University of Nevada, Reno

GENETIC ALGORITHMS OPTIMIZED POTENTIAL FIELDS FOR DECENTRALIZED GROUP TASKING

A Dissertation Submitted in Partial Fulfillment

of the Requirements for the Degree of Doctor of Philosophy in

Computer Science and Engineering

by

Rahul Dubey

Dr. Sushil J. Louis / Dissertation Advisor

August 2021

UNIVERSITY OF NEVADA RENO **THE GRADUATE SCHOOL**

We recommend that the dissertation prepared
under our supervision by

RAHUL DUBEY

entitled

GENETIC ALGORITHMS OPTIMIZED POTENTIAL FIELDS FOR DECENTRALIZED GROUP TASKING

be accepted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

Sushil J. Louis, Ph. D. – Advisor

Shamik Sengupta, Ph. D. – Committee Member

Monica Nicolescu, Ph. D. – Committee Member

Hung La (Jim), Ph. D. – Committee Member

Ramona Houmanfar, Ph. D. – Graduate School Representative

David Zeh, Ph. D. – Dean, Graduate School

August 2021

ABSTRACT

Maneuvering autonomous agents to accomplish complex tasks is a difficult and typically NP-hard optimization problem with many real-world applications. In this thesis, we use potential fields based on task and agent properties to control the movement of groups of agents and use a genetic algorithm (GA) to optimize potential field parameter values to lead to complex task achieving behaviors. More specifically, we control autonomous unmanned aerial vehicles (UAVs) in search and rescue scenarios to find and help people in need, in wildfire coverage scenarios to monitor a wildfire's perimeter, and game agents in real-time strategy (RTS) games to win skirmishes. In all three applications, potential fields control agent movement, genetic algorithms optimize potential field parameters, and a simulation evaluates task performance to guide genetic optimization.

Experimental results show that our potential field representation and problem formulation works well across the three problems. We used UAVs as flying access points and controlled their movement using genetic algorithms optimized potential fields to generate wireless networks. These ad-hoc wireless networks outperformed the current state of the art ad-hoc network deployment algorithm. The same representation with a different set of potential fields was used for successful deployment of UAVs to track the spread of wildfire boundaries and results show that with enough UAVs, complete fire boundary coverage was achieved. Lastly, we used two different RTS game platforms to evolve tactics for a team of heterogeneous game agents by formulating the problem as a multi objective optimization problem. Again using potential fields, a genetic algorithm evolved a diverse set of high quality skirmish tactics ranging from attacking to fleeing against test opponents. Results show that with aggressive attacking tactics, a team of friendly agents was able to eliminate the majority of opponents but suffered significant damage. On the other hand, fleeing tactics resulted in less damage to friendlies but also inflicted less damage to opponents. We also observed the emergence of cooperation between friendly game agents. These results indicate that genetic algorithms optimized potential fields are a viable approach to decentralized group tasking.

ACKNOWLEDGEMENTS

I would like to sincerely thank my advisor, Dr. Sushil J. Louis, for the patience and support throughout this journey. I could not have imagined having a better advisor and mentor, who has guided and helped me to not just think critically to do research but also taught me the importance of professional communication. I would also like to thank my dissertation committee members: Dr. Shamik Sengupta, Dr. Hung La, Dr. Monica Nicolescu, and Dr. Ramona Houmanfar for their valuable feedback, suggestions, and support at various stages of my dissertation. I thank my lab members, especially, Dr. Siming Liu for all the technical discussions and support he provided during my initial days at UNR.

This work was supported by grant number N000141712558 from the Office of Naval Research and in part by the U.S. Department of Transportation, Office of the Assistant Secretary for Research and Technology (USDOT/OSTR) under Grant No. 69A3551747126 through INSPIRE University Transportation Center (http://inspire-utc.mst.edu) at Missouri University of Science and Technology. The views, opinions, findings and conclusions reflected in this dissertation are solely those of the authors and do not represent the official policy or position of the either the Office of Naval Research, USDOT/OSTR, or any State or other entity.

Last, but definitely not the least, I would especially like to thank my wife and family members for their immense support throughout this time period. Without their unconditional support I would not have achieved this.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Controlling autonomous agents has many real-world applications but presents multiple challenges. These challenges arise because of the large search space of control parameter values and the dynamic nature of environments. Searching for solutions in large search spaces in real-time is difficult, and if the environment state changes then the previously obtained solutions may not be valid. In this thesis, we present a new representation and problem formulation to control a large number of autonomous agents for group tasking. Our representation uses task-specific potential fields to guide autonomous agents to complete group tasks. Potential fields based autonomous agent control was first introduced by Khatib [1] in 1986. Owing to their simplicity, many researchers in robotics and in games have used distance based potential fields to control the movement of agents for obstacle avoidance and navigation. However, in this thesis, we augment distance based potential fields with more complex, task specific, and agent properties driven potential fields to guide agents to perform group tasks. Since these potential fields can be highly non-linear, tuning their parameters is a difficult optimization problem and we therefore use genetic algorithms to optimize potential field parameters.

Genetic Algorithms (GAs) have been used extensively to solve poorly understood non-linear optimization problems [2, 3, 4, 5, 6]. The genetic algorithm optimizes potential field parameters offline and we use these optimized potential field parameters to control agents for group tasking in this thesis. Computing potential fields is not computationally expensive and thus our task specific potential fields based approach enables agents to make real-time decisions. Many real world applications require real-time decision making and this makes our approach and

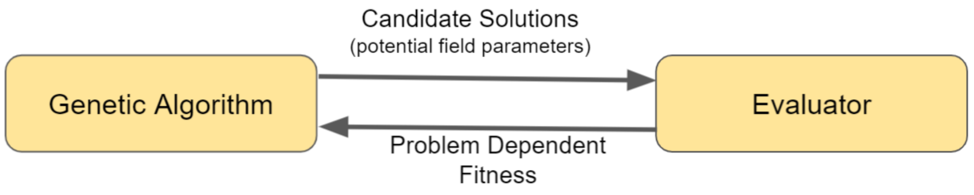


Figure 1.1: Offline optimization of potential field parameters using genetic algorithms.

representation applicable to a wide range of problems. We show the applicability and effectiveness of our approach on three different challenging problems: 1) wireless mesh network deployment (WMND) using UAVs for search and rescue, 2) dynamic fire boundary tracking using UAVs to fight forest fires, and 3) game agent control in real-time strategy games to win skirmishes. In all three problems we use genetic algorithms to optimize potential fields that guide agents.

For WMND, the genetic algorithm optimizes potential fields based on wireless bandwidth coverage and distances. Similarly, we use potential fields based on fire intensity and distances to control the movement of UAVs to track the perimeter of a fire. In these two problems we control only one type of UAV, but in RTS games we control multiple types of game agents using potential fields based on distance, health, and weapons properties to win skirmishes. In all three problems we identify the specific potential fields to use and the GA optimizes this set of potential fields so that agents perform well on the group task.

Figure 1.1 shows a block diagram of our system's architecture which has two components: **the genetic algorithm and the evaluator**. The GA works with a population of candidates solutions each of which encodes possible potential field parameter values. To make progress towards high quality potential field parameters, the GA needs to know the relative merit or utility (the fitness) of a set of parameters at the task. We use a simulation to evaluate this fitness and the GA moves towards higher fitness using genetic operators. Chapter 2 provides more detail about genetic algorithms. To use our approach on a new problem we only need a task specific evaluator and the identification of potential fields that affect agent behavior. Since the evaluator is problem specific, we next describe the three evaluators that evaluate the fitness of a candidate solution.

First, we created an evaluator that simulates a wireless mesh network and given a set of potential field parameters returns the sum of bandwidth coverage and longevity as the fitness of the candidate solution. The experimental results show that our approach outperformed the state-of-the-art wireless network deployment algorithm. The second evaluator simulates the spread of the wildfire and computes the fire boundary coverage and energy consumption of UAVs and returns these as the fitness. Results show that with enough UAVs, we achieved complete fire boundary coverage. Lastly, the third evaluator simulates a real-time strategy game where a friendly team plays against an enemy team and friendly agents are controlled by the genetic algorithm optimized set of potential fields. The evaluator returns the damage done and damage received by friendly agents as the fitness and the results show that our approach evolved a diverse set of high quality solutions.

1.1 Structure of this Thesis

We organize this thesis based on problem complexity rather than chronology. We start with the WMND, a single objective problem, that has a fixed evaluator, and end with RTS micro, a multi objective problem with intelligent dynamic opponents.

The next chapter begins with an overview of genetic algorithms and how they are used in my work. Next, we review prior work in UAV wireless network deployment for search and rescue, dynamic fire boundary coverage, and RTS games.

Chapter 3 starts by mathematically modeling potential fields and how to use potential fields for agent movement control. Subsequently, we describe an agent's motion model. We then describe some important aspects of our experimental setup.

Chapter 4 specifies wireless mesh network deployment using our Genetic Algorithm Network deployment algorithm (GANet). The chapter begins with the problem statement followed by our methodology where we explain our GANet algorithm and identify the set of potential fields to control UAVs. Next, we present experimental results showing that GANet outperformed the state-of-the-art. This was published in the *Proceedings of the 2020 IEEE Congress on Evolutionary Computation* [7]. More extensive experimental results will be published in the *Proceedings of the 2021 Genetic and Evolutionary Computation Conference (GECCO)* [8] and are under review in *IEEE Access* (a journal).

Chapter 5 describes tracking the spread of wildfire boundaries using UAVs and is organized like the previous chapter. We define the problem, the fire simulation model, and the set of potential fields. Results show that, given enough UAVs we can achieve complete fire boundary coverage. These results will be published in the *Proceedings of the 2021 IEEE Congress on Evolutionary Computation* [9].

Chapter 6 specifies the control of heterogeneous game agents in RTS games to win skirmishes. This is a different problem compared to wireless network deployment and dynamic fire coverage in that we have opponents, but we use the same approach to control the movement of game agents. Like in previous chapters, we define the problem and the set of potential fields. We used two different game simulation platforms for experiments and results show that our approach generates a diverse set of tactics. These results were published in the *Proceedings of the 2018 IEEE Conference on Computational Intelligence and Games* [10] and in the *Proceedings of the 2019 IEEE Congress on Evolutionary Computation* [11]. We next increased problem complexity to investigate the scalability of our approach. These results will be published as a workshop paper in the *Proceedings of the 2021 Genetic and Evolutionary Computation Conference (GECCO)* [12]. Finally, the last chapter, Chapter 7, provides the conclusions and presents possible future work.

CHAPTER 2
BACKGROUND

We start with an overview of genetic algorithms. We then review work done in wireless mesh network deployment, fire boundary coverage, and RTS micro.

2.1 Genetic Algorithms

Genetic algorithms are inspired by the process of natural selection. Holland proposed genetic algorithms in the early 1970*s* as computer programs that mimic evolutionary processes in nature, and his canonical genetic algorithm (CGA) is shown in Figure 2.1 [13, 14].

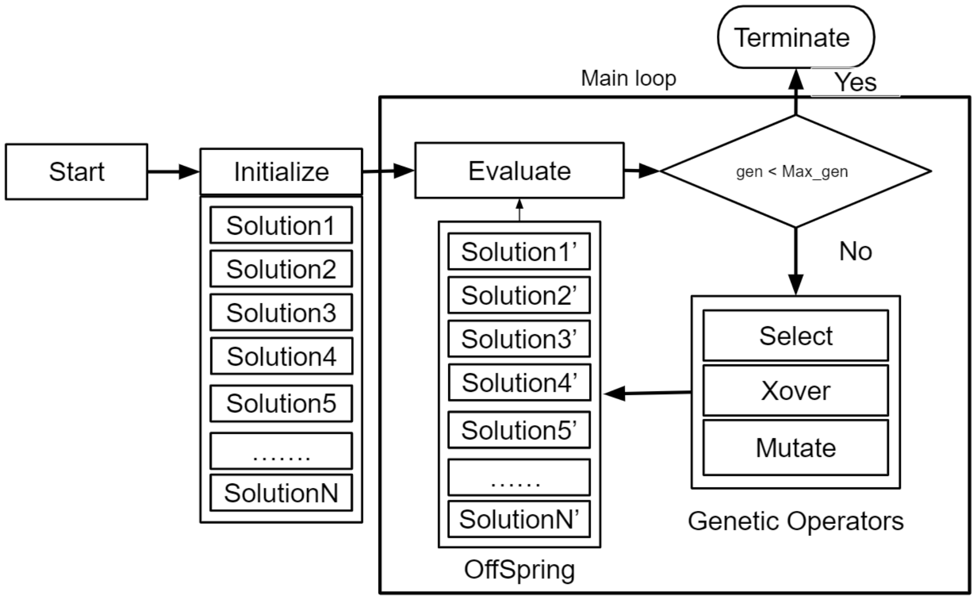


Figure 2.1: Overview of the canonical genetic algorithm.

Genetic algorithms attempt to solve problems with an iterative process starting with a population of randomly initialized individuals. Each individual encodes

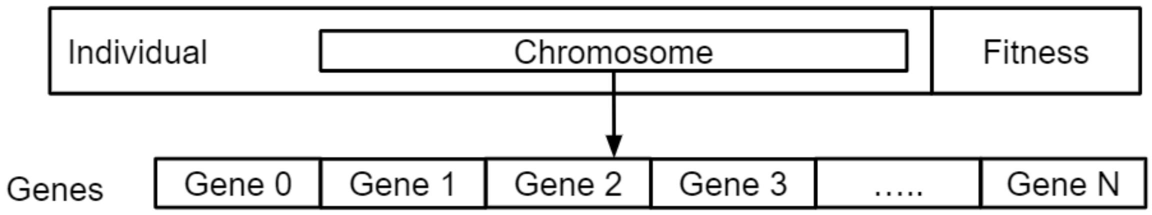


Figure 2.2: An individual represented by a binary chromosome and genes.

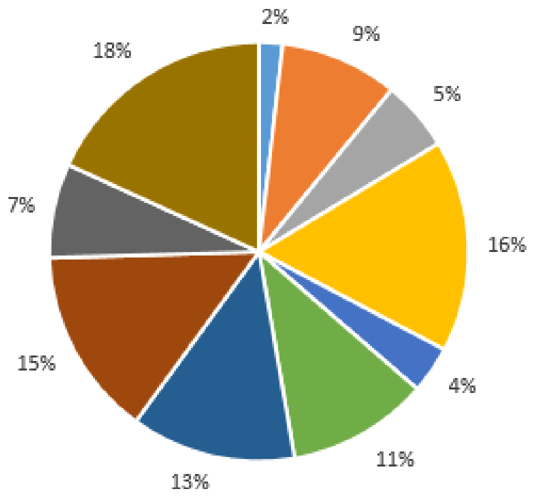


Figure 2.3: Fitness proportional selection scheme with population size of 10.

a candidate solution to the problem in its chromosome that represents a set of parameters called genes, as shown in Figure 2.2. Each gene is in turn encoded into a binary string.

The values interpreted from the genes are called alleles. Chromosomes are not the actual solution to the given problem, they are a blueprint or a set of instructions that dictate how to create a solution. Each individual is evaluated and assigned a real value known as the fitness of the individual. Once all individuals are evaluated, GAs create offspring from individuals of the current population using three genetic operators; selection, crossover, and mutation. The selection operator selects individuals for reproduction based on a fitness proportional selection scheme as shown in Figure 2.3. An individual with higher fitness has higher

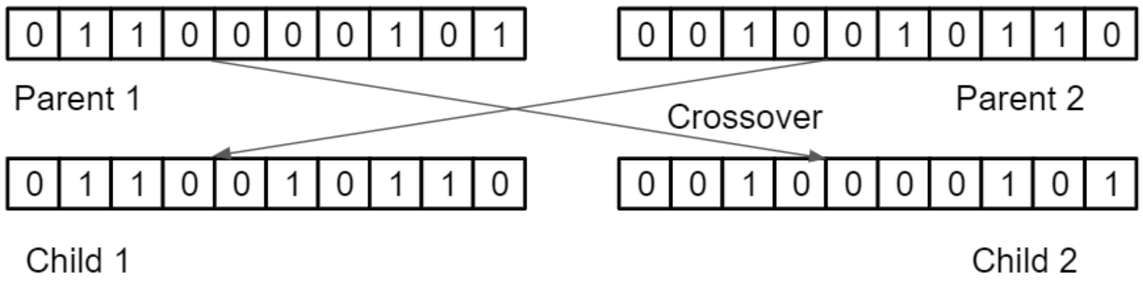


Figure 2.4: One point crossover between two parents to create two children.

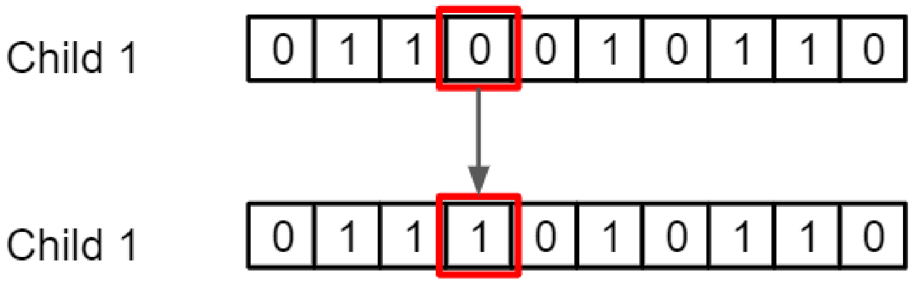


Figure 2.5: Bit-wise mutation, fourth bit form left changed from 0 to 1.

chances of being selected for reproduction and will have a larger representation in subsequent generations. The recombination of individuals in genetic algorithms is simulated through a crossover operator that exchanges portions between individuals as shown in Figure 2.4. The figure shows one point crossover that creates two children by crossing over two individuals. Mutation causes probabilistic alteration of the genes of newly created children. Figure 2.5 shows a bit-wise mutation where the allele of a gene is flipped.

The newly generated offspring are evaluated and the process of selection and recombination using genetic operators is continued until reaching a termination criteria. Table 2.1 lists the common terms and their descriptions used in GAs.

In a canonical GA, individuals generated (good or bad) have a relatively short life span since the population is constantly being replaced by new generations of individuals. Hence, even if a global optimum is generated, it can disappear in a

Table 2.1: Genetic Algorithm Nomenclature

Term	**Description**
Population	A set of individuals
Individual	A candidate solution to the problem
Chromosome	A string of genes
Gene	An element of chromosome
Allele	The values which a gene can assume
Fitness	A value indicating the quality of an individual as a solution to the problem.
Selection	A way to chose one individual from the population.
Crossover	Operation that exchanges information of two selected parents to yield two new children.
Mutation	Operation that probabilistic changes one or more bits in a chromosome

generation or two for long periods of time. Thus it is also possible that despite having higher fitness some good individuals may not be selected and may eventually be lost.

2.1.1 Elitist Genetic Algorithms

To preserve good individuals generated in any generation to the next generation, we use elitism. There are several ways of doing elitism in genetic algorithms [4, 15] and we chose $n + n$ elitism where n is the population size. The three GAs operators create offspring of size n from n parent individuals. As shown in Figure 2.6, when using $n + n$ we create a pool of individuals that is two times larger than the population size ($2n$). In the case of a single objective problem, we sort individuals based on their fitness value and pick the best n individuals. However, in the case of a multi objective problem an individual's fitness has multiple objective values. Thus, to pick the best n individuals we sort $2n$ individuals into different pareto

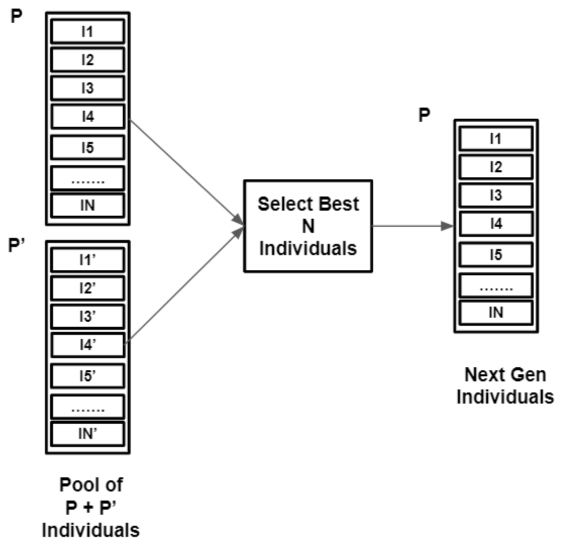


Figure 2.6: $n + n$ elitism

fronts and then choose the best n individuals. In the next section we begin with the first problem, wireless mesh network deployment.

2.2 UAV Wireless Mesh Network Deployment

UAVs are gaining traction as a useful tool in many different domains including search and rescue [16], surveillance [17], cargo transport [18], surveying and mapping [19], agriculture [20], disaster relief [21], IoT [22], and defense and security [23, 24, 25] applications where there is a need to quickly deploy a wireless communications network. A UAV can fly quickly to establish Line of Sight (LoS) communication with users without being affected by different geographical terrain [26]. During emergencies or in remote areas where existing communication networks have failed or do not exist at all, a quickly deployed wireless communication network can help users safely exit the emergency area and can help emergency personnel find and rescue users [27]. Figure 2.7 shows an example wireless

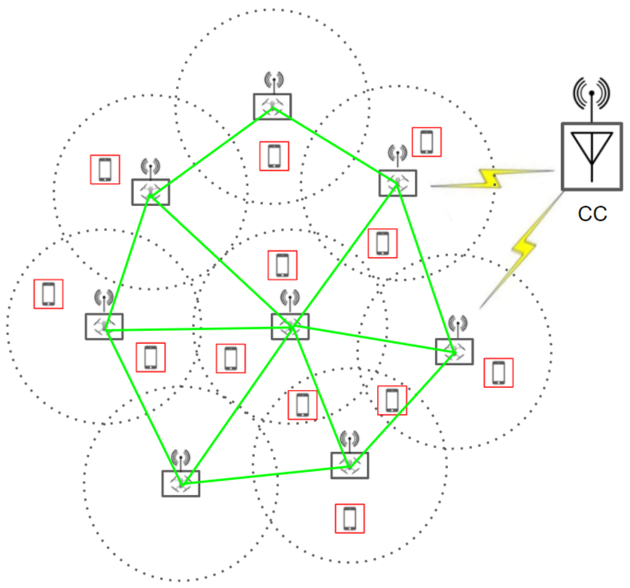


Figure 2.7: An example of UAVs network deployment with 8 UAVs, 12 users, and a command center labelled "CC." UAVs are shown by black boxes and red boxes represents users.

mesh network using eight UAVs to provide bandwidth coverage to twelve users.

In this figure, UAVs are shown inside black rectangular boxes and users are shown by red rectangles. We assume that each UAV can communicate with users within a certain range as shown by black-dashed circles and can communicate with neighboring UAVs shown by green links. A command center is placed on the right and is in direct communication with two UAVs. Each UAV can communicate with the CC either directly or through neighboring UAVs to provide bandwidth coverage to all twelve users. In this work, bandwidth coverage refers to data in Megabits per second (Mbps) provided by UAVs to users. Different users, depending on their needs, may require different bandwidth coverage, and thus, bandwidth requirements combined with users' locations determine optimal collision free UAV movement and positioning.

However, many challenges remain before a large number of UAVs can be efficiently and effectively deployed [28]. For example, in a search and rescue scenario we have to first distribute available UAVs over an area of interest to find all users in the AOI, then dynamically continue re-positioning UAVs to serve found users. Re-positioning should maintain connectivity to a fixed operations command center while serving moving users and changing bandwidth needs [29].

This thesis focuses on the problem of controlling a large number of UAVs that create and maintain a mesh network that connects all users to the command center while serving their bandwidth needs for the maximum amount of time. UAV deployment typically proceeds in two phases. In the first phase, UAVs are deployed to cover the maximum possible area to find users in the given AOI [30, 31]. Once we have found all users and know the distribution of users within the AOI, in the second phase, we need to optimally re-deploy UAVs in order to serve users better [32, 33, 34]. Many researchers have presented different control algorithms for both phases of this wireless network deployment problem.

Historically, the research areas of networking and robotics have examined the first phase. In the field of networking, Wang presented a mobile sensor placement algorithm based on voronoi diagrams [35]. Circle Packing Theorem (CPT) is another computational geometry model for wireless network deployment [36]. Lam deployed a heterogeneous sensor network using circle packing by filling the given AOI with circles of different radii corresponding to different UAV types [37]. In the field of robotics, Howard used potential fields to deploy a mobile sensor network to cover an area [38]. In this approach an agent experienced repelling potential fields based on distance from other agents and obstacles and moved towards unexplored areas. Poduri introduced a mobile network deployment algorithm with the constraint that each agent has at least K neighbors where K is a user defined number [39]. All these approaches only address the first phase and aim to provide area coverage without adapting to distribution of users.

The second phase addresses the problem of deploying UAVs to serve users who have been found in the first phase. In the networking field, Ming introduced an adaptive mobile network deployment algorithm by taking inspiration from Delaunay Triangulation (DT) [40]. Bartolini proposed a voronoi polygon based adaptive network deployment for heterogeneous agents [41]. In [42], the authors initially deploy UAVs using CPT and then proposed an approach to adjust the altitude of deployed UAVs to increase or decrease a UAV's sensing area. Bandwidth area coverage is increased by increasing the altitude of deployed UAVs, but the signal strength reduces as the inverse of distance squared resulting in poor coverage quality. Zhao presented a centralized algorithm and a potential field based distributed algorithm for UAV deployment while maintaining connectivity among UAVs [43]. However, they considered only two different types of user's distributions - uniformly random and in three clusters spread around the AOI with a command center in the middle. All these approaches work well in relatively uniform distributions of users but fail when users are in distinct clusters and distributed non-uniformly.

Other than area coverage and re-positioning of deployed UAVs, mesh network creation and maintenance presents multiple challenges when using UAVs as base stations. Minimizing the number of UAVs to be deployed, minimizing deployment time, extending lifetime of deployed UAVs, routing, and channel allocation, all present significant challenges. In 2016, Lyu proposed a placement optimization technique to minimize the number of unmanned aerial vehicle-mounted mobile base stations while providing wireless coverage to ground terminals [44]. The algorithm works in polynomial time with successive mobile base station placement. To minimize the deployment delay, Zhang presented a fast deployment algorithm [45]. Amar [46] presented a dynamic algorithm to serve a sub-region that requires more bandwidth. The study assumed an uniform distribution of user positions while we look at non-uniform distribution of users.

Closer to our work, researchers have used genetic algorithms to deploy static networks by optimally placing UAVs in the AOI. Reina [47] presented a multi-layout multi-subpopulation genetic algorithm to deploy UAVs optimally to maximize a linear combination of coverage, fault-tolerance, and redundancy. Dina [48] introduced a variable length genetic algorithm to maximize area coverage and minimize deployment cost using non-homogeneous sensors. These papers provide UAV positions as their output. In contrast, we use a genetic algorithm to evolve potential field parameters to guide UAVs. This not only gives us UAV positions, but also guides collision free UAV movement to these positions.

In this thesis, we use GANet to control UAVs movement to generate a wireless mesh network. Our approach uses four potential fields to guide UAVs in an unknown environment and details are provided in Chapter 4.

2.3 Dynamic Fire Boundary Coverage

Wildfires are spontaneous events that cause massive destruction to structures and wildlife. Figure 2.8 shows a wildfire scenario where a firefighter tries to suppress the spreading of the fire. The Congressional Research Service reported that every year since 2000, there has been an average of 70, 685 wildfires that burned an average of 7.1 million acres in the United States [49]. Although on average, there were

Figure 2.8: A wildfire outbreaks in California. Firefighting is really dangerous without continuous fire fronts growth information. Courtesy of NBC News

78, 600 wildfires annually in the US in the 1990s, the total acres burned have more than doubled. Fighting wildfires is dangerous as the behavior of wildfires can be unpredictable and difficult to model. The National Fire Protection Association reports that from 2014 to 2018 on average 65 firefighters' lives were lost annually while fighting wildfire [50] and these numbers do not even account for the number of non-firefighter civilian lives lost to wildfires. The loss of wildlife, human life, and structures highlights the importance of the need to locate, observe and track wildfires. This information is critical to making emergency plans to evacuate civilians to safety and to fight the fires.

UAVs can and have been used to assist humans in emergency and disaster situations by providing situational awareness with imagery and maps [24, 51, 52]. By maintaining proper communication links between the UAVs and ground control stations, we can remotely and safely assess damage in a given region of interest. Thus, UAVs are highly suitable for tackling the wildfire tracking problem by providing imagery and maps while relaying information through each other to firefighters/operators who are at a safe distance [53]. Teaming of UAVs and other robots to collaborate on resolving multiple challenges has risen in popularity for both research and application. Multiple UAVs have been used to collaborate as a network of agents with sensors to build maps and gather information in their local areas to get accurate information of the entire scope of the problem.

We focus on using UAVs for tracking the spread of wildfires in forests. Two fundamental challenges when using UAVs in unknown regions for fire coverage are fire detection and fire tracking, and this thesis focuses on the latter. However, the two are related as we need detection for tracking, so we start with providing an overview of work in detection.

Yuan [54] used Infrared (IR) imaging sensors installed on UAVs to detect the presence of fire and presented techniques to process images gathered using different sensors mounted on UAVs, to study fire spreading behaviors. Afghah [55] proposed a leader-follower formation to cluster a set of UAVs into multiple coalitions that collectively covered a particular area of interest. Merino [56] proposed a cooperative perception system for multiple heterogeneous UAVs for automatic detection of forest fires. They collected data using multiple sensors such as a visual cameras, infrared sensors, and fire detectors mounted on UAV's and fused them together for detection, monitoring, and measurement of forest fires. Another fire detection technique developed by Yuan [57] analyzes fire segmentation in different color spaces. Henry [58] introduced a Forest Fire Detection Index (FFDI) to detect fires through the use of a new color index. The index is based on a method for vegetation classification and used to detect flames and smoke.

The work in fire detection has shown that several robust techniques exist for fire detection and tracking as long as we have good observation platforms with suitable sensors. In this thesis, we assume that UAVs have a fire detection sensor and our challenge is to find the fire boundary and move UAVs to track this boundary as it grows in perimeter length over time. Coordinated control of multiple UAVs is essential for dynamic fire coverage and control techniques are broadly categorized as either distributed or centralized control. David [59] presented a path planning algorithm to track fire using low altitude short endurance UAVs. This centralized path planning computes waypoints for each UAV, with these waypoints being generated along the edge of a fire (along the fire boundary). Phan [60] worked on a similar problem where he proposed a cooperative control framework for a team of UAVs and unmanned ground vehicles (UGVs) to detect and track fires. In this centralized framework, a mission controller monitors a dynamic environment, formulates high level mission plans, and allocates tasks to each vehicle.

Using our potential field based representation, each agent acts independently and thus works in a decentralized manner by collecting information about the given AOI using mounted sensors and by communicating with neighbors. Many researchers have studied decentralized control techniques for fire coverage using UAVs. Manish [61] investigated the cooperative control of multiple UAVs for accurate situational awareness and distribution of fire suppressant fluid at the edges of fire. Maza [62] proposed a distributed decision making architectural framework for multi-UAV configuration in disaster management. Multiple checkpoints in the region of interest can be used to command UAVs to track down the boundary for fire coverage. The authors of [63] presented a deep learning approach to deploy UAVs to collect images for fire classification.

Open challenges still include the coverage of boundaries of a fire when the fire spreads dynamically. Earlier approaches to resolve this problem include [59] and [60] where UAV maneuvering decision making was centralized. Centralized approaches suffer from a single point of failure and may need more computation and communication hardware than available. However, given enough computational and communication resources, these approaches can direct a team of UAVs to continuously track the spread of fire along the boundary as long we have a good fire spread model or external information about the fire boundary. Simultaneous detection and tracking in a decentralized manner remains challenging. La used a team of UAVs working together to track and follow a wildfire as it spreads by tracking fire intensity and heat sources [64]. However, the only objective was to maximize fire boundary coverage and there was no modeling of energy usage.

2.4 Real-Time Strategy Games

RTS games are an active area of research and contain a variety of interesting and challenging problems. In RTS games, players gather resources, build infrastructure, build different game agents, expand control over the map, scout to gather opponent's activities, and ultimately try to destroy their opponent's base while their opponent also attempts to do all of the above [65]. Compared to board games, RTS games are more complex. In most board games, each player has their turn, player actions take effect immediately with deterministic results, and the board state is fully observable. In contrast, RTS games are more complex because there are no discrete player turns and players make moves simultaneously, actions are durative, and the game state is only partially observable. To quantify the increase in complexity, while *Chess* has 10^{50} board states and *GO* has 10^{170} board states, RTS games are estimated to have over $(10^{50})^{36000}$ states to play an entire game to completion, more than the number of protons in the universe [65]. Therefore, traditional Artificial Intelligence (AI) techniques used for playing board games, such as MINIMAX game tree search, cannot be directly applied to RTS games [66].

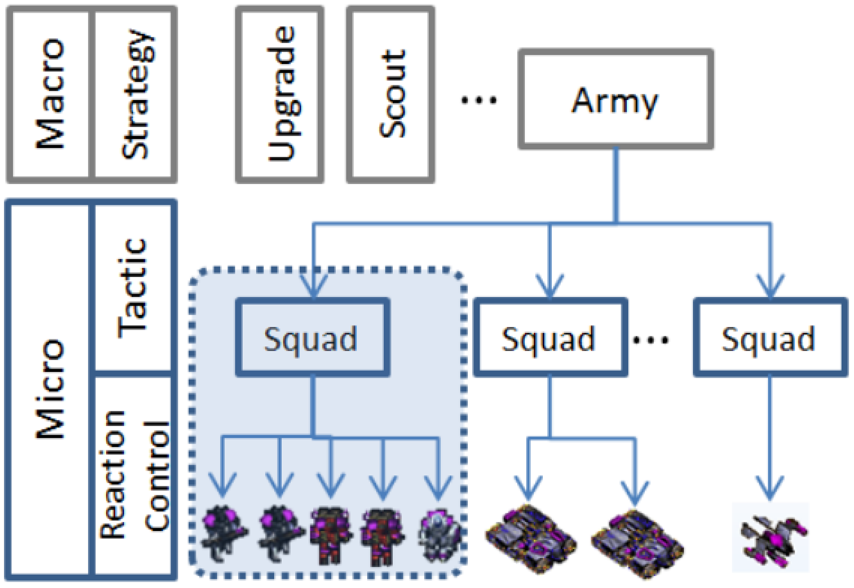


Figure 2.9: Typical RTS AI levels of abstraction[81]

In RTS games, players broadly decompose decision making during game play into two categories; macro-management and micro-management shown in Figure 2.9. Macro-management or macro refers to long term planning in game play. Long term planning involves coming up with an opening strategy, build order planning, managing resources, and scouting to find out an opponent's infrastructure and army composition to infer the opponent strategy. Micro-management or micro deals with controlling a group of agents to win skirmishes against dynamic and intelligent opponents who are also trying to win. We focus on micro because good micro helps to win skirmishes and even games. Many different approaches have been proposed for finding high performance micro in industry and academia. Algorithms developed by industry RTS AI developers are not designed to beat opponents rather these algorithms are designed to entertain players while focusing on testability and repeatability [67]. Hard coded approaches have been extensively used in commercial RTS games and most common hard coded approaches use finite state machines [68]. On the other hand, academic RTS AI research focuses on using learning or reasoning techniques to win skirmishes and games. Our research falls in the academic category.

The AI and Computational Intelligence (CI) research community have presented several techniques for micro. Game-tree search has been explored extensively to search for a good micro in RTS games. Churchill presented an algorithm called Alpha-Beta Considering Duration (ABCD) to control upto eight agents in RTS games [69]. Chung applied a monte carlo planning technique to guide agents in ORTS; a simplified version of StarCraft-Broodwar [70]. Balla used a monte carlo tree search algorithm for tactical assault planning in Wargus, a RTS game [71]. Game-tree search algorithms work well for low complexity problems who's search space is tractable, but become unfeasible for large problems. To make game-tree search applicable, abstractions or simplifications of game state representations and possible actions for a given game state need to be used.

Researchers started focusing on Reinforcement Learning (RL) to solve computationally complex problems. RL achieved great success in playing and winning different games. For example, *AlphaGo*, an agent trained using RL techniques defeated a human champion in a popular board game called GO [72]. *AlphaStar*, a RTS game playing agent defeated grandmasters in StarCraft-II (SC2) [73]. A team at DeepMind presented deep reinforcement learning techniques to achieve human-level control and outperformed state of the art techniques in classic Atari 2600 games [74]. Specifically for RTS micro, Shao used a multi agents gradient descent State-Action-Reward-Action-State (SARAS) algorithm to train a neural network to control a group of agents composed from upto two different types of game agents [75]. All these approaches use Deep Neural Networks (DNNs) and different learning techniques to train DNNs. However, despite having the ability to solve computationally complex problems, DNNs are opaque and its difficult to explain why an action has been taken for a given state [76, 77].

Other than explainability, many hyper-parameters need to be tuned when working with DNNs to get better results, and tuning these hyper-parameters is difficult. Taking these issues into account, researchers have tried to use evolutionary algorithms to not only tune weights and biases but also to evolve the structure of neural networks. Neuroevolution of augmenting topologies (NEAT) is a type of evolutionary learning technique and has been used to generate RTS micro [78]. NEAT evolves the topology, weights, and biases of neural networks and works on the principle of natural selection [79]. Similar to DNNs, neural networks evolved by NEAT are also opaque and do not provide insights into decision making.

This thesis presents a new approach and representation based on potential fields and influence maps that is scalable to a large number of agents, works in real time, and is explainable. Khatib in 1986 first used potential fields to navigate robots while avoiding obstacles [1]. Later on, Hagelback presented preliminary work on RTS micro using potential fields in combination with an A^* path planning [80]. However, the work considers only distance dependent potential fields for all agents. Our representation in this thesis considers a group composed from upto three different types of agents and each different type of agent has a unique set of potential field parameters. Louis and others have used potential fields for 2D and 3D distributed autonomy in games [81, 82, 83, 84]. Apart from using potential fields, Sweetser used a cellular automata and influence map based decision making approach in a three dimensional game environment called EmerGEnt [85]. In RTS games, huge losses can occur if agents move alone and battle separately. To deal with this issue Preuss used evolutionary algorithms to generate flocking behavior and an influence map for path finding in the RTS game Glest [86]. Doherty used an evolutionary computing technique to evolve tactical team behavior for multiple agents, where each agent has different abilities that combine for effective tactics [87]. Uriarte generated kiting behavior for multiple agents using influence maps in an RTS game called Nova [88].

Good micro aims to eliminate opponents (first objective) while keeping friendly agent safe (second objective), and thus we formulated this problem as a multi objective problem to evolve Pareto front of micro behaviors. We use the Non-dominated Sorting Genetic Algorithm (NSGA-II) [89] to optimize potential field paramters and to generate a diverse set of micro behaviors. However, other multi objective algorithms given in the literature [90] can also be used for optimization. NSGA-II uses the same principle of natural selection as canonical genetic algorithms. However, NSGA-II differs in two ways from a canonical genetic algorithm. First, in the selection of individuals, and second in preserving good solutions. The crossover and mutation operators in the NSGA-II function the same as in the canonical genetic algorithm, but because of multiple objectives, the selection of individuals for crossover and mutation is different from the canonical genetic algorithm. Moreover, the canonical genetic algorithm may lose a good candidate solution in the process of selection and reproduction. To mitigate this, NSGA-II uses elitism to preserve good solutions found in each generation and carry them to the next generation.

As shown in Figure 2.10, NSGA-II combines P parent candidate solutions and Q child candidate solutions to choose the next P candidate solutions (in the next generation) from the pool of $P + Q$ candidate solutions. The selection of the next P solutions happens in two steps. In the first step, all solutions are segregated

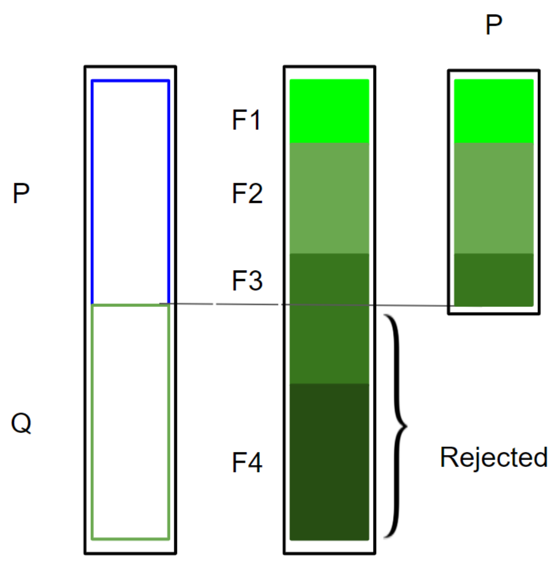


Figure 2.10: NSGA-II Procedure.

into different sets known as ranks such that rank 0 solutions are better than rank 1 solutions, and so on. After ranking solutions, different rank solutions starting from rank 0 are copied directly into the next generation solutions provided the number of solutions copied are less than the population size. In the second step, if there are P_n candidate solutions in the next generation where $P - P_n < m$ and there are m solutions with i^{th} rank then NSGA-II uses crowding distance to select remaining $P - P_n$ solution from m solutions in the next generation. To maintain diversity, a candidate solution with the highest crowding distance is selected. Once next generation candidate solutions are obtained, the entire process repeats until a termination criterion satisfied.

CHAPTER 3
METHODOLOGY

We evolve solutions to control large numbers of autonomous agents in different environments. Our representation uses task-specific potential fields to control the movement of agents and uses genetic algorithms to optimize the potential field parameters. We describe potential fields in general and how we use potential fields to control agents in this thesis. Subsequently, we explain the agent's motion model, and how we evaluate candidate solutions. Lastly, we describe parallel genetic algorithms.

3.1 Potential Fields

Potential fields have been used to guide agents without collision [91]. In physics, there are many potential fields and gravitation is one well-known potential field. The gravitational potential field is usually defined as a distance dependent vector field. Equation 3.1 represents the potential field generated by the force of gravity (F) between two objects of mass m_1 and m_2 separated by a distance d, where G is the gravitational constant.

$$F = G\frac{m_1 m_2}{d^2} \tag{3.1}$$

We can rewrite Equation 3.1 as cd^e, where $c = Gm_1m_2$, and $e = -2$. Each potential field of the form cd^e where d is distance has two tunable parameters (c, e) that determine the field effect. Figure 3.1 shows the interaction of attractive and repulsive potential fields, where an attractive potential field is generated by a goal

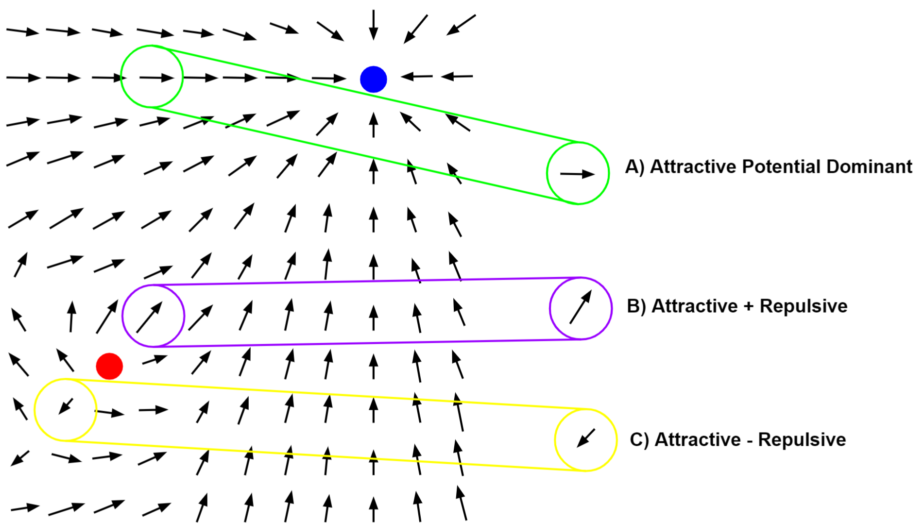

A) Attractive Potential Dominant
B) Attractive + Repulsive
C) Attractive - Repulsive

Figure 3.1: The figure shows a goal by blue circular object and an obstacle by red circular object. The goal exert an attractive potential field, the obstacle exert a repulsive potential field, and arrows represent the direction of resultant potential field.

(blue circle), and a repulsive potential field is generated by an obstacle (red circle). The arrows show the resultant vector sum of these attractive and repulsive potential fields. Figure 3.1 also shows three regions denoted by green, purple, and yellow circles. The green circle shows a region where an attractive potential field dominates, the purple circle shows a region where both attractive and repulsive fields act in the same direction, and the yellow circle shows a region where attractive and repulsive fields act in opposite directions. An agent located anywhere on the map moves in the direction of the arrow at that location. If we inspect closely, we can see in the figure that arrows move away from the obstacle except when the obstacle and the goal are aligned. This shows that if an agent moves in the direction of the arrows, it will avoid the obstacle. Potential fields have been used extensively in games [10, 11, 82] and robotics [92, 93, 94] for fast, real-time, collision free movement. In this thesis, we will focus on problem specific potential fields with a number of potential fields being used to control agent movement.

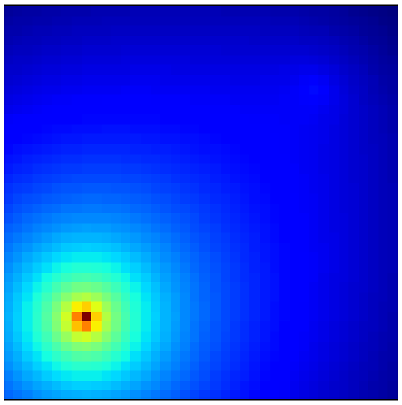


Figure 3.2: Influence of repulsive potential field around an obstacle shown by red and yellow pixels. The blue region shows that the magnitude of attractive potential dominates repulsive potential.

Figure 3.2 shows repulsive potential field around the obstacle. The blue colored region shows the dominance of the attractive potential field and as the agent goes towards the obstacle, the magnitude of repulsive potential increases.

3.1.1 Potential Field Based Movement Control

Potential field based movement control of agents was first introduced by Khatib [1]. In this section, we show how we use potential fields to guide agents. Figure 3.1 shows arrows that represent the direction of the resultant of attractive and repulsive potential fields computed using Equation 3.2.

$$\vec{P} = \vec{a_t} c_a d_g^{e_a} + \vec{a_o} c_r d_o^{e_r} \tag{3.2}$$

Here $\vec{a_t}$ and $\vec{a_o}$ are unit vectors pointing towards the goal and away from the obstacle, c_a and c_r are coefficients of attractive and repulsive fields respectively. e_a and e_r are exponents of attractive and repulsive fields, d_g and d_o are distances from

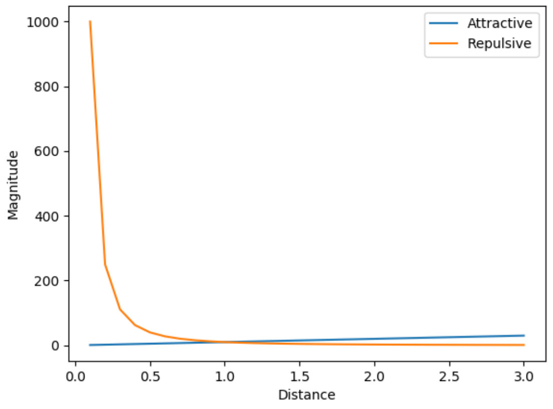


Figure 3.3: Attractive and repulsive potential fields magnitude.

the goal and the obstacle. An agent moves in the direction of the resultant vector $\vec{P}$. The first part of the equation accounts for the attractive potential towards the goal and the second part takes care of repulsion from the obstacle. Since the agent wants to avoid collision with the obstacle, we assume a negative value of exponent e_r so that the repulsive potential field magnitude increases as distance decreases. This make sure that as distance d_o decreases and the magnitude of repulsive potential field increases leading to collision avoidance. Figure 3.3 shows the magnitude of attractive potential field corresponding to d_g and repulsive potential fields corresponding to d_o. The figure shows that the magnitude of the repulsive potential field dominates the magnitude of the attractive potential field as distance decrease, and thus the agent will be pushed away from the obstacle leading to collision avoidance. Now we have shown that using potential fields we can guide UAVs without collision, in the next subsection we specify the agent's motion model used in our simulation.

3.1.2 Agent Motion Model

In our simulation, agents move in 3D by setting a desired heading (dh), a desired altitude (da), and a desired speed (ds). At each time step (δt), the desired heading is computed using the potential field equation and an agent tries to move the direction of desire heading and achieve the desired speed by changing its current speed (s) according to the acceleration (r_s). The values of current speed (s), heading (h), and acceleration (a) are calculated using Equations 3.3, 3.4, 3.5 respectively.

$$s = s \pm r_s \delta t \tag{3.3}$$

$$h = h \pm r_t \delta t \tag{3.4}$$

$$a = a \pm r_c \delta t \tag{3.5}$$

Here $\pm$ depends on whether desired value is grater than or less than current value, r_t is turn rate. From speed, heading, and altitude, we compute 3D agent velocity (vel) and position (pos) as follows:

$$\vec{\mathbf{vel}} = (s * cos(h), 0, s * sin(h))$$

$$\vec{\mathbf{pos}} = \vec{\mathbf{pos}} + \vec{\mathbf{vel}} * \delta t$$

$$\mathbf{pos}.y = a$$

Here, bold text indicates vector variables, the xz plane is the horizontal plane, the y-coordinate is height, and the agent points along its heading. In this thesis, we optimize potential field parameters using genetic algorithms and to evaluate the fitness of individuals/candidate solutions, we ran our problem specific simulation for a pre-defined number of steps. In the next section we describe fitness evaluation of candidate solutions.

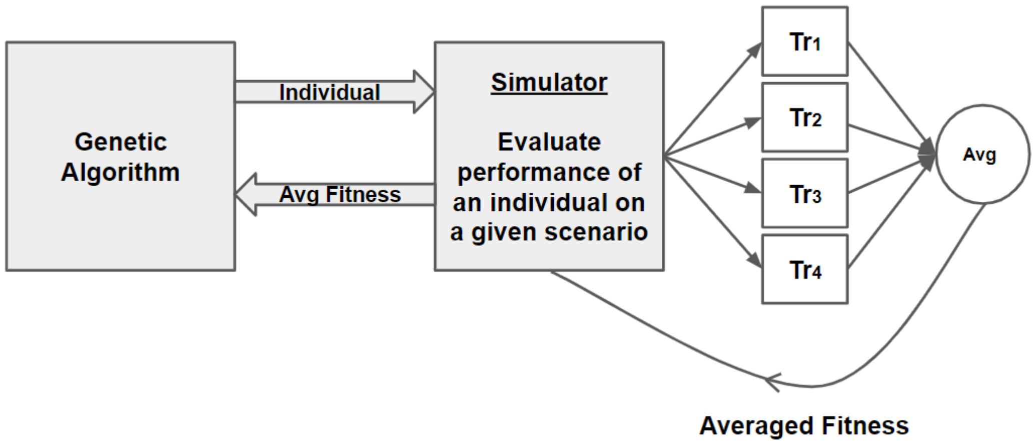


Figure 3.4: Individuals are evaluated on 4 different training scenarios and the fitness is averaged over these 4 training scenarios.

3.2 Evaluation

To find the quality of an individual, we evaluate the individual in a given scenario. However, earlier experience showed that solutions evolved in a scenario may not produce similar performance on unseen scenarios. In other words, evolved solutions may not be robust. In this thesis, different scenarios refer to different locations of users and their bandwidth requirements in the WMND problem, different spreading of fire boundaries, and different locations of game agents in RTS micro. To improve the robustness of evolved solutions, we evaluate individuals on multiple different scenarios. We call these scenarios *training scenarios*. Once we find a good solution, we test the robustness on scenarios not seen during training which we call *testing scenarios*.

Chapter 4, 5, and 6 specify training and testing scenarios for wireless mesh network deployment, dynamic fire boundary coverage, and RTS micro respectively. Figure 3.4 shows a block diagram of how to evaluate an individual through a simulator and return a fitness for the individual being evaluated. Each individual is

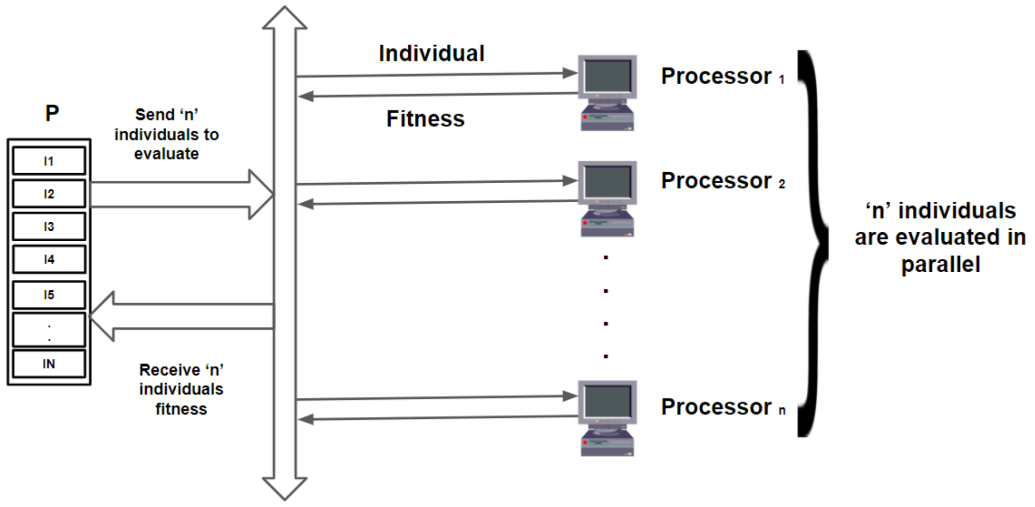

P
I1
I2
I3
I4
I5
IN
Send 'n' individuals to evaluate
Receive 'n' individuals fitness
Individual
Fitness
Processor 1
Processor 2
Processor n
'n' individuals are evaluated in parallel

Figure 3.5: Parallel GA

evaluated on four different training scenarios. Each scenario returns a fitness and the average of these finesses is assigned to the individual being evaluated. Running the simulation for a pre-defined number of steps is the most time consuming process. Since we use 4 different training scenarios to evaluate the quality of an individual, the total time needed for evaluation increases 4 fold. Thus, to reduce the total running time for GA, we parallelize the evaluation of individuals.

3.3 Parallel GAs

One feasible way to run our GA faster is by parallelizing the evaluation since most of the computational load comes from evaluation in the simulator. Most parallel programs adopt the idea of a divide and conquer strategy to split a task into sub-tasks and solve sub-tasks simultaneously using multiple processors. This divide and conquer approach can be used in GAs too. There are three main types of Parallel GAs (PGAs): global single-population server-client GAs, single-population fine-grained GAs, and multiple-population coarse-grained GAs [95]. We use a global single-population server-client PGA to evaluate candidate solutions for all problems. In a server-client PGA, there is a single population as in a canonical GA, but the evaluation of fitness is distributed among several processors, as shown in Figure 3.5. Since selection and crossover operate on the entire population, this type of parallel GA is also called global single-population parallel GA.

CHAPTER 4
WIRELESS MESH NETWORK DEPLOYMENT

Deployment of UAV based ad-hoc networks in unknown environments is a challenging problem. In unknown environments, the users' distribution is not known and thus the first challenge is to find the users' distribution. Once users are discovered and their positions known, re-deploying UAVs in real-time to serve all discovered users specifies the second challenge. In order to solve both problems we divide the network deployment problem into two phases: search and service. In the search phase, UAVs are deployed to cover the entire area of interest to find all users' locations and bandwidth requirements. The second phase utilizes this information to move UAVs to optimize user service.

Given a sufficient number of UAVs, several existing algorithms can position UAVs to cover the area of interest in the first search phase and we show how distance based potential fields may be used to mimic one of these algorithms. Once found, we need to serve users by establishing communication between users and the command center so that we can find out their rescue needs and help them find their way out of the affected area or target rescue efforts to them. In this service phase, we use another set of potential fields to guide UAVs, and we propose a genetic optimization approach to optimize the sum of coverage and longevity in order to optimally deploy UAVs acting as wireless access points providing an ad-hoc wireless network to serve users in the area of interest. UAV based ad-hoc network deployment during the second phase is an NP-hard problem [96] and we call this the network deployment problem in this thesis. Network deployment presents challenges in optimal positioning, routing, and fast deployment. We use potential fields, which are highly non-linear, to guide UAVs, and thus we use a genetic algorithm to search for and find potential field parameters leading to optimal bandwidth coverage and longevity.

In this thesis, we present a new representation and use a genetic algorithm to optimize UAV based wireless network deployment. Our work is different from prior work in that not only do we provide for positioning information we also provide collision free movement and use the same representation, potential fields, for both search and serve phases. Before explaining how potential fields control the movement of UAVs, we first formulate the problem.

4.1 Problem Formulation

We formulate network deployment as an optimization problem and describe how each candidate solution is evaluated. Assume that we have a set of N homogeneous UAVs $U = \{u_1, u_2, ..., u_n\}$ that need to be deployed in a given AOI of 2000×2000 $meters^2$ with the command center located in the middle at (1000, 1000). All UAVs start at the center within a 10×10 $meter^2$ area and fly at a constant altitude of 100 meters. Each UAV is equipped with sensors to find users on the ground within a ground sensing range (U_{gr}) of 100 meters and can communicate with its neighbors within an air to air communication range (U_{ar}) of 300 meters. A total of m users each with a position and a bandwidth requirement (p_i, b_i) are distributed over the AOI. Thus users can be denoted by the list of pairs: $\{(p_1, b_1), (p_2, b_2), ..., (p_m, b_m)\}$. These simulation parameters can be easily changed.

Since users' locations and their bandwidth requirements are not known prior to finding them, UAVs first need to search and find all users and their bandwidth requirements by covering the AOI and then second, move to serve users that have been found and avoid areas devoid of users. Thus, we divided the network deployment problem into two phases; search and service. We formulate the problem of network deployment during the service phase as an optimization problem to maximize bandwidth coverage and longevity of deployed networks. To maximize bandwidth coverage, UAVs must be placed such that all users are covered, and to maximize longevity the energy consumption of UAVs must be reduced.

Each UAV has a limited initial energy of 10^6 joules and consumes energy while hovering/moving and providing data to users. Intuitively, if more UAVs share the bandwidth demanded by users, the per UAV bandwidth service can be reduced leading to less energy consumption and longer UAVs flight times. Thus the objective is to deploy UAVs to maximize bandwidth coverage while sharing bandwidth demands. We classify each UAV either as an Active UAV (AU) or an Inactive UAV (IU) based on whether or not the UAV is serving a user. More specifically, if a UAV finds users within its ground sensing range, U_{gr}, the UAV is an active UAV, otherwise we call it an inactive UAV. This is shown in Figure 4.5.

The number of active UAVs is proportional to network lifetime. The more AUs, the more bandwidth can be distributed among AUs leading to less power consumption per AU and thus longer battery life to stay aloft and keep the network alive. A variable $\alpha_i \in \{0, 1\}$ represents the status of UAVs where α_i is 1 for active and 0 for inactive UAVs. Equation 4.1 specifies the objective function where i sums over N UAVs and u_{bi} represents the bandwidth served by the i^{th} UAV, b_j is the bandwidth requirement of the j^{th} user, m is the number of users, and N is the total number of (homogeneous) UAVs.

$$Maximize \quad f = \left[\frac{\sum_{i=1}^{N} u_{bi}}{\sum_{j=1}^{m} b_j} + \frac{\sum_{i=1}^{N} \alpha_i}{N} \right] \tag{4.1}$$

In Equation 4.1, the first term counts the fraction of bandwidth served over the total demand and the second term counts the fraction of active UAVs. Each term in the equation can have a maximum value of 1 and thus the maximum objective function value (their sum) will be 2. When computing the objective funtion using Equation 4.1, several constraints must be satisfied. Each UAV must be positioned within the specified AOI, thus the UAV's x and z coordinates must be $\in (0, 2000)$[1]. A UAV can provide a maximum of u_{bmax} to users, and thus $u_{bi} \leq u_{bmax}$. UAVs can either communicate with their direct neighbors (1-hop) or with neighbors of neighbors (2-hop), and thus a UAV can communicate with other UAVs up to a distance of $2 \times U_{ar}$. We assume that deployed UAVs use the 2 MHz communications channel with spectral efficiency of 2.5 bps/Hz [97]. Each UAV can serve a data rate of 5 Mbps to users. Users' bandwidth requirements can range from simple text message communication to HD video streaming services during, for example, a health emergency. We thus assume that a user's maximum bandwidth requirement is 3 Mpbs which is enough for video communication. Table 4.1 lists the parameters used to model a UAV.

4.2 Methodology

We start by providing some background on genetic algorithms[2]. We then describe our training and testing scenarios, our model of UAV movement, and how we use potential fields for each of the two phases of network deployment in our simulation.

[1]Our simulation uses the xz plane for the horizontal plane.
[2]If you are familiar with genetic algorithms, you can skip this subsection

4.2.1 Genetic Algorithms

Our approach works in two phases and uses potential fields to control the movement of UAVs in both phases. First, to find the users' distribution, and second, to re-deploy UAVs to serve found users better. Since potential fields are highly non-linear, tuning the potential field parameters is difficult, and thus we used a $(\mu + \lambda)$ elitist genetic algorithm [98] to optimize parameters. Note that we use a GA only in the second phase of network deployment as the first phase of network deployment is a relatively simpler problem. Computing fitness (performance) using Equation 4.1, our elitist genetic algorithm shown in Algorithm 1 searches through the space of potential field parameter values in order to maximize fitness.

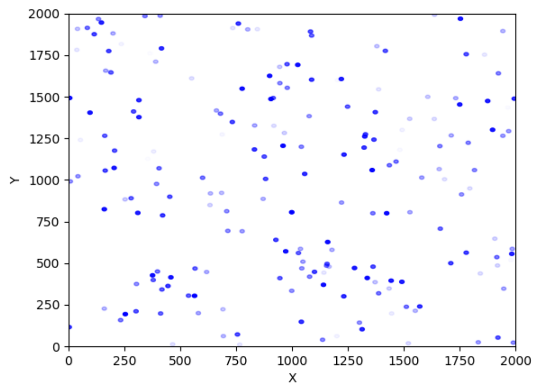
Algorithm 1: An Elitist Genetic Algorithms

```
Input  : $Pop, Gen, P_x, P_m, \lambda, H$
Output: Best Solution
1  $P_0 \leftarrow$ Initialize($Pop$);
2  Evaluate($P_0, H$);
3  t=0;
4  $P_c$ = [];
5  while $t \leq Gen$ do
6      if $i \leq \lambda$ then
7          $p_1, p_2 \leftarrow SelectParents(P_t)$;
8          $c_1, c_2 \leftarrow Crossover(p_1, p_2, P_x)$;
9          $c_1 \leftarrow Mutate(c_1, P_m)$;
10         $c_2 \leftarrow Mutate(c_2, P_m)$;
11         $P_c.add(c_1, c_2)$;
12         $i = i + 2$;
13     end
14     Evaluate($P_c, H$);
15     $P_{t+1} \leftarrow NextGenIndividuals(P_t, P_c)$;
16     $P_c.clear()$;
17     t=t+1;
18 end
```

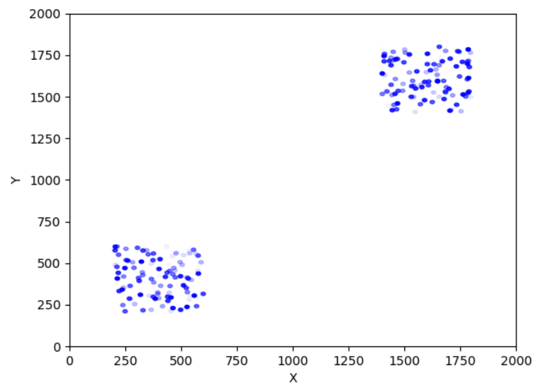
Algorithm 1 describes a our genetic algorithm. *Pop* is the population size, *Gen* is the maximum number of iteration for evolution, P_x, P_m are probabilities of crossover and mutation, and H is the number of hops for communication between UAVs. We use $(\mu + \lambda)$ elitism where μ is the population size (*Pop*) and λ is number of children generated through recombination. Our experiments set $\lambda = \mu$. Initially, the GA randomly generates a parent population (P_0) of candidate solutions (CS) and evaluates the fitness of each CS using our UAV-based network deployment simulator. To generate the next population, we select two candidate solutions (p_1, p_2) from the parent population tournament selection with tournament size of two [11]. Next, these two parent candidate solutions (p_1, p_2) exchange information with each other through a crossover operator with a probability of P_x and produce two children c_1, c_2. Using the mutation operator, each gene of c_1, c_2 is mutated with a small probability of P_m. This process of producing children continues until λ children are produced. Each child solution is then evaluated and assigned a fitness using Equation 4.1. In $(\mu+\lambda)$ elitism, the μ best candidate solutions are selected from the combined parent and offspring pool of $(\mu + \lambda)$ individuals in the population. In Algorithm 1, `NextGenIndividuals` does this down selection operation and the process of evolution goes on for *Gen* number of iterations. Table 4.1 shows all the GA parameter values used in our experiments. In the next subsection we discuss the scenarios used by our simulation to evaluate candidate solutions in the genetic algorithm population.

Evolving Robust Solutions

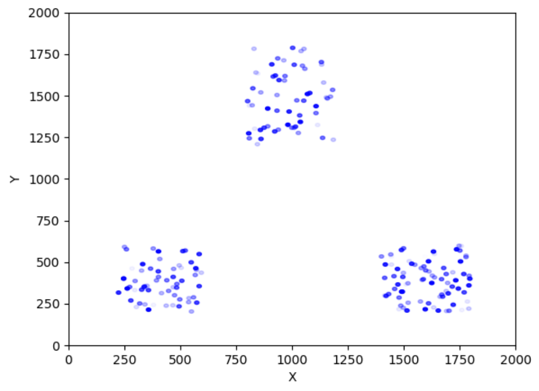
We aim to evolve high quality and *robust* solutions for wireless network deployment. However, earlier work has shown that solutions evolved on one scenario with a particular set of parameters may not be robust [7, 10, 47]. That is, in our



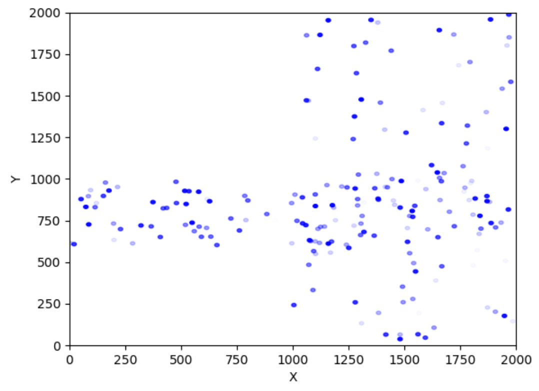
(a) T_{r1}



(b) T_{r2}



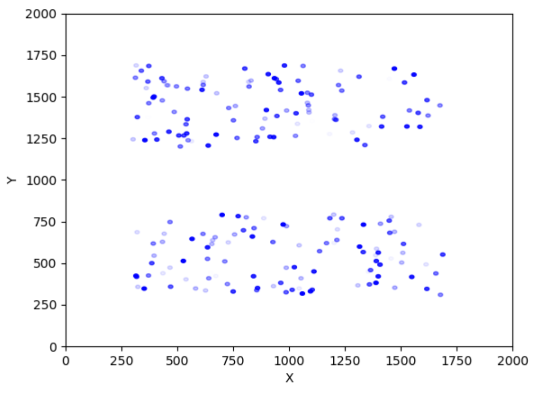
(c) T_{r3}



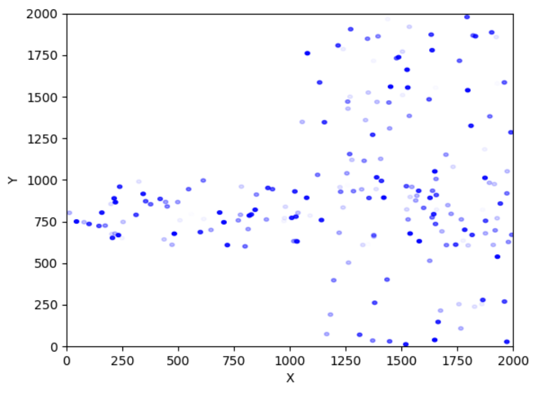
(d) T_{r4}

Figure 4.1: The four training scenarios with different users' distribution. Blue colored dots show the locations of users, darker the dots more bandwidth required and vice-versa.

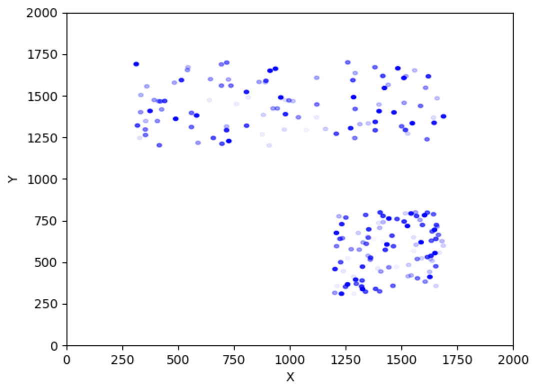
case, potential field parameters evolved on one scenario may over-fit and thus **not** work well in other scenarios. We therefore created four *training* scenarios with different user distributions as shown in Figure 4.1. In these scenarios 200 users are distributed uniformly (T_{r1}), in clusters (T_{r2}, T_{r3}), and in the right and center region of a 2000×2000 $meter^2$ AOI. With a UAV ground sensing range of 100 meters, we need 156 UAVs to cover the entire AOI without coverage gaps in order to find all users. The genetic algorithm then seeks to maximize the average objective function value, computed using Equation 4.1, over all four scenarios. Once the genetic



(a) T_{e1}



(b) T_{e2}



(c) T_{e3}

Figure 4.2: The three testing scenarios distinct from training scenarios and never seen during parameter evolution.

algorithm evolves good parameter values, we test the robustness of these potential field parameters on 100 different unseen *test* scenarios. Three of these test scenarios are shown in Figure 4.2 in order to see that the distribution of users in these scenarios is different from the distribution in training scenarios. The aim is to show that the genetic algorithm evolved potential field parameters work across a wide range of user distributions in the AOI and our results (Section 4.3) show that average performance over the four training scenarios translates robustly to testing scenarios.

4.2.2 UAV Movement Modeling

Initially, each UAV starts near the center of the AOI and has zero speed. The speed (s) increases, with a constant acceleration (r_s) of 0.1 $meter/sec^2$ until speed becomes the max speed of 15 $meter/sec$, using Equation 4.2. Here Δt is the simulation time step interval.

$$s = s \pm r_s \Delta t \tag{4.2}$$

Each UAV moves under the influence of a set of potential fields and the direction of the vector sum of these potential fields provides the desired heading or direction for the UAV. The k^{th} UAV experiences attractive ($\vec{P_a}$) and repulsive ($\vec{P_r}$) potential fields from neighboring UAVs within the range of U_{ar} and given by Equation 4.3. Each potential field is of the form cd^e where d is distance and has two optimizable parameters (c, e) that determine field effect.

$$\begin{aligned} \vec{P_k} &= \sum_{i=0}^{n} \vec{P_a} + \sum_{i=0}^{n} \vec{P_r} \\ \vec{P_k} &= \sum_{i=0}^{n} \vec{a_{ki}} c_a d_{ki}^{e_a} + \sum_{i=0}^{n} \vec{a_{ik}} c_r d_{ki}^{e_r} \end{aligned} \tag{4.3}$$

Here $\vec{a_{ki}}$ is a unit vector pointing from the k^{th} UAV to the i^{th} neighbor and $\vec{a_{ik}}$ is the unit vector pointing from i^{th} neighboring UAV back to the k^{th} UAV. n is the number of h-hop neighbor UAVs, $d_{ki} = d_{ik}$ is the distance between the k^{th} and i^{th} UAVs. c_a, c_r are potential field coefficients and e_a, e_r are potential field exponents. The GA optimizes these coefficients and exponents to specify UAV movement. $\vec{P_k}$'s direction specifies the k^{th} UAV's heading. In our model, UAVs change their current heading to this new desired heading instantaneously. Using speed and heading, we compute a UAV's velocity ($\overrightarrow{vel}$) and position ($\overrightarrow{pos}$) as follows

$$\begin{aligned} \overrightarrow{vel} &= (s \times cos(heading), 100, s \times sin(heading)) \\ \overrightarrow{pos} &= \overrightarrow{pos} + \overrightarrow{vel} \times \Delta t \end{aligned} \tag{4.4}$$

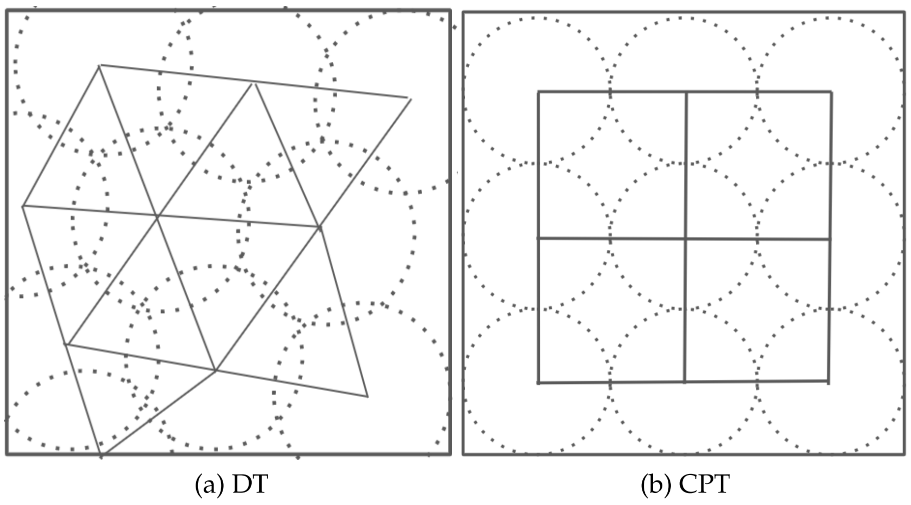

(a) DT (b) CPT

Figure 4.3: Networks deployed using DT and CPT in a square AOI. Dashed circle shows the coverage of a UAV.

Having specified our movement model, we now describe how we set potential field parameters for area coverage during the first phase.

4.2.3 First Phase: Search

UAVs move to cover the entire AOI during the first phase to find users' distribution. This is also called static network deployment and we use potential fields to deploy UAVs to maximize area coverage with minimal overlap. Since Delaunay Triangulation maximizes area coverage with minimal overlap, we want to mimic DT style network deployment with potential fields. Figure 4.3 shows static networks deployed using DT and CPT. We use Ming's [40] Delaunay Triangulation based result that shows that the distance between any two adjacent UAV must be $\sqrt{3}\ U_{gr}$ to cover the AOI with minimum coverage overlap. To model Delaunay Triangulation with potential fields, we thus need to find values of c_a, c_r, e_a, e_r such that when $d_{ki} = \sqrt{3}\ U_{gr}$ the magnitude of the attractive potential field equals the mag-

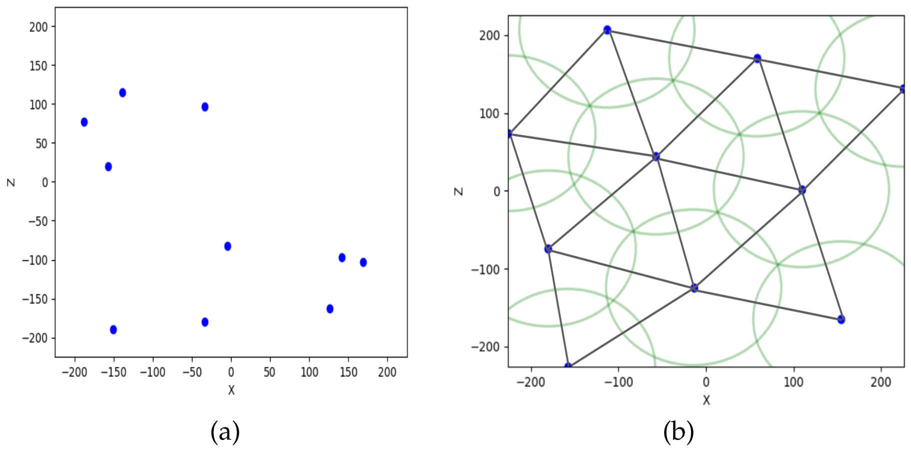


Figure 4.4: 10 randomly distributed UAVs (a), and each UAV moves under the influence of attractive and repulsive potential fields with parameters ($c_a = 1, e_1 = 1, c_r = 3 \times \sqrt{3} \times 100^3, e_r = -2$). Figure (b) shows that using potential fields, we can get network deployment similar to networks inspired from DT.

nitude of the repulsive potential field. This is shown by Equation 4.5. In the equation, the directions of unit vectors $\vec{a_{ki}}$ and $\vec{a_{ik}}$ are opposite to each other and thus we substitute $\vec{a_{ik}}$ = - $\vec{a_{ki}}$. In addition, earlier work in potential fields based movement has shown that the attractive potential should be proportional to distance and the repulsive potential field should be inversely proportional to distance squared to provide collision free cohesive movement [81, 82, 99]. Thus we set $e_a = 1$, $e_r = -2$, and $d_{ik} = \sqrt{3}U_{gr}$ and ignore the unit vectors in Equation 4.5 to find the relationship between c_a and c_r.

$$\begin{aligned} \vec{a_{ki}} c_a d_{ik}^{e_a} &= \vec{a_{ki}} c_r d_{ik}^{e_r} \\ c_a(\sqrt{3}U_{gr})^1 &= c_r(\sqrt{3}U_{gr})^{-2} \\ c_r &= c_a(\sqrt{3}U_{gr})^3 \end{aligned} \tag{4.5}$$

We then tested this derived relationship in our simulation to see whether we can mimic Delaunay Triangulation-like area coverage with potential fields. In our experiment, we used U_{gr} = 100 meters and, for simplicity, assumed $c_a = 1$ to derive the value of c_r to be $3\sqrt{3}\ 100^3$ by simplifying Equation 4.5. These parameter values of $c_a = 1, e_1 = 1, c_r = 3\sqrt{3}\ 100^3$, and $e_r = -2$ were then used to control the movement of 10 randomly distributed UAVs as shown in Figure 4.4(a) and the resulting deployment covering the AOI is shown in Figure 4.4(b).

Noting the similarity to Figure 4.3, Figure 4.4 shows that with these parameter values, UAVs moving under the influence of potential fields can be used to mimic DT and cover the entire area with minimal overlap. Note that there exist multiple sets of values of the coefficients and exponents that will result in similar coverage. Assuming we can, in general, directly derive potential field parameter values for the search phase we focus more on the service phase.

Using the same potential field parameter values, we moved to our main scenarios and deployed 156 UAVs in an AOI of $2000 \times 2000\ meter^2$. UAVs start within an area of $10 \times 10\ meter^2$ near the center of the AOI and our simulation of the first phase runs for 1500 time steps which is sufficient for the UAVs to move into position. These positions at the end of the first phase are shown in Figure 4.5. In the figure, users, represented with blue dots, are distributed in three clusters. Green circles show UAV coverage areas and you may assume that there is a UAV at the center of every green circle. The red circle represents the command center. A black circle shows a UAV serving no users within its ground sensing range (U_{gr}) and is an example of an inactive UAV. Similarly, the magenta circle shows a UAV serving multiple users and is an example of an active UAV.

Since increasing the number of UAVs that serve the same users and thus share bandwidth decreases UAV energy consumption and thus increases network longevity, the second phase seeks to also increase the number of active UAVs by moving UAVs from areas with lower bandwidth requirements (fewer users) to areas with higher bandwidth requirements (more users) while maintaining network connec-

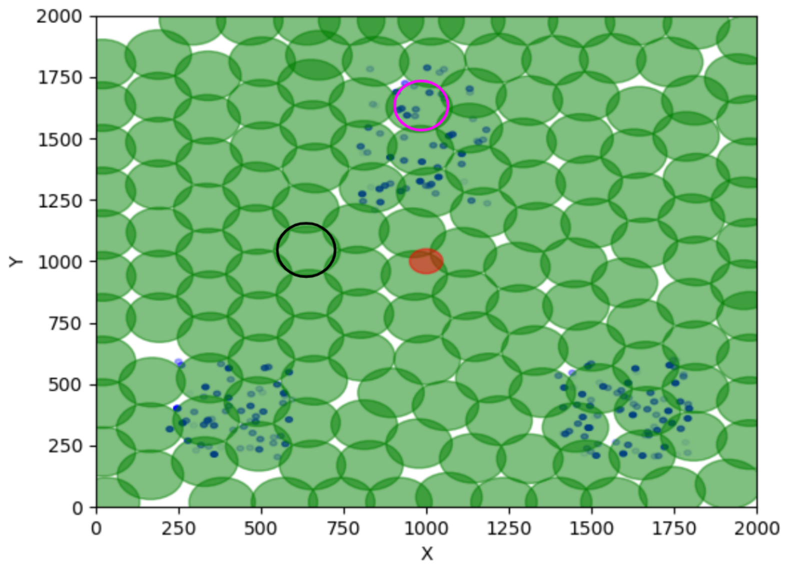


Figure 4.5: Deployment of UAVs at the end of the first phase on the third training scenario. Blue dots represent users, green circles represent UAVs' coverage areas, and the red circle represents the location of the command center. A black circle shows an inactive UAV and the magenta circle shows an inactive UAV.

tivity with the command center.

4.2.4 Second Phase: Service

In this phase, a different set of potential fields optimized by a genetic algorithm re-deploys UAVs to serve found users better. Algorithm 2 specifies the Genetic Adaptive Network Deployment Algorithm (GANet) that runs the simulation to compute the fitness of candidate solutions. That is, the *Evaluate* function of Algorithm 1 runs Algorithm 2 to evaluate each candidate solution's fitness. Once computed, this fitness drives evolutionary optimization.

Algorithm 2: Evaluation and fitness computation

```
Input  : $UAVs, Users, CS, MS, MT, H$
Output: fit
fit = 0;
for scenario in MS do
    t, bwc, AUs = 0;
    while $t < MT$ do
        AssociateUsers($UAVs, Users$);
        FindNeighbors($UAVs$);
        ActivePotentials($H, CS$);
        Dir = ComputeDir();
        MoveAll(Dir);
        t=t+1;
    end
    for i in UAV do
        bwc += $u_{bi}$;
        AUs += $\alpha_i$;
    end
    fit += bwc / MaxBW + AUs / UAVs;
end
fit = fit / MS;
return(fit);
```

Simulation and Evaluation

In Algorithm 2, UAVs and Users are lists of available UAVs and users respectively. The simulation runs for a maximum number of time steps ($MT = 1500$) on each training scenario. As mentioned earlier, we use four training scenarios and thus MaxScenarios (MS) is four. At each time-step (t), `AssociateUsers` scans users within U_{gr} for each UAV in the list and computes these users' bandwidth consumption. `FindNeighbors` finds other UAVs within U_{ar} and records them as neighbors. `ActivePotentials` uses the number of hops and potential field parameter values supplied by a candidate solution from the genetic algorithm's population, to compute potential field values. Now that the potential field information needed is available, `ComputeDir` computes potential field directions at each UAV by summing all potential fields acting on a UAV using Equation 4.6 and stores this in `Dir`. Next, `MoveAll(Dir)` moves each UAV in the direction computed. After *MT* time steps, the algorithm computes the fitness for each training scenario by summing the bandwidth coverage fraction and active AU fraction. Finally, once the evaluation on all four scenarios is over, the algorithm returns the average of these fitness values as the fitness of this candidate solution to the genetic algorithm.

$$\vec{P_k} = \vec{P_b} + \sum_{i=1}^{h}(\vec{P_{u_i}} + \delta_r \vec{P_{r_i}}) + \delta_c \vec{P_c} \tag{4.6}$$

Note that in Equation 4.6, h refers to the number of hops for communication between UAVs. The vector sum of the potential fields for UAV k given by P_k provides a desired heading for the k^{th} UAV to turn towards.

Each term in the equation describes a potential field whose parameters need to be optimized (or tuned). Since we want UAVs to be attracted towards users based on their bandwidth requirements, we use an attractive potential field, ($\vec{P_b}$), to model this attraction. To reduce per UAV bandwidth coverage and save energy, the k^{th} UAV uses an attractive potential field towards neighboring UAVs whose magnitude depends on the neighbors bandwidth load ($\vec{P_u}$). At the same time, to avoid collisions, a repulsive potential field, $\vec{P_r}$, ensures that UAVs do not collide with each other. This potential field only applies to UAVs that are less than d^* distance away and we use the Dirac delta function, δ_r to model this. δ_r has value 1 when UAV j is less than d^* distance away and 0 otherwise. When a UAV has no users within its user sensing range and no UAVs within U_{ar}, that is, the UAV is inactive, we want it to move towards the command center as a default behavior. We model this with an attractive potential field, $\vec{P_c}$, based on distance to the command center. δ_c is another Dirac delta function to ensure that this potential field only acts on inactive UAVs.

Assuming UAVs communicate only with one-hop neighbors, Equation 4.6 can be rewritten as Equation 4.7 below.

$$\vec{P}_k = \sum_{i=0}^{m_k} c_1 \vec{b}_i^{e_1} + \sum_{j=0}^{n_k} (c_2 \vec{u}_{kj}^{e_2} + c_3 \vec{r}_{kj}^{e_3}) + c_4 \vec{s}_k^{e_4} \tag{4.7}$$

Here c_1, c_2, c_3, c_4 are the coefficients and e_1, e_2, e_3, e_4 the exponents of the potential fields in Equation 4.6. The first summation is over m_k which is the number of users within UAV sensing range while the second summation is over n_k which is the number of one hop neighbors. $\vec{b}$ is the bandwidth required by the i^{th} user and points towards the user. $\vec{u}$ is the bandwidth being served by the j^{th} 1-hop neighbor of k and points towards this neighbor. $\vec{r}$ is the vector difference between the positions of UAV k and UAV j and points from j to k. In the last term, $\vec{s}$ is the vector difference between the command center position and UAV k pointing from k to the command center. Including four coefficients, four exponents, and d^*, we have a total of nine parameters to optimize. When UAVs are allowed to communicate with upto two-hop neighbors, Equation 4.6 can be re-written as Equation 4.8.

$$\vec{P}_k = \sum_{i=0}^{m_k} c_1 \vec{b}_i^{e_1} + \sum_{j=0}^{n_k} (c_2 \vec{u}_{kj}^{e_2} + c_3 \vec{r}_{kj}^{e_3}) + \sum_{j'=0}^{n_{k'}} (c_2 \vec{u}_{kj'}^{e_2} + c_3 \vec{r}_{kj'}^{e_3}) + c_4 \vec{s}_k^{e_4} \tag{4.8}$$

Here $n_{k'}$ is the number of two hop neighbors, that is, the number of one-hop neighbors' neighbors. Note that for $h = 1$ and $h = 2$, the k^{th} UAV experience 4 and 6 different potential fields respectively. In general for h-hop communication, the k^{th} UAV will experience $2(h + 1)$ different potential fields. Since each potential field has two tunable parameters (c, e), a total of $2 \times 2(h + 1)$ potential field parameters will need to be optimized. Due to the non-linearity of these potential fields, optimizing these parameters is difficult and we thus use a genetic algorithm. Each coefficient and exponent is encoded in a chromosome, where $c's \in (-8192, 8192)$ with a precision of 1, $e's \in (-5.12, 5.12)$ with precision 0.01, and d^* ranges between $0 \leq d^* \leq 2^8$. We then compared deployed network performance using GANet against the performance of the current state-of-the-art ATRI algorithm.

4.3 Results and Discussion

We start by evaluating genetic algorithm parameter tuning performance on a simple tractable problem by comparing genetic algorithm performance against a random exhaustive search and a hill climber (gradient ascent). Once we show that the genetic algorithm finds the optimum significantly more quickly than either of the other algorithms, we run the GA on the larger, more realistic problem as in the four training scenarios. Table 4.1 show all simulation parameter values used during experiments.

4.3.1 Experiments on A Simple Test Problem

We begin with a test problem to deploy 10 UAVs in a small AOI of 450×450 $meter^2$ where the command center is located at $(0, 0)$. We want to provide bandwidth coverage to 10 users as shown in Figure 4.6(a). Using one-hop information, each UAV moves in a direction computed using Equation 4.7. On this simple problem we applied three search techniques for finding optimal potential field parameters that result in the optimum fitness of 2; an exhaustive search, a hill climber (gradient

Table 4.1: Simulation Parameters

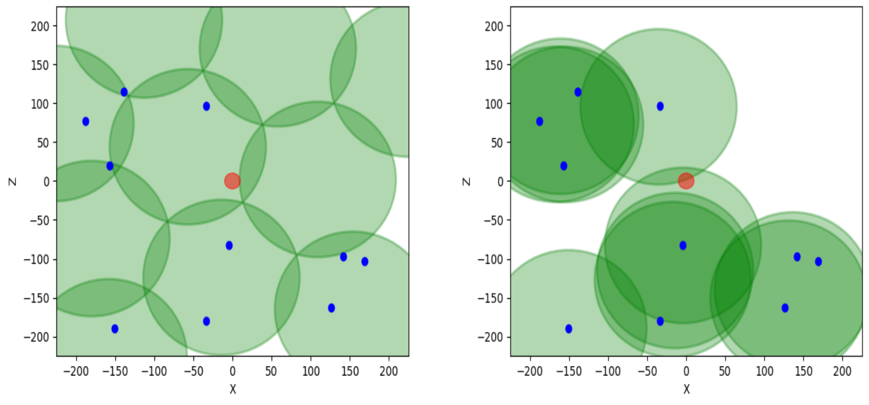
Parameters	**Value**
UAV	
Area of Interest	$2000m^2$
Height of UAVs	100m
Coverage Radius of UAVs	100m
A-2-A communication Range	300m
Max Speed of UAV	15 m/s
Initial energy of UAV	1000×10^3 J
Hovering energy	98 J/s
Energy loss per 1 Mbps data served	5 J
Genetic Algorithm	
Population Size	20
Max Iteration	20
Crossover Rate	0.95
Mutation Rate	0.05
Lambda	20

Table 4.2: Comparing three search algorithms

Techniques	**Evaluations**
Hill Climber	105
Exhaustive Search	92
Genetic Algorithm	**39**

assent) based search technique, and our genetic algorithm.

Exhaustive search randomly samples points in the search space of parameter values and computes their fitness until it finds a point with fitness 2. We run this randomized exhaustive search algorithm with 10 different random seeds and store the number of evaluations (fitness computations). We also run the hill climber and genetic algorithm 10 different times with 10 different random seeds and store the number of evaluations needed by each of them to find the optimum of 2. Table 4.2 compares the performance of the three algorithms and shows that the genetic algorithm finds the optimum in significantly fewer evaluations than the others. The



(a) Initial positions (b) Optimal positions

Figure 4.6: Optimal UAV deployment within a small AOI with 10 users (blue dots). Initial positions covering the area (a), final positions focusing on user locations and bandwidth sharing (b).

genetic algorithm (39) is more than twice as fast as its nearest competitor (92).

The results suggest that this simple problem is multi-modal with multiple different optimal solutions. Starting with UAV positions (center of green circles) at the end of the search phase shown in Figure 4.6(a) the following parameter values found by the genetic algorithm, $c_1 = 50, e_1 = 0.5, c_2 = 0.1, e_2 = 0.1, c_3 = 7000, e_3 = -5, c_4 = 0.1, e_4 = 0.1$, resulted in the optimal UAV deployment (with fitness 2) shown in Figure 4.6(b). Note that with changes in UAV's attributes, users' requirements, or size of AOI, the complexity and scale of the problem changes and we will need to run the genetic algorithm again. However, the GA optimized parameter values are robust to numbers of UAVs and the number and distribution of users.

Having shown that genetic algorithms are good for tuning potential field parameters in our problem domain, we move to a larger more realistic problem and compare performance against the ATRI algorithm of Ming [40].

4.3.2 Evolution on Training Scenarios

We started with 200 users distributed as shown in Figure 4.1, 156 UAVs, and one hop communications. We assume that UAVs start within an area of $10 \times 10\ m^2$ near the command center. During the first phase each UAV moves for 1500 time steps to maximize area coverage and find users. Figure 4.5 shows UAV positions at the end of the first phase on the third training scenario. The figure shows that a few UAVs (who found users within their coverage range) have started providing bandwidth coverage demanded by users while many other UAVs hover idle throughout the coverage area. In the second phase of network deployment we aim to move UAVs towards different sub-regions in the AOI that demand wireless bandwidth. The simulation runs for 1500 time steps for UAVs to move towards users under the influence of a GA optimized set of potential fields.

The potential field parameters were encoded in a population of 20 individuals and the GA evolved solutions on the four training scenarios for 20 generations. As is common practice in the GA community when dealing with very long run time fitness computations, we ran the GA 10 times with different random seeds. The best solution over these 10 runs, ($Best_{1h}$, best one hop fitness) achieved a fitness of 1.64 out of 2. We observed that the network deployed using $Best_{1h}$ contains a number of inactive UAVs which contribute to the reduced fitness. Thus, to improve performance, we next tested with two hop communication. Results shows that the fitness of the best solution obtained with two hop UAVs ($Best_{2h}$) was 1.95 which is significantly better than one hop and is near-optimal. For comparison, the same experiments were conducted using ATRI and the ATRI deployed network achieved a performance of 1.38. Figure 4.7 shows the fitness obtained using $Best_{1h}$, $Best_{2h}$, and ATRI on the four training scenarios as well as the average performance across the

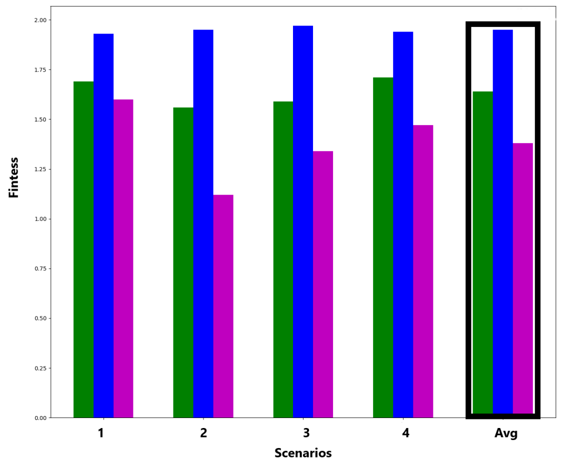
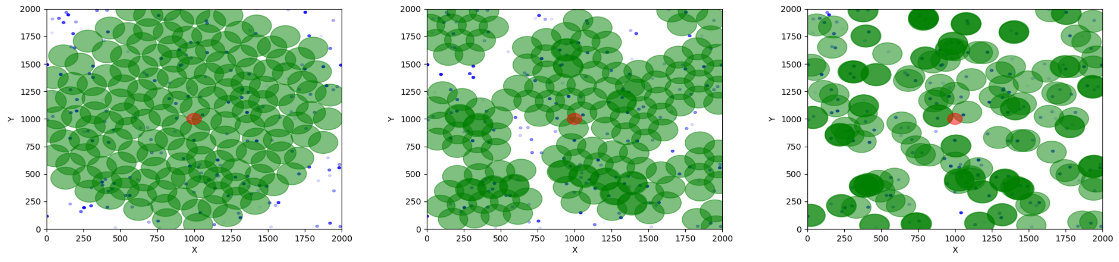


Figure 4.7: Comparing GANet one-hop (green), two-hop (blue) versus ATRI (magenta) on four training scenarios. The rightmost bars show averages across the four scenarios.



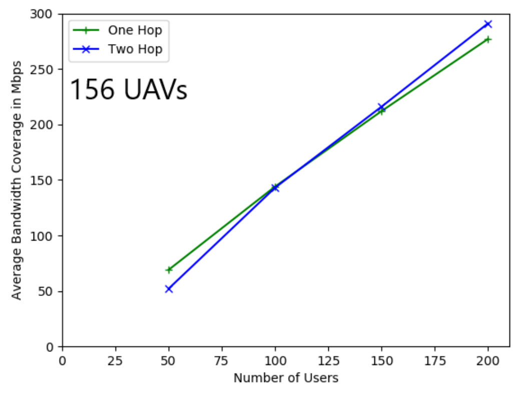
(a) ATRI (fitness=1.60) (b) $Best_{1h}$ (fitness=1.69) (c) $Best_{2h}$ (fitness=1.93)

Figure 4.8: Comparing network deployment on our first training scenario. ATRI (a) versus GANet 1-hop (b) and 2-hop (c). GANet obtains better performance by focusing more on areas with users and avoiding areas with no users.

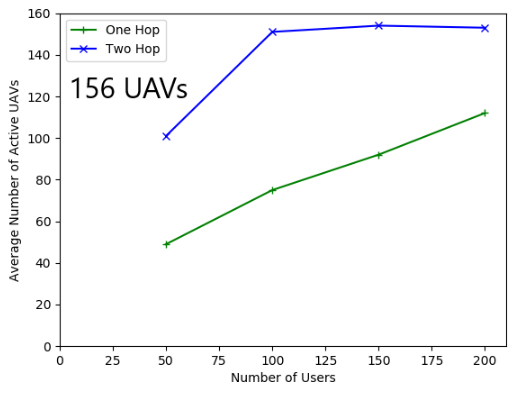
four scenarios. The green and blue bars in Figure 4.7 show $Best_{1h}$ and $Best_{2h}$ performance while magenta shows ATRI performance. The figure shows that GANet network deployment performs better than ATRI across all four training scenarios with two hop GANet providing more significant gains.

The visualizations in Figure 4.8 help explain the differences in performance on the first training scenario. Figure 4.8 (a) shows a network deployed using ATRI with a fitness of 1.60 and we can see that although the UAVs cover the area well, they do not adapt well to the current user distribution. Many users are not covered by any UAV and out of 156 UAVs, only 120 are active, and the rest (36) are inactive. These 120 active UAVs collectively provide a 245.7 Mbps data rate to users. On average therefore, the per UAV data rate is $245.7/120 \approx 2.04$ Mbps. Figure 4.8 (b) shows the network deployed using $Best_{1h}$ with fitness of 1.69 and has 132 active UAVs and 24 inactive UAVs. The figure shows that GANet deployed UAVs more closely follow the user distribution and thus result in more active UAVs and fewer inactive UAVs. These 132 active UAVs provide a data rate of 250 Mbps to users. On average, each active UAV thus provides a data rate of $250/132 \approx 1.89$ Mbps. Finally, Figure 4.8 (c) shows the network deployed using $Best_{2h}$ with a fitness of 1.93. UAVs cluster around users and the number of active UAVs has now increased to 152, so there are only 4 inactive UAVs, and the active UAVs collectively serve a data rate of 282.7 Mbps with each active UAV serving a data rate of $282.7/152 \approx 1.85$ Mbps. These figures provide evidence that GANet with one hop or two hop communication performs better than ATRI on T_{r1}. On this scenario, $Best_{2h}$'s fitness is approximately 14.2% higher than $Best_{1h}$'s fitness and 20.6% higher than ATRI where as $Best_{1h}$'s fitness is 5.6% higher than ATRI. Similar visualizations on the remaining three training scenarios continue to show that GANet more closely adapts to user distributions and thus performs significantly better than ATRI.

To tease out whether bandwidth coverage or longevity is the key factor for the performance difference between 1-hop and 2-hop UAV communication, we plotted the graph of the average bandwidth coverage and the averaged number of active UAVs on training scenarios with 156, 117, 78 UAVs and different numbers of users. Figure 4.9(a) shows average bandwidth coverage for $Best_{1h}$ and $Best_{2h}$



(a) Average bandwidth coverage

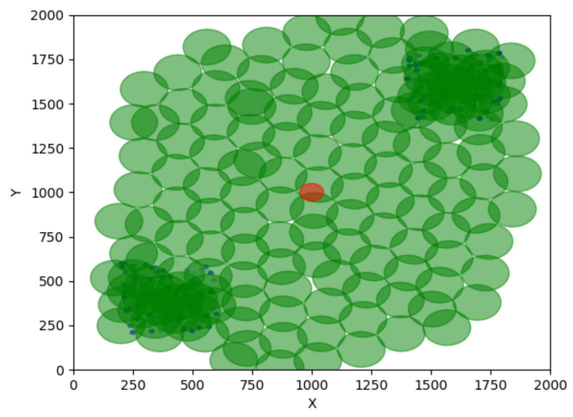


(b) Average number of AUs

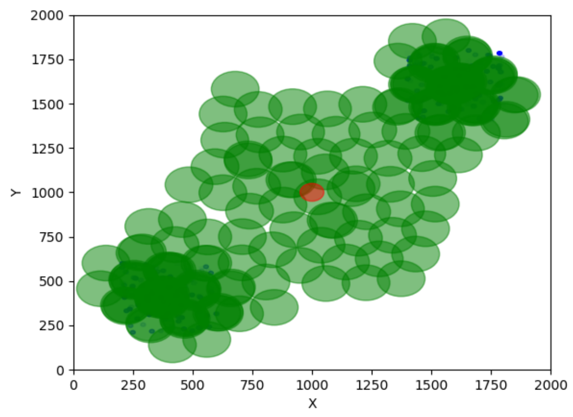
Figure 4.9: Comparing the average (a) bandwidth coverage, and (b) number of AUs of $Best_{1h}$ and $Best_{2h}$ on training scenarios. There is a little difference in the average bandwidth coverage, but a significant difference in the number of AUs.

for $50, 100, 150$, and 200 users on the four scenarios and we see that there is no significant difference in bandwidth coverage. Figure 4.9(b) shows the number of active UAVs. Here we see significantly larger numbers of active UAVs with $Best_{2h}$ compared to $Best_{1h}$ thus providing evidence that the difference in the performance must be caused by the difference in the number of active UAVs. This makes sense as more information from more distant (2-hop) neighbors is used to direct UAVs to share more bandwidth increasing the number active UAVs and thus the network's longevity.

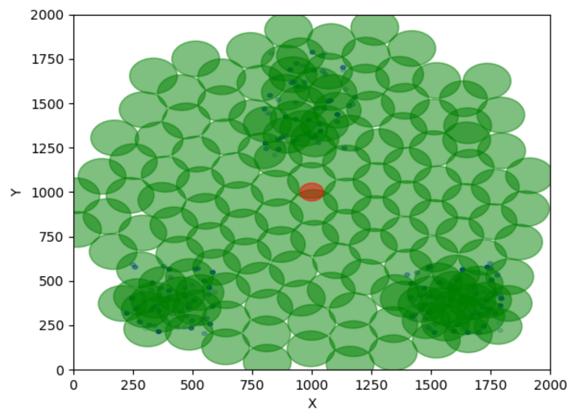
ARTI on the other hand tries to minimize movement away from the Delaunay Triangulated locations and works well with more uniform user distributions. Figure 4.10 shows networks deployed using ATRI and $Best_{1h}$ on our second and third training scenarios from Figure 4.1 (b,c). Tr_2 has two user clusters and ATRI's performance is 1.12 which is 28.2% lower than $Best_{1h}$'s and 42.5% lower than $Best_{2h}$'s performance. Similar behavior can be seen for Tr_3 as well. We can visually see well the differences when comparing the left hand side versus right hand side



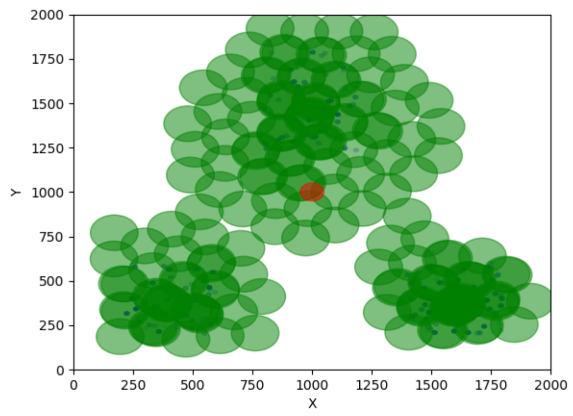
(a) ATRI on T_{r2}



(b) $Best_{1h}$ on T_{r2}



(c) ATRI on T_{r3}



(d) $Best_{1h}$ on T_{r3}

Figure 4.10: Network deployments of $Best_{1h}$ and ATRI.

plots. ATRI UAVs are more uniformly distributed across the AOI while GANet UAVs have learned parameter values that enable them to cluster around users while maintaining connectivity to the command center. On T_{r2} and T_{r3} per AU data rates with ATRI are 4.54 and 3.76 Mbps respectively. Whereas per AU data rate with $Best_{1h}$ are lower at 3.09 and 2.86 Mbps respectively, and with $Best_{2h}$, still lower with 1.99 and 1.88 Mbps respectively. Lower data rates correspond to less energy consumption and longer network lifetimes. Figure 4.11 shows the per AU data on the training scenarios where ATRI per AU data rate is highest and GANet

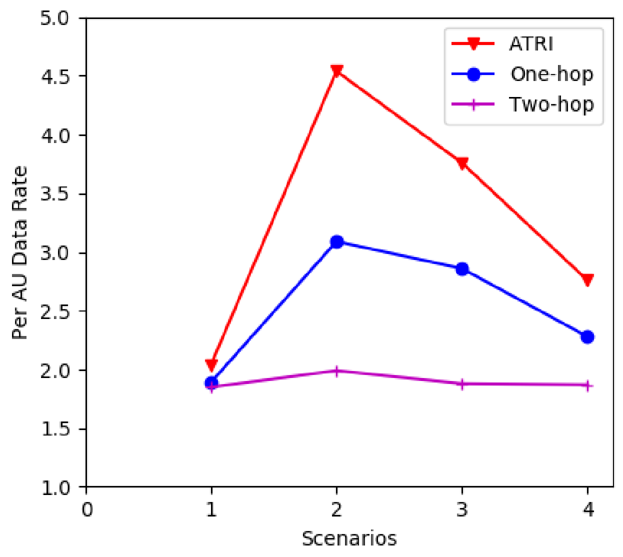


Figure 4.11: Comparing per AU data rate on four training scenarios.

2-hop is lowest. Note also that on the first training scenario where users are distributed more uniformly, ATRI's performance is comparable to GANet.

Note that $Best_{1h}$ and $Best_{2h}$ were evolved with 156 UAVs and 200 users on the four training scenarios but an earlier plot, Figure 4.9 depicted their performance with different numbers of users. This shows that GANet finds parameter values that work on scenarios that have not been used for training and provides early evidence that evolved solutions may be robust. To further investigate robustness, we systematically varied the numbers of UAVs, the numbers of users and computed performance on the four training scenarios. Table 4.3 shows performance obtained using $Best_{1h}$, $Best_{2h}$, and ATRI with $156, 117, 78$ UAVs and $200, 150, 100, 50$ users on the four training scenarios. We thus compare a total of $3 \times 4 \times 4 = 48$ different combination of UAVs, users, and scenarios. Table 4.3 shows that fitness obtained with $Best_{2h}$ is better in all 48 combinations of UAVs, users, and different scenarios compared to both $Best_{1h}$ and ATRI.

Table 4.3: Comparing fitness of $Best_{1h}$, $Best_{2h}$, and ATRI with different number of UAVs and users on training scenarios.

Users	Methods	Tr_1	Tr_2	Tr_3	Tr_4	Average
156 UAVs						
200	*ATRI*	1.60	1.12	1.34	1.47	1.38
200	1 *hop*	1.69	1.56	1.59	1.71	1.64
200	2 *hop*	**1.93**	**1.95**	**1.97**	**1.94**	**1.95**
150	*ATRI*	1.52	1.11	1.29	1.43	1.34
150	1 *hop*	1.58	1.44	1.53	1.62	1.54
150	2 *hop*	**1.97**	**1.96**	**1.95**	**1.93**	**1.96**
100	*ATRI*	1.34	1.13	1.24	1.42	1.28
100	1 *hop*	1.58	1.29	1.39	1.55	1.46
100	2 *hop*	**1.84**	**1.98**	**1.93**	**1.98**	**1.93**
50	*ATRI*	1.18	1.13	1.16	1.20	1.16
50	1 *hop*	1.23	1.20	1.29	1.38	1.27
50	2 *hop*	**1.32**	**1.92**	**1.29**	**1.64**	**1.54**
117 UAVs						
200	*ATRI*	1.50	1.02	1.29	1.40	1.30
200	1 *hop*	1.52	1.53	1.61	1.62	1.57
200	2 *hop*	**1.64**	**1.88**	**1.90**	**1.84**	**1.86**
150	*ATRI*	1.38	1.09	1.28	1.33	1.27
150	1 *hop*	1.52	1.52	1.62	1.71	1.59
150	2 *hop*	**1.69**	**1.88**	**1.80**	**1.91**	**1.82**
100	*ATRI*	1.26	1.40	1.28	1.34	1.25
100	1 *hop*	1.42	1.33	1.48	1.57	1.45
100	2 *hop*	**1.78**	**1.86**	**1.78**	**1.91**	**1.84**
50	*ATRI*	1.05	1.06	1.12	1.06	1.07
50	1 *hop*	1.22	1.27	1.36	1.25	1.27
50	2 *hop*	**1.51**	**1.77**	**1.85**	**1.60**	**1.68**
78 UAVs						
200	*ATRI*	1.32	0.76	1.13	1.26	1.11
200	1 *hop*	1.47	1.25	0.98	1.57	1.30
200	2 *hop*	**1.65**	**1.53**	**1.59**	**1.69**	**1.62**
150	*ATRI*	1.20	0.71	1.14	1.28	1.08
150	1 *hop*	1.36	1.38	0.93	1.60	1.32
150	2 *hop*	**1.63**	**1.66**	**1.64**	**1.66**	**1.65**
100	*ATRI*	1.10	0.81	0.93	1.23	1.02
100	1 *hop*	1.24	1.31	1.50	1.58	1.41
100	2 *hop*	**1.61**	**1.49**	**1.66**	**1.78**	**1.64**
50	*ATRI*	0.81	0.37	0.71	0.84	0.69
50	1 *hop*	1.08	1.34	1.27	1.15	1.20
50	2 *hop*	**1.21**	**1.70**	**1.67**	**1.55**	**1.53**

The last column provides their average performance. $Best_{2h}$ performed statistically significantly better than $Best_{1h}$ with *p-value* less than < 0.00002 while $Best_{1h}$ outperformed ATRI with $p < 0.008$. This provides evidence that even with different numbers of users and UAVs, our approach works well on scenarios with similar user distributions. In the next subsection we provide details on experiments on *testing* scenarios that vary the distributions of users to better investigate robustness.

4.3.3 Experiments on Testing Scenarios

To measure the robustness of $Best_{1h}$ and $Best_{2h}$, we evaluated and compared performance on 100 testing scenarios with different user distributions, three of which are shown in Figure 4.2 and the rest, randomly generated. Table 4.4 compares GANet 1-hop and 2-hop versus ATRI performance with 156, 117, and 78 UAVs and different numbers of users on these testing scenarios. The table is segmented by the number of UAVs. The first two columns list the number of users and the deployment algorithm. The next three columns provide performance on the test scenarios from Figure 4.2 since we can see the non-uniform user distributions for these scenarios in the figure. The last column provides the average performance over **all** 100 test scenarios. The table shows that $Best_{2h}$ performed better on 95% (34 of 36) combinations of UAVs, users, and three testing scenarios, compared to $Best_{1h}$ and in all 36 compared to ATRI. These performance differences are also statistically significant with $p < 0.008$ for GANet 1-hop versus ATRI and $p < 0.00002$ for GANet 2-hop versus 1-hop. These results provide evidence of the robustness of GANet deployment.

Table 4.4: Comparing fitnesses of $Best_{1h}$, $Best_{2h}$, and ATRI with 156, 117, and 78 UAVs and different numbers of users on testing scenarios.

Users	**Methods**	$\boldsymbol{Te_1}$	$\boldsymbol{Te_2}$	$\boldsymbol{Te_3}$	**Average**
156 UAVs					
200	*ATRI*	1.54	1.47	1.43	1.6
200	1 *hop*	1.69	1.71	1.64	1.69
200	2 *hop*	**1.97**	**1.97**	**1.94**	**1.96**
150	*ATRI*	1.47	1.33	1.38	1.51
150	1 *hop*	1.61	1.69	1.54	1.63
150	2 *hop*	**1.98**	**1.99**	**1.97**	**1.97**
100	*ATRI*	1.41	1.31	1.31	1.39
100	1 *hop*	1.53	1.56	1.48	1.52
100	2 *hop*	**1.99**	**1.76**	**1.98**	**1.92**
50	*ATRI*	1.28	1.20	1.26	1.16
50	1 *hop*	1.33	1.35	1.39	1.33
50	2 *hop*	**1.96**	**1.87**	**1.91**	**1.59**
117 UAVs					
200	*ATRI*	1.69	1.37	1.57	1.48
200	1 *hop*	1.80	1.72	1.63	1.56
200	2 *hop*	**1.97**	**1.83**	**1.92**	**1.84**
150	*ATRI*	1.58	1.31	1.47	1.33
150	1 *hop*	1.73	1.71	1.66	1.49
150	2 *hop*	**1.98**	**1.84**	**1.96**	**1.85**
100	*ATRI*	1.50	1.09	1.40	1.23
100	1 *hop*	1.67	1.60	1.58	1.4
100	2 *hop*	**2**	**1.84**	**2**	**1.77**
50	*ATRI*	1.27	1.12	1.31	1.09
50	1 *hop*	1.47	1.34	1.43	1.3
50	2 *hop*	**1.59**	**1.56**	**1.97**	**1.52**
78 UAVs					
200	*ATRI*	1.48	1.28	1.32	1.19
200	1 *hop*	1.74	**1.67**	1.75	1.35
200	2 *hop*	**1.91**	1.65	**1.83**	**1.59**
150	*ATRI*	1.48	1.22	1.34	1.16
150	1 *hop*	1.83	**1.67**	1.72	1.30
150	2 *hop*	**1.9**	1.64	**1.86**	**1.58**
100	*ATRI*	1.50	1.05	1.40	1.1
100	1 *hop*	1.71	1.55	1.72	1.17
100	2 *hop*	**1.91**	**1.59**	**1.91**	**1.53**
50	*ATRI*	1.32	1.02	1.17	0.78
50	1 *hop*	1.62	1.32	1.56	1.01
50	2 *hop*	**1.84**	**1.36**	**1.78**	**1.37**

Once GANet optimizes potential field values for a set of UAVs as specified by the parameters in Table 4.1, on a set of training scenarios, the results indicate that we should expect similar network performance on new unseen scenarios with different numbers of users, UAVs, and user distributions.

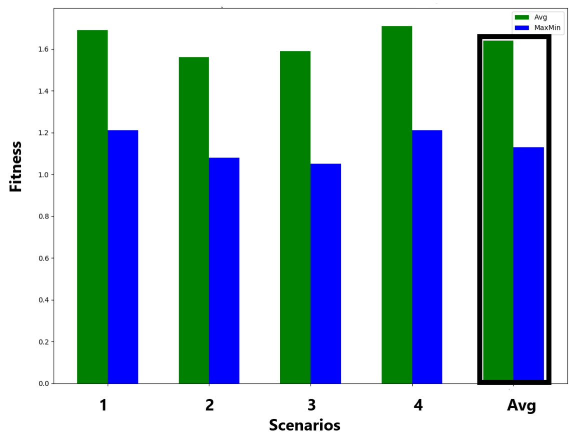


Figure 4.12: GANet performance when maximizing the average performance across four training scenarios (green) compared to maximizing the minimum performance (blue).

4.3.4 Maximizing The Minimum Objective

In all previous experiments, we want to maximize fitness computed as the *average* performance (Equation 4.1) over four training scenarios. A common alternative is to maximize the *minimum* performance [100]. In experiments, comparing GANet one-hop performance with maximizing the minimum (maximin) versus maximizing the average, we found that maximizing the average performs significantly better. Figure 4.12 shows that the best solution evolved using averaged fitness, green bar, performed better than the best solution evolved with maximin fitness, blue bar, on all four scenarios. On average, averaged fitness performed 45.13% better than maximin fitness. We conjecture that averaging performance better maintains diversity in small population (20) genetic algorithms and thus leads to better performance.

4.4 Conclusions

This thesis uses genetic algorithm optimized potential fields to optimize UAV-based ad-hoc wireless network deployment. Unlike prior work, potential fields serve to unify representation across both search and service phases of network deployment and GANet optimizes potential field parameters that work robustly for different user distributions. We showed how to derive potential field parameters for the search phase and designed an elitist genetic algorithm to optimize these non-linear potential field parameters for the service phase. We then formulated the service phase network deployment problem as an optimization problem that maximizes the sum of bandwidth coverage and longevity of the deployed network. Performance is computed from running a simulation of UAVs moving under the influence of GANet optimized potential fields and provides a fitness metric for the genetic algorithm's search for optimal potential field parameter values. We showed that GANet is significantly better than gradient ascent and exhaustive search by finding a global optimum on a tractable problem more than twice as quickly as the nearest competitor. We then attacked larger, more realistic problems using four training scenarios with 156 UAVs and 200 users distributed uniformly and non-uniformly over an area of interest.

Results showed that GANet network deployment is significantly better than the state of the art ATRI deployment and that GANet with two-hop UAV communications beats one-hop UAV performance. These results carry over to 100 test scenarios with never before seen user distributions, numbers, and to different numbers of UAVs. This provides evidence for GANet's robustness and our potential fields based representation. GANet adapts better when users are non uniformly distributed and matches or exceeds ATRI performance with more uniform user distributions. Since with sufficient numbers of UAVs we get to optimal or near-optimal performance with two-hop UAV communication, we believe that going to higher hop counts for communications will not significantly improve performance.

CHAPTER 5
DYNAMIC FIRE COVERAGE

We choose wildfire boundary tracking using UAVs as the second domain specific problem and control the movement of UAVs using the same potential field based approach. Note that we are interested in tracking fire *boundaries* and not the entire area covered by a fire and we therefore wish to position UAVs along fire boundaries. We start with the problem formulation and then describe the fire simulation model, fire boundary extraction. Next, we identify the set of potential fields that control the movement of UAVs and lastly discuss experimental results.

5.1 Problem Formulation

We aim to deploy a set of UAVs to track wildfire spread in unknown environments. Let us assume that we have a set of n homogeneous UAVs $u = \{u_1, u_2...., u_n\}$ and a rectangular Area of Interest (AOI). We assume that the fire is non-uniformly distributed over the area of interest and that each UAV is equipped with GPS devices and downward-facing cameras capable of detecting fire. Each camera has a circular Field of View (FOV) and can capture the area under its FOV. Images captured by each UAV can be sent back to a Command Center (CC) for analysis so we can locate the areas of fire. For experimentation, we simulate a fire model called Fire Area Simulator (FARSITE) [64] and used different FARSITE parameters to get different behaviors of fire spread. Our objective is to track fire spread for as long as possible, and we thus use minimizing energy consumption as an optimization objective in our multi objective optimization problem formulation.

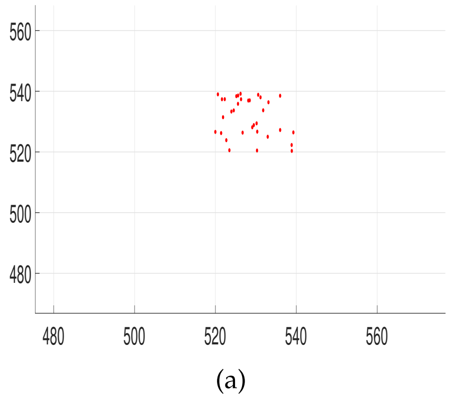
Let us assume that the k^{th} UAV observes I_k fire intensity under the UAV and has E_k remaining energy at the end of a simulation. Since we want to maximize both fire boundary coverage and the amount of time we maintain coverage, the problem can be formulated as a two-objective optimization problem described by Equation 5.1.

$$\text{Maximize} f = \left[\frac{\sum_{k=1}^{n} I_k}{I_{max}}, \frac{\sum_{k=1}^{n} E_k}{E_{max}} \right] \tag{5.1}$$

Here I_{max} is maximum intensity at the boundary and E_{max} is the sum of energy over all UAVs at the start of simulation. n is the number of UAVs and dividing by I_{max} and E_{max} normalizes objective values to lie between 0 and 1. We use a set of potential fields to govern the movement of UAVs to track a fire's boundary generated using fire simulator. In the next section we describe the fire simulator.

5.2 Fire Simulator

We use the Fire Area Simulator, FARSITE, model [101], a well-known core model of existing fire simulation systems. Apart from FARSITE, other fire simulation models [102] can be used to simulate fire propagation but FARSITE seems the most popular, reliable model and is widely used by federal land management agencies such as the US Department of Agriculture [101]. FARSITE estimates fire fronts based on an elliptical model introduced by Richards [103] and requires and uses information on terrain, geography, topography, fuel, and weather to model the spatial distribution and spread of a fire. Since accurate fire simulation is not in the scope of this work, we use simplified fire propagation in FARSITE to calculate fire front growth [64].



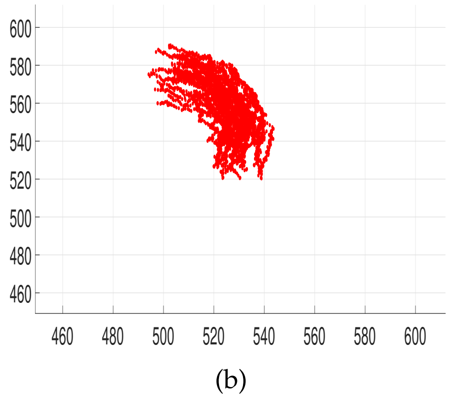


Figure 5.1: A discretized fire influence map. (a) 33 initial fire heat sources at $t = 0.$, and (b) Propagation of fire from initial 33 heat sources at time step $t = 100$.

We assume that fire initially starts with 33 heat sources on the xz plane, $H = \{(x_1, z_1), (x_2, z_2), ..., (x_{33}, z_{33})\}$ where x_1, z_1 are the x and z coordinates of the 1^{st} heat source location. Figure 5.1(a) shows the initial heat source locations on a grid with axes units in meters. Each heat source propagates in the direction of the wind using Equation 5.2 where x_t and z_t are the differentials of the fire on the xz plane and θ, $(0 \leq \theta \leq 2\pi)$, is the direction of the wind. c is the distance from the heat source to the center of the ellipse which can be empirically calculated [101].

$$\begin{aligned} x_t &= c * \cos(\theta), z_t = c * \sin(\theta) \\ c &= \frac{R - \frac{R}{HB}}{2} \\ HB &= \frac{(LB + (LB^2 - 1)^{0.5})}{(LB - (LB^2 - 1)^{0.5})} \\ LB &= 0.936e^{0.2566U} + 0.461e^{-0.1548U} - 0.397 \end{aligned} \tag{5.2}$$

Equation 5.2 computes c which depends on two variables, R, the rate of fire spreading and U, the wind speed. For experiments, we used $25m/s$ for the rate of fire spread, and $5m/s$ for wind speed. The intensity of the fire point location is initially set to random values of a normal distribution ranging from $2-10\ kW/m$ which remains constant throughout the propagation of fire. We compute the location of the new fire front after every Δt and Equation 5.3 computes the next propagation location of the i^{th} heat source.

$$\begin{aligned} x_i(t+\Delta t) &= x_i(t) + \Delta t X_t(t) \\ z_i(t+\Delta t) &= z_i(t) + \Delta t Z_t(t) \end{aligned} \tag{5.3}$$

We ran FARSITE for 6000 time steps for one evaluation. Since we are interested in covering the fire boundary, not in locations covered by fire, we extract the boundary using a fire influence map.

Algorithm 3: Fire Influence Map

```
Input  : Initial heat source locations $H = \{(x_1, z_1), (x_2, z_2), ..., (x_h, z_h)\}$
Output: Grid
for $t$ in maxtimeSteps do
    for $h$ in $H$ do
        $x_h(t+\Delta t) = x_h(t) + \Delta t x_h(t)$
        $z_h(t+\Delta t) = z_h(t) + \Delta t z_h(t)$
        Intensity$_h$ = random(2,10)
        for $i$ in Grid$_x$ do
            for $j$ in Grid$_z$ do
                D = Distance(Grid[$i$][$j$], $(x_h(t), z_h(t))$)
                if $D == 0$ then
                    Grid[i][j] += Intensity$_h$
                end
            end
        end
        H.append($(x_h(t+\Delta t), z_h(t+\Delta t))$)
    end
end
return(Grid)
```

5.3 Fire Influence Map

A Fire Influence Map (FIM) is a grid defining fire spatial information in a given AOI with values assigned to each grid-cell based on fire intensity. Algorithm 3 computes these values for each grid cell. Algorithm 3 tracks the fire starting with initial heat source locations and runs for 6000 time-steps. At each time step, we calculate fire intensities and the spread of the fire using Equation 5.3 Figure 5.1(a) shows the initial 33 heat sources and Figure 5.1(b) shows the fire spread at time step 100. We extract the fire boundary using grid values at the end of each time step, and a team of UAVs maneuver to cover this extracted boundary. The boundary is a set of tuples $B = \{(x_1, z_1, i_1), (x_2, z_2, i_2), ..., (x_k, z_k, i_k)\}$ where x_k, z_k are the x and z coordinates of k^{th} fire location with fire intensity of i_k on the boundary.

In the literature many boundary extraction techniques such as square tracking, moore-neighbor tracing, and radial sweep [104, 105]. For simplicity, we implemented our own version of radial sweep to extract fire fronts. We scan the FIM grid, first, from left to right, and second, from top to bottom, storing the first and last grid locations with positive fire intensity values as the boundary. This gives us the fire boundary that UAVs need to track. In the next section we describe how we deploy UAVs using potential fields.

5.4 UAV Deployment

We use a CPT based algorithm for initial positioning of UAVs at positions given by the CPT and at a starting elevation of 100 meters [36]. Althought there are other techniques such as DT [40] and Voronoi Diagram [35] that can be used to deploy UAVs in the first phase, we use CPT because it is a fast, simple technique and ensures no overlapping between UAVs' coverage radii while maximizing area covered. During this first phase, each UAV identifies locations on fire within its sensor range and communicates this information to neighbors. Only in the second phase do UAVs start moving under the influence of different attractive and repulsive potential fields.

5.4.1 Potential Field and Representation

We augment distance based potential fields with potential fields based on fire intensity to control the movement of UAVs. Equation 5.4 shows the resultant potential fields equation comprised of three distance dependent potential fields and two fire intensity dependent potential fields that govern the heading of the k^{th} UAV. We explain each term in the Equation 5.4 shown below.

$$\vec{P}_k = \vec{P}_{nd} + \vec{P}_{nI} + \vec{P}_{fb} + \vec{P}_{fI} + \vec{P}_{alt} \tag{5.4}$$

$\vec{P}_{nd}$ denotes potential fields based on inter UAV distance so that UAVs do not cluster too close to each other, $\vec{P}_{nI}$ based on neighbors fire coverage. $\vec{P}_{fb}$ denotes one attractive and one repulsive potential field dependent on the distance from the k^{th} UAV to the nearest fire boundary location so UAVs are attracted towards the boundary but do not get too close to the fire itself. $\vec{P}_{fI}$ similarly defines the two potential fields associated with fire intensity and also helps influence coverage. Finally, PF^k_{alt} mediates altitude and coverage since height correlates with energy usage and with coverage FOV.

Each term in Equation 5.4 denotes two potential fields with each potential field having two parameters. The full set of equations with all potential fields and their parameters are show in in Equations 5.5 to 5.9 respectively.

$$\vec{P}_{nd} = \sum_{j \in p} (c_1 \vec{d}_{kj}^{e_1} - c_2 \vec{d}_{kj}^{e_2}) \tag{5.5}$$

$$\vec{P}_{nI} = \sum_{j \in p} (c_3 \vec{n} I_{kj}^{e_3} - c_4 \vec{n} I_{kj}^{e_4}) \tag{5.6}$$

$$\vec{P}_{fb} = \sum_{j \in m} (c_5 \vec{d}_{kj}^{e_5} - c_6 \vec{d}_{kj}^{e_6}) \tag{5.7}$$

$$\vec{P}_{fI} = \sum_{j \in m} (c_7 \vec{f} I_{kj}^{e_7} - c_8 \vec{f} I_{kj}^{e_8}) \tag{5.8}$$

$$\vec{P}_{alt} = c_9 \vec{d}_{k}^{e_9} - c_{10} \vec{d}_{k}^{e_{10}} \tag{5.9}$$

Equation 5.5 shows the attractive and repulsive potential fields based on distance between the k^{th} UAV and its p neighbors within air-to-air communication range. Equation 5.6 computes potential fields similar to equation 5.5 except potential fields are dependent on neighbors fire intensity coverage. Equation 5.7 and 5.8 compute attractive and repulsive potential fields based on distance and fire intensity of m nearest fire points on the boundary of the fire. The coverage radius of UAV is directly proportional to its altitude and thus with more altitude, the coverage radius and area increase. However, a UAV expends more energy to change altitude compared to moving at constant altitude or hovering in place. Thus we generate an attractive and a repulsive potential fields using Equation 5.9 to control the altitude of UAVs and find a good trade-off between altitude and energy consumption.

Since each potential field has two parameters, a coefficient (c) and an exponent (e) to tune, we thus have 20 total parameters needing to be tuned. We encode these parameters in a real-value chromosomes with upper and lower bounds of coefficients $c \in [-500000, 500000]$ and exponents $e \in [-2, 2]$. These values were experimentally determined to work well and led to high fitness solutions.

Algorithm 4: Fitness Computation

```
Input  : B, CS, MS, MT, UAVs
Output: fitness
$obj_1 = obj_2 = 0$;
for scenario in MS do
    timeSteps = 0;
    while timeSteps<MT do
        $AssociateBoundary(B, UAVs)$;
        $FindNeighbors(UAVs)$;
        $Headings = ComputePotentialFields(CS, B, UAVs)$;
        $MoveAll(Headings)$;
        timeSteps++;
    end
    $obj_1$ += $FireCoverage()$;
    $obj_2$ += $RemainingEnergy()$;
end
$obj_1 = obj_1$ / MS;
$obj_2 = obj_2$ / MS;
fitness = $[obj_1, obj_2]$;
return(fitness);
```

5.5 Fitness Computation

The fitness of each chromosome or candidate solution is evaluated in three steps. First, we send potential field parameters to our simulator running one fire scenario. We then load these parameters to the UAV controller to control UAV movement and run the simulation for 6000 time steps. When finished, we compute the two objective values, boundary coverage and energy consumption, for this scenario. We have three training scenarios so we repeat these steps for the remaining scenarios and return the average objective values over these three scenarios as the candidate solution's evaluation. Algorithm 4 specifies this fitness computation.

Note that prior work indicates that solutions evolved on just one scenario may not produce similar results on other unseen scenarios [7, 10]. To evolve more robust parameter values, we used three different scenarios to evaluate each candidate solution and averaged results.

Algorithm 4 takes fire boundary (B), a candidate solution (CS), the number of scenarios (MS), a number of time steps (MT), and a set of UAVs, and returns the averaged fitness assigned to the CS. `AssociateBoundary` allocate a region of fire boundary to the closest UAV based on distance and `FindNeighbors` assigns neighboring UAVs based on air-to-air communication range. Using this boundary and neighbor information, `ComputePotentialFields` computes the vector sum of potential fields acting on this UAV. These *vector sum directions* are used by `MoveAll` to move each in the direction indicated by the vector sum of potential fields acting on the UAV. At the end of the 6000 time steps for each scenario, two objective values are computed using Equations 5.1. The first objective, Fire Coverage (FC) is the sum of the fire intensity covered by the UAVs divided by the sum of the total fire intensity at the fire boundary at the end of the simulation. We do this to normalize the objective value to be between zero to one. If there is an overlap of coverage by two or more UAVs where they cover the same segment of boundary, the intensity will only be counted towards this objective once. The second objective, Remaining Energy (RE) is the sum of the remaining battery life of the UAVs divided by the total number of UAVs. UAV battery life ranges between zero and one with one representing a full charge and zero representing an empty battery. The two objective values are averaged over the three scenarios and assigned as fitness. Table 5.1 provides UAV, the genetic algorithm, and fire simulation parameters.

Table 5.1: Simulation Parameters.

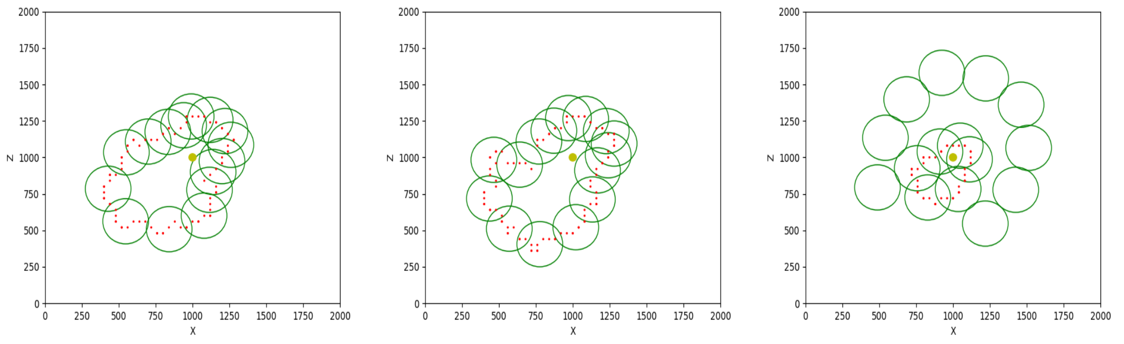
	Parameters	**Symbol**	**Value**
UAV	Area of Interest	AOI	$2000m^2$
	Coverage Radius	U_{gr}	h
	A-2-A Range	U_{ar}	300m
	Max Speed	U_s	5 m/s
	Initial energy	E_t	10^6 J
	Hovering Energy	E_h	138.24 J/t
	Altitude Energy	E_h	164.16 J/t
	Linear Energy	E_h	109.44 J/t
NSGA-II	Population size	Pop_m	20
	Max generation	Gen_m	20
	Crossover rate	P_x	0.95
	Mutation rate	P_m	0.05
FARSITE	Wind Direction	θ	$0 \leq \theta \leq 2\pi$
	Wind Speed	U	$5\ m/sec$
	Fire Spread Rate	R	$25\ m/min$
	Intensity Values	I_{jt}	$(1-10)\ kW/m$

5.6 Results and Discussion

We consider an area of interest of 2000 meters^2 and assume an initial 33 heat sources from which the fire starts and spreads. Each heat source starts near location (1000, 0, 1000) in the xz plane. We used 15 UAVs with a circular field of view. The coverage radius of each UAV is directly proportional to its altitude where minimum and maximum altitudes are restricted to be between 100 to 150 meters. Initially each UAVs starts at an altitude of 100 meters.

5.6.1 Experiments on Training Scenarios

As mentioned earlier, solutions evolved on only one scenario may not produce good results on other unseen scenarios. In other words, evolved solutions may not be robust. Thus we evaluate each candidate solution on three different fire scenarios as shown in Figure 5.2. We obtained these different fire scenarios by changing variables in our FARSITE fire model. At the end of evaluation, objective values averaged over these three training scenarios are assigned as two objective values of a given candidate solution. The NSGA-II evolves the population and quickly produces good potential field parameter values. The Figures 5.2, and 5.3 show UAV positions at the end of the simulation for one of the more balanced solutions on the pareto front. Specifically, we picked a solution from the last generation pareto front with objective values of $(1, 0.71)$ and Figure 5.2 shows the locations of the the fire boundary and the locations of UAVs at the end of simulation for the three training scenarios. Figure 5.2 shows that UAVs are able to cover 100% of the fire boundary in all three training scenarios, with 77.6%, 69.9%, 66.8% energy remaining respectively. Circles indicate coverage and the overlapping circles show that we have more than enough UAVs to cover the entire boundary within our altitude (and energy) constraints. Even with a relatively small population of size 20, the NSGA is able to quickly evolve good solutions.

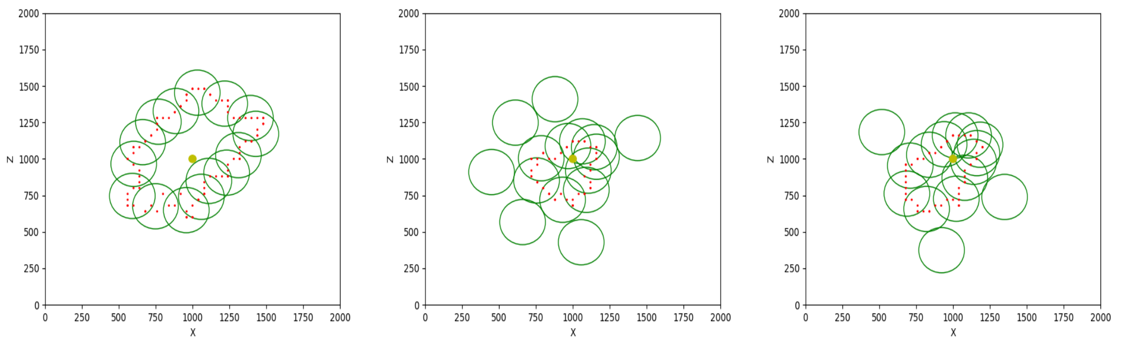


(a) Tr_1 (b) Tr_2 (c) Tr_3

Figure 5.2: Fire tracking over three training scenarios where red dots represent the fire boundary in a given AOI. Green circles represent the coverage range of UAVs. In all three cases, 100% fire boundary coverage is achieved.

5.6.2 Experiments on Testing Scenarios

In order to evaluate the robustness of evolved solutions, we picked the same solution with objective values of $(1, 0.71)$ from the last generation pareto front and ran this solution on three unseen fire scenarios as shown in Figure 5.3. On the first testing scenario Te_1, at the end of 6000 time steps simulation we achieve 100% percentage fire coverage at the boundary with 79.7% energy still remaining. Similar running on the second and third testing scenarios achieved 100% fire boundary coverage and 77.5%, 79.7% remaining energy respectively. On average we achieved 78.1% remaining energy. These preliminary results indicate the viability of our potential fields based approach towards a decentralized control of UAVs performing a fire tracking task. However, much work remains in scaling our results and in establishing a firm theoretical foundation to provide performance and reliability limits.



(a) Te_1 (b) Te_2 (c) Te_3

Figure 5.3: Fire tracking over three testing scenarios. In all three cases, 100% fire boundary coverage is achieved.

5.6.3 Performance With Reduced Number of UAVs

We started with scaling by considering fewer UAVS since given enough UAVs, it is not surprising that we are able to achieve 100% coverage with significant energy to spare. We reduced the total number of available UAVs by 50% ($7.5 \approx 8$) and using the same parameters from earlier $(1, 0.71)$ evaluated performance on all six training and testing scenarios. On average we get only about 88% fire coverage, 76.3% remaining energy, and Figure 5.4 shows that we still have some overlap. We need to better tune potential fields to minimize overlap and we may need to modify altitude limits. Evolving on training scenarios with different numbers of UAVs and explicitly including overlap in the objective function seem promising avenues for future research.

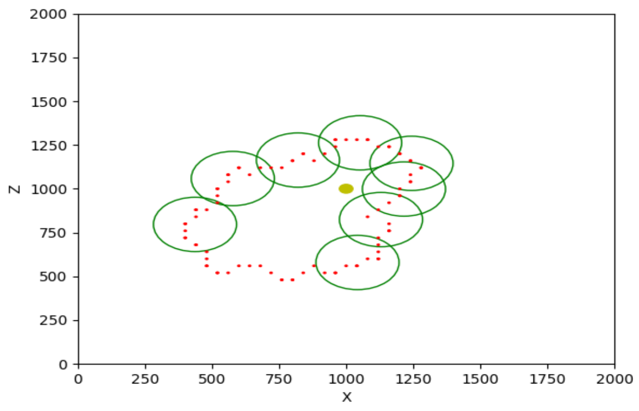


Figure 5.4: Fire tracking with 8 UAVs.

5.7 Conclusion

We again used a potential fields based representation and a multi objective problem formulation to control UAV movement for dynamic fire boundary tracking.

We formulated this problem as a multi-objective optimization problem that maximizes fire coverage while minimizing energy consumption. Using the same representation as in the case of WMND but with a different set of potential fields that govern the movement of UAVs, we optimized parameter values to achieve objectives. Preliminary results indicated that potential fields based solutions show significant promise on this task and that a genetic algorithm evolves robust solutions that degrade gracefully in unseen environments. When trained with 15 UAVs, the NSGA-II evolved solutions with 100% fire boundary coverage even on unseen scenarios but the same parameter values lead to lower coverage when using 8 UAVs. More specifically, the evolved solution performance on different training and testing scenarios show that given enough UAVs (15), we can get 100% fire boundary coverage while having 74.5% remaining energy at the end of our simulation runs. When we reduce the number of UAVs by half we still achieved an average of 88% boundary coverage with 76.3% energy reserves.

CHAPTER 6
REAL-TIME STRATEGY GAME MICRO

We start by describing what is micro in RTS games. Good micro helps to win skirmishes and even the game, but searching for good micro behaviors is a difficult problem because of large search space. We use a set of potential fields to guide game agents during skirmishes and will specify how we use potential fields. Our aim is to guide a team of heterogeneous game agents such that during skirmishes, agents receive less damage and inflict more damage to opponents. As mentioned earlier, solutions evolved on one scenario may not work well on other scenarios, thus we needed different scenarios to evolve micro and different scenarios to test the robustness of evolved micro behaviors. We formulated this problem as a multi objective optimization problem to maximize damage done to opponents and to minimize damage received by friendly agents. We compared the performance of two other techniques to generate micro in RTS games with our representation.

6.1 Problem Formulation

We evolved micro behaviors for a group of agents composed from 2 to 5 different types of game agents against an identical group of agents on two different RTS games simulation platforms. We use the same approach and representation as in the wireless mesh network deployment problem and fire boundary coverage. We identify a set of potential fields to control game agents.

Our objectives are maximizing damage done to the opponent and minimizing the damage received by friendly agents for a group composed of heterogeneous agents. Some of the earlier work, combined both objectives into one objective and used a genetic algorithm to maximize the single objective problem. We keep the objectives separated and use NSGA-II to evolve a diverse pareto front of micro behavior. We normalize damage done and damage received to span the range [0..1], and Equation 6.1 describes our multi objective optimization problem.

$$\begin{aligned} Max\sum_{i=0}^{e} D_{ei} &= Max\sum_{i=0}^{e} \frac{MaxHealth - EH_i}{MaxHealth} \\ Max\sum_{j=0}^{f}(1 - D_{fj}) &= Max\sum_{j=0}^{f} \frac{FH_j}{MaxHealth} \end{aligned} \tag{6.1}$$

Here, D_e represents damage done to enemy agents and D_f represents the damage to friendly agents. Our objectives are to maximize D_e and minimize D_f respectively. $MaxHealth$ is the sum total of the undamaged health of all agents on a side, e and f represent the number of enemy and friendly agents alive at the end of a skirmish. D_{ei} and D_{fj} represents damage done to the i^{th} enemy agent and damage received by the j^{th} friendly agent respectively. EH_i and FH_j represent the remaining health of the i^{th} enemy agent and the j^{th} friendly agent. The normalization against maximum health ensures that damage done and damage received stay within 0 and 1. To evaluate an individual in the NSGAII population, we play the two groups on the different training scenarios and average the damage done and damage received on these scenarios. This normalized two-objective fitness function used within our NSGA-II implementation then produces the results described later in this chapter. To make decisions in RTS games, spatial information of opponent agents is necessary and thus we use Influence Maps (IMs) to gather opponents information and to find a target location to attack.

6.2 Influence Map

Spatial information in RTS games plays a significant role in decision making. IMs used for enemy and terrain spatial information. An IM is a grid virtually placed over a game map, where each position in the game map is represented by a value. Algorithm 5 calculates the grid cell values. Algorithm 5 takes 5 input arguments; weight w_1, w_2, a constant w_3, decay fraction I_f, effect radius r and returns the grid cell values G for entire map. For each agent, Algorithm 5 calculates a starting IM (IM_s) for each agent using `StartingIM` at the agent's location depending on the agent's health and weapons cool-down state. The IM_s of an agent reduces linearly by a fraction I_f to zero as range r increases. `UpdateGrid` calculates grid-cell values by adding the influence of all agents within a range (r) from the cell using Equation 6.2.

$$\begin{aligned} IM_s &= w_1 P_h + w_2 P_w + w_3 \\ I_d &= IM_s * I_f \\ G_c &= \textstyle\sum_{i \in r}(IM_s - d_i I_d) \end{aligned} \tag{6.2}$$

In Equation 6.2 P_h, P_w are health and weapon cooldown time of a game agent, and

Algorithm 5: Influence Map

```
Input  : $w_1, w_2, w_3, I_f, r$
Output: $G$
1 $G$ = InitializeGrid();
2 for Agent in Agents do
3     $IM_s$= StartingIM($w_1, w_2, w_3$, Agent);
4     UpdateGrid( $IM_s, I_f, G, r$, Agent);
5 end
6 return($G$);
```

w_1, w_2, and w_3 are weights. I_f and r are fraction and range of the influence. We choose to evolve w_1, w_2, w_3, I_f, and r to calculate the influence of each agent. Low cell values represent locations with low health enemy agents indicating easy prey and we choose the lowest cell as the target coordinate. Once the target selected a set of potential fields guide agents towards the target location. In order to complete any given task, agents need to cooperate based on their unique ability and different types of agents have different roles to play during skirmishes. To fully utilize the effect of different types of agents, we consider that each different type of agent has a different perspective of the game world. Therefore, we evolve different sets of IM parameters for each different types of agents, for one type of agents 5 IM parameters and for n different types of agents $5n$ IM parameters.

6.3 Potential Fields for Real-Time Strategy Game Micro

We use potential fields to guide agent movement to the target location provided by the IM. Once near the opponent, we would like our agents to maneuver well based on the location of enemy agents, their health, and the state of their weapons. We define potential field in the form of cd^e where c and e are evolvable parameters and d can be distance, health, or weapons state. We thus define and use attractive and repulsive potential fields for each of these factors. Since the fields for friendly agents should be different from the fields for enemy agents, we use two such sets of potential fields. Finally, the target location also exerts an attractive potential. Equation 6.3 shows addition of 4 potential fields based on distance $\vec{P}_d$, health $\vec{P}_h$, weapon state $\vec{P}_w$, and target $\vec{P}_t$. Ignoring the target location's potential field, this results in a total of 2 (attractive, repulsive) $\times 3$ (location, health, weapons state) $\times 2$ (friend, enemy) = 12 potential fields for guiding one type of agent's movement against an enemy also composed of only one type of agent. Equation 6.4 shows 4 potential fields each adding to total 12 potential fields. We use the same techniques from [106] to convert the vector sum of these potential fields into a desired heading and desired speed. Once we move to micro for groups composed of two types of agents, the number of potentials fields increases.

$$\vec{P} = \vec{P}_d + \vec{P}_h + \vec{P}_w + \vec{P}_t \tag{6.3}$$

$$\begin{aligned}
\vec{P}_d &= \sum_{i\in F}(c_1 d_i^{e_1} + c_2 d_i^{e_2}) + \sum_{i\in E}(c_3 d_i^{e_3} + c_4 d_i^{e_4}) \\
\vec{P}_h &= \sum_{i\in F}(c_5 h_i^{e_5} + c_6 h_i^{e_6}) + \sum_{i\in E}(c_7 h_i^{e_7} + c_8 h_i^{e_8}) \\
\vec{P}_w &= \sum_{i\in F}(c_9 w_i^{e_9} + c_{10} w_i^{e_{10}}) + \sum_{i\in E}(c_{11} w_i^{e_{11}} + c_{12} w_i^{e_{12}})
\end{aligned} \tag{6.4}$$

We need different potential fields for different types of agents because each type of agent treats other types of agents differently. For example, a friendly melee agent treats enemy melee agents differently from enemy ranged agents. In RTS games, a melee agent has low firing range but has more health, and a ranged agent has large firing range but comparatively low health. The friendly melee agent should avoid enemy ranged agents and approach enemy melee agents. In contrast, a friendly ranged agent can target any enemy agent during skirmishes. Thus we need different potential fields and IM parameters for each type of agent.

Figure 6.1 shows the four sets of potential fields needed when dealing with groups composed from two types of agents where F_1 represents type one friendly agents and F_2 represents type two friendly agents. For now, we ignore potential fields generated by the target location. Similarly E_1 represents type one enemy agents and E_2 represents type two enemy agents. A total of 4 (two attractive and two repulsive) fields are required corresponding to F_1, F_2, E_1, and E_2 for each of the following properties: distance, health and weapon state. This results in a total of 4 (attractive, repulsive) $\times 3$ (distance, health, weapons state) $\times 2$ (friend, enemy) $= 24$ potential fields per agent type.

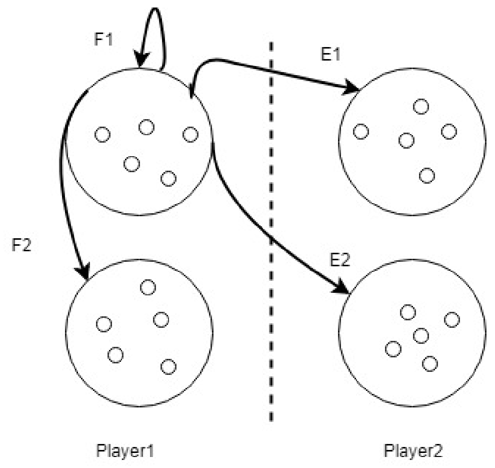


Figure 6.1: Potential fields needed for groups composed from two types of agents.

The above explanation calculated the number potential fields required for one type of agent. If p represents the number of potential fields, then for one type versus one type $p = 12$. For two types versus two types of agents, recall that we needed 24 potential fields of each type of agent. Thus we will need 2 (types of agents) $\times 24 = 48$ potential fields required. Generalizing if we consider skirmishes between n types of agents versus n types of agents, then the number of potential fields required per type of agent is $n \times p$ and the total number of potential required is $n \times n \times p$. Each potential field has two parameters, thus in total we need $2 \times n \times n \times p$ parameters.

We now consider the potential field exerted by the target location. Each type of agent has its own target location, and hence requires a fixed number of parameter for its target potential field (2 parameter) and influence map (5 parameter). Let $q = 7$ be this fixed number of parameters. q grows linearly with n, thus $q \times n$ parameters are required for n different types of agents on each side. We can see that Equation 6.5 gives the total number of parameters required to deal with n different types of agents to a side where *Pnum* refers to total number of parameters. Note that the distance potential field parameters are computed more than once, as different types of agents are added to each side. We can therefore subtract the additional potential field parameters from the total number of required parameters. Thus for two types of agents on each side, we need a total of 106 parameters. In real game play, we usually micro with four or five different types of agents resulting in 388 or 595 parameters according to Equation 6.5. This seems feasible to evolve with our micro representation.

$$Pnum = (q + 2pn)n - \sum_{i \in n} 4(i - 1) \tag{6.5}$$

These parameters provide a target location and guide agent movement towards the target. If enemy agents come within weapons range of a friendly agent, the friendly agent targets the nearest enemy agent. In our game simulation all entities can fire in any direction even while moving from one location to other. With a good set of parameters the agents evolve effective micro that tries to maximize damage done to enemy agents while minimizing damage taken. The c coefficients have a range of [-1000, 1000] and e exponents range from [-7, 8]. The influence map parameter r and w_3 range between 0 to 8 cells while I_f, and w_1, w_2 and range between 0 and 1.

6.4 Experiments on FastEcslent

We ran our experiments in a game simulator called FastEcslent, developed in our lab for evolutionary computing research in games[107]. Unlike other available RTS-like engines, FastEcslent enables 3D movement, and can run without graphics thus providing simpler integration with evolutionary computing approaches. As mentioned earlier, we need more than one scenario to produce robust solutions and thus we created multiple different scenarios for micro evolution.

We created two sides; player1 with 5 Vultures and 5 Zealots and player2 with 5 Vultures and 5 Zealots. A Vulture is a vulnerable agent with low hit-points but high movement speed, a ranged weapon, and considered effective when outmaneuvering slower melee agents. A Zealot is a melee agent with short attack range and low movement speed but has high hit-points. Table 6.1 shows the details of these properties for both Vultures and Zealots which are used in our experiments. Scenarios are constructed from "clumps" and "clouds" of entities; defined by a center and a radius. All agents in a clump are distributed randomly within a sphere defined by radius of 400. Agents in a cloud are distributed randomly within 10 agents of the sphere boundary defined by the center and radius (also 400).

The training scenarios are as follows: (a) A clump of player1 versus a clump of player2, (b) A clump of player1 agents surrounded by a cloud of player2 agents, (c) A clump of player2 agents surrounded by a cloud of player1 agents, (d) A set of player1 agents within range of 250 in all three dimension centered at the origin and a set of player2 agents within 250 in all three dimension centered at 650, and (e) the same distributions of agents but with the players swapping their centers. Our evaluation function ran each of these five scenarios for every chromosome during fitness evaluation and the value returned by the simulation for each objective is averaged over these scenarios. This results in evolving more reliable micro that can do well under different training scenarios.

We enabled 3D movement by adding maximum (1000) and minimum (0) altitudes, as well as a climb rate constant r_c of 2. Agents move in 3D by setting a desired heading, a desired altitude, and a desired speed. This type of set-up is im-

Table 6.1: Game agent simulation properties in FastEcslent

Property	Vulture	Zealot
Hit-points	80	160
MaxSpeed	64	40
MaxDamage	20	16*2
Weapon's Range	256	224
Weapon's Cooldown	1.1	1.24

portant because agents micro is governed by physics; that means micro depends on agents turning rate, speed and acceleration. The effectiveness of an agent that can turn quickly and attack in all directions is different compared to the effectiveness of an agent that does not have the ability to turn quickly and can not attack in all directions. Furthermore, an agent with high-speed and high acceleration has the ability to flee quickly when outnumbered compared to an agent with less speed and less acceleration.

6.4.1 FastEcslent Results and Discussion

In order to produce high quality micro behavior, finding a good opponent to play against is crucial. Instead of hand-coding an opponent, we use a two step approach to find a good opponent. First, we generated 30 random chromosomes that we used as opponents and ran NSGA-II against each one of them with population size of 20 for 30 generations. The best opponent is the one that does most damage to friendly agents. We thus choose this opponent chromosome that does the most damage as the next opponent. Second, we then run our NSGA-II against this chromosome. The last generation pareto front of NSGA-II provides a diverse set of micro behaviors ranging from fleeing; less damage done and less damage received (0, 1) to kiting; more damage done and more damage received (1, 0). We choose the most balanced performance, closest to (0.5, 0.5), as the next opponent micro and repeat this process five times (five steps). These five steps provide $five$ Balanced Opponent Micro (BOM) chromosomes (BOM1 - BOM5). Since we do not use hand-coded opponent micro, we ran 1000 randomly generated chromosomes against BOM1 through BOM5 to better understand their effectiveness.

Figure 6.2 shows the performance of 1000 randomly generated chromosomes against BOM1 to BOM5. In the figure, the line marked BO*i* represents the pareto front of these 1000 random chromosomes against BOM*i*. That is, the points on the line represent the best performers from among these 1000 random individuals against BOM*i*. The x-axis represents damage done, while the y-axis represents (1 - damage received). The point $(1, 1)$ then represents micro that destroys all enemy agents and receives no damage. $(1, 0)$ is micro that does destroys all opponents but also loses all friendly agents. $(0, 1)$ usually indicates fleeing behavior, friendly agents inflict no damage and receive no damage. $(0, 0)$ is bad, friendly agents did no damage and received maximal damage - micro to be avoided. From the figure, we can see that the line marked as BO4 did worse. This means that the 1000 chromosomes did worse against BOM4, the balanced individual from the last generation pareto front in step four above. Furthermore, this implies that BOM4 is the most difficult balanced opponent to play against and thus, with high probability, a good opponent to evolve against.

We use real-coded parameter with simulated binary crossover along with polynomial mutation. After experimenting with different values, we set crossover and mutation distribution indexes to 20. The probabilities of crossover and mutation were set to (0.9) and (0.05) to drive diversity. We evolved micro for groups of 5 vultures and 5 zealots versus an identical opponent also with 5 vultures and 5 zealots

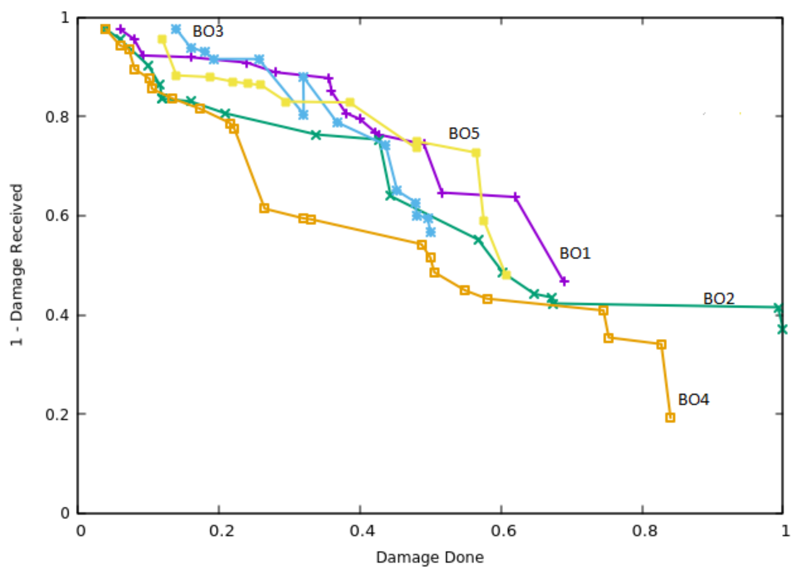


Figure 6.2: Pareto front of 1000 random chromosome against BOM1 to BOM5

considering a population size of 50 for 75 generations. Figure 6.3 shows the evolution of the pareto front at intervals of fifteen (15) generations for one run of our parallelized-evaluation NSGA-II and in total we ran evolution 10 times. Broadly speaking, the pareto front moves towards $(1, 1)$ while maintaining representatives along the tradeoff curve for maximizing damage done and minimizing damage received. We can see the maintainence of a diverse set of micro making a diverse set of tradeoffs between damage done and received. These results provide evidence that we can evolve a diverse set of micro tactics that learns to perform well against an existing opponent.

To test the effectiveness of our evolutionary multi-objective optimization approach, we played a balanced individual and a fleeing individual from the 75th generation pareto front against 3750 randomly generated chromosomes. Figure 6.4 shows how these random chromosomes did against the evolved micro. For comparison, we also ran BOM4 and fleeing micro from last generation pareto front of step four against these random individuals. The figure shows that our evolved bal-

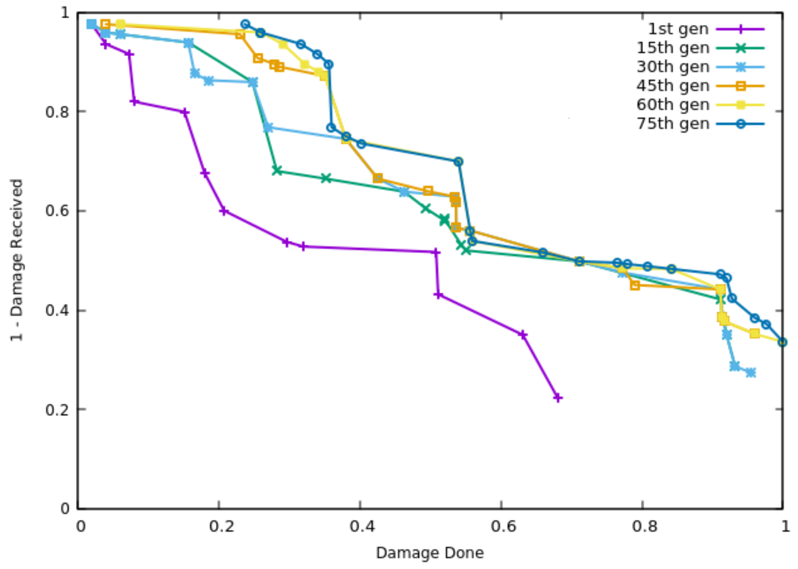


Figure 6.3: Micro evolution for friendly agents in final experiment against BOM4

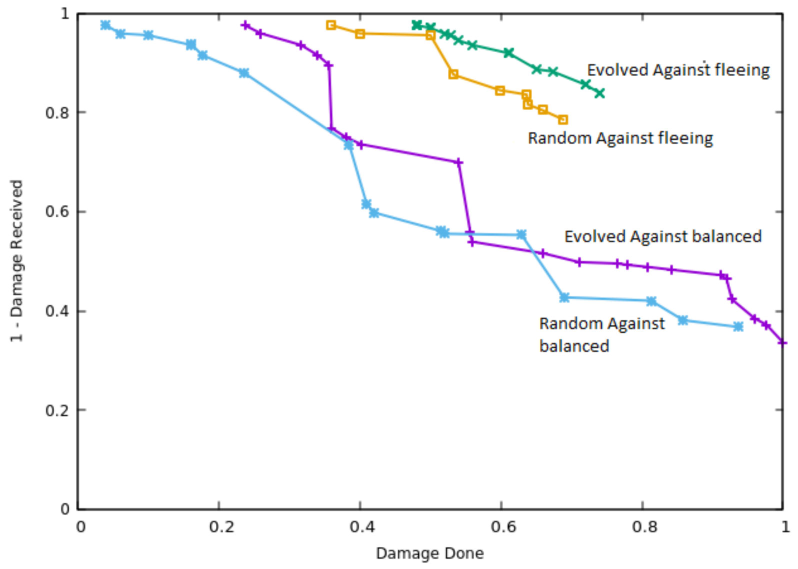


Figure 6.4: Comparing evolved micro against 3750 random chromosomes

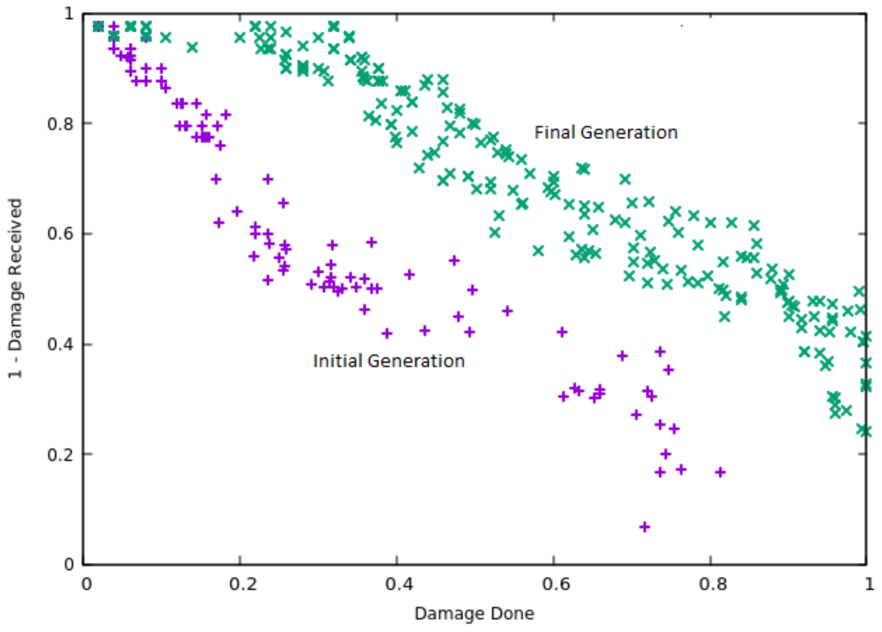


Figure 6.5: Initial and final generation pareto front over ten runs for evolved micro

anced individual does better than fourth step balanced individual, and the evolved fleer also does better than the step four fleer. Evolutionary multi-objective optimization's approach to producing a diverse set of solutions along the pareto front leads seems to lead to robust micro.

Figure 6.5 plots the combined pareto front in the first generation over all ten random seeds versus the combined pareto front in the last generation over the ten random seeds. That is, we first did a set union of the pareto fronts in the ten initial randomly generated populations. The points in this union over all ten runs are displayed as purple + for the initial generation (generation 1) points and as green $\times$ for the points in the final generation (generation 75). The figure then shows progress between the first and last generation over all ten runs. We can see that the last generation pareto front produces micro on one extreme on the left (0.02, 0.98) representing a strong tendency to flee, to the other extreme on the right (1,0.25)

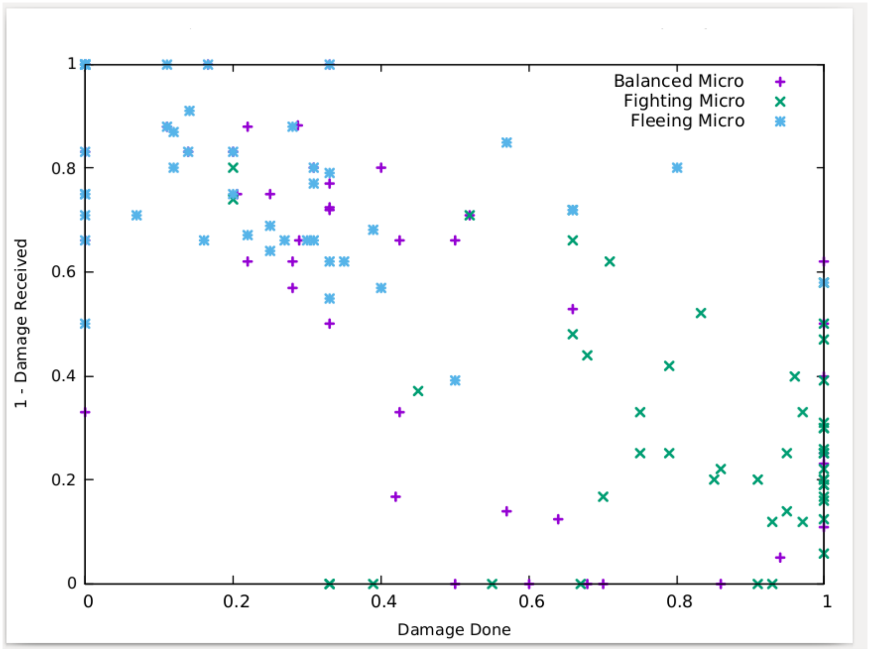


Figure 6.6: Testing the robustness of evolved attacking, balanced, and fleeing micro against BO4 on 50 random scenarios.

denoting an aggressive attacking micro behavior. There are a number of solutions near the middle with balanced micro behavior.

To further check the robustness of our evolved micro in the last generation pareto front, we decided to select one balanced, one fleeing, and one attacking example of micro from this last generation and play against BOM4 in a variety of different randomly generated scenarios. In these 50 scenarios, we randomly varied the numbers of zealots and vultures, both between 5 – 10, and made sure that both sides had identical agents. Figure 6.6 shows results, indicating that the evolved attacking micro (green $\times$s) comes in on the lower right, generally dealing damage while also receiving significant damage. Purple + indicates evolved balanced micro distributed in the middle showing balanced behaviors. Similarly, Blue $*$ indicates evolved fleeing micro and mostly found on the upper right corner of the graph. This shows fleeing behavior of micro. On average over the 50 scenarios, the attacking micro leads to objective function values of $(0.812, 0.291)$, while the balanced micro leads to an average of $(0.39, 0.59)$ and the fleeing micro's average fitness comes to $(0.21, 0.79)$.

Finally, we played the evolved attacking micro against larger numbers of opponents where 5 vultures and 5 zealots controlled by our evolved attacking micro plays against and defeats 5 vultures and 10 zealots controlled by BOM4. The attacking micro manages to destroy all 15 opponent agents showing that better micro can win even when outnumbered.

6.5 Experiments on StarCraft-II

We showed that our approach works well in a RTS game platform developed in our lab for research purposes. Next we choose a popular RTS game, StarCraft-II, to control agents using our approach. This will provide the evidence of generalizability of our approach in different RTS game platforms. Furthermore, in addition we also compare different approaches to micro in RTS games in SC2. To evolve micro behaviors in SC2, we generated 4 different training scenarios as shown in Figure 6.7. Unlike in the FastEcslent RTS game platform where we first evolved a good opponent to play against, in SC2 there is a default AI player and we evolved micro against this default AI. We call this default AI as SC2 AI.

Since SC2 is a commercial game platform, we do not have the level of control on game agents as compared to game agents in FastEcslent. For example, in SC2 different types of agents have different speed and they always move at their respective fixed speed. We can only change the heading of agents, also if agent is a flying agent then the flying altitude for that agent is fixed. We created identi-

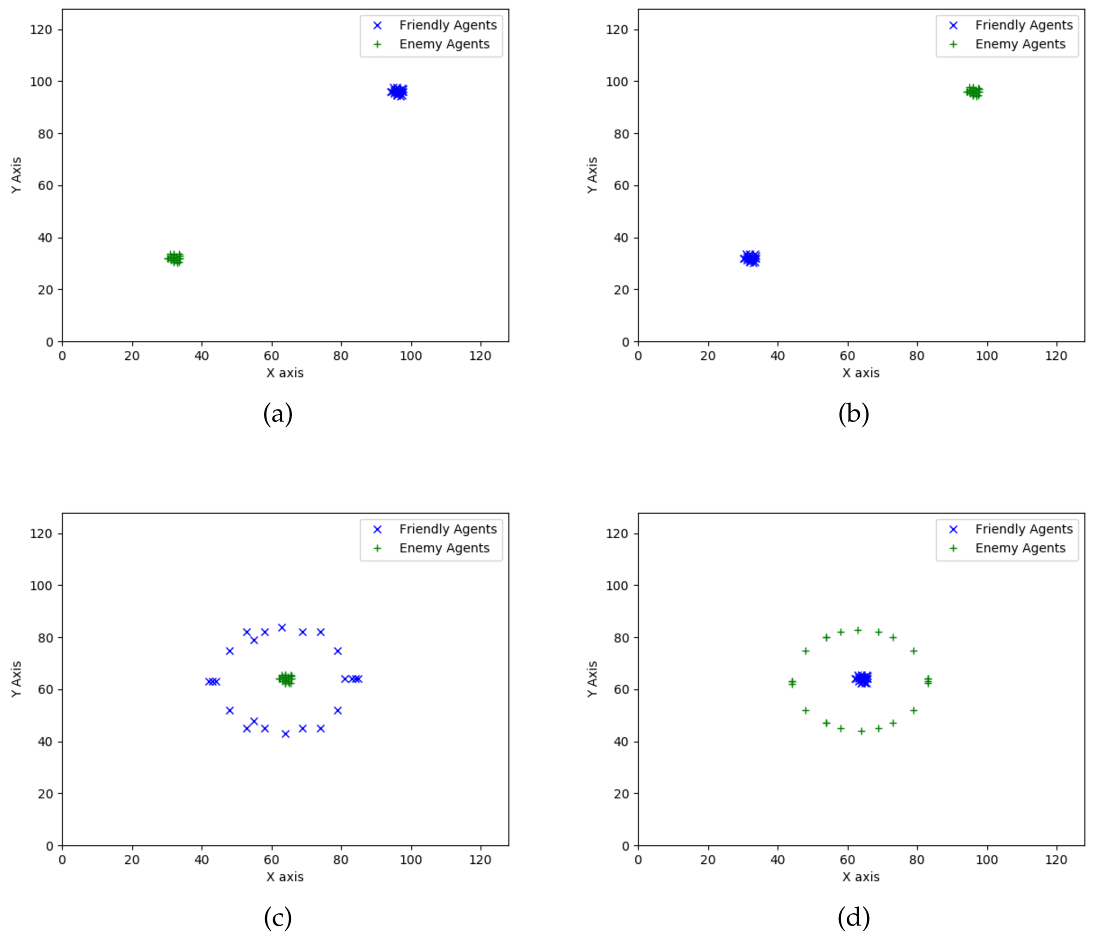


Figure 6.7: Four training scenarios to evolve RTS micro behaviors in SC2.

cal teams where each team is composed from upto three different types of agents. We compare three different approaches to generating micro in SC2. These three approaches are based on Meta Search (MS) [81], Neuroevolution of Augmenting Topologies (NEAT) [78], and our approach. In RTS game micro, we call our approach, a Pure Potential Field (PPF) approach.

6.5.1 StarCraft-II Results and Discussion

We evolve micro over four training scenarios using the three different approaches for teams composed of two different types of agents against an identical group controlled by the SC2 AI. We then repeat the same experiments with groups composed from three different types of agents as well. Due to time constraints, all experiments used a population size of 20, run for 20 generations, with a simulated binary crossover rate of 0.95, and mutation rate of 0.05.

Evolution Against SC2 AI

Figure 6.8 (a) shows the last generation pareto fronts evolved using MS, PPF and NEAT for groups composed of two types of agents on each side. In our experiment, for 2v2, we considered a group composed of 5 Stalkers (ranged) and 15 Zealots (melee), playing against an identical opponent group (5 Stalkers and 15 Zealots) on the training scenarios. Results indicate that all three approaches quickly evolve micro that beats the SC2 AI on the training scenarios. The pareto front stays above $(0.5, 0.5)$. Since MS is hard-coded for kiting we see that MS produces a smaller spread (one point on the far right) on the pareto front compared to PPF. Kiting is a hit and run strategy. Meta search, in addition to kiting, was also designed to be aggressive, so the MS micro tends to eliminate all opponents. Whereas, PPF was not designed for any specific type of agent and does not have a target selection mechanism. This leads PPF to produce diverse micro ranging from fleeing; low damage done and low damage received, to aggressive; high damage done and high damage received. NEAT also generates aggressive micro behaviors in the last generation pareto front on training scenarios.

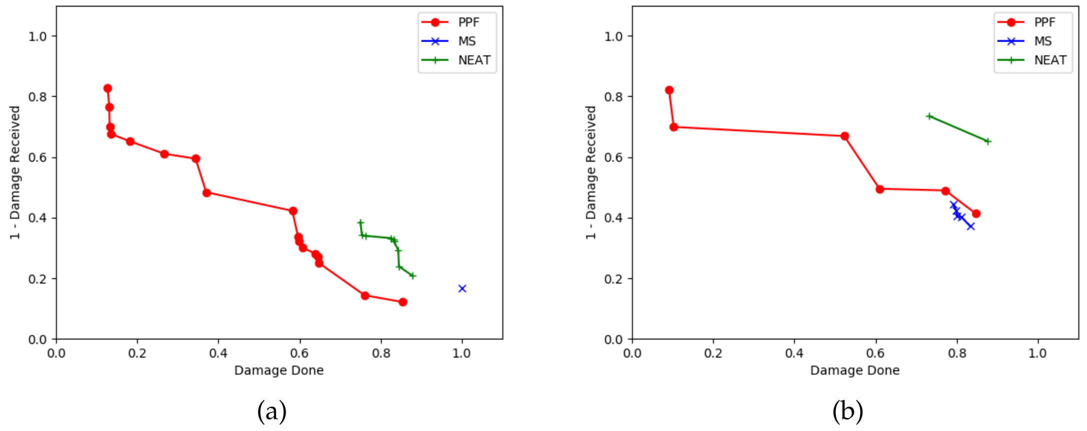
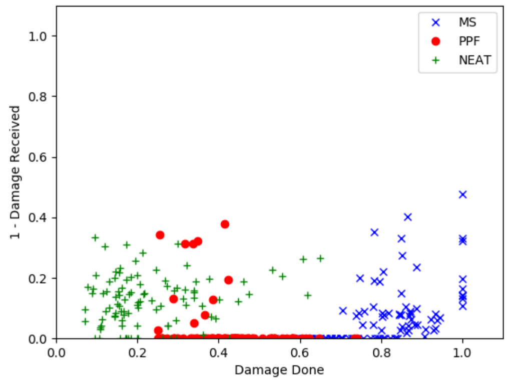


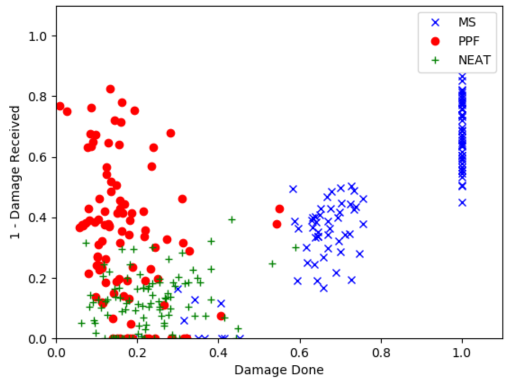
Figure 6.8: Last generation pareto fronts of MS, PPF, and NEAT for (a) 2v2 and (b) 3v3 on training scenarios against SC2AI

Figure 6.8 (b) shows the last generation pareto fronts evolved using MS, PPF and NEAT when there are three types of agents on each side ($3v3$). In these experiments, we considered a group composed of 10 Marines (ranged), 6 Marauders (ranged), and 4 Medivacs (non-ranged) playing against an identical opponents group (10 marines, 6 marauders and 4 medivacs) on our training scenarios. Results shows that all three approaches work well against the SC2 AI. MS and NEAT generate aggressive micro behaviors whereas PPF generates a more diverse set of micro.

To evaluate the performance and robustness of evolved micro behaviors using three different approaches, we test our evolved micro M_0 over two different test scenarios against SC2 AI for 100 games where agent initial positions are randomly generated in each of the two scenarios. In the first test scenario, friendly agents are randomly generated in the lower half of the flat map while enemy agents are randomly generated to occupy the upper half of this map in the 100 different games. The second testing scenario randomly generates all agents across the entire map for each of the 100 different games.



(a)



(b)

Figure 6.9: Performance of M_0 for (a) 2v2 and (b)3v3 on testing scenarios against SC2AI

Figure 6.9 (a) shows results obtained on test scenarios with two types of agents on each side. We generated a total of 300 points, 100 points for each MS, PPF, and NEAT based approaches where each point is a pair of two objective values. As expected, MS produces aggressive behavior but PPF and NEAT did not perform well. The average of $Objective_1$ and $Objective_2$ over 100 games for MS found to be 0.789, and 0.068. Similarly for PPF the averaged objectives are 0.425 and 0.022, and for NEAT averaged objectives are 0.23 and 0.126. We believe this is due to MS having an explicit target selection component. In the absence of such a target selection mechanism randomly dispersed agents find it difficulty to gather in a group to inflict more damage to opponents. NEAT did not perform well and the opaqueness of neural nets does not permit a good understanding of this performance deficit. Furthermore, a 20 population size run for 20 generations also may not have been sufficient for NEAT to evolve the structure and weights of a neural network, or for PPF, to fine tune coefficients and exponent parameters in order to evolve good micro behaviors. The combination of human written algorithm tuned by a genetic algorithm seems to combine human subject matter expertise with algorithm tuning to quickly achieve high performance.

Figure 6.9 (b) shows the results obtained while testing performance and robustness over unseen test scenarios with three different types of agents on each side. Again, MS outperformed both PPF and NEAT. The average of $Objective_1$ and $Objective_2$ over 100 games for MS found to be 0.769, and 0.456. Similarly for PPF, the averaged objectives are 0.17 and 0.334, and for NEAT, the averaged objectives are 0.229 and 0.122.

Evolution Against Evolved Micro

The previous experimental results show that all three approaches work well against the SC2 AI but SC2's micro AI is not considered very good[1]. Thus, to improve the quality of evolved micro and to test co-evolution's viability to improve micro performance, we choose a member from the last generation pareto front as an opponent (M_0) and evolve against M_0. The last generation pareto front may contain members with fleeing, balanced, and aggressive micro behaviors, we choose a member with aggressive micro to evolve against. As explained earlier, we repeated this cycle three times resulting in better and better micro. We then examine M_3's performance on our two testing scenarios for a 100 different randomly initialized agent positions. Figure 6.10(b) shows the performance of M_3 for two different types of agents. Compared to M_0, M_3 improves both objectives for PPF from 0.425 (damage done) and 0.022 (1-damage received) to 0.635 and 0.048 where all objective values are averaged over 100 games. PPF controlled agents were able to inflict more damage as they learned to move in groups (flocking). M_3 from meta search achieves average objective values of 0.717 and 0.0824 which are similar to the re-

[1]For example, SC2's AI does not implement kiting.

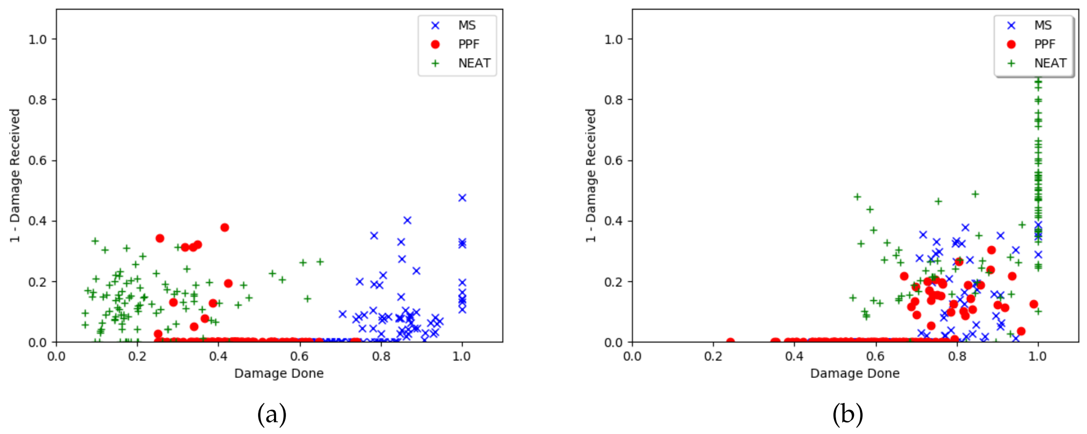


Figure 6.10: Performance of (a) M_0 and (b) M_3 for 2v2 on testing scenarios against SC2AI

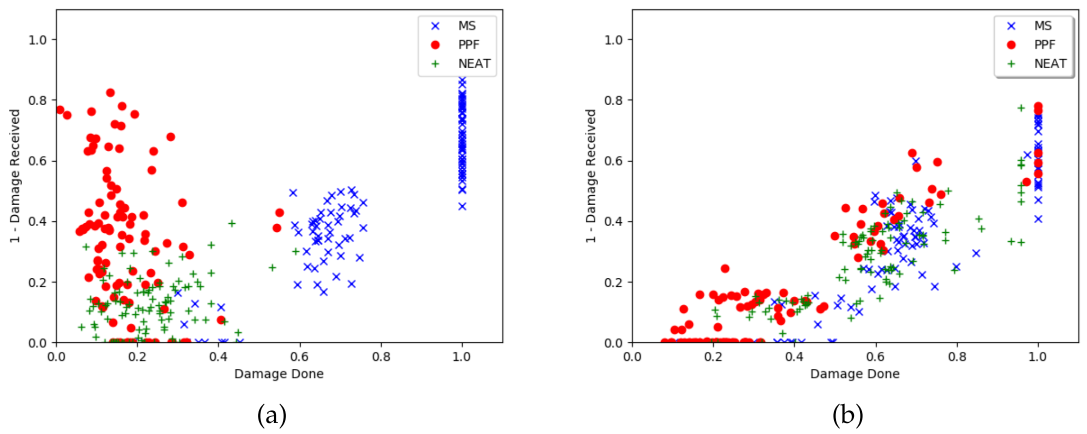


Figure 6.11: Performance of (a) M_0 and (b) M_3 for 3v3 on testing scenarios against SC2AI

sults produced by M_0 - 0.789 and 0.068. NEAT obtained average objective values of 0.23 and 0.126 with M_0 and 0.867 and 0.382 with M_3.

Figure 6.11(b) shows the performance of M_3 for three different types of agents. For 3v3 with PPF, M_0 produces damage done and 1- damage received of 0.17 and 0.334 respectively, compared to M_3's fitness of 0.377 and 0.175. However, MS did not improve significantly from M_0 to M_3. NEAT improves both objectives with

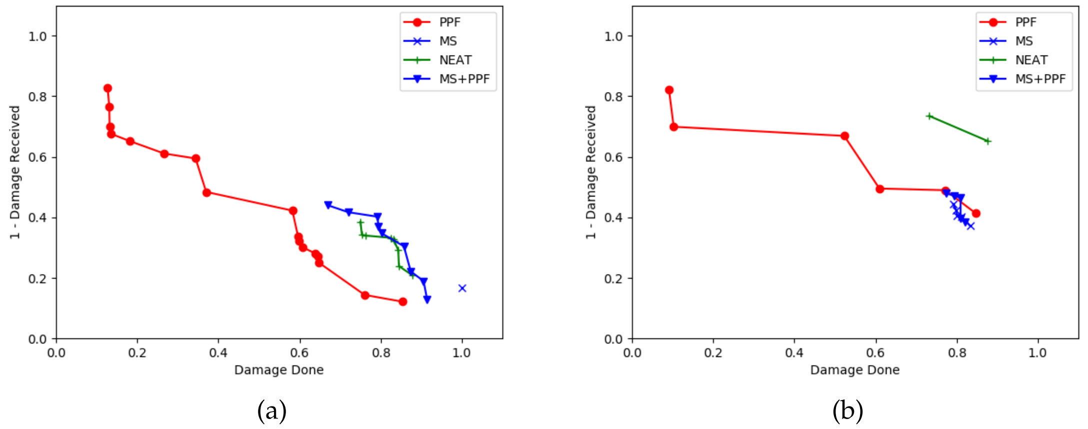


Figure 6.12: Last generation pareto fronts of MS, PPF, NEAT, and MS+PPF for (a) 2v2 and (b) 3v3 on training scenarios against SC2AI

M_3 from 0.229 and 0.122 to 0.589 and 0.28. This shows that NEAT evolved a better neural network structure and tuned it's weights by playing against a stronger opponent compared to SC2 AI.

Evolution with Combination of Approaches

In our last set of experiments we investigated combining meta search with pure potential fields. MS was designed for ranged agents not melee agents, so combining MS with PPF enables us to take advantage of MS to control ranged agents and PPF to control all other agent types. We evolve micro behaviors with this hybrid approach on our four training scenarios. Figure 6.12 (a) and Figure 6.12 (b) show the last generation pareto fronts evolved using MS, PPF, NEAT, and combined MS+PPF for two type versus two type and three type versus three type agent groups respectively. Combining MS and PPF leads to higher performance compared to any of the three approaches alone and produces a higher performing pareto front of micro for our $2v2$ groups. However, this difference is not present on the 3*v*3 scenarios. We believe this is because Medivacs are neither ranged nor melee agents and their control has not yet had time to evolve in the $20 \times 20 = 400$ evaluations available.

These results show that all three approaches MS, PPF, and NEAT work well. However, potential fields based game agent control provides a trade-off between explainability and complexity. MS is a hard coded approach and works for ranged game agents and not for melee. However, MS is understandable, explainable but not scalable. Whereas NEAT evolves neural networks to control the movement of game agents and can be scaled to control a large number of different types of game agents. But NEAT is not understandable and explainable because of the large numbers of weights in the neural network. Different from these two, our genetic optimized potential field approach is both understandable and explainable. Unlike NEAT we know the property (such as distance) that each potential field is based on and we can do a sensitivity analysis for each of these properties to understand and explain their effect on agent behavior. Thus our approach is more flexible than meta search and can be easily extended to more types of agents and more types of micro while being more understandable and explainable than neural network based approaches. This is a significant advantage of our approach.

Evolving Micro With Many Different Types of Agents

We created identical teams to eliminate the issue of bias and evolved pareto fronts of micro behaviors on four different training scenarios. The NSGA-II ran for 50 generations with a population size of 50, a probability of crossover of 0.95, and a probability of mutation of 0.05. The experiments were conducted considering different agent combinations as shown in Table 6.2 and the attributes of each of the

Table 6.2: Different agent combinations

$XvsX$	Combinations
$PvsP$	10 Z_e, 6 S_t, 4 S_e, 3 V_t, 4 A_d
$TvsT$	10 M_r, 6 M_d, 4 M_e, 3 B_n
$ZvsZ$	10 Z_r, 6 B_e, 4 M_u

Table 6.3: Attributes and characteristics of agents. Agent types and attributes are taken from Starcraft 2.

Agent Type	Health	Range	Attack	Type
$Marine(M_r)$	45	5	A/G	R
$Marauder(M_d)$	125	6	G	R
$Medivac(M_e)$	150	11	heal, carrier	R
$Banshee(B_n)$	140	6	G	R
$Zealot(Z_e)$	100	0.1	G	M
$Stalker(S_t)$	80	6	A/G	R
$Adept(A_d)$	70	4	G	R
$VoidRay(V_r)$	150	6	A/G	R
$Sentry(S_t)$	40	5	A/G, force field	R
$Zergling(Z_r)$	35	0.1	G	M
$Baneling(B_l)$	30	0.25	G	M
$Mutalisk(M_s)$	120	3	A/G	R

agents is shown in Table 6.3. We next discuss the evolved micro behaviors for each of our three configurations.

Three Different Types of Agents

When considering three different types of agents, we created teams composed of 20 agents composed of these three types. Specifically, we used 10 Z_r, 6 B_l, and 4 M_u for both sides of the skirmish in our training and testing scenarios. Here, Z_r and B_l are melee agents and M_u is a ranged agent. Using our potential field based representation, the NSGA-II evolved a diverse set of micro behaviors. Figure 6.13 shows the evolution of micro behavior for every 10^{th} generation and we observed

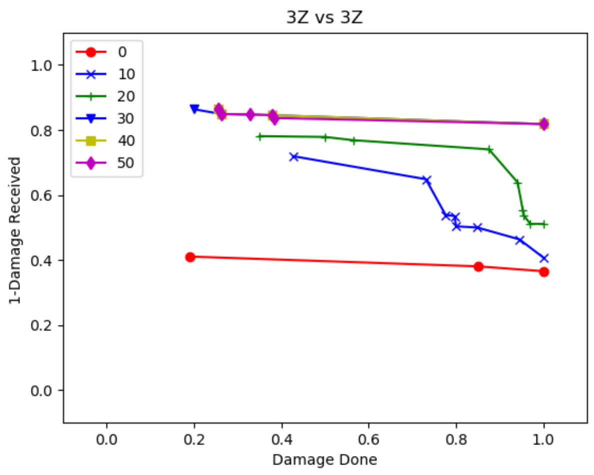


Figure 6.13: Pareto fronts at every 10th generation. Each team is composed from three different types of agents.

a significant difference between the first (0th) and the last (50th) generation pareto front solutions. A fitness $(1, 0.81)$ specifies that all enemy agents died and friendly agents had 81% remaining health at the end of skirmish. In Figure 6.13, the last generation pareto front solutions shows a range of behaviors from fleeing to aggressive. Friendly team agents were able to eliminate all enemy team agents while receiving little damage $(1, 0.81)$.

Four Different Types of Agents

Figure 6.14 shows pareto fronts when playing with friendly terran agents against identical enemy terran agents controlled through SC2AI. The terran agents combination has many different unique abilities. For example, Marine M_r is a ranged agent with ability to attack ground and air targets but low on health. Whereas Marauder M_d is a ranged, strong agent (high health), attacks only ground targets and can also protect M_r from receiving damage thus both form a good combination together. Additionally, Medivacs M_e have the ability to heal both M_r and M_d. M_e is

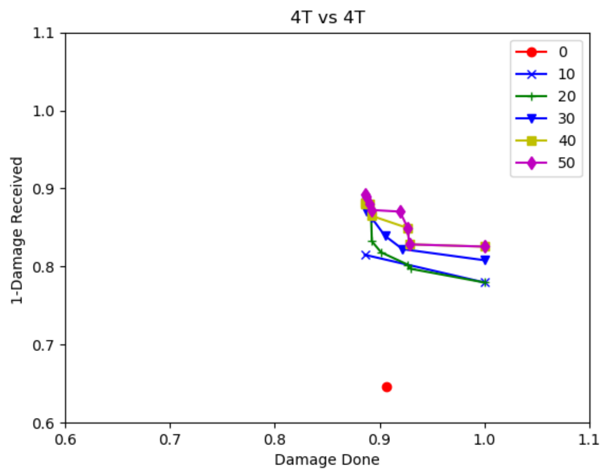


Figure 6.14: Pareto fronts at every 10^{th} generation. Each team is composed from four different types of agents.

a healer not attacker thus to add more fire power Banshees B_n are added into $4v4$ terran team configuration. B_n is a strong flying agent and can only attack ground targets. As shown in Figure 6.14, terran agents improved their performance significantly from initial generation to last generation in terms of damage done and damage received. Friendly terran agents learned effective strategies to counter the swarm of enemy terran agents by figuring out weakness in their strategy. B_n of friendly team eliminates the M_r of enemy team and remember in this team setting, M_r is the only agent that can attack aerial targets.

Five Different Types of Agents

Table 6.2 shows the team combination with 27 agents from five different types of protoss race agents. Figure 6.15 shows the pareto fronts at every 10^{th} generation when friendly protoss agents play against an identical enemy protoss agents controlled through SC2AI. The agents in such combination includes agents with variety abilities.For example, zealots Z_e are strong ground attack melee agents. The

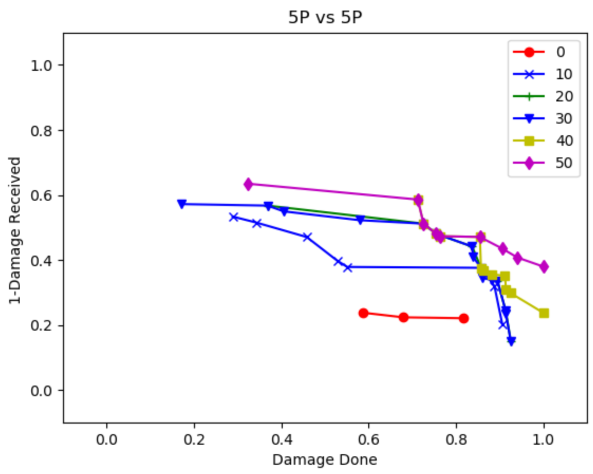
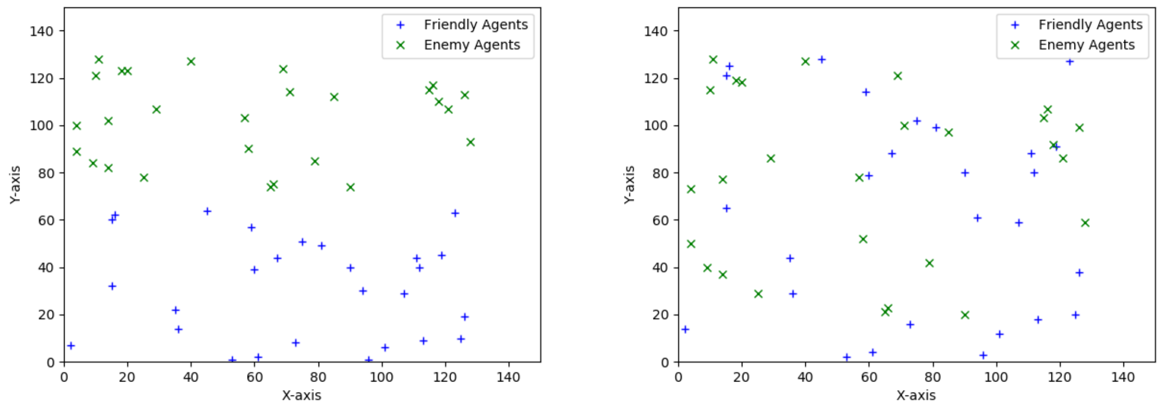


Figure 6.15: Pareto fronts of every 10th generation. Each team composed from five different types of agents.

melee nature of Z_e makes them more vulnerable against agents with large firing range. On the other hand, Stalkers S_t are fast moving, ranged agents and have the ability to attack both ground and air targets. Z_e and S_t together make a formidable combination against any enemy. The 5v5 protoss agents combination includes an agent called "Sentry" S_e, an agent with a unique ability to create force fields (for a time) that block movement of any ground agents. These force fields are used extensively to defend own agents or infrastructure. In addition to these agents, the combination also includes Void Ray V_r and Adept A_d. V_r is a flying agent with the ability to attack both ground and air targets, and A_d is a ground attack agent. The five types, Z_e, S_t, S_e, V_t, and A_d make a strong combination of agents with high defense and attack capability. As shown in Figure 6.15, initially, in the 0th generation our agents performed poorly but gradually improve over a period of 50 generations. The last generation pareto front (50th) shows a variety of micro including fleeing, balanced, and aggressive.



(a) (b)

Figure 6.16: Two types of testing scenarios with a group composed of different types of agents against the same group of agents

Experiments on Testing Scenarios

To measure the robustness of evolved solution, we picked an aggressive micro from the last generation pareto front and tested the performance on two unseen scenarios as shown in Figure 6.16. We ran 25 simulation with different starting position on each of the two testing scenario. For $5v5$, the averaged fitness obtained on the first scenario (a) is $(0.88, 0.27)$, and the averaged fitness on the second scenario (b) is $(0.83, 0.11)$. On the first testing scenario, agents get the chance to quickly gather to attack opponents and this leads to better fitness compared to the second scenario. Similar experiments were conducted with different teams as well and the results indicate that evolved solutions performed well on unseen scenarios. This provides evidence of the generalizability and scalability of our approach to evolve high performing diverse micro against opponents.

6.6 Conclusions

We evolved micro behavior in two different RTS game simulation platforms using the same approach. The influence map provided a target location and game agents were controlled using a set of potential fields towards the target location. In FastEcslent, we first evolved the opponent's AI and used that as a final opponent. Next, we considered a different and popular RTS game, StarCraft-II to evolve micro against the default SC2 AI. In SC2, we compared three different approaches with different levels of explainability; meta-search, our approach (pure potential fields) and NEAT, to generate micro behaviors for groups composed from multiple different types of agents in RTS games. First, we evolved micro against the SC2 AI over four training scenarios and subsequently tested their performance and robustness over two unseen testing scenarios. Results showed that all three approaches generated micro that can defeat SC2 AI over our training scenarios. Meta search, based on a human coded algorithm and tuned by a GA, turned out to be more robust and outperformed both PPF and NEAT on our 200 randomized testing scenarios.

To improve the quality of evolved micro and to test self-play or co-evolution's viability to improve micro, we manually co-evolved better micro over three cycles. The resulting micro improved performance for all three approaches for skirmishes composed of two types of agents on each side. For three types of agents on each side, PPF and NEAT improved their micro performance but MS did not show significant improvement. Lastly, we combined MS to control ranged agents and PPF to control melee/non-ranged agents. This combination improved the quality of evolved micro for groups composed from ranged and melee agents. These results encourage us to focus on more hybrid approaches to improve performance.

Lastly we evolved micro for groups composed of three to five different types of game agents and up to 27 total numbers of agents using NSGA-II. As the number of different types of agents increases, the complexity of the problem increases. The results showed that our approach evolved different micro by tuning IMs and potential field parameters. These results show that our approach is generalizable, scalable, and because the coefficients and exponents of known potential fields control the movement agent, our approach is also understandable.

CHAPTER 7
CONCLUSIONS AND FUTURE WORK

7.1 Conclusions

We used genetic algorithms optimized potential fields to control the movement of agents for group tasking. Our approach used task specific potential fields to guide agents. Since potential fields are highly non-linear and tuning their parameters is difficult, we used genetic algorithms to tune tasks specific potential field parameters. We choose three different challenging problems to show the applicability and effectiveness of our approach. Each agent acted independently under the influence of potential fields on all three problems and worked in a decentralized manner.

We proposed GANet to deploy wireless mesh networks using UAVs for search and rescue. Potential fields based on bandwidth and distances controlled the movement of UAVs to generate mesh networks where each UAV acted independently. We formulated the problem as a single objective optimization problem and a genetic algorithm optimized the potential field parameters to maximize the sum of bandwidth coverage and longevity. Results on a sample test problem showed that the genetic algorithm is faster compared to the random exhaustive search and a hill climber (gradient ascent based technique) when searching for optimal solutions in the search space of potential field parameters. This justified use of the genetic algorithm to tune potential field parameters. Results on complex problems showed that GANet outperformed ATRI with one hop communication between UAVs. Furthermore, experimental results showed that with two hops UAVs communication the performance improved significantly compared to one hop communication.

We used the same approach as in the case of wireless mesh network deployment, to control the movement of UAVs for fire perimeter coverage. Potential fields based on fire intensity and distances controlled the movement of UAVs. We formulated the problem as a multi objective problem to maximize fire coverage while minimizing energy consumption of UAVs. Here, fire coverage refers to only the fire perimeter, not the entire region covered by the fire. Results showed that we achieved complete fire coverage with enough UAVs. In both the problems: wireless mesh network deployment and fire boundary coverage, we assumed a single type of UAV.

The third and last problem was to control multiple types of game agents to win skirmishes in RTS games. We used potential fields based on distance, health, and weapon properties to control game agents and formulated the problem as a multi objective optimization problem that maximizes damage done to opponents and minimizes damage received by friendly agents. We conducted experiments on two different RTS game simulation platforms: FastEcslent and SC2. Results on FastEcslent, showed that our approach generated a pareto front of good quality micro behaviors. We also compared our approach performance with two other techniques for generating micro behaviors on SC2 and the results showed that the all three approaches performed well. However, our approach provides a trade-off between explainability and complexity. Lastly, we increased the complexity of the problem by adding different types of game agents to friendly and enemy teams and the results showed that our approach was able to evolve a diverse set of good quality micro. This showed the scalability of our approach to large numbers of heterogeneous agents for group tasking.

These results show that our approach is generalizable to different domain specific problems and scalable to a large number of different types of agents. Our potential field based approach is simple to implement and since we know what potential fields are based on we can better understand and explain agent's behavior.

7.2 Future Work

We plan to extend the work on wireless mesh network deployment in three ways in the future. Results show that a few users are not covered by any UAVs, we plan to modify our potential field based representation to ensure all users are covered and follow users as they move. Second, we assumed only one type of UAV for network deployment but in the future we aim to deploy wireless mesh networks that use different types of UAVs. Lastly, we assumed that enough UAVs were available for deployment. In reality, this might not be the case, thus we plan to modify our approach to work with a lower number of UAVs for both search and service phases.

In RTS game micro, we plan to investigate cooperation between different types of agents and design a performance metric to measure cooperation. Furthermore, we aim to conduct experiments to measure an individual game agent's contribution to achieve the objectives of given tasks. Finally, we also plan to investigate other machine learning techniques for decision making in dynamic environments.

Earlier, we had planned to use our potential fields based approach to control the movement of autonomous surface vehicles (ASVs) for safe navigation in compliance with the collision regulations (COLREGs). This work is currently in progress and some of the experimental results on generating safe and COLREGs complained paths have been accepted for publication in the *Proceedings of the IEEE/MTS Oceans 2021 Conference*. In the future, we plan to investigate the scope of our approach to navigate ASVs.

BIBLIOGRAPHY

[1] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Autonomous robot vehicles*, pages 396–404. Springer, 1986.

[2] John H Holland. Genetic algorithms. *Scientific american*, 267(1):66–73, 1992.

[3] Mandavilli Srinivas and Lalit M Patnaik. Genetic algorithms: A survey. *computer*, 27(6):17–26, 1994.

[4] Robin C Purshouse and Peter J Fleming. Why use elitism and sharing in a multi-objective genetic algorithm? In *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, pages 520–527, 2002.

[5] J. Schmitt and H. Kostler. A multi-objective genetic algorithm for simulating optimal fights in starcraft II. *IEEE Conference on Computational Intelligence and Games*, 2016.

[6] Ridong Zhang and Jili Tao. A nonlinear fuzzy neural network modeling approach using an improved genetic algorithm. *IEEE Transactions on Industrial Electronics*, 65(7):5882–5892, 2017.

[7] Rahul Dubey, Sushil J Louis, and Shamik Sengupta. Evolving dynamically reconfiguring uav-hosted mesh networks. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, 2020.

[8] Rahul Dubey and Sushil J Louis. Evolving potential field parameters for deploying uav-based two-hop wireless mesh networks. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2021.

[9] Kripash Shrestha, Rahul Dubey, Ashutosh Singandhupe, Sushil J Louis, and Hung La. Multi objective uav network deployment for dynamic fire coverage. In *2021 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, 2021.

[10] Rahul Dubey, Joseph Ghantous, Sushil Louis, and Siming Liu. Evolutionary multi-objective optimization of real-time strategy micro. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8, 2018.

[11] Rahul Dubey, Sushil Louis, Aavaas Gajurel, and Siming Liu. Comparing three approaches to micro in rts games. In *2019 IEEE Congress on Evolutionary Computation (CEC)*, pages 777–784, 2019.

[12] Rahul Dubey and Sushil J Louis. Multi-objective evolutionary algorithms for distributed tactical control of heterogeneous agents. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2021.

[13] John H Holland. *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence.* U Michigan Press, 1975.

[14] Darrell Whitley. A genetic algorithm tutorial. *Statistics and computing*, 4(2):65–85, 1994.

[15] Chang Wook Ahn and Rudrapatna S Ramakrishna. Elitism-based compact genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 7(4):367–385, 2003.

[16] T. Tomic, K. Schmid, P. Lutz, A. Domel, M. Kassecker, E. Mair, I. L. Grixa, F. Ruess, M. Suppa, and D. Burschka. Toward a fully autonomous uav: Research platform for indoor and outdoor urban search and rescue. *IEEE Robotics Automation Magazine*, 19(3):46–56, 2012.

[17] E. Semsch, M. Jakob, D. Pavlicek, and M. Pechoucek. Autonomous uav surveillance in complex urban environments. In *2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, volume 2, pages 82–85, 2009.

[18] S. J. Lee, D. Lee, and H. J. Kim. Cargo transportation strategy using t3-multirotor uav. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 4168–4173, 2019.

[19] Francesco Nex and Fabio Remondino. Uav for 3d mapping applications: a review. *Applied geomatics*, 6(1):1–15, 2014.

[20] P. Tokekar, J. V. Hook, D. Mulla, and V. Isler. Sensor planning for a symbiotic uav and ugv system for precision agriculture. *IEEE Transactions on Robotics*, 32(6):1498–1511, 2016.

[21] Milan Erdelj, Michał Król, and Enrico Natalizio. Wireless sensor networks and multi-uav systems for natural disaster management. *Computer Networks*, 124:72–86, 2017.

[22] N. H. Motlagh, M. Bagaa, and T. Taleb. Uav-based iot platform: A crowd surveillance use case. *IEEE Communications Magazine*, 55(2):128–134, 2017.

[23] Qingqing Wu, Jie Xu, and Rui Zhang. Capacity characterization of uav-enabled two-user broadcast channel. *IEEE Journal on Selected Areas in Communications*, 36(9):1955–1971, 2018.

[24] Mohammad Mozaffari, Walid Saad, Mehdi Bennis, Young-Han Nam, and Mérouane Debbah. A tutorial on uavs for wireless networks: Applications, challenges, and open problems. *IEEE Communications Surveys & Tutorials*, 2019.

[25] Samira Hayat, Evşen Yanmaz, and Raheeb Muzaffar. Survey on unmanned aerial vehicle networks for civil applications: A communications viewpoint. *IEEE Communications Surveys & Tutorials*, 18(4):2624–2661, 2016.

[26] Qixing Feng, Eustace K Tameh, Andrew R Nix, and Joe McGeehan. Wlcp2-06: Modelling the likelihood of line-of-sight for air-to-ground radio propagation in urban environments. In *IEEE Globecom 2006*, pages 1–5. IEEE, 2006.

[27] Isabelle Bucaille, Serge Héthuin, Andrea Munari, Romain Hermenier, Tinku Rasheed, and Sandy Allsopp. Rapidly deployable network for tactical applications: Aerial base station with opportunistic links for unattended and temporary events absolute example. In *MILCOM 2013-2013 IEEE military communications conference*, pages 1116–1120. IEEE, 2013.

[28] Yong Zeng, Rui Zhang, and Teng Joon Lim. Wireless communications with unmanned aerial vehicles: Opportunities and challenges. *IEEE Communications Magazine*, 54(5):36–42, 2016.

[29] Dalimir Orfanus, Edison Pignaton de Freitas, and Frank Eliassen. Self-organization as a supporting paradigm for military uav relay networks. *IEEE Communications Letters*, 20(4):804–807, 2016.

[30] Akram Al-Hourani, Sithamparanathan Kandeepan, and Simon Lardner. Optimal lap altitude for maximum coverage. *IEEE Wireless Communications Letters*, 3(6):569–572, 2014.

[31] Mohammad Mozaffari, Walid Saad, Mehdi Bennis, and Merouane Debbah. Drone small cells in the clouds: Design, deployment and performance analysis. In *2015 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2015.

[32] Rajdeep Dutta, Liang Sun, and Daniel Pack. A decentralized formation and

network connectivity tracking controller for multiple unmanned systems. *IEEE Transactions on Control Systems Technology*, 26(6):2206–2213, 2017.

[33] Hao Fang, Yue Wei, Jie Chen, and Bin Xin. Flocking of second-order multiagent systems with connectivity preservation based on algebraic connectivity estimation. *IEEE transactions on cybernetics*, 47(4):1067–1077, 2016.

[34] Haitao Zhao, Lingchu Mao, and Jibo Wei. Coverage on demand: A simple motion control algorithm for autonomous robotic sensor networks. *Computer Networks*, 135:190–200, 2018.

[35] Guiling Wang, Guohong Cao, and Thomas F La Porta. Movement-assisted sensor deployment. *IEEE Transactions on Mobile Computing*, 5(6):640–652, 2006.

[36] Zsolt Gáspár and Tibor Tarnai. Upper bound of density for packing of equal circles in special domains in the plane. *Periodica Polytechnica Civil Engineering*, 44(1):13–32, 2000.

[37] Miu-ling Lam and Yun-hui Liu. Heterogeneous sensor network deployment using circle packings. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 4442–4447. IEEE, 2007.

[38] Andrew Howard, Maja J Matarić, and Gaurav S Sukhatme. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In *Distributed Autonomous Robotic Systems 5*, pages 299–308. Springer, 2002.

[39] Sameera Poduri and Gaurav S Sukhatme. Constrained coverage for mobile sensor networks. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, volume 1, pages 165–171. IEEE, 2004.

[40] Ming Ma and Yuanyuan Yang. Adaptive triangular deployment algorithm for unattended mobile sensor networks. *IEEE Transactions on Computers*, 56(7):946–847, 2007.

[41] Novella Bartolini, Tiziana Calamoneri, Tom F La Porta, and Simone Silvestri. Autonomous deployment of heterogeneous mobile sensors. *IEEE Transactions on Mobile Computing*, 10(6):753–766, 2010.

[42] Mohammad Mozaffari, Walid Saad, Mehdi Bennis, and Mérouane Debbah.

Efficient deployment of multiple unmanned aerial vehicles for optimal wireless coverage. *IEEE Communications Letters*, 20(8):1647–1650, 2016.

[43] Haitao Zhao, Haijun Wang, Weiyu Wu, and Jibo Wei. Deployment algorithms for uav airborne networks toward on-demand coverage. *IEEE Journal on Selected Areas in Communications*, 36(9):2015–2031, 2018.

[44] Jiangbin Lyu, Yong Zeng, Rui Zhang, and Teng Joon Lim. Placement optimization of uav-mounted mobile base stations. *IEEE Communications Letters*, 21(3):604–607, 2016.

[45] Xiao Zhang and Lingjie Duan. Fast deployment of uav networks for optimal wireless coverage. *IEEE Transactions on Mobile Computing*, 18(3):588–601, 2018.

[46] Amar Nath Patra, Paulo Alexandre Regis, and Shamik Sengupta. Distributed allocation and dynamic reassignment of channels in uav networks for wireless coverage. *Pervasive and Mobile Computing*, 54:58–70, 2019.

[47] DG Reina, Hissam Tawfik, and SL Toral. Multi-subpopulation evolutionary algorithms for coverage deployment of uav-networks. *Ad Hoc Networks*, 68:16–32, 2018.

[48] Dina S. Deif and Yasser Gadallah. Wireless sensor network deployment using a variable-length genetic algorithm. In *2014 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 2450–2455, 2014.

[49] Wildfire statistics- congressional research service. https://fas.org/sgp/crs/misc/IF10244.pdf. last accessed: February 21, 2021.

[50] Firefighter fatalities in the united states. https://www.nfpa.org/News-and-Research/Data-research-and-tools/Emergency-Responders/Firefighter-fatalities-in-the-United-States. last accessed: February 21, 2021.

[51] Dalimir Orfanus, Edison Pignaton de Freitas, and Frank Eliassen. Self-organization as a supporting paradigm for military uav relay networks. *IEEE Communications Letters*, 20(4):804–807, 2016.

[52] Yazan Mualla, Amro Najjar, Alaa Daoud, Stephane Galland, Christophe Nicolle, Elhadi Shakshuki, et al. Agent-based simulation of unmanned aerial

vehicles in civilian applications: A systematic literature review and research directions. *Future Generation Computer Systems*, 100:344–364, 2019.

[53] H. X. Pham, H. M. La, D. Feil-Seifer, and M. Deans. A distributed control framework for a team of unmanned aerial vehicles for dynamic wildfire tracking. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6648–6653, 2017.

[54] C. Yuan, Z. Liu, and Y. Zhang. Fire detection using infrared images for uav-based forest fire surveillance. In *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 567–572, 2017.

[55] F. Afghah, A. Razi, J. Chakareski, and J. Ashdown. Wildfire monitoring in remote areas using autonomous unmanned aerial vehicles. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 835–840, 2019.

[56] Luis Merino, Fernando Caballero, J. Ramiro Martinez-de Dios, J. Ferruz, and Anibal Ollero. A cooperative perception system for multiple uavs: Application to automatic detection of forest fires. *J. Field Robotics*, 23:165–184, 03 2006.

[57] C. Yuan, Z. Liu, and Y. Zhang. Uav-based forest fire detection and tracking using image processing techniques. In *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 639–643, 2015.

[58] Henry Cruz, Martina Eckert, J. Meneses, and J. Martínez. Efficient forest fire detection index for application in unmanned aerial systems (uass). *Sensors (Basel, Switzerland)*, 16, 2016.

[59] D. Casbeer, R. W. Beard, T. Mclain, S. Li, and R. Mehra. Forest fire monitoring with multiple small uavs. *Proceedings of the 2005, American Control Conference, 2005.*, pages 3530–3535 vol. 5, 2005.

[60] C. Phan and H. H. T. Liu. A cooperative uav/ugv platform for wildfire detection and fighting. In *2008 Asia Simulation Conference - 7th International Conference on System Simulation and Scientific Computing*, pages 494–498, 2008.

[61] Manish Kumar, Kelly Cohen, and Baisravan HomChaudhuri. Cooperative control of multiple uninhabited aerial vehicles for monitoring and fighting wildfires. *Journal of Aerospace Computing, Information and Communication*, 8:1–16, 01 2011.

[62] Ivan Maza, Fernando Caballero, Jesus Capitan, J. Ramiro Martinez-de Dios, and Anibal Ollero. Experimental results in multi-uav coordination for disaster management and civil security applications. *Journal of Intelligent and Robotic Systems*, 61:563–585, 01 2011.

[63] C. Kyrkou and T. Theocharides. Emergencynet: Efficient aerial image classification for drone-based emergency monitoring using atrous convolutional feature fusion. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 13:1687–1699, 2020.

[64] Huy Pham, Hung La, David Feil-Seifer, and Matthew Deans. A distributed control framework for a team of unmanned aerial vehicles for dynamic wildfire tracking. 09 2017.

[65] S. Ontañon, G. Synnaeve, A. Uriarte, F. Richoux, D. Churchill, and M. Preuss. A survey of real-time strategy game AI research and competition in StarCraft. *IEEE Transactions on Computational Intelligence and AI in games*, 5(4):1–19, 2013.

[66] Stuart Russell and Peter Norvig. Artificial intelligence: a modern approach. 2002.

[67] Michael Buro and Timothy Furtak. Rts games as test-bed for real-time ai research. In *Proceedings of the 7th Joint Conference on Information Science (JCIS 2003)*, volume 2003, pages 481–484, 2003.

[68] Dan Fu and Ryan Houlette. The ultimate guide to fsms in games. *AI game programming Wisdom*, 2:283–302, 2004.

[69] David Churchill and Michael Buro. Build order optimization in StarCraft. In *Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2011)*, 2011.

[70] Michael Chung, Michael Buro, and Jonathan Schaeffer. Monte carlo planning in rts games. In *CIG*. Citeseer, 2005.

[71] Radha-Krishna Balla and Alan Fern. Uct for tactical assault planning in real-time strategy games. In *Twenty-First International Joint Conference on Artificial Intelligence*, 2009.

[72] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda

Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[73] Google-DeepMind. Alphastar: Mastering the real-time strategy game StarCraft II. https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/.

[74] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[75] Kun Shao, Yuanheng Zhu, and Dongbin Zhao. Starcraft micromanagement with reinforcement learning and curriculum transfer learning. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 3(1):73–84, 2018.

[76] Maria Fox, Derek Long, and Daniele Magazzeni. Explainable planning. *arXiv preprint arXiv:1709.10256*, 2017.

[77] Wojciech Samek and Klaus-Robert Müller. Towards explainable artificial intelligence. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 5–22. Springer, 2019.

[78] Aavaas Gajurel ; Sushil J Louis ; Daniel J Méndez ; Siming Liu. Neuroevolution for rts micro. *IEEE Conference on Computational Intelligence and Games*, pages 1–8, 2018.

[79] K. O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.

[80] Johan Hagelbäck. Potential-field based navigation in starcraft. In *2012 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 388–393. IEEE, 2012.

[81] S. Liu, S. Louis, and C. Ballinger. Evolving effective micro behaviors in real-time strategy games. *IEEE Transactions on Computational Intelligence and AI in Games*, PP(99):1–1, 2016.

[82] Sushil J Louis and Siming Liu. Multi-objective evolution for 3d rts micro. In *2018 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, 2018.

[83] Siming Liu, Sushil J Louis, and Christopher Ballinger. Evolving effective

micro behaviors in rts game. In *2014 IEEE Conference on Computational Intelligence and Games*, pages 1–8. IEEE, 2014.

[84] Navin K Adhikari, Sushil J Louis, and Siming Liu. Multi-objective cooperative co-evolution of micro for rts games. In *2019 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2019.

[85] P. Sweetser and J. Wiles. Combining influence maps and cellular automata for reactive game agents. *Intelligent Data Engineering and Automated Learning-IDEAL*, pages 209–215, 2005.

[86] M. Preuss, N. Beume, H. Danielsiek, T. Hein, B. Naujoks, N. Piatkowski, R. Stuer, A. Thom, , and S. Wessing. Towards intelligent team composition and maneuvering in real-time strategy games. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(2):82–98, 2010.

[87] D. Doherty and C. ORiordan. Evolving tactical behaviours for teams of agents in single player action games. *9th International Conference on Computer Games: AI, Animation, Mobile, Educational and Serious Games*, pages 121–126, 2006.

[88] A. Uriarte and S. Ontañón. Kiting in RTS games using influence maps. *Eighth Artificial Intelligence and Interactive Digital Entertainment Conference*, pages 31–36, 2012.

[89] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[90] Kamal Taha. Methods that optimize multi-objective problems: A survey and experimental evaluation. *IEEE Access*, 8:80855–80878, 2020.

[91] Yadollah Rasekhipour, Amir Khajepour, Shih-Ken Chen, and Bakhtiar Litkouhi. A potential field-based model predictive path-planning controller for autonomous road vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 18(5):1255–1267, 2016.

[92] Ding Fu-guang, Jiao Peng, Bian Xin-qian, and Wang Hong-Jian. Auv local path planning based on virtual potential field. In *IEEE International Conference Mechatronics and Automation, 2005*, volume 4, pages 1711–1716. IEEE, 2005.

[93] Alexander C Woods and Hung M La. A novel potential field controller for use on aerial robots. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 49(4):665–676, 2017.

[94] Bence Kovács, Géza Szayer, Ferenc Tajti, Mauricio Burdelis, and Péter Korondi. A novel potential field method for path planning of mobile robots by adapting animal motion attributes. *Robotics and Autonomous Systems*, 82:24–34, 2016.

[95] Erick Cantú-Paz. A survey of parallel genetic algorithms. *Calculateurs paralleles, reseaux et systems repartis*, 10(2):141–171, 1998.

[96] DG Reina, Hissam Tawfik, and SL Toral. Multi-subpopulation evolutionary algorithms for coverage deployment of uav-networks. *Ad Hoc Networks*, 68:16–32, 2018.

[97] Raj Jain, Fred Templin, and Kwong-Sang Yin. Analysis of l-band digital aeronautical communication systems: L-dacs1 and l-dacs2. In *2011 Aerospace Conference*, pages 1–10. IEEE, 2011.

[98] Larry J Eshelman. The chc adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination. In *Foundations of genetic algorithms*, volume 1, pages 265–283. Elsevier, 1991.

[99] Sushil J Louis, Tianyi Jiang, and Siming Liu. Real-time strategy game micro for tactical training simulations. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1656–1663. ACM, 2018.

[100] Jeffrey W Herrmann. A genetic algorithm for minimax optimization problems. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, volume 2, pages 1099–1103. IEEE, 1999.

[101] M. Finney. Farsite : Fire area simulator : model development and evaluation. 1998.

[102] Dorothy Albright and BN Meisner. Classification of fire simulation systems. *Fire management notes*, 1999.

[103] G. D. Richards. An elliptical growth model of forest fire fronts and its numerical solution. *International Journal for Numerical Methods in Engineering*, 30:1163–1179, 1990.

[104] Godfried Toussaint. Grids, connectivity, and contour-tracing. *URL:¡ http://www-cgrl. cs. mcgill. ca/˜ godfried/teaching/pr-notes/contour. ps*, 1988.

[105] Michael Alder. An introduction to pattern recognition. *Mike Alder*, 2001.

[106] S.J. Louis and S. Liu. Multi-objective evolution for 3D RTS micro. *Neural and Evolutionary Computing, arXiv:1803.02943*, 2018.

[107] FastEcslent. (2016) Evolutionary computing systems lab, unr. [Online]. Available. `http://ecsl.cse.unr.edu/`.