Chapter 3

First Formulations

3.1. Minimal version

3.1.1. *Swarm size*

Let us recall that for the moment the size of the swarm is fixed once for all. Intuitively, we feel of course that, the more particles, the faster the search will be in terms of the number of iterations. But, this iteration count is not really a relevant criterion. Rather, what counts is the number of times that the function to be minimized must be evaluated, because in the majority of real problems, this evaluation requires a considerable time. And, obviously, for an iteration, the number of evaluations is equal to the number of particles. Therefore, if we want to reduce the total number of evaluations needed to find a solution, we are tempted to decrease the size of the swarm. But too small a swarm is likely to take longer to find a solution or even not to find it at all.

In short, a compromise must be reached. Empirically, the experimenters proposed sizes of about 20 to 30 particles, which, indeed, proved entirely sufficient to solve almost all classic test problems. Note how small this value is, compared with those usually used, for example in the genetic algorithms (GA), a fact which does not facilitate comparisons. Those who are for GA say “Since I use 100 genes in my algorithms, I will take 100 particles for a comparison with PSO”. At once, obviously, they find that PSO, although finding a solution at least as often as the genetic algorithms, is not very effective in terms of number of evaluations, since this number of particles is rather too large and there is no selection. Conversely, if those who are for PSO use GA with only 20 genes, they will observe that the algorithm finds the solution less often, which is quite normal as well. In fact, and we will

return to this later, like any algorithm, PSO has its "field of competence", its "ecological niche", where it turns out to be the best choice [EBE 98, GUD 03, JEN 96, SET 03].

In the examples below we will systematically use a swarm of 20 particles, eventually showing that even this small number is sometimes larger than necessary. But we will also see later, in the chapter on performance maps, that a slightly greater number is more comfortable, in the sense that for a wide range of test problems it increases the average probability of success. Anyway, we must now make a move through the search space, first by defining their initial positions and velocities, then by specifying the equations of motion.

3.1.2. *Information links*

The information links are redefined randomly with each iteration: each particle informs K others chosen randomly. We note that it means that the group of informants corresponding to a particle has an average size slightly less than K, owing to the fact that the same information receiver can be selected several times. In the same way, it means that the average size of the groups of informants is also slightly less than K, though that is a little less obvious. The exact formula and the manner of finding it are given at the end of the chapter, for the benefit of mathematical amateurs.

It is enough for us here simply to note that the smaller the swarm, the lower the average number of informants of a given particle in respect of K. For example, for a swarm of 20 particles, with K = 3 one finds that the average size of the group of informants is 2.85, whereas it is 2.71 for a swarm of 10 particles.

This is relevant when one decreases the size of the swarm in the hope of reducing the total number of evaluations needed to achieve the goal. With fewer particles, the swarm is certainly a little less ready to explore the search space, but there is a kind of automatic partial offsetting by the correlative reduction of the average size of the groups of informants. As we have seen and will examine further, this reduction actually encourages exploration by increasing diversity.

3.1.3. *Initialization*

Note that, for the moment, we are interested only in continuous problems with real variables. A search space is defined, for example, classically, like one (hyper)cube of the form $[x_{\min}, x_{\max}]^D$. We will see, in Chapter 12, how it is possible to define much more general search spaces (with discrete variables and more complex forms) without changing the guiding principles of the method.

Initialization simply consists of initially randomly placing the particles according to a uniform distribution in this search space. This is a stage that one finds in virtually all the algorithms of stochastic iterative optimization.

But here, moreover, the particles have velocities. By definition, a velocity is a vector or, more precisely, an operator, which, applied to a position, will give another position. It is in fact a displacement, called velocity because the time increment of the iterations is always implicitly regarded as equal to 1.

In practice, it is not desirable that too many particles tend to leave the search space as early as the first increment, or for that matter later. We will see below what occurs in this case, but, for the moment, let us be satisfied with deriving at random the values of the components of each velocity, according to a uniform distribution in:

$$\left[\left(x_{\min}-x_{\max}\right)/2,\left(x_{\max}-x_{\min}\right)/2\right]$$

3.1.4. *Equations of motion*

The dimension of the search space is D. Therefore, the current position of a particle in this space at the moment t is given by a vector $x(t)$, with D components. Its current velocity is $v(t)$. The best position found up to now by this particle is given by a vector $p(t)$. Lastly, the best position found by informants of the particle is indicated by a vector $g(t)$. In general, we will write simply x, v, p, and g. The d^{th} component of one of these vectors is indicated by the index d, for example x_d. With these notations, the equations of motion of a particle are, for each dimension d:

$$\begin{cases} v_d \leftarrow c_1 v_d + c_2\left(p_d - x_d\right) + c_3\left(g_d - x_d\right) \\ x_d \leftarrow x_d + v_d \end{cases} \qquad [3.1]$$

The confidence coefficients are defined in the following way:

– c_1 is constant (confidence in its own movement);

– c_2 and c_3 (respectively confidence in its best performance and that of its best informant) are randomly selected with each step of time according to a uniform distribution in a given interval $[0, c_{max}]$.

This is why equation [3.1] can be rewritten in a more explicit way, by highlighting the random elements:

$$\begin{cases} v_d \leftarrow c_1 v_d + c_{\max} alea(0,1)\left(p_d - x_d\right) + c_{\max} alea(0,1)\left(g_d - x_d\right) \\ x_d \leftarrow x_d + v_d \end{cases} \qquad [3.2]$$

To use this model, the two parameters c_1 and c_{max} must be defined. The latter can be regarded as the maximum confidence granted by the particle to any performance transmitted by another. For each problem, "the right" values can be found only by experiment, with the help, however, of two empirical rules, made available after many tests.

The first rule stipulates that c_1 must have an absolute value less than 1. It is understood intuitively if one considers what occurs in the course of several successive time increments, in the specific case where the particle is and remains itself its best informant. We then have $p_d = x_d = g_d$ and, with each increment, velocity is simply multiplied by c_1. If its absolute value is greater than 1, velocity increases unceasingly and convergence is impossible. Note that, in theory, nothing prevents this coefficient being negative, the behavior obtained being strongly oscillatory, but this is never the case in traditional PSO. So, we will assume it to be positive.

In practice, this coefficient should be neither too small, which induces a premature convergence, nor too large, which, on the contrary, can slow down convergence excessively. The authors of the first work on PSO recommended that it be equalized to 0.7 or 0.8.

The second rule states simply that the parameter c_{max} should not be too large, a value of about 1.5 to 1.7 being regarded as effective in the majority of cases. When it was originally stated, this rule did not have a justification, even an intuitive one. It was purely experimental.

In fact, the recommended values are very close to those deduced later from mathematical analyses showing that for a good convergence the values from c1 and c_{max} should not be independently selected [CLE 02, TRE 03, VAN 02]. For example, the pairs of values (0.7 1.47) and (0.8 1.62) are indeed correct. The first experimenters, James Kennedy and Russel Eberhart, with the possible addition of Yuhui Shi [SHI 9a], did good work! The existence of this relation between these two parameters will help us later establish performance maps in only two variables: a parameter φ and the size of the swarm.

3.1.5. *Interval confinement*

During the first experiments of PSO, the test functions used were defined for all values. For example, the function:

$$f(x) = \sum_{d=1}^{D} x_d^2$$

(historically called Sphere, but which is in fact a paraboloid) in any point of real space R^D can be calculated. During the evolution of the swarm, it may have happened that a particle left the search space as initially defined, but that was of no importance, since the value of its position could in fact still be calculated without "crashing" the data-processing program. Nevertheless, obviously, that is not always the case. For example, in the majority of programming languages and with the majority of compilers, the evaluation of a function such as:

$$f(x) = \sum_{d=1}^{D} \sqrt{x_d}$$

returns an error message as soon as one of the coordinates x_d is negative.

More generally, a number of functions have a space of definition that is not infinite. Consequently, it was necessary to add very quickly a mechanism to prevent a particle leaving the search space. The simplest is the *interval confinement*. Let us always assume, for the sake of simplicity, that the search space is $[x_{\min}, x_{\max}]^D$. Then this mechanism stipulates that, if a coordinate x_d calculated according to equations of motion [3.2] leaves the interval $[x_{\min}, x_{\max}]$, one allots to it the nearest value of the border point. In practice, therefore, it amounts to replacing the second line of [3.2] by:

$$x_d \leftarrow MIN\left(MAX\left(x_d + v_d, x_{\min}\right), x_{\max}\right) \qquad [3.3]$$

However, this simple form, while giving correct results, has a disadvantage. Indeed, we are in a scenario where the proper velocity of the particle tends to make it leave the search space. Confinement [3.3] certainly brings back the particle to the border of the search space, but does not change its velocity. This is calculated again and thus in general is modified next time, but it is not uncommon for it to remain oriented more or less in the same direction. Thus the particle will tend to cross the border again, be brought back to that point by confinement, and so on. In practice, it will be as though it "were stuck" to this border.

That is why one must supplement the mechanism of confinement with a velocity modification. One can replace the component that poses a problem by its opposite, possibly balanced by a coefficient less than 1, or one can simply cancel it. If cancellation is chosen, the complete mechanism is then described by the following operations:

$$x_d \notin [x_{\min}, x_{\max}] \Rightarrow \begin{cases} v_d \leftarrow 0 \\ x_d < x_{\min} \Rightarrow x_d \leftarrow x_{\min} \\ x_d > x_{\max} \Rightarrow x_d \leftarrow x_{\max} \end{cases} \qquad [3.4]$$

The adaptation is immediate in case the intervals defining the search space are different for each dimension. But what is to be retained above all is the very principle of confinement, which stipulates that "if a particle tends to leave the search space, then bring it back to the nearest point in this space and consequently modify its velocity". We will see in particular that this principle can be used to define confinements necessary to problems in non-null granularity (positions with integer values, for example) or to problems (typically combinatorial) whose solutions must have all coordinates different.

3.1.6. *Proximity distributions*

What is the consequence of introducing random coefficients into equations of motion? For a better understanding, let us consider all the possible displacements obtained while varying independently c_2 and c_3 between 0 and $c_{\max}$. Let us call $\tilde{p}$ the vector whose d^{th} component is:

$$alea\left(0, c_{\max}\right)\left(p_d - x_d\right)$$

and $\tilde{g}$ the one whose d^{th} component is:

$$alea\left(0, c_{\max}\right)\left(g_d - x_d\right)$$

It is easy to see that if one places the origin of $\tilde{p}$ (respectively $\tilde{g}$) in x, its end then traverses a D-parallelepiped whose two opposite tops are x and $c_{max}p$ (respectively $c_{max}g$). This D-parallelepiped is called the *proximity* of p (respectively g). It is an example of formalization of what we described in the preceding chapter by using the expression "towards . . .".

The distribution of the possible points in the proximities of p and g is uniform. On the other hand, the distribution of the new possible positions for the particle, even if its field is also a hyperparallelepid, is not itself uniform. Indeed, for a given dimension d, the random variable whose occurrence is the d^{th} component of the new velocity is the sum of two random variables having each one a density of constant probability on an interval. To clarify these ideas, let us suppose that one has $p_d < g_d$ and $v_d = 0$. Then the probability density of the sum of these two variables has a trapezoidal form. It increases linearly on $\left[0, c_{\max} p_d\right]$, from 0 to p_d / g_d, preserves this last value in the interval $[c_{max}p_d,\ c_{max}g_d]$ then decreases linearly to 0 on the interval $[c_{max}g_d,\ c_{max}\ (p_d + g_d)]$. The resulting distribution thus makes it a "truncated pyramid", whose center is at the point $\left(c_{\max}\left(p_1 + g_1\right)/2, c_{\max}\left(p_2 + g_2\right)/2\right)$. It is uniform on a rectangle and decreases

linearly beyond the edges of this rectangle. Figure 3.1 shows a sample of 1,000 points in the proximity of p, 1,000 points in that of g and 1,000 next possible positions which result from this by linear combination.

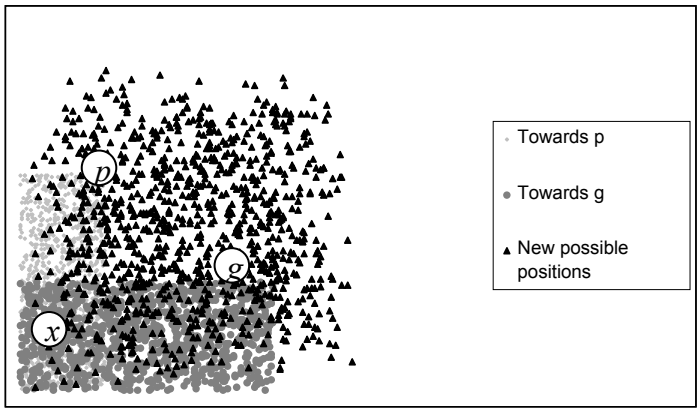

Figure 3.1. *Example of proximities in two dimensions. The proximity of p (the best position found up to now by particle x) is a rectangle of which one of the tops is x and the other* $c_{max}(p - x)$ *and the distribution of possibilities is uniform there. Similarly for g (the best position found by informants of x). By linear combination, one obtains the next possible positions of the particle. Their envelope is also a rectangle, but the distribution is not uniform there (less dense on the edges). To clarify the Figure, the velocity of the particle was assumed to be null and for each distribution only a sample of 1,000 points was represented*

Let us emphasize this concept of the distribution of the next possible positions or, briefly, the *distribution of the possibles.* This is the basis of all the algorithms of iterative optimization calling for randomness (stochastic). With each time increment, certain positions are known and starting from this information, it is a question of choosing the next position(s) for it (or them). Whatever the method used to work out the answer, the result is always of the same type: a set of candidate positions, each one being assigned a probability of being selected.

This is why it is so important, for any method of this type, to examine carefully the distributions obtained with each increment and to ask whether they can be made more effective. For PSO, we will see that this step easily induces interesting improvements. *A contrario*, let us quickly mention two rather common errors that impoverish the distributions of the possibles.

3.2. Two common errors

The equations of motion [3.2] are sometimes written in vectorial form:

$$\begin{cases} v \leftarrow c_1 v + alea(0, c_{\max})(p - x) + alea(0, c_{\max})(g - x) \\ x \leftarrow x + v \end{cases} \quad [3.5]$$

In this case, in accordance with the definition of the multiplication of a vector by a coefficient, it means that all the components, for example vector $p - x$, are multiplied by the same random number. This is an error in the sense that it corresponds to an algorithm different from that of PSO, but we can also regard this form as an alternative. It should, however, be noted that the best parameter settings for c_1 and c_{max} bypass the use of a constriction coefficient (see Chapter 6) and that this alternative is then much less effective than the classic form.

The proximity of p (respectively g) is a simple segment here and the distribution of possibles for the next displacement is a D-parallelepiped located "between" p and g (these two points are on its surface), which restricts exploration, in particular because an entire set of points located close to p (respectively g) has no chance of being selected.

The other error, or alternative, consists of carrying out a factorization in the first equation of motion:

$$v_d \leftarrow c_1 v_d + alea(0, c_{\max})(p_d + g_d - 2x_d) \quad [3.6]$$

or:

$$v_d \leftarrow c_1 v_d + alea(0, 2c_{\max})\left(\frac{p_d + g_d}{2} - x_d\right) \quad [3.7]$$

In this form, we see that the next position will then be taken randomly according to a uniform distribution in a hyperparallelepid whose edge for dimension d is length $c_{\max}|p_d + g_d|$ and whose center is found by adding to vector x the vector $c_1 v + c_{\max}(p + g)/2$. Actually, one could simply describe this as an alternative rather than an error, because this distribution is almost as rich as the original.

3.3. Principal drawbacks of this formulation

The repeated experiments using the version of PSO defined by equations [3.2] and [3.4] (the version that, for brevity, we will name OEP 0) highlight certain insufficiencies or anomalies that can also be seen as ideas for improvements in subsequent versions.

3.3.1. *Distribution bias*

We saw that, with each time increment and for each particle, the distribution of possibles is non-uniform and of (hyper-)rectangular envelope. In itself, it would not be a defect if it corresponded at least to an empirical rule, aiming, for example, to favor a certain area of the search space. For example, one might think of searching "preferentially" around one of the two best-known positions of the particle (p and g) or "around" a point located between p and g, but closer to g than p, etc.

However, this is not the case. There is no reason why the median point of the distribution obtained should be at the center of a "promising" area. Actually, the very particular form of this distribution is an artifact resulting only from the simple choice of coding of random elements. Since the majority of data-processing languages have only the function *alea* (0,1), one immediately has *alea* (0, c_{max}) = c_{max} *alea* (0,1). However, coding a distribution of different envelope (spherical, for example) is appreciably more difficult, at least if the computing time is not to increase exponentially with the number of dimensions. We will see examples of this later.

Moreover, it should be noted that this distribution depends on the coordinate system (see Figures 3.2 and 3.3). If by bad luck the point p is on a coordinate axis, the D-rectangle of its proximity loses a dimension. For a problem with two dimensions, for example, it is reduced to a segment. A simple rotation of the coordinate system completely modifies the whole ensemble of next possible positions and thus strongly influences the behavior of the particles. Convergence is as likely to be accelerated as slowed down, but, again, in an unforeseeable way.

This phenomenon is often concealed, because the majority of traditional test functions are symmetrical around the origin of the coordinates.

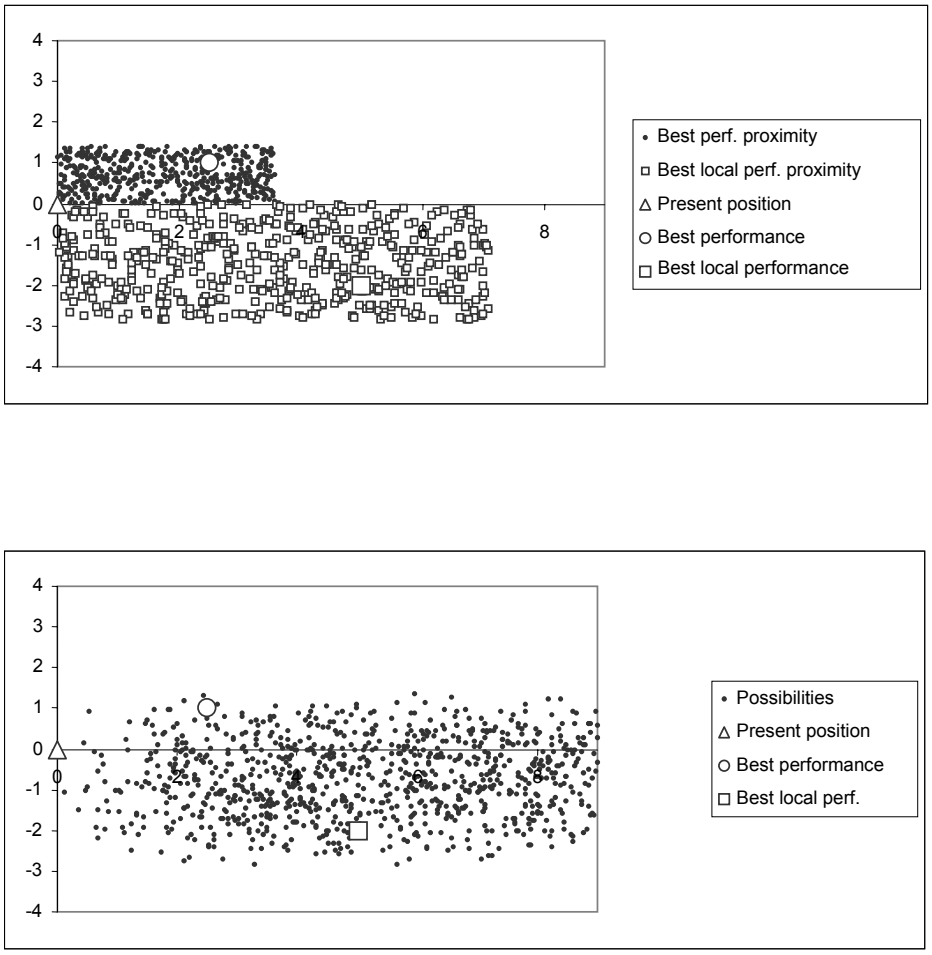

Figure 3.2. *Distribution of the next possible positions. The upper diagram shows each of the two elementary distributions and the lower their combination (sample of 1,000 points)*

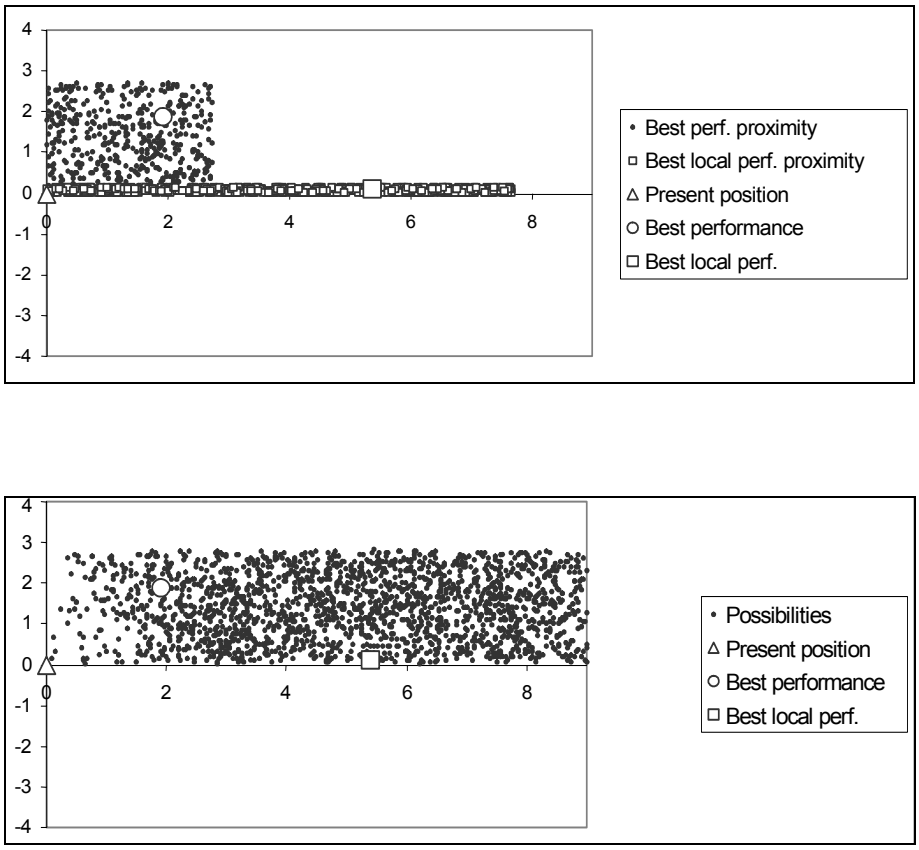

Figure 3.3. *Depending on the coordinate system chosen, the distribution of the next possible positions can be very variable. Here, a rotation of the coordinate axes was carried out, one of the axes practically aligning itself on the vector g − x*

The second bias led to alternatives privileging distributions with a center of symmetry (spheres, Gaussian, etc.) or whose form depends only on the respective positions *x*, *p,* and *g* (Gaussian "distorted"). To mitigate the first bias at the same time, these distributions are placed in a way that is *a priori* wiser. For example, by centering them on the segment $p - g$ and a little closer to *g* than *p*, one can hope to take advantage of a possible favorable "gradient effect" from *p* towards *g*.

3.3.2. *Explosion and maximum velocity*

If one does not want to subject oneself to a parameter c_1 less than 1, to support exploration, then it is necessary to face the phenomenon of the "explosion" of the swarm. Indeed, *roughly speaking*, as we saw, with each time increment velocity is multiplied by c_1. If this coefficient is greater than 1, then it will tend to increase more and more. That is why certain authors introduce an additional parameter, in the form of a maximum velocity: any velocity tending to exceed it is brought back to it. This maximum velocity v_{max} is a real number, which can be different for each dimension. An empirical rule requires that, for a given dimension, one takes it to be equal to half the range of possible values for the search space. Any larger value would ensure that the particles are made to leave the search space too often.

For example, if for a dimension d the search space is the interval [0.5], one will take a maximum velocity of 2.5 for this dimension. It means that if the first calculation of equation [3.2] gives a velocity v_d greater than 2.5, one will take it instead to equal 2.5. If the values are discrete, for example $\{0,1,...,5\}$, the greatest extent covered by the possible values remains from 0 to 10, but the maximum velocity could be selected as being 2 or 3.

Unfortunately, whoever says "additional parameter" says also "choice of this parameter", which still complicates the task of the user a little, since, in OEP 0, all the parameters are up to him.

3.4. Manual parameter setting

Table 3.1 recapitulates the various parameters of the model which have to be defined and the few empirical rules which could be worked out to guide the choice. These rules are very approximate and, for a given problem, we are faced with the strong possibility of searching at length before finding a "good" set of parameters. The good news, nevertheless, is that PSO is very robust, in the sense that broad variations in the parameters do not prevent convergence, even if, of course, it can be more or less rapid.

In this respect, in the majority of the problems, the informant group size is the parameter to which the behavior of the swarm is the least sensitive. One can take it systematically equal to 3 without much risk. Even if this is not the best value for your precise problem, the performances, in general, are not seriously degraded as a result. Nevertheless, if you are sure that the function to be minimized does not present local minima, you will probably find it beneficial to increase this value, to even consider that each particle informs all the others and thus to take it equal to N.

Parameter	Title and nature	Empirical rule of choice and comment
c_1	Self-confidence; real number	In]0,1[. Suggestion: 0.7
c_{max}	Confidence in others; real number	About 1.5. Suggestion: 1.43
N	Swarm size; integer	From 20 to 40. Suggestion: 20
K	Group size of informed; integer	From 3 to 5. To N for the simple problems without local minima. Suggestion: 3
$v_{\max}$	Maximum velocity; real number	Essential only if c_1 is greater than 1. Value about half of $x_{\max} - x_{\min}$. Possibly different for each dimension.

Table 3.1. *Parameters of OEP 0. The fifth, maximum velocity, is useful only if one wants to force a greater exploration of the search space by balancing velocity by a "self-confidence" greater than 1*

The number of evaluations of the function to be minimized is equal, with each time increment, to the number of particles. Consequently, the degradation of the performances according to this criterion is at most proportional to the size of the swarm. Actually it is often much less, since the increase in the number of particles also increases the probability of finding a solution more quickly. That is why the recommended values 20 to 40 are very generally satisfactory.

For the two parameters of confidence, precise values are suggested. As indicated previously, they form a pair initially found in experiments but subsequently confirmed mathematically. Other values are naturally possible and it is even possible, by choosing them judiciously, more or less to induce a given behavior of the particles, in particular oscillating or not around a solution [TRE 03, VAN 02].

3.5. For "amatheurs": average number of informants

One supposes that each particle of a swarm of total size N randomly chooses, with putting back, K particles to be informed. The probability that a particle is not selected is $p = (1 - 1/N)^K$ and the probability that it is selected is $q = 1 - p$.

Let s be the number of informants of a given particle. The probability that s is null is the probability that it is chosen by nobody, i.e. neither by particle 1, nor by particle 2 . . . nor by particle N. This probability is thus p^N.

In the same way, for s to equal 1, it must be chosen by one particle (N possibilities) and not chosen by all the others. Its probability is thus $Np^{N-1}q$. More

generally, for an unspecified value of s between 0 and N, the probability is $C_N^s p^{N-s} q^s$, where C_N^s is the number of combinations of s elements among N.

Thus, finally, by taking the sum of the possible values weighted according to their probability, the average value of the number of informants is:

$$\sum_{s=0}^{N} sC_N^s p^{N-s} q^s = \sum_{s=0}^{N} sC_N^s \left(1-1/N\right)^{K(N-s)} \left(1-\left(1-1/N\right)^K\right)^s$$

From a graph theory point of view, it is the average number of ancestors by node when, in a graph of size *N*, the arcs are built by randomly taking *K* downward for each node. Figure 3.4 shows, for $K = 3$, the evolution of this value according to *N*.

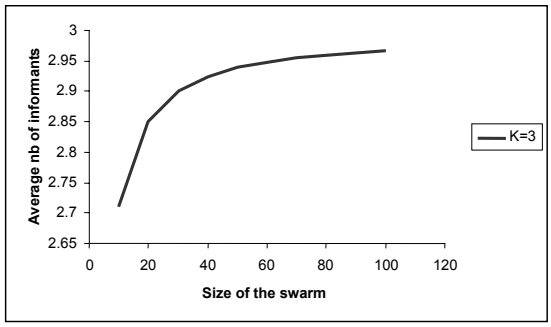

Figure 3.4. *Average number of informants by particle when each particle informs K others at random, according to the size of the swarm.*
Here K = 3. This number is all the less than K as the swarm is small

3.6. Summary

From the basic principles presented in the preceding chapter, we propose a first simple formulation, called OEP 0, which specifies the rules of displacement of the particles. The information links between particles are randomly selected with each iteration. The equations of motion combine linearly, thanks to confidence coefficients, vectors of position randomly drawn according to non-uniform distributions whose supports are (hyper-)rectangles in the search space.

The various parameters (size of the swarm, coefficients, number of informed particles chosen at random, etc.) depend entirely upon the user for the moment and some semi-empirical rules are given to guide these choices.

Certain insufficiencies of this first version are noted here. Highlighting them will guide the improvements brought about later on.