Chapter 1

# EVOLVING SWARMING AGENTS IN REAL TIME

H. Van Dyke Parunak[1]
[1]*Altarum Institute*

**Abstract** An important application for population search methods (such as particle swarm optimization and the several varieties of synthetic evolution) is the engineering problem of configuring individual agents to yield useful emergent behavior. While the biological antecedents of population-based search operate in real time, most engineered versions run off-line. For some applications, it is desirable to evolve agents as they are running in the system that they support. We describe two instances of such systems that we have developed and highlight lessons learned.

**Keywords:** applications, real-time, emergence, agents, population-based search, evolution

## 1. Introduction

Research in the Emerging Markets Group of the Altarum Institute focuses on practical applications of swarm intelligence. We[1] is the result of the exploit the emergent system-level behavior exhibited by interacting populations of fairly simple agents to solve a wide range of real-world problems, including control of uninhabited air vehicles (Parunak et al., 2002, Sauter et al., 2005), sensor coordination (Parunak and Brueckner, 2003, Brueckner and Parunak, 2004), resource allocation (Savit et al., 2002), information retrieval (Weinstein et al., 2004), and prediction (Parunak et al., 2005), among others.

The central problem in engineering emergent behavior is determining the individual behaviors that will yield the required system-level behavior. The most

[1]The results described in this paper reflect the creative ideas and implementation skill of my colleagues, including Rob Bisson, Steve Brophy, Sven Brueckner, Paul Chiusano, Jorge Goic, Bob Matthews, John Sauter, Peter Weinstein, and Andrew Yinger.

promising techniques that we have identified are those drawing on techniques such as particle swarm optimization and various forms of synthetic evolution. We describe these techniques collectively as population-based search (PBS), since they use interactions among a population of searchers to solve a problem. It is philosophically reinforcing to our basic approach, and perhaps not coincidental, that these techniques themselves exemplify the emergent paradigm of deriving global results from local interactions.

This paper emphasizes two aspects of this approach: the elements of the population are individual agents rather than representations of the whole system, and the evolution takes place in real time, while the system runs. The first aspect has antecedents in the literature, but should be more widely explored. The second appears to be novel.

In Section 2, we summarize some other examples of agent-centered evolution in order to provide a context for our methods. Sections 3 and 4 discuss two examples from our work, using real-time agent-based evolution to solve a Configuration problem and a Fitting problem, respectively. Section 5 draws lessons from our experience and concludes.

## 2. Background

Evolutionary and particle swarm methods take their inspiration from natural agents that adapt in the same temporal space in which they are born, live, and die. Yet applications of these techniques differ from their metaphorical roots in two ways. First, many applications have little to do with computational agents, and instead focus on optimization of structures or functions that cut across individual agents, even when the domain naturally lends itself to an agent-based model. Second, even when PBS is applied to individual agents, most applications execute in a temporal space distinct from that occupied by the agents. That is, the PBS is a planning or configuration process that determines agent parameters off-line, for later deployment.

In this section we first distinguish agent-based applications from other approaches, then describe two broad uses of agent-based PBS, and consider some previous work on real-time agent-based PBS.

### Three Perspectives on PBS

It is useful to distinguish three different applications of PBS: structure optimization, function optimization, and agent optimization. While the three categories can readily be mapped into one another, each suggests a particular perspective on the problem. For many engineering problems, the agent perspective offers particular benefits.

*Structure optimization* includes spatial organization problems such as the traveling salesperson problem (TSP), layout of VLSI chips, or design of me-

chanical mechanisms. It also includes problems of temporal organization such as factory scheduling. Population-based search is typically applied to these problems by constructing a population whose members are complete candidate structures, and taking this approach encourages the practitioner to view the structure holistically. Indeed, the value of PBS for such problems is largely in overcoming the tendency to local sub-optimization that results from traditional mechanisms such as greedy search. Symbolic regression may be considered an instance of structural optimization in which the structure being manipulated is an abstract mathematical expression.

In *function optimization*, each member of the population is a vector that constitutes an argument to some mathematical function, and the objective of the search is to find a vector that yields a desired value for the function (such as an extremum or an inflection point). Effective application of PBS to such problems often requires adjustments to take advantage of the ordered nature of the domain of each allele (Corne et al., 1999). Reduction of an engineering problem to a mathematical function that needs to be optimized is the utmost in abstraction. While such abstraction can help develop general solutions that are applicable across multiple domains, it also makes it difficult to take advantage of domain-specific heuristics, which may not readily be cast as closed-form mathematical expressions.

*Agent optimization* is a natural way to apply PBS to domains that are effectively modeled as sets of interacting autonomous agents. These domains may be engineered or natural.

Engineered domains that lend themselves to multi-agent modeling include processing information from networks of sensors, coordinating the movement of multiple vehicles, retrieving information from large collections of documents, and managing extended communication networks. Agent architectures are particularly attractive for engineering problems when the domain consists of discrete elements that are distributed in some topology, where central control is difficult or impossible, and whose environment is changing dynamically (so that adaptiveness is more important than reaching a steady-state optimum).

Natural domains that lend themselves to multi-agent modeling include many biological systems, ranging from predator-prey ecologies and insect colonies to human communities.

In both cases, the behaviors of these systems emerge from the interactions of their parts, and a central problem in configuring them is determining the behavior of individuals that will yield the desired overall system behavior. In applying PBS from this perspective, each member of the population is a candidate for a single agent in the system. Taking an agent-centered perspective on PBS aligns well with the natural modularity of such system.

Recently, agent-based mechanisms such as ant colony optimization (ACO) have been applied to structure optimization (e.g., TSP and scheduling), and

population search has been used to tune these mechanisms. It would seem most natural to search over populations of individual agents (White et al., 1998). However, these mechanisms include some system-wide parameters (such as the number of agents used), and so population members are sometimes defined at the level of the system rather than the individual agents (Botee and Bonabeau, 1998).

This latter approach violates the distinction between the individual agents and their environment (Weyns et al., 2004), a distinction that is important from the point of view of engineering effectiveness. On the one hand, it is usually appropriate to consider issues such as the number of agents and the physics of pheromone evaporation as part of the environment. Though they may emerge from interactions among the agents, no single agent can change them. On the other hand, deposit rates and sensitivity to different pheromones clearly pertain to individual agents, and it makes sense to model them in the chromosomes of each agent. If one wishes to explore the total space of both agent and environmental variables, it would be cleaner to co-evolve the agents and the environment as two different populations. (The whole area of engineering environments for agents is quite new in the agent software community, and we do not know of anyone who has explored the pros and cons of these alternative ways of applying PBS to such systems.)

## Varieties of the Agent Approach

We are not by any means the first to apply PBS to individual agents in order to improve their collective behavior. Two areas where this approach has been widely applied are robotics and biology.

Biologists use PBS (particularly its genetic varieties) retrospectively, in at least two distinct ways. *Ethologists* seek to discover possible processes by which various animal behaviors have evolved. The actual behavior of the agent is known, and in fact provides the standard against which the fitness of an evolved agent is evaluated. Examples of work in this very large field include the development of communications (Quinn, 2001, Steels, 2000), the evolution of cooperation (Riolo et al., 2001), and the development of foraging (Panait and Luke, 2004), to name only a few. *Ecologists* are more concerned with the overall patterns of interactions among multiple agents (e.g., food webs and population dynamics), without so much focus on the individual behaviors. These examples can be viewed as attempts to fit a model to observed agent and system behaviors, respectively.

Roboticists have long used PBS prospectively, to find behaviors (equivalently, control laws) that satisfy various functional requirements. A variety of representations have been adopted for programming the behavior of these agents, including GP-like higher-order operations (Brooks, 1992), tropistic ex-

ecution engines (Agah and Bekey, 1996), and neural networks (Harvey et al., 1992). These examples can be viewed as configuration problems, seeking to configure the agent's behavioral engine to achieve desired outcomes.

Most of these instances run "off-line". That is, the timeline within which the PBS operates is disjoint from the timeline within which the system being studied or designed operates. While ubiquitous among practitioners of PBS, off-line search is at variance with the natural processes that inspired these mechanisms. Our examples illustrate the potential of on-line search (conducted while the system itself operates).

## Examples of Real-Time PBS

A few examples of PBS have been published[2] in which evolution takes place as the system runs, and merit comparison with our approach.

Nordin and Banzhaf (Nordin and Banzhaf, 1997) use GP to evolve the controller for a Khepera robot to improve its ability to avoid obstacles. The evolution runs as the robot operates, but the objective is to evolve a single algorithm that can handle various inputs, not to vary the algorithm to accommodate environmental changes. While the system is learning (40-60 minutes in one version, 1.5 in another), the robot does not successfully avoid obstacles. Dadone and VanLandingham (Dadone and VanLandingham, 1999) take a similar approach in evolving a controller for a chemical plant. Each member of the population is given a chance to run the plant while its fitness is evaluated, and when every member of the population has been evaluated, a new population is generated. These systems deal only with a single entity (the robot or the controller), and are not concerned with developing appropriate emergent behavior from a system of agents.

Spector and colleagues (Spector et al., 2005) evolve the behaviors of a population of simulated mobile entities living in 3-d space, whose behavior evolves as they execute. They describe two systems. In one, the agents' behavior is a version of Reynolds' flocking behavior (Reynolds, 1987), and the genotype is a list of coefficients for the various vectors that are summed in that algorithm. In the other, it is a program that yields a flocking algorithm. This work exhibits emergent group behavior across the population of agents. However, that behavior is achieved over the course of the run. The dynamics of the environment are handled by the adaptive capabilities of the flocking algorithm that is evolved, not the ongoing adaptation of that algorithm by evolution.

These examples are robotics applications. They develop control instructions for robots, like the more common off-line applications of PBS, but do so fast

[2]We are grateful to participants in GPTP2005 and other reviewers for suggesting a number of examples, of which these are illustrative.

enough to be deployed on the robot as it executes. They both rely on adaptive mechanisms in the evolved behavior to handle a changing environment, rather than using evolution itself as the main adaptive mechanism.

Dynamic Flies (Boumaza and Louchet, 2001) is a vision processing algorithm for obstacle avoidance. A population of points in three-space evolve to fit their coordinates in the robot's visual field to occupy the surfaces of obstacles. The fitness function is based on the observation that the pixels in the vicinity of a fly on a surface will vary relatively little from two different vantage points, compared with the pixel neighborhoods of flies that are in free space. The flies influence one another, in that the fitness is adjusted to penalize grouping. The aggregate fitness of the flies in each cell of a square lattice that maps the robot's environment generates a repulsive field to guide the robot. This application is like ours in both dimensions. It is truly emergent, generating a system-level behavior (obstacle avoidance) from the evolution of individual flies. Also, it uses evolution as its adaptive engine. However, the individual flies, consisting only of the coordinates of a point in three-space and a fitness value, have no intrinsic behavior, and fall below the threshold of what most researchers would consider an agent. While the application as a whole is robotic, the actual adaptation of the flies to the surfaces of obstacles in the environment can be considered a retrospective or fitting application of real-time PBS, since the flies are evolving to provide a model of an exogenous feature of the environment.

The evolving entities in classifier systems (Booker et al., 1989) and artificial immune systems (Forrest et al., 1997), unlike Dynamic Flies, do have (very simple) behaviors associated with them, and could be considered minimal agents. These systems exhibit real-time PBS.

Li and colleagues (Li et al., 2000a)(Li et al., 2000b) evolve the strategies of agents playing the minority game, a simple model of emergent market dynamics. The agents' fitnesses are evaluated as the game proceeds, but the population is updated all at once every 10,000 time steps, rather than permitting each agent to evolve asynchronously with respect to the others, as in nature.

## 3. A Configuration Application

The most direct application of PBS to swarming systems is finding configurations of the individual agents so that their interactions yield the desired system-level behavior. We illustrate this application in the context of ADAPTIV (Adaptive control of Distributed Agents through Pheromone Techniques and Interactive Visualization), a system developed for planning flight paths for uninhabited robotic vehicles (URV's). This system uses a digital analog of insect pheromone mechanisms to guide vehicles around threats and toward targets.

Our implementation of digital pheromones has four components.

1 A distributed network of *place agents* maintains the pheromone field and performs aggregation, evaporation, and diffusion. Each place agent is responsible for a region of the physical space. In our simulation, we tile the physical space with hexagons, each represented by a place agent with six neighbors, but in principal both regular and irregular tiling schemes can be employed. Place agents ideally are situated physically in the environment using unattended ground sensors distributed over an area and connected to their nearby neighbors through a wireless network. They may also be located in a distributed network of command and control nodes.

2 *Avatars* represent physical entities. Red avatars represent the enemy targets and threats, while blue avatars represent friendly URVs. Blue avatars are normally located on the robot vehicle. The name "Avatar" is inspired by the incarnation of a Hindu deity, and by extension describes a temporary manifestation (a software agent) of a persistent entity (a robot vehicle).

3 Blue avatars create *Ghost agents* that wander over the place agents looking for targets and then continually building a path from the avatar to the target. The avatars and ghosts all deposit pheromones at their current locations.

4 Different classes of agents deposit distinct *pheromone flavors*. Agents can sense pheromones in the place agent in whose sector they reside as well as the neighboring place agents. The underlying mathematics of the pheromone field, including critical stability theorems, is described in (Brueckner, 2000).

Battlefield intelligence from sensors and reconnaissance activities causes the instantiation of Red avatars representing known targets and threats. These agents deposit pheromones on the places representing their location in the battlespace. The field they generate is dynamic since targets and threats can move, new ones can be identified, or old ones can disappear or be destroyed. A blue avatar representing a URV is associated with one place agent at any given time, the place agent within whose physical territory the URV is currently located. It follows the pheromone path created by its ghost agents.

Ghosts initially wander through the network of place agents, attracted to pheromones deposited by targets and repelled by threat pheromones. Once they find a target, they return over the network of place agents to the avatar, depositing pheromones that contribute to building the shortest, safest path to the target. The basic pheromone flavors are *RTarget* (deposited by a Red target avatar, such as the Red headquarters), *RThreat* (deposited by a Red threat avatar, such as an air defense installation), *GTarget* (deposited by a ghost that has

encountered a target and is returning to its blue avatar, forming the path to the target), and *GNest* (deposited by a ghost that has left the blue avatar and is seeking a target).

A ghost agent chooses its next sector stochastically by spinning a roulette wheel with six weighted segments (one for each of its six neighbors). The size of each segment is a function of the strength of the pheromones and is designed to guide the ghost according to the algorithm above. We experimented with several different forms of the function that generates the segment sizes. Evolution of such a form using genetic programming would in itself be a useful exercise. In our case, manual experimentation yielded the form (for outbound ghosts):

$$F_n = \frac{\theta \cdot RTarget_n + \gamma \cdot GTarget_n + \beta}{(\rho GNest_n + \beta)(Dist_n + \varphi)^{\delta + \alpha(RThreat_n + 1)} + \beta}$$

$F_n$ is the resultant attractive force exerted by neighbor *n* and *Dist* is the distance to the target if it is known. Table 1-1 lists the tunable parameters in the equation and the effect that increasing each parameter has on the ghost's behavior. Though this table provides general guidance to the practitioner, in practice, the emergent dynamics of the interaction of ghost agents with their environment makes it impossible to predict the behavior of the ghosts. Thus tuning the parameters of this or any pheromone equation becomes a daunting task. Even if a skilled practitioner were able to tune the equation by hand, the system would still be impractical for end users who don't think of their problem in terms of $\alpha$, $\beta$, and $\gamma$. This observation led us to investigate the possibility of using evolutionary methods to tune the parameters of the equation.

*Table 1-1.* Tunable Parameters and their Effects on Ghosts.

| | |
|---|---|
| $\alpha$ | Increases threat avoidance further from the target |
| $\delta$ | Increases probability of ghosts moving towards a known target in the absence of RTarget pheromone |
| $\gamma$ | Increases sensitivity to other ghosts |
| $\rho$ | Increases ghost exploration (by avoiding GhostNest pheromone) |
| $\theta$ | Increases attraction to RTarget pheromone |
| $\beta$, $\phi$ | Avoids divison by zero |

We explored several PBS algorithms on the problem of defining ghost parameters, including three varieties of evolution strategies (ES) and a genetic algorithm (GA). Details on these approaches and the scenarios on which they were tested are described in our original paper (Sauter et al., 2002). In all cases, ghosts have a fixed lifetime. Within this lifetime they first execute a search, and then breed sexually until they die. Thus ghosts that complete their search

faster have longer to breed, and generate more offspring. The GA and one of the ES approaches took account of threats that the ghost encountered during its search, and the GA also rewarded the ghost for the value of the target that it discovered. In all cases, as each ghost returns to the URV, it is evaluated and selectively participates in generating subsequent generations of ghosts. Thus the ghosts being emitted by the avatar are evolved in real time, as the system runs.

One could envision evolving the parameters for the ghosts off-line. The success of this approach would depend on the stability of the environment. In the test examples reported here, the environment was static, and we were exploring the speed with which the evolutionary process converged, and the resulting performance achieved. However, on different runs we gave the system different scenarios, to which it developed distinct parameters. In a real-world application, scenarios are not static, and a set of parameters evolved for one scenario would not function well on another. By adapting the parameters in real time, we can accommodate dynamic changes in the environment.

Figure 1-1 shows the performance of the system, measured by the strength of the GTarget pheromone adjacent to the avatar (and thus available to guide it). The left-hand plot shows two benchmarks. The “Hand Tuned” line shows the behavior of a set of parameters derived by manual experimentation. The “Random” line shows the behavior when ghosts are generated with small random excursions around the hand tuned values.

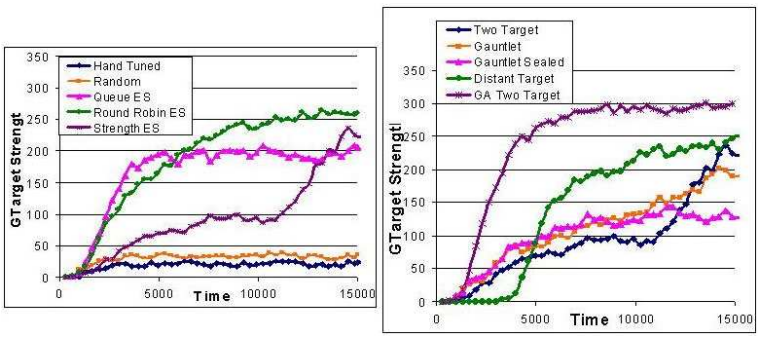


*Figure 1-1.* Performance of PBS on path planning. Left: comparison of ES’s on Two Target scenario. Right: comparison of Strength ES on various scenarios, and GA on Two Target scenario.

The left-hand plot shows that all three versions of the ES outperformed the hand tuned and random configuration by an order of magnitude. The Strength ES takes into account the damage suffered by the ghost in simulated encounters with threats, and while it takes longer to converge, it outperforms the other ES

approaches on a wider range of scenarios. The slight superiority of the random to the hand tuned configuration is an interesting illustration of the value of stochasticity in breaking symmetries among swarming agents and permitting more effective exploration of the environment.

The right-hand plot compares the Strength ES on four different scenarios with the GA on one of them.

This system has striking similarities with the Dynamic Flies system, though each was developed without knowledge of the other. In both cases, interacting entities continuously evolve under the influence of the environment, and generate a field that guides the movement of a physical vehicle. Table 1-3 makes this comparison explicit.

*Table 1-3.* Comparison of ADAPTIV and Dynamic Flies.

| **Feature** | **ADAPTIV** | **Dynamic Flies** |
|---|---|---|
| Entities | Ghosts | Flies |
| Environmental Influences | Targets and Threats | Obstacles |
| Generated Field | GTarget pheromone | Aggregate Fly fitness |
| Physical Agent | URV | Robot |

The systems differ in their specificity and their dynamics. Both of these differences reflect the distinction between ADAPTIV's ghosts (which are real, though simple, agents, with autonomous behaviors) and the flies (which are simply the coordinates of points in three-space).

- **Specificity**.—Dynamic Flies specifically supports processing of stereo vision for obstacle detection. The only output from the flies to the rest of the system is their fitness, linking the evolutionary process directly to the obstacle avoidance behavior. In ADAPTIV, evolution adjusts the characteristics of the ghosts, whose impact on the rest of the system is through a digital pheromone that is part of a larger pheromone vocabulary. Thus a ghost has a richer set of inputs than a fly (including not only pheromones from targets and obstacles but also pheromones from other ghosts), and the system can reason about attractors as well as repellers.

- **Dynamics**.—The Dynamic Flies system has no memory. A fly repels the vehicle only while it is actually at a location, and only in proportion to its current fitness. This feature is appropriate for the specific obstacle avoidance application for which the system is designed. The ADAPTIV architecture supports more general geospatial reasoning, including the need to maintain a memory of a threat or target that may not currently be visible. Because pheromones are distinct from the agents that deposit

them, they can persist in a location after the agent has moved on, or they can vanish almost immediately, depending on the setting of the evaporation rate associated with a given pheromone flavor.

## 4. A Behavior Fitting Application

Our second example addresses the problem of predicting the future behavior of soldiers in urban combat, based solely on their observed past behavior. We assume that an individual soldier's behavior is a function of his[3] individual personality as well as his interactions with other soldiers and with the urban environment. Prediction in this highly nonlinear system merits comparison with prediction in nonlinear systems without the social and psychological aspects of combat (Kantz and Schreiber, 1997). The general approach in such systems is to extrapolate future behavior using functions fitted to the recent past. While the nonlinear nature of the systems may lead to divergence of trajectories over time, continuously refreshing the fit and limiting the distance of the projection into the future can yield useful predictions (Figure 1-2).

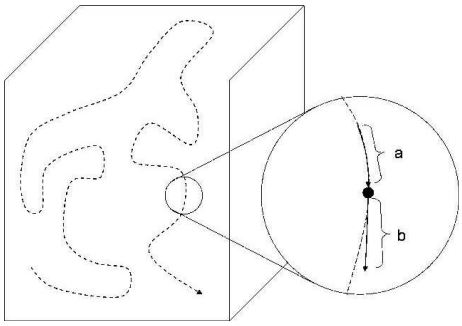


*Figure 1-2.* By constantly updating a fit of the system's trajectory through state space on the basis of the recent past (a), one can generate useful predictions a short distnce into the future (b).

Historically, this approach has been applied to systems that can be described analytically, permitting a functional form to be fit to recent behavior. We have extended this approach to entities, such as soldiers, whose behavior cannot readily be fit using analytical techniques. The basic approach is to represent the entity by a software agent whose behavioral parameters are fit using PBS. We call this approach "Behavioral Emulation and Extrapolation," or BEE.

BEE must operate very rapidly, in order to keep pace with the ongoing evolution of the battle. Thus we use simple agents coordinated using pheromone mechanisms similar to those described in our configuration example.

[3]We use the masculine gender generically.

Figure 1-3 explains BEE further. Each active entity in the battlespace has an avatar that continuously generates a stream of ghost agents representing itself. The ghosts' behavioral parameters are selected from distributions to explore possible intentions of the entity they represent. Thus BEE mimics at the agent level the nonlinear track analysis outlined in Figure 1-2.

Ghosts live on a timeline indexed by $\tau$ that begins in the past at the insertion horizon and runs into the future to the prediction horizon. The avatar inserts the ghosts at the insertion horizon. The ghosts representing different entities interact with one another and with the terrain. These interactions mean that their fitness depends not just on their own actions, but also on the behaviors of the rest of the population, which is also evolving. Because $\tau$ advances faster than real time, eventually $\tau = t$ (actual time). At this point, the ghosts are evaluated based on their locations compared with the entity represented by their avatar.

The fittest ghosts have two functions. First, they are bred and their offspring are reintroduced at the insertion horizon to continue the fitting process. Second, they are allowed to run past the avatar's present into the future. Each ghost that is allowed to run into the future explores a different possible future of the battle, analogous to how some people plan ahead by mentally simulating different ways that a situation might unfold. Analysis of the behaviors of these different possible futures yields predictions.

This entire process runs continuously, in real time, as the system monitors the environment. Ghosts are evolving against the world as its state changes. As in the Dynamic Flies system, the evolution of the swarming agents is what enables them to track a dynamic environment. Unlike the Dynamic Flies, but like ADAPTIV, the output of the ghosts in BEE is not an immediate by-product of the evolutionary process (the fitness of the agents), but a second-order phenomenon produced by the agents (their behavior as they run into the future).

The personality of each ghost includes four categories of information, all represented as scalars (Parunak et al., 2005).

1 *Desires* are anticipated future state of the world toward which the agent is positively disposed. We have defined a basic set of desires relevant to the combat scenario. An agent's goals are considered to be stable over the time horizon that we are considering. They may be mutually exclusive, since they have no effect on an agent's actions until they are instantiated as goals in the face of environmental information. A desire might be "occupy key sites."

2 *Goals* are selected by the agent from among its desires based on its current state and recent history, and it chooses its actions in an effort to accomplish the goals. Unlike desires, the set of goals held by an agent at a given time are believed by the agent to be consistent with one another, and may change over the time horizon of the battle. A goal instantiated from

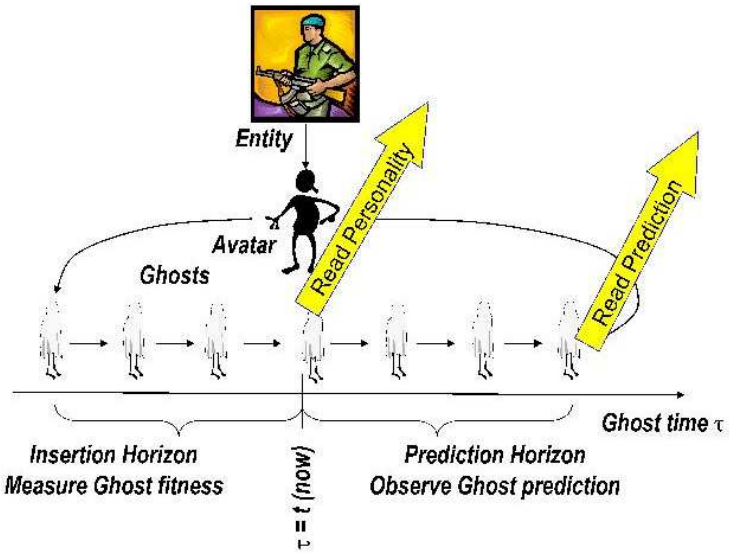


*Figure 1-3.* Behavioral Emulation and Extrapolation. Each avatar generates a stream of ghosts that sample the personality space of the entity it represents. They are evolved against the observed behavior of the entity in the recent past, and the fittest ghosts then run into the future to generate predictions.

the "key sites" desire might be "occupy building 34 by time = 1520." The agent continually reviews its goals to ensure their consistency with the current state of the world. If it discovers that two goals are inconsistent with one another, it will drop at least one of them.

3 *Emotions* are defined following the OCC model (Ortony et al., 1988) as "valenced reactions to events, agents, or objects." Emotions vary based on the events, agents, or objects that the agent experiences, and modulate its analysis of which goals to instantiate over time. For example, an event of being attacked will raise the level of an agent's fear emotion.

4 *Dispositions* reflect an agent's tendency to adapt a given emotion. For example, an agent with a high level of the "cowardice" disposition will experience a faster rate of increase of fear in the presence of an attack than an agent with a low level of this disposition. Dispositions are assumed to be constant over the time horizon in question.

Figure 1-4 shows how these four personality elements interact with one another and with environmental stimuli to generate the agent's behavior.

This system has been tested in a series of realistic wargaming experiments in which the actions of the red and blue fighters were directed by experienced

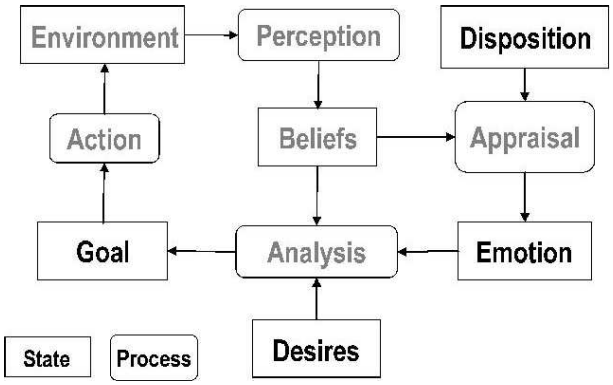


*Figure 1-4.* Desires, beliefs, dispositions, and emotions generate an agent's basic behaviors in response to environmental stimuli.

military commanders. While the results of these experiments have not yet been released for publication, BEE was successful in detecting which units were being played to exhibit specified dispositions.

## 5. Discussion and Conclusion

These applications are both instances of agent-centered PBS. The configuration problem is directly comparable to the many applications of PBS to robotic configuration, while the fitting problem can be compared to biological studies that seek to understand existing behavior in the natural world.

What sets these applications apart from most others is their real-time nature. In many instances of PBS, the entire population is synchronized. Even when the focus of search is the single agent rather than the system as a whole, it is common to update all agents at the same time, replacing the entire population at each generation. In our approach, breeding occurs in parallel with the evaluation of the ghosts. In the configuration example, only 1% of the population is replaced in each generation. Since the evaluation of an individual can take 100 - 300 time steps (the round trip distance with room for wandering), forcing a complete evaluation cycle before breeding would have slowed down the algorithm considerably. Similarly, in the fitting example, ghosts are continually compared with the behavior of the entities they represent as the battle unfolds, and breeding affects only a small fraction (about 3%) of the ghosts at each time step. By changing only a fraction of the population at each time step, we leave the bulk of the agents to carry on the work that the system is intended to do and avoid catastrophic shifts due to maladaptive individuals. At the same time,

we limit the ability of the system to respond rapidly to catastrophic exogenous events, a weakness to which natural real-time evolution is not immune.

It is common in agent-centered PBS to evaluate the fitness of an individual in isolation, or in a tournament where individuals from separate populations compete with each other. In our examples, the ghosts are part of a mixed population. Each of them is depositing pheromones and reacting to pheromones in a common environment. Thus, unfit individuals are depositing pheromones in the same environment being sensed by fit individuals, potentially causing the fit individuals to score lower than they would otherwise. This fact initially concerned us. We weren't sure whether PBS would even work under those circumstances. However, this particular problem appears to have a number of reasonable solutions, so the effect of having a mixed population did not prevent the algorithms from identifying and rewarding the better individuals.

Three details of our approach make it possible to apply PBS in real time.

1 Real-time PBS is facilitated by an agent-centric approach so that some components of the system can be modified while others carry on the system's work.

2 This approach is realistic only with populous systems, so that the effect of a change in a single agent do not discontinuously change the dynamics of the whole system. We know empirically that the systems described in this paper can function with populations on the order of 100 and more, but we have not systematically explored the lower bound.

3 Agents should be light-weight, so that multiple copies can be executed fast enough to keep up with the real world. We have found that the digital pheromone model, using simple functions to combine the pheromones sensed by the agent in its environment, is efficient enough to support tens of thousands of agents concurrently, thus providing both the population sizes and repeated cycles needed for effective evolution while keeping pace with real time.

Our experiments show that it is feasible to evolve a complex system in real time, element by element, rather than in a planning step that is temporally discontinuous with the system's operation. This approach opens new opportunities for applying PBS to dynamically changing systems that do not lend themselves to lengthy planning cycles.

## References

Agah, A. and Bekey, G.A. (1996). A genetic algorithm-based controller for decentralized multi-agent robotic systems. In *The 1996 IEEE International Conf. on Evolutionary Computation*, pages 431–436, Nagoya, Japan. IEEE.

Booker, L. B., Goldberg, D. E., and Holland, J. H. (1989). Classifier systems and genetic algorithms. *Artificial Intelligence*, 40:235–282.

Botee, Hozefa M. and Bonabeau, Eric (1998). Evolving ant colony optimization. *Adv. Complex Systems*, 1:149–159.

Boumaza, Amine M. and Louchet, Jean (2001). Dynamic flies: Using real-time parisian evolution in robotics. In Boers, Egbert J. W., Cagnoni, Stefano, Gottlieb, Jens, Hart, Emma, Lanzi, Pier Luca, Raidl, Gunther R., Smith, Robert E., and Tijink, Harald, editors, *Applications of Evolutionary Computing*, volume 2037 of *LNCS*, pages 288–297, Lake Como, Italy. Springer-Verlag.

Brooks, Rodney A. (1992). Artificial life and real robots. In Varela, Francisco J. and Bourgine, Paul, editors, *The First European Conference on Artificial Life*, pages 3–10.

Brueckner, Sven (2000). *Return from the Ant: Synthetic Ecosystems for Manufacturing Control*. Dr.rer.nat., Humboldt University Berlin.

Brueckner, Sven A. and Parunak, H. Van Dyke (2004). Swarming distributed pattern detection and classification. In Weyns, Danny, Parunak, H. Van Dyke, and Michel, Fabien, editors, *Workshop on Environments for Multi-Agent Systems (E4MAS 2004)*, volume LNAI 3374, New York, NY. Springer.

Corne, D., Dorigo, M., and Glover, F., editors (1999). *New Ideas in Optimisation*. McGraw-Hill, New York.

Dadone, P. and VanLandingham, H.F. (1999). Adaptive online parameter tuning using genetic algorithms. In *Proceedings of WSC4: 4th Online World Conference on Soft Computing in Industrial Applications*.

Forrest, S., Hofmeyr, S., and Somayaji, A. (1997). Computer immunology. *Communications of the ACM*, 40:88–96.

Harvey, I., Husbands, P., and Cliff, D. (1992). Issues in evolutionary robotics. In Meyer, J-A, Roitblat, H, and Wilson, S, editors, *The Second International Conference on Simulation of Adaptive Behaviour (SAB92)*, pages 364–373.

Kantz, Holger and Schreiber, Thomas (1997). *Nonlinear Time Series Analysis*. Cambridge Nonlinear Science Series. Cambridge University Press, Cambridge, UK.

Li, Yi, Riolo, Rick, and Savit, Robert (2000a). Evolution in minority games. i. games with a fixed strategy space. *Physica A*, 2000(276):234 – 264.

Li, Yi, Riolo, Rick, and Savit, Robert (2000b). Evolution in minority games ii. games with variable strategy spaces. *Physica A*, 2000(276):265 – 283.

Nordin, P. and Banzhaf, W. (1997). Real time control of a khepera robot using genetic programming. *Cybernetics and Control*, 26(3):533–561.

Ortony, A., Clore, G.L., and Collins, A. (1988). *The cognitive structure of emotions*. Cambridge University Press, Cambridge, UK.

Panait, Liviu A. and Luke, Sean (2004). Learning ant foraging behaviours. In Pollack, Jordan, Bedau, Mark, Husbands, Phil, Ikegami, Takashi, and

Watson, Richard A., editors, *Artificial Life XI Ninth International Conference on the Simulation and Synthesis of Living Systems*, pages 575–580, Boston, Massachusetts. The MIT Press.

Parunak, H. Van Dyke, Bisson, Robert, Brueckner, Sven, Matthews, Robert, and Sauter, John (2005). Representing dispositions and emotions in simulated combat. In Thompson, Simon, Ghanea-Hercock, Robert, Greaves, Mark, Meyer, Andre, and Jennings, Nick, editors, *Workshop on Defence Applications of Multi-Agent Systems (DAMAS05, at AAMAS05)*, page (forthcoming), Utrecht, Netherlands.

Parunak, H. Van Dyke and Brueckner, Sven (2003). Swarming coordination of multiple UAV's for collaborative sensing. In *Second AIAA "Unmanned Unlimited" Systems, Technologies, and Operations Conference*, San Diego, CA. AIAA.

Parunak, H. Van Dyke, Purcell, Michael, and O'Connell, Robert (2002). Digital pheromones for autonomous coordination of swarming UAV's. In *First AIAA Unmanned Aerospace Vehicles, Systems,Technologies, and Operations Conference*, Norfolk, VA. AIAA.

Quinn, M. (2001). Evolving communication without dedicated communication channels. In Kelemen, J. and Sosik, P., editors, *Advances in Artificial Life: Sixth European Conference on Artificial Life: ECAL2001*, pages 357–366, Prague, Czech Republic. Springer.

Reynolds, Craig W. (1987). Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics*, 21(4):25–34.

Riolo, Rick L, Axelrod, Robert, and Cohen, Michael D. (2001). Evolution of cooperation without reciprocity. *Nature*, 414(22 Nov):441–443.

Sauter, John A., Matthews, Robert, Parunak, H. Van Dyke, and Brueckner, Sven (2002). Evolving adaptive pheromone path planning mechanisms. In *Autonomous Agents and Multi-Agent Systems (AAMAS02)*, pages 434–440, Bologna, Italy.

Sauter, John A., Matthews, Robert, Parunak, H. Van Dyke, and Brueckner, Sven A. (2005). Performance of digital pheromones for swarming vehicle control. In *Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems*, page (forthcoming), Utrecht, Netherlands.

Savit, Robert, Brueckner, Sven A., Parunak, H.Van Dyke, and Sauter, John (2002). Phase structure of resource allocation games. *Physics Letters A*, 311:359–364.

Spector, Lee, Klein, Jon, Perry, Chris, and Feinstein, Mark (2005). Emergence of collective behavior in evolving populations of flying agents. *Genetic Programming and Evolvable Machines*, 6. Prepublication Date: 6 August 2004.

Steels, L. (2000). The puzzle of language evolution. *Kognitionswissenschaft*, 8(4):143–150.

Weinstein, Peter, Parunak, H. Van Dyke, Chiusano, Paul, and Brueckner, Sven (2004). Agents swarming in semantic spaces to corroborate hypotheses. In *AAMAS 2004*, pages 1488–1489, New York, NY.

Weyns, Danny, Parunak, H. Van Dyke, Michel, Fabien, Holvoet, Tom, and Ferber, Jacques (2004). Multiagent systems, state-of-the-art and research challenges. In Weyns, Danny, Parunak, H. Van Dyke, and Michel, Fabien, editors, *Workshop on Environments for Multi-Agent Systems (E4MAS 2004)*, volume LNAI 3374, New York, NY. Springer.

White, Tony, Pagurek, Bernard, and Oppacher, Franz (1998). ASGA: Improving the ant system by integration with genetic algorithms. In Koza, John R., Banzhaf, Wolfgang, Chellapilla, Kumar, Deb, Kalyanmoy, Dorigo, Marco, Fogel, David B., Garzon, Max H., Goldberg, David E., Iba, Hitoshi, and Riolo, Rick, editors, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 610–617, University of Wisconsin, Madison, Wisconsin, USA. Morgan Kaufmann.