Evolutionary Inversion of Swarm Emergence Using Disjunctive Combs Control

Winston Ewert, Daniel Jepson & Robert J. Marks II
Dept. of Electrical & Computer Engineering
Baylor University
Waco, Texas

Benjamin B. Thompson
Pennsylvania State University
Applied Research Laboratory
The Pennsylvania State University
State College, PA

Albert Yu
University of Washington
Dept. of Electrical Engineering
Seattle, Washington

***Abstract*—Given simple agent rules, a swarm's emergent behavior can be difficult to predict. The inverse problem is even more difficult: given a desired emergent behavior, what are the rules by which swarm agents should operate? Disjunctive fuzzy control is proposed as a method to model swarm agents. Compared to more commonly used conjunctive fuzzy control such as that proposed by Mamdani, disjunctive fuzzy control is robustly fault tolerant and disjointly connected. Swarms are inherently disjunctive. Instead of agents working in coordination with one another, each swarm agent contributes individually to the result. The disjunctive attribute can also be applied at the sensor level for each individual agent. Disjunctive control allows adaptation of the describing membership function, as is commonly done in conjunctive control. The inversion process is illustrated with numerous simulation examples, including a predator-prey game, gang warfare and escaping agents. The swarm is instructed what to do, but not how to do it. Imposition of fitness constraints and repeated generations of evolutionary molding of agent performance can then result in unexpected emergent behaviors of the swarm, *e.g.* use of decoys, self sacrifice, flanking maneuvers, and shielding of the weak.**

***Index Terms*—Keywords: swarm intelligence, fuzzy control, disjunctive control, inverse problem, emergent behavior**

I. Introduction

Swarm intelligence is based on the emergent behavior of groups of individual social agents performing simple tasks. Certain insects [11] and bacteria [60] are examples. Swarm intelligence has found application is telecommunications [19], [26], business [12], robotics [9], [29], and optimization [20], [21], and makes use of a plurality of highly disjoint agents interacting using simple rules. Simple swarm algorithms have been employed to assist with load balancing of peer-to-peer networks [48], routing within mobile ad hoc radio networks [28], and self-organizing construction and assembly [42].

Simulations of swarm algorithms have parameters that can be tuned [69]. Pioneering work has focused on enhancement of the emergent behavior for which the swarm is designed [1], [2], [8], [10], [25], [30], [31], [49], [52]. We focus on a more general problem of evolving unspecified emergent behavior based on goal without regard to the manner success is achieved. Results are often unexpected. In one case, for example, sacrificial agents were evolved that act as decoys to distract predators with the goal of maximizing the lifetime of the swarm collective. In another, an evolved swarm developed deceptive flanking tactics to avoid capture. Some swarms maximize their life span by being aggressive towards the enemy. Others swarms extend their lifetime by prolonged strategic retreat.

Often, determination of emergent behavior from simple rules of interaction in swarm intelligence escapes both analytic and intuitive inspection [27]. Here are some examples from the literature.

1) Each agent randomly roams on a floor covered with wood particles, picking up particles if it bumps into one. When an agent bumps into a second particle it unloads its load. Now empty handed, it continues roaming looking for another particle and the process is repeated.[11]
2) Each agent randomly identifies two other agents and moves to place itself between two agents [12].
3) Each agent randomly identifies two other agents, and tries to move such that one agent, a protector, is between it and the other agents, an aggressor.

The rules in these are expressed clearly and without ambiguity. The identification of the emergent behavior of the swarm, however, is not readily evident.[1]

These three simple examples of the forward swarm problem illustrate the difficulty of the analysis of emergent behavior in even simple swarms.[2] The inversion of the swarm, or swarm design, is even more daunting. Given a desired emergent behavior, what are the set of simple rules needed? We investigate such a design applied to a plurality of cases.

This paper has two distinct parts. The first provides a short review of disjunctive Combs control in comparison to the more widely used conjunctive Mamdani control. Combs control is shown to be more effective in the swarm inversion process largely because the corresponding search dimension is reduced. We demonstrate that Combs control is equivalent to use of actuator functions. The manipulation of these actuator function effects agent actions and consequently the emergent behavior of the swarm. The second part of the paper presents a number of swarms evolved using Combs control. The emer-

[1]Once an emergent behavior is identified, however, the relationship between the rules and the emergent behavior can become more clear.

[2]The emergent behavior for these three cases is in the Appendix (Section VI.

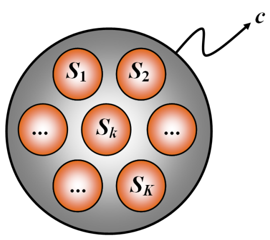

Fig. 1. The pth agent in a swarm team of P agents. As shown here, each agent has K sensors $\{S_k|1 \leq k \leq K\}$. The kth sensor makes a reading of s_k which is subjected to the actuator function in (4) to generate the sensor consequent c_k. The c_k's from all sensors are aggregated to generate the scalar consequent, c, for the agent. When there more than a single consequent, we denote them as $c[1]$, $c[2]$, etc. In the examples in this paper, each agent has two consequents: one for each of two dimensions of movement. Each consequent can have a separate set of actuator functions.

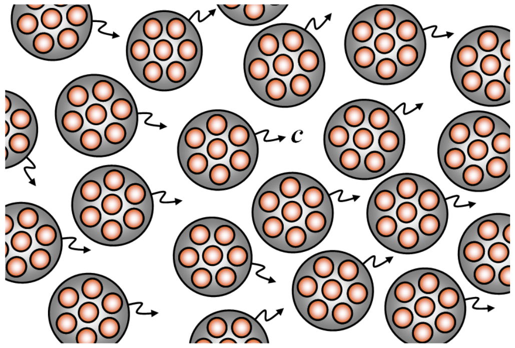

Fig. 2. A collection of P agents of the type shown in Figure 1 form a swarm. We consider only the case where the control for all agents is identical. Each agent has a consequent in accordance to its sensor readings. Acting together, the swarm has a goal to achieve as measured by a fitness measurable only after the swarm operates. Through evolutionary inversion of actuator functions, the swarm can be made to perform better and better. The manner in which the swarm achieves its global objective consequent ($\mathbb{C}$) through emergent behavior can be unexpected.

gent behaviors resulting from the swarm inversion are often unexpected yet are effective and, upon reflection, reasonable.

II. Swarm Formulation and Control

We now make the case that evolutionary determination of disjunctive fuzzy logic parameters [16], [17], [65], [66], [67], [68] is ideally suited for evolving the emergent behavior of swarms.

There are two levels of control in the swarm.

1) *Agent Control.* There are P swarm agents on a team. One is shown in Figure 1. Each agent has K sensors $\{S_k|1 \leq k \leq K\}$. For our simulations, a homogeneous swarm is used so that each agent has the same resident control rules. Fuzzy logic is used to control each agent. Each sensor provides antecedents to the fuzzy control of each individual agent. A swarm of P agents is illustrated in Figure 2.
2) *Global Swarm Control.* For the second level of the swarm control, the consequents of each agent action provide the emergent behavior, $\mathbb{C}$, of the swarm. The individual agent actions are aggregated into the overall swarm performance which is measured by a fitness function. For a swarm of prey, for example, $\mathbb{C}$ might denote the median survival time of all agents, *i.e.* the time it takes for half of the swarm to be destroyed. In the common scenario of randomness within the swarm, including initialization and agent jitter, $\mathbb{C}$ will be a random variable whose character can be assessed only through repeated trials [69].

III. Conjunctive Versus Disjunctive Fuzzy Control

For each swarm agent consequent, traditional Mamdani conjunctive implication can be expressed as

$$\bigcap_{k=1}^{K} A_k \to C. \tag{1}$$

where A_k is the kth fuzzy descriptor of the fuzzification of the nth sensor and C the fuzzy consequence.

For example, consider automobile control and let C = "turn slightly right." Mamdani control rules then take on the following form.

$$\begin{array}{lll} \text{If} & (A_1 = \textit{turn front tires slightly right} & \text{AND} \\ & \ \ A_2 = \textit{turn rear tires slightly left} & \text{AND} \\ & \ \ A_3 = \textit{lightly brake right tires} & \text{AND} \\ & \ \ A_4 = \textit{slightly accelerate left tires}) & \\ & \qquad \text{THEN } C & \end{array}$$

Disjunctive implication used in Combs [15] control[3] is

$$\bigcup_{k,n} (A_k \to C) \tag{2}$$

Here, the contribution of each sensor to the agent's performance is aggregated to assess the resultant consequent. The corresponding Combs control rules in the car turning example are:

$$\begin{array}{lll} (\text{If } A_1 = \textit{turn front tires slightly right} & \text{THEN } C) & \text{OR} \\ (\text{If } A_2 = \textit{turn rear tires slightly left} & \text{THEN } C) & \text{OR} \\ (\text{If } A_3 = \textit{lightly brake right tires} & \text{THEN } C) & \text{OR} \\ (\text{If } A_4 = \textit{slightly accelerate left tires} & \text{THEN } C) & \end{array}$$

In propositional Boolean logic,[4] there is an identity between the disjunctive and conjunctive implications dubbed the *law of importation* [36]:

$$\left(\bigcap_{k=1}^{K} A_k \to C \right) \equiv \bigcup_{k=1}^{K} (A_k \to C). \tag{3}$$

The fuzzy logic generalization of the law of importation is not an identity. There is, however, often commensurate

[3]Application of disjunctive implication to fuzzy inference is commonly called the *Combs method* [24], [34], [36], [61].

[4]For implication, $0 \to 0$ is 1, $0 \to 1$ is 1, $1 \to 0$ is 0 and $1 \to 1$ is 1.

performance in comparison to the use of fuzzy conjunctive Mamdani rule matrices [36], [67].

Details contrasting the characteristics of conjunctive and disjunctive implications can be found elsewhere [16], [17], [65], [66], [67], [68]. Combs control has many advantages. If sensors are lost, for example, an agent can straightforwardly adapt by seamlessly applying redundant resources. Likewise, new sensors can be easily added. The conjunctive form is brittle in comparison. Loss of a sensors requires reassessment of the implication (Mamdani rule matrix) structure.

If there are K sensors and a single consequent each requiring $\{N_k | 1 \leq k \leq K\}$ fuzzy sets, the Mamdani fuzzy rule matrix for implementation requires

$$N_\cap = \prod_{k=1}^{K} N_k$$

fuzzy rules. If all of the $N_k = N$ for all k, then $N_\cap = N^K$. Disjunctive control, on the other hand, requires

$$N_\cup = \sum_{k=1}^{K} N_k$$

rules [16], [17], [65], [66], [67], [68]. If the number of fuzzy sets is the same, then $N_\cup = NK$. The number of fuzzy rules therefore increases linearly with respect to the number of antecedents rather than exponentially.[5]

In the inversion of the swarm, we will be searching through a space whose dimension is determined by the number of fuzzy rules. Therefore, besides its operational advantages, disjunctive control reduces the search space size thereby avoiding the *curse of dimensionality* [56] for the swarm inversion process.

A. Functional representation.

We now show that fuzzy Combs control can be reduced to use of actuator functions for each sensor followed by aggregation.

An example of kth sensor contributes to the agent consequent $A_k \to \mathcal{C}$. If the kth sensor is tesselated into N_k fuzzy membership functions, any scalar measurement, say s_k, will be fuzzified into a vector of N_k membership values. For disjunctive control, all of the elements in this vector are used to weight the fuzzy membership functions of the single consequent which is then defuzzified into a single crisp consequent scalar for the kth sensor, c_k. From this process, we conclude that each sensor reading is assigned a single consequent value which, in turn, is aggregated with the consequents of the other sensors to specify the composite consequent for the pth agent. Thus, *the value of the consequent is a simple one dimensional function of the sensor antecedent* which we can write as

$$c_k = \zeta_k(s_k) \tag{4}$$

[5]A common observation is that conjunctive control with more rules and therefore more degrees of freedom has greater fine grained flexibility than disjunctive control.

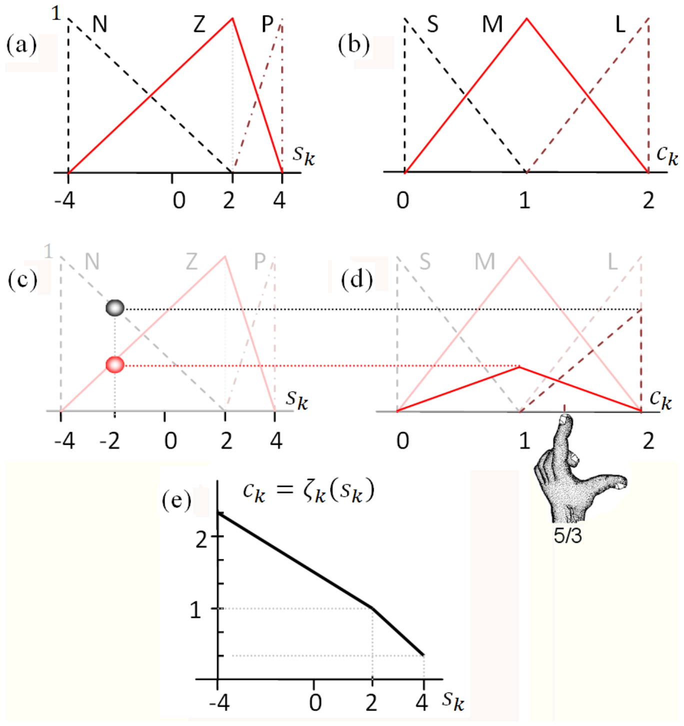

Fig. 3. Illustration of the actuator from fuzzy rules is a one dimensional equivalent of a fuzzy control surface. The antecedent and consequent fuzzy membership shown respectively in (a) and (b) follow the rules "N → L AND Z → M AND P → S." For a sensor reading of $s_k = -2$, the antecedent fuzzy membership functions for (N, Z, P) read $(\frac{2}{3}, \frac{1}{3}, 0)$. This is illustrated in (c). To defuzzify, the consequent membership functions are weighted as shown in (d) and added. The center of mass balance point of the sum of the weighted consequent functions, as shown in (d), is $c_k = \frac{5}{3}$. Thus $\zeta_k(-2) = \frac{5}{3}$. Repeating this process for all $s_k \in [-4, 4]$ generates the piecewise linear actuator function ζ_k shown in (e). To our knowledge, this is the first time Combs control has been shown to equate to use of an actuator function.

The *actuator functions*, $\zeta_k(\cdot)$, is the one dimensional equivalent to the control surfaces in fuzzy inference systems [14], [35]. A detailed example is shown in Figure 3.

Fuzzification and defuzzification of membership functions for disjunctive fuzzy control has been performed in previous treatments [16], [17], [65], [66], [67], [68]. As illustrated in Figure 3, however, formation and adaptation of actuator functions can be performed directly without consideration of the intermediate fuzzy components.

In summary, sensor S_k on an agent measures s_k from which we find the consequents, c_k, using the actuator function in (4). The c_k's are then combined using, for example, an aggregate function [13], into each agent's consequent, c.

Each agent may have more than one consequent in which case $c_k[1]$, $c_k[2]$, etc. are generated using possibly different actuator functions acting on the same sensor inputs. Each is aggregated into consequents $c[1]$, $c[2]$, etc. for each agent. The interaction of all agent actions then contributes to the emergent behavior of the swarm.

The inverse problem [7], which has found use in many areas of computational intelligence [32], [33], [37], [38], [57], therefore reduces to adapting the actuator functions to maximize the fitness of the emergent behavior, $\mathbb{C}$. This is a variation of fuzzy membership function adaptation applied in Mamdani type inference systems [3], [4], [5], [45] and similar to inversion of trained neural networks . Use of Combs control, however, can reduce the search space dimension significantly.

IV. Inversion Examples

Using Combs control of agents, here are some examples of emergent behaviors observed from inversion of competitive swarms. Results will be illustrated with figures, but videos of the swarm performance are more instructive. They are available on line [50]. Each example has some variations as to engagement rules; however, the nature of the disjunctive control is the same in all of the cases. Elucidation of fine grained details in each program is beyond the scope and length constraints of this paper. Details, though, can be found in documented code also available online [50].

For the examples to follow, all swarm agents on a team contain the same sensors, actuator functions and control rules. A performance fitness function for the swarm is chosen and the swarm is run. Since there are stochastic components in every simulation (*e.g.* initialization and jitter), the fitness is a random variable and each run of the simulation generates a single sample of the underlying random variable. Running a swarm from start to finish[6] results in a single stochastic fitness value. Using the measured fitness of a number of simulations of swarms with different actuator functions, personal and global best results were noted and, after updating the actuator functions consistent with *particle swarm* search [22], [39], [40], [41], [58], [59], another generation of swarm performance is assessed in another generation of swarms. Application of repeated generations resulted in emerged behavior that displays a high fitness. We refer to numerous evolutionary steps in the optimization of a swarm team an *era*.

To address the stochastic contribution of swarm interaction, each swarm was run for 20 iterations and the measured fitness of these swarms are averaged to estimate the mean of the underlying random variable. The particle swarm uses a population of 80 agents and was run for 10 iterations at a time. The parameters of particular swarm optimization, CG and CP, were both set to 2.0.

All swarm agents exist on a two-dimensional square planar *playground* using floating-point coordinates between 1 and -1 for both axes. All agents must remain within the playground at all times. Directions are kept on a trajectory by bounding increments in velocity change. At each time step, a velocity is added to the current position. Additionally, a small amount of random jitter [51], [53], [54], [55] is added to the current position. A number of consequents are assigned to each agent in a manner specific to each game.

For each sensor, an adaptive actuator function $c_k = \zeta(a_k)$ is formed for each of the consequents. The shape of $\zeta(a_k)$ is defined by three values of s, at the points 0, 1, and 2.

[6]In many instances, a stop criteria is imposed before the swarm runs its total course.

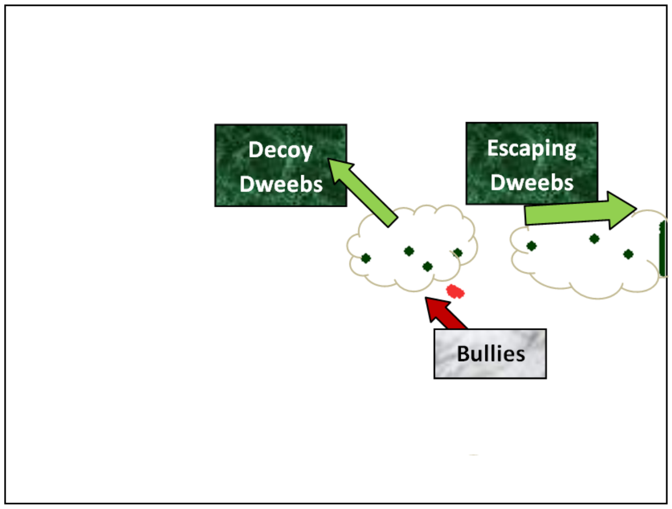

Fig. 4. The bullies are distracted by the decoy dweebs, eventually a single sacrificial dweeb emerges and is chased by the bullies until it is caught. The remaining dweebs seek refuge as far as away from the bullies as possible. Simulation software and a more illustrative and insightful video of the evolved swarm is on Neowarm.com [50] #1.

The values are then connected in a piecewise linear fashion to form the actuator function. The actuator function has a range for c of $[-1, 1]$. The point locations are evolved to maximize the desired emergent behavior property. For the simulations to follow, the control properties of all of the agents on a team are the same.

A. Bullies and Dweebs

The first game is based on a predator (bullies) and prey (dweebs) model. The dweebs are killed when they come into contact with the bullies. The bullies and dweebs begin uniformly randomly distributed across the playground. Only the dweeb strategy is evolved.

Both the bullies and dweebs make use of the same sensors

- S_1 = distance to nearest team agent
- S_2 =distance to the nearest agent on the other team
- S_3 =distance to the center of the playground.

They both have the consequents

- Movement towards nearest enemy agent
- Movement towards nearest friendy agent
- Movement towards center of play area

The swarm's fitness is defined as:

$$\mathbb{C} = \sum_{i=1}^{T} \mathfrak{d}_i \tag{5}$$

where $\mathfrak{d}_i$ is the number of living dweebs at time step i and $T = 2000$. The dweebs are evolved around the fixed operation of the bullies. Variations were observed to emerge dependent on the fixed point behavior of the bullies. Two cases are considered.

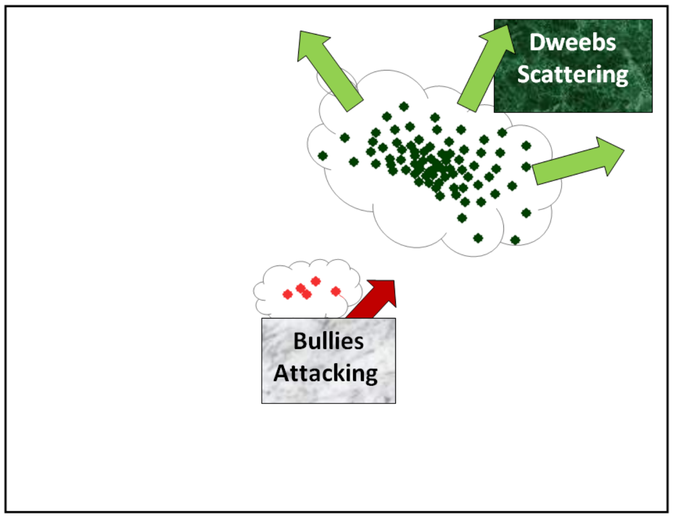

Fig. 5. The swarm of dweebs spread out as the bullies attack. Simulation software and a more illustrative and insightful video of the evolved swarm is on Neowarm.com [50] #2.

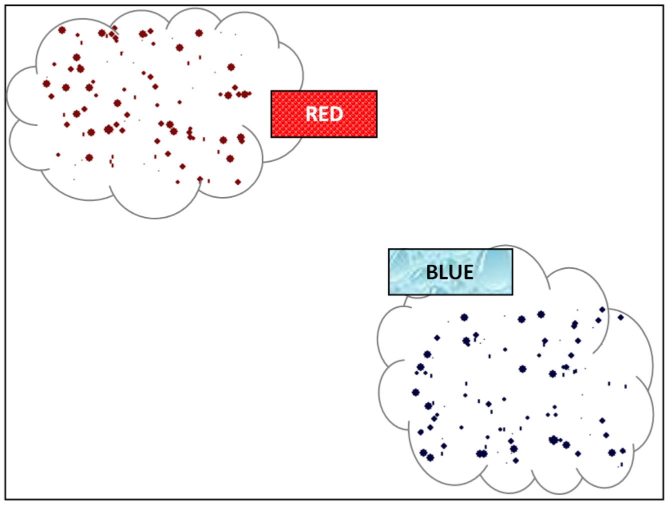

Fig. 6. The initial positions of the opposing gangs. Snapshots of the conflict are shown in Figures 8 through 9. Also see the on line video [50].

1) Slow Chases.: In this scenario, the speed of a bully towards the nearest dweeb is set proportional to the separating distance, *i.e.* a bully will run quickly towards a dweeb that is far away, and slowly towards a dweeb nearby. The dweebs are able to find a "sweet spot" where they can run slowly immediately in front the bully and avoid getting caught. One of the dweebs would typically "dance" with the bully while the others would hide out of range.

The interesting emergent dweeb behavior is one of self sacrifice. One dweeb at a time attracts the bullies in a prolonged chase while the remaining dweebs move to a nonthreatening position. Eventually the sacrificial dweeb is killed. After transient activity, a fresh self sacrificing dweeb emerges and the cycle continues. This emergent self sacrifice strategy performed the desired function of maximizing the overall lifetime of the dweeb swarm as measured by the fitness. See Figure 4 and the video available on line [50].

2) Center.: In a second scenario, bullies attack dweebs at a speed independent of their separation. Swarm inversion results in the dweeb strategy of clustering in the center of playing field. The bullies attack causing a scattering of the dweebs. The bullies then concentrate on sacrificial dweebs while other dweebs return to temporary safety in the center. As in the *slow chase*, the effect was of self sacrifice albeit not as dramatic. See Figure 5 and the video available on line [50].

B. Gang Warfare

In *gang warfare*, a second type of wargame, there are two gangs, red and blue, both able to retreat or to attack and kill the other. Each team agent has a randomly assigned strength, and when agents collide the weak is killed and the strongest agent survives. However, the winning agent loses strength. When an agent is killed, the strength of the survivor is decayed by a factor of 0.9. Furthermore, when an agent is far enough away from any enemy agent its strength slowly increases. If the nearest enemy agent is at least 0.1 units away, strength increase by .001 per iteration. Both teams are alternately evolved and a number of different strategies emerge.

The sensors used for both the red and the blue team are

- $\mathcal{S}_1$ = Distances to nearest enemy agent
- $\mathcal{S}_2$ = Agent's self assessment of its strength.
- $\mathcal{S}_3$ = Strength of nearest enemy agent

The two consequents for each agent are:

- Movement towards nearest enemy agent
- Movement perpendicular to nearest enemy agent

Fitness is defined to be:

$$\mathbb{C} = \sum_{i=1}^{T}(t_i - e_i) \tag{6}$$

where t_i is the number of agents of one team alive at time step i, and e_i is the number of agents alive of the opposing team at time step i and $T = 2000$.

The gangs are alternatingly evolved. Blue is first. As shown in Figure 6, the two gangs begin at the opposite sides of the play area. Snapshots of the game are shown in Figures 6 through 9. Also see the video [50].

Different strategies are observed to arise:

- *Orbiting*
  Strategically, it makes sense to attack the weak members of the enemy forces while keeping your own weak members out of the fight. The agents begin from the initial configuration in Figure 6. The strong and weak agents blue separate in Figure 7 and the stronger agents head into enemy territory to kill off the enemy agents in Figure 8. After the weak members have been killed off, the agents spin around the stronger enemy agents in Figure 9.
  The orbiting action comes from the agents moving both towards and perpendicular to the enemy agent. Spinning around the enemy agent would possible bring a weaker

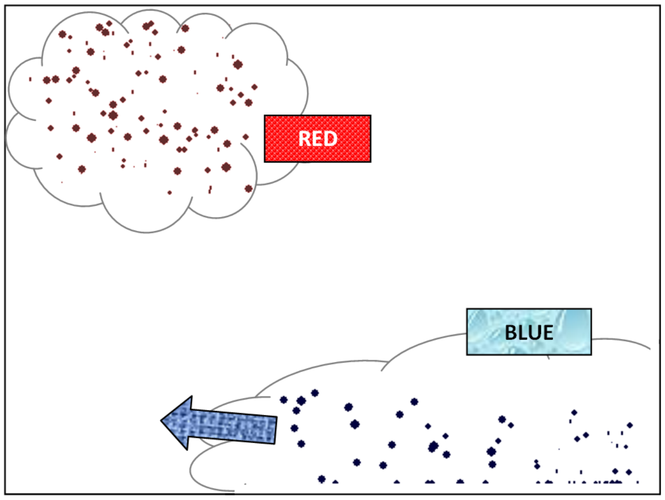

Fig. 7. The blue team divides.

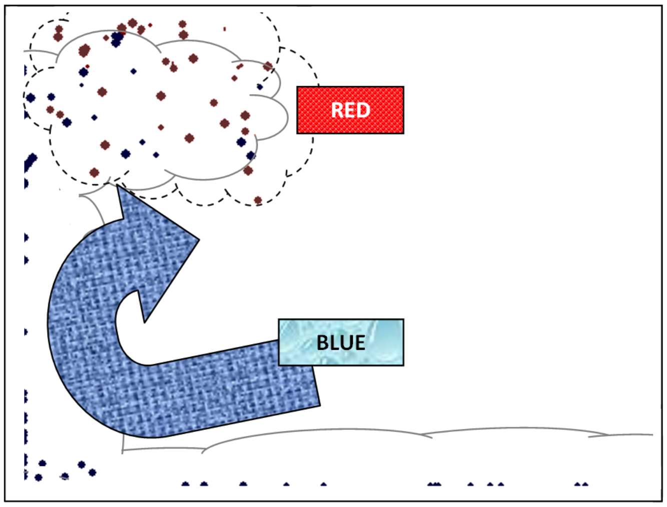

Fig. 8. Attacking from the west wall.

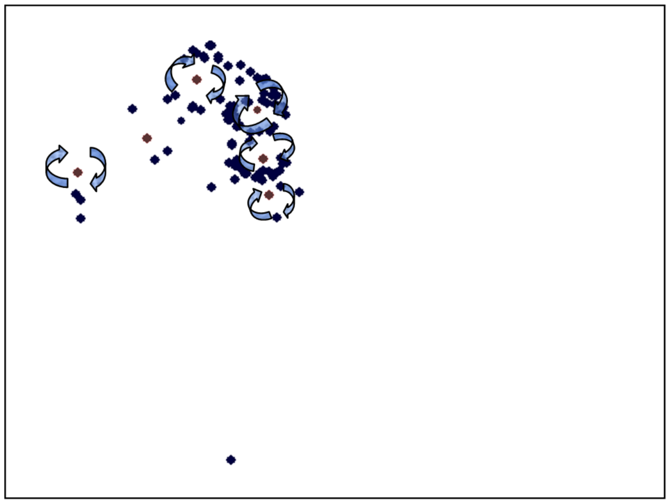

Fig. 9. Red clockwise circling the blue finishes the contest.

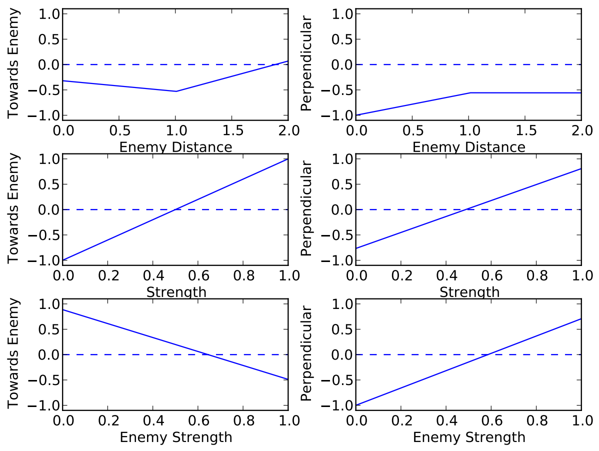

Fig. 10. The actuator functions evolved for the blue agents as used in Figures 6 through 9. The upper left function indicates the agent should retreat from the enemy unless the enemy is far away. The middle left function indicates that, the stronger the blue agent, the more it is inclined to move towards its enemy. Conversely, as shown in the bottom left, the stronger an enemy, the more inclined an agent is to retreat. The clockwise circular motion of blues around reds in Figure 9 is due to the actuator functions controlling perpendicular motion. As is the case with satellites orbiting earth, the combination perpendicular motion (the satellite's momentum) and the attraction to earth result in orbiting.

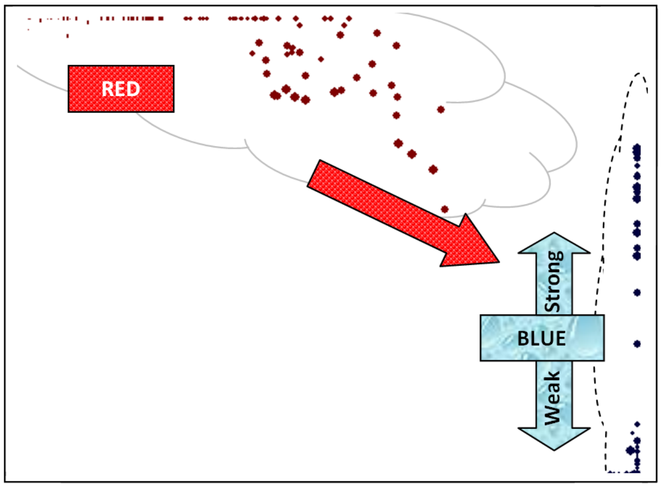

Fig. 11. The initial move after four eras of evolution: $\rightarrow B \rightarrow R \rightarrow B$. Blue, last evolved, adopts an defensive posture.

agent into view. In the end-stage the perpendicular movement prevents the circling agents from coming into contact with a strong enemy agent and being killed. Figure 10 shows the optimized actuator functions optimized for the blue to fight this battle.

- *Defensive Strategy*
After the blue team is evolved for an era, its behavior is fixed and the red team is evolved to counter the performance of the blue team. In this section, we look at the effects of the blue team being evolved a second time to counter the counters of the red team. We denote this

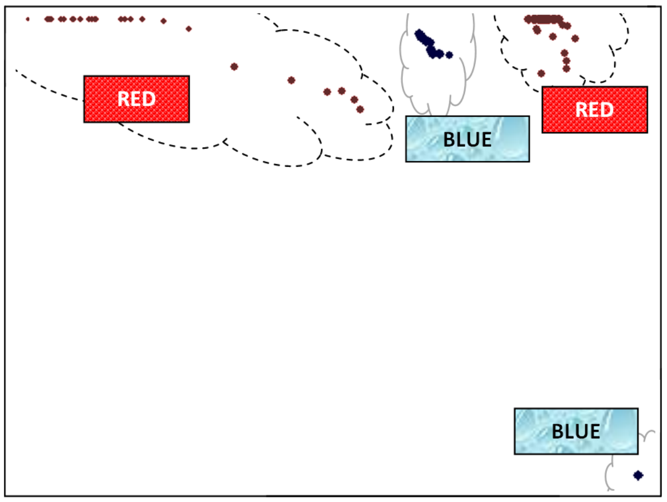

Fig. 12. With blue engaging in a defensive strategy, the red enemy goes after the nearer strong army, but does not actually attack it due to its strength.

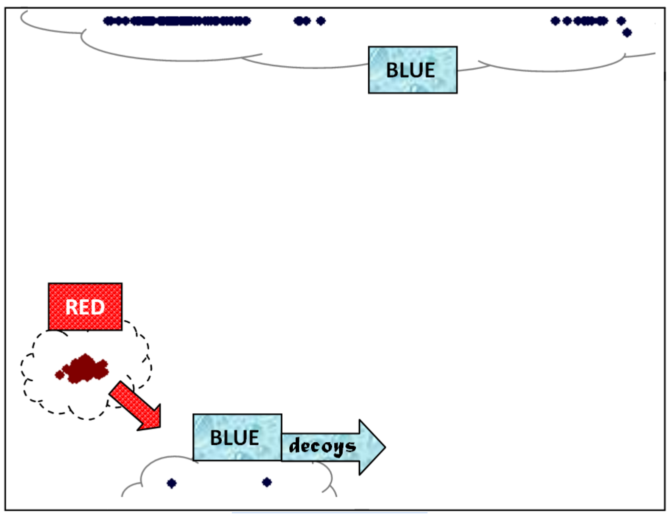

Fig. 15. Red pursues blue decoys.

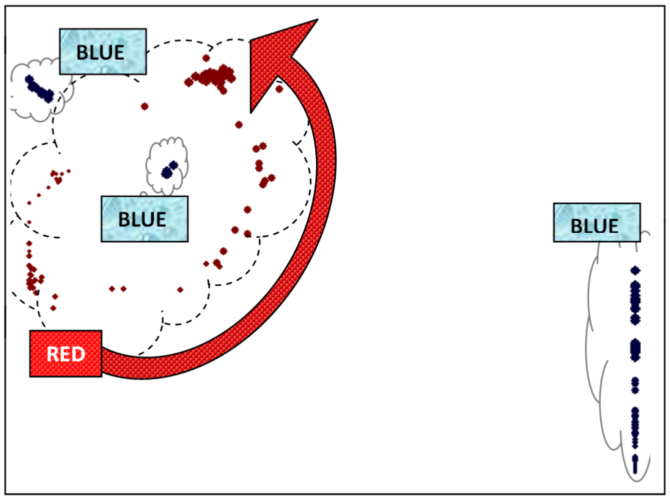

Fig. 13. Red circles blue counterclockwise.

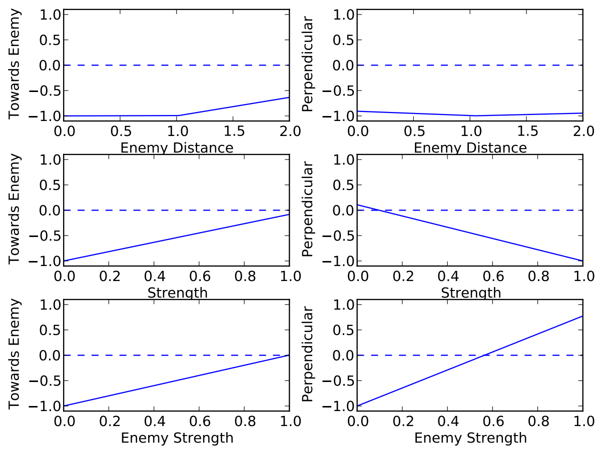

Fig. 16. The functions for the blue team engaged in a defensive strategy. Figures 6, through show this strategy as blue against an aggresive red strategy.

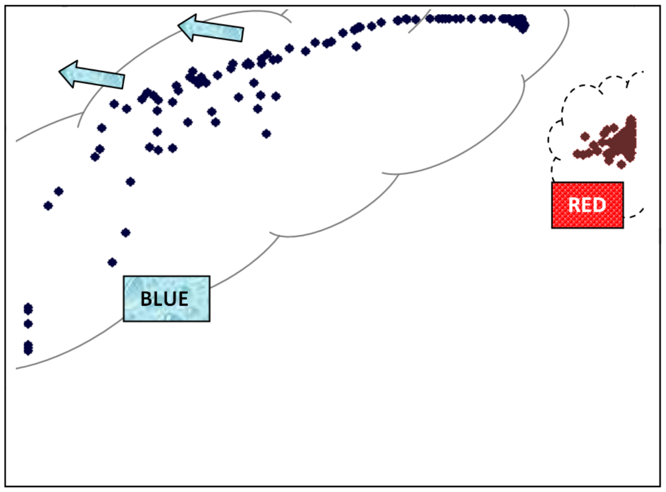

Fig. 14. Blue retreats from red.

process by $\rightarrow B \rightarrow R \rightarrow B$. Each right arrow denotes an evolution era. Snapshots of the swarming are shown in Figures 11 through 16. Also see the on line video [50]. The blue team evolved a defensive strategy. Instead of attempting to kill the enemy agents, the strategy now prioritizes not getting killed.

- Figure 11. The battle begins as shown. The blue team is applying a defensive strategy. The stronger blue agents move closer to the red enemy, thereby attracting their attention. The weaker blue agents retreat to the southeast corner.
- Figure 12. The red enemy agents continue to chase the blue agents but do not attack because the blue agents have too much strength. The weaker blue agents at the bottom right gain strength over time while this is happening.
- Figure 13. The blue agents in the southeast corner

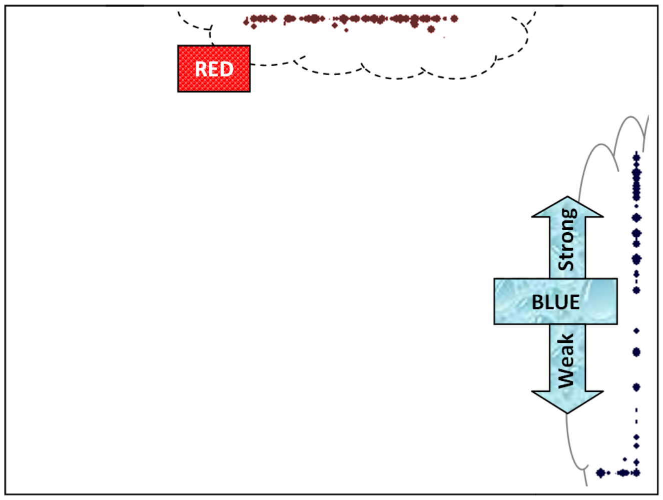

Fig. 17. Red uses an aggressive strategy against blue's defensive strategy. While the defensive strategy divides its army, the aggressive strategy pushes against the top wall and moves towards the upper left corner. Sequential snapshots of the conflict are shown in Figures 18 through fig.war.aggressive.3.

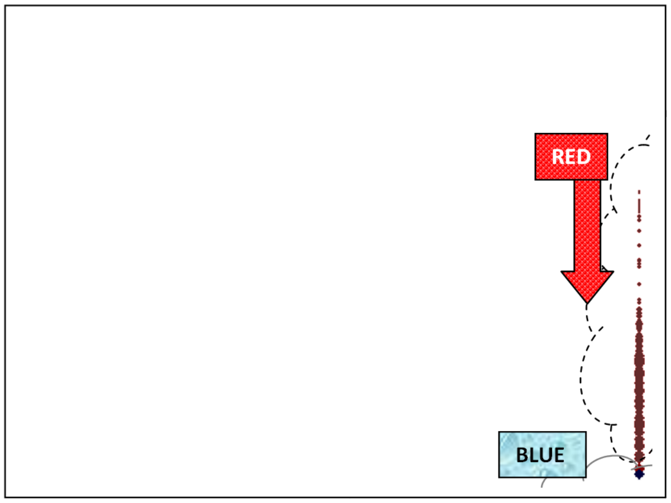

Fig. 19. Red uses an aggressive strategy against blue's defensive strategy. Blue attempts to hide in the corner while Red's entire army comes upon it resulting in a quick red victory.

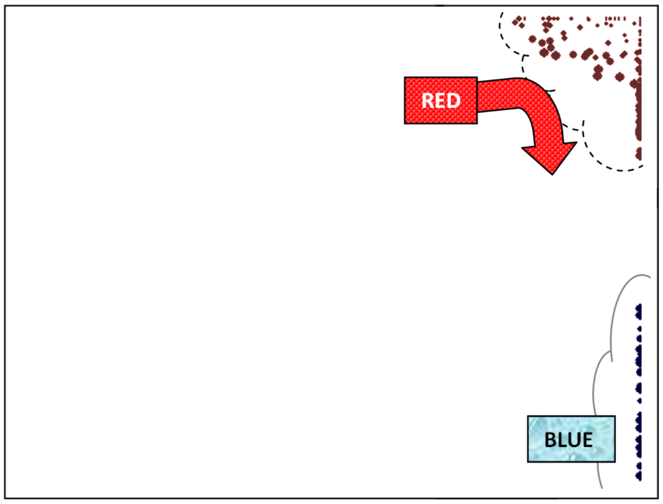

Fig. 18. Red uses an aggressive strategy against blue's defensive strategy. The defensive blue strategy retreats from the corner, while the aggressive red strategy follows. Note how the weaker red elements are in the back of the army.

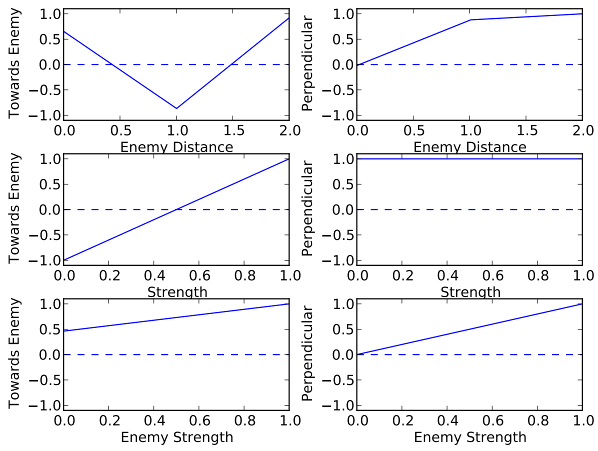

Fig. 20. The actuator functions for the red team engaged in an aggressive strategy. Figures 6, 17, 18, and **??** show the result of this strategy for red against the blue strategy in Figure 16.

have gained strength and are moving up the east wall in order to join the battle. The blue agent cluster at the top of Figure 12 has bifurcated. One portion is in the middle of a string of red agents who are circling in a counter clockwise direction. The other blue cluster in the northwest corner has temporarily escaped the attention of the red agents.

- Figure 14. After some time has passed, all agents are now at or near maximum strength. Using a philosophy that "The best offense is hiding," the blue agents are retreating from the red agent cluster by moving to the north and west.
- Figure 15. Blue decoys have broken from the group and are being pursued by the single red cluster. The other blues hide along the north wall. The red continues to chase the blue decoys, but never captures them. In this way, blue has evolved a retreat strategy that allows it to survive for a very long time.

Figure 16 shows the functions for the blue's defensive logic. All values for attraction to enemy red agents are negative, so blue will always try to run from the red enemy. However, blue's propensity to run is tempered by its strength which allows the separation shown in Figure 11.

- *Aggressive Strategy*

The defensive strategy depended on the opponent blue not being willing to attack strong red agents. Additional generations of evolution of the the red army addresses this. The evolution eras are now $\rightarrow B \rightarrow R \rightarrow B \rightarrow R$.

When given a chance to optimize against the defensive strategy, an aggressive strategy arose. The defensive blue strategy, effective in the $\rightarrow B \rightarrow R \rightarrow B$, remains unchanged. Evolution of the reds, though, has made the blue strategy ineffective. Here are some snapshots of the action. Also see video #5 [50].

- Figure 17. The reds move along the north wall to the right. There is some separation of the blues. Mostly stronger blues move up the east wall and some of the weaker blues move down.
- Figure 18. The reds make a sharp right turn at the northeast corner and move towards the meek blues who begin to retreat down the east wall.
- Figure 19. The red agents move downward on the east wall. The strong agents lead the way. The red agents catch the defensive agents in the bottom left corner, and rapidly kill them.

Figure 20 shows the functions used to control the red aggressive agents. The actuator functions for the blue remain those shown in Figure 16. Rather then avoiding strong agents, the red agents aggressively attack them to good effect. The propensity for moving towards enemy agents is related to an agent's strength. The constant value of 1.0 for perpendicular movement based on strength results in the wall following behavior.

C. *Foxes vs. Rabbits*

Foxes vs. rabbits is a third example used to illustrate swarm inversion using disjunctive Combs control. In this model, a bury of rabbits attempts escape to a hole guarded by an earth of predator foxes. Like the war model, each rabbit and fox agent is assigned a strength which determines whether or not it will survive in combat. However, unlike before, strength neither degrades or increases over time. There are 200 foxes and 50 rabbits.

The sensors used in this model were

- $\mathcal{S}_1$ = distance to center of friendly agents.
- $\mathcal{S}_2$ = distance to nearest opponent agent
- $\mathcal{S}_3$ = distance to exit (hole)
- $\mathcal{S}_4$ = difference between self strength and nearest opponent's strength

Fitness is defined as

$$\mathbb{C} = \sum_{i=1}^{T} (10e_i + s_i) \tag{7}$$

where e_i is the number of rabbits who have escape by time i, and s_i is the number of rabbits who have neither escaped or been killed and $T = 2000$.

- *Rush.* First, the foxes are evolved, and then the rabbits. So the scenario is

$$\rightarrow F \rightarrow R$$

Performance snapshots are shown in Figures 21 through 24. In this simplest case, the foxes head for the rabbits and the rabbits run away from the foxes. The rabbits have not yet learned to run to the hole.

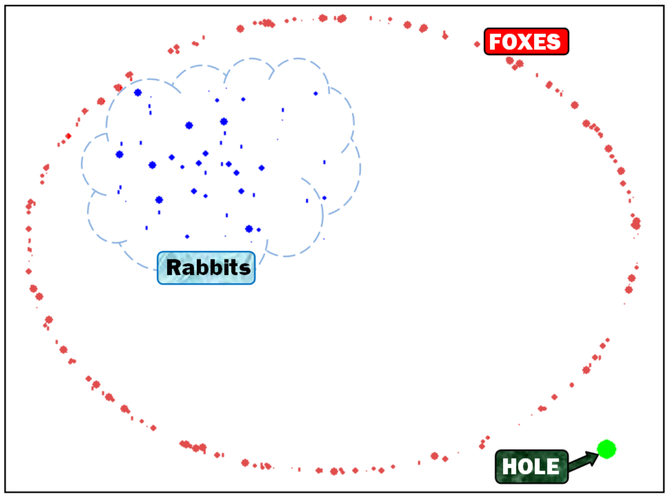

Fig. 21. The starting position for the foxes and rabbits. The foxes are red while the rabbits are blue. The green circle is the hole.

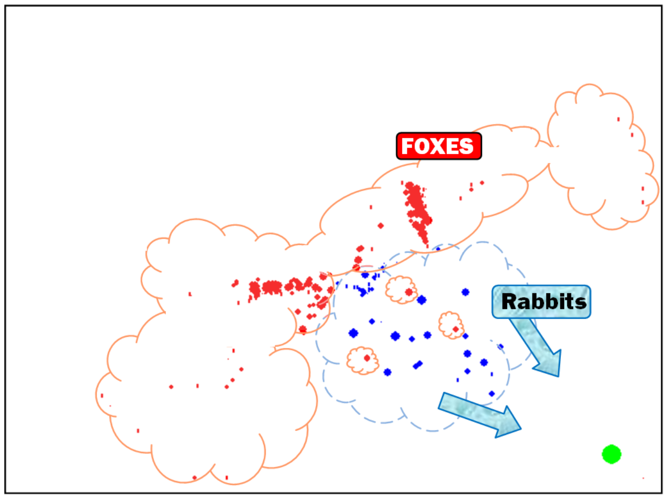

Fig. 22. The rabbits head for the hole, while the foxes chase after them

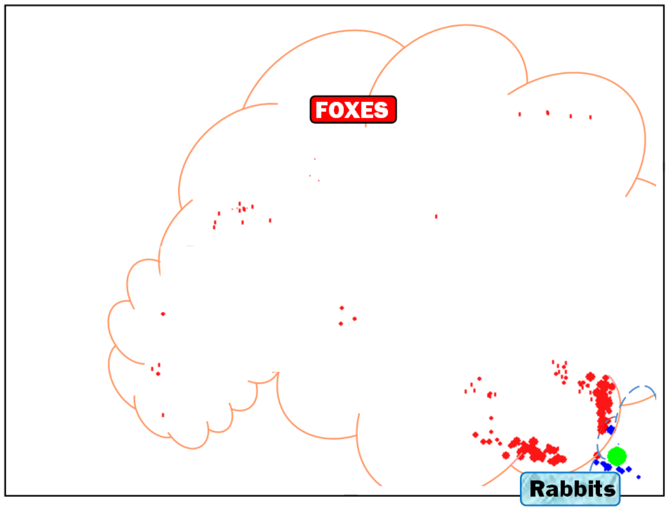

Fig. 23. The rabbits head for the hole, while the foxes chase after them

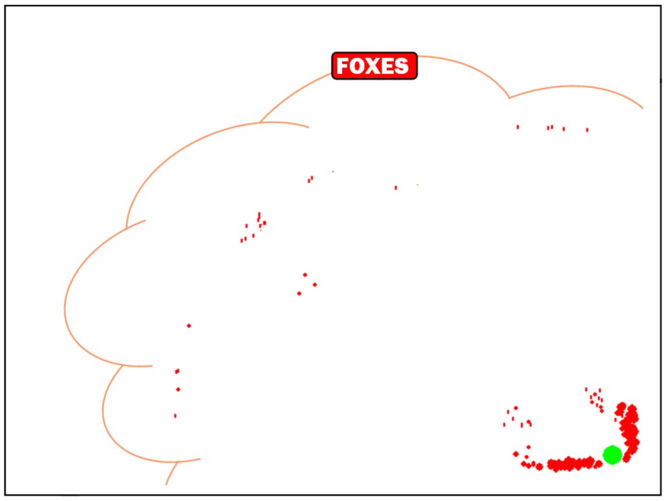

Fig. 24. All rabbits have either escaped or been caught

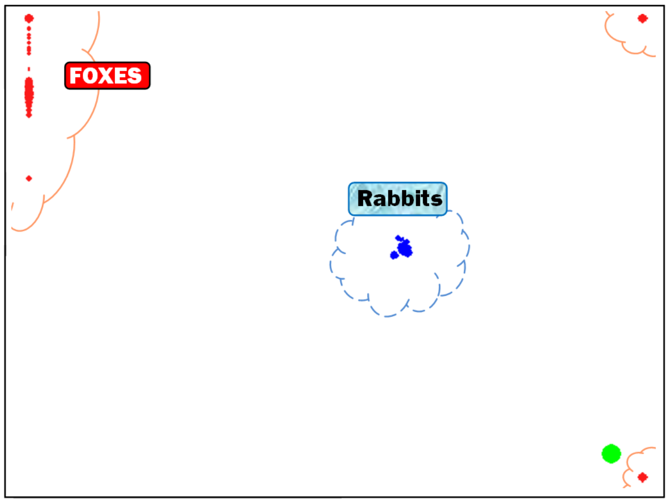

Fig. 25. The rabbits are confused and thus sit in the middle rather than running for the hole

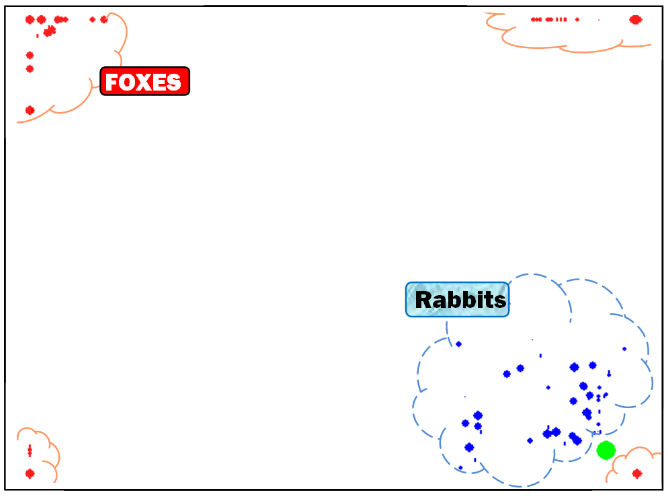

Fig. 26. The foxes hang around the outside corners, while the rabbits make it directly to the hole

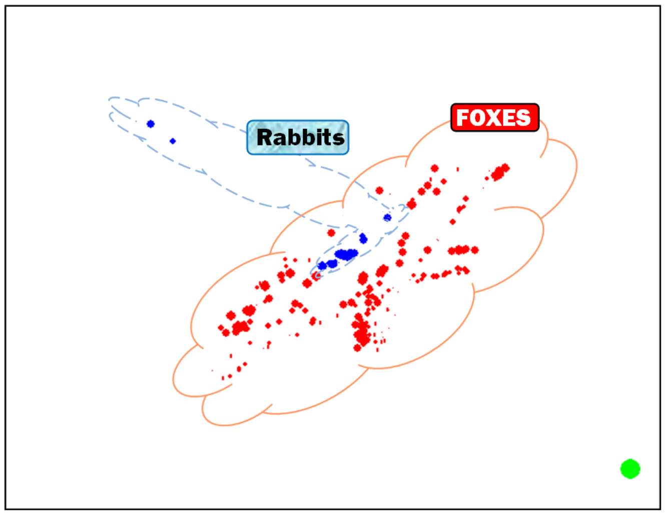

Fig. 27. In the initial stages of flanking strategy, the rabbits wait the foxes close in

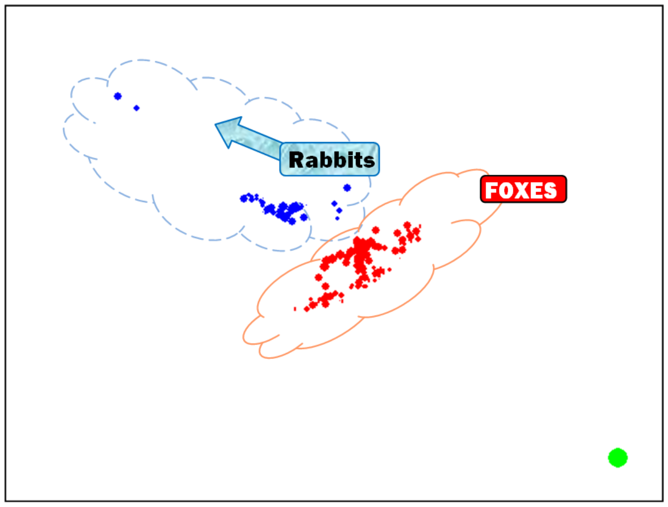

Fig. 28. After the foxes have gotten close enough in the flanking strategy, the rabbits run leaving the foxes in a clump

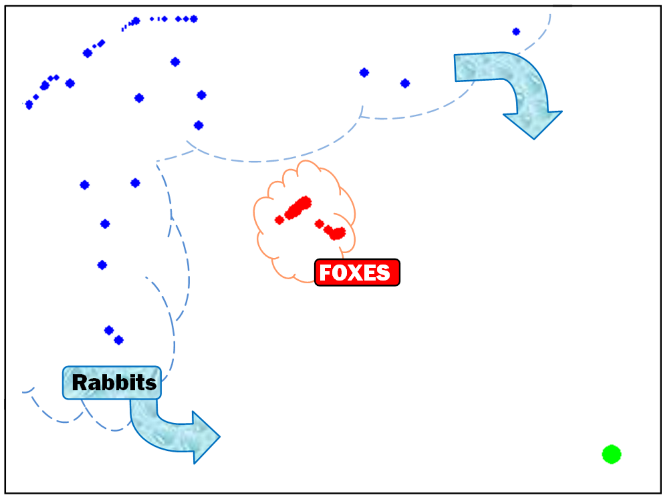

Fig. 29. The rabbits head both directions around the foxes.

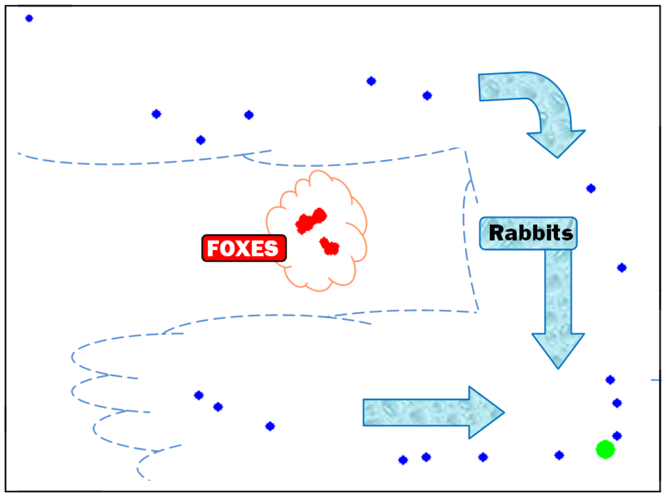

Fig. 30. Since the foxes are distracted by the nearest agents, the other rabbits sneak to the hole

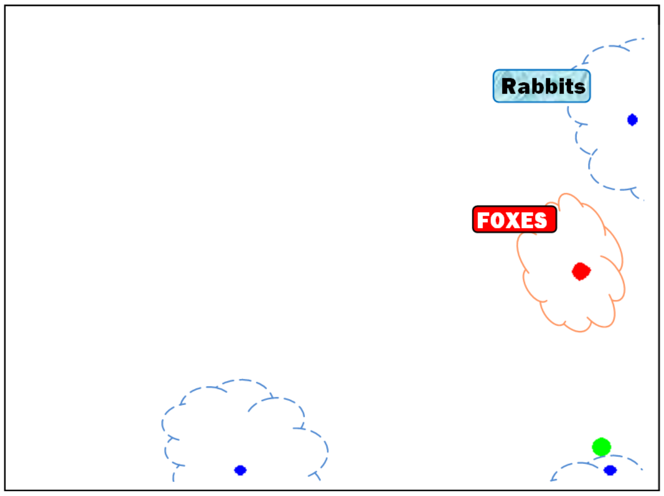

Fig. 31. Eventually there are insufficient rabbits to attract the attention, and the foxes move close to the hole preventing any future rabbit escapes.

  - ◦ Figure 21. This is the initialization of the contest.
  - ◦ Figure 22. As the foxes close in, the rabbits move towards the hole in order to escape the foxes.
  - ◦ Figure 23. The foxes continue to push the rabbits into the hole. The rabbits are not actually trying to move towards the hole, but rather away from the foxes. This happens to cause them to move close to the hole, allowing many to escape.
  - ◦ Figure 24.The simulation ends quickly as the foxes kill any non-escaping rabbits.

- *Confusion.*
  The foxes are next evolved to counter the *Rush* scenario.

$$\rightarrow F \rightarrow R \rightarrow F$$

  Since the rabbits were running away from the foxes rather than towards the hole, when the foxes were evolved against this behavior they developed a strategy of confusion. The foxes occupy the corners of the area thus keeping the rabbits in the center, This is shown in Figure 25.
- *Confusion Resolution.* The rabbits were again evolved

$$\rightarrow F \rightarrow R \rightarrow F \rightarrow R$$

  Predictably, the rabbits head towards the target unimpeded while the foxes, still applying their confusion tactics, remain huddled in the corners. This is shown in Figure 26. However, the agents are not actually trying to move towards the hole. Instead, they are heading to the nearest foxes who are behind the hole.
- *Flanking.* Evolution continued. At the end of a rabbit evolution, deception and flanking emerged as a winning strategy. The repeated evolutions resulted in clever rabbit behavior. As the foxes placed themselves between the rabbits and the hole, the rabbits move away from the foxes thereby drawing the foxes further from the hole. The rabbits then sneak little by little around the foxes and make their way to the rabbit hole. Snapshots are shown in Figure 27 through 31.
  - ◦ Figure 27. The rabbits allow themselves to be surrounded.
  - ◦ Figure 28. Just before they the foxes close in for the kill, the rabbits break out moving away from the hole leaving the foxes in a clump.
  - ◦ Figure 29. At the northwest corner, the rabbits bifurcate and begin to run around the flanks of the fox clump.
  - ◦ Figure 30. The sneaky rabbits then manage to make it to the exit. This works because the foxes only have a sense of the nearest rabbit. The flanking rabbits are, in this sense, invisible to the foxes.
  - ◦ Figure 31. Eventually, the foxes detect the flanking rabbits.The strategy of the rabbits then ceases to work, and the foxes move to a better defensive position. The remaining rabbits are now blocked from the hole and are doomed to a life above ground.

V. Conclusions

The inversion of swarms has numerous variations not considered. Application of coevolution [43] can potentially evolve strategies superior in a large number of strategies. Each agent can have a different control mechanism. Evolution can then be applied *in vitro* wherein killed agents are immediately replaced by replication of a mutated agent more fit. Agents can be fitted with adaptive states as is the case, for example, when worker ants are recruited to be soldier ants when the colony is under attack [11]. Use of Combs control can keep the dimensionality of these and other swarm scenarios within reason.

Swarm inversion can result in effective survival strategies in swarm games. Through a process of evolution, simple rules in agents allow emergent behavior that extends survival time for the swarms. More sophisticated simulations could find use in the development and experimental analysis of swarming in business models [12], military tactics [6], [23], finance [44], social science [62] and game theory [64].

Acknowledgements

This work was supported in parts by grants from the ONR and NSF.

References

[1] B. Adenso-Diaz, and M. Laguna. Fine-tuning of algorithms using fractional experimental designs and local search. Operations Research, 54(1):99 114 (2006).

[2] C. Ansótegui, M. Sellmann, and K. Tierney. A gender-based genetic algorithm for the automatic configuration of solvers. In Gent, I. P., editor, *Principles and Practice of Constraint Programming* CP 2009, volume 5732 of LNCS, pages 142157. Springer, Heidelberg, Germany

[3] P. Arabshahi, R.J. Marks II, and T.P. Caudell. Adaptation of Fuzzy Inferencing: A Survey. Proceedings of the IEEE/Nagoya University WWW on Learning and Adaptive Systems, pp.1-9, October 22-23, 1993, Nagoya University, (Nagoya, Japan)

[4] Arabshahi, P., Choi J.J., R.J. Marks II and T.P. Caudell. Fuzzy Parameter Adaptation in Optimization: Some Neural Net Training Examples, Computational Science and Engineering, (IEEE Computer Society), vol 3, No 1, Spring 1996, pp.57-65.

[5] P. Arabshahi, R.J. Marks II, Seho Oh, T.P. Caudell , J.J. Choi, and B.G. Song. Pointer Adaptation and Pruning of Min-Max Fuzzy Estimation. IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, vol.44, no.9, September 1997, pp.696-709.

[6] John Arquilla and David Ronfeldt. *Swarming and the Future of Conflict*. RAND Corporation (2000).

[7] Richard C. Aster and Brian Borchers, *i.e.* Parameter Estimation and Inverse Problems, Academic Press; 2 edition (2012)

[8] T. Bartz-Beielstein, *Experimental Research in Evolutionary ComputationThe New Experimentalism*. Springer, Berlin, Germany (2006).

[9] R. Beckers, O. E. Holland, and J. L. Deneubourg. From local actions to global tasks: Stigmergy and collective robotics. in *Prog. Artificial Life IV.* Cambridge, MA: MIT Press, 1994, pp. 181189.

[10] Birattari, M. (2009). *Tuning Metaheuristics: A machine learning perspective*. Springer, Berlin, Germany.

[11] E Bonabeau, M. Dorigo, and G. Thereulaz. *Swarm Intelligence: From Natural to Artificial Systems*, NY: Oxford Univ. Press, 1999

[12] E. Bonabeau and C. Meyer, "Swarm intelligence, a whole new way to think about business," Harvard Bus. Rev., vol. 79, no. 5, pp. 106114, May 2001

[13] Cristian Calude. *Multiset processing: mathematical, computer science, and molecular computing points of view*. Springer (2001)

[14] Oscar Castillo, Witold Pedrycz. *Soft Computing for Intelligent Control and Mobile Robotics* Springer (2010)

[15] William E. Combs. Reconfiguring the fuzzy rule matrix for large time-critical applications. in 3rd Annu. Int. Conf. Fuzzy-Neural Applicat., Syst., Tools, Nashua, NH, Nov. 1995, pp. 18:118:7.1

[16] William E. Combs and J. E. Andrews. Combinatorial rule explosion eliminated by a fuzzy rule configuration. IEEE Transactions on Fuzzy Systems, vol. 6, no. 1, pp. 1-11, Feb. 1998.

[17] William E. Combs, Jeffrey J. Weinschenk, Robert J. Marks II. Genomic Systems Design: A novel, biologically-based framework for enhancing the adaptive, autonomous capabilities of computer systems. FUZZ-IEEE 2004, IEEE International Conference on Fuzzy Systems, 25-29 July, 2004, Budapest.

[18] William A. Dembski and Robert J. Marks II. Conservation of Information in Search: Measuring the Cost of Success. IEEE Transactions on Systems, Man and Cybernetics A, Systems & Humans, vol.5, #5, September 2009, pp.1051-1061

[19] G. Di Caro and M. Dorigo. AntNet: Distributed stigmergetic control for communications networks. J. Artif. Intell. Res., vol. 9, pp. 317365, 1998.

[20] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: Optimization by a colony of cooperating agents. IEEE Trans. Syst., Man, Cybern. B, Cybern., vol. 26, no. 1, pp. 2941, Feb. 1996.

[21] M. Dorigo and T. Stützle, *Ant Colony Optimization.* Cambridge, MA: MIT Press, 2004.

[22] Eberhart, R.C.; Shi, Y. (2000). Comparing inertia weights and constriction factors in particle swarm optimization. Proceedings of the Congress on Evolutionary Computation. 1. pp. 8488.

[23] Sean J. A. Edwards. *Swarming on the Battlefield Past, Present, and Future*. RAND Corporation (2000).

[24] J.E. Ervin and S.E.Altekin. Combs Method Used in Intuitionistic Fuzzy Logic Application. in *Applications of fuzzy sets theory: 7th International Workshop on Fuzzy Logic* edited by Francesco Masulli, Sushmita Mitra, Gabriella Pasi. Springer (2007)

[25] A. S. Fukunaga. Automated discovery of local search heuristics for satisfiability testing. Evolutionary Computation, 16(1):3161 (2008).

[26] I. Kassabalidis, M. A. El-Sharkawi, R. J. Marks, P. Asabshahi, and A. A. Gray, "Swarm intelligence for routing in communication networks," in Proc. IEEE Globecom, San Antonio, TX, Nov. 2001, pp. 36133617.

[27] I.A. Gravagne and R.J. Marks II. Emergent Behaviors of Protector, Refugee and Aggressor Swarm. IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics, Volume 37, Issue 2, April 2007, pp. 471 - 476.

[28] M. Günes, U. Sorges, and I. Bouazizi. ARA – The ant-colony based routing algorithm for MANETs. in Proc. IEEE ICPPW, 2002, pp. 7985.

[29] A. T. Hayes, A. Martinoli, and R. M. Goodman. Swarm robotic odor localization. in Proc. IEEE/RSJ Int. Conf. IROS, Oct. 2001, pp. 10731087.

[30] F. Hutter, H. H. Hoos, K. Leyton-Brown and K. P. Murphy. An experimental investigation of model-based parameter optimisation: SPO and beyond. In Rothlauf, F., editor, *Genetic and Evolutionary Computation Con- ference*, GECCO 2009, pages 271278. ACM press, New York

[31] F. Hutter, H. H. Hoos, K. Leyton-Brown and T. Stützle. ParamILS: An automatic algorithm configuration framework. Journal of Artificial Intelligence Research, 36:267306 (2009)

[32] J.N. Hwang, C.H. Chan, R.J. Marks II, "Frequency selective surface design based on iterative inversion of neural networks", Proceedings of the International Joint Conference on Neural Networks, San Diego, 17-21 June 1990, vol. I, pp.I39-I44.

[33] J.N. Hwang, J.J. Choi, S. Oh and R.J. Marks II. "Query based learning applied to partially trained multilayer perceptrons,", IEEE Transactions on Neural Networks, Vol. 2, pp.131-136, (1991).

[34] Andrew Ilachinski. *Artificial War: Multiagent-Based Simulation of Combat*, World Scientific Pub Co Inc (2004)

[35] Mohammad Jamshidi, Nader Vadiee, Timothy Ross. *Fuzzy Logic and Control: Software and Hardware Applications* (v.2) Prentice Hall (1993)

[36] B. Jayaram. On the Law of Importation $(x \wedge y) \longrightarrow z \equiv (x \longrightarrow (y \longrightarrow z))$ in Fuzzy Logic. IEEE Transactions on Fuzzy Systems, vol.16, no.1, pp.130-144, Feb. 2008. doi: 10.1109/TFUZZ.2007.895969

[37] Craig A. Jensen, Russell D. Reed, Mohamed A. El-Sharkawi, Robert J. Marks II, "Location of Operating Points on the Dynamic Security Border Using Constrained Neural Network Inversion", Proceedings of the International Conference on Intelligent Systems Applications to Power Systems (ISAP), pp.209-217, Seoul, Korea, July 6-10, 1997.

[38] Craig A. Jensen, Russell D. Reed, Robert J. Marks II, Mohamed A. El-Sharkawi, Jae-Byung Jung; Miyamoto, R.T.; Anderson, G.M.; Eggen, C.J., "Inversion of feedforward neural networks: algorithms and applications", Proceedings of the IEEE, Volume: 87 9, Sept. 1999 , Page(s): 1536 -1549

[39] J. Kennedy and R.C. Eberhart. Particle Swarm Optimization. Proceedings of IEEE International Conference on Neural Networks. IV. pp. 19421948 (1995) doi:10.1109/ICNN.1995.488968.

[40] J. Kennedy. The particle swarm: social adaptation of knowledge. Proceedings of IEEE International Conference on Evolutionary Computation. pp. 303308 (1997).

[41] J. Kennedy and R.C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann (2001) ISBN 1-55860-595-9.

[42] H. Leung, R. Kothari, and A. Minai. Phase transition in a swarm algorithm for self-organizing construction. Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top., vol. 68, no. 4, pp. 046 111.1046 111.5, 2003.

[43] Y. Liu, X.Yao, Q. Zhao, T. Higuchi. Scaling up fast evolutionary programming with cooperative coevolution. Proceedings of the 2001 Congress on Evolutionary Computation, pp. 1101 - 1108 vol. 2 (2001)

[44] Francesco Luna and Alessandro Perrone. *Agent-based methods in economics and finance: simulations in Swarm.* (Kluwer, 2002)

[45] R.J. Marks II, S.Oh, P. Arabshahi, T.P. Caudell, J.J. Choi and B.G. Song. Steepest descent of min-max fuzzy if-then rules. Proceedings of

the International Joint Conference on Neural Networks, Beijing, vol. III, pp. 471-477, November 3-6, 1992.

[46] R.J. Marks II, Editor, *Fuzzy Logic Technology and Applications,* IEEE Technical Activities Board, Piscataway, 1994.

[47] Joseph McRae Mellichamp, *Go Fast, Turn Left*, ISBN #978-1427602008

[48] A. Montresor, H. Meling, and Ö. Babaoglu, "Messor: Load-balancing through a swarm of autonomous agents," Dept. Comput. Sci., Univ. Bologna, Bologna, Italy, Tech. Rep. UBLCS-2002–11, 2002.

[49] V. Nannen and A. E. Eiben. Relevance estimation and value calibration of evolutionary algorithm parameters. In Proc. of IJCAI 2007, pages 975-980. AAAI Press/IJCAI, Menlo Park, CA

[50] Neoswarm War Games http://neoswarm.com/war_games.html .

[51] S.Oh, R.J. Marks II and M.A. El-Sharkawi. Query based learning in a multilayered perceptron in the presence of data jitter. Applications of Neural Networks to Power Systems, (Proceedings of the First International Forum on Applications of Neural Networks to Power Systems), July 23–26, 1991, Seattle, WA, (IEEE Press, pp.72-75).

[52] M. Oltean. Evolving evolutionary algorithms using linear genetic programming. Evolutionary Computation, 13(3):387-410 (2005).

[53] Russell D. Reed, Seho Oh and R.J. Marks II. Regularization using jittered training data. Proceedings of the International Joint Conference on Neural Networks, Baltimore MD, pp.III147-III152, June 1992.

[54] Russell D. Reed, R.J. Marks II and S.Oh. An equivalence between sigmoidal gain scaling scaling and training with noisy (jittered) input data. Proceedings of the RNNS/IEEE Symposium on Neuroinformatics and Neurocomputing, (Rostov-on-Don, Russia, October, 1992), pp. 120-127

[55] Russell D. Reed, R.J. Marks II and Seho Oh. Similarities of error regularization, sigmoid gain scaling, target smoothing and training with jitter. IEEE Transactions on Neural Networks, vol. 6, no.3, May 1995, pp. 529-538.

[56] Russell D. Reed and R.J. Marks II, *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*, (MIT Press, Cambridge, MA, 1999.)

[57] Emad W. Saad and Donald C. Wunsch II. Neural network explanation using inversion. Neural Networks, Volume 20, Issue 1, January 2007, pp.78-93

[58] Y. Shi and R.C. Eberhart. A modified particle swarm optimizer. Proceedings of IEEE International Conference on Evolutionary Computation. pp. 6973 (1998).

[59] Y. Shi and R.C. Eberhart. Parameter selection in particle swarm optimization. Proceedings of Evolutionary Programming VII (EP98). pp. 591600 (1998).

[60] Adi Shklarsh, Gil Ariel, Elad Schneidman, Eshel Ben-Jacob. Smart Swarms of Bacteria-Inspired Agents with Performance Adaptable Interactions. PLoS Computational Biology, 2011; 7 (9): e1002177 DOI: 10.1371/journal.pcbi.1002177

[61] William Siler and James J. Buckley, *Fuzzy Expert Systems and Fuzzy Reasoning*, Wiley-Interscience (2004)

[62] Pietro Terna. Simulation Tools for Social Scientists: Building Agent Based Models with SWARM. Journal of Artificial Societies and Social Simulation. vol. 1, no. 2, (1998)

[63] Benjamin B. Thompson, Robert J. Marks II, Mohamed A. El-Sharkawi, Warren J. Fox, and Robert T. Miyamoto. Inversion of Neural Network Underwater Acoustic Model for Estimation of Bottom Parameters Using Modified Particle Swarm Optimizers. 2003 International Joint Conference on Neural Networks, July 20-24, 2003, Portland, Oregon (pp. 1301-1306).

[64] Kagan Tumer and David H. Wolpert. *Collectives and the design of complex systems*. (Springer, 2004).

[65] Jeffrey J. Weinschenk, William E. Combs, Robert J. Marks II. Avoidance of rule explosion by mapping fuzzy systems to a disjunctive rule configuration. 2003 International Conference on Fuzzy Systems (FUZZ-IEEE), St. Louis, May 25-28, 2003.

[66] Jeffrey J. Weinschenk, Robert J. Marks II, William E. Combs. Layered URC fuzzy systems: a novel link between fuzzy systems and neural network. 2003 International Joint Conference on Neural Networks, July 20-24, 2003 , Portland , Oregon (pp. 2995-3000).

[67] Jeffrey J. Weinschenk, Robert J. Marks II, William E. Combs. On the use of Fourier methods in URC fuzzy system design. FUZZ-IEEE 2004, Proceedings 2004 IEEE International Conference on Fuzzy Systems, Budapest, Volume 2, 25-29 July 2004, pp. 911 - 916.

[68] Jeffrey J. Weinschenk, William E. Combs, Robert J. Marks II. On the avoidance of rule explosion in fuzzy inference engines. International Journal of Information Technology and Intelligent Computing, vol.1, #4 (2007).

[69] Zhi Yuan, Marco de Oca, Mauro Birattari and Thomas Stützle. Modern Continuous Optimization Algorithms for Tuning Real and Integer Algorithm Parameters, in *Swarm Intelligence*, edited by Marco Dorigo, Mauro Birattari, Gianni Di Caro, René Doursat, Andries Engelbrecht, Dario Floreano, Luca Gambardella, Roderich Gross, Erol Sahin, Hiroki Sayama and Thomas Stützle. Springer Berlin / Heidelberg. isbn = 978-3-642-15460-7, pp. 203-214, (2010)

VI. Appendix

The emergent behavior resulting from the three simple swarm rules in the Introduction (Section I) are, as follows,

1) This is a simple model used for termites stacking wood or ants collecting their dead into piles.
2) This is the *protector* mode analyzed by Gravagne and Marks [27]. The emergent behavior is attraction of all the agents to a fixed set of points. If, in the coupling, all agents can be linked in the rules to all other agents, convergence is to a single point.
3) This is the *refugee mode* mode analyzed by Gravagne and Marks [27], all agents diffusively disperse.

Additional details and movies illustrating the behavior of these storms are available on NeoSwarm.com.