Proceedings of the 2007 IEEE Workshop on
Evolvable and Adaptive Hardware (WEAH 2007)

Swarm applications: a potential future application for evolvable hardware

James Hereford
Murray State University
Department of Engineering and Physics
Murray, KY 42071

Abstract

This paper looks at the applicability of using evolvable hardware principles to develop control strategies for swarm applications. It discusses the expected hardware capabilities for each agent in the swarm, the advantages of using evolvable algorithms to program the swarm, and some of the evolvable approaches being investigated by researchers. The paper concludes with a list of constraints that swarm applications place on the evolutionary algorithms and then analyzes which evolutionary algorithms are appropriate to derive controllers for swarms.

1. Introduction

The principles of evolvable hardware (EHW) have proven successful at designing individual hardware components. Over the past several years, EHW researchers have derived/evolved circuits using both extrinsic [Koza 2003, Lohn 2004, among others] and intrinsic [Greenwood 2004, Hereford 2005, Stoica 2002, Higuchi 1999, Thompson 1996, among others] techniques. This paper looks at the possibilities and potential of applying EHW principles to programming a swarm of agents/robots. The challenge is to find an effective, practical method to derive cooperative behavior among the multiple agents. We want to evolve an intelligent "swarm".

Intelligent swarms can be used in a number of applications. The U.S. military is investigating swarm techniques for controlling unmanned vehicles. NASA is investigating the use of swarm technology for planetary mapping. A 1992 paper discusses the possibility of using swarm intelligence to control nanobots within the body for the purpose of killing cancer tumors [Lewis 1992]. Swarm technology is particularly attractive because it is cheap, robust, and simple. Swarm intelligence concepts can even be found throughout popular culture; examples include the science fiction novel Prey by Michael Crichton and the movie Jason X [wiki 2006].

The idea of applying EHW concepts to swarms is related to swarm robotics. Swarm robotics is the study of how large number of relatively simple physically embodied agents can be designed such that a desired collective behavior emerges from the local interactions among agents and between the agents and the environment. (This definition is taken from the swarm robotics web page [swarm].) It is a novel approach to the coordination of large numbers of robots. It is inspired from the observation of social insects ---ants, termites, wasps and bees--- which stand as examples of how a large number of simple individuals can interact to create collectively intelligent systems. Social insects are known to coordinate their actions to accomplish tasks that are beyond the capabilities of a single individual: termites build large and complex mounds, ants organize impressive foraging raids and can collectively carry large preys. Such coordination capabilities are still beyond the reach of current multi-robot systems.

The challenge is to determine the best way to program the individual agents so that there is cooperative behavior among the agents/bots. Cooperative behavior among individual bots will then allow the swarm to engage in meaningful tasks. Some of the possible tasks are:

- Move about and explore an area (body, building, warehouse, field)
- Monitor the environment
- Identify a target
- If desired, neutralize the target
- Track a plume
- Make a bridge
- Form an antenna pattern and communicate with another entity

Note that each of the above tasks is considered a system-level activity. Each individual bot is not responsible for exploring the entire region, finding every target and communicating the results. Instead, the entire swarm, working in unison, is responsible for accomplishing the desired task.

When there is a large number of agents in a swarm (say, >1000 agents), there are both programming issues and logistical issues. The logistical issues include how to transport the agents, how to communicate with each one, how to provide power, how to upload new programs/data to each agent, and how to identify each agent (unique name?, unique ID?). If there is a large number of agents, then even turning them on becomes logistically troublesome.

In addition to the logistical issues, there are challenging programming issues. One of the programming issues is how to how program the individual agents in the swarm. Does one have identical programs in each agent/bot or can there possibly be slightly different programs? And if each agent has the same program, how is that program derived? Does a person take an algorithm that works with a swarm size of 1 and just assume that it will work well with a swarm size of 1000 or are there fundamental differences between the two sizes? How does one sift through the information from the agents to winnow out the redundant and unnecessary data and determine what is important? And, generally, one wants the algorithm to be scalable to different swarm sizes. The swarm size may be decreased because of individual bot failures. The swarm size may be increased due to merging of multiple swarms or an infusion of more bots to increase capabilities and/or decrease the search time. How does a user derive a program when the number of agents/hardware elements is unknown ahead of time and may even change during the course of the operation? EHW principles offer an interesting approach to deal with at least some of these programming issues.

This paper will look at the effectiveness of using evolvable hardware techniques to derive control startegies for swarm applications. In Section 2 we will discuss the expected hardware capabilities for each agent in the swarm. In Section 3, we will look at the advantages of using evolutionary approaches to program hardware swarms. Section 4 looks at the approaches researchers are currently using to evolve control strategies for swarm applications. And in Section 5, we look at the specific characteristics that evolutionary algorithms must have to work with mult-agent/swarm type applications.

2. Hardware system capabilities

The hardware part of this evolvable hardware application consists of multiple interacting agents. (We will often refer to the agents as bots, since they have behaviors similar to small robots.) The hardware requirements for each agent in the swarm are discussed in this section. At this point, we will assume that each agent has identical characteristics but will discuss that assumption more in Section 3.

2.1 Requirements for the swarm

There has been discussion on what the term "swarm robotics" should mean in relation to other terms such as "multi-robot systems", "distributed robotic systems", "collective robotics", etc. For our purposes, we assume that a swarm meets the following set of requirements [swarm]:

- The robotic system should consist of large numbers of robots.
- The system should consist of homogeneous groups of robots, and that the number of robots in each group be large.
- The robots should be relatively simple and have simple capabilities such that the tasks tackled require the co-operation of the individual robots.
- The robots should only have localized and limited sensing and communication abilities.
- The swarm should be mobile – capable of moving.

Based on these system capabilities, we list the requirements for each bot in the swarm.

2.2 Requirements for each agent

Each bot must be mobile and able to change positions. It does not necessarily need to be fast, but it should be able to move based on the desired activity (form a cluster with other bots, transverse a ditch, search, etc.) We also assume that the movement is "purposeful". That is, the bot does not randomly move around the search space (either in a Brownian motion fashion or via random jumps based on a spring-like trigger) but instead can move to a new position in the search space based on the controlling algorithm. If the agents are stationary, then it is merely a wireless sensor network.

Each bot needs to know its own position. The position information can either be in an absolute coordinate system, via beacons or perhaps a Global Positioning System, or relative to other bots in the swarm. If position is unknown, then there is no way for the bot to know which direction to move and how far; it would be just randomly moving about the search space without coordinated behavior.

Each bot needs to be small in size. Of course, small is a relative term and we do not presume to place a quantifiable number on the max size of each bot. However, small size allows the swarm to be easily transported and, hopefully, will reduce the power consumption for each bot. (If a swarm has, say, 1000 bots and each bot is the size of shoe box, then it would take a small trailer, 220 ft^3 = 6.2 m^3, to transport just the bots.) Plus, small bots will be able to maneuver around/through clutter and small areas such as ductwork better than larger robots. As technology continues to advance and improvements are made in nano-technology, then we expect that the bots will naturally progress to smaller and smaller sizes. Presently, processor systems the size of fingernails are available [science 2003] – which includes the wireless communications package and memory – which makes bots the size of beetles possible.

The small size of each bot places limits on other components. The bot will have a small, and hence, simple processor with limited memory and a small power source. The simple processor will thus require algorithms that require simple decision making and processing capability. Because of the

small power source, the bot will have to be power efficient and/or be able to recharge its batteries.

Each bot will require a sensor or sensors to monitor environment. The type of sensor used will be based on the type of interaction that is desired with the environment. If the user wants the bot to interact with other bots and avoid obstacles, then sensors are required that can sense other bots and detect boundaries. If the user wants to search, then the bot must have a sensor that is able to detect what is being monitored (chemical sensor, radiation sensor, etc).

Each bot must be able to communicate with the other bots. If there is no communication, then the swarm becomes a group of individual agents and there is no advantage to using a swarm other than just increasing the statistical probability of "hitting" the target. Bot communications do not have to be wireless (communications could occur when 2 bots are joined together) but it simplifies the process.

TABLE 1: EXPECTED HARDWARE CAPABILITIES FOR EACH AGENT/BOT IN THE SWARM

Able to move
Can determine position
Small in size
Simple processor
Power efficient
Able to communicate with other bots
Able to monitor environment

In addition to the above requirements, there are many desired attributes of each agent. For example, it is desirable if each bot is (a) able to be mass produced, (b) interchangeable, and (c) disposable or, at least, replaceable. We note that none of these characteristics is absolutely essential to building a useful swarm of agents. Researchers may design swarms without one or more of these characteristics.

3.0 Applicability of evolvable hardware

3.1 Advantages of evolvable training algorithms

There needs to be an algorithm that can guide/control the bots as they perform the desired operation. It would be very inefficient to just turn on the bots and then hope that they "get lucky" and accomplish the mission. To derive the control algorithm, evolutionary algorithms offer several advantages over traditional, manually-designed controllers for swarm applications. The control algorithm itself is not (necessarily) an evolutionary algorithm but an evolutionary algorithm is used as a "learning algorithm" to obtain or refine the control algorithm. The advantages of using an evolutionary learning algorithm are listed in Table 2 and discussed in the following paragraphs.

Table 2: Capabilities of evolvable algorithms that make them amenable to swarm applications.

Able to derive a controller for unknown environment
Able to evolve swarm behavior and not just individual behavior
Possible for swarm to learn/adapt to changing environment
Can find solutions in complicated applications
Allow for unique or unexpected solutions
Able to adapt to non-identical bots/agents

First, it is difficult to design manually a controller when the physical environment is unknown ahead of time or when the physical environment changes over time. Most real life applications of swarm systems will be in areas where the user does not have perfect knowledge of the physical layout. Either the building/area has not been scouted or there could be clutter due to fallen walls, overturned furniture, and people. In addition, the swarm environment could be changing. The bots will move and thus change their relative proximity to each other (distance, nearest neighbors change) and the target of the search may change (increase or decrease in intensity, move in location). Evolutionary algorithms can exploit the richness of possible solutions encountered in the agent-environment interactions. In a multi-bot system, these dynamic/changing aspects are enriched not only by the presence of multiple bots, but also by the possibly changing physical environment [Dorigo 2004]. Generally, these aspects are difficult to exploit using manually-designed controllers. Evolutionary algorithms can take advantage of these dynamic system properties to derive effective controllers.

Second, there is a need to evolve group behavior and not an individual robot controller. Evolutionary algorithms bypass the problem of decomposition at both the level of finding the mechanisms that lead to global behavior and at the level of implementing those mechanisms in a bot controller. In fact, they rely on the evaluation of the system as a whole; that is, on the emergence of the desired global behavior [Dorigo 2004].

Third, evolvable hardware offers the possibility that the swarm can learn or get better over time. The swarm can continually monitor the fitness value and thus improve performance. For example, in a search scenario, the individual bots are monitoring, say, radiation intensity, and can thus track the highest value and close in on the source. Or, the source may move over time and the swarm can continue to track it.

Fourth, evolutionary techniques have shown the capability of finding the peak or optimum point in convoluted or difficult search environments [Eberhart 1995, Parsopoulos 2004, Hendtlass 2006]. The swarm application can be thought of as a search in a complicated environment. For example, the unknown and perhaps cluttered physical environment is the search space and the swarm needs to find the "target". That target can either be a physical target, as is the case for search applications, or it could be a particular desired swarm

behavior among the range of possible swarm behaviors. Thus, we expect that evolvable techniques will find solutions in complicated swarm applications as well.

Fifth, evolvable hardware techniques also allow for the possibility that the swarm may find a unique solution to a problem that is not expected or predicted by human experts. In many traditional EHW applications, the evolutionary algorithm found a solution that was much different than the one by human-expert designers [Lohn 2004]. Thus, evolvable techniques may find a solution to a swarm problem that utilizes group interactions or other aspects that are unexpected.

3.2 Variances among the bots

We have assumed that each agent has identical hardware characteristics to every other agent. This assumption simplifies the programming of the swarm. However, the "identicality" assumption is not a limitation on using an EHW approach. The swarm can still be programmed using evolutionary techniques even if there are some differences among the bots. For example, bots may have different mobilities due to different hardware (dissimilar wheel sizes or axles) or due to the fact that one battery has run down and thus delivers a different voltage to the drive motors. More substantive differences such as different sensors or different processor/memory setups can also be accounted for within a evolutionary learning environment. The evolutionary learning system will gravitate toward the good solutions, independent of what type of hardware enhancements or deficiencies are present.

4.0 Current evolvable hardware approaches to swarm applications

Some researchers have begun to apply evolvable algorithms and EHW techniques to swarm robotic applications. This section looks at two different approaches.

4.1 Evolve individual controllers

One approach researchers are taking to apply EHW principles to swarm intelligence is to evolve individual controllers for each agent/bot. Dorigo et al. [Dorigo 2004] and Pugh et al. [Pugh 2005] use simulations to derive a good controller for each bot and then load that controller onto the hardware bot for actual experiments. Dorigo et al. use a Genetic Algorithm to derive the weights for a neural network that controls the motors on the bot. The evolvable controller allows the user to program the swarm to perform a variety of functions such as cluster together, cross a ditch, and search.

Pugh et al. [Pugh 2005] use a modified Particle Swarm Optimization algorithm to derive the weights for a neural network controller for the bots. They focus on the performance of the algorithm/bots in noisy environments. The noise is due to sensor and actuator noise or local (mis-) perception of the bots. Their fitness function was designed to avoid obstacles in the search space.

When training a controller, researchers can either have the same weights (controller) for each bot or different weights for each bot. Different weights in each bot allow the system to test several different controllers in "parallel", but it leads to the credit assignment problem [Pugh 2006]. If a behavior is being learned and each robot is evaluating a different controller, the individual bot does not know whether a good/bad fitness score was due to its own performance or to that of other bots. This effect is more pronounced in cases where robots do not explicitly share their intentions through communications channels.

4.2 Evolve the swarm

Another EHW approach to the swarm intelligence problem is to let each bot/agent in the swarm be one particle or member of the evolutionary algorithm. Thus, each bot is a "member" or candidate solution of the evolutionary algorithm. In [Hereford 2006], each bot is one particle of the Particle Swarm Optimization (PSO). Thus, each bot takes measurements and moves based on the PSO update equation. In this approach, all the calculations are done "locally", that is, on each local bot. The only data that is potentially needed from other bots is the value and location of the global best – the best measurement found so far. Thus, there are no communications unless one of the bots finds a point in the search space that is better than any point found up to that time during the search.

Simulation results show that this approach is a very good way of coordinating simple bots for a search operation. For target locations in the middle of the search space, bots find the target 99 or 100 % of the time. In addition, the algorithm appears to be scalable to large numbers of bots since the communications requirements do not increase as the number of bots is increased.

5.0 Impact on evolutionary algorithms/evolvable hardware

5.1 Evolutionary algorithm characteristics

As mentioned in section 3, evolutionary algorithms offer several advantages for deriving control algorithms for swarm applications. To meet the needs of evolvable hardware swarms, the evolutionary algorithms must have several characteristics. The characteristics are summarized in Table 3. We are assuming that the evolutionary algorithm is embedded into the swarm itself. That is, there is not an external agent that programs the swarm off-line and then the bots are released. Instead, all of the training is done within the swarm itself.

TABLE 3: CHARACTERISTICS OF EVOLUTIONARY ROBOT FOR TRAINING SWARMS

Distributed – no central agent
Computationally simple
Minimum amount of bot-bot communications
Adapt to changing conditions
Allow for asynchronous bot operation
Number of hardware elements may change during operation
Able to handle uncertainty and noise
Evolve behavior of entire swarm
Contiguous movement of bots – no hyperspace jumps

1. There needs to be an algorithm that does not require a central agent or processor – the algorithm needs to be distributed among all of the bots. If the algorithm is located in one robot, then the system will fail if that robot fails. This provides a natural fault tolerance for the system. The swarm/system will continue to function even if one or several bots become damaged or lose power. Also, there would be a lot of communications among the robots if each robot had to wait for movement commands from a central source. Thus, the control algorithm and the learning algorithm need to be de-centralized.

2. The control algorithm and the learning algorithm should be computationally simple. The processor is small, has limited memory, and there is a limited power source (a battery) so the processor needs to be power efficient. Therefore, the processor will be a simple processor. The control algorithm needs to be tailored to such a processor. Difficult or time-consuming operations such as FFTs and iterative solution methods will take a long time and not lead to a quick, effective search. Even common mathematical operations such as trigonometric functions and square roots are time-consuming to implement on simple processors, so they should be minimized.

3. The control and learning algorithms should have a minimum amount of communications among the robots. The algorithm needs to be scalable from one robot up to 10's, 100's, even 1000's of robots. The upper limit on the number of robots will most likely be set by the communication links among the robots. If each robot has to wait for information from all other robots, then the system will break down as the number of robots increases. Instead, there needs to be a way to share information among the robots without requiring lots of communication traffic. For example, we do not want an algorithm where every bot communicates to every other bot because then the number of communications links is equal to N(N-1)/2, where N is the number of bots. As N becomes large, this is an $O(N^2)$ process and thus severely limits the scalability of the system.

4. The control algorithm needs to be able to adapt to changing conditions. The bots will encounter environments and circumstances that will change over time. The evolutionary algorithm needs to be able to continually monitor the overall fitness and update the bots as necessary. This might be as simple as continuing to monitor the environment to improve a search or adjust a parameter or as complicated as initiating a new training cycle if there is a cataclysmic event.

5. The evolutionary algorithm needs to allow for asynchronous operation of the agents. It is logistically difficult to have a synchronous clock for all of the bots because (a) there is no centralized agent to generate a clock signal, (b) we want to reduce the number of communications links and a clock would require a lot of communications packets and (c) the movement of the bots will change the distances between the bots and thus affect the timing delays. Approaches based on Time-Triggered Architecture (TTA) concepts may be appropriate but they require a minimum of four sources and would require a mechanism for distribution which would raise the communication overhead. Therefore, the evolutionary algorithm must allow the bots to function semi-independently with only intermittent coordination with the rest of the bots.

6. The evolutionary algorithm needs to be able to work with different amounts of hardware elements. The number of hardware elements in the swarm will vary from situation to situation. In fact, the swarm size will probably change during the operation as some bots fail and other bots join. Therefore, the evolutionary needs to be handle a changing swarm size.

7. The evolutionary algorithm needs to be able to handle uncertainty and noise. This is true for any swarm control algorithm.

8. For best performance, the evolutionary learning algorithm needs to evolve group behavior and not individual robot controller. The main reason is because of the credit assignment problem, where a good fitness score can occur even if an individual bot does poorly. The second reason is the time required to transfer an effective evolved control algorithm to all of the other agents in the swarm. If one bot has the best controller, then that must be communicated to all of the other bots which greatly increases the learning time.

9. The control algorithm must allow for contiguous movement of the robots. There are search algorithms that work well in simulation (e.g., genetic algorithms) but they require "step" changes in the solutions at each iteration. This would not be feasible in this application.

5.2 Restraints on evolutionary algorithms

The list of characteristics in section 5.1 narrows the possible evolutionary algorithms that can be used to program a controller for a swarm. The primary conclusion is that the generic generation-based Genetic Algorithm (GA) is not appropriate. If a user tries to have each bot in the swarm be one member of the GA, then the crossover operation will require lots of communication packets between parent bots. In addition, crossover and mutation cause each member of the population to "jump" to different places in the search space, rather than allow continuous movement of the bot.

Even if a GA is used to monitor the output of the swarm as a whole, there are issues. It requires a central processor to store and rank the fitness for each generation and it requires all of the bots to operate synchronously (so that fitness can be calculated for each generation). Thus, we conclude that a standard GA embedded in the swarm will not be effective at deriving a good controller.

Variants of the generic GA, however, may be appropriate for swarm learning. An evolution strategy (ES) that requires only mutation and no crossover [Beyer 2002] will help with the scalability/communication problem. An evolutionary strategy may also allow for asynchronous operation since each member of the GA could track its own fitness. There is still the problem of contiguous bot movement, even with the ES. However, a modified ES could perhaps overcome this hurdle.

Other types of evolvable algorithms appear to be more useful for deriving control algorithms for swarms. The Ant Colony Optimization (ACO) algorithm is based on swarm behavior – ants [Dorigo 2006]. It has proven successful at optimization problems but has not yet been applied to program an artificial swarm of bots.

The PSO has shown promise as an optimization technique [Eberhart 1995]. Hereford has already shown decent results using a PSO as a search algorithm for a swarm [Hereford 2006]. The PSO focuses on system-level output (the global best or best solution found so far) and can be modified to minimize the number of bot-bot communications. Plus, it can easily incorporate different numbers of bots in the swarm via the concept of neighborhoods. There is a question, however, as to how to extend the PSO to more complicated swarm operations such as coordinated movement, bridge building, etc.

6.0 Conclusions

This paper looked at the applicability of using evolvable hardware principles to develop control strategies for swarm applications. There are a number of advantages of using evolvable algorithms to program the swarm: evolutionary algorithms are able to find unique solutions in noisy, changing environments and are able to evolve swarm behavior and not just individual behavior. It appears that genetic algorithms are a bad fit for evolving the swarm behavior because of their need of a central processor and the "jump" changes required by the agents. (However, genetic algorithms have been shown to be useful for evolving individual bot behavior.) The PSO appears to have the most promise to derive an effective control strategy for swarm behavior.

References

[Beyer 2002] H. Beyer, H. Schwefel, "Evolution strategies: A comprehensive introduction", Natural computing, vol 1, pp. 3 – 52, 2002.

[Lewis 1992] M. Lewis and G. Bekey, "The behavioral self-organization of nanorobots using local rules", Proc. 1992 IEEE/RSJ Int. Conf. Intelligent Robots and Systems, pp. 1333-1338, 1992.

[Dorigo 2004] M. Dorigo, V. Trianni, E. Sahin, R. Gross, T. Labella, G. Baldassarre, S. Nolfi, J. Deneubourg, F. Mondada, S. Floreano, L. Gambardella, "Evolving self-organizing behaviors for a swarm-bot", Autonomous Robots, vol 17, pp. 223-245, 2004.

[Dorigo 2006] M. Dorigo, M. Birattari, T. Stutzle, "Ant colony optimization: Artificial ants as a computational intelligence technique," IEEE Computational Intelligence Magazine, vol. 1, pp. 28 – 39, November 2006.

[Eberhart 1995] R. Eberhart, J. Kennedy, "A new optimizer using particle swarm theory", Proceedings of the sixth international symposium on micro machine and human science, Japan, pp. 39-43, 1995.

[Greenwood 2004] G. Greenwood, D. Hunter, E. Ramsden, "Fault recovery in linear systems via intrinsic evolution", 2004 NASA/DoD Conference on Evolvable Hardware, Seattle, WA, Zebulum et al. (ed.), 2004, pp. 115 – 122.

[Hendtlass 2006] T. Hendtlass, "A particle swarm algorithm for complex quantized problem," 2006 Congress on Evolutionary Computation, Vancouver, BC, July 2006.

[Hereford 2005] J. Hereford, C. Pruitt, "Robust sensor systems using evolvable hardware", 2004 NASA/DoD Conference on Evolvable Hardware, Seattle, WA, Zebulum et al. (ed.), 2004, pp. 161 – 168.

[Hereford 2006] J. Hereford, "A distributed Particle Swarm Optimization algorithm for swarm robotic applications", 2006 Congress on Evolutionary Computation, Vancouver, BC, pp. 6143 – 6149, July 2006.

[Higuchi 1999] T. Higuchi, M. Iwata, D. Keymeulen, H. Sakanashi, M. Murakawa, I. Kajitani, E. Takahashi, K. Toda, M. Salami, N. Kahihara, N. Otsu, "Real-World applications of analog and digital evolvable hardware", IEEE Transactions on Evolutionary Computation, Vol. 3, No. 3, pp. 220 – 235, September 1999.

[Koza 2003] J. Koza, M. Keane, M. Streeter, W. Mydlowec, J. Yu, G. Lanza, Genetic Programming IV: Routine Human-Competitive Machine Intelligence, Kluwer, 2003.

[Lohn 2004] J. D. Lohn, D. S. Linden, G. D. Hornby, W. F. Kraus, A. Rodriguez, S. Seufert, "Evolutionary Design of an X-Band Antenna for NASA's Space Technology 5 Mission," Proc. 2004 IEEE Antenna and Propagation Society International Symposium, Vol. 3, pp. 2313-2316, 2004.

[Parsopoulos 2004] K. Parsopoulos, M. Vrahatis, "On the computation of all global mimizers through particle swarm optimization", IEEE Transactions on Evolutionary Computation, vol. 8, pp. 211 – 224, June 2004.

[Pugh 2005] J. Pugh, A. Martinoli, Y. Zhang, "Particle swarm optimization for unsupervised robotic learning," Proceedings of the 2005 Swarm Intelligence Symposium, pp. 92 – 99, June 2005.

[Pugh 2006] J. Pugh, A. Martinoli, "Multi-robot learning with particle swarm optimization," Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems (AAMAS), pp. 441 – 448, May 2006.

[science 2003] "Researchers create wireless sensor chip the size of glitter," Science Daily, http://www.sciencedaily.com/releases/2003/06/030616091423.htm, June 16, 2003.

[Stoica 2002] A. Stoica, R. Zebulum, M. I. Ferguson, D. Keymeulen, V. Duong, X. Guo, "Evolving circuits in seconds: Experiments with a stand-alone board-level evolvable system", 2002 NASA/DoD Conf. on Evolvable Hardware, July 2002, pp. 67-74.

[swarm] http://www.swarm-robotics.org/

[Thompson 1996] A. Thompson, I. Harvey, P. Husbands, "The natural way to evolve hardware", Proceedings IEEE International Symposium on Circuits and Systems, 1996, pp. 37 – 40.

[wiki 2006] http://en.wikipedia.org/wiki/Swarm_intelligence