where the Q-value is simply the reward in that state r_f. For pure optimization in a deterministic context, this α parameter may be set to one, since the same set of actions given in the same set of states will always yield the same result. Simplifying the expression makes the learning assignment:

$$Q(s_t, a_t) \leftarrow r_t + \gamma * max(Q(s_{t+1}, a_{t+1})) \tag{3.36}$$

Given that the only rewards in an optimization context are given after all of the variable values are chosen, the reward r_t is set to zero, except at the final state, where it is the value of the objective. Additionally, since there is no "current reward" the preference for future rewards γ may be set to one. This means, for each variable (including the final variable), the learning technique to be used is:

$$Q(s_t, a_t) \leftarrow max(Q(s_{t+1}, a_{t+1})) \tag{3.37}$$

Finally, the agent may end up in any number of final states $s_f(a_{t_I}, \vec{a_{t\neq i}})$ based on the action a_{t_i} of the agent i in combination with the actions of all of the other agents $\vec{a_{t_{\neq i}}}$. In this final state, the objective value may finally be calculated, constituting the reward r_f. Thus the Q-values of the final states are:

$$Q(s_f(a_{t_i}, \vec{a_{t_{\neq i}}})) \leftarrow r_f(a_{t_i}, \vec{a_{t_{\neq i}}}) \tag{3.38}$$

which gives a unique final state for every possible design to each agent. This makes learning in the action-taking state:

$$Q(s_t, a_t) \leftarrow max(Q(s_f(\vec{a_t}))) \tag{3.39}$$

Since the Q-values of the final states only change when the states have been entered for the first time, the Q-value in the action-taking state only increases when a new state has been reached with a higher value than the state with the highest recorded value. As a result, this learning technique may be implemented without keeping track of states and final Q-values by simply assigning the value of the action taken by the agent as the maximum of its current value and the reward returned when all of the actions of all of the agents are taken, as shown below:

$$V(a_i) \leftarrow max(V(a_i), r_f(a_i, \vec{a_{\neq i}})) \tag{3.40}$$

where $V(a_i)$ is the value of the action for the agent and $r(a_i, \vec{a_{\neq i}})$ is the reward generated by that action when combined with the actions of all of the other agents.

As shown here, the decentralized learning heuristic used in Section 3.2.2 and introduced in previous work [32] is equivalent to a special case of the multiagent-learning approach adapted to standard design optimization. This provides a theoretical basis for its effectiveness, in addition to the experimental basis shown in Section 4.1.

3.2.2 Multiagent Learning Based Design Optimization Method

Designers in this framework are best thought of as meta-agents controlling the exploration of sub-agents which in turn pick which variable values to submit to the model at each iteration of the algorithm, as shown in Figure 3.2.

Each computational designer is delegated the design of one variable value in the model and learns two things based on feedback with the environment: which variable values are good (through the sub-agent) and whether to explore new variable values or exploit the current best values (through the meta-agent). The sub-agents use the heuristic introduced in Section 3.2.1, while the meta-agents use stateless reinforcement learning to control the degree of exploration or exploitation of those sub-agents in picking variable values, since the required amount of exploration is likely to change throughout the design process. The general steps of the algorithm shown below are illustrated in Figure 3.3.

1. **Meta-Agent Action Selection:** Using an epsilon-greedy action selection process, the meta-agents choose the temperature (level of exploration) to use in picking the value of the design variable.

2. **Sub-Agent Design Parameter Selection:** Based on this temperature

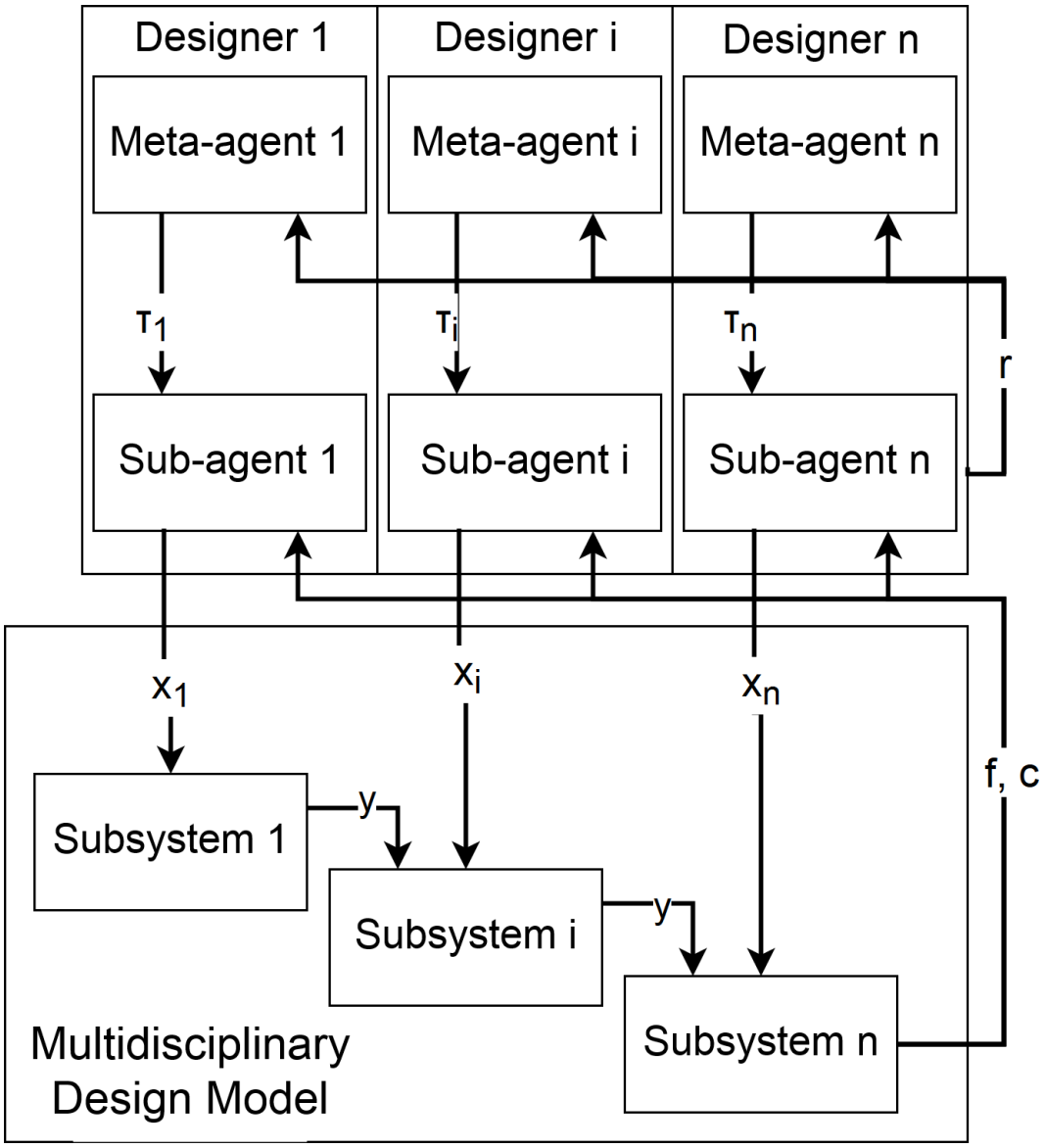

Figure 3.2: High-level structure of the multiagent optimization method and design model. Designers consist of meta-agents control sub-agents which pick design parameter and receive rewards based on design performance.

value and a table listing the merit of each parameter value, the sub-agents choose the parameter values of the design.

3. **Sub-Agent Merit Update:** The model gives the performance of the resulting design in terms of the objective and overall constraint violation. Per the learning technique introduced in Section 3.2.1, if this performance is better than the performance previously in the table of any agent, the value in that table is updated.

4. **Meta-Agent Rewards and Learning:** The increase in value in each of table is given to the Meta-agent as a reward. Depending on the reward structure, these rewards are added together (global reward) or given individually (local reward) to each agent. The meta-agents learn this reward for their chosen temperature using reinforcement learning.

Steps 1-4 are repeated until a stopping condition is reached. Each step is described in depth in the following sections. Note that the notation for actions and rewards differs from the general notation, with actions and rewards for the sub-agent i stated as x_i, and a combination of the objective and constraint values f and c, respectively, while actions and rewards generated by meta-agent i is denoted as τ_i and $r_(f_i)$ and $r_(c_i)$, respectively.

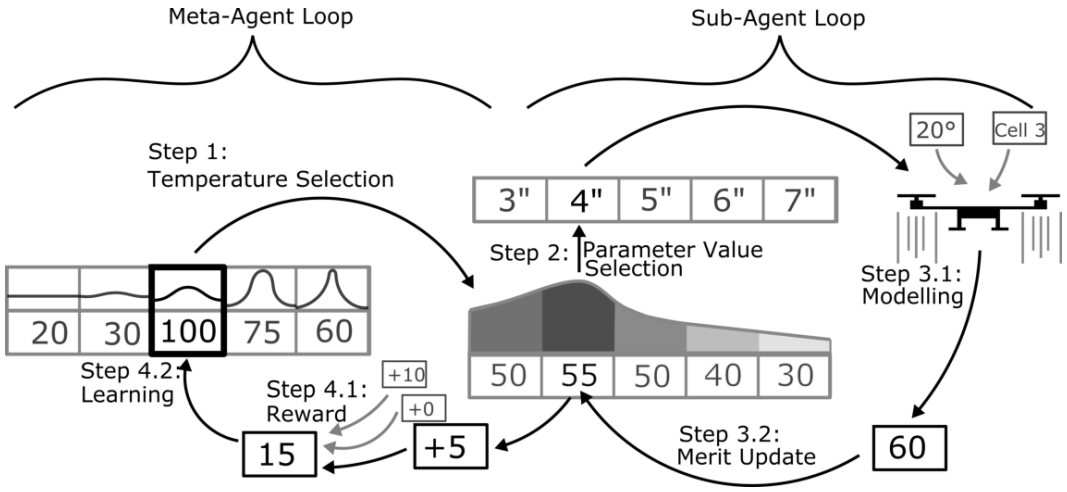

Figure 3.3: Overview of method described in Section 3.2.2 shown for a single agent in the multiagent system.

Meta-Agent Action Selection

Meta-agents are mechanisms which control the exploration and exploitation level in order to act based on current knowledge or seek new knowledge of the merit of the design variable. They are used to coordinate the sub-agents to find the optimal design parameters quicker by causing sub-agents to explore untried combinations of variable values. This is done by having each meta-agent i choose the temperature τ_i based on its value table for a given set of temperatures. The action selection process for meta-agents is epsilon-greedy, as described in Section 2.2.1, choosing the best temperature with a certain probability and a random temperature with a certain probability.

Sub-Agent Design Parameter Selection

Sub-agents realize the level of exploration or exploitation by using the selected temperature to pick the variable value based on the performance of past designs. This performance is encoded in a table which stores the best previous value of each parameter value in terms of objectives $f_s(x_i)$ and constraint violation $c_s(x_i)$. By storing the best previous value of each parameter value, the sub-agents collectively store the best design point found so far, as well as an estimate of the relative future promise of each design parameter, which allows them to explore without losing any previous information about the design merit of a variable value. To generate a single metric of merit for the action selection process, these quantities are combined using:

$$M(x_i) = f_s(x_i) - \sigma * c_s(x_i) \tag{3.41}$$

where $M(x_i)$ is the combined metric of merit, x_i is the variable value , $f_s(x_i)$ is the stored objective value, $c_s(x_i)$ is the stored constraint violation value, and σ is a scaling factor which is similar to the idea of a penalty used in common constrained optimization methods [20].

This merit is then normalized using a softmax normalization, putting the variable values with the worst possible merit at 0 and the variable values with the best possible merit at 1. This normalization is used to reduce the influence of outlying stored values on the action selection process, and allows the action selection process to work regardless of the scale it acts on. This normalization follows the

equation:

$$M(x_i)_n = \frac{1}{1 + e^{-\frac{M(x_i)-\mu_x}{\sigma_x}}} \tag{3.42}$$

where $M(x_i)_n$ is the normalized merit, $M(x_i)$ is the non-normalized merit, and μ_x and σ_x are the mean and standard deviation of the merit of the n variable values $[x_1...x_i...x_n]$ available to the agent, respectively.

The variable value is then chosen based on the softmax action selection process outlined in Section 2.2.1, with the probability of choosing a variable value determined by the equation:

$$p(x_i) = \frac{e^{(M(x_i)_n/\tau)}}{\sum_{i=1}^{n} e^{(M(x_i)_n/\tau)}} \tag{3.43}$$

where $p(x_i)$ is the probability of choosing a variable value x_i, $M(x_i)_n$ is the normalized merit of the variable value x_i, $M(x_i)_n$ is the normalized merit of variable value i, and τ is the temperature. In this framework, this temperature is chosen by the meta-agent as outlined in Section 3.2.2.

Sub-Agent Merit Update: Design Performance

Sub-agents learn the merit of each variable value through interaction with the model using an adaptation of the learning heuristic introduced in Section 3.2.1. After each of the variable values has been chosen by the sub-agents, the design is modelled with each of those variable values, generating an objective function value f and constraint violation value c. This design information is captured by updating

the stored value of each variable value $f_s(x_i)$ or $c_s(x_i)$ if the found objective and/or constraint violation value is better than the currently stored value, per the learning heuristic introduced in Section 3.2.1, but with further adaptations for determining how values compare with each other given both objective and constraint values. In addition, the reward for the meta-agents is calculated based on this learning process as the difference between the old value and the updated value.

The control logic for this is shown in Algorithm 1. If the found constraint value is better than the stored constraint value, the objective and constraint value is updated, and a reward is calculated for the meta-agent based on the decrease of the constraint value and the objective value, if the objective value decreased. If the found constraint value is the same as the stored constraint value, but the objective function is better than the stored objective value, the stored objective value is updated and a reward is calculated. Otherwise, the current stored merit is kept and a reward of zero is returned for that agent's variable. The result of this adaptation is that feasibility is always considered the primary consideration in saving a design value, followed by performanceno high performance (but ultimately meaningless) design is considered "better" than a feasible design for the purposes of learning.

Adaptation to Continuous Variables

This learning process can be further adapted to continuous variables by separating continuous space into zones $[z_1...z_i...z_n]$ represented by the best possible values

Algorithm 1 Control Flow for Merit Update and Rewards

1: **if** $c < c_s(x_i)$ **then**
2: $c_s(x_i) \leftarrow c$
3: $f_s(x_i) \leftarrow f$.
4: $r_{ci} = c - c_s(x_i)$
5: $r_{fi} = max(f - f_s(x_i), 0)$.
6: **else if** $c = c_s(x_i)$ and $f < f_s(x_i)$ **then**
7: $c_s(x_i) \leftarrow c_s(x_i)$
8: $f_s(x_i) \leftarrow f$.
9: $r_{ci} = 0$
10: $r_{fi} = max(f - f_s(x_i), 0)$.
11: **else**
12: $c_s(x_i) \leftarrow c_s(x_i)$
13: $f_s(x_i) \leftarrow f_s(x_i)$.
14: $r_{ci} = 0$
15: $r_{fi} = 0$.

$[f_{zr1}...f_{zri}...f_{zrn}]$ at points $[x_{zr1}...x_{zri}...x_{zrn}]$ inside the respective zones. A piecewise cubic hermite polynomial interpolation is then made between each point, creating objective and constraint functions f_s and c_s for the variable at each value of the variable analogous to the value table used for the discrete variables. The vectors returned by this interpolation $x = [x_1...x_n]$, $c_s = [c_{x1}...c_{xn}]$ and $f_s = [f_{x1}...f_{xn}]$ are then used in in the same way they are used with discrete merit tables to pick a variable value. Learning for continuous variables then follows the same process as with discrete variables, except instead of having a better value than the stored value of the current variable value, the found value must be better than the point which represents of the zone. This is, f_{zri} and c_{zri} are used in Algorithm 1 instead of f_s and c_s. This process is illustrated in Figure 3.4.

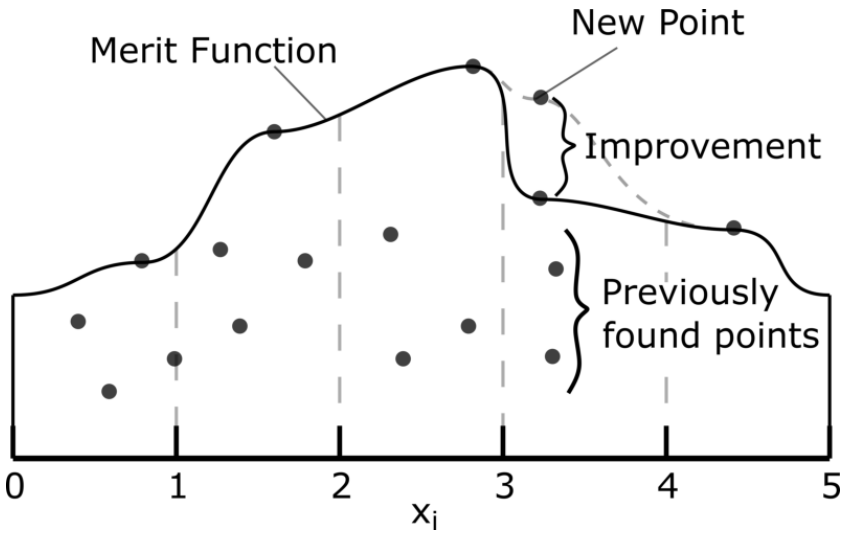

Figure 3.4: Visualization of the continuous merit update process described in Section 3.2.2. A spline is fit through the best points found in each zone of the continuous space.

Meta-Agent Rewards and Learning: Improvement Over Expected Values

The rewards for the meta-agents are calculated as the increase in knowledge gained by exploring or exploiting that variable. This reward is calculated from the increases in objective and constraint value r_{f_i} and r_{c_i} for each agent/variable i which are calculated as shown in Algorithm 1. The reward r_i generated by each agent is then:

$$r_i = r_{f_i} + \sigma * r_{c_i} \tag{3.44}$$

where r_{f_i} and r_{c_i} is the reward for constraints and objectives calculated for each agent i in Section 3.2.2, and σ is a scaling factor which takes the place of a penalty

factor.

Two reward structures can be easily constructed based on these rewards: a local reward structure in which meta-agents are rewarded solely on the information gained by their sub-agents, and a global reward in which the meta-agents are rewarded by the sum of the rewards of the sub-agents. In this case it is expected that the global reward should be used, since there are no barriers to calculation and because it better aligns the agents actions with the desired purpose of the overall behavior, and achieves better results on common multiagent systems domains [1], however both will be tested in Section 4. The local reward L_i and global reward G_i are calculated for each meta-agent i as:

$$L_i = r_i \tag{3.45}$$

$$G_i = \sum_{i=1}^{m} r_i \tag{3.46}$$

where r_i is the reward based on new knowledge generated by each corresponding sub-agent i.

3.2.3 Implementation

When applying this framework, a few adjustment parameters should be considered. These parameters, as well as a short explanation and the values used in the following tests are shown in Table 3.4. Stopping conditions must also be considered, which may be (like other optimization methods) based on the number of function

evaluations without improvement, the total number of evaluations, or reaching a desired objective function value. This optimization method would also allow for unique stopping conditions based on learning, such as the number of evaluations without learning or the decrease in the magnitude of rewards over time.

When applying the method on a given problem, the method must also be given a few things in order to initialize and define the agents. For each integer variable, the method must be given the number of available values the variable can take. For each continuous variable, the method must be given the upper and lower bounds of the variable, the number of zones to split that variable into, and the minimum tolerance for the variable at which there is no discernible difference. Additionally, the method must be adapted to the constrained problem through a single metric of feasibility, which in these tests was chosen to be:

$$c = \sum_{j=1}^{m} c_j^2 \tag{3.47}$$

where c is the metric of feasibility, m is the number of constraints, and c_j is an individual constraint. These constraints have a large impact on the problem, and in practice it was found that the penalty factor works best when it is increased over time according to:

$$\sigma = \sigma_{max} * (1 - e^{-k*e}) \tag{3.48}$$

where e is the current evaluation, k is a decay constant, and σ_{max} is the maximum value of the penalty parameter. This method of combining constraints with the objective used in this paper is similar to what is done in most penalty methods,

such as SUMT, in which the penalty is increased over time when the optimization approaches an apparent minimum [20].

Table 3.1: Design choices for each component for the quadrotor design application. Shown are the components each variable is associated with, the design choice that variable represents, the symbol used by that variable, and the parameter range (for continuous variables) or number of options available (for discrete variables) for that variable.

Component	Design Choice	Symbol	Parameter Range/Number of Options
Motor	Motor	x_m	9
Battery	Cell	x_{bc}	6
	Cells in Series	x_{bs}	7
	Cell in Parallel	x_{bp}	4
Propeller	Airfoil	x_{pai}	7
	Diameter	x_{pd}	0.02-0.2 meters
	Angle	x_{pan}	0-45 degrees
	Twist	x_{ptw}	0-1 (unitless)
	Chord	x_{pc}	0.005-0.02 meters
	Taper	x_{pta}	0-1 (unitless)
Rod	Material	x_{rm}	4
	Thickness	x_{rt}	0.0009-0.006 meters
	Width	x_{rw}	0.0065-0.0380 meters
	Height	x_{rh}	0.0065-0.0380 meters
ESC	ESC	x_e	6
Landing skid	Material	x_{sm}	4
	Leg Angle	$x_{s\theta}$	20-60 degrees
	Diameter	x_{sd}	0.0065-0.0380 meters
	Thickness	x_{pst}	0.0009-0.006 meters
Mission	Climb Velocity	x_{ovc}	0.1-30 meters/second
	Steady Flight Angle	$x_{o\theta}$	0.1-45 degrees

Table 3.2: Design constraints for quadrotor design application. Shown are the components the constraints belong to, the equation of the constraint in negative-null form, and an explanation of the constraint.

Component	Constraint	Explanation
System	$c_1 = failure = 0$	Performance calculations must not fail–model must not return an error.
	$c_2 = 1 - (T_{hov} + tol)/T_{req} \leq 0$	System must produce enough thrust (within a small tolerance) to hover.
Motor	$c_3 = I_{oper}/I_{mmax} - 1 \leq 0$	Current drawn while hovering must not exceed motor max current rating.
	$c_4 = V_{oper}/V_b - 1 \leq 0$	Voltage required by the motor must not exceed what the battery can provide.
Battery	$c_5 = I_{oper}/I_{bmax} - 1 \leq 0$	Current drawn must not exceed the battery's current rating.
	$c_6 = P_{oper}/P_{mmax} - 1 \leq 0$	Power drawn while hovering must not exceed motor max power rating.
Propeller	$c_7 = \sigma_p/\sigma_{pmax} - 1 \leq 0$	Must not exceed maximum stress
Rod	$c_8 = 1 - \frac{f_{sysnx}}{3*f_f} \leq 0$	Must have a natural frequency in the x over three times the motor's frequency.
	$c_9 = 1 - \frac{f_{sysny}}{3*f_f} \leq 0$	Must have a natural frequency in the y over three times the motor's frequency.
	$c_{10} = \delta_x - 0.01 * L_r \leq 0$	Must not deflect more than one percent of its length.
	$c_{11} = \sigma_x/\sigma_{x,max} - 1 \leq 0$	Must not exceed maximum bending stress in the x direction.
	$c_{12} = \sigma_y/\sigma_{y,max} - 1 \leq 0$	Must not exceed maximum bending stress in the y direction.
ESC	$c_{13} = 1 - V_b/V_{emin} \leq 0$	Voltage range must support provided voltage of the battery.
	$c_{14} = V_b/V_{emax} - 1 \leq 0$	Voltage range must support provided voltage of the battery.
	$c_{15} = I_{oper}/I_{emax} - 1 \leq 0$	Current must not exceed ESC maximum current.
Landing Skid	$c_{16} = F_s/F_{s,max} - 1 \leq 0$	Must not transmit a set amount of force in a minor fall.

Residual Quantity	Value
Residual mass	$M_{res} = 0.3kg$
Residual power use	$P_{res} = 5W$
Frame width	$W_{res} = 7.5cm$
Residual planform area	$A_{res} = W_{res}^2 = 56.25cm^2$
Residual cost	$C_{res} = 50$

Table 3.3: Residual quantities in the design.

Table 3.4: Method adjustment parameters, including the symbol they are referred to in the text, an explanation of the symbol, and the value taken

ϵ	Fraction of times the meta-agent chooses a random temperature	0.05
$[\tau_1, \tau_2, ...\tau_n]$	The temperatures available for the meta-agents to choose	[0.5,0.1,0.05,0.01, 0.005, 0]
n	Number of zones for each continuous parameter	5
σ_{max}	Max constraint scale factor	35000
k	Penalty decay factor	0.0005
c_{tol}	Constraint violation tolerance (after which design is considered feasible)	0.2
$f_{s_{init}}$	Initial stored objective values	10000
$c_{s_{init}}$	Initial stored constraint values	10000

4 Results and Discussion

The following tests demonstrate the effectiveness of the multiagent approach presented and then use the approach to study the complex systems design process. This is done in the following sections by:

1. comparing the multiagent method with centralized algorithms in Section 4.1, demonstrating the effectiveness and validity of the representation

2. showing the effect of meta-agents on the computational designers in Section 4.2, showing how the multidisciplinary design process is affected by designers preferences for exploration over time, and

3. showing how synchronization and independence of computational designers affects multiagent design in Section 4.3 to study collaboration in multidisciplinary design.

The results are discussed in their respective sections and summarized in Section 4.4.

4.1 Viability of the Multiagent Design Method

The following tests compare the effectiveness of the decentralized multiagent optimization method with existing centralized optimization methods to show the

effectiveness of the method for complex systems design in order to demonstrate the validity of the representation. This is important because the validity of the problem representation forms the basis for the validity of the results presented in Section 4.2 and 4.3, and is be used to make conclusions about the design process. Therefore, it is important to show that this method is a viable optimization method, and not simply an optimizing process which may, due to some flaw in process or implementation, reach an artificial minimum. To test this, this paper compares the method presented here with a common stochastic optimization method (a genetic algorithm), and a stochastic hill-climbing algorithm which we expect to perform poorly by reaching a local minimumessentially showing the behavior we would expect from a flawed stochastic optimizing process.

The comparison methods are a centralized genetic algorithm and a stochastic hill-climbing algorithm. This centralized genetic algorithm was set up using default parameters for MATLAB's ga function, with 200 generations of population 100 to show how a global optimization method searches the space. The stochastic hill-climbing algorithm was set up using MATLAB's simulannealbnd function by choosing a low temperature (T=5) to simulate stochastic hill-climbing to simulate how a flawed stochastic optimization process would search the space, as it is known that stochastic hill-climbing will converge to a local minimum. Custom annealing functions for the mixed continuous-integer problem to show how an optimizer that gets stuck in local optima searches the space. These comparisons give a picture of whether the algorithm developed in this paper is capable of global optimization (causing performance comparable to the centralized genetic algorithm) or is prone

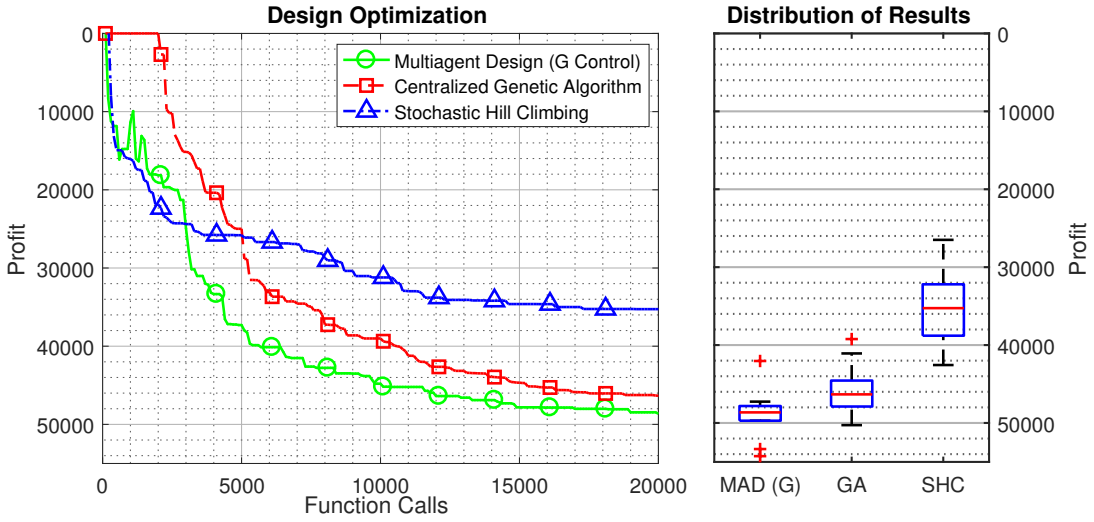

Figure 4.1: Performance of multiagent design compared with a centralized genetic and stochastic hill-climbing algorithm. Multiagent design outperforms both, demonstrating the validity of the optimization method introduced in Section 3.2.2 .

to getting stuck in local minima (causing performance comparable to stochastic hill-climbing).

Figure 4.1 shows the general trend of the distributed multiagent optimization method using the global reward structure compared with centralized optimization methods like the genetic algorithm and stochastic hillclimbing over 20000 objective function evaluations by showing the median value of the optimization and distribution of results over 10 runs. Throughout the process, the multiagent method outperforms the centralized genetic algorithm, reaching better objective values in less computational time. In addition to showing the general optimization effectiveness of the method, this result also shows the ability of the method to not get caught in early local minima, as is the case with stochastic hill-climbing on

this domain. This shows that the method indeed optimizes without obvious process flaws holding it back. Additionally, the final results distribution shows that the overall variability in results of the method is similar or better than the comparison algorithms, showing that the method finds the minimum reliably. While this result shows the general effectiveness of the method, it should be noted that it is not a comprehensive comparison–just a check to confirm the validity of the method. While it shows the multiagent method to perform well on this problem, future work is needed to show how it performs on a variety of problems to judge its effectiveness in practice.

4.2 Meta-Agents for Multiagent Design

The following tests show the effect of different methods of controlling the multidisciplinary design process by comparing the reinforcement-learning meta-agents introduced in Section 3.2.2 with annealing and a random table selection. Annealing was used in previous work by the authors as a heuristic to control the exploration/exploitation parameter τ of the agents [32], so this result shows further developments in this algorithm and reflects on annealing as a model of design. Additionally, a random selection of temperatures from the same tables the agents choose from is presented to give a control for how much reinforcement learning meta-agents really improve the process. Two reward structuresthe "global reward" based on the improvement of all of the agents' value functions and the "local reward" based on the improvement of the individual agent's value functionare additionally compared

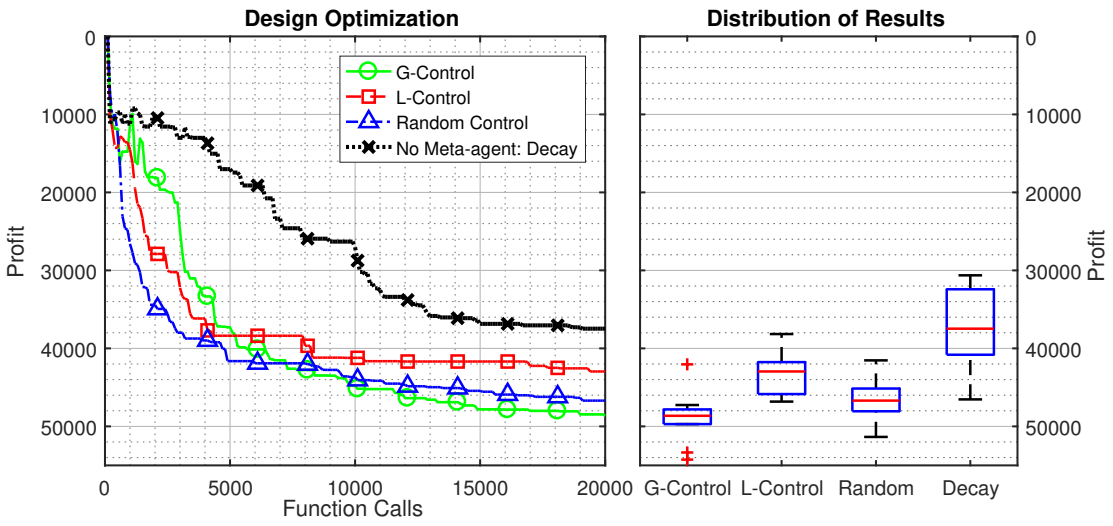

Figure 4.2: Impact of meta-agent on multiagent design. Learning improves performance over random table selection using the global reward (G) and decreases performance using the local reward (L). All strategies outperform decaying temperatures over time.

to show how rewards influence the performance of the meta-agent.

The results are shown in Figure 4.2. As can be seen, agent-based table selection (denoted by "Random Control," "G-Control," and "L-Control") vastly outperforms decaying temperatures over the entire optimization process. Additionally, a few different trends are visible for the controllers. First, controlling using the global reward, while slow to start, outperforms a random control strategy at about 75000 function calls, in the end settling in lower minima than a random strategy. The random control strategy, on the other hand, seems to find good solutions quickly (in the first 75000 iterations or so) compared to the other strategies, but loses this advantage later in the process. Finally, the local reward actually decreases the performance of the algorithm compared to random at each

step of the process, although it does outperform the global reward in the first 5000 iterations. These results provide interesting insights, both for the development of this optimization method and for distributed design in general. First, it shows how reinforcement learning can be used to increase performance of this optimization algorithm, and how, in this case, the reward structure can improve or hinder that control. Reinforcement learning, which is used to maximize rewards in a dynamic and probabilistic environment, has been shown here to increase performance of the optimization algorithm when used with a reward system that promotes exploration based on the increase in knowledge gained by that exploration. Second, it shows how annealingconverging on a single solution over time, reducing the impact of possible changes that may be mademay not be a preferred behavior for designers that are distributed across disciplinary boundaries. Previously, annealing has been used as a model for design processes, as designers will often explore the design space for changes before slowly converging around a design, reducing the impact of proposed design changes over time [10] [52]. This result shows how this behavior may lead to sub-optimal outcomes in a multidisciplinary context when trying to find an optimal set of interacting components. Instead, this result shows that it is far better for the purposes of design exploration to continue to try any and all levels of exploration at all times of the process (random selection) and even better to explore changes which increase the collective knowledge about the design space (learning based control with the G reward structure). It should be noted for this discussion that the designers not decaying their preference for exploration does not mean that should consider all designs equal throughout the design pro-

cessthat knowledge (encoded in the sub-agent) is still always used to generate a design change. Instead, this means knowledge should be sought out or leveraged at varying degrees throughout the process that should not be based on the time taken so far. If a designer exploring a variable reveals unknown parts of the design space that are better than expected, that exploration is shown here to be useful throughout the process, as is a designer keeping their variables constant to allow other designers to explore. While, in practice, it may be necessary to solidify parts of the design to perform more detailed design work, this result shows that slowly and permanently solidifying the entire design while exploring the design space can prevent the designers from finding the best-performing design. Instead, this result suggests that designers should instead be incentivized by the increase in design knowledge that their exploration causes. This is an important insight for the early and embodiment design stages, when the goal is not a fully analyzed detailed design, but an optimal set of interacting component parameters to be used as a basis for further design. Finally, the difference in performance between the global and local reward structures suggests that there may be competition between agents depending on how they receive rewards from their environment which can hinder their ability to explore or exploit variables effectively. These problems having to do with the local reward, in which some agents acting greedily can hinder the overall performance of the system is a typical result in multiagent learning systems, and is often solved through the reward [59]. Within design settings, it suggests that designers not only need to be attentive to how their exploratory behaviors increase their own knowledge of the design space, but how they increase the knowledge of

other designers when collaborating. Otherwise, designers could hinder each other by exploring the design space in contrary ways which do not help either find the global minimum.

4.3 Collaboration and Decomposition in Multiagent Design

The following results compare the effects of designers designing independently or collaboratively. This is represented by allowing the computational designers to submit changes synchronously, with only individual components designed synchronously, or with all variables chosen asynchronously. This is how a design team that collaborates or treats components as independent would behave–if designers communicate and interact, they investigate joint changes between subsystems. On the other hand, if designers remain autonomous, they will only see how their own subsystem changes overall performance, and will make changes individually. While independent component changes most realistically represent this problem, independent variable changes are shown to further illustrate the trend.

Figure 4.3 shows the influence of these three behaviorscollaboration, component asynchrony, and variable asynchronyon the best objective function found by the algorithm over ten runs. As can be seen, there is a significant difference in performance between both the trends of these design processes and the final results. While collaboration continues to improve performance throughout the iterations in the test, both asynchronous design strategies seem to reach process minima, or "floors" mid-way through, after which they are unable to efficiently improve

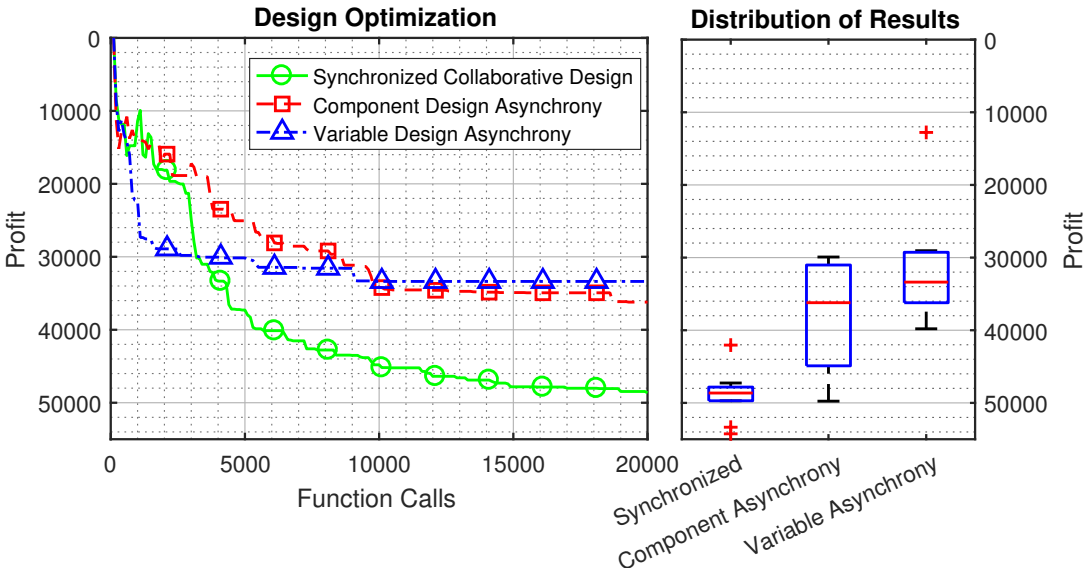

Figure 4.3: Effect of synchronization on agents ability to design. Agents are better able to optimize when they collaborate by designing synchronously than when they are decomposed into groups (by component or by variable) which design asynchronously.

the design. Additionally, the asynchronous variable design strategy seems to be more efficient in the earlier stages, but more quickly stops improving, while the asynchronous component design strategy is slower in the early stages but continues improving, if slowly. Intuitively, it is easy to understand why asynchrony causes the algorithm to prematurely stop improving the design. One of the major defining features of complex engineered systems is the interconnections between componentsit is often said that "the inputs of one subsystem are the outputs of another subsystem" [42]. As a result, the overall performance of the system is related not just to an individual component's contribution, but how different components work together. Therefore, combined changes between components must occur for a design process to be effective. For example, a phone could not be designed with a bigger screen without also changing the size of the supporting frame, the graphics chip, and all of the other supporting componentsotherwise the resulting design would surely fail by violating a constraint, having no design value. In the multidisciplinary context, this must be done through communication and collaboration between designers, which enables designers to design changes which happen in multiple subsystems at the same time.

This problem is further understood when considering the example problem shown in Figure 3.1. This problem essentially has two Nash equilibria–design A-A, and design B-B–the feasible designs. As such, if the agents used design A-A as the current design point, the agents would not be able to proceed to design B-B without collaborating, since they would need to first accept A-B or B-A as the current design point, and both of those designs are much worse than the current design A-

A. As a result, it can be said that collaboration in this model allows the designers to leave designs that are Nash equilibria approach better designs in the space, while independence prevents them from leaving the first Nash equilibrium they enter. This is similar to how the design of federated systems has been theorized– as a "Stag Hunt" game in which collaboration increases design performance but requires the actions of both design teams [25] [26]. In our formulation of the design problem, however, we are able to see this as not just a problem of federated systems, but of systems in general in which design agency has been decomposed.

In addition to making intuitive sense, these results confirm findings of design research about integrated concurrent engineering and suggest a mechanic by which increasing collaboration can produce higher-value solutions. The conclusion most often reached by research studying integrated concurrent engineering is that it decreases the total time in producing a design compared to a more siloed or independent design process [69] [49]. The typical explanation for this is that radically co-location enables engineers to complete a design happens more quickly because of fewer communication delaysengineers design quicker because they can speak face-to-face with the designers of connected subsystems, rather than waiting for an email to return [12]. However, this result shows that decreasing communication barriers between designers can not only increase solution time, but solution quality because of the coupled nature of complex engineered systems. While all design processes were able to reach designs which satisfied requirements by meeting constraints, the collaborative design process did not get stuck in process minimums like the independent processes did. This is because the designers in the multiagent

system could make combined changes which allowed the optimization process to navigate inter-component constraints. This suggests that the relative ease of making combined changes in an integrated collaborative engineering process should also allow design teams to produce higher-quality designs in addition to achieving a faster design process.

4.4 Summary

The results presented in the previous sections are summarized in Table 4.1, in terms of the median and standard deviation of the optimums found, the constraint violation at the final iteration (using $\sum c_i^2$) , and important model parameters, including mass, component cost, energy stored, and mission time. Note that these model parameters are not considered objectives in themselves, but are instead taken into account using the mission model, which calculates the objective based on the value of a theoretical mission (labeled profit). As can be seen, the parameters most correlated with the objective are mission time and stored energy, while others such as component cost and mass do not correlate well with the objective, likely due to the smaller influence of component cost and mass compared with mission revenue and energy storage in the model. In summary, all methods were able to produce feasible designs, but the best-performing method was using the multiagent method with controlled by reinforcement learners rewarded by the global exploratory reward.

Table 4.1: Summary of results. Shown are the median and standard deviation of the optimum objective, constraint violation, mass, cost, energy stored, and mission time for each of the methods tested in the paper: the genetic algorithm and stochastic hill-climbing comparison methods, the multiagent method controlled using the global reward, local reward, a temperature decay and random temperature selection, and the multiagent method using by-variable autonomy and by-component autonomy.

	Comp. Methods		Controllers				Autonomy	
	GA	SHC	G	L	Anneal	Rand	Var	Comp
Med Obj	-46329	-35261	-48641	-42959	-37472	-46697	-33375	-36208
SD of Obj	3335.1	5400.0	3360.3	2992.1	5081.8	2581.3	7630.2	7248.5
Med Cons	0	0	0	0	0	0	0	0
SD Cons	0	0	0	0	0	0	0	0
Med Mass	1.4840	0.6138	1.8746	1.5461	1.48	1.6240	0.8312	1.1088
SD Mass	0.2130	0.3727	0.2459	0.4250	0.3760	0.3398	0.3196	0.3881
Med Cost	774.65	658.16	799.51	777.03	780.94	792.53	745.25	754.77
SD Cost	15.60	34.76	17.14	51.02	40.52	43.08	53.16	32.51
Med Ener	532800	133200	799200	566100	530136	632700	237095	375624
SD Ener	108760	193030	137570	191430	172990	165520	134450	200260
Med Time	41.90	37.84	51.21	45.14	39.50	49.69	35.60	38.33
SD Time	3.59	7.43	3.41	2.89	5.20	3.11	7.91	7.43

5 Conclusion

This thesis explores how the design process should be performed, given the complexity of design across a system which has been decomposed. Specifically, it developed a multiagent model which shows how collaboration can improve the merit of the overall design after a system has been decomposed to different designers or disciplinary groups. This model required developing a new multiagent optimization method which may be used to model the multidisciplinary engineering process by distributing authority over variables in the design problem to computational designers represented as sets of learning agents. This optimization method was shown to perform similarly to a centralized genetic algorithm applied to the same domain when using reinforcement learning and an exploratory reward to control the agents. It further required developing a design problem that allowed for direct comparison between different designs with a number of different objectives of comparison, which was accomplished using a value-driven design approach.

When used as a design model, this model provided a number of insights based on behaviors simulated by the agents. When exploration behaviors were simulated, learning-based control of exploration (and even random control) was shown to outperform annealing, calling into question the idea that engineers should converge on a design in the early design exploration stages. It instead suggests that designers in a collaborative multidisciplinary setting should explore potentially high or low-

impact changes throughout the design exploration process based on the increase in design knowledge. When collaboration (modelled by allowing designers to explore variables at the same time) was compared with designer autonomy (modeled by allowing designers to explore variables asynchronously), collaboration was shown to perform better, while autonomy was shown to become trapped in process minima. This result suggests a mechanism by which increased cooperation can increase design quality: the ability of collaborating designers to explore joint design changes which would otherwise be unavailable if the system was designed autonomously.

5.1 Limitations and Future Work

The model presented in Chapter 3 is able to show the effect of collaboration on design outcomes; however, there are many parts of the design process which could be represented better for further insight. They are listed here, with possible paths forward for future research.

In the context of decomposed design, this model highlights the need for collaboration in order to effectively explore the space of designs, under the assumption that designers act like agents. This justifies the use of many of the design frameworks outlined in Section 1, however, it does not necessarily guarantee their success or failure, as each of these approaches may perform differently than the idealized "autonomous" and "collaborative" behaviors modeled here. As such, further work in agent-based modelling will be needed to compare the effect of using these frameworks to organize design across a decomposed system.

It is known that collaboration can lead to a faster or slower design process due to the transactional costs of collaborating [23]. However, these costs cannot be captured in our model since it only gives the number of function evaluations and models collaboration rather simplistically as the synchrony of the agents choosing new design variable values. This could likely be resolved in future work by assigning times to each individual (during asynchrony) and group (during collaboration) evaluation corresponding to experimentally-determined times for design tasks in autonomous and collaborative settings. While this does not affect conclusions presented here about design outcomes, it would allow for stronger claims about the efficiency of each approach in developing good-enough designs.

Furthermore, a number of behaviors and properties manifest themselves within the design process which simply cannot be encoded in the multiagent model presented here, such as negotiation, differing objectives, communication, and a changing problem. While these individual behaviors and properties are studied in other models (such as [65] and [68] [52]), it is difficult to judge from individual models how each of the potential interactions of these behaviors will affect a design process. Further work needs to be done that draws from each of these models to create an overall model that will be comprehensively representative of team-based design.

5.1.1 Optimizing the System Decomposition: A Direction for Future Work

While this thesis studied how designers should act when a design problem has decomposed, it did not study how that decomposition should be performed in the first place. Since the highest-impact decisions regarding a concept is determined in the early design phase, when a system is first being decomposed, making informed decisions about these concepts is of critical importance to generating a good design. This work is expected to approach the question:

What metrics should a design organization use to compare product decompositions, given information about the cost and risk of individual functions, and how can those metrics be used to automatically generate optimal designs?

5.2 Acknowledgements

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

Bibliography

[1] Adrian K Agogino and Kagan Tumer. Analyzing and visualizing multiagent rewards in dynamic and stochastic domains. *Autonomous Agents and Multi-Agent Systems*, 17(2):320–338, 2008.

[2] Adrian K Agogino and Kagan Tumer. A multiagent approach to managing air traffic flow. *Autonomous Agents and Multi-Agent Systems*, 24(1):1–25, 2012.

[3] Turki Alelyani, Ye Yang, and Paul T Grogan. Understanding designers behavior in parameter design activities. In *ASME 2017 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages V007T06A030–V007T06A030. American Society of Mechanical Engineers, 2017.

[4] James T Allison, Michael Kokkolaras, and Panos Y Papalambros. Optimal partitioning and coordination decisions in decomposition-based design optimization. *Journal of Mechanical Design*, 131(8):081008, 2009.

[5] Mark V Arena, Obaid Younossi, Kevin Brancato, Irv Blickstein, and Clifford A Grammich. Why has the cost of fixed-wing aircraft risen? a macroscopic examination of the trends in us military aircraft costs over the past several decades. Technical report, RAND NATIONAL DEFENSE RESEARCH INST SANTA MONICA CA, 2008.

[6] Donald Birchler, Gary Christle, and Eric Groo. Investigating concurrency in weapons programs. Technical report, CENTER FOR NAVAL ANALYSES ALEXANDRIA VA, 2010.

[7] Daan Bloembergen, Michael Kaisers, and Karl Tuyls. Empirical and theoretical support for lenient learning. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pages 1105–1106. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[8] Joseph George Bolten. *Sources of weapon system cost growth: Analysis of 35 major defense acquisition programs*, volume 670. Rand Corporation, 2008.

[9] Dan Braha, Nam Suh, Steven Eppinger, Michael Caramanis, and Dan Frey. *Complex Engineered Systems*, pages 227–274. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[10] Jonathan Cagan and Kenneth Kotovsky. Simulated annealing and the generation of the objective function: a model of learning during problem solving. *Computational Intelligence*, 13(4):534–581, 1997.

[11] Matthew I Campbell, Jonathan Cagan, and Kenneth Kotovsky. A-design: an agent-based approach to conceptual design in a dynamic environment. *Research in Engineering Design*, 11(3):172–192, 1999.

[12] John Chachere, John Kunz, and Raymond Levitt. The role of reduced latency in integrated concurrent engineering. Technical report, CIFE Working Paper# WP116, 2009.

[13] Wei Chen and HJ Wassenaar. An approach to decision-based design with discrete choice analysis for demand modeling. *Journal of Mechanical Design*, 125:490–497, 2003.

[14] Yong Chen, Ze-Lin Liu, and You-Bai Xie. A multi-agent-based approach for conceptual design synthesis of multi-disciplinary systems. *International Journal of Production Research*, 52(6):1681–1694, 2014.

[15] Paul D Collopy and Peter M Hollingsworth. Value-driven design. *Journal of aircraft*, 48(3):749–759, 2011.

[16] Aris L Dimeas and Nikos D Hatziargyriou. Operation of a multiagent system for microgrid control. *IEEE Transactions on Power systems*, 20(3):1447–1455, 2005.

[17] Shelley D Dionne, Hiroki Sayama, Chanyu Hao, and Benjamin James Bush. The role of leadership in shared mental model convergence and team performance improvement: An agent-based computational model. *The Leadership Quarterly*, 21(6):1035–1049, 2010.

[18] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. Ant colony optimization. *IEEE computational intelligence magazine*, 1(4):28–39, 2006.

[19] Boris Eisenbart, Kilian Gericke, and Luciënne Blessing. An analysis of functional modeling approaches across disciplines. *AI EDAM*, 27(3):281–289, 2013.

[20] Anthony V Fiacco and Garth P McCormick. Extensions of sumt for nonlinear programming: equality constraints and extrapolation. *Management Science*, 12(11):816–828, 1966.

[21] Donald Gerwin and Nicholas J Barrowman. An evaluation of research on integrated product development. *Management Science*, 48(7):938–953, 2002.

[22] Dan L Grecu and David C Brown. Guiding agent learning in design. In *Knowledge intensive computer aided design*, pages 275–293. Springer, 2000.

[23] Paul T Grogan and Olivier L de Weck. Collaboration and complexity: an experiment on the effect of multi-actor coupled design. *Research in Engineering Design*, 27(3):221–235, 2016.

[24] Paul T Grogan and Olivier L de Weck. Collaborative design in the sustainable infrastructure planning game. In *Proceedings of the 49th Annual Simulation Symposium*, page 4. Society for Computer Simulation International, 2016.

[25] Paul T Grogan, Koki Ho, Alessandro Golkar, and Olivier L de Weck. Bounding the value of collaboration in federated systems. In *Systems Conference (SysCon), 2016 Annual IEEE*, pages 1–7. IEEE, 2016.

[26] Paul T Grogan, Koki Ho, Alessandro Golkar, and Olivier L de Weck. Multi-actor value modeling for federated systems. *IEEE Systems Journal*, 2016.

[27] Xiaoyu Gu, John E Renaud, Leah M Ashe, Stephen M Batill, Amrjit S Budhiraja, and Lee J Krajewski. Decision-based collaborative optimization. *Journal of Mechanical Design*, 124(1):1–13, 2002.

[28] George A Hazelrigg. A framework for decision-based engineering design. *Journal of mechanical design*, 120(4):653–658, 1998.

[29] George A Hazelrigg. An axiomatic framework for engineering design. *Journal of Mechanical Design*, 121(3):342–347, 1999.

[30] N Hirschi and D Frey. Cognition and complexity: an experiment on the effect of coupling in parameter design. *Research in Engineering Design*, 13(3):123–131, 2002.

[31] Hans-Peter Hoffmann. Systems engineering best practices with the rational solution for systems and software engineering. *IBM Software Group, Deskbook Release*, 3(2), 2011.

[32] Daniel Hulse, Brandon Gigous, Irem Y Tumer, Christopher Hoyle, and Kagan Tumer. Towards a distributed multiagent learning-based design optimization method. In *ASME 2017 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers, 2017.

[33] J Jänsch and H Birkhofer. The development of the guideline vdi 2221-the change of direction. In *DS 36: Proceedings DESIGN 2006, the 9th International Design Conference, Dubrovnik, Croatia*, 2006.

[34] Y Jin and SC-Y Lu. Agent based negotiation for collaborative design decision making. *CIRP Annals-Manufacturing Technology*, 53(1):121–124, 2004.

[35] Yan Jin and Raymond E Levitt. i-agents: Modeling organizational problem solving in multi-agent teams. *Intelligent Systems in Accounting, Finance and Management*, 2(4):247–270, 1993.

[36] Yan Jin and Raymond E Levitt. The virtual design team: A computational model of project organizations. *Computational & Mathematical Organization Theory*, 2(3):171–195, 1996.

[37] Yan Jin and Stephen C-Y Lu. An agent-supported approach to collaborative design. *CIRP Annals-Manufacturing Technology*, 47(1):107–110, 1998.

[38] Dervis Karaboga and Bahriye Basturk. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *Journal of global optimization*, 39(3):459–471, 2007.

[39] James Kennedy. Particle swarm optimization. In *Encyclopedia of machine learning*, pages 760–766. Springer, 2011.

[40] Mark Klein, Hiroki Sayama, Peyman Faratin, and Yaneer Bar-Yam. The dynamics of collaborative design: insights from complex systems and negotiation research. *Concurrent Engineering*, 11(3):201–209, 2003.

[41] Steven M. Kosiak. Is the u.s. military getting smaller and older?: And how much should we care? Technical report, Center for a New American Security, 2017.

[42] Ilan Kroo, Steve Altus, Robert Braun, Peter Gage, and Ian Sobieski. Multidisciplinary optimization methods for aircraft preliminary design. In *5th Symposium on Multidisciplinary Analysis and Optimization*, page 4325, 1994.

[43] Benjamin Kruse, Clemens Münzer, Stefan Wölkl, Arquimedes Canedo, and Kristina Shea. A model-based functional modeling and library approach for mechatronic systems in sysml. In *ASME 2012 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages 1217–1227. American Society of Mechanical Engineers, 2012.

[44] Jesko G Lamm and Tim Weilkiens. Functional architectures in sysml. *Proceedings of the Tag des Systems Engineering (TdSE10). Munich, Germany*, 2010.

[45] Susan E Lander. Issues in multiagent design systems. *IEEE expert*, 12(2):18–26, 1997.

[46] Lindsay Hanna Landry and Jonathan Cagan. Protocol-based multi-agent systems: examining the effect of diversity, dynamism, and cooperation in heuristic optimization approaches. *Journal of Mechanical Design*, 133(2):021001, 2011.

[47] Jae Won Lee. Stock price prediction using reinforcement learning. In *Industrial Electronics, 2001. Proceedings. ISIE 2001. IEEE International Symposium on*, volume 1, pages 690–695. IEEE, 2001.

[48] Charlie Manion, Nicolás F Soria, Kagan Tumer, Chris Hoyle, and Irem Y Tumer. Designing a self-replicating robotic manufacturing factory. In *ASME 2015 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages V01BT02A045–V01BT02A045. American Society of Mechanical Engineers, 2015.

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

[50] Joaquim RRA Martins and Andrew B Lambe. Multidisciplinary design optimization: a survey of architectures. *AIAA journal*, 51(9):2049–2075, 2013.

[51] Christopher McComb, Jonathan Cagan, and Kenneth Kotovsky. Lifting the veil: drawing insights about design teams from a cognitively-inspired computational model. *Design Studies*, 40:119–142, 2015.

[52] Christopher McComb, Jonathan Cagan, and Kenneth Kotovsky. Rolling with the punches: An examination of team performance in a design task subject to drastic changes. *Design Studies*, 36:99–121, 2015.

[53] Christopher McComb, Jonathan Cagan, and Kenneth Kotovsky. Optimizing design teams based on problem properties: computational team simulations and an applied empirical test. *Journal of Mechanical Design*, 139(4):041101, 2017.

[54] George E Monahan. State of the arta survey of partially observable markov decision processes: theory, models, and algorithms. *Management Science*, 28(1):1–16, 1982.

[55] Jarrod Moss, Jonathan Cagan, and Kenneth Kotovsky. Learning from design experience in an agent-based design system. *Research in Engineering Design*, 15(2):77–92, 2004.

[56] Gerhard Pahl and Wolfgang Beitz. *Engineering design: a systematic approach*. Springer Science & Business Media, 2007.

[57] Liviu Panait, Keith Sullivan, and Sean Luke. Lenient learners in cooperative multiagent systems. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 801–803. ACM, 2006.

[58] Liviu Panait, Karl Tuyls, and Sean Luke. Theoretical advantages of lenient learners: An evolutionary game theoretic perspective. *Journal of Machine Learning Research*, 9(Mar):423–457, 2008.

[59] David C Parkes and Lyle H Ungar. Learning and adaption in multiagent systems. In *Proc. of AAAI-97 Workshop on Multiagent Learning, Providence, RI*, 1997.

[60] Manisa Pipattanasomporn, Hassan Feroze, and Saifur Rahman. Multi-agent systems in a distributed smart grid: Design and implementation. In *Power Systems Conference and Exposition, 2009. PSCE'09. IEEE/PES*, pages 1–8. IEEE, 2009.

[61] Mark Price, Srinivasan Raghunathan, and Richard Curran. An integrated systems engineering approach to aircraft design. *Progress in Aerospace Sciences*, 42(4):331–376, 2006.

[62] Kevin L Priddy and Paul E Keller. *Artificial neural networks: an introduction*, volume 68. SPIE press, 2005.

[63] Mohammad Reza Danesh and Yan Jin. An agent-based decision network for concurrent engineering design. *Concurrent Engineering*, 9(1):37–47, 2001.

[64] J Richardt, F Karl, and C Müller. Connections between fuzzy theory, simulated annealing, and convex duality. *Fuzzy sets and systems*, 96(3):307–334, 1998.

[65] Hiroki Sayama, Dene L Farrell, and Shelley D Dionne. The effects of mental model formation on group decision making: An agent-based simulation. *Complexity*, 16(3):49–57, 2011.

[66] Vishal Singh, Andy Dong, and John S Gero. Effects of social learning and team familiarity on team performance. In *Proceedings of the 2009 Spring Simulation Multiconference*, page 6. Society for Computer Simulation International, 2009.

[67] Vishal Singh, Andy Dong, and John S Gero. Developing a computational model to understand the contributions of social learning modes to task coordination in teams. *AI EDAM*, 27(1):3–17, 2013.

[68] Vishal Singh, Andy Dong, and John S Gero. Social learning in design teams: The importance of direct and indirect communications. *AI EDAM*, 27(2):167–182, 2013.

[69] J Smith. Concurrent engineering in the jet propulsion laboratory project design center. 1998.

[70] Ricard V Solé, Ramon Ferrer-Cancho, Jose M Montoya, and Sergi Valverde. Selection, tinkering, and emergence in complex networks. *Complexity*, 8(1):20–33, 2002.

[71] Nicolás F Soria, Mitchell K Colby, Irem Y Tumer, Christopher Hoyle, and Kagan Tumer. Design of complex engineering systems using multiagent coordination. In *ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages V02AT03A001–V02AT03A001. American Society of Mechanical Engineers, 2016.

[72] Matthijs TJ Spaan. Partially observable markov decision processes. In *Reinforcement Learning*, pages 387–414. Springer, 2012.

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

[74] Robert B Stone and Kristin L Wood. Development of a functional basis for design. *Journal of Mechanical design*, 122(4):359–370, 2000.

[75] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press Cambridge, 1998.

[76] Deborah L Thurston. Utility function fundamentals. In *Decision making in engineering design*. ASME Press, 2006.

[77] T Tomiyama, P Gu, Y Jin, Diederick Lutters, Ch Kind, and F Kimura. Design methodologies: Industrial and educational applications. *CIRP Annals-Manufacturing Technology*, 58(2):543–565, 2009.

[78] David Ullman. *The mechanical design process*. McGraw-Hill Science/Engineering/Math, 2009.

[79] Steven Ulrich, Karl Tand Eppinger. *Product design and development*. McGraw-Hill Education, 2012.

[80] John Von Neumann and Oskar Morgenstern. *Theory of games and economic behavior (commemorative edition)*. Princeton university press, 2007.

[81] Henk Jan Wassenaar and Wei Chen. An approach to decision-based design with discrete choice analysis for demand modeling. *Journal of Mechanical Design*, 125(3):490–497, 2003.

[82] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[83] Gerhard Weiss. *Multiagent Systems*. MIT Press, 2nd edition edition, 2013.

[84] Kristen L Wood, RJ B Stone, DRJ Mcadams, J Hirtz, and Simon Szykman. A functional basis for engineering design: Reconciling and evolving previous efforts. Technical report, National Institute of Standards and Technology, 2002.

[85] Unal Yildirim, Felician Campean, and Huw Williams. Function modeling using the system state flow diagram. *AI EDAM*, 31(4):413–435, 2017.

[86] Logan Yliniemi, Adrian K Agogino, and Kagan Tumer. Multirobot coordination for space exploration. *AI Magazine*, 35(4):61–74, 2014.

[87] Nicolás F Soria Zurita, Mitchell K Colby, Irem Y Tumer, Christopher Hoyle, and Kagan Tumer. Design of complex engineered systems using multi-agent coordination. *Journal of Computing and Information Science in Engineering*, 18(1):011003, 2018.