Multi-objective Evolutionary Algorithms for Distributed Tactical Control of Heterogeneous Agents

Rahul Dubey
University of Nevada Reno
rdubey018@nevada.unr.edu

Sushil J. Louis
University of Nevada Reno
sushil@unr.edu

ABSTRACT

Controlling large numbers of heterogeneous agents in dynamic environments has a number of civilian and defense applications, but presents challenges in cooperation among agents and decision making in uncertain environments. This paper investigates a new problem representation using genetic algorithm tuned potential fields and a new multi-objective problem formulation that evolves distributed control for large numbers of cooperating and competing heterogeneous agents in dynamic environments. Using real-time strategy game-like simulation as a test-bed, results show that the proposed approach scales to controlling a number of and different **types** of agents. Our representation uses influence maps to choose a target and a set of potential fields to control the maneuverability of agents in real-time. We formulated this problem as a multi-objective optimization problem and used an evolutionary multi objective optimization technique to maximize two conflicting objectives in simulation skirmishes. Results indicate that our evolutionary algorithm based representation produces good cooperative behavior and generalized well across groups composed from several different types of agents.

CCS CONCEPTS

• **Computing methodologies** → **Genetic Algorithms**;

KEYWORDS

Tactics evolution, Distributed Control, RTS Games

ACM Reference Format:
Rahul Dubey and Sushil J. Louis. 2021. Multi-objective Evolutionary Algorithms for Distributed Tactical Control of Heterogeneous Agents. In *2021 Genetic and Evolutionary Computation Conference Companion (GECCO '21 Companion), July 10–14, 2021, Lille, France.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3449726.3463201

1 INTRODUCTION

Distributed control of large numbers of heterogeneous agents has many real world applications. Four application areas are defense, search and rescue, self-driving vehicles, and video games. Advancements in distributed control in these domains will lead to better control and execution of given tasks. However, making significant progress for large numbers of agents requires addressing first, the challenge of cooperation among large numbers of agents to complete any given task, and second, decision making in dynamic environments. Cast as a search problem, because of the combinatorial explosion in the number of possible control parameters for different types of agents, a brute force search method to search for the best parameter set in reasonable time becomes unfeasible. Furthermore, environmental and regulatory constraints can further increase the complexity of problems, for example, strict and non-revocable traffic guidelines can significantly increase the level of difficulty for navigation while meeting safety requirements for autonomous vehicles.

In addition to these two problems, the explainability of models or algorithms that control agents in real world applications is another challenge. The need for an explainable model or algorithm arises in designing trustworthy, better human interactable, more transparent agents [8]. Agent trustworthiness is a crucial aspect in real world applications domains such defense and health. Transparency in decision making mechanisms helps to understand the actions taken by agents in given circumstances.

In this work, we present an influence map and potential fields based representation and a new multi objective problem formulation to evolve distributed control for a large number of heterogeneous agents in game-like simulation modeled after the well known StarCraft-II (SC2) Real-Time Strategy (RTS) game. Agents need to cooperate with each other to win skirmishes. The number of possible control parameters among large numbers of agents explodes rapidly, thus we used an Evolutionary Multi Objective Genetic Algorithm (EMOGA) to find a set of pareto optimal solutions. Specifically, in this work, we used our own implementation of the well know Non-dominated Sorting Genetic Algorithm-II (NSGA-II) [5]. Our new problem representation not only helps agents to make real time decisions in a dynamic environments but also provides more easily understood action explanations. This improves the transparency of our proposed representation and problem formulation.

RTS games are a sub-genre of strategy video games and an active area of research [12]. RTS games are war simulation games and can be used as an effective tool to learn long term and short term planning or strategy. The decision making in RTS games can be broadly classified into two categories; macro-management and micro-management. Macro-management or Macro describes long-term strategic planning to manage resource gathering, choosing a location to build new infrastructure, and producing a winning mix of agent types and numbera of agents. The more infrastructure built, the more flexibility players have to build new agents for micro and win combat. Micro-management or Micro is short-term tactical planning to guide agents movement in order to eliminate

GECCO '21 Companion, July 10–14, 2021, Lille, France

ACM ISBN 978-1-4503-8351-6/21/07…$15.00
https://doi.org/10.1145/3449726.3463201

opponent agents and infrastructure while keeping friendly agents undamaged and alive. We can further decompose micro in two sub-categories: tactical and reactive control. Tactical control governs the positioning and movement of a group of agents, whereas reactive control guides a specific agent in order to move, attack or flee. We evolve micro behaviors using EMOGA for different groups of agents controlled through our proposed representation against identical groups of agents controlled through SC2's default AI controller (SC2-AI).

For experiments, we created two identical teams: Own Team (OT) and Enemy Team (ET) in SC2, where OT agents are controlled using our representation and ET agents are controlled though SC2-AI. We created identical teams to ensure that no side has an advantage. Each team is composed of large numbers of different types of agents. Different types of agents have different abilities, strengths, weakness thus good cooperation between agents can produce an effective tactical strategy to win skirmishes and perhaps even win the entire game. We discuss our experimental setup and different scenarios in more detail in Section 3 and we assume that all game information is available for decision making. Our representation uses different Potential Fields (PFs) and Influence Maps (IMs) to guide the movement of large numbers of agents in dynamic environment in order to maximize damage done to opponent agents and minimize damage received by own OT agents.

Earlier work has shown the effectiveness of IMs and PFs [6], [7] to evolve a variety of micro behaviors that maximize damage done to opponents and minimize damage received by friendly agents. This paper explores the same approach with more complex scenarios and more different types of agents with different unique abilities. In order to win, agents with different abilities need to learn how to exploit the strengths and weakness of other agents. We conducted experiments by making a variety of combinations of different types of agents to combat against similar combinations of agents. The experimental results show that agents controlled through our approach coordinated well together and ET agents controlled by the hard coded SC2-AI.

The remainder of this paper is organized as follows: Section 2, describes prior work in generating micro behaviors using different approaches. Section 3, describes the experimental setup, and training and testing scenarios in detail. Subsequently, Section 4, describes our representation and fitness evaluation. In Section 5, we show and discuss experimental results. Lastly, in Section 6, we draw conclusion and present possible future work.

2 RELATED WORK

Many researchers have proposed and investigated different approaches for distributed control of multiple agents. These include rule based techniques, tree search, reinforcement learning (RL) [16], and neuro-evolution of augmenting topologies (NEAT) [9]. In rule based techniques, an agent execute an action form a pre-defined set of actions for a given input/situation, but is unable to perform well in other situations for which actions are not pre-defined. In tree search, an agent searches for a solution in the search space for a given situation. The size of search space depends on the complexity of problems, for example, the search space for the game of chess is on the order of 10^{50} and for RTS games $10^{50^{5000}}$ [12]. Searching for a solution over a large search space may violate an agent's allowed time constraints. This limits the use of tree search based methods to problems with low or moderate search spaces. In order to solve complex problems with large search spaces, researchers paid much attention to reinforcement learning based approaches, where an agent interacts with the given environment and learns through the process of trial and error. However, when using Neural Networks (deep or shallow) to represent policy, it is difficult to explain the learned behaviors of agents [8, 15]. Similarly NEAT, as the name suggests, uses evolutionary algorithms to evolve both the structure and weights of neural networks. Again, NEAT control strategies are difficult to explain.

RTS games can be considered simulation games where a player needs to develop long term and short term strategies and tactics to win against opponents. Long term strategy involves infrastructure build order planning, managing resources, and scouting to find out an opponent's infrastructure and army composition and to predict opponent strategy [13]. We are specifically interested in short term planning or tactics to win skirmishes against opponents. Short term planning or micro governs the movement of a group of agents during skirmishes. Many researchers have used influence maps, potential fields [2], neuro-evolution [17], and reinforcement learning based algorithms [16] to generate micro behaviors in different RTS games. Churchill presented an algorithm called Alpha-Beta Considering Duration (ABCD) to control upto eight agents in RTS games [4]. Chung applied a monte carlo planning technique to guide agents in ORTS; a simplified version of the StarCraft-Broodwar [3] RTS game. Balla used a monte carlo tree search algorithm for tactical assault planning in Wargus; another research RTS game [1]. Gabriel presented a Bayesian model to generate strategies and control a group of agents and their positioning [18].

In RTS games, badly controlled agents may lead to losses when superior in numbers. 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 [14]. Early work in our lab used influence maps encoding enemy spatial information to evolve complete RTS game players [11]. In [7] we compared three different approaches to generate micro behaviors. These three approaches are NEAT, a meta-search (hand-coded), and potential field based approach to generate micro. The next section introduces our new RTS game environment in more detail.

3 PROBLEM FORMULATION AND SIMULATION

We aim to evolve micro/short term tactics to win skirmishes against an intelligent opponent. The problem is formulated as a multi objective optimization problem to maximize damage done to opponents and minimize damage received by friendly agents. For experiments, two identical teams were generated on the game map. Each agent has a starting location, health, and is classified as ground or aerial and as ranged or melee type. An agent with a large firing range is termed as ranged agent and an agent with smaller firing range is termed as melee agent. Assume that both teams have N agents of different types where the k^{th} agent is represented by the tuple $\{P_k, H_k, Attack_k, R/M, F/E\}$. Here P_k represents 2D position and H_k represents health. $Attack_k$ represents whether the k^{th} agent is

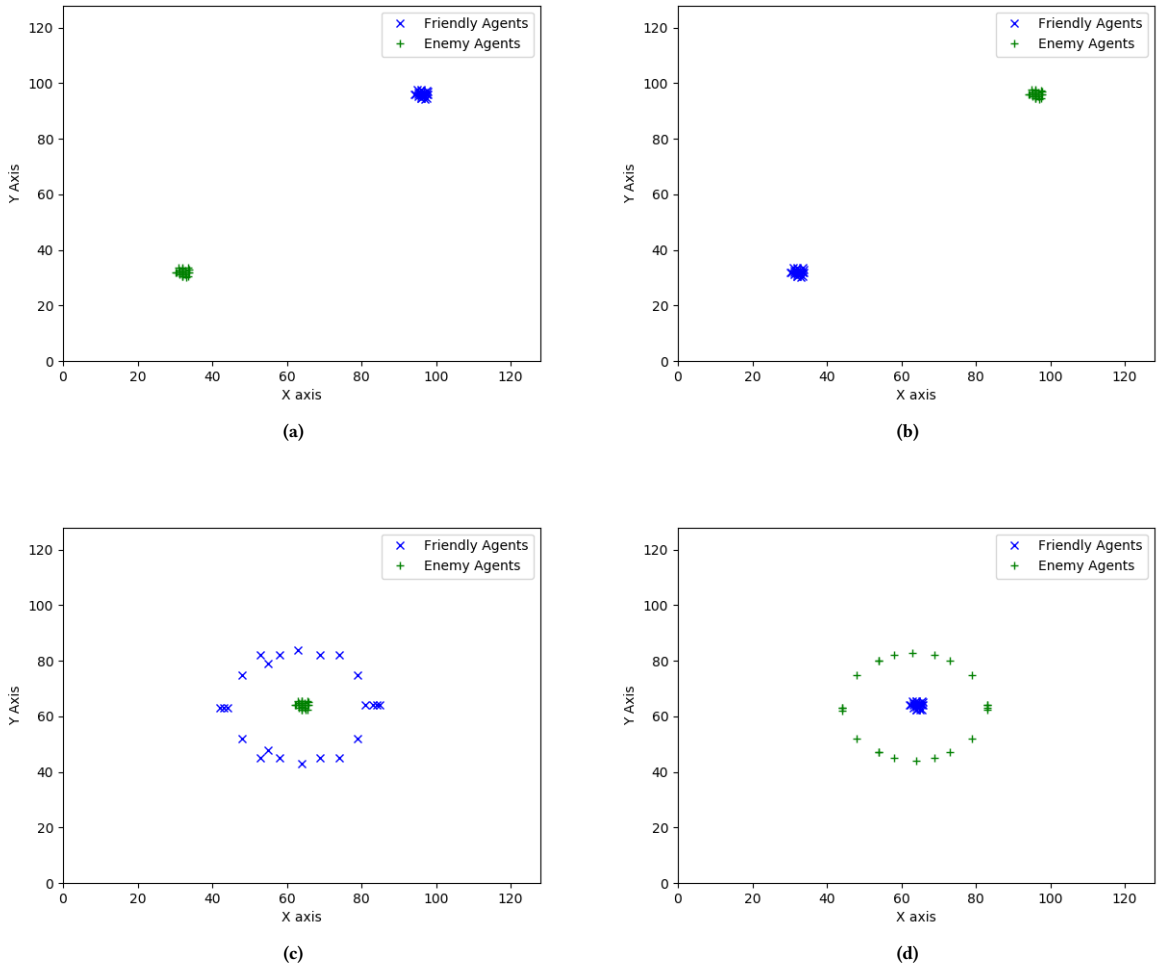


Figure 1: Training scenarios showing positioning of agent teams.

a ground or aerial agent, R/M represents the type of agent, ranged (R) or melee (M). Finally, F/E represents whether the k^{th} agent is friendly or enemy agent. Table 1 shows agent's attributes.

3.1 Problem

We aim to evolve micro behaviors or short term tactics using our proposed evolutionary distributed control approach to eliminate enemy agents by inflicting more damage while keeping friendly agents unharmed and alive by avoiding damage received. In order to inflict heavy damage to a team of opponent agents, friendly agents may need to be aggressive which may also lead to lower health. In contrast, when friendly agents concentrate on avoiding damage received, they tend to flee and inflict less damage to opponents. Thus balanced tactics that fight well need to tradeoff damage done with damage received. We aim to evolve tactics that maximize damage done to opponents and minimize damage received by friendly agents. Thus, we formulated this problem as a multi

Table 1: Attributes and characteristics of Agents. Agent types and attributes are taken from Starcraft-II.

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

objective optimization problem. The two objectives are computed using equation 1 and 2.

$$\text{Maximize} f = \left[1 - \frac{\sum_{k=1}^{N} H_{ek}}{MaxHealth}, \frac{\sum_{k=1}^{N} H_{fk}}{MaxHealth} \right] \quad (1)$$

$$\text{MaxHealth} = \sum_{k=0}^{N} InitialHealth_k \quad (2)$$

The first term of equation 1 computes the damage done to opponent's agent. Here H_{ek} represents remaining health of k^{th} enemy agent. We normalized remaining health with respect to initial health. Thus the first term represents the percentage damage done to opponents. Similarly, the second term computes the percentage remaining energy of friendly agents at the end of skirmishes. We aim to maximize both objectives using a well known multi objective optimization technique, NSGA-II. Earlier research [6],[10] has shown that solutions evolved on a specific scenario may be work well on other unseen scenarios. In other words, the evolved solutions may not be robust. To mitigate this issue we train agents over four different *training* scenarios and test the quality and robustness of evolved micro over two previously unseen *test* scenarios.

3.2 Scenarios

In skirmishes, human players use different micro-behaviors and strategies depending on opponent's types, numbers, and positioning of game agents. For example, a ranged agent can do effective kiting (hit and run) against a melee agent. For a group composed of ranged and melee agents, good tactics may become more complex. One preferred strategy can be a combination of focus attack and kiting. Thus in order to evolve complex micro behaviors, we created identical teams composed from different types of game agents to train on four different scenarios, here different scenarios refers to different starting position of agents on the game map. Figure 1 shows the initial positioning of agents in these four different scenarios in SC2.

We are interested in the evolution of a good quality robust tactics, thus we use four different training scenarios. However, to measure the generalizability of evolved tactics, we test the evolved micro over two unseen scenarios shown in Figure 6 that are significantly different in initial agent positioning from our testing scenarios.

4 METHODOLOGY

In this work, our objective is to generate a set of micro behaviors for large numbers of heterogeneous agents. This section explains our representation to generate micro behaviors, first we explain influence maps and then potential fields.

4.1 Influence Maps

Spatial information in RTS games plays a significant role in decision making. Influence Maps (IMs) are commonly used to represent spatial information such as enemy's position and terrain spatial information in games. An IM is a grid composed of many cell with each cell value computed based on enemies locations and health. To calculate a grid-cell value, we add the influence of all agents within

0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	40	40	40	0	0	0	0	0
0	0	0	0	0	40	80^{1}	40	0	0	0	0	0
0	0	0	40	40	80	40	80	40	40	0	0	0
0	0	0	40	80^{8}	40	0	40	80^{2}	40	0	0	0
0	40	40	80	40	40	0	40	40	80	40	40	0
0	40	80^{7}	40	0	0	0	0	0	40	80^{3}	40	0
0	40	40	40	0	0	0	0	0	40	40	40	0
0	0	0	40	40	40	0	40	40	40	0	0	0
0	0	0	40	80^{6}	40	0	40	80^{4}	40	0	0	0
0	0	0	40	40	80	40	80	40	40	0	0	0
0	0	0	0	0	40	80^{5}	40	0	0	0	0	0
0	0	0	0	0	40	40	40	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2: Influence map cell values, yellow colored cell is agent position

a range (r) from the cell. The influence of each agent decreases linearly by a fraction (δ) to zero as range increases. If a cell is influenced by more than one agent, the cell's value is the sum of all influences. Equation 3 specifies the equation that determines each agent's influence at the grid cell occupied by the agent.

$$IM_s = w_1 P_h + w_2 \quad (3)$$

$$I_d = IM_s * \delta \quad (4)$$

$$G_c = \sum_{i \in r} (IM_s - d_i I_d) \quad (5)$$

Here IM_s is an agent's influence, P_h is the health of the agent, and d_i is the distance from the agent. As explained earlier, the influence of an agent decreases by a fraction as range increases. Equation 4 computes this decrease. δ is the fraction to reduce influence for every agent of range upto a range of r. Equation 5 then gives the value for an influence map grid cell. Here w_1, w_2, δ, and r are tunable influence map parameters. A lower cell value represents weak enemy influence and high cell values represents strong enemy influence. We choose the lowest cell value as the target attack location towards which to move. When multiple cells have the same lowest value, the closest minimal value cell from our agents is chosen as the target. Figure 2 shows an example IM generated by the above equations for 8 agents (Zealots) with IM parameter values of $w_1 = 1$, $w_2 = 0$, $\delta = 0.5$, $r = 2$ and $P_h = 80$. Substituting these values in equation 3, we get the starting IM value of 80 at the starting position of agents as shown in yellow colored cells in figure 2. As mentioned earlier, the influence of an agent decrease as the distance (d_i) between the agent's starting location and a grid-cell increase. In figure 2, the influence decreases by 50% per unit length of distance and thus cells around the agent's starting location has 50% less value of the starting influence as shown by brown colored cells, and when distance is 2 the influence reduced to zero as shown by blue colored cells. There are several cells where more than one agent has influence and when computing the influence value of these cells, we add the influence of each agent. In figure 2, the red colored cells represent those cells that combine influence of two agents. There are many cells with the value of 40 and a friendly agent choose a cell, closest to it, as the target location to attack.

In this work, we consider agents with different abilities and thus each different type of agent maneuvers and is best used differently. An agent with large firing range can attack from a larger distance and run away to avoid damage received. This is known as kiting. But, an agent with a smaller firing range needs to go close to opponent agents to inflict damage. Thus, for each different types of agents, we tune different set of influence map parameters $\{w_1, w_2, \delta, r\}$. With different sets of parameters, agents with different abilities can choose different target locations to attack. Hence for m different types of agents on a team, we tune $4m$ influence map parameters.

4.2 Potential Fields

We aim to control the movement of friendly game agents to eliminate opponent agents and win skirmishes. To guide agents in real time, we use a set of potential fields based on distance, health, and the target location as shown in equation 6. Here $\vec{PF}$ is the resultant potential field, $\vec{PF_d}$, $\vec{PF_h}$ are potential fields generated using distance and health, and $\vec{PF_t}$ is distance dependent potential field generated by the target.

$$\vec{PF} = \vec{PF_d} + \vec{PF_h} + \vec{PF_t} \tag{6}$$

$$\vec{PF_d} = \vec{PF_{adf}} + \vec{PF_{rdf}} + \vec{PF_{ade}} + \vec{PF_{rde}} \tag{7}$$

$$\vec{PF_h} = \vec{PF_{ahf}} + \vec{PF_{rhf}} + \vec{PF_{ahe}} + \vec{PF_{rhe}} \tag{8}$$

In equation 7, $\vec{PF_{adf}}, \vec{PF_{rdf}}$ are attractive and repulsive potential fields based on distance from friendly agents, $\vec{PF_{ade}}, \vec{PF_{rde}}$ are attractive and repulsive potential fields based on distance from enemy agents. Similarly, $\vec{PF_{ahf}}, \vec{PF_{rhf}}$ are attractive and repulsive potential fields based on health of friendly agents, $\vec{PF_{ahe}}, \vec{PF_{rhe}}$ are attractive and repulsive potential fields based on health of enemy agents. The resultant potential field, PF, controls the agent's desired heading. Each potential field has two tunable parameters [10], a coefficient (c) and an exponent (e) Parameter ranges are as follows: $c \in \{-10000, 10000\}$ and $e \in \{-7, 8\}$. The influence map parameters w_2 and $r \in \{0, 8\}$, and w_1 and $\delta \in \{0, 1\}$. Equation 9 gives a generalized expression for the number of parameters required for m different types of agents on each side, where P_{num} represents the number of parameters. q represents IM and target location parameters $\{w_1, w_2, \delta, r, c_t, e_t\}$ where c_t and e_t are target potential field coefficient and exponents respectively. The second part of Equation 9, deals with symmetry reduction. The number of parameters using equation 9 grows as a second order polynomial.

$$Pnum = 2 + (q + 8 \times 2 \times m)m - \sum_{i \in m} 4(i-1) \tag{9}$$

4.3 Fitness Evaluation

Our objectives are maximizing damage done to the opponent and minimizing the damage received by friendly agents for a group composed of heterogeneous agents. We use an evolutionary multi-objective optimization approach to evolve a diverse pareto front. We normalize damage done and damage received to span the range [0..1] as shown in equation 1. Algorithm 1 computes averaged fitness over MaxScenarios (MS) for each Candidate Solution (CS). For each scenario, we ran our game-like simulation for a fixed number of time steps ($timeSteps$) and computed objective values at the end of simulation. For each time step, we compute a target location for each friendly agent and compute the desired heading based on potential fields. Each agent moves in the direction of desired heading with a fixed speed. Algorithm 1 finally returns an averaged objective value over a given number of scenarios.

Algorithm 1: Fitness Computation

```
Input  : CS
Output : fitness
obj1 = obj2 = 0;
for scenario in MS do
    timeSteps = 0;
    while timeSteps<MT do
        InfluenceMap();
        Headings = ComputePotentialFields(CS);
        MoveAll(Headings);
        timeSteps++;
    end
    obj1 += DamageDone();
    obj2 += DamageReceived();
end
obj1 = obj1 / MS;
obj2 = obj2 / MS;
fitness = [obj1, obj2];
return(fitness);
```

5 RESULTS AND DISCUSSION

We created identical teams to eliminate the issue of bias and evolved pareto fronts of micro behaviors on four different training scenarios. These pareto fronts include a diversity of candidate solutions where each candidate solution has objective values in terms of damage done and 1 - damage received averaged over four 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 2 and the attributes of each of the agents is shown in Table 1. We next discuss the evolved micro behaviors/tactics for each of our three team configurations.

5.1 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

Table 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_l, 4 M_u

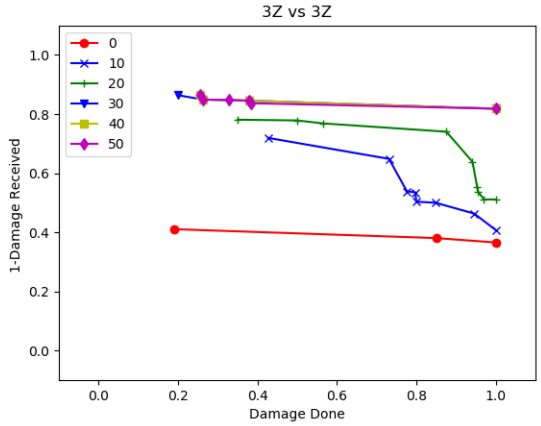


Figure 3: Pareto fronts at every 10^{th} generation. Each team is composed from three different types of agents.

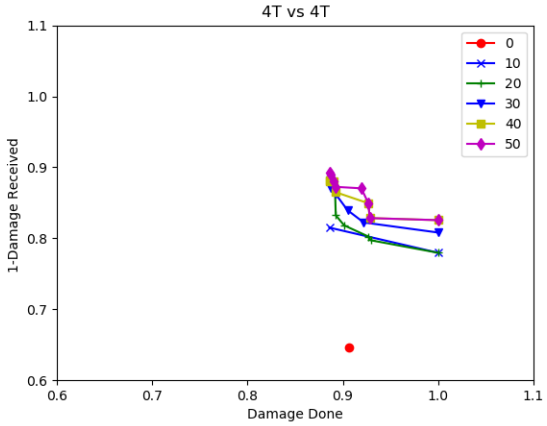


Figure 4: Pareto fronts at every 10^{th} generation. Each team is composed from four different types of agents.

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 3 shows the evolution of micro behavior for every 10^{th} generation and we observed a significant difference between the first (0^{th}) and the last (50^{th}) generation pareto front solutions. A fitness $(1, 0.81)$ specifies that all ET agents died and OT agents had 81% remaining health at the end of skirmish. In figure 3, the last generation pareto front solutions shows a range of behaviors from fleeing to aggressive. Own team agents were able to eliminate all enemy team agents while receiving little damage $(1, 0.81)$.

5.2 Four Different Types of Agents

Figure 4 shows pareto fronts when playing with OT terran agents against identical ET 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 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 fig 4, terran agents improved their performance significantly from initial generation to last generation in terms of damage done and damage received. OT terran agents learned effective strategies to counter the swarm of ET terran agents by figuring out weakness in their strategy. B_n of OT eliminates the M_r of ET and remember in this team setting, M_r is the only agent that can attack aerial targets.

5.3 Five Different Types of Agents

Table 2 shows the team combination with 27 agents from five different types of protoss race agents. Figure 5 shows the pareto fronts at every 10^{th} generation when OT protoss agents play against an identical ET 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 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 ETs. 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 5, initially, in the 0^{th} generation our agents performed poorly but gradually improve over a period of 50 generations. The last generation pareto front (50^{th}) shows a variety of micro including fleeing, balanced, and aggressive.

5.4 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. 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 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 of our proposed evolution distributed control approach to evolve high performing diverse tactics against opponents.

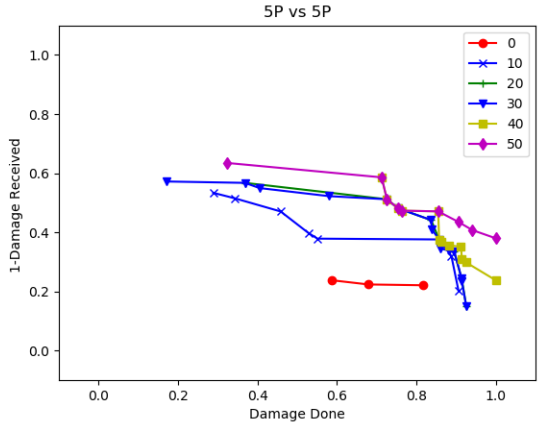


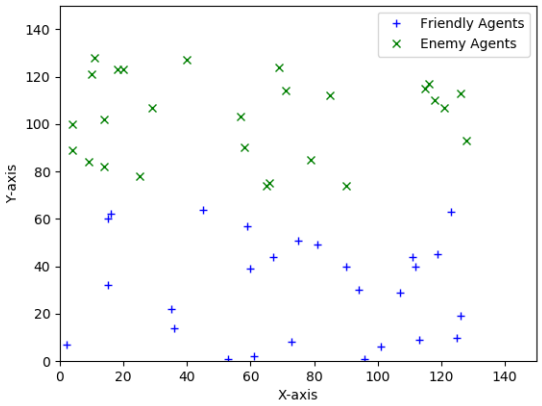
Figure 5: Pareto fronts of every 10^{th} generation. Each team composed from five different types of agents.

6 CONCLUSION

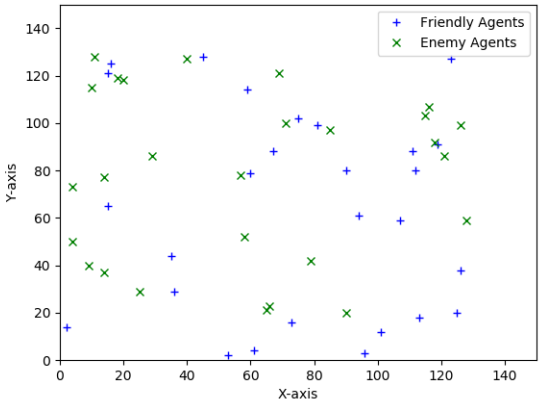
Using an RTS game as a test-bed, we evolved tactics for groups composed from upto five different types of units and up to 27 total numbers of units using NSGA-II. Our approach uses influence maps and a set of potential fields to guide agents in dynamic environment in real-time. We formulated this problem as a multi objective optimization problem and used NSGA-II to tune potential fields and influence map parameters. The results shows that our evolutionary multi-objective optimization based approach evolves different strategies by tuning IMs and potential field parameters. Our approach is simple to implement, understand and thus enables human operators to better understand actions taken by an agent. This has potential to make overall human machine cooperation effective and safer.

ACKNOWLEDGMENTS

This work was supported by grant number N00014-17- 1-2558 from the Office of Naval Research. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Office of Naval Research. This work was also supported in part by the U.S. Department of Transportation, Office of the Assistant Secretary for Research and Technology (USDOT/OST-R) 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 publication are solely those of the authors and do not represent the official policy or position of the USDOT/OST-R, or any State or other entity.



(a)



(b)

Figure 6: Two testing scenarios with a group composed of different types of agents against the same group of agents

REFERENCES

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

[2] S. Liu; S. Louis; C. Ballinger. 2016. Evolving Effective Micro Behaviors in Real-Time Strategy Games. *IEEE Transactions on Computational Intelligence and AI in Games* 8, 4 (2016), 351–362.

[3] Michael Chung, Michael Buro, and Jonathan Schaeffer. 2005. Monte Carlo Planning in RTS Games.. In *CIG*. Citeseer.

[4] David Churchill and Michael Buro. 2011. Build Order Optimization in StarCraft. In *Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2011)*.

[5] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA- II. *IEEE Transactions on Evolutionary Computation* 6, 2 (2002), 182–197.

[6] Rahul Dubey, Joseph Ghantous, Sushil Louis, and Siming Liu. 2018. Evolutionary Multi-objective Optimization of Real-Time Strategy Micro. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 1–8.

[7] Rahul Dubey, Sushil Louis, Aavaas Gajurel, and Siming Liu. 2019. Comparing Three Approaches to Micro in RTS Games. In *2019 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 777–784.

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

[9] Aavaas Gajurel ; Sushil J Louis ; Daniel J Méndez ; Siming Liu. 2018. Neuroevolution for RTS Micro. *IEEE Conference on Computational Intelligence and Games* (2018), 1–8.

[10] Sushil J Louis; Siming Liu. 2018. Multi-Objective Evolution for 3D RTS Micro. *IEEE Congress on Evolutionary Computation (CEC)* (2018), 1–8.

[11] C. Miles, J. Quiroz R. Leigh, and S. Louis. 2007. Co-evolving influence map tree based strategy game players. *IEEE Symposium on Computational Intelligence and*

Games (2007), 88–95.

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

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

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

[15] Wojciech Samek and Klaus-Robert Müller. 2019. Towards Explainable Artificial Intelligence. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Springer, 5–22.

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

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

[18] SGabriel Synnaeve and Pierre Bessière. 2016. Multiscale Bayesian Modeling for RTS Games: An Application to StarCraft AI. *IEEE Transactions on Computational Intelligence and AI in Games* 8, 4 (2016), 338–350.