AN ABSTRACT OF THE DISSERTATION OF

Logan Michael Yliniemi for the degree of Doctor of Philosophy in Robotics and Mechanical Engineering presented on April 21, 2015.

Title: Multi-Objective Optimization in Multiagent Systems

Abstract approved: _____

Kagan Tumer

Cooperative multiagent systems are used as solution concepts in many application domains including air traffic control, satellite communications, and extra planetary exploration. As systems become more distributed and complex, we observe three phenomena. First, these systems cannot be accurately modeled, rendering traditional model based control methods inadequate. Second, system parameters are highly coupled in a nonlinear manner, making it difficult for humans to develop heuristic based control policies. Finally, these systems are distributed to the point that a centralized controller is either impractical or infeasible. These types of systems are often inherently multi-objective; unfortunately, they are not treated as such in most multiagent research. To date, there has been little research attention given to multi-objective multiagent systems.

This dissertation addresses these systems from a learning-based approach to optimize system performance in four ways: (i) deriving a form of credit assignment compatible

for use with multi-objective problems (ii) deriving multiagent equivalents to state-of-the-art multi-objective evolutionary algorithms (MOEAs); (iii) developing a fast, effective multiagent multi-objective algorithm that outperforms state-of the art MOEAs in as little as one tenth of the computation time; and (iv) integrating the previously developed algorithm into a multiagent system.

Multi-Objective Optimization in Multiagent Systems

by

Logan Michael Yliniemi

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented April 21, 2015
Commencement June 2015

Doctor of Philosophy dissertation of Logan Michael Yliniemi presented on April 21, 2015.

APPROVED:

Major Professor, representing Robotics and Mechanical Engineering

---

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

---

Dean of the Graduate School

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

Logan Michael Yliniemi, Author

ACKNOWLEDGEMENTS

I would like to acknowledge and thank my advisor, Dr. Kagan Tumer, for supporting my research and providing guidance in pursuit of this work.

I would also like to thank Dr. Sam Devlin and Drew Wilson for their collaboration.

Additionally, I would like to thank my friends and colleagues in the AADI lab for their support, criticisms, and demeanour throughout this process.

A special thanks also to Dr. Joseph K. Davidson at Arizona State, for starting me down this path.

Finally, I would like to thank Carey E. and Elizabeth P. Yliniemi for their support and guidance through the years.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF TABLES

DEDICATION

This work is dedicated to those who have guided my path:

Madeline and Jake Yliniemi

And to the next generation:

Cruz, Presley, Marley, Turner, and Collins

Chapter 1 – Introduction

In large, complex systems, the control of any individual component can be deceptively complicated to perform efficiently. First, controlling any one component may have downstream effects on different parts of the system, which may not be fully realized until some amount of time has passed. Second, controlling an individual component to maximize some locally available measure may have detrimental effects on the system at large, as the component may — in simply seeking to perform its task well — make the tasks of other components more difficult. Finally, simply attempting to control all of the components from one central location is often infeasible, due to constraints on communication, latency, or simply the massive computation time that may be involved with trying to manage the operation of every component in a system simultaneously.

As one example, consider the prospect of autonomous, self-driving cars. A regulatory body wants to ensure public safety, maximize the throughput of the available traffic infrastructure, and minimize emissions. The customers want the to ensure their personal safety, and have the cheapest, most luxurious, lowest maintenance transportation experience possible. The car companies want to maximize their profit, maximize their market share, and minimize the regulations that are imposed on them. All of these interests are then intermingled with concerns from disciplines as varied as robust control, automotive engineering, computer vision, material science, economics, legal/policy, and human–robot interaction. This is a complex, distributed, multiagent cyber-physical sys-

tem with multiple priorities shared among many people and entities.

In addition, oftentimes these problems begin to outstrip the human ability to understand the dynamics of the system as a whole. It is circumstances like these where a decentralized "multiagent" approach — using locally available information, on-board computation, and making decisions based on measures that are formulated with respect to the team's interest (instead of the individual components) — can offer significant benefits.

Cooperative multiagent systems focuses on producing a set of autonomous agents to achieve a system-level goal [121]. Multiagent frameworks have been used to study complex, real-world systems like air traffic [106], teams of satellites [29], and extra-planetary rover exploration [3]. In each case, the goal is to optimize a single, well-defined objective function.

But, in many of these cases, the problems lend themselves more naturally to multiple objectives: for example, air travel should be as expedient as possible while minimizing the congestion in the airspace, and each carrier wants to maximize their profits. Satellites may need to make observations for multiple separate institutions. Extra-planetary rovers should acquire multiple different types of scientific data. However, most research in multiagent systems does not take a multi-objective viewpoint: they typically seek to find a single usable solution, without considering the tradeoffs between potential alternatives that would increase one objective' s value at the cost of another.

Having multiple criteria or objectives that each of these solutions are being evaluated on makes a fundamentally difficult problem: the solutions cannot be ranked in a simple manner, and some form of higher reasoning must be used. These sorting mechanisms

can be as simple as a linear combination of the objectives, which comes with various well-documented advantages and drawbacks, or as complex as an algorithm that takes the objective evaluations for many candidate solutions and compares them all pairwise against each other before creating a multi-stage ranking system. No matter the method used for calculating how good an individual solution is, the goal is for a set of solutions as a whole to accurately embody the optimal tradeoffs between the objectives. These tradeoff solutions, which form the *Pareto front*, are a key solution concept in multi-objective problems. A *Pareto optimal solution* is one in which the evaluation of one objective cannot be improved without worsening the evaluation on at least one other objective. Within the setting of autonomous agents, the goal is to create an algorithm which develops polices that achieve these points of Pareto optimal performance without requiring much knowledge of the system being optimized.

Developing successful agent policies in multiagent systems can be challenging. One successful approach is to use adaptive agents with tools like reinforcement learning or evolutionary algorithms. Each agent seeks to maximize its own reward or evaluation; with a properly designed reward signal, the whole system will attain desirable behaviors. This is the science of credit assignment: determining the contribution each agent had to the system as a whole. Clearly quantifying this contribution on a per-agent level is essential to multiagent learning. This is an issue that has not been studied within the context of multiple objectives. In this work we address the challenges that arise when multiagent systems are combined with multi-objective problems.

The primary challenge in designing adaptive agents suited for multiagent, multi-objective problems is the incorporation of credit assignment into algorithms that handle

the tradeoffs between multiple objectives in a way that generates a wide spread of solutions along the Pareto front. We consider two classes of multi-objective methods to achieve this. First, *a priori* methods explicitly define what makes any solution more or less desirable than another solution before the optimization process begins. Second, *a posteriori* methods do not explicitly define what makes solutions desirable beforehand, and instead seek to create a wide variety of solutions that accurately represent the tradeoffs between the different objectives.

Within these two classes, there are a wide variety of different methods that have been developed, and integrating credit assignment into these algorithms, at this time, must be done on a case-by-case basis. In this work, we focus on one sub-class of algorithms within each: for *a priori* methods we use aggregation methods, where the objective evaluations are combined into a single scalar value to be optimized. For *a posteriori* methods, we focus on Pareto-based multi-objective evolutionary algorithms (MOEAs), which generate (at least) one population of solutions and compare each of these solutions to every other in order to generate a ranking of solutions within the population. When altering each of these methods for use inside multiagent systems, we identify weaknesses of each approach, and later in this work, develop an approach that circumvents both the weaknesses of aggregation methods and the weaknesses of MOEAs. It is important to note that there are many other different types of both *a priori* and *a posteriori* methods, which are not discussed in this work. Developing these algorithms for use with multiagent systems is deferred to later work.

Contributions This dissertation sits firmly at the intersections of the fields of autonomous agents, multiagent systems, and multi-objective problems, and addresses the issues that arise when these fields are combined. The specific contributions of this dissertation are to:

- derive methods for automatically assigning credit for a team's success or failure to members of that team in the presence of multiple objectives (Chapter 3)
- derive effective multiagent equivalents to state-of-the-art multi-objective algorithms (Chapter 4)
- develop a fast, effective multi-objective algorithm that outperforms state-of the art MOEAs in as little as one tenth of the computation time (Chapter 5)
- theoretically prove that this fast multi-objective algorithm will produce Pareto optimal results that cover the entire Pareto front to an arbitrarily fine resolution (Chapter 5–6)
- develop a framework for integrating this fast multi-objective algorithm into multiagent systems (Chapter 7)

Roadmap In the remainder of this introduction, we briefly discuss the contents of each of the chapters to follow. In Chapter 2, we introduce all necessary background and definitions of multiagent systems, credit assignment, and multi-objective problems. Because each chapter beyond this uses a different portion of the background provided in this chapter, in each following chapter we explicitly identify the sections of background that are necessary for a complete understanding of that chapter.

In Chapter 3, we first examine the use of a multi-objective aggregation within a multiagent reinforcement learning context [125]. We show that the use of credit assignment in multi-objective, multiagent systems is of paramount importance, but also that the method of multi-objective aggregation is an important factor in system performance as well. To achieve the best possible performance, both credit assignment and the correct choice of aggregation is necessary, but the use of credit assignment boosts system performance even when a sub-optimal aggregation is used.

Chapter 4 develops the concept of credit assignment for use in the popular MOEA NSGA-II [128]. We show that NSGA-II is ill-defined for use in multiagent systems, and develop an equivalent ranking scheme that works well with credit assignment. We compare the use of NSGA-II to the use of aggregation methods, and show that NSGA-II provides powerful performance benefits when paired with proper credit assignment techniques. However, we identify that the run-time of NSGA-II in a multiagent system is prohibitively slow.

In Chapter 5, we draw a concept from the origins of multi-objective optimization, that of *indifference*, and develop a single-parameter transformation that allows us to use this concept to create a search space that is biased in favor of aggregation methods [126]. The optimization in the transformed space then does not have the computational slowdown of MOEAs, and does not bear the weaknesses of the aggregation methods without the transformation. We also offer theoretical proofs that state that the solutions discovered in this transformed space will be Pareto optimal in the original space, even in portions of the Pareto front that would not be discovered by the aggregation methods in the original space.

In Chapter 6, we offer three extensions for the transformation developed in Chapter 5, which provide various benefits allowing the system designer to specify additional parameters to control the behavior of the transformation to better suit an individual problem instance [127]. One of these extensions also allows us to offer a proof that the complete Pareto front will be discovered to an arbitrarily fine resolution, which is also provided in this chapter.

In Chapter 7 we discuss the incorporation of the transformation from Chapter 5 into a multiagent system. We identify a problem that arises when the original transformation is used in a multiagent system related to the concept of Nash Equilibria [81], and provide a simple method for circumventing this problem. We compare our results to the multiagent NSGA-II developed in Chapter 4.

Finally, in Chapter 8, we draw the conclusion of this work and identify avenues for future research in this area.

Chapter 2 – Background

In this chapter we introduce concepts from adaptive autonomous agents (Section 2.1), multiagent systems and credit assignment (Section 2.2), we provide definitions related to multi-objective problems (Section 2.3), and discuss established multi-objective methods (Section 2.4).

In the beginning of each chapter to follow, we briefly overview the relevant background, as well as specifically identifying the sections of this background chapter that are necessary for a full understanding of that chapter.

2.1 Adaptive autonomous agents

Adaptive autonomous agents require four properties: they must be able to sense information about their environment, reason based on that information, act upon the environment, and receive some form of feedback from the environment regarding their actions. This feedback is typically designed by a human system designer, who specifies a function that, when maximized, achieves the goals that the system designer seeks to achieve through the autonomous agent. This goal may be simple or complex, but the agent will myopically seek to increase that evaluation, forsaking all else.

Many methods exist for developing the policies by which the agent maps its sensory information to the actions it will take on the environment. In this dissertation, we work

with two: reinforcement learning (RL) and evolutionary algorithms (EA). When multiple agents are performing evolutionary algorithms cooperatively and simultaneously, this process is called a cooperative coevolutionary algorithm (CCEA).

These different communities have different terminology for a measure of performance. In reinforcement learning, an agent is provided with a *reward* measuring its impact toward achieving a goal during a time step. In evolutionary algorithms, an agent is provided with an *fitness evaluation* of its entire policy. In game theory, this measure is known as a *payoff*. Finally, in multi-objective problems, these are known as *objectives*. In this work, we refer to the system-level team performance as an objective, and the value given to the individual agent as a reward (reinforcement learning) or evaluation (evolutionary algorithms). When speaking in general terms, we refer to these concepts as an evaluation.

In both RL and EAs, each autonomous agent performs four processes in order, at each timestep. This process is described in Algorithm 2.1. First, the agent gathers sensory data about where its body is in the environment (sense(B,E)). Next, the agent uses its policy, π, and the previously gathered sensory information to reason about what action it should take (decide(π)). Third, it takes the action that it decided upon in the previous step, which affects its body and the environment (act(B,E,π)). Finally, it receives some feedback from the environment based on how its action affected its body and the environment around it (react(B,E,π)). How this feedback is incorporated to change the agent's policy differs between RL and EAs, which we discuss below.

Algorithm 2.1: Typical adaptive autonomous agent algorithm

Input: Initial agent policy π_0, agent body B, environment E

foreach *timestep t* **do**

    sense(B,E) // gather information about the environment

    decide(π) // reason about the information gathered

    act(B,E,π) // take action

    react(B,E,π) // update policy (RL) or gather partial evaluation (EA)

end

Output: Updated policy π_T (RL) or fitness evaluation F (EA)

2.1.1 Reinforcement learning

One common approach to producing effective solutions in single- or multiagent problems is the use of reinforcement learning: An agent uses trial and error to learn how to increase the reward signal that it receives based on the actions it takes [102]. With a properly designed reward signal, the agent will learn to perform a task in a way that is desirable to the system designer. The goal of a reinforcement learner is to maximize the long-term expected payoff of the actions that it takes, and it records values which it associates with actions as it receives feedback from the environment. By usually choosing the action with the maximum reward (exploitation) but occasionally choosing an action with a lower-perceived reward (exploration), an agent can simultaneously achieve high performance while also improving the accuracy of its value function. One method of doing this is through the use of Q-learners, which use the update [102]:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[R + \gamma\, Qmax_{a'}(s',a') - Q(s,a)] \tag{2.1}$$

to update their estimation of the expected long-term reward of taking action a while in state s, where $Q(s, a)$ is the Q-value for the state-action pair (s, a), γ is the discount factor, and $Qmax_{a'}(s', a')$ returns the maximum Q value over all actions available to the next state-action pair [64]. In cases where only a single state exists, γ may be set to zero, and the Q-learner reduces to an action-value learner [102]. In the case of a multiagent system, multiple independent Q-tables can be maintained, one corresponding to each agent. The state and action then correspond to the individual agent's state and action, instead of the system's joint state and action.

2.1.2 Evolutionary algorithms

Algorithm 2.2 describes a typical evolutionary algorithm, which are a biologically-inspired computational technique in which a population of agent policies is first randomly generated (Input line), and then tested in some domain (Line 4). After calculating a scalar evaluation of an agent's "fitness" for each agent (Line 4), those with lower fitness are removed from the population (Line 6), and replaced with slightly-altered copies of their higher-fitness counterparts (Line 2). Through this random alteration and intelligent selection, system performance increases as the agents adapt to the domain to maximize their fitness evaluation calculation.

An evolutionary algorithm is defined by the number of individuals in the population at maximum and minimum size (before and after selection), whether members from the previous generation can survive into the new generation, and four operators: initialization, mutation, evaluation, and selection. Initialization describes how the population is

Algorithm 2.2: Typical evolutionary algorithm

Input: Initialize population of k policies $\vec{\pi}$
foreach *generation* $g \in 1 : G$ **do**
    replenish($\vec{\pi}$) `// add k mutated copies of solutions to` $\vec{\pi}$
    foreach $\pi_i \in \vec{\pi}$ **do**
        evaluate(π_i) `// Simulate policy (Alg. 2.1)`
    end
    select($\vec{\pi}$) `// Remove lower-performing policies`
end
Output: Final solution population $\vec{\pi}_G$

built in the beginning. Mutation describes how new population members are created from surviving policies. Evaluation describes the methods by which a policy is evaluated. Selection describes how the lower-performing policies are selected for removal from the population. These four operators fully describe an evolutionary algorithm. In this work, we focus on the evaluation step, and how creating a bespoke evaluation specifically for multiagent, multi-objective processes can boost performance over a less-reasoned approach.

2.2 Multiagent systems

In a large, complex system where a high level of coordination between different elements of the system is necessary, there are two paradigms that may be used to develop policies to control the system.

First, consider a centralized system in which one centralized authority is making all necessary decisions for the entire system as a whole, and all components in the system are merely following orders. The advantages here are that perfect coordination is pos-

sible, and the pieces of the system as a whole will cooperate to increase system performance. This typically works well for small systems consisting of just a few agents [102]. However, such a centralized system can fall prey to complexities such as communication restrictions, component failures — especially where a single point of failure can stop the entire system — and simply the difficulty of simultaneously solving a problem for hundreds or thousands of agents simultaneously [124].

Second, consider a decentralized approach such that many independent elements are each attempting to solve a smaller piece of the problem. These elements then, working in tandem, could achieve a system level goal through cooperation. This paradigm does not guarantee that this central point of failure will not exist, but a decentralized solution concept may not have a centralized point of failure, while a centralized controller always will.

However, in a sufficiently complex system, the question of exactly what tasks each of the elements should be seeking to perform becomes very difficult to determine. Simply specifying an overarching system-level goal is often not sufficient to achieve that goal through adaptive autonomy, and additional steps must be taken to allow the system to succeed.

2.2.1 Evaluation shaping

A multiagent learning system depends on a way to measure the value of each agent's behavior. This measurement is called an evaluation function, and changing what form the evaluation function takes is the science of "reward shaping" [17,51,59,74,101,110].

An agent will seek to solely increase its evaluation function, forsaking all other concerns, so it is important that it has two specific properties.

First, the evaluation function must be "sensitive" to the actions of the agent [119]. An agent taking good actions should receive a high evaluation, and an agent taking poor actions should receive a lower evaluation. In an unpredictable, stochastic, or multiagent environment, there are other factors affecting the evaluation that the agent will receive. An ill-developed evaluation function will allow these random factors to insert a large amount of "noise" into the "signal" offered by the evaluation function, and as the signal-to-noise ratio decreases, so does the agent's performance.

Second, the evaluation function must be "aligned" with the overall mission that the agent team must achieve [119]. That is, an agent that increases its own evaluation should simultaneously be increasing the system performance. A lack of alignment can lead to situations such as the Tragedy of the Commons [27, 52], wherein a group of rationally self-concerned agents lead to a drop in system performance due to working at cross-purposes. That is, agent A does what it perceives in its own best interest, as does agent B; in some way, their actions deplete their shared environment, and lead to both agents being worse off than they would be had they cooperated for the communal good.

Both of these properties — sensitivity and alignment — are critical to multiagent systems. An agent must be able to clearly discern what it has done to earn a high evaluation, and continuing to earn that high evaluation or improving upon that evaluation must be in the best interest of the system as a whole [124].

2.2.2 Credit assignment

A **local evaluation function** (L_i) is the evaluation based on the part of the system that an agent i can directly observe. Using this reward signal often encourages "self-ish" behavior, in which the agent may act at cross-purposes with other agents while blindly increasing its own evaluation, causing poor overall system performance. A local evaluation will be highly sensitive, but will often lack alignment.

The **global evaluation function** (G) is the system performance of the team as a whole. Training on this signal encourages the agent to act in the system's interest, but includes a large amount of noise from other agents acting simultaneously. The global evaluation will be perfectly aligned, but lack sensitivity.

The **difference evaluation function** (D_i) is a shaped reward signal that helps an agent quickly learn the consequences of its actions on the system [4]. It is defined as:

$$D_i(z) = G(z) - G(z_{-i}) \tag{2.2}$$

where $G(z)$ is the global system performance for the system considering the joint state-action z, and $G(z_{-i})$ is $G(z)$ for a theoretical system without the contribution of agent i. Any action taken to increase D_i simultaneously increases G, while agent i's impact on its own reward is much higher than its relative impact on G [4]. Difference evaluations, then, have high sensitivity and alignment.

Difference evaluations have seen a wide variety of applications, such as data routing over a telecommunication network [113], multiagent gridworld [110], conges-

tion games such as traffic toll lanes [40, 111, 112, 119], distributed product marketing [109], creativity evaluation in product design [92], rover coordination [68], and urban road traffic management [115], and optimization problems such as bin packing [120], and faulty device selection [107].

Additionally, difference evaluations offer proven performance benefits over using a global evaluation: in difficult domains, difference evaluations provide a higher probability of selecting the optimal action after training, while in simple domains, difference evaluations provide at least an equal probability of selecting the optimal action compared to training with the global evaluation [26].

2.2.3 Cooperative coevolutionary algorithms

Cooperative coevolutionary algorithms leverage the concept of evolutionary algorithms for team-based domains. Algorithm 2.3 describes a typical cooperative coevolutionary algorithm. In coevolutionary algorithms, multiple separate populations are maintained, and are used in a shared simulation environment, where their fitness is evaluated based on how well they perform an assigned task as a member of a team made up of members from each population. An evolutionary algorithm is carried out on each population individually, such that the populations eventually produce agent policies that are well-suited in the team-based environment, to maximize the team's calculated fitness.

Coevolutionary algorithms have the potential to speed up a search through a complex space, but can often lead to a suboptimal area of the search space [83, 89]. This can be

Algorithm 2.3: Typical cooperative coevolutionary algorithm

Input: Initialize n populations of k policies: $\vec{\Pi}_0 = \{\vec{\pi_1}, \vec{\pi_2} \cdots \vec{\pi_n}\}$

foreach *generation* $g \in 1 : G$ **do**
  foreach *population* $\vec{\pi}_i \in \vec{\Pi}^g$ **do**
    replenish($\vec{\pi_i}$) `// add k mutated copies of policies to` $\vec{\pi}_i$
  end
  foreach $i \in 1 : size(\vec{\pi})$ **do**
    `// form random teams`
    randomly select one policy p (no replacement) from each population: $\pi_{i,p}$
    add $\pi_{i,p}$ to team T_i
    evaluate(T_i) `// Simulate team, assign fitness to members`
  end
  foreach *population* $\vec{\pi}_i^g \in \vec{\Pi}^g$ **do**
    $\vec{\pi}_i^{g+1} \leftarrow$ select($\vec{\pi}_i^g$) `// Remove lower-performing policies`
  end
end

Output: Final solution populations $\vec{\Pi}^G$

due to the agents learning to take a conservative strategy, being able to cooperate with a broader range of teammates [83, 84].

2.3 Multi-objective terminology

A multi-objective problem, also known as vector optimization, is a problem in which the quantity to be optimized takes the form of a vector instead of a scalar. In these problems, there are multiple objectives to be optimized simultaneously, and the goal is to find the best solutions for each objectives individually, as well as the many solutions that describe the optimal tradeoffs between the objectives, the Pareto front.

Multi-objective problems appear in an extremely wide variety of different engineer-

ing domains, including evacuation planning [69], traffic-based route planning [65], hydrologic modeling [123], air traffic management [15], nurse scheduling [12], supply chain networks [18], the design of high-speed transport planes [76], the design of trusses [24], job shop scheduling [130], urban planning [10], and greywater reuse [87].

Within the multi-objective community, the convention for optimization is minimization. This conflicts with the convention within the multiagent community, and because we combine concepts from the two communities, this convention does change internally in this dissertation. In this chapter, we assume (without loss of generality) pure minimization of k objectives $\Lambda \in \mathbb{R}^k$ through the control of the n design variables $\Omega \in \mathbb{R}^n$.

Multi-objective spaces: $\Omega \in \mathbb{R}^n$ is the *design variable space* (domain). $\Lambda \in \mathbb{R}^k$ is *objective space* (range or codomain) [117]. The mapping from $\Omega \rightarrow \Lambda$ is unknown, and must be determined through simulation.

Domination A solution u dominates another solution v ($u \prec v$) if it scores lower on all criteria (objectives $c \in \Lambda$): $\forall c \in \Lambda[f_c(u) < f_c(v)]$. A solution u weakly dominates another solution v ($u \preceq v$) if it scores equal on some objectives, but less on others: $\forall c \in \Lambda[f_c(u) \leqslant f_c(v)] \wedge \exists j \in \Lambda[f_j(u) < f_j(v)]$ [117].

Pareto Front A solution which is not dominated by any other feasible solution is part of the Pareto front $\mathbf{P}^*$. As an incomplete optimizer solves a problem, it will approximate $\mathbf{P}^*$ with a *Pareto approximation* P_I^* at iteration I. P_I^* is the subset of solutions discovered by an incomplete optimizer during training which are not dominated by any other point discovered before iteration I. This is a distinct concept from the true Pareto

Algorithm 2.4: Maintaining an estimate of the Pareto front, P_I^*

Input: v, P_I^*

```
// attained vector v, previous Pareto front estimate P_I^*
foreach vector u ∈ P_I^* do
    // Does u dominate v?
    if u ≺ v then
        // If yes, P_I^* is unchanged
        return P_{I+1}^* ← P_I^*;
    end
    // If no, P_I^* may change
end
foreach vector u ∈ P_I^* do
    // Does v dominate any u?
    if v ≺ u then
        // If so, remove u
        P_I^* ← P_I^* \ u;
    end
end
P_I^* ← P_I^* ∪ v;
P_{I+1}^* ← P_I^*;
```

Output: P_{I+1}^*

front, as many points in P_I^* in the early stages of training will be found to be dominated at later iterations. The true Pareto front is always on the border of the attainable and unattainable space [42].

Utopia and nadir vectors Two important concepts in multi-objective problems are the utopia and nadir points. The utopia point takes on the best possible value for each objective, minus some small amount so that it is always infeasible. This point is difficult to find, requiring an optimization for each objective individually. Instead, we approxi-

mate:

$$\hat{u}^{\circ}(c) = min(P_I^*(c)) - \Delta \tag{2.3}$$

where $\hat{u}^{\circ}(c)$ is the c^{th} element in the estimated utopia vector, $min(P_I^*(c))$ is the minimum c^{th} element of any vector in P_I^*, and Δ is a small value [33]. The nadir point takes the worst value for each objective in the Pareto optimal set [75], which we approximate:

$$\hat{u}^{\bullet}(c) = max(P_I^*(c)) \tag{2.4}$$

These approximations are computationally very cheap, and offer no guarantees of accuracy. They also vary over time: as P_I^* is refined to be a closer approximation of $\mathbf{P}^*$, these approximations will be closer to the true utopia and nadir point, but their accuracy is completely dependent on the accuracy of P_I^*.

Empirical attainment functions Empirical attainment functions (EAFs) are a method for statistical analysis of multi-objective methods, developed by Fonseca et al. [49]. EAFs show the area of the objective space that a solution concept being analyzed "attains" (dominates) over a number of statistical runs. They typically show contour lines that correspond to points in the solution space which are attained once, in 50% of cases, and in all cases, though they may show contour lines for any percent of attainment. Figure 2.1 shows an example EAF. The median attainment line is the most reliable way to compare multiple solution methods, as it represents the portion of the objective space that will be attained by a typical single statistical run. The worst- and best-case contours, by definition, describe extremes. For problems with more than two objectives,

EAFs become increasingly difficult to calculate and represent visually [49].

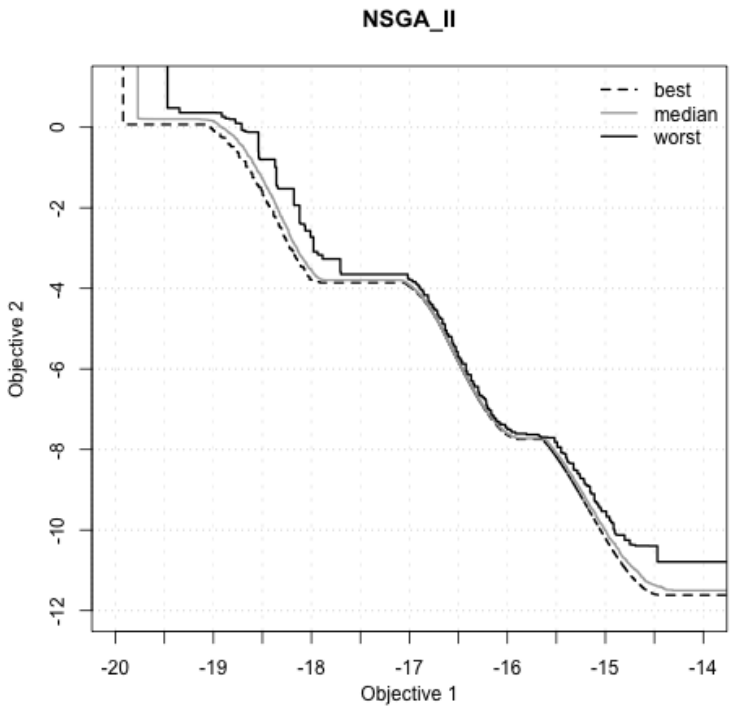

Figure 2.1: Empirical Attainment Function (EAF) example.

2.4 Established multi-objective methods

In this work we consider two categories of multi-objective methods: *a priori* methods, where the preferred tradeoffs between the objectives is defined before the optimization begins, and *a posteriori* methods, those which seek to provide a large array of tradeoff solutions, for decision-making by a concerned party after the optimization process is completed [72]. *A priori* methods typically function on some form of aggre-

gation, where the vector is reduced to a scalar value through a calculation. *A posteriori* methods typically function by identifying the non-dominated set from a population of solutions, and conduct a number of calculations based on this metric and some form of "crowding" metric to determine which solutions are in sparser areas of the objective space. Most *a posteriori* methods are Pareto-based multi-objective evolutionary algorithms (MOEAs).

2.4.1 A priori aggregation methods

Within the class of *a priori* methods for multi-objective problems, there are many different ways to scalarize the objectives into a single evaluation. Three popular solutions are a linear combination, reference point methods, and a dominated hypervolume calculation. These methods have well-studied and theoretically provable properties, and different strengths and weaknesses. There are many different types of *a priori* methods, but the pattern tends to be that they do well in problems with certain properties, and struggle with problems that have a different set of properties.

For robust performance, it is often necessary to normalize the two objective values before conducting additional calculations. If normalization is not done, the contribution of one objective to a scalar calculation can easily dwarf the contribution of the other objectives, simply based on the scale of the numbers involved.

Normalization Objectives may be transformed in many ways to make their contribution to a scalarization function more consistent with respect to the other objectives that

also contribute to the function. One method that is robust to objectives on vastly different scales while also not requiring any input from the system designer is to use the maximum and minimum value for each objective individually and use these limits to normalize by:

$$F_{norm} = \frac{F - F_{min}}{F_{max} - F_{min}} \tag{2.5}$$

where F_{norm} is the normalized vector, F is vector being normalized, and F_{min} and F_{max} are the minimum and maximum values. The exact definitions of F_{min} and F_{max} can vary, and can change both the computational efficiency and the accuracy of the normalization [72].

In this work, when we normalize we use a computationally cheap approximation of this normalization that changes as the optimization proceeds. We define $F_{min} \equiv \hat{u}^{\circ}$. We define $F_{max} \equiv \hat{u}^{\bullet}$. This approximation is extremely simple to calculate, but necessarily means that the normalization will change as the optimization process proceeds. This normalization then takes the form:

$$v^{\text{norm}}(c) = \frac{v(c) - \hat{u}^{\circ}(c)}{\hat{u}^{\bullet}(c) - \hat{u}^{\circ}(c)} \tag{2.6}$$

In this normalization, all components of members of the Pareto approximation P_I^* take on values in the range [0:1], and the normalization changes as P_I^* updates throughout the training process.

Linear combination One simple metric that is sufficient, but not necessary, for finding Pareto optimal points is a linear combination of objectives [72]:

$$LC(w, v) = \sum_{c} w(c)v(c) \tag{2.7}$$

where $LC(w, v)$ is the linear combination evaluation or L_1 norm of vector v, $v(c)$ is the evaluation of vector v on the c^{th} objective, w is the vector of weights, and $w(c)$ is the weight for the c^{th} objective.

This method is computationally cheap when paired with typical optimizers like an evolutionary algorithm [116], but presents three primary problems. First, as the number of objectives increases, the choice of weights can become difficult. Second, this method is incapable of finding certain areas of the Pareto front, those that are non-convex [23, 63, 67, 116] (this drawback has been very well-studied, and additional examples can be found in [2, 7, 22, 45, 47, 71, 82, 86, 94, 114]). Third, incrementing the weights evenly to converge to different parts of the Pareto front does not necessarily lead to evenly-spaced solutions along the front [31].

Reference point methods Reference point methods seek to insert system designer preferences or previous system knowledge into the problem to discover points in the solution space near a specific point, the reference point [36, 78]. This requires that a reference point ref is defined before the optimization process occurs. The evaluation in this case is:

$$RP(w, v) = \sum_{c} w(c) \cdot |v(c) - ref(c)| \tag{2.8}$$

Reference point methods are sufficient to find Pareto optimal solutions if the reference point is chosen in an unattainable portion of the objective space [42]. The solutions generated will tend to be as close to the reference point as possible, and therefore will not spread across the entire Pareto front.

Dominated hypervolume Another metric that does not have the same problems with concave areas of the Pareto front is taking the product of the objective values. This measures the amount of hypervolume that is dominated by a particular solution (which equates to the dominated area in two objectives, or volume in three objectives). It also requires the use of a reference point, which indirectly affects the types of solutions that are attained. This reference point must be dominated by the entirety of the Pareto front to avoid any negative $v(c)$ values [13]. Any negative $v(c)$ values can result in a positive or negative evaluation, depending on the quantity of negative $v(c)$ values, which creates an unstable evaluation. In contrast to reference point methods, the reference point in a hypervolume calculation is not typically manipulated to change the types of solutions that are generated.

The weighted dominated hypervolume calculate takes the form:

$$HV(w, v) = \prod_{c}(ref(c) - v(c))^{w(c)} \tag{2.9}$$

Measuring the dominated hypervolume tends to produce solutions which are roughly equally spaced away from the reference point on each objective.

2.4.2 Multi-objective evolutionary algorithms (MOEAs)

MOEAs are a form of evolutionary algorithm that are designed specifically to work with multiple simultaneous objectives. There are many MOEAs, which each have their own mechanisms for the selection process, by which the population is first reduced in size, and then replenished [23, 103]. Many modern successful MOEAs are Pareto-based methods, where the dominance relations between each of the members of the population are used to develop the surface on which their fitness will be measured. This creates a constantly changing fitness surface that, when designed properly, encourages solutions to tend toward the Pareto frontier, and to spread out along this Pareto frontier.

Two of the most successful and popular MOEAs are the Nondominated Sorting Genetic Algorithm-II, (NSGA-II) and the Strength-Pareto Evolutionary Algorithm-2 (SPEA2), which we discuss in the following sections.

2.4.3 NSGA-II

NSGA-II functions on a two-stage sorting operator. For each point it calculates a "non-domination rank" based on the points which dominate it, and a "crowding distance", based on its proximity to other points with the same non-domination rank. Points are sorted first by non-domination rank, and secondarily by crowding distance [34].

It has been used in a wide variety of applications including chemical reaction engineering [80], facial recognition [100], HIV therapy [55], mechanical design [35], electrical power planning and dispatching [62, 91], rain water reuse [66], reservoir manage-

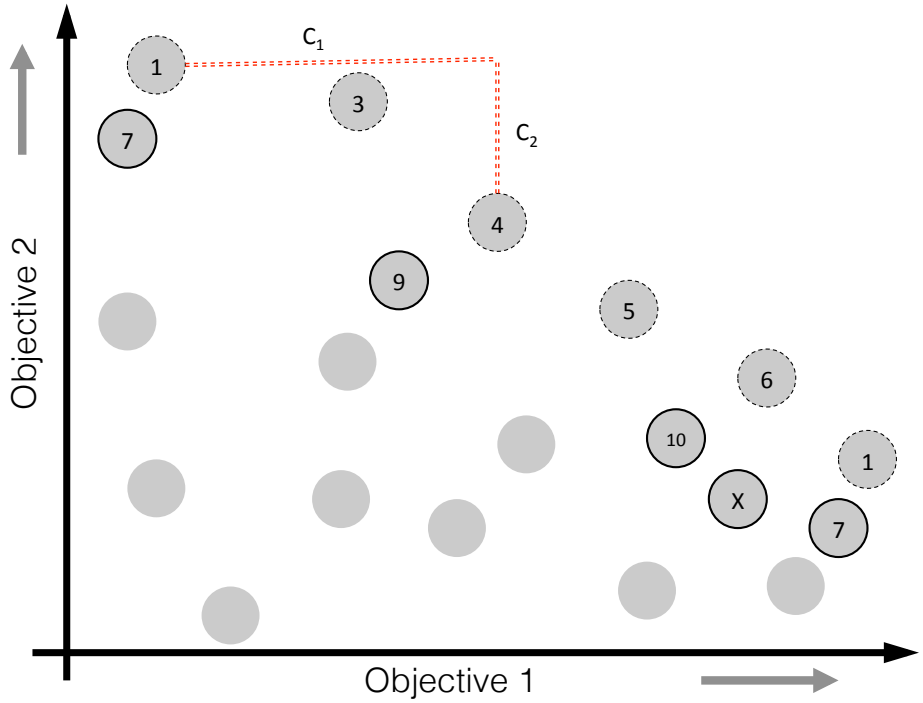

Figure 2.2: Illustration of the NSGA-II process for $N = 20$ points on a maximization problem. NSGA-II ranks these points in the order denoted by the number. The non-dominated fronts are identified in turn (first non-dominated front has dotted outline; second has solid outline). Crowding distance is calculated as the total L_1 distance between the two neighboring points on the front: e.g. Point 3 has a crowding distance of $c_1 + c_2$. X is not selected because the population is full, and its crowding distance is low. Unlabeled points are not selected because they have a high non-domination rank.

ment [93], telecommunication [60], and water distribution [8].

NSGA-II assigns a fitness evaluation to each member of its population of solutions in three sequential steps (See Figure 2.2):

1. At each step, the non-dominated front is identified, and these solutions are as-

signed a rank of 1 and temporarily removed from the pool. Then, the non-dominated front is identified from the remaining solutions, and these are assigned a ranking of 2. This repeats until all solutions are ranked [34].

2. It then estimates the local density of the points along each non-dominated front, by taking the total length of the sides of the cuboid (rectangle in two dimensions) anchored on the nearest neighboring points, that contains no other solutions of the same domination rank [34].

3. Solutions are preferentially selected based first upon their domination rank, and then based on the local Pareto Front density near that solution [34].

NSGA-II provides a total ordering of solutions, but does not directly calculate a fitness or *how much* a given solution would be preferred over another. Because developing a multiagent-compatible version of NSGA-II is the central focus of Chapter 4, we defer a complete algorithm to the background section of that chapter for the convenience of the reader.

2.4.4 SPEA2

SPEA2 [132,133] is an evolutionary algorithm which assigns each vector a "strength" equal to the number of vectors in the current population it dominates. Each vector then sums the strengths of all vectors which dominate it, and this forms a raw fitness evaluation. This is altered by a local k-nearest neighbor density calculation, and the best solutions survive.

SPEA2 has been very successful in real-world applications, including air traffic control [56], diesel engine optimization [57], step voltage regulators in power systems [129], stock picking [53], and wind power generation [131].

SPEA2 works as follows:

- The strength S_i of solution i is calculated as the number of solutions which the solution dominates.

$$S_i = size(\{j|\mathbf{u_i} \succeq \mathbf{u_j}\}) \tag{2.10}$$

- Non-dominated solutions are assigned a raw fitness of 0. Solutions that are dominated by any solutions take on a fitness value that is the sum of the strengths of all of those solutions that dominate it.

$$R_i = \sum_{\forall j \succeq i} S_j \tag{2.11}$$

- The local density D_i is estimated as the inverse of the euclidian distance between the solution and the kth nearest neighbor, where k is typically set to the square root of the population size:

$$D_i = \frac{1}{\sigma_i^k + 2} \tag{2.12}$$

  Two is added to the denominator to ensure that the local density is always a positive value less than one.

- The fitness of the solution (which in this algorithm is to be minimized) is the sum

of the local density and raw fitness.

$$F_i = R_i + D_i \tag{2.13}$$

This process sorts solutions first based on their raw fitness (a scalar value), and then by the density metric (a decimal value always less than one). Thus, it provides a total ordering of solutions as well as a true fitness evaluation, though this fitness landscape is very discontinuous, as each solution that dominates a point (or which that point dominates) changes the raw fitness evaluation, which thereby changes the total fitness evaluation drastically, compared to any change that the crowding metric may make).

SPEA2 is highly effective in 2- and 3-objective problems, but is known to suffer performance degradation in higher numbers of objectives [118]. We use SPEA2 in this work as a baseline algorithm with which to compare performance of the algorithms we developed.

Chapter 3 – Multi-Objective Aggregation in Multiagent Systems

As a first step toward integrating multiagent systems with multi-objective problems, this chapter examines the class of multi-objective methods known as *a priori* methods (Section 2.4.1), specifically scalarization. We incorporate credit assignment (Section 2.2.2), and examine the effects that two scalarizations have on the Pareto front approximation (Section 2.3) that is attained: a linear combination of objectives and a hypervolume measure, in a multiagent problem. We use two multiagent domains that have previously been studied as single-objective problems, with a second objective that adds to each, keeping in the spirit of the original problems.

The necessary background for this chapter includes evolutionary algorithms (Section 2.1.2), Pareto optimality, utopia and nadir vectors, (Section 2.3), and normalization and scalarization (Section 2.4.1).

3.1 Scalarization of objectives

Within the class of *a priori* methods for multi-objective problems, there are many different ways to scalarize the objectives into a single reward signal. In this chapter we examine two: a linear combination and a hypervolume calculation. In each case we normalize the objectives to the range [0:1] before combining them in one of two ways:

a linear combination of objectives

$$R_+ = \sum_{c \in C} w_c f_c^{\text{norm}} \tag{3.1}$$

or a hypervolume measure,

$$R_\lambda = \prod_{c \in C} f_c^{\text{norm}} \tag{3.2}$$

where R_+ is the linear combination reward delivered to the reinforcement learner, R_λ is the hypervolume reward delivered to the reinforcement learner, C is the set of all criteria or objectives, and f_c^{norm} is the normalized score on objective c. In each case we give all agents either R_+ or R_λ, but never any combination of the two. The form which f_c^{norm} takes varies depending on the credit assignment schema used, which is discussed in the following section. Other types of scalarizations do exist, like an exponentially weighted set of objectives or distance from a target point, but we limit the scope of this chapter to consider only these two.

3.2 Multiobjective bar problem (MOBP)

The first domain we consider in this work is an extension of the El Farol Bar Problem originally introduced by Arthur [6]. In this extension, a group of agents $\mathbf{A}$ are each assigned a static type m or f and must independently choose to attend one of several bars. There are multiple objectives: first, the agents wish to attend a bar that is not too crowded, and not too empty. Second, the agents wish to attend a bar with an even mixing of agents of type m and f.

The first "capacity" objective for each bar is modeled as a smooth curve that takes on a value of 0 with no agents attending, near 0 with many agents attending, and a maximum at the ideal capacity ψ. This models the enjoyment of the agents (of quantity x_b) attending bar b. The second "mixture" objective for each bar is maximized when $M_b = F_b$, where these are the number of agents of type m and f attending bar b, *regardless of the number of agents at the bar*. Formally:

$$L_{\text{cap}_b} = x_b \cdot e^{\frac{-x_b}{\psi}} \qquad \Big| \qquad L_{\text{mix}_b} = \frac{\min(M_b, F_b)}{(M_b + F_b)W} \tag{3.3}$$

where W is the number of bars available for the agents to choose from. L_{mix_b} evaluates to 0 if the agents are all of the same type, and $0.5/W$ if there is an equal mixture of types. The number of bars, W, is a constant (and therefore does not change the reinforcement learning process), and serves to limit G_{mix} to values in the range [0:0.5] for easier interpretation of results.

The global rewards for each of these objectives are simply the sum of the local rewards across all bars:

$$G_{\text{cap}} = \sum_{b \in B} L_{\text{cap}_b} \qquad \Big| \qquad G_{\text{mix}} = \sum_{b \in B} L_{\text{mix}_b} \tag{3.4}$$

And the Difference rewards for each are calculated by Equation 2.2 as the global reward minus the global reward in a fictional world had agent i never attended any of the bars:

$$D_{\text{cap}_i} = x_{\text{a}} \cdot e^{\frac{-x_{\text{a}}}{\psi}} - (x_{\text{a}} - 1) \cdot e^{\frac{-(x_{\text{a}}-1)}{\psi}} \tag{3.5}$$

$$D_{\text{mix}_i} = \begin{cases} \frac{\min(M_\text{a},F_\text{a})}{(M_\text{a}+F_\text{a})W} - \frac{\min((M_\text{a}-1),F_\text{a})}{(M_\text{a}+F_\text{a}-1)W} & : i \in m \\ \\ \frac{\min(M_\text{a},F_\text{a})}{(M_\text{a}+F_\text{a})W} - \frac{\min(M_\text{a},(F_\text{a}-1))}{(M_\text{a}+F_\text{a}-1)W} & : i \in f \end{cases} \tag{3.6}$$

where x_a is the attendance in the bar attended by agent i, and M_a and F_a are the number of agents of types m or f respectively that attended the same bar as agent i. D_{mix_i} depends on the type of the agent; the second term represents the system with agent i removed from bar b.

Procedure The procedure for running the MOBP is simple. Each agent simultaneously selects a bar to attend based on no sensory information. The local rewards L_{cap_b} and L_{mix_b} are calculated for each bar b. Then the global rewards G_cap and G_mix are calculated. Finally, D_{cap_i} and D_{mix_i} are calculated for each agent i. Once these are calculated, the selected reward type (local, global, or difference) is normalized and put through Equation 3.1 or 3.2 depending on the desired scalarization. The result is then provided to the agent as the reward R, calculated with a value of $\gamma = 0$, because the problem is only a single step.

Tradeoffs and independence of objectives We take measures to prevent a trivial solution for either objective, or a single dominating solution:

- G_cap: There are many more agents (100) than capacity across all bars (a capacity of 5 for 7 bars).
- G_mix: Agent types are 70% type m, 30% type f.

- *Tradeoff*: L_{cap_i} is maximized at 5 agents; L_{mix_i} is maximized only when an even number of agents attend a bar.
- *Tradeoff*: A maximum G_{mix} case involves many bars with one agent of each type and the rest attending a single bar, which conflicts with G_{cap}.

We calculate the coefficient of determination (R^2) value for the correlation between the two objectives across 10^6 random Monte Carlo trials using a linear, exponential, and polynomial fit. The maximum value was the linear fit at 0.0034, which reinforces that the objectives are distinct, though they are coupled through the actions of the agents.

3.2.1 MOBP results

To exhibit the benefits of Difference rewards in multi-objective problems, we examine 4 types of results:

- Average system performance on both system objectives (Figure 3.1)
- Dominance and Pareto front approximation (Figure 3.2)
- Impact of training time (Figure 3.2)
- Robustness to disturbances (Figure 3.3)

Simulation information We execute 30 statistical runs of the MOBP for seven independent experiments: training all agents on each structure-scalarization combination in turn (D+, Dλ, G+, Gλ, L+, Lλ), and on a random policy (rand).

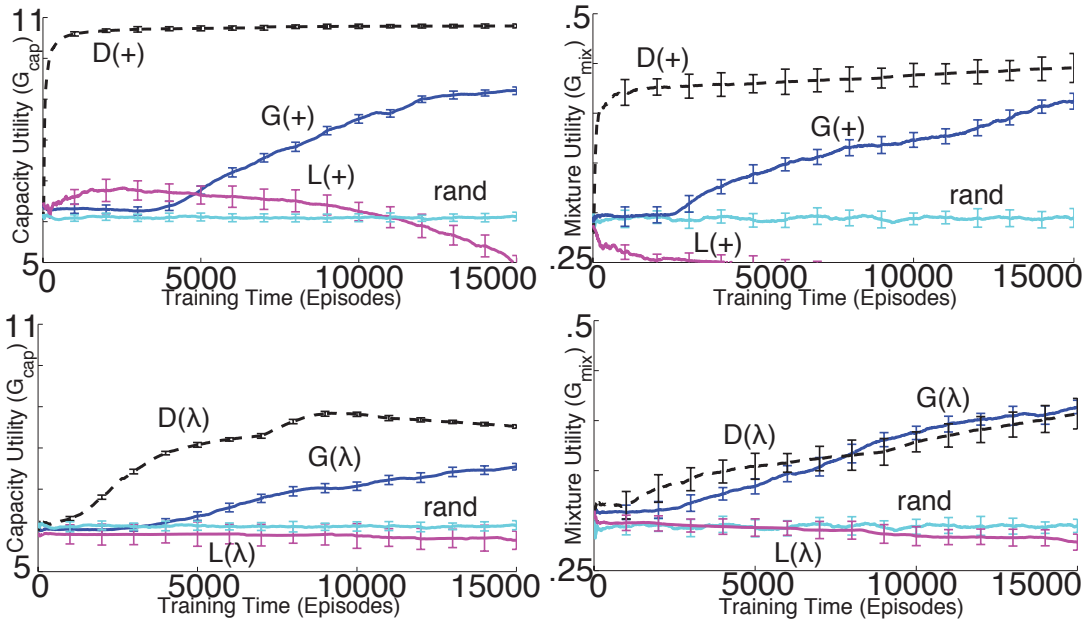

Figure 3.1: Performance on G_{cap} (left) and G_{mix} (right), for agents trained on the linear scalarization (+,top) and hypervolume calculation (λ,bottom) of the three reward structures (D,G,L) and the random baseline (rand). Each of these objectives is to be maximized.

Each agent selects an action using an ϵ-greedy mechanism, with an initial $\epsilon = 0.05$ for local and difference rewards, and $\epsilon = 0.1$ for global rewards[1] (both multiplied by a factor of 0.999 every episode to reduce exploration), with a learning rate $\alpha = 0.10$.

We performed a full sweep through w_c values, but due to the large effect each agent has on the overall system performance near the Pareto front, we found that an even weight, combined with the natural exploration, resulted in a spread of solutions discovered along the Pareto front.

In Fig. 3.1, a 100-episode moving average (across 30 statistical runs) of system

[1]These values were chosen through a parameter sweep to create the best performance for each reward, though the results are not very sensitive to ϵ or α values.

performance was used. Error bars report the error in the mean, calculated as $\frac{\sigma}{\sqrt{N}}$, where N is the number of statistical trials. We identify the Pareto front approximation for each structure-scalarization combination (e.g. "Hypervolume of Global Reward", Gλ), across all 30 statistical runs, and aggregate these into a single non-dominated set, for clarity [48].

3.2.2 Average performance on system objectives

It is informative to look at the performance of the system on each objective individually (Figure 3.1), as this performance drives the behavior of the non-dominated set. For both the linear combination and hypervolume scalarization, the local reward (L+,Lλ) performs poorly; the agents work at cross-purposes, undermining each other's efforts by all trying to attend low-attendance days. This leads to low performance, and will never lead to good system behavior, even with an extreme amount of training time. The global reward (G+,Gλ) does learn, slowly. For the hypervolume scalarization, Dλ increases system performance at a slightly higher rate than Gλ. The linear combination of difference rewards, D+, performs at a very high level very quickly, and reaches near its final performance after only 1500 episodes.

3.2.3 Pareto front approximation and training time

In addition to performing well on the individual objectives, solutions produced by D+ or Dλ produce superior Pareto front approximations compared to the global and

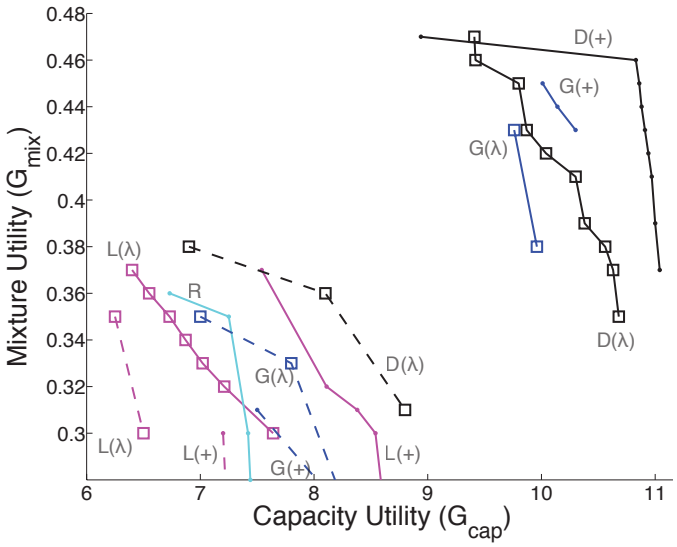

Figure 3.2: The set of non-dominated episodes created over the entire training process through using hypervolume (λ) or a linear combination ($+$); dotted lines show the Pareto front approximation after 1500 learning episodes; solid lines, the Pareto front approximation after 15,000 episodes. Agents trained on D(+) peak in performance before 1500 episodes, so both D(+) Pareto front approximations are identical.

local rewards with the same scalarization. The Pareto front approximations are shown in Figure 3.2. In fact, every solution produced by the local or global rewards is dominated by a solution produced by the difference reward.

The dotted lines in Figure 3.2 represent the Pareto front approximation produced in the first 1500 episodes (10% of the training). In all cases the Pareto front approximation improve between 1500 and 15000 episodes, except D+, which has already produced its best episodes (dominating all other credit assignment/scalarization combinations). Solutions produced by Dλ dominate the solutions produced by other methods in the same time, except D+.

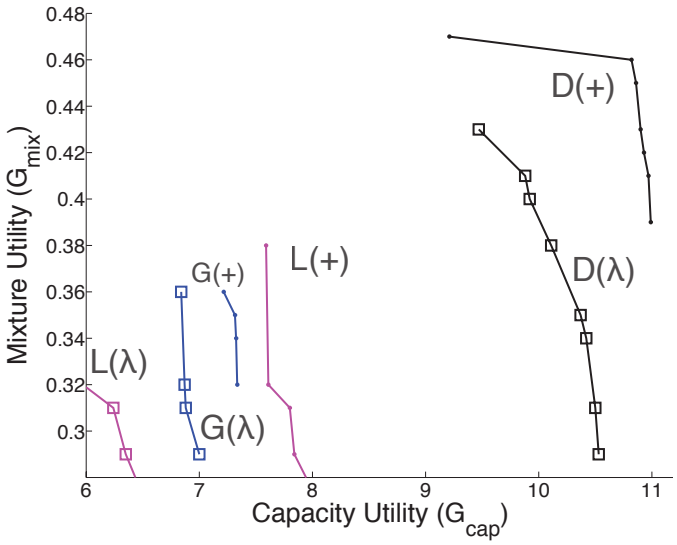

Figure 3.3: The Pareto front approximation produced in the 5000 training episodes after the conversion of 20% of agents to "selfish" behavior. Compare with solid lines in Figure 3.2: D+ and Dλ recover well; G+ and Gλ suffer a disastrous drop in performance.

3.2.4 Robustness to disturbances

To model outside disturbances, after 15000 episodes 14 agents of type m and 6 of type f "fail". They have their Q-table reset to zero values and continue learning using the local reward policy regardless of the learning signal they were using previously (acting selfishly). The remaining 80 agents continue learning using the same signal they were using previously. Additional exploration was found to be necessary in this case, so we reset ϵ to initial values. All agents continue the learning process as before.

Figure 3.3 shows that D+ maintains its dominant Pareto front approximation. G+ and Gλ are affected catastrophically by the selfish agents, while D+ loses 98.4% less dominated hypervolume. Dλ only loses performance on G_{mix}.

3.3 Collective transport domain

We additionally performed experiments in a collective transport domain, modeled after [95], in which a team of small robots must cooperate to transport an item (which we also refer to as a body or load) across a surface in much the same way that ants transport objects.

We formulate this as a time-extended, stateful reinforcement learning problem in which the robot agents try to (i) collectively transport the object as quickly as possible to the goal, while (ii) expending minimum effort.

Each robot is given discretized state information about the load's position and velocity, and is allowed to take one of nine actions, applying a force to the load in a cardinal direction (N,S,E,W), an intermediate direction (NE,SE,SW,NW), or no force. The robots are assumed to be attached to the object, and receive discretized state information based on the object's current location and speed.

The body's acceleration (acc), velocity (vel), position (pos) at time t are found with particle kinetics:

$$acc(t) = \sum_{i \in \mathbf{A}} [F_{\text{x}}(i)\hat{i}] + \sum_{i \in \mathbf{A}} [F_{\text{y}}(i)\hat{j}] - F_f \hat{f} \tag{3.7}$$

$$vel(t) = vel(t-1) + acc(t) \cdot t_{\text{step}} \tag{3.8}$$

$$pos(t) = pos(t-1) + vel(t) \cdot t_{\text{step}} \tag{3.9}$$

where $\mathbf{A}$ is the set of all agents, $F_{\text{x}}(i)$ is the force applied by agent i in the $\hat{i}$ direction, $F_{\text{y}}(i)$ is the force applied by agent i in the $\hat{j}$ direction, and F_f is the force of friction, which acts in the $\hat{f}$ direction, which points opposite the direction of motion of the body.

We omit mass from this calculation of Newton's second law because we assume the mass of the body and transporting robots to be 1 unit total. In this context a local reward loses meaning as all agents are collectively acting to move the same object, so we only look at global and difference reward in this case.

The first objective (proximity) is to move the load close to the goal as quickly as possible. This takes the form:

$$G_{\text{prox}}(t) = -Tdist(t) \tag{3.10}$$

$$D_{\text{prox}_i}(t) = -Tdist(t) + Tdist_{-i}(t) \tag{3.11}$$

where $Tdist()$ is a function that returns the body's Euclidian distance from the target at time t, and $Tdist_{-i}()$ returns the distance from the target if agent i took no action during timestep t.

The second objective is to minimize the effort exerted by the team to move the load to the desired target location:

$$G_{\text{effort}}(t) = \sum_{i \in \mathbf{A}} [1 - E_{i,t}] \tag{3.12}$$

$$D_{\text{effort}_i}(t) = 1 - E_{i,t} \tag{3.13}$$

where $E_{i,t}$ is 1 if the agent applied a force to the object at time t, and 0 if the agent did not apply a force.

We perform a Q-update at every time step. To visualize the performance, we aggregate these into one point for each time the load reaches the goal state. For the purpose of

learning, however, we use the distance to the goal state after each time step, as this provides a smoother gradient for learning [64]. The process for conducting this experiment is described in Algorithm 3.1. For each credit assignment schema and scalarization combination, step 19 would use the proper evaluation (one of G or D_i), and use the desired scalarization from Equation 3.1 or 3.2.

In this domain the two objectives are in conflict with one another: minimizing the time to deliver the load will maximize the effort required, and minimizing effort will lead to a longer time.

Algorithm 3.1: Collective Transport Domain using Difference Reward of Dominated Hypervolume (Dλ)

initialize Q-values to zero: $Q(s,a) = 0 \; \forall \; s, a$;
initialize body position to starting location ;
initialize velocity and acceleration to $\vec{0}$;
foreach $timestep = 1 \rightarrow max\_timesteps$ **do**
    foreach $i = 1 \rightarrow total\_agents$ **do**
        choose an action with ϵ-greedy selection:
        $\{$none,N,NE,E,SE,S,SW,W,NW$\}$;
        add force contribution to body ($F_{\mathrm{x}}(i)$,$F_{\mathrm{y}}(i)$);
    end
    evaluate body acceleration (Equation 3.7);
    evaluate body velocity (Equation 3.8);
    evaluate body position (Equation 3.9);
    if *body position is out of bounds* **then**
        set body position to nearest in-bounds position;
        set body velocity to $\vec{0}$;
    end
    evaluate global reward (Equations 3.10, 3.12);
    for $i = 1 \rightarrow total\_agents$ **do**
        evaluate difference rewards (Equations 3.11, 3.13);
        evaluate $R \leftarrow R_{\lambda}$ (Equation 3.2);
        update $Q(s,a)$ values ;
    end
    if *body is in goal state* **then**
        set body to starting location;
        set velocity and acceleration to $\vec{0}$;
    end
    reduce ϵ;
end

3.3.1 CTD results

In the collective transport domain, we examine two types of results:

- Dominance and Pareto front approximations (Figure 3.4, Left)
- Impact of training time (Figure 3.4, Right)

Simulation Information We perform 4 different trials following Algorithm 3.1; one each for G+, Gλ, D+, and Dλ. For each, we conduct 30 statistical runs of 5000 time steps for teams of 50 agents attempting to transport a load across a surface with maximum static force of friction $F_f = 8$ units and kinetic force of friction of $F_f = 2$ units. The body's starting state is initialized as $(x, y) = (1, 1)$, with the goal as a square at $\{x_{\text{min}}, x_{\text{max}}, y_{\text{min}}, y_{\text{max}}\} = \{900, 1000, 900, 1000\}$. The boundaries are a larger square at

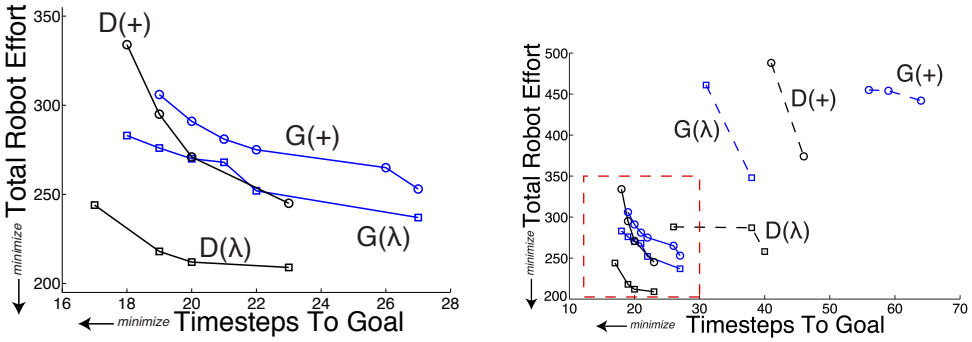

Figure 3.4: (Left) Collective Transport Domain results. D(λ) creates solutions that dominate all other methods (solutions below and to the left are superior in this domain). D+ outperforms G+, and creates an overlapping Pareto front with G(λ). (Right) Dotted lines denote early system performance after 500 time steps. The denoted highlighted area is the range of the figure on the left.

$\{x_{\text{min}}, x_{\text{max}}, y_{\text{min}}, y_{\text{max}}\} = \{0, 1000, 0, 1000\}$. Though the calculations of the body's velocity and position are continuous, we us an approximation via tile coding [102] and discretize into 10 states each for $(x_{\text{vel}}, y_{\text{vel}}, x_{\text{pos}}, y_{\text{pos}})$ creating 10,000 states. In this domain, we find the best performance when we vary the weights for the objectives as a function of learning step, starting by with a value of $\{w_{\text{prox}}, w_{\text{effort}}\} = \{1, 0\}$ changing linearly to $\{0, 1\}$ at the final learning step. This produces policies which do find the goal state, and learn to reduce effort over time. This produces better initial performance and a spread of solutions. Initial weights favoring the effort objective led to policies of inaction, never reaching the goal.

3.3.2 Dominance and Pareto front approximation

Figure 3.4 shows the final Pareto front approximation for each method. The teams of agents trained on the scalarizations of the difference reward (D+, Dλ) outperform their global counterparts in the final produced Pareto front approximation. In this domain, however, the hypervolume calculations (λ) perform better than the linear combinations ($+$). We find nearly equivalent performance between Gλ and D+, suggesting that using the proper multi-objective scalarization is as important as proper multiagent credit assignment. The Dλ result shows that these benefits can be symbiotic.

3.3.3 Impact of training time

We also identify the Pareto front approximation produced by each solution after 10% of the training time in Figure 3.4. Again, the difference reward using the preferable scalarization attains performance close to its final performance very quickly, while the global methods are not as near their final performance values. Dλ dominates all solutions formed by other scalarizations.

Additionally, in this domain we noticed that the performance of the global reward signals was sensitive to the learning parameters, while the difference reward signals were robust to these changes. In additional trials we found the agents trained with the difference reward to be robust to noisy actuators, noisy sensors, failing agents, and unmodeled disturbances (externally applied forces) as well.

3.4 Conclusion

Multiagent systems are a powerful concept for dealing with complex systems. Many multiagent systems are intrinsically multi-objective, but this has received scant attention. In this chapter we explicitly addressed one of the key concerns in multiagent systems — credit assignment — under the conditions of a multi-objective problem. We found that credit assignment is important under multi-objective conditions: our results show (i) a 10x increase in learning speed, (ii) a 98.4% increase in robustness to unmodeled disturbances, and (iii) the production of solutions which dominate all solutions found by a traditional global reward. These results show that proper credit assignment is of

paramount importance in a multiagent multi-objective system. However, the choice of multi-objective algorithm is still extremely important. Difference rewards boosted performance in both domains, for both scalarizations. The gains from credit assignment through difference rewards were independent of the scalarization used and the domain.

Chapter 4 – Adapting NSGA-II for Multiagent Systems

In the previous chapter (Chapter 3), we developed the concept of credit assignment for use in reinforcement learning with a scalarized objective function. We identified that credit assignment is at least as important as the type of multi-objective algorithm used. In this chapter, we introduce credit assignment into the Pareto-based MOEA, NSGA-II (Non-dominated Sorting Genetic Algorithm-II). We first have to modify the internal structure of NSGA-II to provide a real-valued evaluation, before incorporating difference evaluations. We discover that the order of operations in this process is of paramount importance, and identify the reasons that this occurs.

For this chapter, it is therefore necessary to be familiar with cooperative coevolutionary algorithms (Section 2.2.3) , and the concepts of Pareto optimality (Section 2.3), and credit assignment (Section 2.2.2). We use empirical attainment functions (Section 2.3) to compare performance between this modification and a normalized linear combination (Section 2.4.1). Because we do direct modifications to the NSGA-II algorithm, the initial background section on NSGA-II (Section 2.4.3) is also recommended for an intuition on how NSGA-II functions. In this chapter we also offer a complete algorithm section describing the mechanisms employed by NSGA-II.

We compare the performance of the linear combination and NSGA-II using a global evaluation and the difference evaluation — in various combinations and with various orders of operation — on a multi-objective rover domain, in which different points of

interest have one of two different types of data to gather, creating the two distinct and competing objectives.

4.1 Contributions

NSGA-II is an accomplished algorithm for solving multi-objective problems in widely varying domains, including facial recognition [100], HIV therapy [55], rain water reuse [66], and telecommunications [60]. Though it has been successful in such domains, NSGA-II is explicitly created as a centralized solution generator, and does not incorporate multi-agent concepts.

Difference evaluations have been studied in a wide variety of single-objective domains, including data routing over a telecommunication network [113], congestion games [40], multiple robot coordination [3], and air traffic control [108]. Though it has been successful in such domains, it has only been studied briefly in the context of multiple objectives [125]. As many interesting problems involve both multiple agents and multiple objectives, it is critical to develop approaches tuned to the intersection of these two paradigms. In this chapter we reformulate NSGA-II such that it may incorporate difference evaluations more readily, and study the benefits offered by NSGA-II using a global or difference evaluation compared to a linear combination of objectives using a global or difference evaluation.

The key contributions of this chapter are:

- derive real-valued NSGA-II, allowing additional operations to shape the fitness for each agent in a multiagent system.

- derive a modified difference evaluation function that is compatible with the NSGA-II framework and produces good solution quality
- provide an intuitive understanding of why naive, but seemingly reasonable combinations of NSGA-II and difference evaluations lead to poor solutions.

To demonstrate both the subtle interaction between the multiagent and multi-objective paradigms, we use the conceptually simple, but operationally complex domain of multiple robots exploring an unknown landscape as a running example. In this problem, multiple robots must collect as much data of various types (seismographic, radiological, atmospheric, electromagnetic, or soil analysis, to name a few possibilities) as they can, but the points of interest (POIs) that they must observe may only contain one type of data. As a team, they must maximize the amount of each type of data that they observe. When there are no obvious, linear tradeoffs between the types of data, this problem becomes a multiagent, multi-objective optimization problem, providing a rich environment to develop and test the proposed approach.

This chapter is organized as follows: Section 4.2 describes the necessary specific background for this chapter. Section 4.3 describes the proposed algorithms. Section 4.4 describes the multi-objective continuous rover exploration problem used in this chapter. Section 4.5 shows the results of our experiments. Finally, Section 4.6 concludes the chapter.

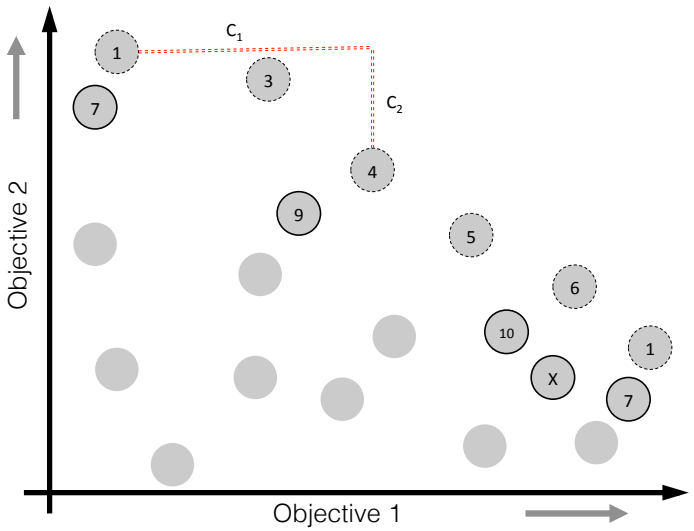

Figure 4.1: Illustration of the NSGA-II process for $N = 20$ points. NSGA-II ranks these points in the order denoted by the number. The non-dominated fronts are identified in turn (first non-dominated front has dotted outline; second has solid outline). Crowding distance is calculated as the total L_1 distance between the two neighboring points on the front: e.g. Point 3 has a crowding distance of $c_1 + c_2$. X is not selected because the population is full, and its crowding distance is low. Unlabeled points are not selected because they have a high non-domination rank.

4.2 NSGA-II Algorithm

The Elitist Non-Dominated Sorting Genetic Algorithm, NSGA-II [34], is one of the standard benchmark multi-objective methods by which many other methods measure themselves. It is an evolutionary algorithm that works in three sequential steps:

1. At each step, the non-dominated front is identified, and these solutions are assigned a rank of 1 and temporarily removed from the pool. Then, the non-

dominated front is identified from the remaining solutions, and these are assigned a ranking of 2. This repeats until all solutions are ranked.

2. It then estimates the local density of the points along the non-dominated front, by taking the total length of the sides of the cuboid (rectangle in two dimensions) anchored on the nearest neighboring points, that contains no other solutions of the same domination rank.

3. Solutions are preferentially selected based first upon their domination rank, and then based on the local Pareto Front density near that solution.

Because it functions solely based on pairwise comparisons between the population members, and there is no absolute gradient to the function, a larger population size gives a finer-grained feedback signal of the fitness landscape. NSGA-II does not explicitly give a decimal fitness value for each population member.

Algorithm 4.1: NSGA-II Base Algorithm

Data: input set of solutions S_t

$\mathcal{F} \leftarrow$ `fast-nondominated-sort`(S_t) // [34]

while $|S_{t+1} < N|$ **do**

  `crowding-distance-assignment`$(\mathcal{F}_i)$ // [34]

  $S_{t+1} \leftarrow S_{t+1} \cup \mathcal{F}_i$

  $i \leftarrow i + 1$

end

`sort`$(S_{t+1}, \geqslant_n)$ // [34]

$S_{t+1} \leftarrow S_{t+1}[0 : N/2]$ // select first $N/2$ members

$S_{t+1} \leftarrow$ `expand-pop`(S_{t+1}) // [34]

Result: S_{t+1}

Each of the methods in Algorithm 4.1 are defined in [34], but we give the intuition for them here, and illustrate them in Figure 4.1.

- $\mathcal{F} \leftarrow$ `fast-nondominated-sort`(S_t): Takes S_t, and sorts it into non-dominated fronts, $\mathcal{F}_i$. After a non-dominated front $\mathcal{F}_i$ is identified, it is temporarily removed from S_t so that the next front may be identified.

- `crowding-distance-assignment`$(\mathcal{F}_i)$: For one non-dominated front, $\mathcal{F}_i$, calculates the crowding distance (L_1 distance between immediate neighbors) for each point in that front. Extreme points are assigned infinite crowding distance.

- `sort`$(S_{t+1}, \geqslant_n)$: Sorts S_{t+1} first on non-dominated ranking, and breaks ties with crowding distance.

- $S_{t+1} \leftarrow$ `expand-pop`(S_{t+1}): Creates $N/2$ new population members through the mutation operator, adds them to S_{t+1}.

4.2.1 Multiagent Background

To solve larger, more complex problems, two routes exist: first, a single agent can be made to solve the problem, at the cost of more and more computational time or power; second, multiple simpler agents can be made to cooperate to solve the problem based on locally available information, with the solution arising from the interaction between the agents. The second option offers many benefits from an exploration or resource gathering standpoint. If one rover out of a team ventures into a dangerous area, the loss

of that rover does not cripple the team, allowing for potentially more dangerous missions to be considered.

Beyond the disposability of the individual team members, there are other benefits to this team-based approach. Savings can be realized in construction, as each rover can be designed with parts from a lower-cost parts portion of the reliability curve. Similar savings are available in the design process, as a new team of rovers can be formed with some members that have been previously designed.

In addition, a team of rovers can have capabilities that a single monolithic rover cannot, like having presence in multiple locations at once, which is incredibly useful for planetary exploration. Ephemeral events can be simultaneously observed from separate locations [46], even from the ground and from orbit simultaneously [20], which can make interpreting the situation significantly easier. Construction tasks that might be impossible for a single rover with limited degrees of freedom become much easier. Teams can survey areas separated by impassible terrain and share long-range communication resources [19].

However, with multiple team members, the question quickly becomes how the team members may coordinate such that their actions lead to desirable system-level performance. The concept of *adaptive agents* allows the team members to use an algorithm that searches through the space of available policies, biasing the search to more favorable areas of the search space. This work is addressed in Section 2.1 and 2.2.3.

The interactions between the team members may still result in agents working at cross-purposes, where two agents unintentionally perform redundant tasks, or worse, undo the efforts of another rover. To combat this phenomenon, the study of *shaping*

seeks to create functions that, when maximized on an agent-level, will lead to good performance on the system-level.

When conducted properly, such shaping techniques can promote coordination and improve the robustness of the learning process [73, 104]. One shaping technique that has been successful in a wide variety of domains is the difference evaluation function, which we discuss in depth in Section 2.2.2. In this chapter we offer a brief overview in Section 4.2.3

4.2.2 Policy Search

A key issue in multiagent problems is that the policies which the agents should follow are not always obvious, and "optimality" is almost impossible to achieve. Thus, there are a number of methods that exist for intelligently searching through the parameter space of agent policies (analogous to Ω in the multi-objective problem formulation) that have been widely successful. In this chapter we use cooperative coevolutionary algorithms, a multiagent implementation of evolutionary algorithms.

4.2.3 Difference Evaluation Functions

The **global evaluation function** (G) is the system performance of the team as a whole. This encourages the agent to act in the system's interest, but includes a large amount of noise from other agents acting simultaneously. An agent's own contribution to the global reward may be dwarfed by the contribution of many other agents, resulting

in a low "signal to noise ratio".

The **difference evaluation function** (D_r) is a shaped reward signal that helps an agent learn the consequences of its actions on the system objective by removing a large amount of the noise created by the actions of other agents in the system [119]. It is defined as:

$$D_r(z) = G(z) - G(z_{-r}) \tag{4.1}$$

where $G(z)$ is the global system performance for the system considering the joint state-action z, and $G(z_{-r})$ is $G(z)$ for a theoretical system without the contribution of agent r. Any action taken to increase D_r simultaneously increases G, while agent r's impact on its own reward is much higher than its relative impact on G [119].

For difference evaluations to be effective, it is important that a significant change in an agent's actions leads to a change in the global evaluation. In the event that a change in action results in no change in global evaluation, the difference evaluation does not offer a mechanism to prefer one action over the other.

4.2.4 Gap in current understanding

Previous work in multi-objective multiagent systems has shown that credit assignment is very important in policy iteration reinforcement learning using aggregation functions [125]. A technique has not been developed for incorporating credit assignment into Pareto-based MOEAs for multiagent systems. There are a number of questions that have not been answered, that this chapter addresses:

- does NSGA-II possess properties that naturally subsume credit assignment?

- how does the order of the multiagent credit assignment and multi-objective evaluation affect system performance?
- Is system performance affected by performing a centralized NSGA-II calculation (thereby increasing the relative population size)?

In this work we create experiments to answer each of these questions in turn (Section 4.5).

4.3 Algorithms

This section describes the novel contributions of this work, in which we derive NSGA-II for multiagent systems. We first establish a real-valued fitness assignment mechanism which gives equivalent total ordering of solutions to NSGA-II. Then, because it is not immediately apparent how multiagent systems and NSGA-II should best be combined, we do so in a number of different ways.

4.3.1 Real Valued NSGA-II

In order to be able to take a difference evaluation on NSGA-II calculations, it is important that there be a real-valued fitness associated with each population member. While NSGA-II does create a complete ordering of individuals, it does not explicitly create a real scalar value that can be compared to show *how much* one solution is preferred over another.

Other MOEAs do perform this kind of real-valued total ordering. Here, we adapt the

technique from SPEA2 [132], which performs a real-valued total ordering, to NSGA-II's fitness calculation. The result is Algorithm 4.2, which uses the same functions from [34] as Algorithm 4.1, but produces a real-valued total ranking that is precisely equivalent to the ranking produced by Algorithm 4.1.

Solutions on the first non-dominated front will take on R values in the range [0:1); on the second non-dominated front, [1:2); and so forth. After Algorithm 4.2, solutions can be sorted by R values, and a purely greedy selection (minimizing $R_{i,j}$) always selects the same solutions as NSGA-II.

4.3.2 Multiagent NSGA-II

In creating a multiagent implementation of NSGA-II, there are many options to consider:

Algorithm 4.2: Real-Valued NSGA-II Fitness Assignment: $R \leftarrow$ `RVNSGA-II`(V_t)

Data: Set of solutions V_t

1 $\mathcal{F} \leftarrow$ `fast-nondominated-sort`(V_t) ;
2 **foreach** $i \in \mathcal{F}$ **do**
3 **foreach** $j \in \mathcal{F}_i$ **do**
4 $R_{i,j} = i - \mathcal{C}_{i,j} / \max(\mathcal{C}_i)$;
5 **end**
6 **end**

Result: R

7 †: Since we divide by maximum crowding distance in line 4, extreme points are given a large, non-infinite $\mathcal{C}_i$ value;

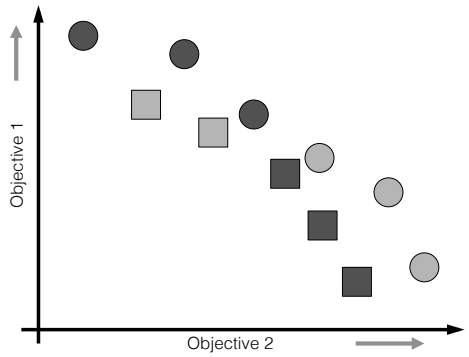

Figure 4.2: Two agents' policy populations (light, dark) that lead to different NSGA-II fitnesses depending on centralization. If centralized, the circles attain a Pareto ranking of 1, and will be preferentially selected over the squares. If decentralized, the squares will have a Pareto ranking of 1 also, and different members will be maintained from each population.

- **Centralized vs. Decentralized**: In the case of the decentralized implementation, we have each agent run an NSGA-II algorithm independent of each other agent. In the case of a centralized implementation, we run one single NSGA-II algorithm for fitness assignment, but still select policies to survive as a cooperative co-evolutionary algorithm. This can make a difference in which policies survive from each population, as is shown in Figure 4.2.

- **Global vs. Difference**: Which evaluation that we use in tandem with the NSGA-II algorithm. In the case of the global evaluation, all agents in one trial receive the same team evaluation. In the case of the difference evaluation, each agent receives its own difference evaluation (Equation 4.1).

- **Prior vs. Post**: When NSGA-II is conducted, with respect to the difference evaluation. In the case of prior, the evaluation the agent receives is a "difference of NSGA-II-based evaluations". In post, the agent receives an "NSGA-II evaluation on difference evaluations".

This creates a total of 8 different NSGA-II evaluations that can be performed. In this section we provide algorithms for two formulations, and the others may be constructed similarly. In these algorithms, a pair of lower-case indices (a, n) refers to an individual agent a's n^{th} policy. Capital indices refer to a whole population: (a, N) referring to agent a's entire population of policies; (A, N) referring to all agents' populations of policies. Last, an index of $-a$ refers to a value not considering the contribution of agent a.

Centralized Global NSGA-II Algorithm 4.3 describes a centralized NSGA-II calculation on the global evaluation. The function `simulate-all`($S_{t,A,N}$) conducts a series of trials, one for every $n \in N$, with the different agents randomly drawing policies (no replacement) for each simulation, so that each policy receives one global fitness vector $\vec{G}$ (of length k, the number of objectives), and one counterfactual vector, $\vec{G}_{-a}$ (of length k). The function `select`$(S_{t,a,N}, R_a)$ then selects survivors for $t+1$ from the population $S_{t,a,N}$ based on their real-valued fitness $R_{a,N}$.

Decentralized Difference NSGA-II, Prior Algorithm 4.4 describes a decentralized difference of NSGA-II values calculation, with NSGA-II fitness assignment happening before difference evaluation. The functions used in Algorithm 4.4 are the same as those

Algorithm 4.3: Centralized Global NSGA-II Fitness Assignment

Data: Evolutionary time t
Data: Populations of solutions $S_{t,a,n} \forall a \in A, \forall n \in N$
$\{\vec{\mathbf{G}}_{A,N}, \vec{\mathbf{G}}^{-a}_{A,N}\} \leftarrow$ `simulate-all`$(S_{t,A,N})$;
$R_{A,N} \leftarrow$ `RVNSGA-II`$(\vec{\mathbf{G}}_{A,N})$ // Alg. 4.2 ;
$S_{t+1,A,N} \leftarrow$ `select`$(S_{t,A,N}, D_{A,N})$;

Algorithm 4.4: Decentralized Difference NSGA-II, Prior Fitness Assignment

Data: Evolutionary time t
Data: Populations of solutions $S_{t,a,n} \forall a \in A, \forall n \in N$
$R_{a,N} \leftarrow$ `RVNSGA-II`$(\vec{\mathbf{G}}_{a,N})$ // Alg. 4.2 ;
$R'_{a,N} \leftarrow$ `RVNSGA-II`$(\vec{\mathbf{G}}^{-a}_{a,N})$ // Alg. 4.2;
$D_{a,N} \leftarrow R_{a,N} - R'_{a,N}$ // Eq. 4.1;
$S_{t+1,a,N} \leftarrow$ `select`$(S_{t,a,N}, D_{a,N})$;

used within Algorithm 4.3.

Computational Complexity The base NSGA-II algorithm has a known computational complexity that is in the average case $\mathcal{O}(kN^2)$, where k is the number of objectives and N is the population size.

Calculating the difference evaluation and conducting the NSGA-II calculation before or after difference rewards are calculated make no difference in the overall complexity. Though they can change the computation time required, they do not scale differently than the base NSGA-II algorithm.

Decentralization leads to no change in the computational complexity, as each agent can conduct their search in parallel. Centralized calculations use the joint population of size $A \cdot N$, resulting in a complexity of $\mathcal{O}(kA^2N^2)$, a factor of A^2 worse.

4.4 Experimental Domain

We perform a series of experiments with the different algorithms in a continuous rover domain. In this domain, a team of 10 rovers must work together to observe a set of 50 heterogeneous points of interest (POIs) on a 100 x 100 Euclidian plane, which have vector-valued importance $\vec{V}$ of length $k = 2$ objectives. Each of the elements of this vector represents a type of scientific data that the rovers may collect from a POI, which acts as an objective. We desire to maximize the system (global) performance vector $\vec{G}$.

Rover Motion Each rover policy selects a series of waypoints (μ_x, μ_y), which are then sent to a low-level planner that determines the path the rover should take to reach that waypoint. After that waypoint is reached (with some variance σ_x and σ_y), the rover takes an observation of any POIs that it can observe, before proceeding to the next waypoint. We assume the rovers can localize en route, so variance from the intended waypoints does not increase with time or distance travelled.

Observations The quality of the observations created by the rovers is a function of the Euclidian distance at which they observe the POI, δ:

$$\delta(x, y) = \max\{\|x - y\|^2, d^2\} \tag{4.2}$$

where x is the location of one object, y is the location of a second object, and d is the minimum observation distance (to prevent singularity, we use $d = 1$). This then allows us to calculate the vector of global evaluations $\vec{G}$ based on rover locations L_R and POI

locations L_P:

$$\vec{G} = \sum_p \frac{\vec{V_p}}{\min_r \delta(L_{P,p}, L_{R,r})} \tag{4.3}$$

where "$\min_r \delta(L_{P,p}, L_{R,r})$" is the minimum distance between any rover and POI p. Applying Equation 4.1 directly to Equation 4.3, we can calculate the difference evaluation vector $\vec{D}_r$ for rover r:

$$\vec{D}_r = \sum_p \frac{\vec{V_p}}{\min_r \delta(L_{P,p}, L_{R,r})} - \sum_p \frac{\vec{V_p}}{\min_{r' \neq r} \delta(L_{P,p}, L_{R,r'})} \tag{4.4}$$

Note, only when rover r is the rover whose observation of a POI is used in the global reward will $\vec{D}_r$ be non-zero.

Cooperative co-evolutionary algorithm We use a coevolutionary algorithm as outlined in Section 2.2.3. For each rover we maintain a population of 100 solutions. We calculate fitness in many different combinations of difference evaluations and NSGA-II calculations, as outlined in Section 4.3.2.

- Initialization: We initialize 100 solution policies for each rover. They are initialized randomly through the space.
- Fitness Calculation: Each policy is simulated with a team of randomly-selected policies from the other rovers. Raw global fitness vectors ($\vec{G}$) and counterfactual vectors ($\vec{G}_{-a}$) are returned and used in various combinations of NSGA-II and difference evaluations, as outlined in Section 4.3.
- Selection: Binary tournament selection [50]. Two policies are randomly selected

with replacement, and the one with the highest fitness is placed in the survivor population, continuing until the population is full.

- Mutation: For all waypoints, we perform the following mutation: $\{\mu_x, \mu_y\} \leftarrow \{\mu_x, \mu_y\} + \{\mathcal{N}(0,5), \mathcal{N}(0,5)\}$, where $\mathcal{N}(\mu, \sigma)$ returns a random number drawn from a normal distribution of mean μ and standard deviation σ.

4.5 Results

For each experiment, we report the empirical attainment function (EAF) of the team's performance over 100 statistical trials [49]. These plots, Figures 4.3–4.9, show the best case attainment surface (points which were dominated at least once in the 100 trials), median attainment surface (points which were dominated in at least 50 of the 100 trials), and worst case attainment surface (points which were dominated in all 100 trials). Note that the best and worst cases are extremes, and comparing medians is the most reliable comparison between different methods.

4.5.1 Global LC aggregation vs. MOEAs

In our first experiment, we examine whether MOEAs provide a boost in performance over a simple linear combination of objectives in a multiagent continuous rover problem. Figures 4.3 and 4.4 show the resulting EAFs. It is clear that the performance is similar, with NSGA-II failing to produce significant gains without considering credit assignment.

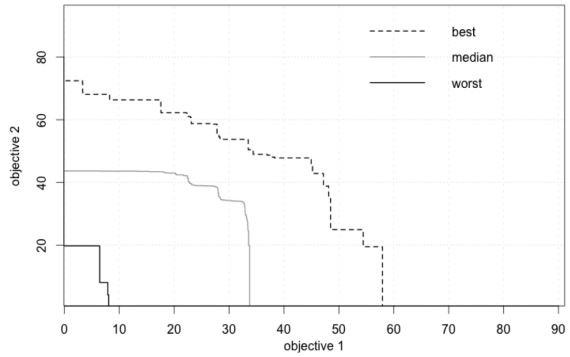

Figure 4.3: Global Linear Combination EAF; Decentralized.

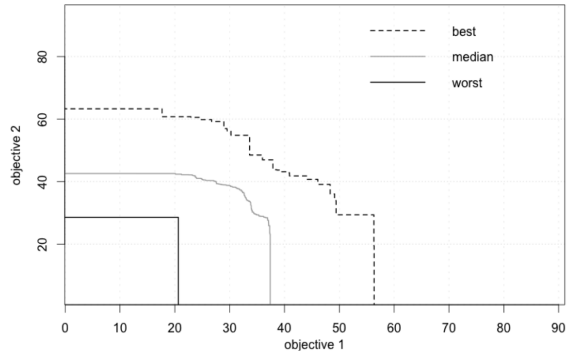

Figure 4.4: Global NSGA-II EAF; Decentralized.

4.5.2 Credit assignment with difference evaluations

Next, consider Figure 4.5. It is readily apparent that the difference evaluation makes a very large difference in system performance, with median performance using the difference evaluation in a linear combination taking on a similar profile to best-case performance compared to either the LC or NSGA-II using a global evaluation.

However, when NSGA-II is used in tandem with difference evaluations (decentralized, before), once again a large boost in performance is seen (Figure 4.6). Be-

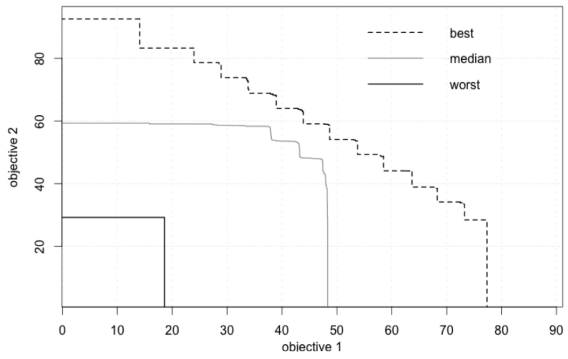

Figure 4.5: Linear Combination of Difference Evaluation EAF; Decentralized

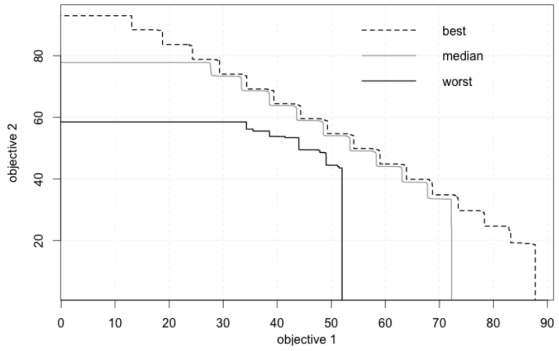

Figure 4.6: NSGA-II Calculation of Difference Evaluation EAF; Decentralized

cause NSGA-II and difference evaluations are solving fundamentally different problems (NSGA-II assigns value to each objective based on others within the same population, while difference evaluations assign value to each agent based on others within the same simulation), their benefits are complementary and may be combined. In this case we see worst case performance that is comparable to median performance with the linear combination.

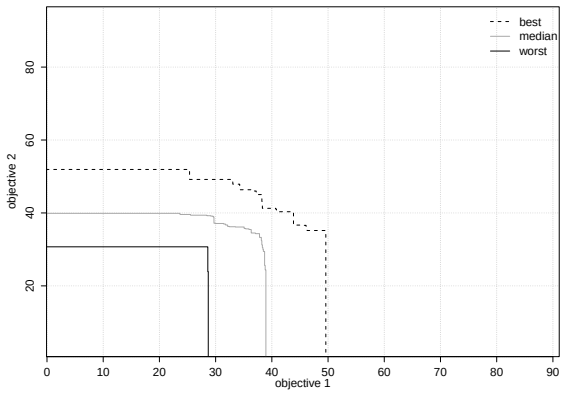

Figure 4.7: Global NSGA-II EAF; Centralized

4.5.3 Centralization of Fitness Calculation

While centralization of the NSGA-II process provides a less discretized version of the fitness space, it also produces a noticeable slowdown over the decentralized version (as an additional time complexity factor of A^2, where A is the number of agents is included in the fitness assignment steps).

Additionally, it tends to force the global system performance toward the center tradeoff solutions of the Pareto front (Figures 4.7–4.8). This is because of the phenomenon illustrated in Figure 4.2, in which each agent's population will tend toward extreme solutions. On average, then, the tradeoff solutions will be developed, at the cost of solutions near the single-objective optimals.

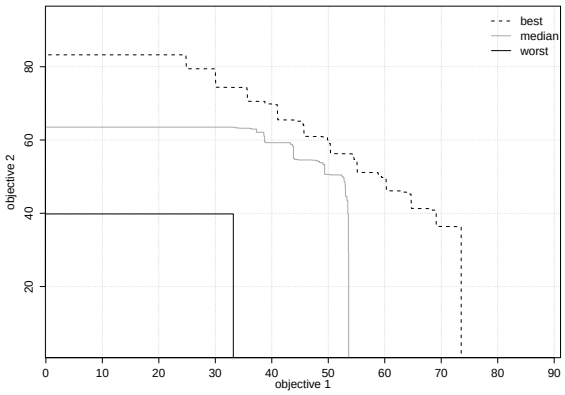

Figure 4.8: Difference Evaluation NSGA-II EAF; Centralized

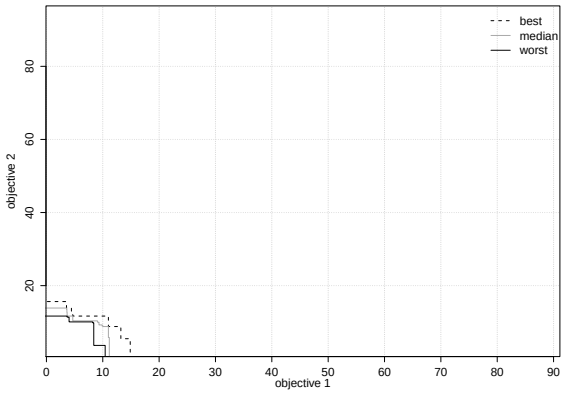

Figure 4.9: Difference of NSGA-II calculations EAF; Prior; Decentralized

4.5.4 Order of Operations

Finally, we examine the possible order of operations between the NSGA-II operation and the difference evaluation. Figure 4.6 shows that in the case of difference evaluations first (“Post”), the calculation becomes a “NSGA-II calculation of difference evaluations”, which leads to extremely high performance.

In the case of NSGA-II first (“Prior”), however, the calculation becomes a “differ-

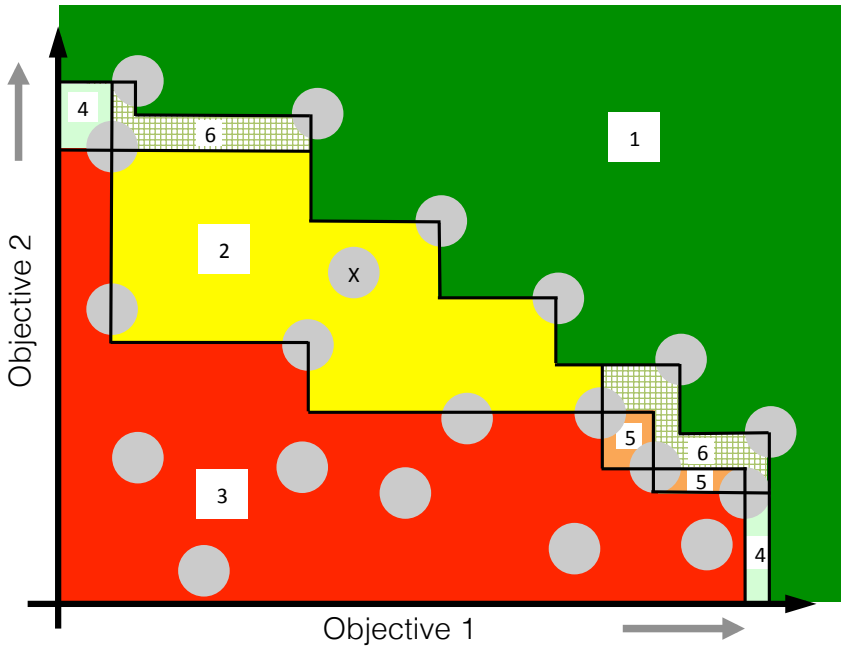

Figure 4.10: The lack of gradient information available in NSGA-II. Point X can move anywhere in Region 2, and will receive no change in evaluation. It can only surely improve its evaluation by moving to Region 1, or surely decrease its evaluation by moving to Region 3. Regions 4, 5, and 6 depend on the positions of neighbors. Total order of Regions: $3 \ll 5 < 2 < 4 < 6 \ll 1$. The difference evaluation of RVNSGA-II is only nonzero for agent a in trial X when X_{-a} is in a different region than X.

ence of NSGA-II evaluations", which leads to extremely poor performance (Figure 4.9). This is because the calculation of the NSGA-II fitness is not affected by fine-grained adjustments to the vector itself. The crowding distance is not a function of the position of vector, but only its neighbors on the non-dominated front (Figure 4.10). Thus, a small move produces zero change in NSGA-II evaluation, creating a zero difference evaluation. The only way an agent can "make a difference" is by changing the system performance enough to dominate a point that wasn't previously dominated. This is incredibly uninformative, and leads to poor system performance: an agent can improve

system performance on all objectives, and attain the same difference reward (in this formulation) as an agent who decreases system performance on all objectives. Thus, our recommendation is to perform a decentralized NSGA-II calculation of difference rewards: this attains high performance (Figure 4.6) at low computational effort.

4.6 Conclusion

In this chapter we have presented a novel method for integrating the successful multi-objective algorithm NSGA-II into multiagent systems. We first derived a real-valued fitness assignment method that provides an equivalent total ordering of policies to NSGA-II for use with additional calculations. Our experimental results confirm that credit assignment is vital in multiagent, multi-objective systems, and without considering credit assignment, NSGA-II performs only slightly better than a linear combination of objectives when both use a global evaluation.

Using difference evaluations for credit assignment improves the performance of linear combinations, confirming the results in other domains from the previous chapter. Using a decentralized NSGA-II calculation in tandem with difference evaluations results in the best performance, since they address different sides of the problem: difference evaluations assign credit to individual agents, while NSGA-II determines the relative importance of improvements in each individual objective. We also determined that a centralized NSGA-II calculation offers no additional benefits over the decentralized calculation, despite the additional computation time ($\mathcal{O}(Agents^2)$) and increased resolution on the fitness space. This formulation tends to push system performance to-

ward tradeoff solutions in the middle of the Pareto front, ignoring the extremes. Finally, we determined that calculating the "difference of NSGA-II fitness" is a destructive fitness function, and discovered that this was caused by a lack of gradient information due to NSGA-II's formulation.

Chapter 5 – The Pareto Concavity Elimination Transformation: PaCcET

In the previous two chapters, we have developed the concept of credit assignment for use in reinforcement learning with a scalarized objective function (Chapter 3), and introduced credit assignment into NSGA-II (Chapter 4). However, each of these have drawbacks associated with them: a priori scalarization has a number of well-documented flaws [31], including an inability to discover Pareto fronts with certain properties. NSGA-II, while capable of discovering arbitrary Pareto fronts, requires orders of magnitude more computation, which may be prohibitive to deploy in a multiagent setting with limited computation.

In this chapter, we develop a computationally-inexpensive alternative for multi-objective optimization. In this chapter, we only consider a centralized solution concept (a single agent), and address multiagent concerns in later chapters. At the core of this method is the concept of indifference [44, 85]: a tradeoff for which an agent should be willing to accept either alternative with no preference between the two.

The necessary background for this chapter includes evolutionary algorithms (Section 2.1.2), Pareto optimality, utopia and nadir vectors, (Section 2.3), and normalization and scalarization (Section 2.4.1). Among other comparisons, we look at empirical attainment functions (Section 2.3) to compare the performance of the algorithm we develop against NSGA-II (Section 2.4.3) and SPEA2 (Section 2.4.4) on multi-objective benchmark problems [37, 70].

5.1 Contributions

The primary contribution of this chapter is to present the Pareto Concavity Elimitation Transformation (PaCcET), a novel, optimizer-independent, iterative multi-objective transformation. It transforms the objective space so that the Pareto Front is convex, and requires only a single user-defined parameter. This allows an linear combination with unit weights (in the transformed objective space) to find concave areas of the Pareto front (in the original objective space), removing the major drawbacks of a linear combination, and allowing a simple linear combination to be used instead of more computationally expensive multi-objective evolutionary algorithms, and produce similar results. We furthermore provide a theoretical proof that the true solution to the PaCcET optimization problem is always Pareto optimal, and will find Pareto optimal points even in concave areas of the Pareto frontier. Finally, we test PaCcET and its associated extensions on a 10-objective problem, and compare performance to NSGA-II and SPEA2.

PaCcET Intuition Intuitively, the purpose of the PaCcET transformation is to make the current Pareto approximate set equally valuable, as we are indifferent between these solutions [44]. One way to do this is to register the Pareto approximation (P_I^*) on to points on the normalized utopia hyperplane [77]. This can be achieved through a non-rigid registration [21], which forms the core of the transformation.

PaCcET generalizes to k objectives, but for the intuition, consider two objectives. PaCcET can be seen as radially expanding or contracting the scaling of the space, centered on $\hat{u}^{\circ,norm}$, with the scaling changing with the angle θ from the x-axis (See Fig-

ure 5.1). In locations where P_I^* is concave, that scaling factor will be < 1, contracting the space along that vector until that point in P_I^* is on the utopia hyperplane. Where P_I^* is convex, that scaling factor will be > 1, expanding the space so that the P_I^* point is on the utopia hyperplane. The normalized utopia hyperplane is equally valuable to an unweighted linear combination of objectives (See Figure 5.5). This guarantees that all points in P_I^* have the same linear combination evaluation.

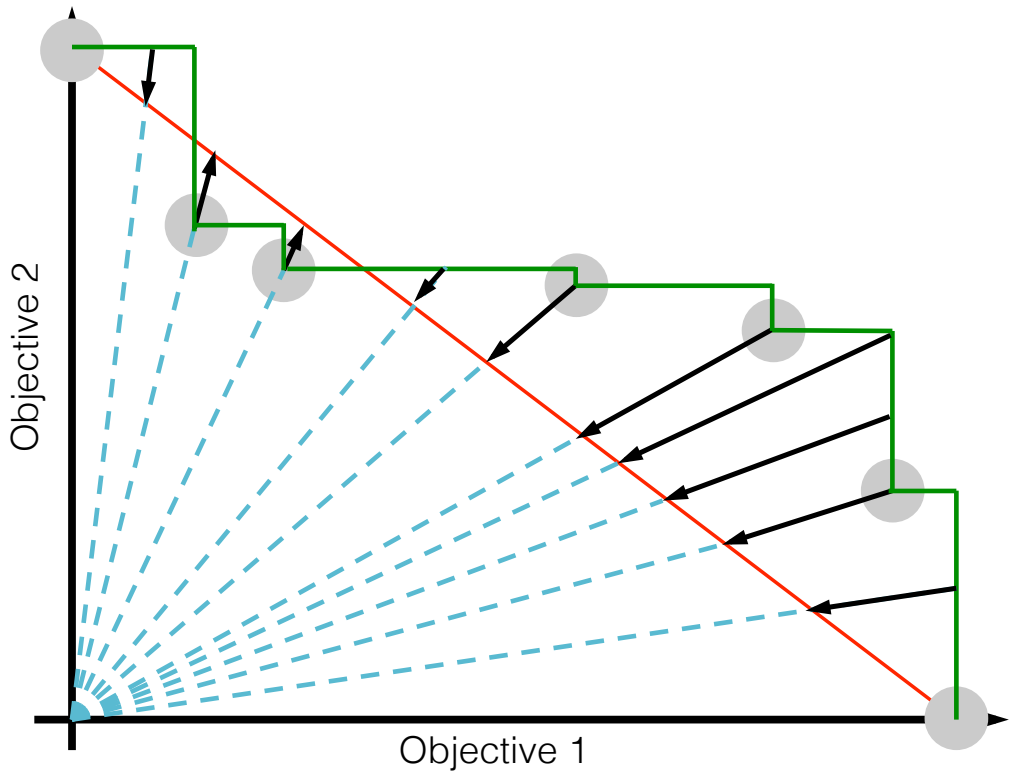

Figure 5.1: Intuition for the PaCcET process in a two-objective problem (PaCcET generalizes to k objectives). Grey points in P_I^* are scaled radially (centered on the approximate utopia point) away if they are in front of the utopia hyperplane, and scaled radially closer if they are behind the utopia hyperplane. All points in P_I^* then have the same unweighted linear combination.

Moreover, PaCcET takes a conservative approach to interpolating between the points, always using the upper bound of the non-dominated hyperspace (Λ_B, see Figure 5.3) for determining what the scaling factor should be between points in P_I^*. This means that points that are not dominated by P_I^* will have a linear combination of $< LC(P_I^*)$, while points that are dominated by P_I^* will have a linear combination of $> LC(P_I^*)$. Thus, non-dominated solutions are preferred, and will be discovered during optimization.

5.2 Pareto Concavity Elimination Transformation (PaCcET)

If we consider the goal of multi-objective optimization to make our approximation of the Pareto front, P_I^* as close as possible to the true Pareto front $\mathbf{P}^*$, then since P_I^* serves as an upper bound (for minimization problems) of the Pareto front, it follows that we should prefer to discover solutions that lie between our current P_I^* and $\mathbf{P}^*$. However, since we have no knowledge in general of the form of $\mathbf{P}^*$, we cannot arbitrarily hope to improve on every objective from any point in the objective space Λ, as when part of P_I^* approaches $\mathbf{P}^*$, it will necessarily be approaching the edge of the attainable space. Instead of arbitrarily placing a scalar value function over Λ, then, we take the approach of forming the value function based on our current P_I^*, and as this approximation improves over time, our value function will form to the contours of $\mathbf{P}^*$. How we achieve this effect is described in the remainder of this section.

Each point in the current Pareto-approximate set, P_I^* represents a tradeoff between which we are indifferent [44]. PaCcET makes each solution on Λ_{B} (including P_I^*) equally valuable to a linear combination in Λ^τ through a two step transformation, which

first transforms from Λ to Λ^{norm}, and then transforms from Λ^{norm} to Λ^{τ}, where the τ superscript on any space or set denotes the transformed space or set. This means that any Pareto-approximate solution will have a linear combination evaluation of $(k-1)$ when all weights are set to 1. All solutions in $\Lambda^{\tau}_{\text{N}}$ (the non-dominated hyperspace) will have a linear combination evaluation $< (k-1)$, and all solutions in $\Lambda^{\tau}_{\text{D}}$ (the dominated hyperspace) will have a linear combination evaluation $> (k-1)$.

Algorithm: To determine the transformed evaluation for a given solution vector v, we require the current Pareto approximate set P^*_I, from which we can calculate the approximate utopia point $\hat{u}^{\circ}$ based on P^*_I (Eq. 2.3), and the matching nadir approximation $\hat{u}^{\bullet}$ (Eq. 2.4). The first step is to normalize the target vector v such that each objective takes

Algorithm 5.1: PaCcET for iteration I

Data: Set of solutions $\mathbf{V}$
Data: Pareto Approximate Set P^*_I
Find $\hat{u}^{\circ}, \hat{u}^{\bullet}$ (Eq. 2.3–2.4) ;
$\forall c \; w_c = 1$;
forall the *Solutions* $i \in \mathbf{V}$ **do**
    Find v_i^{norm} (Eq. 5.1);
    Find $\vec{r}$ (Eq. 5.2);
    Find $||v^{\text{norm}}||_1$ (Eq. 5.3);
    Find $||v||_{\text{B}}$ (Eq. 5.4);
    Find $||v||_{\text{hp}}$ (Eq. 5.5);
    Find d^{τ} (Eq. 5.6);
    Find v_i^{τ} (Eq. 5.7);
    $Fit_{\text{PaCcET}}(v_i) = LC(v_i^{\tau})$ (Eq. 2.7);
end

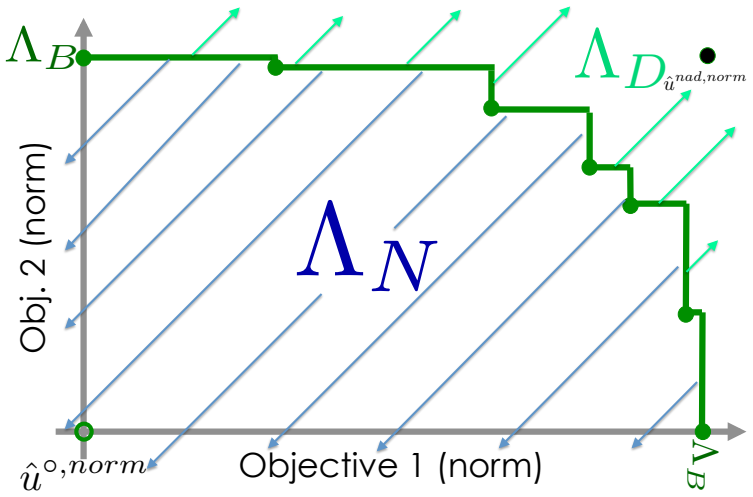

Figure 5.2: Visualization of the partitions in the multi-objective space. Green dots correspond to vectors in $P_I^{*,\text{norm}}$ (which form the border, Λ_B, between the non-dominated hyperspace Λ_N and the dominated hyperspace, Λ_D).

on a value not less than 0, transforming Λ to Λ^{norm} [33, 72]:

$$v^{\text{norm}}(c) = \frac{v(c) - \hat{u}^{\circ}(c)}{\hat{u}^{\bullet}(c) - \hat{u}^{\circ}(c)} \tag{5.1}$$

By definition $\hat{u}^{\bullet,\text{norm}} \equiv \mathbf{1}$ and $\hat{u}^{\circ,\text{norm}} \equiv \mathbf{0}$, and each element of a member of P_I^* will be in the range [0:1].

The second step is to perform the transformation from Λ^{norm} to Λ^{τ}. Within this process, we use the unit vector $\vec{r}$ that points from $\hat{u}^{\circ,norm}$ toward v^{norm}:

$$\vec{r} = \frac{v^{\text{norm}}}{|v^{\text{norm}}|} \tag{5.2}$$

All distance measurements in the transformation process are taken along the direc-

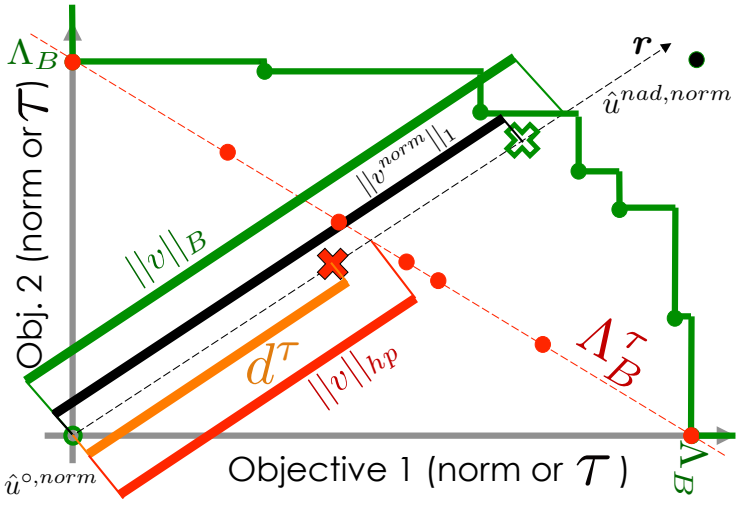

Figure 5.3: Visualization of quantities used in transformation. The vector v^{norm} is represented by the hollow green X mark, and v^τ by the solid red X mark. v^{norm} lies outside of the dominated hypervolume, so is a desirable point to discover. Green dots correspond to vectors in $P_I^{*,\text{norm}}$. Red correspond to their transformations in $P_I^{*^\tau}$. All measurements are Manhattan Distance (L_1 norm) along $\vec{r}$.

tion of $\vec{r}$. We measure three distances for use in PaCcET:

- L_1 distance (linear combination or Manhattan Distance) from $\hat{u}^{\circ,norm}$ to v^{norm}:

$$||v^{\text{norm}}||_1 = \sum_i v_i^{\text{norm}} \tag{5.3}$$

- L_1 distance from $\hat{u}^{\circ,norm}$ to the normalized dominated border $\Lambda_{\text{B}}^{\text{norm}}$ along $\vec{r}$:

$$||v||_{\text{B}} = min(\gamma) \ni \gamma\vec{r} \succeq P_I^* \tag{5.4}$$

- L_1 distance from $\hat{u}^{\circ,norm}$ to the normalized utopia hyperplane $\Lambda_{\mathrm{B}}^{\tau}$ [77] along $\vec{r}$:

$$||v||_{\mathrm{hp}} = \beta \ni \sum_i \beta \vec{r}_i = (k-1) \tag{5.5}$$

We then calculate d^{τ}, which determines where v^{τ} is located:

$$d^{\tau} = ||v||_{\mathrm{hp}} \frac{||v^{\mathrm{norm}}||_1}{||v||_{\mathrm{B}}} \tag{5.6}$$

And finally we determine the location of v^{τ}, enclosing the whole process:

$$v^{\tau} = d^{\tau}\vec{r} = PaCcET(v) \tag{5.7}$$

Choosing the Maximum Size of P_I^*, the Pareto approximate set: P_I^* is maintained in the same way as any Pareto optimality calculation. However, for computation and memory concerns, its size must be limited. The size of P_I^* is the only user-defined parameter in PaCcET, and corresponds directly to the granularity of the Pareto front estimation. In our experiments we use 250 as the size. We ran tests with a size as small as 50, in which the algorithm still functions, but provides a very coarse approximation of the true Pareto front. Once over the chosen size, we used random elimination of non-extreme elements. We also tested with nearest-neighbor elimination and k-nearest neighbor elimination, the performance of PaCcET was not sensitive.

5.3 Theoretical Properties of PaCcET

In this section we provide two theorems which together prove that PaCcET finds Pareto optimal solutions, even in concave areas of the Pareto front. We begin by assuming:

Assumption A1. *The system designer specifies k points that are incomparable to the Pareto front, which describe a hyper-prism that completely bounds the Pareto front.*

Assumption A2. *Optimizer Ξ solves the PaCcET problem exactly in a single iteration.*

Assumption A3. *The feasible region has no solutions that are weakly dominated by the Pareto front.*

Assumption A4. *The Pareto front is continuous.*

A1 provides us vectors with which we seed P_I^*, and assures PaCcET is calculable in the whole feasible objective space. A2 allows us to use the exact solution to the PaCcET minimization problem to determine how P_I^* changes over iterations. A3 and A4 allow us to draw conclusions in k-objective space without any other restrictions on the shape of the Pareto front.

In practice, Assumptions A2 and A3 are rarely met, and Assumption A4 is met only in some types of problems, but whether or not it is met is not known before optimization. Assumption A1 requires some external knowledge of the environment that can be provided by a system expert, and is also not always available. We use these assumptions for the below proofs, but in the experiments that follow, we relax all four of these assumptions and find that we still achieve favorable performance.

Theorem T1. *The solution to the PaCcET optimization problem will be Pareto Optimal.*

Proof. There exists an infinite number of possible rays $\vec{r} \in \vec{R}$ (where $\vec{R}$ is the set of all rays originating from **0**) on which the true solution may exist. This solution exists only along one of those rays, which must pass through the feasible space. We do not seek to determine which $\vec{r}$ it lies on. For any individual $\vec{r}$, the PaCcET optimization problem takes the form (Eq. 5.6, reorganized):

$$\min(d^{\tau}) = \min\left(||v||_{\text{hp}} \frac{||v^{\text{norm}}||_1}{||v||_{\text{B}}}\right) \tag{5.8}$$

And for a constant $\vec{r}$, $||v||_{\text{B}}$ and $||v||_{\text{hp}}$ are constant on a given iteration:

$$\min(d^{\tau}) = \min(\alpha||v^{\text{norm}}||_1) \tag{5.9}$$

where α is some positive constant. $||v^{\text{norm}}||_1$ increases monotonically as distance from the origin increases, therefore d^{τ} does as well. The minimum of d^{τ}, then, will be on the border of the feasible space, a Pareto Optimal Solution or a weakly dominated solution [72]. By A3 and A4, this is a Pareto optimal solution. This can also be assured by the same logic as [31], since it is equivalent to a scaled linear combination. □

Theorem T2. *PaCcET finds solutions in concave areas of the Pareto front.*

Proof. Assume a globally concave search space. By theorem T1, in the worst case, the solution to the PaCcET optimization problem will lead to the k anchor points (single objective extremes) in the first k iterations. By A4, we know additional Pareto optimal points exist. We show that the d^{τ} calculations for those points in the current P_I^* is

greater than those in Λ_{N} (super/sub-scripts denoting the calculation for a member of the set named in the super/sub-script):

$$d^{\tau}_{P^*_I} > d^{\tau}_{\Lambda_{\text{N}}} \tag{5.10}$$

$$||v||^{P^*_I}_{\text{hp}} \frac{||v^{\text{norm}}_{P^*_I}||_1}{||v||^{P^*_I}_{\text{B}}} > ||v||^{\Lambda_{\text{N}}}_{\text{hp}} \frac{||v^{\text{norm}}_{\Lambda_{\text{N}}}||_1}{||v||^{\Lambda_{\text{N}}}_{\text{B}}} \tag{5.11}$$

$$(k-1)\frac{||v^{\text{norm}}_{P^*_I}||_1}{||v||^{P^*_I}_{\text{B}}} > (k-1)\frac{||v^{\text{norm}}_{\Lambda_{\text{N}}}||_1}{||v||^{\Lambda_{\text{N}}}_{\text{B}}} \tag{5.12}$$

By definition $||v||^{P^*_I}_{\text{hp}} = (k-1)$. Also, $\frac{||v^{\text{norm}}_{P^*_I}||_1}{||v||^{P^*_I}_{\text{B}}} = 1$ because $||v^{\text{norm}}_{P^*_I}||_1 = ||v||^{P^*_I}_{\text{B}}$, and the quantity $\frac{||v^{\text{norm}}_{\Lambda_{\text{N}}}||_1}{||v||^{\Lambda_{\text{N}}}_{\text{B}}} \in [0 : 1)$, because it is in the non-dominated subspace (so $||v^{\text{norm}}_{\Lambda_{\text{N}}}||_1 < ||v||^{\Lambda_{\text{N}}}_{\text{B}}$), and the inequality in Eq. 5.12 holds. Because of Theorem T1, we know that the solution will be Pareto optimal, and because of the globally concave assumption, we know this point is on a concave region of the Pareto front. □

Implications: The significance of these two theorems is as follows: the true solution to the PaCcET problem will always be a Pareto optimal solution, and PaCcET will be able to find concave areas of the Pareto front. Because the assumptions used to generate these conclusions are restrictive, in the following empirical results sections, we take steps to violate each of the assumptions categorically, and PaCcET is still able to find good coverage over concave Pareto fronts.

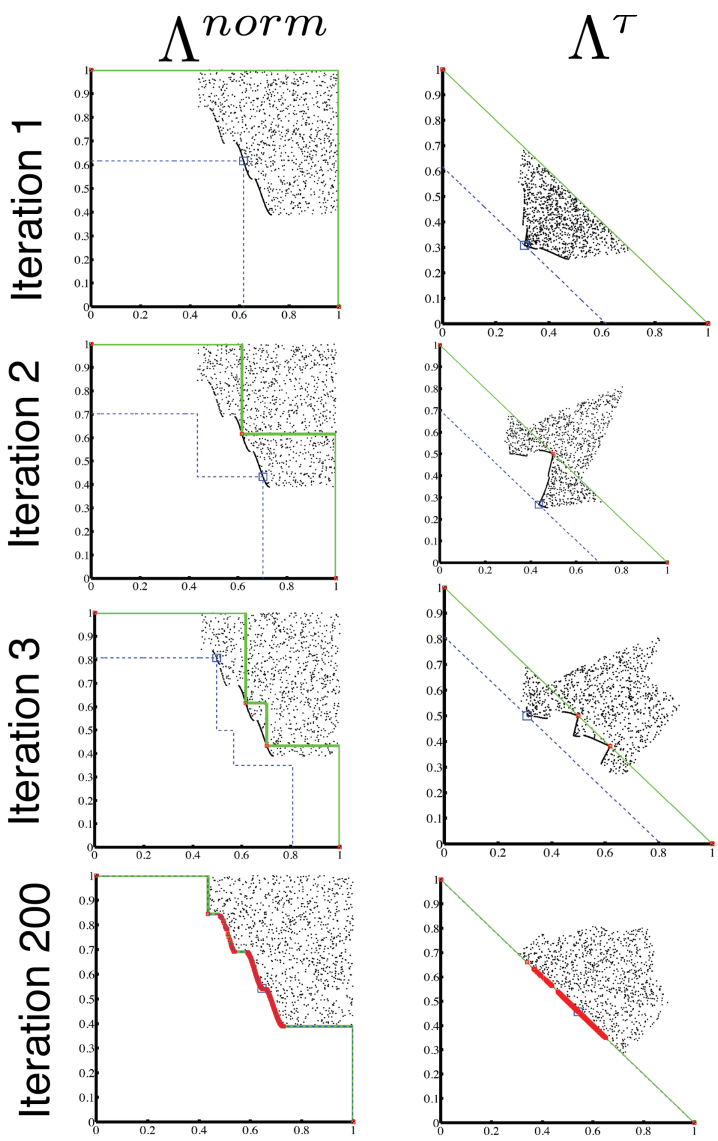

Figure 5.4: Visualization of PaCcET procedure over iterations. The left column is the normalized objective space Λ^{norm}. The right column is the transformed objective space, Λ^{τ}. The rows show, in turn, the optimizer working at the 1st, 2nd, 3rd, and 200th iteration. In the left column, Black points are candidate solutions. Red points are solutions in P_I^*, the Pareto approximate set. The green solid line denotes $\Lambda_{\text{B}}^{\text{norm}}$ The blue square denotes the true solution to the PaCcET optimization problem at that iteration. The blue dashed line is the level curve of the PaCcET evaluation on which all solutions are as valuable as the discovered solution. In the right column, the colors and symbols map to the transformed versions of the same points as described previously, in Λ^{τ}.

5.4 Experiment: KUR

As a first experimental domain, we use a test problem (KUR) from multi-objective optimization with a discontinuous and locally concave Pareto front (which breaks A4) [33]:

$$f_1(\mathbf{x}) = \sum_{i=1}^{2} \left[-10 \; exp\left((-0.2)\sqrt{x_i^2 + x_{i+1}^2} \right) \right] \tag{5.13}$$

$$f_2(\mathbf{x}) = \sum_{i=1}^{3} \left[|x_i|^{0.8} + 5\sin(x_i)^3 \right] \tag{5.14}$$

Where f_1 and f_2 are to be minimized by controlling the decision variables:

$$x_i \in [-5, 5] \quad ; \quad i \in \{1, 2, 3\} \tag{5.15}$$

A vector **x** is evaluated:

$$Fit_{\text{LC}}(\mathbf{x}) = w_1 f_1 + w_2 f_2 \tag{5.16}$$

where altering the weights can lead to different portions of the Pareto front being covered. For PaCcET:

$$Fit_{\text{PaCcET}}(\mathbf{x}) = f_1^\tau + f_2^\tau \tag{5.17}$$

where f_1^τ and f_2^τ represent the transformed objectives, within Λ^τ, calculated as:

$$\{f_1^\tau, f_2^\tau\} = PaCcET(\{f_1, f_2\}) \tag{5.18}$$

As the optimizer Ξ, we use an evolutionary algorithm (which breaks A2), in which

the population members are vectors of length 3 that meet the criteria set forth in Eq. 5.15. We maintain a population of 100 solutions, with the 50 worst-performing solutions removed after each generation, replaced by copies of the winner of 50 binary tournaments, with each element of the vector changed by a random number chosen by a normal distribution centered around 0 with standard deviation 0.25. We do not seed P_I^* (which breaks A1).

Figure 5.6 shows the Empirical Attainment Function (EAF) [49] for each method, respectively. It shows PaCcET's worst performance exceeds that of the linear combination's median performance, and PaCcET's worst performance exceeds NSGA-II's worst performance. SPEA2 and PaCcET perform comparably after 5000 generations.

Figure 5.7 shows the percent of dominated hypervolume by PaCcET and two successful multi-objective methods, SPEA2 and NSGA-II, as a function of number of indi-

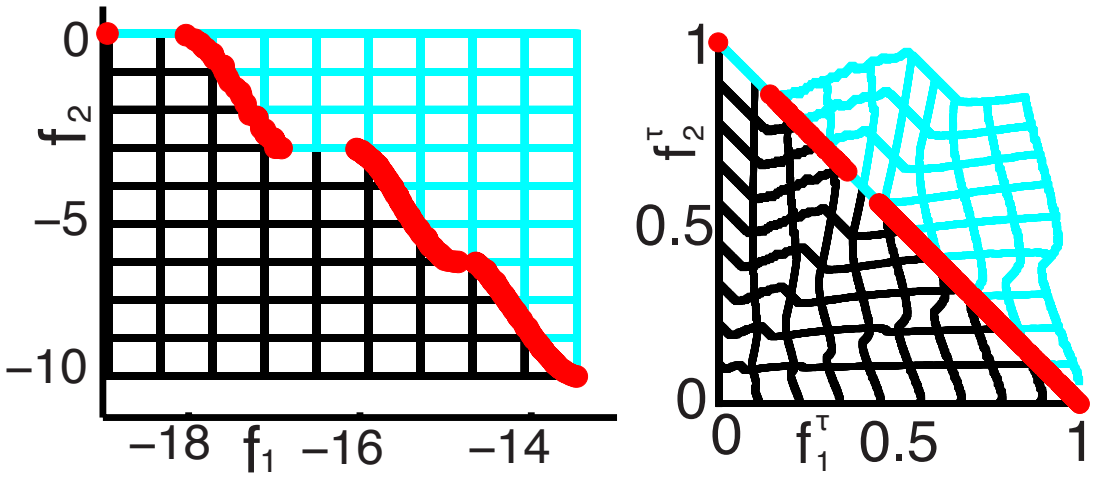

Figure 5.5: A grid of points in Λ (Left) and Λ^τ (Right). After many iterations, cyan points are dominated by the red set P_I^*. Nondominated points shown in black. PaCcET distorts the objective space such that a linear combination in the transformed space is a complex non-linear combination in the un-transformed space.

vidual fitness evaluations. PaCcET proceeds faster than the other two methods toward the Pareto front. All methods shown eventually converge to a good approximation of the Pareto front, and dominate a similar amount of hypervolume.

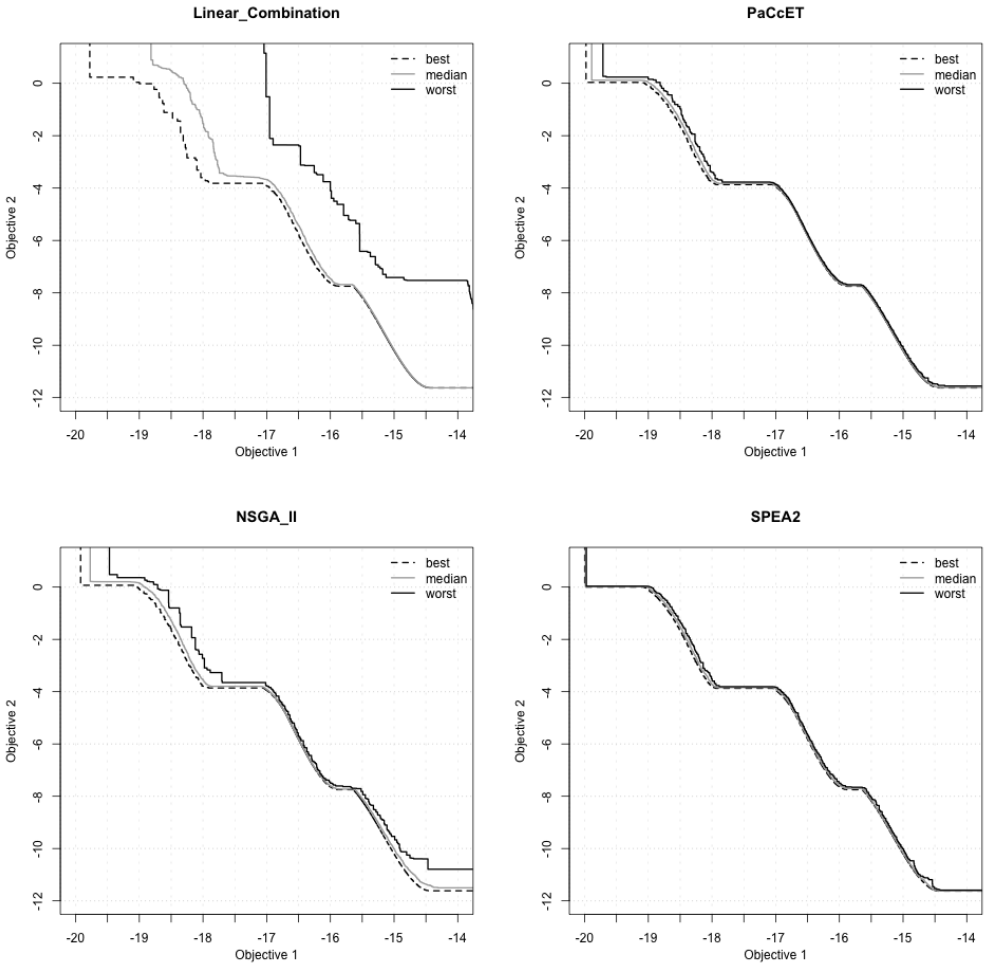

Figure 5.6: KUR Empirical Attainment Functions, shown in Λ.

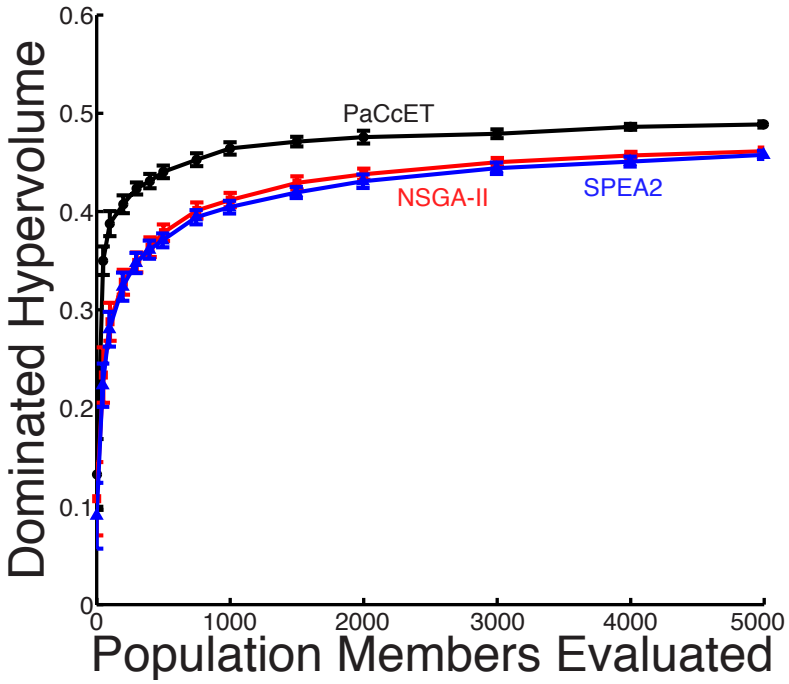

Figure 5.7: Percent of hypervolume dominated in KUR, calculated using the limits of Fig. 5.6 over 50 statistical runs. Error in the mean ($\sigma/\sqrt{N}$ where $N = 50$), is smaller than the plotted symbols.

5.5 Experiment: DTLZ2

As a second experimental domain, we use one of the test problems out of the battery developed by Deb, Thiele, Laumanns and Zitzler, DTLZ2 [37, 38]. A solution is described by a vector ($\mathbf{x} = \{x_1, x_2, \mathbf{x_M}\}$) of length 12, where 2 elements (x_1, x_2) determine at what angles in the 3 dimensional objective space evaluation v will lie and the remaining 10 elements ($\mathbf{x_M}$) determine the distance from the origin at which v will lie. The three functions to be minimized are:

$$f_1(\mathbf{x}) = (1 + g(\mathbf{x}_M)) \cos\left(x_1 \frac{\pi}{2}\right) \cos\left(x_2 \frac{\pi}{2}\right) \tag{5.19}$$

$$f_2(\mathbf{x}) = (1 + g(\mathbf{x}_M)) \cos\left(x_1 \frac{\pi}{2}\right) \sin\left(x_2 \frac{\pi}{2}\right) \tag{5.20}$$

$$f_3(\mathbf{x}) = (1 + g(\mathbf{x}_M)) \sin\left(x_1 \frac{\pi}{2}\right) \tag{5.21}$$

subject to each element of $\mathbf{x}$ remaining in the range [0:1], and the evaluation $g(\mathbf{x}_M)$ calculated as:

$$g(\mathbf{x}_M) = \sum_{x_i \in \mathbf{x}_M} (x_i - 0.5)^2 \tag{5.22}$$

This results in a known Pareto front that can be described by the octant of a sphere of radius 1 for which $f_1, f_2,$ and f_3 are all positive. The feasible space has a large area that is weakly dominated by the Pareto front (which breaks A3).

The fitness of a vector $\mathbf{x}$ is calculated as:

$$Fit_{\text{LC}}(\mathbf{x}) = w_1 f_1^{\text{norm}} + w_2 f_2^{\text{norm}} + w_3 f_3^{\text{norm}} \tag{5.23}$$

and for PaCcET,

$$Fit_{\text{PaCcET}}(\mathbf{x}) = f_1^\tau + f_2^\tau + f_3^\tau \tag{5.24}$$

where f_1^τ, f_2^τ, and f_3^τ represent the transformed objectives, within Λ^τ, calculated as:

$$\{f_1^\tau, f_2^\tau, f_3^\tau\} = PaCcET(\{f_1, f_2, f_3\}) \tag{5.25}$$

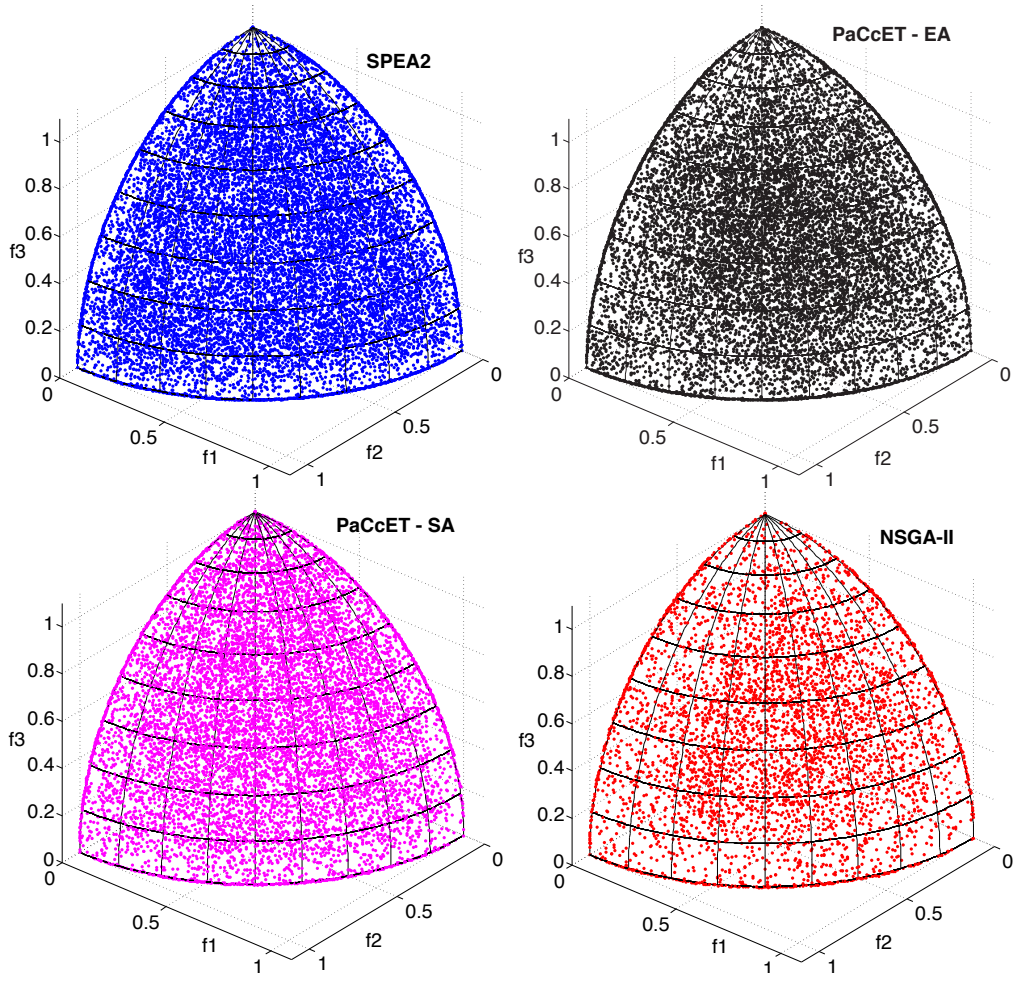

Figure 5.8: All Pareto optimal points discovered in one statistical run of DTLZ2 in Λ.

We use the same optimizer Ξ for DTLZ2 as for KUR (which breaks A2), except members are vectors of length 12 with each element in the range [0:1], and the mutation operator alters each element by a random number drawn from a normal distribution centered around 0 with standard deviation 0.05. We do not seed P_I^* (which breaks A1).

Figure 5.8 shows the results on DTLZ2 for a typical experimental run of 5000 generations for each method (simulated annealing allowed the same number of global function calls as the EAs), reporting the non-dominated points found through the entire experimental run (Note that this is distinct from P_I^*, which was kept at a size of 250). SPEA2 and PaCcET using an evolutionary algorithm (PaCcET – EA) both find a similar number of solutions spread all across the Pareto front. PaCcET using simulated annealing (PaCcET – SA) is slightly less successful but still generates good coverage, even though it is not using a population-based optimizer. NSGA-II produces fewer Pareto optimal points, but still maintains coverage. The linear combination (not shown) converges to one of the extremes very quickly, producing very poor coverage, regardless of the choice of weights.

5.6 Empirical Runtime Study

To examine the properties of PaCcET's run time, we performed a series of experiments in which we had SPEA2, NSGA-II, and PaCcET perform their evaluations for every member in a population of solutions, and report the number of CPU ticks required for each to complete. Specifically, these times only include the time to run each respective algorithm, and do not include the time to evaluate from $\Omega \rightarrow \Lambda$. Including these

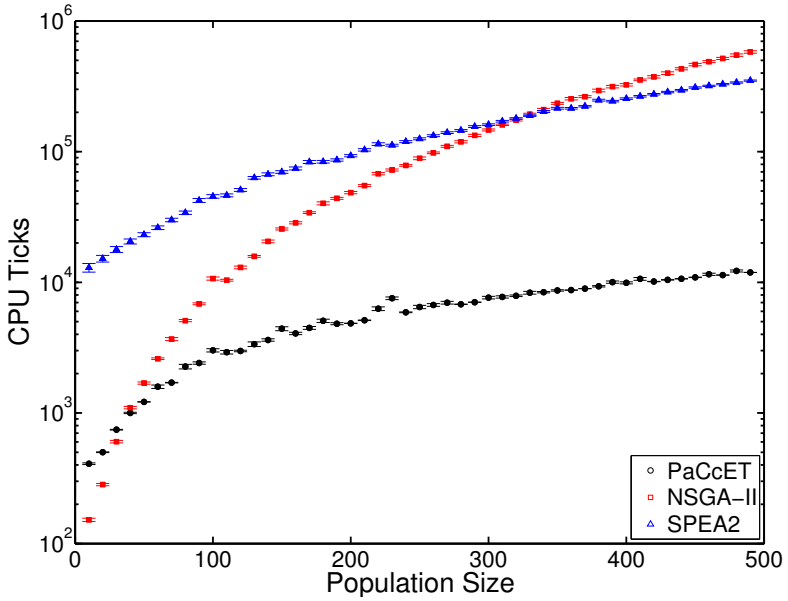

Figure 5.9: Runtime analysis as a function of population size with three objectives with runtime on a logarithmic scale (lower is better). Error in the mean ($\sigma/\sqrt{N}$ where $N = 30$), is sometimes smaller than the plotted symbols. PaCcET is computationally cheaper than NSGA-II and SPEA2.

evaluations would add a constant to each computation time.

Figure 5.9 shows that in a 3-objective problem with a low population size (less than 40), NSGA-II has the lowest computation time of the three algorithms, but with a larger population scales very poorly. SPEA2 starts with a higher computation burden, but scales more gracefully as the population size is increased. PaCcET performs more than an order of magnitude better than either of the other methods at high population sizes. In highly complex problems, a large population size can be desirable to maintain diverse sets of solutions, so being able to evaluate these populations quickly is a benefit of PaCcET.

5.7 Discussion and Conclusion

In this work we have presented a low computational cost way to improve the performance of a linear combination in multi-objective problems. PaCcET convexities concave regions of the Pareto front for the sake of training, and allows for solutions in these areas to be found by an optimizer using a linear combination of transformed objectives.

The primary benefits of PaCcET displayed in this work are:

1. It allows a linear combination of transformed objectives to find concave areas of the Pareto front in the original objective space.
2. It acts independently of the chosen optimizer.
3. It creates a wide spread of solutions along the Pareto front on concave or discontinuous fronts.
4. It removes the need for the system designer to choose weights.
5. It functions in higher-than-two objective problems.

The first benefit (1) allows a simple linear combination to be applied to a much broader class of multi-objective problems than it could be otherwise. Benefit (2) means that optimizers like evolutionary algorithms, A* search, simulated annealing, or particle swarm optimization can be applied to multi-objective problems through PaCcET with little alteration; it also means that future developments in single-objective optimizers are immediately useful to a large class of multi-objective problem, but comes at the cost that PaCcET is limited by the quality of the optimizer. Benefit (3) reinforces (1): Even on

challenging Pareto fronts, PaCcET develops a desirable array of solutions to choose between. Benefits (4,5) remove one of the primary challenges in using a linear combination on more-than-two objective problems.

PaCcET offers a fundamentally different possible avenue for multi-objective research: the elimination of concavity as opposed to the development of methods that deal well with concave Pareto fronts. Future work in this area includes testing the PaCcET on a large testbed of multi-objective problems.

Chapter 6 – Extending PaCcET for Complete Coverage and Steerability

In the previous chapter, we developed the PaCcET Transformation (Chapter 5), which transforms the objective space such that it favors the use of a linear combination of objective values. We identified that it executes quickly and provides good solution quality in the single-agent case, but there are also a number of ways in which it can be improved. Since only a single parameter may be specified (the maximum size of P_I^*), this offers the user limited control over the behavior of the algorithm.

In this chapter, we introduce three extensions to provide the system designer with additional methods by which the performance of PaCcET may be controlled to provide a finer grain of solutions on a certain area of the Pareto front, to avoid a certain area of the Pareto front, or to provide an even spread across the entire Pareto front. We additionally offer a theoretical proof that the extension for evenly spreading solutions will discover solutions within an arbitrarily small distance of any point on the Pareto front.

This chapter again deals with multi-objective concepts, so the necessary background includes the PaCcET transformation (Chapter 5), evolutionary algorithms (Section 2.1.2), Pareto optimality, utopia and nadir vectors, (Section 2.3), and normalization and scalarization (Section 2.4.1).

6.0.1 The Pareto Concavity Elimination Transformation (PaCcET)

The core functionality of PaCcET is through a transformation, which is then fed into a linear combination, which is then given to an optimizer Ξ. For this chapter, we notate this process as $\mathcal{TR} \rightarrow \mathcal{LC} \rightarrow \Xi$.

In this chapter, we introduce three extension algorithms that can be incorporated into the PaCcET procedure either independently, or in tandem. These extensions are the complete coverage extension ($\mathcal{CC}$, Section 6.1.1), the reference point extension ($\mathcal{RP}$, Section 6.2.1), and the interactive extension ($\mathcal{I}$, Section 6.2).

6.0.2 Established Theoretical Properties

In the previous chapter, we established that with the following assumptions, theoretical guarantees arise. These are:

Assumption A1. *The system designer specifies* k *points that are incomparable to the Pareto front, which describe a hyper-prism that completely bounds the Pareto front.*

Assumption A2. *Optimizer Ξ solves the PaCcET problem exactly in a single iteration.*

Assumption A3. *The feasible region has no solutions that are weakly dominated by the Pareto front.*

Assumption A4. *The Pareto front is continuous.*

Theorem T1. *The solution to the PaCcET optimization problem will be Pareto optimal.*

Theorem T2. *PaCcET finds solutions in concave areas of the Pareto front.*

In the previous chapter A1-A4 are categorically violated, yet PaCcET still achieves high system performance that partially supports T1 and T2. Note that despite T1 and T2, there are no formal guarantees regarding coverage of the entire Pareto front.

6.1 Algorithms

In this chapter we introduce 3 novel extensions for use in PaCcET: the $\mathcal{CC}$ extension guarantees complete coverage of the Pareto front; the $\mathcal{RP}$ extension allows the system designer to focus PaCcET's computational effort on a particular area of the Pareto front; and $\mathcal{I}$ allows the system designer to specify areas of the Pareto front to be avoided.

6.1.1 $\mathcal{CC}$ (Complete Coverage) Extension Algorithm

Algorithm 6.1 describes the process, and Figure 6.1 illustrates it. At an iteration I, the members of P_I^* are compared pairwise. Those that are sufficiently close ($< \delta$, a user-defined parameter specifying how closely spaced the discovered solutions should be along the Pareto front) generate a surrogate (Line 10), which is always in the unattainable space. If any of their objective values are too close ($< \epsilon$), a modified surrogate is generated (Line 12), guaranteeing that a nontrivial amount of hypervolume on the Pareto front is dominated by the surrogate (L3). In both cases an anti-surrogate is generated (Line 8). An anti-surrogate/surrogate pair describes the opposite corner points of a surrogate hyperprism (important in L1, T3, and T4). Finally, the surrogates are put through

a Pareto filter against the other surrogate points to reduce the size of this set. The $\mathcal{CC}$ extension then functions by using the united set $P_I^* \cup S$ during the calculation of $||v||_B$ in $\mathcal{TR}$.

6.2 $\mathcal{I}$ (Interactive) Extension Algorithm

The $\mathcal{I}$ extension functions by the user defining a custom surrogate set S at any time during problem execution, which is used in the $||v||_B$ by the PaCcET algorithm in the same way that $\mathcal{CC}$ does. These points are interactively generated by the user, instead of automatically generated. This allows for more control over where the algorithm's effort

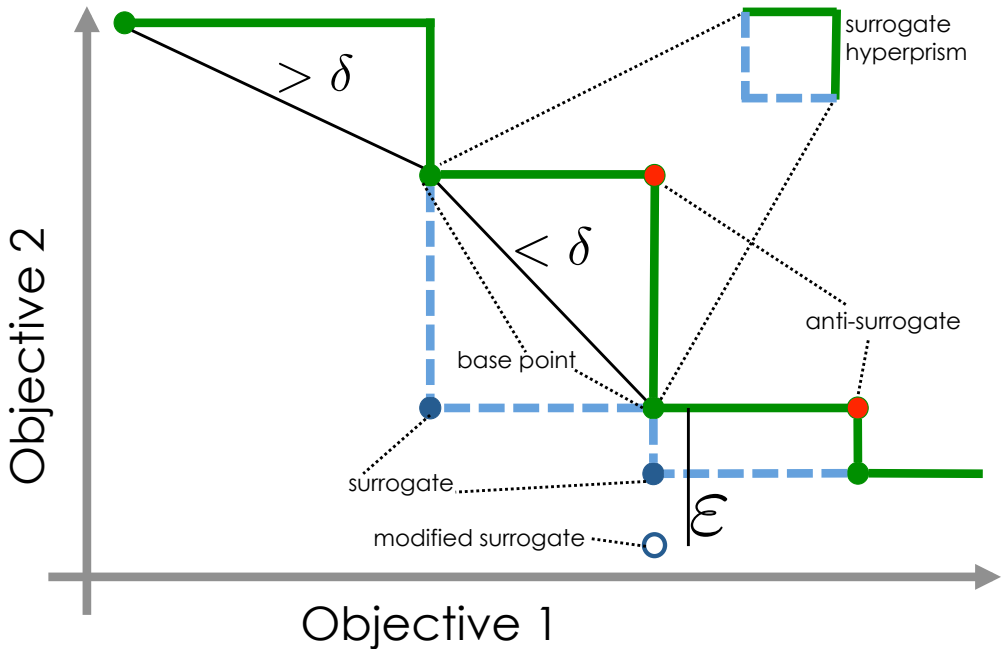

Figure 6.1: Diagram of terms included in $\mathcal{CC}$ extension, and visualization of surrogate and modified surrogate process.

is focused: points dominated by S will be valued less by Ξ, preferably ignored.

Algorithm 6.1: $\mathcal{CC}$ Module for Iteration I

Data: Solution v_i
Data: Pareto Approximate Set P_I^*
Data: Empty S
forall the *Members* $p \in P_I^*$ **do**
  forall the *Members* $q \neq p, q \in P_I^*$ **do**
    $z = size(S) = size(AS)$;
    $v_1 = P_{I,p}^*$;
    $v_2 = P_{I,q}^*$;
    if $||P_{I,p}^* - P_{I,q}^*|| > \delta$ **then**
      $S = S$
    end
    if $||P_{I,p}^* - P_{I,q}^*|| < \delta$ **then**
      $\forall c \in C : AS_{z+1}(c) = max(v_1(c), v_2(c))$ (Anti-Surrogate)
    end
    if $|P_{I,p}^*(c) - P_{I,q}^*(c)| > \epsilon$ **then**
      $\forall c \in C : S_{z+1}(c) = min(v_1(c), v_2(c))$ (Surrogate)
    else
      $\forall c \in C : S_{z+1}(c) = max(v_1(c) - \epsilon, v_2(c) - \epsilon)$ (Modified Surrogate);
    end
    $Pareto\_Filter(S)$;
  end
end

6.2.1 $\mathcal{RP}$ (Reference Point) Extension Algorithm

The $\mathcal{RP}$ extension uses a reference point calculation in place of the $\mathcal{LC}$ process. For vector v and reference point ref this is:

$$\mathcal{RP}(v, ref) = \sum_{c \in C} |v^{\tau}(c) - ref(c)| \tag{6.1}$$

The distinct benefit of $\mathcal{RP}$ is that any ref with an L_1 norm $||ref||_1 < (k-1)$ will eventually be unattainable as PaCcET refines P_I^*. This guarantees a Pareto optimal solution [42]. A standard reference point method requires prior knowledge of the shape of the true Pareto front to specify an unattainable reference point.

6.3 $\mathcal{CC}$ Theoretical Properties

We first establish that this surrogate process dictates that PaCcET find solutions on other areas of the Pareto front than those covered by the surrogate (L1). We then establish that the $\mathcal{CC}$ extension will provably execute and generate a surrogate point. This covers one extreme (solutions too spread out) (L2). Conversely, solutions can be clustered too close together, so we then establish that each new hyper prism introduced will cover a minimum amount of hypervolume on the Pareto front (L3). Finally (T3), we show that we can drive down ϵ (involved in L3) to attain the guaranteed spacing (T4).

Lemma L1. *A Module $\mathcal{CC}$ surrogate hyper prism will prevent the solution to the PaCcET optimization problem from being within its boundaries until the entire Pareto Front is*

covered with surrogate hyperprisms.

Proof.

P1. By T1, only Pareto optimal solutions will be found. The $\mathcal{CC}$ extension uses the set $P_I^* \cup S$ when calculating $||v||_B$.

P2. Therefore, of the points not dominated by P_I^*, points dominated by the S set will necessarily take on a higher (worse) value than those not dominated by S.

P3. The optimal solution at each iteration will be non-dominated by S.

□

Lemma L2. *A minimum number of surrogates will be created as a function of iterations and the number of problem objectives k.*

Proof.

P1. The Pareto front has finite length (area, volume, hypervolume) L.

P2. In the worst case, points will be discovered that are δ plus some infinitesimal distance away from each other. The number of Pareto optimal points that can be discovered in this fashion (η) is a function of optimal hypersphere packing in $(k-1)$ dimensions.

P3. After a finite number of iterations ($\eta = (L/\delta + 1)$ for $k = 2$), at least one pair of solutions will lie within δ of each other.

P4. For iterations $I > \eta$, there will be at least $I - \eta$ surrogates in existence at any iteration.

P5. The number of required iterations for the creation of the first surrogate is determined by hypersphere packing in dimension $k-1$. This is line segment packing in the two objective case, hexagonal packing of circles in 3 objectives, hexagonal close packing of

spheres in 4 objectives, and so on [99].

□

Lemma L3. *Each surrogate dominates an amount of hypervolume on the Pareto front not less than* ϵ^{k-1}.

Proof.

P 1. If any dimension of the surrogate hyperprism is less than ϵ, the surrogate point is modified such that that dimension is equal to ϵ (Algorithm 6.1, lines 9 and 12).

P2. The minimum hypervolume of the Pareto front dominated by the surrogate is equal to the hypervolume of the smallest side of the surrogate hyperprism.

P 3. Since the lowest dimension of a surrogate hyperprism is ϵ, the smallest possible hypervolume dominated is ϵ^{k-1}.

□

L3 introduces one new problem that must be addressed: the Pareto front may have an extreme slope near a discovered Pareto point, such that the ϵ-modification of the surrogate point may dominate an exceedingly large amount of the Pareto front (not contained within the surrogate hyperprism). This is illustrated in Figure 6.2. While rare in practice, Theorem T3 addresses this issue.

Theorem T3. $\exists\ \epsilon\ \ni maxdist < \delta$

There exists an ϵ value such that the maximum distance between two neighboring discovered Pareto optimal points is less than δ.

Proof.

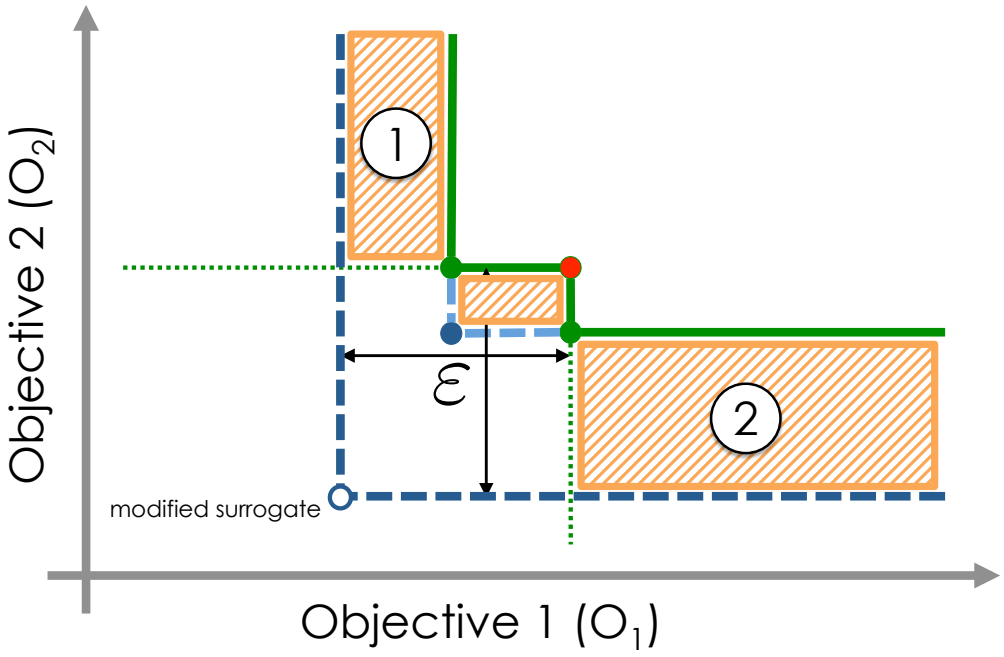

Figure 6.2: The need for T3. The Pareto front will pass through both green points, and the shaded orange areas. If, in region 1, $\frac{\partial O_2}{\partial O_1} \gg 0$, then the Pareto front will exit out of the top of the region, and a long distance will be covered before it becomes nondominated by the modified surrogate. On the other hand, in region 2, if $\frac{\partial O_1}{\partial O_2} \gg 0$, a similar problem arises.

P1. If a point is discovered that is dominated by S, the Pareto front is wholly dominated by S (L1).

P 2. If all surrogate hyperprisms are joined, the process is complete, and T4 will be satisfied.

P3. If the surrogate hyperprisms are not joined, ϵ may be reduced by a factor of 2, and areas of the Pareto front not within surrogate hyperprisms may become nondominated by S. ϵ can be reduced in this manner ad infinitum.

P4. As $\epsilon \rightarrow 0$, the amount of hyperspace that the modified surrogate dominates outside

of the surrogate hyperprism reduces to 0.

P5. As P4 occurs, points on the Pareto Front which were previously dominated by S will no longer be dominated by S. When ϵ is sufficiently small, a point will be discovered that lies within δ of a point in P_I^*.

□

Theorem T4. *Using Module $\mathcal{CC}$ guarantees the discovery of two Pareto optimal solutions within δ Euclidian distance from any point on the true Pareto front.*

Proof.

P1. The extension provably executes and produces a surrogate or modified surrogate. It does so in a finite number of iterations (L2).

P2. Each surrogate or modified surrogate covers part of the Pareto front, with a minimum size (L3).

P3. It will continue to fire using newly-introduced base points not already dominated by a surrogate or modified surrogate (L1) until the whole Pareto front is dominated by surrogates (L2).

P4. Once contained in a non-modified surrogate hyperprism, all points on the Pareto front are within δ of the base points of that hyperprism.

□

The implications of these two theorems are that the user can choose any δ value that suits their needs, and two Pareto optimal solutions will be found within that distance of every point on the Pareto front. This δ can be chosen to be arbitrarily small, and full coverage is still attained as ϵ decreases.

6.4 Domains of Study

In this work, we use the DTLZ-2 scalable test problem [37] with 3 and 10 objectives, and Kursawe's test problem (KUR) [70].

The DTLZ-2 problem is a problem that can have any number of objectives, with a globally concave Pareto front. A solution is described by a vector ($\mathbf{x} = \{x_1, x_2, \cdots, x_{k-1}, \mathbf{x_M}\}$) of length $k - 1 + m$, where $k - 1$ elements $(x_1, \cdots, x_{k-1})$ determine at what angles in the k dimensional objective space evaluation v will lie and the remaining m elements $(\mathbf{x_M})$ determine the distance from the origin at which v will lie. We use $k = 10$ and $size(\mathbf{x}_M) = 10$, for a 10-objective instantiation of the problem with 20 design variables; and a $k = 3$ and $size(\mathbf{x}_M) = 10$ for a visualizable 3-objective problem. The k functions to be minimized are:

$$\begin{aligned}
F_1(\mathbf{x}) &= (1+g)cos\left(x_1\frac{\pi}{2}\right)cos\left(x_2\frac{\pi}{2}\right)\cdots cos(x_{k-1}\frac{\pi}{2}) \\
F_2(\mathbf{x}) &= (1+g)cos\left(x_1\frac{\pi}{2}\right)\cdots cos\left(x_{k-2}\frac{\pi}{2}\right)sin\left(x_{k-1}\frac{\pi}{2}\right) \\
F_3(\mathbf{x}) &= (1+g)cos\left(x_1\frac{\pi}{2}\right)\cdots sin\left(x_{k-2}\frac{\pi}{2}\right) \\
&\vdots \\
F_k(\mathbf{x}) &= (1+g)sin\left(x_1\frac{\pi}{2}\right)
\end{aligned} \tag{6.2}$$

with each element of **x** remaining in the range [0:1], and:

$$g(\mathbf{x}_M) = \sum_{x_i \in \mathbf{x}_M} (x_i - 0.5)^2 \tag{6.3}$$

Deb et. al. recommend this problem "to investigate a multi-objective evolutionary algorithm's ability to scale up its performance in large numbers of objectives" [37], which is our purpose in using this problem. It also breaks assumption A3 intrinsically.

The KUR problem is a two-objective problem with a discontinuous, locally concave Pareto front. It takes the form:

$$\begin{aligned} F_1(\mathbf{x}) &= \sum_{i=1}^{2} \left[-10 \; exp\left((-0.2)\sqrt{x_i^2 + x_{i+1}^2} \right) \right] \\ F_2(\mathbf{x}) &= \sum_{i=1}^{3} \left[|x_i|^{0.8} + 5\sin(x_i)^3 \right] \end{aligned} \tag{6.4}$$

Where f_1 and f_2 are to be minimized by controlling the decision variables:

$$x_i \in [-5, 5] \quad ; \quad i \in \{1, 2, 3\} \tag{6.5}$$

We use this test problem to show the impact of the three extensions introduced in this work in an easily visualizable manner, and because it breaks assumption A4.

6.5 Results

We display results on multi-objective benchmark problems, to display each of the unique benefits conferred by the three extensions introduced in this paper. We also examine PaCcET and its extensions' performance on a 10-objective instantiation of DTLZ-2 [37].

In each of these cases, we use the original $\mathcal{TR} \Rightarrow \mathcal{LC} \Rightarrow \Xi$ formulation of PaCcET

in tandem with the extensions being tested, except in $\mathcal{RP}$, which replaces $\mathcal{LC}$. Unless otherwise noted, for Ξ we use an evolutionary algorithm with a population of 100 solutions for 1000 generations for each evolutionary algorithm, and binary tournament selection. This breaks A2. We do not seed P_I^*, breaking A1. Thus, all 4 assumptions A1-A4 are broken.

Guaranteed coverage of Pareto Front ($\mathcal{CC}$) Figure 6.3 shows the density of solutions produced by PaCcET on KUR. In this implementation, the top curve is mostly ignored. With $\mathcal{CC}$ ($\delta = 0.1$) (Figure 6.5), the entire Pareto front is more fairly covered with solutions.

Avoiding areas of the Pareto Front ($\mathcal{I}$) Figure 6.4 shows that the $\mathcal{I}$ extension successfully forces the algorithm away from areas of the Pareto front that were deemed

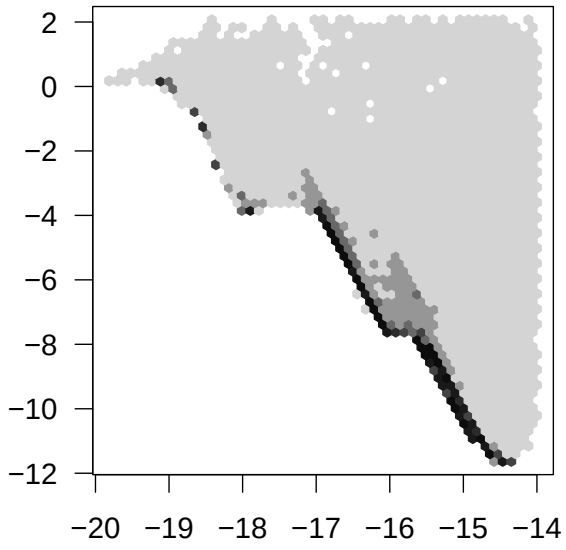

Figure 6.3: PaCcET solution densities in KUR on $\{F_1, F_2\}$ using $\mathcal{TR} - \mathcal{LC} - \Xi$.

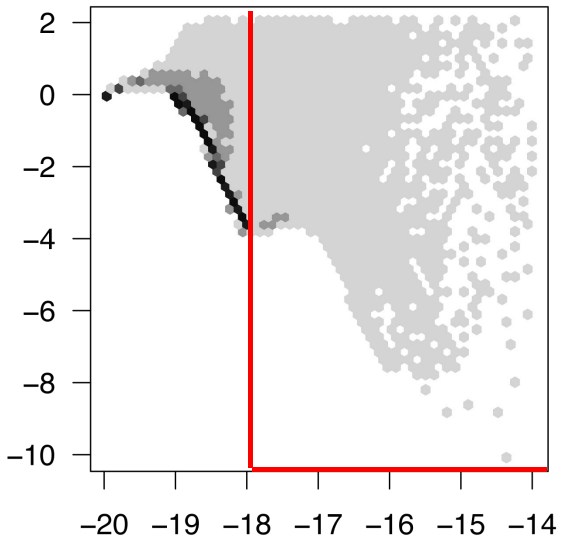

Figure 6.4: PaCcET solution densities in KUR on $\{F_1, F_2\}$ using $\mathcal{I} - \mathcal{TR} - \mathcal{LC} - \Xi$. Red lines denote the area dominated by the single point introduced by $\mathcal{I}$, which is avoided.

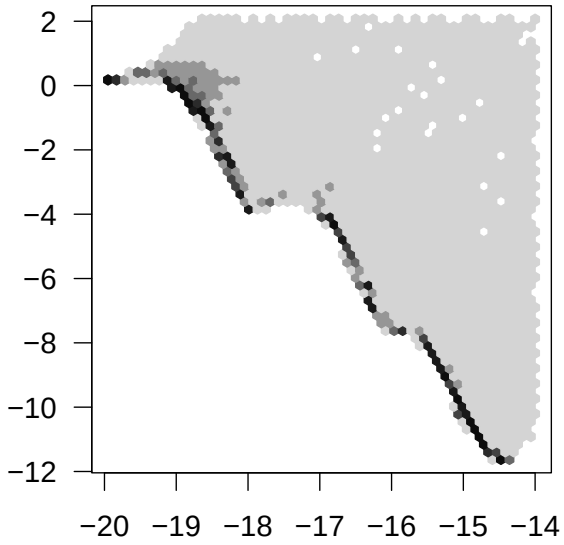

Figure 6.5: PaCcET solution densities in KUR on $\{F_1, F_2\}$ using $\mathcal{CC} - \mathcal{TR} - \mathcal{LC} - \Xi$, which encourages a more even spread of solutions across the Pareto front than without the use of the $\mathcal{CC}$ extension.

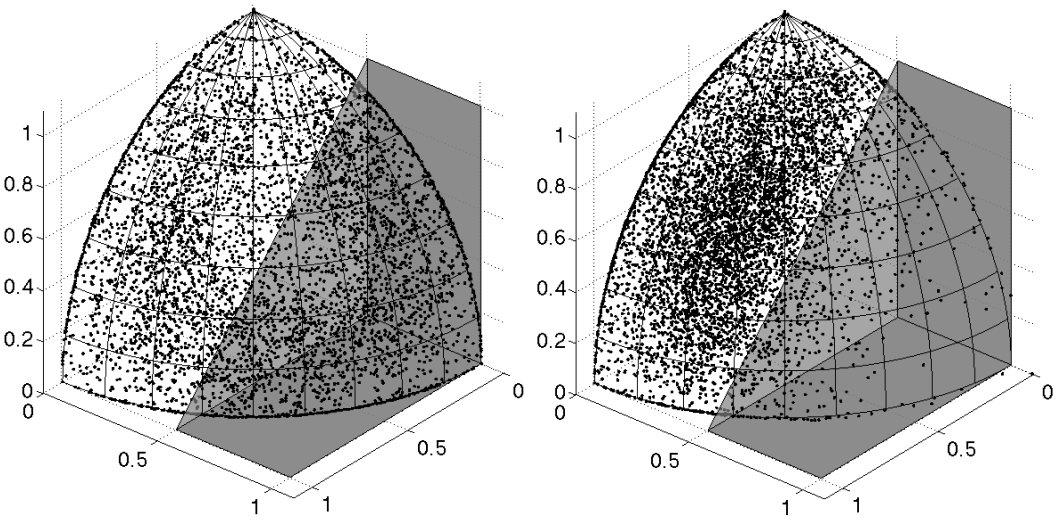

Figure 6.6: (Left) 3-Objective DTLZ-2. PaCcET without using $\mathcal{I}$ attains an even spread across the entire Pareto front. Shaded prism for comparison only. (Right) When $\mathcal{I}$ is used with one point, $\{0, 0.5, 0\}$, the area dominated by it (inside the shaded prism) is ignored, and computational effort is directed to other parts of the Pareto front.

undesirable. This produces the highest density of solutions on the upper curve that was originally ignored.

Figure 6.6 shows the performance of PaCcET with and without the $\mathcal{I}$ extension on the 3-objective DTLZ2 problem. Without the $\mathcal{I}$ extension, a fair spread across the entire Pareto front is attained. With the $\mathcal{I}$ extension, an area of the Pareto front can be ignored, and computational effort concentrated elsewhere.

Guided search ($\mathcal{RP}$) Figure 6.7 (left) shows how the reference point $\{0.2, 0.7\}$ (static in the transformed space) moves through the untransformed space as P_I^* is refined in three independent runs. The point starts in the right portion of the plot, and proceeds toward the left as time increases. The reference point eventually moves into the unattain-

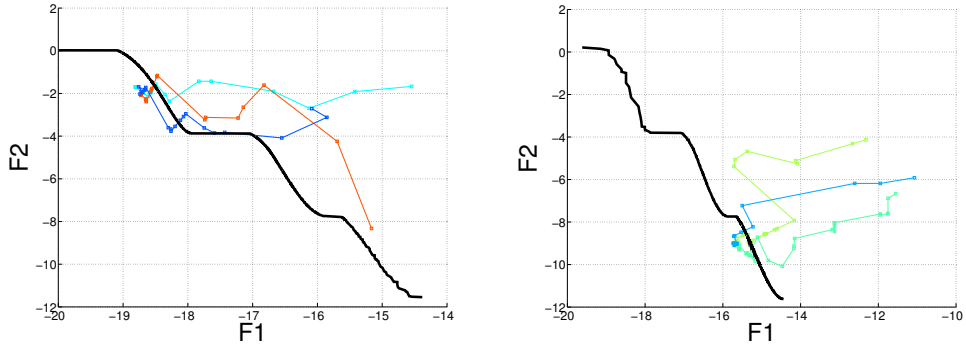

Figure 6.7: Movement of the reference point in the pre-$\mathcal{TR}$ space, for various independent runs, over 1000 iterations. (left) $ref = \{0.2, 0.7\}$; (right) $ref = \{0.7, 0.2\}$

able portion of the search space, guaranteeing Pareto optimal solutions [42], while requiring no previous knowledge of the shape of the true Pareto front.

Figure 6.7 (right) shows how the reference point $\{0.2, 0.7\}$ (static in the transformed space) moves through the untransformed space as P_I^* is refined in three independent runs. The point starts in the right portion of the plot, and proceeds toward the bottom-left as time increases.

Figure 6.8 shows the final reference points over 100 independent runs for each reference point. The reference point is always forced into the unattainable portion of the space.

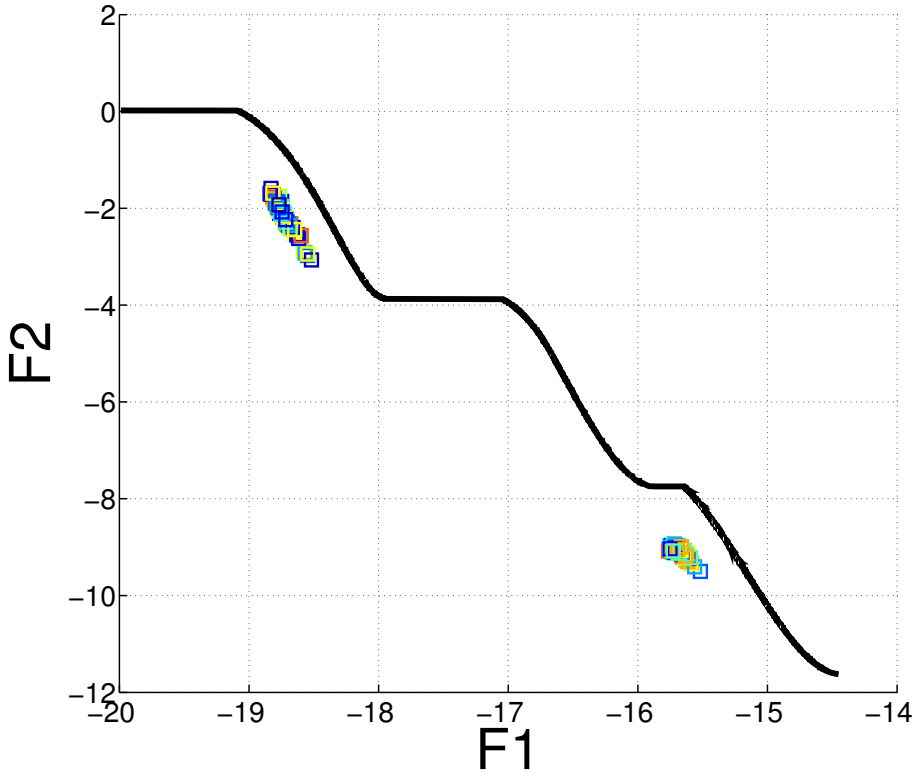

Figure 6.8: The final reference points for both described above over 100 independent runs for each reference point. Each final reference point is unattainable, guaranteeing Pareto optimal solutions.

Algorithm	**Mean % Dom.**	**St. Dev.**
PaCcET + $\mathcal{CC}$	83.39%	.2258%
PaCcET ($\mathcal{TR}$-$\mathcal{LC}$-Ξ)	80.95%	.3317%
NSGA-II	72.45%	.5970%
PaCcET (Sim. An.)	68.59%	1.3231%
SPEA2	3.78%	.1875%

Table 6.1: 10-objective DTLZ-2 problem performance. All instances of PaCcET perform highly, with the $\mathcal{CC}$ extension improving over the original PaCcET.

Many-Objective Problem (10 Objectives) We used the 10-objective instantiation of DTLZ-2 to evaluate the performance of NSGA-II, SPEA2, and three variations of PaCcET in a many-objective problem. This problem's true Pareto front is known to be bounded by the $\vec{0}$ and $\vec{1}$ vectors in $\mathbb{R}^{10}$. We compare the points generated by each algorithm to 10,000 uniform randomly generated vectors in that range.

Table 6.1 shows the mean and standard deviation (over 30 independent statistical runs) percent domination by each method. SPEA2 dominated only 3% of the randomly generated vectors. NSGA-II performs significantly better, dominating 72.45% of the vectors. PaCcET using an evolutionary algorithm performed better yet, and the $\mathcal{CC}$ extension improved over PaCcET alone. PaCcET with a simulated annealing algorithm (allowed as many global evaluations as the evolutionary algorithms) performed almost as well as NSGA-II.

6.6 Conclusions

In this work we introduced three extensions to PaCcET, a multi-objective transformation that forces the Pareto front to become non-concave so that computationally simple optimizers may solve complex multi-objective problems. The $\mathcal{I}$ and $\mathcal{CC}$ extensions use a surrogate set to steer the solution through the objective space manually and automatically, respectively. We provided a guarantee for full coverage of the Pareto front using $\mathcal{CC}$. The $\mathcal{RP}$ extension lets reference points be used with PaCcET, without the need to know the shape of the attainable space. Finally, we tested PaCcET in a 10-objective problem, showing that it scales into many-objective problems, dominating more sampled hypervolume than NSGA-II or SPEA2.

Chapter 7 – Multiagent PaCcET

In the previous chapters, we have developed the PaCcET Transformation (Chapter 5), and provided three extensions to allow a system designer to bias which areas of the Pareto front an optimizer using PaCcET will favor (Chapter 6). We identified that it executes quickly and provides good solution quality in the single-agent case, which makes it a good candidate for multiagent systems. In this chapter, we incorporate the concept of credit assignment into the PaCcET framework.

For this chapter, it is therefore necessary to be familiar with the PaCcET Transformation (Chapter 5), cooperative coevolutionary algorithms (Section 2.2.3), and the concepts of Pareto optimality (Section 2.3), and credit assignment (Section 2.2.2). We use empirical attainment functions (Section 2.3) to compare this multiagent implementation of PaCcET against the multiagent NSGA-II algorithm developed previously (Chapter 4) and a linear combination of objectives (Section 2.4.1). We compare performance on the multi-objective rover problem originally introduced in Chapter 4, which we reproduce here for completeness. Note that there are differences in parameters (20 rovers instead of 10 previously; 40 POIs instead of 50 previously) that make this a more difficult coordination problem to solve than the previous version.

7.1 Multi-Objective Rover Domain

We perform a series of experiments with the different algorithms in a continuous rover domain. In this domain, a team of 20 rovers must work together to observe a set of 40 heterogeneous points of interest (POIs) on a 500 x 500 Euclidian plane, which have vector-valued importance $\vec{V}$ of length $k = 2$ objectives. Each of the elements of this vector represents a type of scientific data that the rovers may collect from a POI, which acts as an objective. We desire to maximize the system (global) performance vector $\vec{G}$.

Rover Motion Each rover policy selects a series of waypoints (μ_x, μ_y), which are then sent to a low-level planner that determines the path the rover should take to reach that waypoint. After that waypoint is reached (with some variance σ_x and σ_y), the rover takes an observation of any POIs that it can observe, before proceeding to the next waypoint. We assume the rovers can localize en route, so variance from the intended waypoints does not increase with time.

Observations The quality of the observations created by the rovers is a function of the Euclidian distance at which they observe the POI, δ:

$$\delta(x, y) = \max\{\|x - y\|^2, d^2\} \tag{7.1}$$

where x is the location of one object, y is the location of a second object, and d is the minimum observation distance (to prevent singularity, we use $d = 1$). This then allows us to calculate the vector of global evaluations $\vec{G}$ based on rover locations L_R and POI

locations L_P:

$$\vec{G} = \sum_p \frac{\vec{V}_p}{\min_r \delta(L_{P,p}, L_{R,r})} \tag{7.2}$$

where "$\min_r \delta(L_{P,p}, L_{R,r})$" is the minimum distance between any rover and POI p. Applying the difference evaluation (Equation 2.2) directly to Equation 7.2, we can calculate the difference evaluation vector $\vec{D}_r$ for rover r:

$$\vec{D}_r = \sum_p \frac{\vec{V}_p}{\min_r \delta(L_{P,p}, L_{R,r})} - \sum_p \frac{\vec{V}_p}{\min_{r' \neq r} \delta(L_{P,p}, L_{R,r'})} \tag{7.3}$$

Note, only when rover r is the rover whose observation of a POI is used in the global evaluation will $\vec{D}_r$ be non-zero.

7.2 Naive implementation of credit assignment within PaCcET

As PaCcET is built on the use of a linear combination, it may seem reasonable that credit assignment may be incorporated into PaCcET in much the same was as it was into a linear combination in Chapter 3. However, because PaCcET transforms the objective space before the linear combination of objectives takes place, the linear combination in the PaCcET Λ^τ space is a non-linear combination in the original Λ space. Couple this with the fact that the PaCcET transformation changes as the current best estimate of the Pareto front (P_I^*) is refined, and the potential difficulties for simply using the same methodology for the incorporation of credit assignment are many. We first tested PaCcET with credit assignment in a multiagent system using the multi-objective rover domain (Section 7.1). We compare against multiagent NSGA-II as we developed in

Algorithm 7.1: Naive algorithm

```
1 simulate rover domain in coevolutionary manner, with randomly-formed teams;
2 perform difference evaluations d⃗_{a,i} for all agents a, for all population members i;
3 foreach Population a do
4 |   foreach Population member i do
5 |   |   fitness_i = PaCcET_a(d⃗_{a,i}) // P_I^* independent by population
6 |   end
7 end
8 remove less-fit members of populations;
9 replenish populations with mutated copies of members;
```

Chapter 4. Figure 7.2 shows the EAF attained by NSGA-II. A naive PaCcET implementation (Algorithm 7.1) in Figure 7.1 shows that a very conspicuous pattern arises.

No combination of difference evaluations and PaCcET attained any better performance than that shown in Figure 7.1. We experimented with various orders of operations, as was previously shown to be important in some MOEAs, and while some proved to be destructive to system performance, none improved performance beyond this. However, upon closer inspection, there is a readily available explanation to the poor performance that PaCcET attains in this multiagent system. First, note that the performance tends toward the center of the Pareto front. This is because each agent individually settles on a policy that is good at achieving performance on objective 1 or objective 2, since each POI has only one type of data available. On average, then, the team as a whole tends toward the center of the Pareto front as each agent optimizes one or the other objective. Achieving performance on either end of the Pareto front requires some level of coordination among the team, as they all must choose to optimize the same objective at the same time.

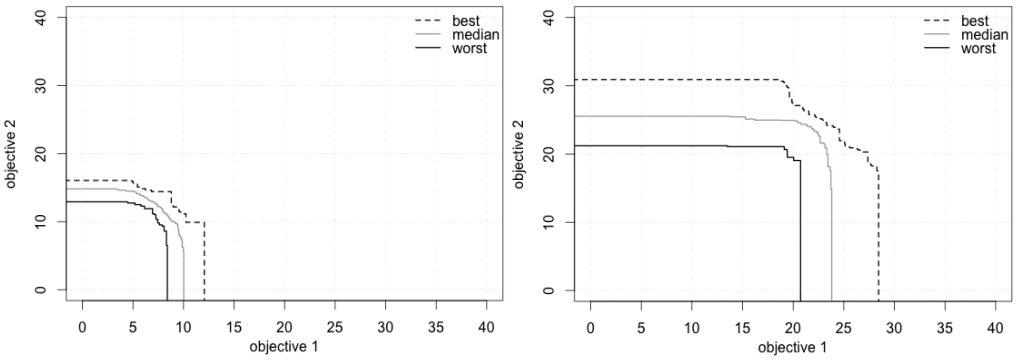

Figure 7.1: A naive implementation of PaCcET in the multi-objective rover domain using the global evaluation (left) and difference evaluations (right). Even with difference evaluations, the agents trained with PaCcET only discover the central portion of the Pareto front.

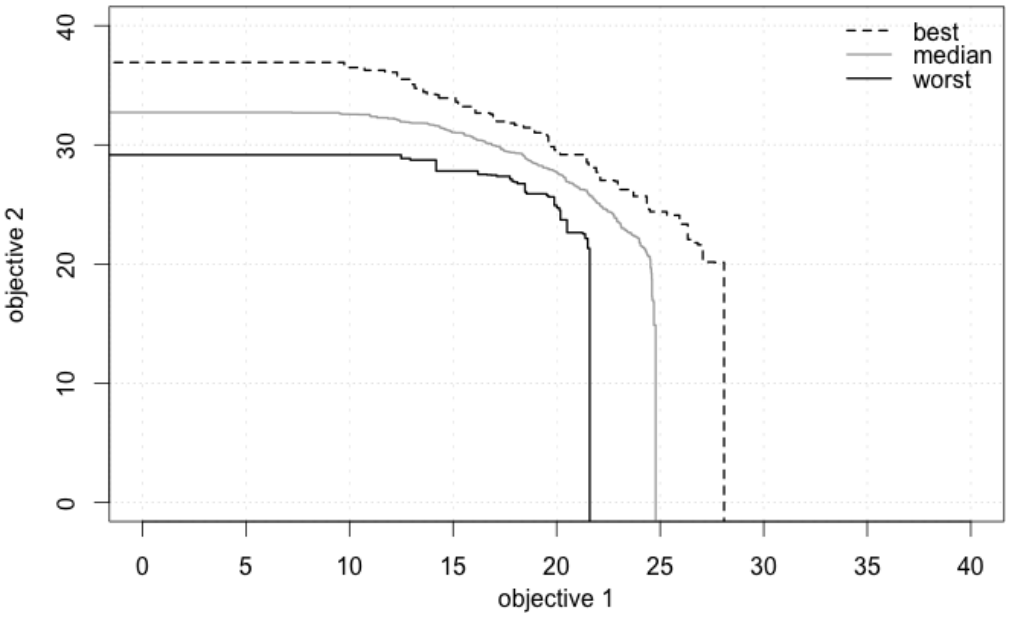

Figure 7.2: Multiagent NSGA-II EAF.

7.3 The Problem with Nash Equilibria

So why does this not happen? It does appear that there is some variation along the Pareto front, but this variation is only as large as the impact a single agent can make.

For any variation further along the Pareto front, multiple agents would have to develop policies to optimize the same objective at the same time — and if any agent developed a policy to optimize the other objective at the same time, the team would again tend toward the center of the Pareto front. Furthermore, it is not even as simple as this: an agent can't simply to change its contribution in the objective (Λ) space; it only can change its policy within the policy space (Ω), so even if it had incentive to change its contribution in the Λ space, it would have to develop a policy to achieve this. Beyond these difficulties, this process does not happen incrementally because of the concept of Nash Equilibria [81]. Even if it were instantaneously able to do so, each agent does not have an incentive to change its policy to move along the Pareto front, because it simply moves to another point very near the current P_I^*, which is by definition valued equally by a linear combination in the PaCcET τ space. Thus, after these one-agent-wide variations along the Pareto front have been discovered once, there is no incentive to go that direction again.

This is compounded by a related problem: the agents cannot simply choose to change the contribution that they make to each of the objectives; they must develop policies that map to their contributions, while interacting with all of the other agents. In the setting of the multi-objective rover problem, this means that the agent's policy must discover a point of interest that is not yet discovered by the other agents, which creates a difficult coordination problem, on top of the multi-objective considerations.

7.4 Two-Level PaCcET for Multiagent Systems

There exist a number of possible solutions to this: stabilizing the PaCcET process so that it must discover a point many times before adding it to P_I^*; or updating P_I^* only incrementally so that it takes many iterations for a newly discovered point to be fully included in P_I^*; or using a multiagent hall-of-fame method [25] keeping multiple halls of fame, one associated with every point in P_I^*; all of these bear some merit, and are grounds for future research. In the section, however, we take a very simple approach: we simplify the mapping between $\Omega \longrightarrow \Lambda$ by first using PaCcET with Difference evaluations to generate successful policies, storing each of these policies in a shared library, and then conducting another higher-level optimization that simply chooses policies from this library of successful policies. This process is clarified in the following paragraphs, and described in Algorithms 7.2 and 7.3.

The lower-level algorithm functions similarly to the naive implementation of credit assignment within PaCcET discussed in Section 7.2. This method is highly capable of generating successful individual agent policies, as was shown in Figure 7.1. It simply was a lack of team-wide coordination that prevented this method from achieving high performance on the team-level. In order to generate a variety of different policies, we perform a series of rapid random restarts. This allows the agents to develop many $\Omega \longrightarrow \Lambda$ mappings that, while they may achieve similar performance in Λ, come from different portions of Ω, giving the team a wide variety of policies to choose from at the higher level.

The higher-level algorithm is a combinatorial optimizer that chooses successful poli-

Algorithm 7.2: Lower-level algorithm

```
   Data: gen_per_restart
 1 foreach Random Restart R do
 2     initialize num_rovers populations a of pop_size solutions;
 3     carry_capacity ← pop_size/2;
 4     foreach Generation  g ∈ 1 : gen_per_restart do
 5         simulate rover domain in coevolutionary manner, with randomly-formed
           teams;
 6         perform difference evaluations d⃗_{a,i} for all agents a, for all population
           members i;
 7         foreach Population a do
 8             foreach Population member i do
 9                 fitness_i = PaCcET_a(d⃗_{a,i}) // P_I^* maintained independently
                       for each population
10             end
11         end
12         remove less-fit members of populations;
13         if g=gen_per_restart then
14             add all remaining policies to library L;
15         else
16             replenish populations with mutated copies of members;
17         end
18     end
19 end
```

cies from the policy library, and maintains a population of sets of policies. The set, in this way, acts as the individual in a higher-level evolutionary algorithm. This higher level was conducted in parallel with the low-level algorithm, such that each time the low level performed a new random restart, the policies developed by the previous restart are added to the current library. Because this quickly adds to the size of the library, we only perform a small number of random restarts, and this provides enough variety to achieve good performance. There are refinements that can be made to this process to increase

Algorithm 7.3: Higher-level algorithm

```
Data: Library of candidate policies L
initialize pop_size solutions // each solution consists of indexes
    corresponding to policies from the library L
foreach generation g ∈ 1 : total_generations do
    if lower-level algorithm random-restarted then
        update library L from lower-level algorithm;
    end
    foreach solution s in population do
        simulate s in rover domain;
        evaluate team fitness T_s;
    end
    remove less-fit solutions;
    replenish populations with mutated copies of solutions;
end
```

the generalizability, which we relegate to future work.

Figure 7.3 shows the empirical attainment achieved by this two-level algorithm, which can be compared to Figure 7.2. The two achievement functions show that the performance with this two-level algorithm is very readily comparable to the performance achieved by the multiagent NSGA-II algorithm. The performance benefits offered by PaCcET, discussed in previous chapters, carry through into this, so this comparable performance is achieved with significantly reduced computation being dedicated to the fitness evaluation step.

7.5 Two-Level PaCcET with Reference Points

Finally, in this section we incorporate the use of reference points into the two-level algorithm developed in the previous section. We segment the population into three sub-

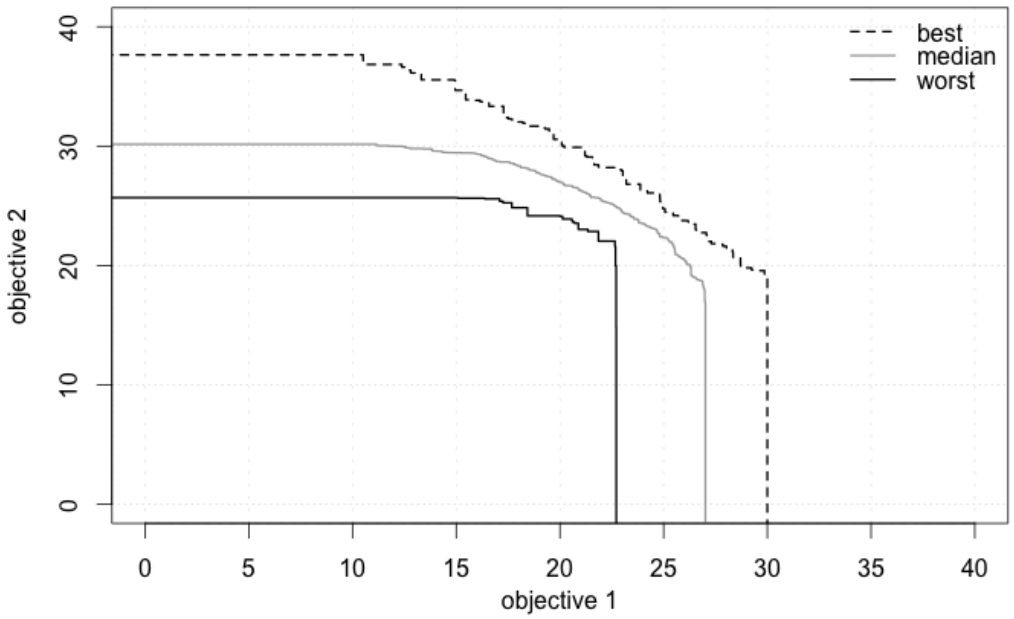

Figure 7.3: Multiagent PaCcET EAF.

populations, each of which are evaluated on a different reference point in the PaCcET space. In this way we can force a fairer coverage of the Pareto front. Two of the reference points that we create emphasize the ends of the Pareto front, and the third emphasizes the middle:

$$RP_1 = \{0, 0.9\} \tag{7.4}$$

$$RP_2 = \{0.9, 0\} \tag{7.5}$$

$$RP_3 = \{0.45, 0.45\} \tag{7.6}$$

By each of these sub-populations attempting to reach a different portion of the Pareto

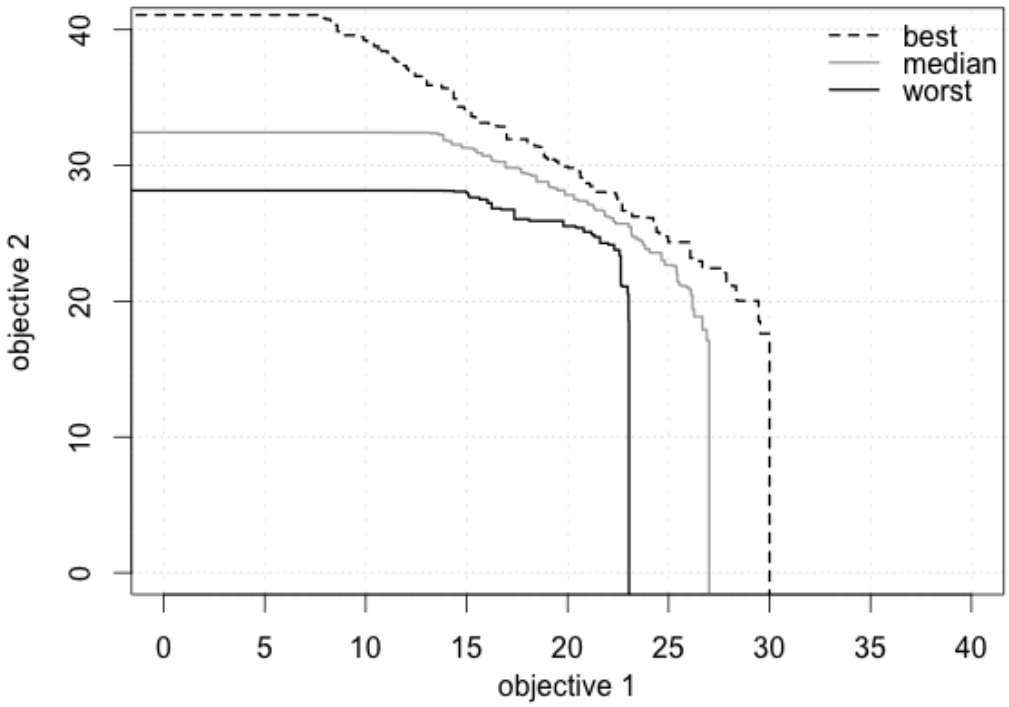

Figure 7.4: Multiagent PaCcET with three reference points: EAF.

front, the population as a whole attains good coverage over the whole Pareto front. The EAF they produce can be seen in Figure 7.4. This produces superior performance to NSGA-II and the two-level PaCcET algorithm not using reference points. In the case of a more complex, tougher to cover Pareto front, or a higher-objective problem, additional reference points could be specified throughout the PaCcET space. As long as each of the reference points specified meets the criteria that the (non-negative) components sum to less than one, the reference point in the original space will move to the unattainable portion of the space. Specifying such a set of reference points in the PaCcET space, with no previous knowledge of the Pareto front in the problem space, creates a set that forms to the Pareto front in the original space.

7.6 Conclusion

In this chapter, we have discussed the incorporation of PaCcET into a multiagent system. We demonstrated that the formulation of PaCcET creates a predisposition for Nash Equilibria to cause a lack of solution spread along the Pareto front, and that because of a lack of incentive to deviate from these Nash strategies, the agents do not coordinate to find additional solutions on other portions of the Pareto front in the team objective space, despite achieving highly based on the difference evaluation in the PaCcET space. The team, on average, tended toward solutions in the central area of the Pareto front, and did not create a broad solution that covered from extreme to extreme.

To remedy these problems, we introduced a two-level algorithm in which the lower-level algorithm used difference evaluations and PaCcET to produce a library of high-performing agent policies over a short series of rapid random restarts, and the higher-level algorithm performed a combinatorial optimization over that library of policies developed by the lower-level algorithm. This two-level algorithm offers significantly reduced computation compared to NSGA-II, while offering similar performance. It offers a reasonable set of tradeoff solutions, from which a compromise solution may be selected. We then improved upon these results by using reference points in the PaCcET space, with a segmented population seeking different portions of the Pareto front. This allowed each sub-population to specialize in reaching solutions on a particular portion of the Pareto front, with no need for prior knowledge about the space of the Pareto front.

Future work in this area consists of refining the way in which the library of agent policies is developed and maintained, as well as developing alternative methods for

maintaining diversity in the agent populations in a more traditional evolutionary algorithm, possibly by mechanisms like hall of fame or leniency-based approaches.

Chapter 8 – Conclusion

As large, complex systems become increasingly common in the modern world, it will be important to use these systems to their fullest. The study of multiagent systems seeks to understand how the components of these systems interact and can be controlled to maximize their usefulness, but this is not enough in itself. These systems can rarely be considered as having a single objective to be myopically optimized; instead, multiple objectives need to be considered simultaneously.

Despite this, the intersection between these two fields, multi-objective optimization and multiagent systems, has received scant attention until now. This dissertation serves to provide a foothold from which additional adaptive multiagent multi-objective research can be launched. We have:

- derived methods for automatically assigning credit for a team's success or failure to members of that team in the presence of multiple objectives
- derived multiagent equivalents to state-of-the-art multi-objective algorithms
- developed PaCcET, a fast, effective multi-objective algorithm that outperforms state-of the art MOEAs in as little as one tenth of the computation time
- theoretically proved that PaccET will produce Pareto optimal results that cover the entire Pareto front to an arbitrarily fine resolution

- developed a framework for integrating this fast multi-objective algorithm into multiagent systems

The remainder of this chapter discusses how the content of each of the previous chapters developed these contributions.

In Chapter 3, we derived methods for automatically assigning credit for a team's success or failures in a multi-objective context. This resulted in system performance that dominated an approach not considering credit assignment, and while choosing an *a priori* aggregation that was well-suited to the problem was found to be important, using difference evaluations to assign credit improved performance regardless of the scalarization chosen.

In Chapter 4, we introduced an algorithm which produced equivalent total rankings of populations to the successful multi-objective algorithm NSGA-II, which was better-defined for incorporating difference evaluations. We showed that difference evaluations and NSGA-II have orthogonal benefits, and when used in tandem, produce results that are superior to using either alone while neglecting the other. We showed that the order in which they are combined is of paramount importance, and combining them the wrong way can destroy system performance. We identified the underlying mechanisms for this, and offered guidelines for future use of difference rewards in other multi-objective evolutionary algorithms.

In Chapter 5, we produced PaCcET, a fast multi-objective algorithm that performs as well or better than other state-of-the-art multi-objective methods with a radical reduction of computation time, and provided two theoretical guarantees: (i) that PaCcET produces Pareto optimal solutions, and (ii) despite being built on a linear combination, which

cannot find concave areas of the Pareto front, PaCcET can discover solutions even in these concave areas.

In Chapter 6, we identified a number of weaknesses associated with the PaCcET framework, and offered three extensions to offset these weaknesses, and allow a system designer more control over the behavior of the algorithm. We provided an extension that guarantees complete coverage of the Pareto front to an arbitrarily fine resolution, and theoretically proved this guarantee.

Finally, in Chapter 7, we introduced PaCcET into a multiagent system. We discovered that, alone, PaCcET did not perform as well as the multiagent implantation of NSGA-II that we produced in Chapter 4, even after incorporating difference evaluations. We then created a two-stage algorithm, in which a team of agents coevolved using separate PaCcET instances to produce successful agent policies, which are then passed to a higher-level algorithm, which performed a multi-objective evolutionary algorithm using PaCcET to combinatorially optimize which agents should be used with each other.

This area remains a fertile ground for future research. There exist many other *a priori* multi-objective methods and many other multi-objective evolutionary algorithms that have never been studied in the context of credit assignment in multiagent systems. Additionally, other evaluation-shaping techniques from the multiagent community have not been studied in the context of multiple objectives. The concepts of *alignment* and *sensitivity*, which are related to the difference evaluation, have never been studied in the context of multiple objectives, either. Additionally, PaCcET must be tested in a wide variety of standard multi-objective problems, and the two-level algorithm used for PaCcET in a multiagent system could be altered in a number of ways to increase its performance

or generalizability.

The demand for fast, effective multi-objective optimization suited for use with distributed systems will only grow with time, and the contributions of this dissertation provide a platform from which future multiagent multi-objective research can continue.

Bibliography

[1] H.A. Abbass, R. Sarker, and C. Newton. Pde: a pareto-frontier differential evolution approach for multi-objective optimization problems. In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, volume 2, pages 971 –978 vol. 2, 2001.

[2] M. A. Abido. Environmental/economic power dispatch using multiobjective evolutionary algorithms. *IEEE Transactions on Power Systems*, 18(4):1529–1537, 2003.

[3] A. K. Agogino and K. Tumer. Analyzing and visualizing multi-agent rewards in dynamic and stochastic domains. *Journal of Autonomous Agents and Multiagent Systems*, 17(2):320–338, 2008.

[4] A. K. Agogino and K. Tumer. Analyzing and visualizing multiagent rewards in dynamic and stochastic environments. *Journal of Autonomous Agents and Multiagent Systems*, 17(2):320–338, 2008.

[5] F. Altiparmak, M. Gen, L. L., and R. Paksoy. A genetic algorithm approach for multi-objective optimization of supply chain networks. *Computers and Industrial Engineering*, 51(1):196 – 215, 2006.

[6] W. B. Arthur. Inductive reasoning and bounded rationality (the El Farol Problem). *American Economic Review*, 84(406), 1994.

[7] T. W. Athan and P. Y. Papalambros. A note on weighted criteria methods for compromise solutions in multi-objective optimization. *Engineering Optimization*, 27(155-176), 1996.

[8] M. Atiquzzaman, S.-Y. Liong, and X. Yu. Alternative decision making in water distribution network with nsga-ii. *Journal of water resources planning and management*, 132(2):122–126, 2006.

[9] J. Bader and E. Zitzler. Hype: An algorithm for fast hypervolume-based many-objective optimization. *Evolutionary Computation*, 19(1):45–76, March 2011.

[10] R. Balling, J. Taber, M. Brown, and K. Day. Multiobjective urban planning using genetic algorithm. *Journal of Urban Planning and Development*, 125(2):86–99, 1999.

[11] E. G Bekele and J. W. Nicklow. Multi-objective automatic calibration of SWAT using NSGA-II. *Journal of Hydrology*, 341(3):165–176, 2007.

[12] I. Berrada, J. A. Ferland, and P. Michelon. A multi-objective approach to nurse scheduling with both hard and soft constraints. *Socio-Economic Planning Sciences*, 30(3):183 – 193, 1996.

[13] N. Beume, B. Naujoks, and M. Emmerich. SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653 – 1669, 2007.

[14] J. Branke, K. Deb, H. Dierolf, and M. Osswald. Finding knees in multi-objective optimization. In *Parallel Problem Solving from Nature - PPSN VIII*, volume 3242 of *Lecture Notes in Computer Science*, pages 722–731. Springer Berlin / Heidelberg, 2004.

[15] E. K. Burke, P. D. Causmaecker, G. D. Maerea, J. Mulder, M. Paelinck, and G. V. Berghe. A multi-objective approach for robust airline scheduling. *Computers and Operations Research*, 2009.

[16] B. Cakir, F. Altiparmak, and B. Dengiz. Multi-objective optimization of a stochastic assembly line balancing: A hybrid simulated annealing algorithm. *Computers and Industrial Engineering*, 60(3):376 – 384, 2011.

[17] G. Chalkiadakis and C. Boutilier. Coordination in multiagent reinforcement learning: A bayesian approach. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-03)*, Melbourne, Australia, July 2003.

[18] C.-L. Chen and W.-C. Lee. Multi-objective optimization of multi-echelon supply chain networks with uncertain product demands and prices. *Computers and Chemical Engineering*, 28(6–7):1131 – 1144, 2004.

[19] S. Chien, A. Barrett, T. Estlin, and G. Rabideau. A comparison of coordinated planning methods for cooperating rovers. *International Conference on Autonomous Agents*, 2000.

[20] S. Chien and J. Doubleday et. al. Combining space-based and in-situ measurements to track flooding in thailand. *Geoscience and Remote Sensing*, 2011.

[21] H. Chui and A. Rangarajan. A new point matching algorithm for non-rigid registration. *Computer Vision and Image Understanding*, 89(2):114–141, 2003.

[22] S. E. Cieniawski, J. Wayland Eheart, and S. Ranjuthan. Using genetic algorithms to solve a multiobjective groundwater monitoring problem. *Water Resources Research*, 31(2):399–409, 1995.

[23] C. A. C. Coello. A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information Systems*, 1(3):269–308, 1999.

[24] C. A. C. Coello and A. D. Christiansen. Multiobjective optimization of trusses using genetic algorithms. *Computers and Structures*, 75(6):647–660, 2000.

[25] M. Colby and K. Tumer. Shaping fitness functions for coevolving cooperative multiagent systems. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 425–432. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[26] M. Colby and K. Tumer. An evolutionary game theoretic analysis of difference evaluation functions. In *Proceedings of Genetic and Evolutionary Computation Conference (GECCO)*, 2015.

[27] B. L. Crowe. The tragedy of the commons revisited. *Science*, 166:1103–1107, 1969.

[28] D. Cvetkovic and I. Parmee. Evolutionary design and multi–objective optimisation. *6th European Congress on Intelligent Techniques and Soft Computing*, pages 397–401, September 1998.

[29] S. Damiani, G. Verfaillie, and M.-C. Charmeau. An earth watching satellite constellation: How to manage a team of watching agents with limited communications. *Autonomous Agents and Multiagent Systems*, 2005.

[30] D. Daniel, S. Oussedik, and P. Stephane. Airspace congestion smoothing by multi-objective genetic algorithm. In *Proceedings of the 2005 ACM symposium on Applied computing*, pages 907–912. ACM, 2005.

[31] I. Das and J. E. Dennis. A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems. *Structural Optimization*, pages 63–69, 1997.

[32] K. Deb. Multi-objective genetic algorithms: Problem difficulties and construction of test problems. *Evolutionary Computation*, 7(3):205–230, 1999.

[33] K. Deb. Multi-objective optimization. In Edmund K. Burke and Graham Kendall, editors, *Search Methodologies*, pages 273–316. Springer US, 2005.

[34] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast elitist multi-objective genetic algorithm: NSGA-II. *Evolutionary Computation*, 6:182–197, 2002.

[35] K. Deb, A. Pratap, and S. Moitra. Mechanical component design for multiple ojectives using elitist non-dominated sorting ga. In *Parallel Problem Solving from Nature*, pages 859–868. Springer, 2000.

[36] K. Deb and J. Sundar. Reference point based multi-objective optimization using evolutionary algorithms. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 635–642, New York, NY, USA, 2006. ACM.

[37] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable test problems for evolutionary multi-objective optimization. Technical report, ETH Zurich, 2001.

[38] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable multi-objective optimization test problems. In *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*, volume 1, pages 825 –830, may 2002.

[39] K. Deb and S. Tiwari. Omni-optimizer: A generic evolutionary algorithm for single and multi-objective optimization. *European Journal of Operational Research*, 185(3):1062 – 1087, 2008.

[40] S. Devlin, L. Yliniemi, D. Kudenko, and K. Tumer. Potential-based difference rewards for multiagent reinforcement learning. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*, pages 165–172, Richland, SC, 2014. International Foundation for Autonomous Agents and Multiagent Systems.

[41] P. Dorato, W. Yang, and C. Abdallah. Robust multi-objective feedback design by quantifier elimination. *Journal of Symbolic Computation*, 24(2):153 – 159, 1997.

[42] L. Duckstein. Multiobjective optimization in structural design: The model choice problem. In E. Atrek, R.H Gallagher, K. M. Ragsdell, and O. C. Zeinkiewicz, editors, *New Directions in Optimum Structural Design*, pages 459–481. John Wiley and Sons, 1984.

[43] A. A. Economides and J. A. Silvester. Multi-objective routing in integrated services networks: A game theory approach. In *IEEE Infocom '91: Proceedings of the Conference on Computer Communication*, volume 3, 1991.

[44] F. Y. Edgeworth. *Mathematical Psychics: An essay on the application of mathematics to moral sciences*. C. Kegan Paul and Company, 1881.

[45] R. R. Egudo. Efficiency and generalized convex duality for multiobjective programs. *Journal of Mathematical Analysis and Applications*, 1989.

[46] T. Estlin and S. Chien et al. Coordinating multiple spacecraft in joint science campaigns. *i-SAIRAS*, August 2010.

[47] G. W. Evans. An overview of techniques for solving multiobjective mathematical programs. *Management Science*, 30(11):1268–1282, 1984.

[48] C. M. Fonseca and P. J. Fleming. On the performance assessment and comparison of stochastic multiobjective optimizers. *Lecture Notes in Computer Science*, 1141:584–593, 1996.

[49] C. M. Fonseca, A. P. Guerreiro, M. Lopez-Ibanez, and L. Paquete. On the computation of the empirical attainment function. *Lecture Notes in Computer Science*, 6576:121–135, 2011.

[50] D. E. Goldberg and K. Deb. A comparative analysis of selection schemes used in genetic algorithms. *Urbana*, 51:61801–2996, 1991.

[51] C. Guestrin, M. Lagoudakis, and R. Parr. Coordinated reinforcement learning. In *Proceedings of the 19th International Conference on Machine Learning*, 2002.

[52] G. Hardin. The tragedy of the commons. *Science*, 162:1243–1248, 1968.

[53] G. Hassan and C. D. Clack. Robustness of multiple objective GP stock-picking in unstable financial markets. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 1513–1520. ACM, 2009.

[54] R. A. Hassan and W. A. Crossley. Multi-objective optimization of communication satellites with two-branch tournament genetic algorithm. *Journal of spacecraft and rockets*, 40(2):266–272, 2003.

[55] S. M. K. Heris and H. Khaloozadeh. Open-and closed-loop multiobjective optimal strategies for HIV therapy using NSGA-II. *Biomedical Engineering, IEEE Transactions on*, 58(6):1678–1685, 2011.

[56] J. G. Herrero, A. Berlanga, and J. M. M. Lopez. Effective evolutionary algorithms for many-specifications attainment: application to air traffic control tracking filters. *Evolutionary Computation, IEEE Transactions on*, 13(1):151–168, 2009.

[57] T. Hiroyasu, M. Miki, S. Nakayama, and Y. Hanada. Multi-objective optimization of diesel engine emissions and fuel economy using SPEA2+. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 2195–2196, 2005.

[58] C. Horoba and F. Neumann. Benefits and drawbacks for the use of epsilon-dominance in evolutionary multi-objective optimization. Technical Report CI-248/08, Technische Universitat Dortmund Reihe Computational Intelligence Collaborative Research Center 531, May 2008.

[59] J. Hu and M. P. Wellman. Multiagent reinforcement learning: Theoretical framework and an algorithm. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 242–250, June 1998.

[60] B. Huang, B. Buckley, and T.-M. Kechadi. Multi-objective feature selection by using nsga-ii for customer churn prediction in telecommunications. *Expert Systems with Applications*, 37(5):3638–3646, 2010.

[61] C. Igel, N. Hansen, and S. Roth. Covariance matrix adaptation for multi-objective optimization. *Evolutionary Computation*, 15(1):1–28, 2012/04/24 2007.

[62] S. Jeyadevi, S. Baskar, C.K. Babulal, and M Willjuice Iruthayarajan. Solving multiobjective optimal reactive power dispatch using modified NSGA-II. *International Journal of Electrical Power & Energy Systems*, 33(2):219–228, 2011.

[63] Y. Jin, M. Olhofer, and B. Sendhoff. Dynamic weighted aggregation for evolutionary multi-objective optimization: Why does it work and how? Technical report, Honda Research and Development, Europe, 2002.

[64] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 1996.

[65] H. Kanoh and K. Hara. Hybrid genetic algorithm for dynamic multi-objective route planning with predicted traffic in a real-world road network. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 657–664, New York, NY, USA, 2008. ACM.

[66] S. T. Khu and H. Madsen. Multiobjective calibration with pareto preference ordering: An application to rainfall-runoff model calibration. *Water Resources Research*, 41(3), 2005.

[67] I.Y. Kim and O.L. de Weck. Adaptive weighted-sum method for bi-objective optimization: Pareto front generation. *Structural and Multidisciplinary Optimization*, 29:149–158, 2005.

[68] M. Knudson and K. Tumer. Coevolution of heterogeneous multi-robot teams. *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, 2010.

[69] J. Kou, S. Xiong, H. Liu, and X. Zong. Particle swarm and NSGA-II based evacuation simulation and multi-objective optimization. *International Conference on Natural Computation*, 2011.

[70] F. Kursawe. A variant of evolution strategies for vector optimization. *Parallel Problem Solving from Nature*, pages 193–197, 1991.

[71] R. T. Marler and J. S. Arora. The weighted sum method for multi-objective optimization: new insights. *Structural and Multidisciplinary Optimization*, 2009.

[72] R.T. Marler and J. S. Arora. Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, 26:369–395, 2004.

[73] M. J. Mataric. Reward functions for accelerated learning. In *Machine Learning: Proceedings of the Eleventh International Conference*, pages 181–189, San Francisco, CA, 1994.

[74] M. J. Mataric. Coordination and learning in multi-robot systems. In *IEEE Intelligent Systems*, pages 6–8, March 1998.

[75] G. Mavrotas. Effective implementation of the ϵ-constraint method in multi-objective mathematical programming problems. *Applied Mathematics and Computation*, 213(2):455 – 465, 2009.

[76] A. Messac and P. D. Hattis. Physical programming design optimization for high speed civil transport (HSCT). *Journal of Aircraft*, 33(2):446–44, March 1996.

[77] A. Messac, A. Ismail-Yahaya, and C. A. Mattson. The normalized normal constraint method for generating the pareto frontier. *Structural and Multidisciplinary Optimization*, 25:86–98, 2003.

[78] K. Miettinen and M. M. Makela. Comparative evaluation of some interactive reference point-based methods for multi- objective optimisation. *Journal of the Operational Research Society*, 50(9):949–959, September 1999.

[79] S. Mostaghim and J. Teich. Strategies for finding good local guides in multi-objective particle swarm optimization (MOPSO). In *Swarm Intelligence Symposium, 2003*, pages 26 – 33, april 2003.

[80] A. D Nandasana, A. K. Ray, and S. K. Gupta. Applications of the non-dominated sorting genetic algorithm (NSGA) in chemical reaction engineering. *International Journal of Chemical Reactor Engineering*, 1:No–pp, 2003.

[81] J. F. Nash. Equilibrium points in N-person games. *Proceedings of the National Academy of Sciences of the United States of America*, 36(48-49), 1950.

[82] P. Ngatchou, A. Zarei, and M. A. El-Sharkawi. Pareto multi objective optimization. *Intelligent Systems Application to Power Systems*, pages 84–91, 2005.

[83] L. Panait. Theoretical convergence guarantees for cooperative coevolutionary algorithms. *Evolutionary Computation*, 18(4):581–615, 2010.

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

[85] V. Pareto. *Manual of Political Economy*. MacMillan Press Ltd., 1927.

[86] K.E. Parsopoulos and M. N. Vrahatis. Particle swarm optimization method in multiobjective problems. *ACM Symposium on Applied Computing*, 2002.

[87] R. Penn, E. Friedler, and A. Ostfeld. Multi-objective evolutionary optimization for greywater reuse in municipal sewer systems. *Water research*, 47(15):5911–592, 2013.

[88] Margarita R.-S. and C. A. C. Coello. Multi-objective particle swarm optimizers: A survey of the state-of-the-art. *International Journal of Computational Intelligence Research*, 2(3):287–308, 2006.

[89] Padmini Rajagopalan, Aditya Rawal, and Risto Miikkulainen. Emergence of competitive and cooperative behavior using coevolution. *GECCO*, pages 1073–1074, 2010.

[90] V. Ramanujam, N. Venkatraman, and J. C. Camillus. Multi-objective assessment of effectiveness of strategic planning: A discriminant analysis approach. *The Academy of Management Journal*, 29(2):pp. 347–372, 1986.

[91] S Ramesh, S Kannan, and S Baskar. Application of modified NSGA-II algorithm to multi-objective reactive power planning. *Applied Soft Computing*, 12(2):741–753, 2012.

[92] C. Rebhuhn, B. Gilchrist, S. Oman, I. Tumer, R. Stone, and K. Tumer. A multiagent approach to evaluating innovative component selection. In J. S. Gero, editor, *Design, Computing, and Cognition*, 2014.

[93] M. J. Reddy and D. N. Kumar. Multiobjective differential evolution with application to reservoir system optimization. *Journal of Computing in Civil Engineering*, 21(2):136–146, 2007.

[94] W. Rosehart, C. A. Cañizares, and V. H. Quintana. Multi-objective optimal power flows to evaluate voltage security costs in power networks. *IEEE Transactions on Power Systems*, 2001.

[95] M. Rubenstein, A. Cabrera, J. Werfel, G. Habibi, J. McLurkin, and R. Nagpal. Collective transport of complex objects by simple robots: Theory and experiments. *Autonomous Agents and Multiagent Systems*, 2013.

[96] H. Sayyaadi. Multi-objective approach in thermoenvironomic optimization of a benchmark cogeneration system. *Applied Energy*, 86(6):867 – 879, 2009.

[97] J. D. Schaffer and J. J. Grefenstette. Multi-objective learning via genetic algorithms. In *Proceedings of the 9th international joint conference on Artificial intelligence - Volume 1*, pages 593–595. Morgan Kaufmann Publishers Inc., 1985.

[98] M. Sierra and C. A. C. Coello. Improving PSO-based multi-objective optimization using crowding, mutation and ϵ-dominance. In *Evolutionary Multi-Criterion Optimization*, volume 3410 of *Lecture Notes in Computer Science*, pages 505–519. Springer Berlin / Heidelberg, 2005.

[99] M. Skoge, A. Donev, F. H. Stillinger, and S. Torquato. Packing hyperspheres in high-dimensional euclidian spaces. In *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, volume 74, 2006.

[100] H. Soyel, U. Tekguc, and H. Demirel. Application of NSGA-II to feature selection for facial expression recognition. *Computers & Electrical Engineering*, 37(6), 2011.

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

[102] R. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[103] H. Tamaki, H. Kita, and S. Kobayashi. Multi-objective optimization by genetic algorithms: a review. In *Evolutionary Computation, Proceedings of IEEE International Conference on*, pages 517 –522, May 1996.

[104] M. E. Taylor and P. Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 2009.

[105] R. A. Teixeira, A. P. Braga, R. H.C. Takahashi, and R. R. Saldanha. Improving generalization of mlps with multi-objective optimization. *Neurocomputing*, 35(1–4):189 – 194, 2000.

[106] C. Tomlin, G. J. Pappas, and S. Sastry. Conflict resolution for air traffic management: A study in multiagent hybrid systems. *IEEE Transaction on Automatic Control*, 43(4):509 – 521, APRIL 1998.

[107] K. Tumer. Designing agent utilities for coordinated, scalable and robust multi-agent systems. In P. Scerri, R. Mailler, and R. Vincent, editors, *Challenges in the Coordination of Large Scale Multiagent Systems*. Springer, 2005.

[108] K. Tumer and A. Agogino. Distributed agent-based air traffic flow management. In *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 330–337, Honolulu, HI, May 2007.

[109] K. Tumer and A. Agogino. Multiagent learning for black box system reward functions. *Advances in Complex Systems*, 12:493–512, 2009.

[110] K. Tumer, A. Agogino, and D. Wolpert. Learning sequences of actions in collectives of autonomous agents. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 378–385, Bologna, Italy, July 2002.

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

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

[113] K. Tumer and D. H. Wolpert. Collective intelligence and Braess' paradox. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, pages 104–109, Austin, TX, 2000.

[114] P. Vamplew, R. Dazeley, A. Berry, R. Issabekov, and E. Dekker. Empirical evaluation methods for multiobjective reinforcement learning algorithms. *Machine Learning*, 2010.

[115] M. Vasirani and S. Ossowski. A market-inspired approach to reservation-based urban road traffic management. *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems*, 2009.

[116] D. A. Van Veldhuizen. *Multiobjective Evolutionary Algorithms: Classifications Analyses and New Innovations*. PhD thesis, Air Force Institute of Technology, 1999.

[117] D. A. Van Veldhuizen and G. B. Lamont. Multiobjective evolutionary algorithms: Analyzing the state-of-the-art. *Evolutionary Computation*, 8(2):125–147, 2000.

[118] T. Wagner, N. Beume, and B. Naujoks. Pareto-, aggregation-, and indicator-based methods in many-objective optimization. In Shigeru Obayashi, Kalyanmoy Deb, Carlo Poloni, Tomoyuki Hiroyasu, and Tadahiko Murata, editors, *Evolutionary Multi-Criterion Optimization*, volume 4403 of *Lecture Notes in Computer Science*, pages 742–756. Springer Berlin Heidelberg, 2007.

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

[120] D. H. Wolpert, K. Tumer, and E. Bandari. Improving search algorithms by using intelligent coordinates. *Physical Review E*, 69:017701, 2004.

[121] M. Wooldridge. *An Introduction to Multiagent Systems*. John Wiley and Sons, 2002.

[122] H. Xu, Z. Zhang, K. Alipour, K. Xue, and X.Z. Gao. Prototypes selection by multi-objective optimal design: application to a reconfigurable robot in sandy terrain. *Industrial Robot: An International Journal*, 38(6):599–613, 2011.

[123] P. O. Yapo, H. V. Gupta, and S. Sorooshian. Multi-objective global optimization for hydrologic models. *Journal of Hydrology*, 204(1–4):83 – 97, 1998.

[124] L. Yliniemi, A. K. Agogino, and K. Tumer. Multi-robot coordination for space exploration. *AI Magazine*, 2014.

[125] L. Yliniemi and K. Tumer. Multi-objective multiagent credit assignment through difference rewards in reinforcement learning. *10th International Conference on Simulated Evolution And Learning (SEAL)*, 2014.

[126] L. Yliniemi and K. Tumer. PaCcET: An objective space transformation to iteratively convexify the pareto front. In *10th International Conference on Simulated Evolution And Learning (SEAL)*, 2014.

[127] L. Yliniemi and K. Tumer. Complete coverage in the multi-objective PaCcET framework. In S. Silva, editor, *Genetic and Evolutionary Computation Conference*, 2015.

[128] L. Yliniemi, D. Wilson, and K. Tumer. Multi-objective multiagent credit assignment in NSGA-II using difference evaluations. *Autonomous Agents and Multiagent Systems*, 2015.

[129] T. Yoshida and H. Mori. Application of SPEA2 and monte-carlo simulation in correlation of specific values to multi-objective optimal allocations of SVRs. *IEEJ Transactions on Power and Energy*, 131:283–289, 2011.

[130] G. Zhang, X. Shao, P. Li, and L. Gao. An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem. *Computers and Industrial Engineering*, 56:1309–1318, 2009.

[131] Z.-H. Zheng, Q. Ai, W.-H. Xu, L. Han, C.-W. Jiang, S.-G. Feng, and C.-H. Gu. Multi-objective load dispatch in wind power integrated system based on pseudo-parallel SPEA2 algorithm. *Journal of Shanghai Jiaotong University*, 8:009, 2009.

[132] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm. *Computer Engineering*, 3242(103), 2001.

[133] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *Evolutionary Computation, IEEE Transactions on*, 3(4):257 –271, Nov 1999.