Recovery from Sensor Failure in an Evolving Multiobjective Swarm

Jon H. Roach, *Member, IEEE*, Robert J. Marks, II, *Fellow, IEEE*, and Benjamin B. Thompson

***Abstract*—One of the benefits of using Combs disjunctive control in swarm intelligence is the ability of the swarm to continue to operate even when one or more of its sensors are broken. Instead of failing, the swarm continues to perform based on the inputs of the remaining sensors. In this technical correspondence, a multiobjective scenario is designed for a swarm of agents to complete. Once optimized, the swarm is modified by removing one of its sensors. Maintaining the original objective, the evolution process is continued to allow the swarm to compensate for its handicap. This technical correspondence describes the different emergent behaviors of the swarm to compensate for the loss of function. The solution for the diminished swarm can vary significantly from that of the original.**

***Index Terms*—Combs control, emergent behavior, fuzzy control, multistate, swarm intelligence, task switching.**

I. Introduction

Swarms in nature are composed of a large number of individual agents, such as bees or ants, that each follows a set of basic rules. Often these rules can be described as a simple cause effect relationship between an external stimuli and an agent's response [1]–[7]. Each agent's response contributes to the emergent behavior of the swarm, which can often be unpredictable [1], [2], [7], [8]. However, what if agents were unable to sense a certain stimulus? For example, what if a group of drones was unable to sense the difference between friend and foe? Or what if they had difficulty sensing certain types of enemy units altogether? How would that reduction in awareness affect the emergent behavior of the swarm?

The robustness of disjunctive control, also called Combs control [12]–[18], has the advantage of seamlessly recovering from failed sensors by exploiting possibly redundant information available from other sensors when available [19], [21], [24]. A simple example is steering a car to the right. This can be done by: 1) turning the front wheels of the car to the right; 2) turning the back wheels of the car to the left; 3) braking the two wheels on the right side of the car; or 4) accelerating the rotation of the two wheels on the left side of the car. As long as one of these control actions exists, the car can be steered to the right even when the remaining three functions fail. Redundancy in the information available in sensors is often not as obvious as in this example. In the case of swarms, the failing of one or more sensors may prompt the swarm to adopt a different emergent strategy in order to meet the objective of the collective.

Combs control also avoids the exponential explosion of required rules in comparison with the more traditional conjunctive Mamdami fuzzy control [20], [22], [23]. The adjustment of parameters is therefore not hampered by the curse of dimensionality [25]–[29] often encountered in machine learning. Although there has been significant work in tuning artificial swarms [30]–[37], the goal is to enhance a specific identified emergent behavior such as path finding or foraging efficiency. Particle swarm, in particular, has used adaptive parameters to increase the swarm's ability to perform optimization [38]–[40], [42].

The evolutionary inversion of manageable Combs controlled swarms, on the other hand, can result in fascinating and often unexpected emergent behaviors [1], [6], [7]. We have presented application of Combs control to swarm, including its characterization and properties [7]. This technical correspondence builds on these results.

Although disjunctive Combs control indicates robustness to sensor failures, its resilience in this regard has, to our knowledge, never been tested. Sensors with redundancy should be less important in the control process than less redundant inputs. Identification of the value of a sensor is often not obvious. This is especially true in a complex swarm scenario. Using Combs controlled swarming, we explore the effects of sensor failure. The swarms, as predicted, are able to adapt and survive with only a moderate loss in performance. Also, as expected, some sensors are shown to be more important than others. The importance can be measured by the reduction in value of the survival metric.

What was not expected was that sensor loss would manifest itself by new and interesting survival strategies for the swarms. In other words, sensor loss can result in a significantly modified swarm emergent behavior.

A previously tested simulation is used as a baseline for this experiment [5]–[7]. The emergent behaviors that result from individually removing four of the agents' sensors are chronicled.

Although emerging swarm behavior will be explained herein in detail, there is no substitute for watching swarming in real-time. The motivated reader is therefore encouraged to view our video which both explains and displays the fascinating and emergent behaviors described in this technical correspondence [42].

II. Swarm Intelligence

In the simulation, a swarm is tasked with completing two objectives: 1) guarding a central base from enemy attacks and 2) searching out and destroying enemy units. In order to encourage recruitment, at least three agents are required to successfully kill an enemy unit. The simulation ends when the base sustains a set amount of damage: in this case, when ten enemy projectiles hit the base. The movement of the agents is controlled by sensors connected to weighting functions. When an agent senses an object, the distance to that object is fed into a function that tells the agent how to move with respect to the object. In addition to a series of object sensors and their corresponding weighting functions, agents are also equipped with a center sensor that tracks how far away the agents are away from the base and pulls them back in if necessary. All of the sensors have limited range, except for this center sensor.

In order to accomplish both objectives, the agents are allowed to take on one of multiple states: defender, scout or recruiter. For each state, there is a different set of sensor weighting functions, including

Manuscript received August 21, 2013; revised January 23, 2014; accepted May 7, 2014. Date of publication September 4, 2014; date of current version December 12, 2014. This paper was recommended by Associate Editor G. Biswas.

J. H. Roach is with L3 Communications, Greenville, TX 75402 USA (e-mail: jon.roach@L-3com.com).

R. J. Marks, II is with the Department of Electrical and Computer Engineering, Baylor University, Waco, TX 76798-7356 USA (e-mail: robert_marks@baylor.edu).

B. B. Thompson is with the Applied Research Laboratory, Pennsylvania State University, State College, PA 16804-0030 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMC.2014.2347254

the center sensor functions. When an object switches states, it is changing the set of rules that it follows. Since the swarm needs to dynamically adjust its resources, the agents are able to switch between states using threshold functions. If an object senses there are too many agents working on the same task, or too few, the agents are able to change states or request other agents to change states as necessary. How the agents make this decision is determined by a threshold value. Each weighting function and threshold is an adjustable parameter that represents a rule that the agents follow. These rules are optimized through the use of an evolutionary learning algorithm. This technical correspondence discusses what happens when one of these rules is removed from the swarm's decision making.

III. Evolution Process

A predatory swarm's performance can be maximized using an evolutionary learning algorithm [9], [10], [12], [18], [19], [21]. Performance is measured by tracking the number of enemy kills and the swarms' fitness scores. Each swarm is assigned a fitness score, which is equal to the time survived, multiplied by the percentage of surviving agents within the field of play (P_{agents}), multiplied by an efficiency factor, *E,* that represents the reduction of the distance traveled by the swarm, as shown

$$F = \text{Time} * P_{\text{agents}} * E * C \tag{1}$$

where E is the efficiency factor and C is a constant.

An initial random population of teams, each represented by a set of weighting functions and thresholds, is evolved by simulating the scenario with each member of the population and comparing the results. Teams with higher kill totals and fitness scores survive and advance to the next generation of evolution, while the poorer teams are removed. Each surviving team is copied and mutated by adding random Gaussian noise to the weights and thresholds. For weights with values ranging from negative five to five, the standard deviation of the Gaussian noise added is approximately 1.25. This algorithm allows the population of swarms to learn which rules and strategies successfully accomplished both objectives of attacking and defending [14], [35], [37], [43]–[46]. This process is repeated for approximately 800 generations, resulting in a set of weights that were optimized for the given scenario. A brief description of the emergent behaviors is given below.

To accomplish the first objective of defense, defenders learn to loosely circle the base and intercept enemy projectiles by moving between them and the base. If the number of defenders drops too low, nearby scouts switch tasks to help defend the base. For the second objective, scouts learn to spread out from the base and from each other while looking for enemy units. When an agent finds an enemy, it transforms into a recruiter and return to base. At the base and along the way, scouts join the recruiters until the recruiter senses that the group is strong enough to destroy the enemy unit. The recruiter leads the group back to the enemy in order to eliminate the enemy. Any surviving agents in the group then go back to exploring the map for enemies.

Now that we have a solution for the multitask scenario, our next step is to begin disabling sensors one at a time in order to determine how the failure of one of the sensors will impact the emergent behavior of the swarms. The first sensor that is removed for each swarm agent is the projectile sensor. This sensor is used by defending agents to see incoming enemy projectiles that are headed toward to friendly base. Defenders are normally able to intercept the projectiles. We want to see what would happen if the defenders are no longer able to sense these projectiles. The population of swarms that were evolved with all of their sensors intact is used as the initial population for a second round of evolution without this projectile sensor.

Fig. 1. Results of the evolution before and after breaking the four sensors are shown here. Before generation 200, the swarm is evolving with all of its sensors. At the discontinuity, the graph splits to represent the fitness scores after the removal of the sensors. The swarm performs well with its projectile and base sensors removed and okay without its group sensor, but very poorly when the center sensor is disabled.

This process is then repeated for another failed sensor. The projectile sensor is turned back on and the group sensor is removed. The group sensor allows the agents to track how many other agents are nearby. This is used in recruitment since it allows the recruiting agents to know if they have enough friendly agents around them to destroy an enemy unit. The evolving population is reset to the initial group of teams that are evolved with all their sensors and the evolutionary algorithm is run again. This is repeated two more times: once with the center sensor broken and again without the base sensor.

IV. Results

Fitness scores for each of the four evolutionary cycles are shown in Fig. 1. In each case, the fitness scores are lower with the sensor removed. However, these scores do improve over time as the swarms' weights are adjusted to maximize the use of the remaining sensors the swarms did have. Even though handicapped, in some cases a swarm is still able to achieve scores almost as high as a swarm with all sensors available. One of the main differences in emergent behaviors that developed is how the defending agents guarded the base. These strategies are depicted in Fig. 2.

Combs control subjects each sensor to a nonlinear actuator who shapes are determined through the evolutionary process. Examples of actuator nonlinearities are illustrated Fig. 3. The actuator outputs are aggregated to determine agent action [7].

A. Projectile Sensor

Removing the projectile sensor does not greatly affect the fitness scores, which drop slightly compared to the results of the previous evolution. This is because the swarms are able to adapt to the loss of the sensor and still accomplish their objectives similarly to before. With the sensor, defending agents circling the base are attracted to projectiles and move in for the kill. Without the sensor, defenders are unable to see the projectiles. The swarm's solution is to make its circling behavior tighter and faster. By rapidly circling around the base, agents are able to intercept most enemy projectiles simply by running into them. The tighter circle uses a smaller group of defenders to effectively defend the base and allows more scouts to look for enemy units. This is an example of a behavior that existed previously, but was adjusted in order to compensate for the loss of sensor

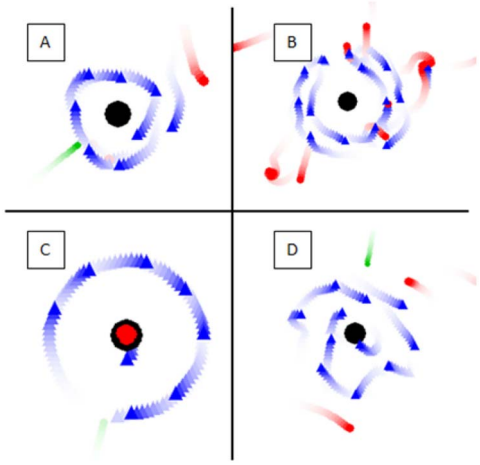

Fig. 2. This figure demonstrates the emergent behavior of defending agents evolved with different sensors disabled. (a) Tight ring formed when the projectile sensor is removed. (b) Attackers forming groups before scouting because the group sensor is off. (c) Ring formed around a group of replacement scouting agents when the center sensor is disabled. (d) Mob of defenders surrounding the base with the base sensor broken.

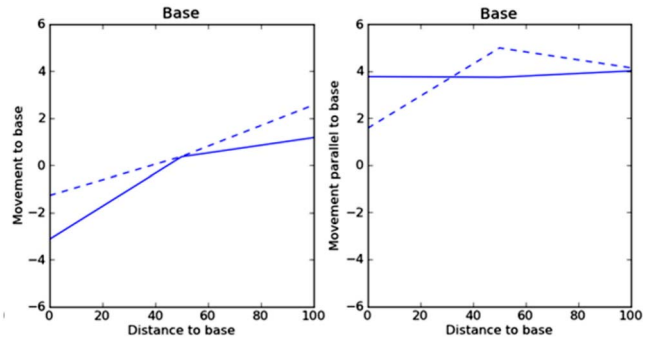

Fig. 3. Without the projectile sensor, the swarms adapt by tightening the ring around the base and speeding up their rotations. In the first graph, the old strategy is represented by the solid line and the new function is the dotted line. Initially, the "sweet spot" that the defenders converged to was around 50. This is lowered by evolutionary adaptation to a radius of 40 by the new method. In second graph, the speed at the "sweet spot" (40) from a little under four to almost five.

function. The behavior of the scouts and recruiters remains the same, since their task does not involve defending the base from projectiles.

B. Group Sensor

The group sensor is proven to have more of an effect on the swarms' behavior. This sensor is crucial for the recruitment techniques developed previously since it allows recruiters to count the number of nearby agents. When this ability is taken away, the agents are no longer able to determine if their group is large enough to destroy an enemy. Through the evolutionary process, the swarms respond by modifying their strategy. Instead of scouts spreading out, over time the scouts form up in clumps and scout in groups. This removes the need for recruiters to return to the base to find other agents. When an enemy is found by a group, recruitment is no longer needed. Instead, assuming a group of at least three agents, the group can simply move in immediately and destroy the enemy. The defensive strategy remains the same, but with a smaller number of defenders to allow more scouts to explore. Compared to the original evolution, scores drop since the effectiveness of the search is reduced when the scouts search in groups. However, considering the previous recruitment strategy no longer works at all, this emergent behavior is

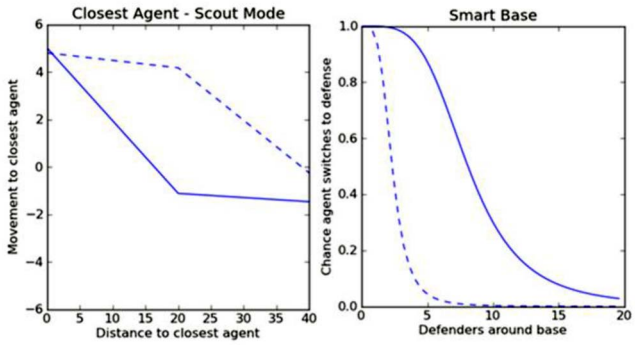

Fig. 4. Since the group sensor is disabled, the swarms have to adapt. Originally, scouts are repelled from each other. The value of the solid line is positive (indicating attraction) at small distances. However, since the agents are repelled at larger distances, the agents should never get close enough to each other for that to matter. The new function causes agents to be more attracted to each other. At large distances the attraction is weak, but still positive, and increases as the agents move closer. The second graph shows how many defenders the base is expecting. The swarm learns to use a smaller number of defenders in order to allow more agents to explore.

a reasonable alternative and allows the swarm to still find and destroy enemy units while maintaining the previous defensive strategy of surrounding the base in a rotating ring. The evolved graphs are shown in Fig. 4.

C. Center Sensor

When the center sensor is removed, the swarm scores drop off drastically. The center sensor is of foundational importance for the swarm mission. Without it, there is nothing to prevent exploring agents from wandering off the edge of the map. Since the swarm is unable to perform well in its task of finding and destroying enemies, it focuses on the second task of defending its base. The unexpected emergent behavior that develops from evolving without this sensor is the clumping of scouts in the base. When the simulation begins, some of the scouts form groups and begin searching, but most of the scouts form one large group and converge right on top of the base. These scouts, while not scouting, are used as reserves to replace destroyed defenders. The defending ring is similar to before, but with a much larger radius and a group of eight evenly spaced defenders surrounds the base. As the defenders detonate to take out incoming projectiles, they are replaced by scouts in the base. The defenders are far enough away from other defenders and the base (which is full of scouts) so that detonations do not take out any friendly units. This strategy allows a large portion of the swarm to stay defending the base which keeps the base alive for as long as possible. While the swarm is unable to properly complete both objectives, this modification does allow the swarm to perform well in at least one mission objective. The evolved graphs are shown in Fig. 5.

D. Base Sensor

Without the base sensor, the swarms are able to still achieve fitness scores on par with previous results. Since the defenders are unable to sense the base, the previous rotating ring strategy is impossible. To compensate for this loss, the defenders form a mob and surrounded the base. The defenders also learn to spread out from each other instead of linking up in the circling method. Previously, defenders used the attraction from the base sensor to stay close to the base. After evolution, the swarm learns to use the center sensor to accomplish the same objective by pulling in defenders that wander off more than 50 units away. This group of randomly moving defenders makes it difficult for enemy projectiles to reach the base, even though it is not

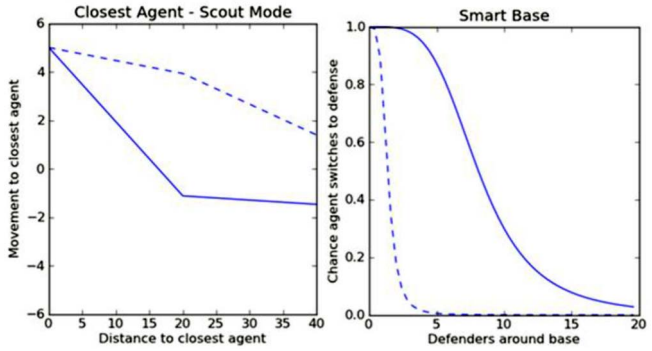

Fig. 5. These graphs look similar to those in Fig. 4. However, there are some significant differences. In the first graph, the new strategy developed without the center sensor is represented by the dotted line. Scouting agents are always attracted to each other since the function is positive for all distances, including an attraction of 1.5 at the maximum distance of 40. This is a much stronger attraction than that in Fig. 4. Also, the second graph shows how the swarms have learned to use fewer defenders, even less than when the group sensor is removed.

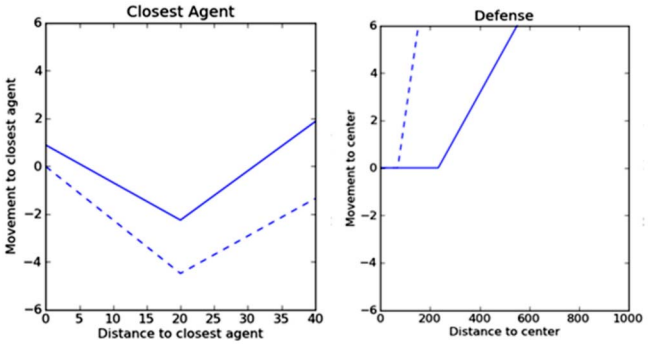

Fig. 6. These graphs represent some of the changes in rules that develop when the base sensor is turned off. The first graph shows how defenders react with respect to other agents. Initially, defenders are attracted to the "sweet spot" at the zero-crossing of the graph: around 30 units away. After evolving without the base sensor, the defenders learn to spread out and are always repelled based on the negative value of the function. The second graph shows the changes in the center sensor. Initially the sensor pulls back defenders that get more the 250 units away from the base. After evolution, the defenders are kept in a tighter radius of closer to 50.

as efficient as the ring strategy. The scouting and recruiting methods are not affected by the loss of the base sensor, so overall fitness scores remain high. The evolved graphs are shown in Fig. 6.

V. Conclusion

We have shown that while decreasing a trained swarm's abilities will negatively affect its performance, an evolutionary algorithm can allow the swarm to learn a new strategy that utilizes its remaining strengths. In four separate cases, a swarm is evolved to complete a multiobjective task, each time with the loss of one sensor. Sometimes an emergent behavior is refined to maintain swarm performance. An example of this is when the rotating defenders form a smaller, tighter circle around the base to better defend it when their projectile sensors were removed. Other times, a completely different strategy emerges, such as when scouts began exploring in groups to remove any need for recruitment. Also, if a swarm is unable to accomplish one objective due to sensor loss, then it would put more of its focus into the other objective. For instance, disabling the center sensor greatly reduces the swarm's ability to search for enemies so instead the swarm began to keep a large percentage of its agents near the base in order to survive for as long as possible. In each case, even though the fitness scores drop initially, the swarms are able to use other sensors in order to at least partially compensate for the sensor reduction.

The Python code used for these simulations is available online at NeoSwarm.com [42].

Acknowledgment

The authors would like to thank Baylor University, the Applied Research Laboratory at the Pennsylvania State University, and especially the Office of Naval Research's University Laboratory Initiative for funding for this technical correspondence.

References

[1] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*. New York, NY, USA: Oxford Univ. Press, 1999.

[2] E. Bonabeau and C. Meyer, "Swarm intelligence: A whole new way to think about business," *Harvard Bus. Rev.*, vol. 79, no. 5, pp. 106–115, 2001.

[3] A. K. Das, R. J. Marks, M. El-Sharkawi, P. Arabshahi, and A. Gray, "The minimum power broadcast problem in wireless networks: An ant colony system approach," in *Proc. IEEE CAS Workshop Wireless Commun. Netw.*, 2002, pp. 5–6.

[4] I Kassabalidis, M. A. El-Sharkawi, R. J. Marks, P. Arabshahi, and A. A. Gray, "Adaptive-SDR: Adaptive swarm-based distributed routing," in *Proc. IEEE Int. Joint Conf. Neural Netw. (IJCNN)*, vol. 1. Honolulu, HI, USA, 2002, pp. 351–354.

[5] J. Roach, R. J. Marks, II, and B. B. Thompson, "Tactical task allocation and resource management in non-stationary swarm dynamics," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN),* Dallas, TX, USA, Aug. 2013, pp. 1–5.

[6] J. H. Roach, B. B. Thompson, and R. J. Marks, II. (2014, Aug. 25). *Mapping an Underwater Minefield with a Multi-State Swarm and the Effects of Swarm Size on Performance* [Online]. Available: http://robertmarks.org/REPRINTS/2013_Mapping an Underwater Minefield with a Multi-State Swarm and the Effects of Swarm Size on Performance.pdf

[7] J. Roach, W. Ewert, R. J. Marks, II, and B. B. Thompson, "Unexpected emergent behaviors from elementary swarms," in *Proc. IEEE 45th Southeastern Symp. Syst. Theory (SSST)*, Waco, TX, USA, Mar. 2013, pp. 41–50.

[8] I. Gravagne and R. Marks, II, "Emergent behaviors of protector, refugee, and aggressor swarms," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 37, no. 2, pp. 471–476, Apr. 2007.

[9] Z. Yuan, "Continuous optimization algorithms for tuning real and integer parameters of swarm intelligence algorithms," in *Proc. ANTS 7th Int. Conf. Swarm Intell.*, Berlin, Germany, 2010, pp. 203–214.

[10] M. Clerc and J. Kennedy, "The particle swarm—Explosion, stability, and convergence in a multidimensional complex space," *IEEE Trans. Evol. Comput.*, vol. 6, no. 1, pp. 58–73, Feb. 2002.

[11] W. Ewert, R. J. Marks, II, B. B. Thompson, and A. Yu, "Evolutionary inversion of swarm emergence using disjunctive combs control," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 43, no. 5, pp. 1063–1076, Sep. 2013.

[12] W. E. Combs, "Reconfiguring the fuzzy rule matrix for large time-critical applications," in *Proc. 3rd Annu. Int. Conf. Fuzzy-Neural Appl. Syst. Tools*, Nashua, NH, USA, Nov. 1995, pp. 18.1–18.7.

[13] J. E. Ervin and S. E. Alptekin, "Combs method used in an intuitionistic fuzzy logic application," in *Applications of Fuzzy Sets Theory.* Berlin, Germany: Springer, 2007, pp. 306–312.

[14] W. E. Combs, "Method and system for controlling command execution," U.S. Patent 7 577 870, Aug. 18, 2009.

[15] W. E. Combs and J. E. Andrews, "Combinatorial rule explosion eliminated by a fuzzy rule configuration," *IEEE Trans. Fuzzy Syst.*, vol. 6, no. 1, pp. 1–11, Feb. 1998.

[16] S. Dick, A. Kandel, and W. E. Combs, "Comments on 'Combinatorial rule explosion eliminated by a fuzzy rule configuration'," *IEEE Trans. Fuzzy Syst.,* vol. 7, no. 4, pp. 475–478, Aug. 1999.

[17] J. M. Mendel, Q. Liang, and W. E. Combs, "Comments on 'Combinatorial rule explosion eliminated by a fuzzy rule configuration'," *IEEE Trans. Fuzzy Syst.*, vol. 7, no. 3, pp. 369–373, Jun. 1999.

[18] J. J. Weinschenk, W. E. Combs, and R. J. Marks, II, "On the avoidance of rule explosion in fuzzy inference engines," *Int. J. Inf. Technol. Intell. Comput*., vol. 1, no. 4, pp. 1–26, 2007.
[19] S. Narayanan *et al*., "Set constraint discovery: Missing sensor data restoration using auto-associative regression machines," in *Proc. IEEE World Congr. Comput. Intell. Int. Joint Conf. Neural Netw.*, Honolulu, HI, USA, May 2002, pp. 2872–2877.
[20] E. H. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *Int. J. Man-Mach. Stud.,* vol. 7, no. 1, pp. 1–13, 1975.
[21] B. B. Thompson, R. J. Marks, II, and M. A. El-Sharkawi, "On the contractive nature of autoencoders: Application to missing sensor restoration," in *Proc. Int. Joint Conf. Neural Netw.*, Portland, OR, USA, Jul. 2003, pp. 3011–3016.
[22] M. Zarozinski and L. Than, "Imploding combinatorial explosion in a fuzzy system," in *Game Programming Gems 2*. Newton Centre, MA, USA: Charles River Media, 2001, pp. 342–350.
[23] R. J. Marks, II, Ed., *Fuzzy Logic Technology and Applications*. New York, NY, USA: IEEE Technical Activities Board, 1994.
[24] M. A. El-Sharkawi and R. J. Marks, II, "Missing sensor restoration for system control and diagnosis," in *Proc. Symp. Diagn. Electr. Mach. Power Electron. Drives*, Atlanta, GA, USA, Aug. 2003, pp. 338–341.
[25] L. Chen, "Curse of dimensionality," in *Encyclopedia of Database System*. New York, NY, USA: Springer, 2009, pp. 545–546.
[26] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *Proc. 30th Annu. ACM Symp. Theory Comput*., 1998, pp. 604–613.
[27] J. H. Friedman, "On bias, variance, 0/1–loss, and the curse-of-dimensionality," *Data Mining Knowl. Discov.*, vol. 1, no. 1, pp. 55–77, 1997.
[28] M. Palaniswami, R. J. Marks, II, and D. B. Fogel, *Computational Intelligence: A Dynamic System Perspective*. New York, NY, USA: IEEE Press, 1995.
[29] R. D. Reed and R. J. Marks, II, *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*. Cambridge, U.K.: MIT Press, 1998.
[30] L. Li, A. Martinoli, and Y. S. Abu-Mostafa, "Learning and measuring specialization in collaborative swarm systems," *Adapt. Behav.*, vol. 12, nos. 3–4, pp. 199–212, 2004.
[31] Tattersall *et al.*, "Swarm-based adaptation: Wayfinding support for lifelong learners," in *Adaptive Hypermedia and Adaptive Web-Based Systems.* Berlin, Germany: Springer, 2004, pp. 336–339.
[32] T. Schmickl, C. Möslinger, R. Thenius, and K. Crailsheim, "Individual adaptation allows collective path-finding in a robotic swarm," in *Proc. Int. J. Factory Autom., Robot. Soft Comput.*, 2007, pp. 102–108.
[33] W. Liu, A. F. T. Winfield, and J. Sa, "Modelling swarm robotic systems: A case study in collective foraging," in *Proc. Towards Autonom. Robot. Syst. (TAROS),* 2007, pp. 25–32.
[34] Liu *et al.*, "Strategies for energy optimisation in a swarm of foraging robots," in *Swarm Robotics*. Berlin, Germany: Springer, 2007, pp. 14–26.
[35] D. Cvetkovic and I. Parmee, "Evolutionary design and multi-objective optimisation," in *Proc. 6th Eur. Congr. Intell. Techn. Soft Comput. (EUFIT)*, Aachen, Gennany, 1998, pp. 397–401.
[36] S. Hettiarachchi and W. M. Spears, "Distributed adaptive swarm for obstacle avoidance," *Int. J. Intell. Comput. Cybern.*, vol. 2, no. 4, pp. 644–671, 2009.
[37] K. Liang, X. Yao, and C. Newton, "Dynamic control of adaptive parameters in evolutionary programming," in *Simulated Evolution and Learning*. Berlin, Germany: Springer, 1999.
[38] Y. Shi and R. C. Eberhart, "Fuzzy adaptive particle swarm optimization," in *Proc. IEEE Congr. Evol. Comput.*, vol. 1. Seoul, Korea, 2001, pp. 101–106.
[39] X. Li, J. Branke, and T. Blackwell, "Particle swarm with speciation and adaptation in a dynamic environment," in *Proc. 8th Annu. Conf. Genet. Evol. Comput.*, 2006, pp. 51–58.
[40] Z.-H. Zhan, J. Zhang, Y. Li, and H. S.-H. Chung, "Adaptive particle swarm optimization," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 39, no. 6, pp. 1362–1381, Dec. 2009.
[41] X. Yang, J. Yuan, J. Yuan, and H. Mao, "A modified particle swarm optimizer with dynamic adaptation," *Appl. Math. Comput.*, vol. 189, no. 2, pp. 1205–1213, 2007.
[42] (2014, Aug. 25). *NeoSwarm.com* [Online]. Available: http://neoswarm.com/videos.html
[43] D. Fogel, *Blondie24*. San Mateo, CA, USA: Morgan Kaufmann, 2002.
[44] C. Fonseca and P. Fleming, "An overview of evolutionary algorithms in multiobjective optimization," *Evol. Comput.,* vol. 3, no. 1, pp. 1–16, 1995.
[45] F. Kursawe, "A variant of evolution strategies for vector optimization," in *Parallel Problem Solving from Nature.* Berlin, Germany: Springer, 1991.
[46] S. Carlson, "A general method for handling constraints in genetic algorithms," in *Proc. 2nd Annu. Joint Conf. Inf. Sci.*, 1995, pp. 663–667.