chapter

seven

The Particle Swarm

This chapter introduces the particle swarm in its binary and real-numbered forms. The book so far has been preparing a context, describing related paradigms in computer science and social science, discussing culture and norms and language and other scientific and philosophical developments that, if we have been successful, will make the particle swarm seem like an obvious thing to propose.

The Adaptive Culture Model in the previous chapter hints at what can happen as a result of the simplest imaginable interactions of the simplest imaginable agents—if these can even be called "agents." Given a large space of possibilities, the population is often able to find multivariate solutions, patterns that solve problems, through a stripped-down form of social interaction.

It is worth emphasizing that individuals in the culture model are not *trying* to solve problems. They are only following the simple rules of the algorithm, which say nothing about the existence of a problem or how to solve it. Yet through reciprocal social influence each individual betters its "fitness" (the term is less appropriate here than in discussion of evolutionary algorithms), and the performance of the population improves. We would not say that the adaptive culture algorithm is an especially powerful way to solve problems, but it is a good introduction to some social algorithms that are.

The particle swarm algorithm is introduced here in terms of social and cognitive behavior, though it is widely used as a problem-solving method in engineering and computer science. We have discussed binary encoding of problems, and the first version of the particle swarm we present here is designed to work in a binary search space. Later in the chapter we introduce the more commonly used version, which operates in a space of real numbers. ■

Team LRN

Sociocognitive Underpinnings: Evaluate, Compare, and Imitate

A very simple sociocognitive theory underlies the Adaptive Culture Model and particle swarms. We theorize that the process of cultural adaptation comprises a high-level component, seen in the formation of patterns across individuals and the ability to solve problems, and a low-level component, the actual and probably universal behaviors of individuals, which can be summarized in terms of three principles (Kennedy, 1998):

- Evaluate
- Compare
- Imitate

Evaluate

The tendency to evaluate stimuli—to rate them as positive or negative, attractive or repulsive—is perhaps the most ubiquitous behavioral characteristic of living organisms. Even the bacterium becomes agitated, running and tumbling, when the environment is noxious. Learning cannot occur unless the organism can evaluate, can distinguish features of the environment that attract and features that repel, can tell good from bad. From this point of view, learning could even be defined as a change that enables the organism to improve the average evaluation of its environment.

Compare

Festinger's social comparison theory (1954) described some of the ways that people use others as a standard for measuring themselves, and how the comparisons to others may serve as a kind of motivation to learn and change. Festinger's theory in its original form was not stated in a way that was easily tested or falsified, and a few of the predictions generated by the theory have not been confirmed, but in general it has served as a backbone for subsequent social-psychological theories. In almost everything we think and do, we judge ourselves through comparison with others, whether in evaluating our looks, wealth, humor, intelligence (note that IQ scales are normed to a population average; in other words,

Team LRN

your score tells you how you compare to others—which is really the point, isn't it?), or other aspects of opinion and ability. Individuals in the Adaptive Culture Model—and in particle swarms—compare themselves with their neighbors on the critical measure and imitate only those neighbors who are superior to themselves. The standards for social behaviors are set by comparison to others.

Imitate

You would think that imitation would be everywhere in nature; it is such an effective way to learn to do things. Yet, as Lorenz has pointed out, very few animals are capable of real imitation; in fact, he asserts that only humans and some birds are capable of it. Some slight variations of social learning are found among other species, but none compare to our ability to mimic one another. While "monkey see, monkey do," well describes the imitative behavior of our cousins, human imitation comprises taking the perspective of the other person, not only imitating a behavior but realizing its purpose, executing the behavior when it is appropriate. In *The Cultural Origins of Human Cognition,* Michael Tomasello argues that social learning of several kinds occurs in chimpanzees, but true imitation learning, if it occurs at all, is rare. For instance, an individual's use of an object as a tool may call another individual's attention to the object; this second individual may use the same object, but in a different way. True imitation is central to human sociality, and it is central to the acquisition and maintenance of mental abilities.

The three principles of evaluating, comparing, and imitating may be combined, even in simplified social beings in computer programs, enabling them to adapt to complex environmental challenges, solving extremely hard problems. Our view diverges from the cognitive viewpoint in that nothing besides evaluation, comparison, and imitation takes place *within* the individual; mind is not found in covert, private chambers hidden away inside the individual, but exists out in the open; it is a public phenomenon.

A Model of Binary Decision

Consider a bare-bones individual, a simple being with only one thing on its mind, one set of decisions to make, yes/no or true/false, binary decisions, but very subtle decisions, where it is hard to decide which choices

Team LRN

to make. For each decision, this supersimplified individual can be in one state or the other, either in the yes state, which we will represent with a 1, or the no = 0 state. It is surrounded by other yes/no individuals, who are also trying to decide. Should I say yes? Should I say no? They all want to make the best choices.

Two important kinds of information are available to these primitive beings. The first is their own experience; that is, they have tried the choices and know which state has been better so far, and they know how good it was. But these social beings have a second consideration; they have knowledge of how the other individuals around them have performed. In fact they are so simple that all they know is which choices their neighbors have found most positive so far and how positive the best pattern of choices was. If these stripped-down beings are anything like people, they know how their neighbors have done by observing them and by talking with them about their experiences.

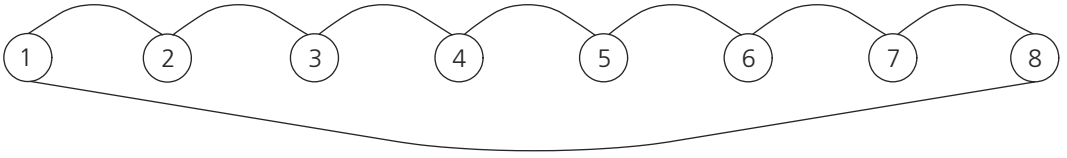
These two types of information correspond to Boyd and Richerson's individual learning and cultural transmission. The probability that the individual will choose "yes" for any of the decisions is a function of how successful the "yes" choice has been for them in the past relative to "no." The decision is also affected by social influence, though the exact rule in humans is admittedly not so clear. Social impact theory states that the individual's binary decisions will tend to agree with the opinion held by the majority of others, weighted by strength and proximity. But even that rule is somewhat vague, given ambiguities in the concepts of strength and proximity.

For the present introductory model we will just say that individuals tend to be influenced by the best success of anyone they are connected to, the member of their sociometric neighborhood that has had the most success so far. While we admit this is an oversimplification, it has a kernel of truth that justifies the parsimony it brings to the model.

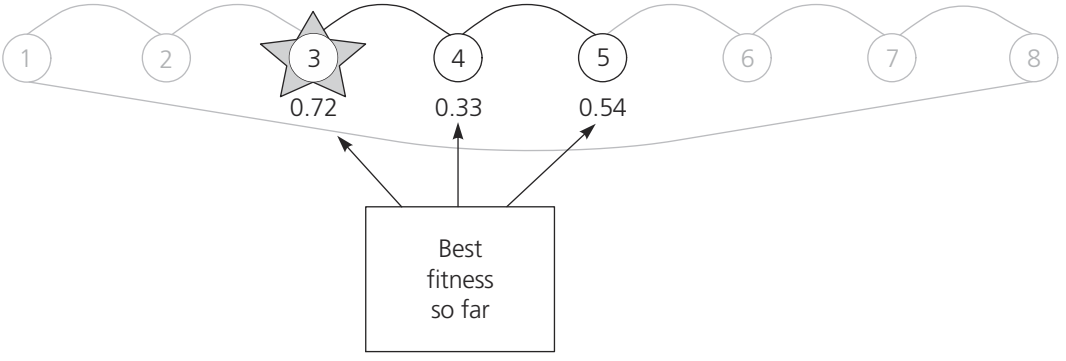
Individuals can be connected to one another according to a great number of schemes, some of which will be mentioned in Chapter 8. Most particle swarm implementations use one of two simple sociometric principles (see Figure 7.1). The first, called *gbest,* conceptually connects all members of the population to one another. The effect of this is that each particle is influenced by the very best performance of any member of the entire population. The second, called *lbest* (*g* and *l* stand for "global" and "local"), creates a neighborhood for each individual comprising itself and its *k* nearest neighbors in the population. For instance, if $k = 2$, then each individual i will be influenced by the best performance among a group made up of particles $i - 1$, i, and $i + 1$. Different neighborhood topologies may result in somewhat different kinds of effects. Unless stated

Team LRN

The "lbest" neighborhood with $k = 2$. Each individual's neighborhood contains itself and its two adjacent neighbors. The first and last are connected.



Individual #3 has found the best position so far in #4's neighborhood. Therefore, #4's velocity will be adjusted toward #3's previous best position and #4's own previous best position.



The "gbest" neighborhood. Assuming that #3 has found the best fitness so far in the entire population, all others' velocities will be attracted toward its previous best position.

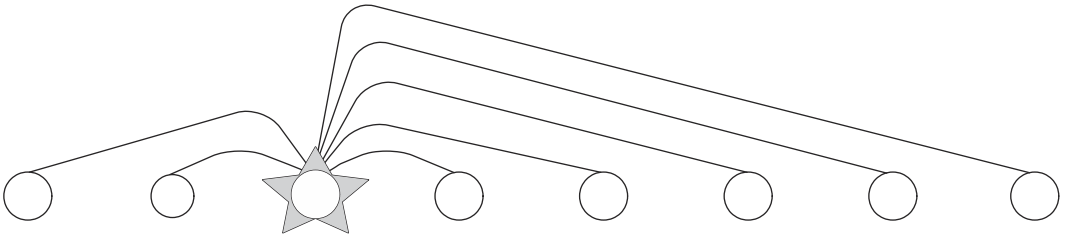


Figure 7.1 The two most common types of neighborhoods.

otherwise, the following discussions will presume lbest neighborhoods with $k = 2$ (sometimes described as "neighborhood = 3").

In a sociocognitive instance the individual must arrange an array of decisions or judgments in such a way that they all fit together, what we call "making sense" or "understanding" things. The individual must be able to evaluate, compare, and imitate a number of binary choices simultaneously.

Evaluation of binary strings can be accomplished in one step. In the psychological case, that is, if we are talking about humans, we can again use the concept of cognitive dissonance to evoke the sense of tension that exists when an array of decisions contains inconsistencies. We experience the state as discomfort and are motivated to change something to reduce the tension, to improve the evaluation. Dissonance as described by Festinger provides a single measure of cognitive evaluation, exactly as "fitness" is a single measure of genetic or phenotypic goodness.

How do we improve cognitive fitness? Of course there are plenty of theories about this. In Ajzen and Fishbein's *Reasoned Action Model,* (1980) *intent* is seen as a function of two kinds of things that should be getting familiar by now (see Figure 7.2). On the one hand, intent is affected by the person's *attitude* toward the behavior; for instance, if they believe violence is harmful or immoral, then they may intend not to act violently. This attitude is formed, in Ajzen (pronounced "eye-zen") and Fishbein's theory, by a linear combination of beliefs that the behavior will result in some outcomes (b_i) times the individual's evaluation of those outcomes (e_i):

$$A_o = \sum_{i=1}^{n} b_i e_i$$

This kind of expectancy-value model of attitude has existed in some form for many years, and we will not criticize its linearity or asymptotic issues here (never mind the decades-old debate about summing versus averaging). We are interested in the fact that intent has a second cause, which Ajzen and Fishbein call the *subjective norm.* The subjective norm regarding a behavior is also built up, in their theory, as a linear sum of products, but this time the factors entering into the formula are social. The individual's subjective norm toward a behavior is a sum of the products of their beliefs that certain others think they should or should not perform the behavior, multiplied by the motivation to comply with each of those others:

$$SN_o = \sum_{i=1}^{n} b_i m_i$$

Team LRN

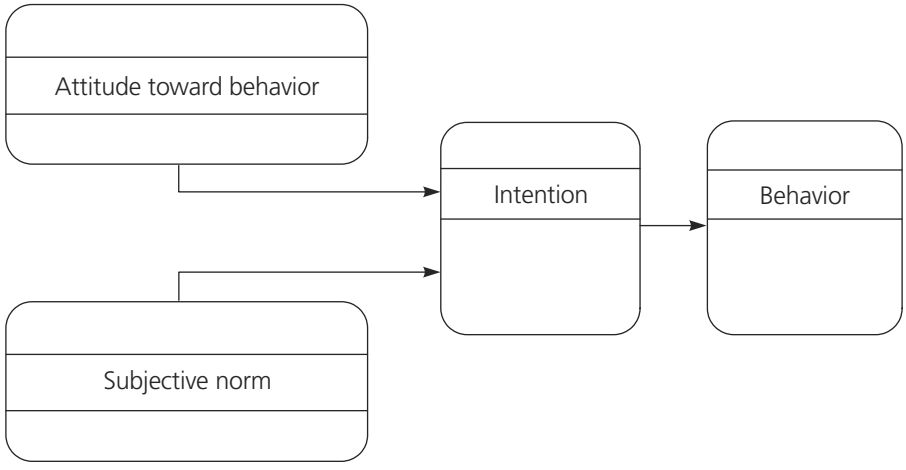


Figure 7.2 According to the Reasoned Action Model, behavior is a function of intention and only remotely of the individual's attitude about the behavior.

To point out the obvious, these two components of the theory of reasoned action map easily onto the components of Boyd and Richerson's cultural transmission model; that is, there is an individual term (individual learning or attitude toward a behavior) and a social term (cultural transmission or subjective norm). These two kinds of concepts are found in other theories as well and are represented in our decision model as the two terms that make up the change formula. We theorize that the coexistence of these two modes of knowledge, that is, knowledge acquired by the senses through experience in the world and knowledge acquired from others, gives humans the intellectual advantage; it is the source of our intelligence.

Besides their past experience and inputs from the social environment, another factor that affects the individual's decision is their current propensity or position regarding the issue. They may start with a strongly negative attitude and have subsequent positive experiences regarding the choice or attitude object—but still have a negative feeling about it. The positive experiences may make the individual *more* likely to choose the positive alternative, but in order to shift the individual's general propensity into the positive domain, the decision threshold would still have to shift upwards. If the individual's initial position is extreme, the probability is lower of its changing—for one thing, the individual is less likely to try the other alternative.

In mathematical terms, we are proposing a model wherein the probability of an individual's deciding yes or no, true or false, or making some

Team LRN

other binary decision, is a function of personal and social factors (Kennedy and Eberhart, 1997):

$$P(x_{id}(t) = 1) = f(x_{id}(t-1), v_{id}(t-1), p_{id}, p_{gd})$$

where

- $P(x_{id}(t)=1)$ is the probability that individual i will choose 1 (of course the probability of their making the zero choice is $1 - P$) for the bit at the dth site on the bitstring
- $x_{id}(t)$ is the current state of the bitstring site d of individual i
- t means the current time step, and $t - 1$ is the previous step
- $v_{id}(t - 1)$is a measure of the individual's predisposition or current probability of deciding 1
- p_{id} is the best state found so far, for example, it is 1 if the individual's best success occurred when x_{id} was 1 and 0 if it was 0
- p_{gd} is the neighborhood best, again 1 if the best success attained by any member of the neighborhood was when it was in the 1 state and 0 otherwise

The decisions themselves will be stochastic, if for no better theoretical reason than that we never know all the forces involved—it is very unlikely that any decision is made based solely on isolated facts pertaining directly to that decision alone. A lot of randomness allows exploration of new possibilities, and a little bit allows exploitation by testing patterns similar to the best one found so far; thus we can balance between those two modes of search by adjusting the uncertainty of decisions.

The parameter $v_{id}(t)$, an individual's predisposition to make one or the other choice, will determine a probability threshold. If $v_{id}(t)$ is higher, the individual is more likely to choose 1, and lower values favor the 0 choice. Such a threshold needs to stay in the range [0.0, 1.0]. We have already seen one straightforward function for accomplishing this, when we talked about neural networks. The sigmoid function

$$s(v_{id}) = \frac{1}{1+\exp(-v_{id})}$$

squashes its input into the requisite range and has properties that make it agreeable to being used as a probability threshold (though there is nothing magical about this particular function).

We wish to adjust the individual's disposition toward the successes of the individual and the community. To do that we construct a formula for each v_{id} in the current time step that will be some function of the difference between the individual's current state or position and the best points found so far by itself and by its neighbors. We want to favor the best position, but not so much that the individual ceases searching prematurely. If we simply added ($p_{id} - x_{id}(t-1)$) and ($p_{gd} - x_{id}(t-1)$) to $v_{id}(t)$, it would move upward when the difference between the individual's previous best and most recent states, or the difference between the neighborhood's best and the individual's most recent states, equaled 1, and would be attracted downward if either difference equaled −1. The probability threshold moves upward when the bests are ones and downward when they are zeroes.

In any situation we do not know whether the individual-learning or the social-influence terms should be stronger; if we weight them both with random numbers, then sometimes the effect of one, and sometimes the other, will be stronger. We use the symbol φ (the Greek letter phi) to represent a positive random number drawn from a uniform distribution with a predefined upper limit. In the binary version the limit is somewhat arbitrary, and it is often set so that the two φ limits sum to 4.0. Thus the formula for binary decision is

$$v_{id}(t) = v_{id}(t-1) + \varphi_1(p_{id} - x_{id}(t-1)) + \varphi_2(p_{gd} - x_{id}(t-1))$$

$$\text{if } \rho_{id} < s(v_{id}(t)) \text{ then } x_{id}(t) = 1; \text{ else } x_{id}(t) = 0$$

where ρ_{id} is a vector of random numbers, drawn from a uniform distribution between 0.0 and 1.0. These formulas are iterated repeatedly over each dimension of each individual, testing every time to see if the current value of x_{id} results in a better evaluation than p_{id}, which will be updated if it does. Boyd and Richerson varied the relative weighting of individual experience and social transmission according to some theoretical suggestions; the current model acknowledges the differential effects of the two forces without preconceptions about their relative importance. Sometimes decisions are based more on an individual's personal experience and sometimes on their perception of what other people believe, and either kind of information will dominate sometimes.

One more thing: we can limit v_{id} so that $s(v_{id})$ does not approach too closely to 0.0 or 1.0; this ensures that there is always some chance of a bit flipping (we also don't want v_i moving toward infinity and overloading the exponential function!). A constant parameter $V_{\max}$ can be set at the

Team LRN

start of a trial to limit the range of v_{id}. In practice, $V_{\max}$ is often set at ± 4.0, so that there is always at least a chance of $s(V_{\max}) \approx 0.0180$ that a bit will change state. In this binary model, $V_{\max}$ functions similarly to mutation rate in genetic algorithms.

Individuals make their decision in a population, where they are influenced by the successes of their neighbors. As each individual's decision is affected by $(p_{gd} - x_{id}(t-1))$, that is, (usually) some other individual's success, they influence one another and tend to move toward a common position. As an individual begins to approximate its neighbor's best position, it may perform better and influence *its* neighbors, and on and on; good decisions spread through the population. We are comfortable calling this the formation of a culture in a computational population.

In this section we have developed an extremely parsimonious model of binary choice as a function of individual learning and social influence. Individuals tend to gravitate probabilistically toward the decisions that have resulted in successes for themselves and their colleagues. The result is optimization of each individual's decision vector and convergence of the population on an optimal pattern of choices.

The entire algorithm, maximizing goodness, is shown in pseudocode:

```
Loop
  For i = 1 to number of individuals
    if G(x_i) > G(p_i) then do          //G() evaluates goodness
      For d = 1 to dimensions
        p_id = x_id                     //p_id is best so far
      Next d
    End do

    g = i                               //arbitrary
    For j = indexes of neighbors
      If G(p_j) > G(p_g) then g = j     //g is index of best performer
                                        in the neighborhood
    Next j
    For d = 1 to number of dimensions
      v_i(t) = v_id(t-1) + φ1(p_id - x_id(t-1)) + φ2(p_gd - x_id(t-1))
      v_id ∈ (-V_max, + V_max)
      if ρ_id < s(v_id(t)) then x_id(t) = 1; else x_id(t) = 0;
    Next d
  Next i
Until criterion
```

Testing the Binary Algorithm with the De Jong Test Suite

It may seem confusing to jump back and forth between "cognitive models" and "test functions." We are maintaining a generous definition of cognitive models, given the lack of consensus among psychologists about the internal structure of the mechanisms of thought. Thus, to us, a cognitive model is just like any other multidimensional problem where elements interact with one another in a combination possessing some measurable goodness.

Kennedy and Eberhart (1997) tested the binary particle swarm using a binary-coded version of the classic De Jong suite of test problems. The binary versions had already been prepared for experimentation with binary genetic algorithms, so importing them into a binary particle swarm program was straightforward. A population size of 20 was used for all tests. In all cases the global optimum was at $(0.0)^n$. The algebraic forms of the functions are given in Table 7.1.

The binary particle swarm converged quickly on $f1$, also known as the sphere function, encoded as a 30-dimensional bitstring. The best

Table 7.1 Functions used by De Jong to test various aspects of optimization algorithms.

Function	Dimension
$f1(x_i) = \sum_{i=1}^{n} x_i^2; -5.12 \le x_i \le 5.12$	30
$f2(x_i) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2; -2.048 \le x_i \le 2.048$	24
$f3(x_i) = \sum_{i=1}^{n} \text{int}(x_i); -5.12 \le x_i \le 5.12$	50
$f4(x_i) = \sum_{i=1}^{n} i x_i^4 + Gauss(0,1); -1.28 \le x_i \le 1.28$	240
$f5(x_i) = 0.002 + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6}; -65.536 \le x_i \le 65.536;$ $[a_{ij}] = \begin{bmatrix} -32 & -16 & 0 & 16 & 32 & -32 & -16 & 0 & 16 & 32 & -32 \ldots \text{etc.} \\ -32 & -32 & -32 & -32 & -32 & -16 & -16 & -16 & -16 & -16 & 16 \ldots \text{etc.} \end{bmatrix}$	34

solution the particle swarm found was 0.000002 away from the "perfect" result of 0.0, which it found on 10 of the 20 trials. It is presumed that the difference between the found optimum and the target is due to imprecision in the binary encoding rather than a failure of the algorithm to hit the target.

On the second function, De Jong's $f2$, in 24 dimensions, the particle swarm was able to attain a best value of 0.000068, compared to a target of 0.0; again, the difference is thought to derive from the precision of the encoding rather than the algorithm. This function was encoded in a 24-dimension bitstring. $f2$ was the hardest of the De Jong functions for the particle swarm; the system converged on the best-known optimum 4 times in this set of 20. The hardness of the function might be explained by the existence of very good local optima in regions that are distant in Hamming space from the best-known optimum. For instance, the local optimum

010111111101111000000111

returns a value of 0.000312, while the bitstring

110111101001110111011111

returns 0.000557, and

111000011001011001000001

returns 0.005439. The best-known optimum, returning 0.000068, was found at

110111101110110111101001.

Thus bitstrings that are very different from one another, in terms of Hamming distance, are all relatively good problem solutions. A search algorithm that relies on hill climbing is unlikely to make the leap from a locally optimal region to the global optimum. The function itself has only one optimum and is hard because of the wide flat regions where movement, whether it is toward or away from the optimum, likely results in no real change in fitness.

The third function, $f3$, is an integer function encoded in 50 dimensions whose target value was attained easily on every trial. De Jong's $f4$ function introduces Gaussian noise to the function, and performance

Team LRN

was measured as an average over the entire population rather than a population best. Finally, on $f5$ the algorithm was able to attain a best value of 0.943665 on 20 out of 20 attempts, in 34 dimensions; we presume that to be the global optimum. The system converged rapidly on this fitness peak every time.

The five functions were implemented in a single program, where the only code changed from one test to another was the evaluation function. All other aspects of the program, including parameter values, ran identically on the various functions. Thus it appeared from this preliminary research that the binary particle swarm was flexible and robust.

No Free Lunch

There is some controversy in the field regarding the evaluation of an algorithm, and maybe our claims that particle swarm optimization is "powerful" or "effective" should be disregarded. Imagine two optimization algorithms, one that searches by following the gradient, that is, a hill-climbing algorithm, and another that searches by hopping randomly around the landscape. Now imagine two problems, one, like the sphere function, where the gradient leads inevitably to the optimum and another that has many optimal regions, some better and some worse, a landscape peppered with hills and mountain ranges.

Of course the descriptions have been contrived so that it will be obvious that each algorithm will perform better on one of the problems. It would be foolish to search this way and that when there is a clear-cut yellow brick road leading directly to Oz, and it is equally foolish to climb the nearest hill in a rugged landscape. In this particular case it is clear that the performance of the algorithm depends on the kind of problem.

In important and controversial papers in 1996 and 1997, David Wolpert and William Macready formalized and generalized this observation, and their analysis has some surprising implications. Not only are some algorithms relatively more or less appropriate for certain kinds of problems—but averaged over all possible problems or cost functions, the performance of all search algorithms is *exactly the same*. This includes such things as random search; no algorithm is better, on average, than blind guessing. Provocatively, Wolpert and Macready question whether natural selection is an effective biological search strategy and suggest that breed-the-worst might work as well as breed-the-best, except that no one has ever conducted the experiment on the massively parallel scale of natural evolution.

Team LRN

The No Free Lunch (NFL) theorem, as Wolpert and Macready (1997) called it, has generated considerable discussion among researchers. Where previously there had been hope that some search strategy could be found—and evolutionary computation researchers thought they had it—that would be a best first-guess approach to any class of problems, research has more recently focused on finding exactly what the strengths and limitations of various search strategies are.

Some observers take NFL to mean that no optimization algorithm can be any better than any other. Of course—that's exactly what the theorem says, isn't it? Actually, the theorem says that no algorithm can be better than any other *averaged over all cost functions*. This is a hugely important condition.

What does it mean to average over all possible cost functions? Think of it this way. We have an optimization problem: exiting a room in the dark. Our special algorithm follows these steps (see Figure 7.3(a)):

- Move in a straight line until you reach a wall.
- Move along the wall until you feel an opening.
- Go through the opening.

There could easily be other algorithms, such as stumble-around-waving-your-arms-in-the-air, ever-widening circles, and so on, and some may help us exit better, some worse, than our own proprietary algorithm.

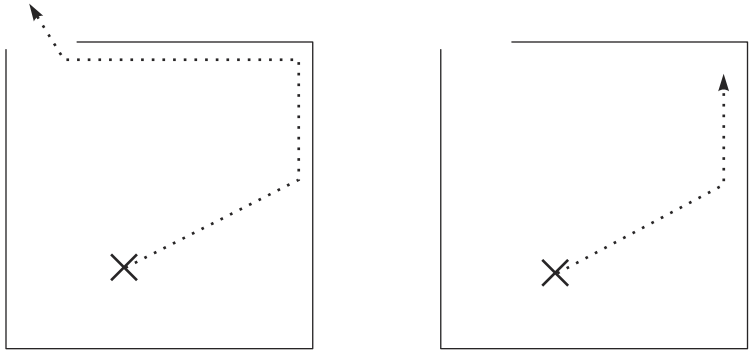


Figure 7.3 Two different algorithms, equally good at different things: the stop-in-corners algorithm is good for finding the way out of a room (a), and the find-corners algorithm is good at getting stuck in the corner of a room (b).

Team LRN

Now imagine that a nefarious NFL advocate has an algorithm called stop-in-corners, which he claims is just as good as our feel-for-opening algorithm. His algorithm goes like this (see Figure 7.3(b)):

- Move in a straight line until you reach a wall.
- Move along the wall until you feel a corner.
- Stop there.

But (you argue) you don't see how he will ever be able to exit a room in the dark, behaving like that! His response is, of course I won't leave the room very well, but there are four corners to the room and only one door, so I will on average be four times as successful at getting stuck in a corner as you will be at exiting. But (you insist) you don't want to get stuck in a corner, you want to exit the room—and your method is better than his for that. He admits that, but adds, my algorithm is as good as yours—averaged over all problems.

The NFL theorem says that, in order to evaluate an algorithm, you have to average it over all cost functions—and there can be very many of those. Some might be

- Exit the room (which is your problem)
- Get stuck in a corner (his problem)
- Find the center
- Find a point halfway between the center and the edge
- Find a point a third of the way between the center and the edge (et cetera, ad infinitum)
- Avoid walls altogether

and so on (another condition is that the search space must be finite, so we don't need to worry about problems from outside our domain). And averaged over all of those, his algorithm and yours are equally good. You might argue that you would never want to find the center or avoid walls altogether, and that gets to the limitation of the No Free Lunch theorem.

While it may be true that no algorithm is better than any other, when averaged over every absurd task that can possibly be imagined, it is perfectly possible that an algorithm would be better than others on the kinds of tasks that we call "problems." If anything, the NFL theorem makes us think about what it is that we try to address with an

Team LRN

optimization algorithm. Most things, even in a finite universe, do not qualify as problems; this may be more a reflection of our way of thinking than anything inherent in mathematics or in the world. Suffice it to say, we do not feel uncomfortable saying that one algorithm, which can reliably find global optima in real problem spaces, is better than another, which can find answers to questions that no one would ever ask.

Multimodality

As part of Bill Spears' investigations of the strengths and weaknesses of genetic algorithms at the Naval Research Laboratory (NRL) and as a former graduate student of Ken De Jong's, he has assembled and posted online a collection of interesting test functions, problems that push and pull and stretch an optimization algorithm to its limit to see what it can and can't do. If there is No Free Lunch, there might be at least Some Kind of Lunch, and researchers want to know what their algorithm is good at.

In collaboration with his NRL colleague Mitch Potter, Spears designed and programmed a "multimodal random problem generator" (De Jong, Potter, and Spears, 1997). The rationale was this: obviously, if a researcher precision-tunes an optimization algorithm to work on one problem, there is a danger that it will fail on everything else. There was a need for a way to come up with different problems, but with some controllable characteristics. The random problem generator offers a way to test an algorithm on novel problems, controlling some aspects of the problems that are expected to affect performance.

Multimodality, in this context, means that a problem has more than one solution or global optimum, conceived as peaks on the fitness landscape. For instance, the problem $x^2 = 25$ is multimodal; it has two optimal solutions: $x = +5$ and $x = -5$. Since a genetic algorithm is often implemented using binary encoding, Spears wrote the program to create multimodal binary problems for the GA to solve. The concept is very straightforward. The researcher defines the dimensionality of the problem, that is, the length of the bitstring, and how many modes or peaks are desired, and the program creates that number of bitstrings, made of random sequences of zeroes and ones. For instance, imagine a researcher has specified that dimensionality $N = 10$ and multimodality or number of peaks $P = 5$. The problem generator might produce these bitstrings:

Team LRN

0100110111

1110010010

1101101010

0100000000

1110100101

With 10-dimensional bitstrings there are $2^{10} = 1{,}024$ possible patterns of bits. The goal for the optimizing algorithm is to find any one of the five peaks that have been defined by the program. The Hamming distance between a bitstring and the nearest optimum provides a fitness evaluation; that is, the more similar the bitstring is to one of the specified peaks, the fitter it is. For 10-dimensional bitstrings, the farthest an individual can be from a peak is 10 Hamming units, and of course a perfect match is a distance of zero from one of the peaks.

Multimodal problems can be hard for genetic algorithms. Recall that in GAs, chromosomes cross over in every generation; sections of successful ones are joined together to produce the next generation's population. In a multimodal situation it is entirely possible that the parts that are joined together come from chromosomes whose fitness derives from their proximity to different optima. For instance, the chromosome 0100110110 is only one bit different from the first optimum defined above, and 0110010010 is only one bit different from the second solution. Putting them together (we'll cut it right in the middle to be fair) could produce the child chromosome 0100110010, which is three bits different from the first optimum (Hamming distance = 3) and three bits different from the second—moving away from both of them. It is exactly the multimodality of the problem that makes crossover ineffective in this case.

GAs rely not only on recombination but on mutation (and sometimes other operators) for moving through a problem space. De Jong, Potter, and Spears tried several modifications of GAs, including one whose only operator was mutation—no crossover—in the multimodal random problem generator, calling it GA-M. In this algorithm, each site on each bitstring has a low probability of changing from a zero to a one or vice versa, usually less than 0.01. At each generation the population is evaluated, the fittest ones are selected, and mutation is applied to them.

Team LRN

Through this process, generations tend to improve; this amounts to a kind of stochastic hill climbing, as the more fit members of the population are more likely to be retained and mutated.

De Jong, Potter, and Spears also tested a GA implemented with crossover only—no mutation—and found that this kind tended to flounder in the early generations, but once the population started to converge on one particular peak or another, improvement came relatively fast. These crossover-only GAs, called GA-C, were very successful at finding one of the optima, if you waited long enough. The same was true of traditional GAs with both crossover and mutation. Mutation-only GAs, on the other hand, constantly improved, generation by generation, but if the dimensionality of the problem was high, the chance of mutating in a direction that led to improvement was very small and grew smaller as the population approached the optimum. When bitstrings were short, mutating chromosomes found optima quickly and efficiently, but "the curse of dimensionality" made bigger problems too difficult for them. Though they might have eventually found the global optimum, improvement

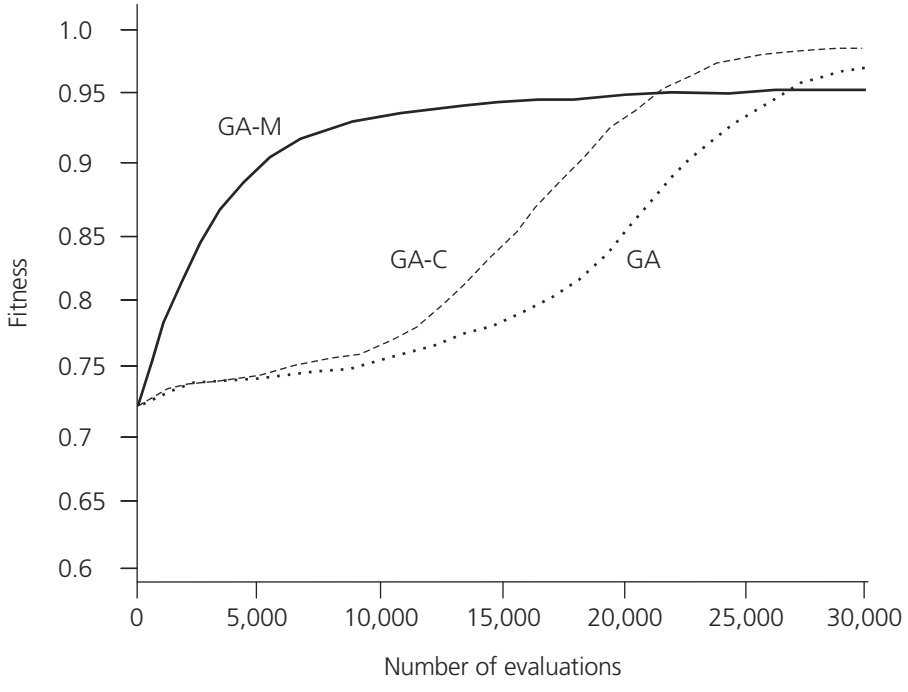


Figure 7.4 Average best-so-far performance of three types of genetic algorithms on 500-peak problems. (From De Jong, Potter, and Spears, 1997.)

Team LRN

decelerated over time. Figure 7.4 shows the performance of the three types of genetic algorithms.

In a follow-up to the De Jong et al. study, Kennedy and Spears (1998) compared the binary particle swarm algorithm with the three variations of GAs in the multimodal random problem generator. The study was constructed in the form of an experiment, where three independent variables were manipulated—algorithm, dimensionality, and multimodality. There were four kinds of algorithms (GA-C, GA-M, GA, and PS), two levels of dimensionality (20 and 100), and two levels of multimodality (20 and 100). In each of the 16 conditions of the experiment, there were 20 observations, and the population size was 100. (This population size, which is much bigger than a typical particle swarm, was used to make the two paradigms commensurate.)

The dependent variable in Kennedy and Spears' experiment was the shape of the best-so-far performance curves over time. This is a multivariate measure, more complicated than those found in the typical experiment, but easily computable with good statistical software. Each condition in the experiment was run 20 times for 20,000 evaluations. The mean best performance was calculated after 20 evaluations, and after 1,000, 2,000, and so on up to 20,000. It was possible to statistically compare the shapes of the performance curves for all comparisons, the question being not how well the various algorithms perform in the long run over the dimensionality and multimodality conditions, but how changes in their performance differed over time (see Figure 7.5).

GA-M performed best of all the algorithms in the early iterations of every condition, but was quickly overtaken by all the others, except in the "lite" condition, with short, 20-bit bitstrings and only 20 peaks. When either dimension or modality or both increased, however, GA-M suffered in its ability to find one of the peaks. The two GA variations with crossover, that is GA-C and GA—which implemented both crossover and mutation—started in every condition with a "dip" in performance, and then rose toward an optimum, almost always finding one of the peaks by the 20,000th evaluation.

The binary particle swarm performed the best in all conditions except the "lite" one (where it was second best); it found a global optimum on every trial in every condition and did it faster than the comparison algorithms. This is not to say that it would have performed better than *any* GA on these problems, and it may be possible to tune the parameters of a GA to optimize its performance in a particular situation. On the other hand, the significance of these results—which were statistically significant in a multivariate analysis of variance (MANOVA)—should not

Team LRN

以上内容仅为本文档的试下载部分，为可阅读页数的一半内容。如要下载或阅读全文，请访问：https://d.book118.com/208137131132006051