Cooperation Mechanisms in Particle Swarm Optimisation

Maurice Clerc

Cooperation Mechanisms in Particle Swarm Optimisation

Maurice Clerc

Abstract

We define five cooperation mechanisms in Particle Swarm Optimisation, loosely inspired by some models occurring in nature, and based on two quantities: a *help matrix*, and a *reputation vector*. We call these five mechanisms, respectively, Reciprocity, Vicinity, Kin, Reputation, and Anybody. It appears that Kin is better than the rest by a slight margin, but needs more parameters that have to be tuned (mutation and selection). However, Reciprocity, with less parameters, shows almost equivalent performance. The appendix gives some details about fair comparison of success rates, and the concepts of *valued topology* and *chains of information*, which may be worth further investigation.

1 Introduction

When a bee finds an interesting place, it comes back to the hive, and dances to inform some other bees. But which ones, and how many? As far as we know, it is quite random, and only a few bees are informed; these being just the bees which are present at the hive at that moment. This is the idea that lies behind the variable topology that has been defined in Clerc (2005, 2006), used by Standard PSO since 2006 PSC, and generalized under the name of stochastic star in Miranda et al. (2008). More formally, at each time step, and if there has been no global improvement, an agent A may help (inform) an agent B with a probability p. In SPSO, the rule is: "each agent informs, at random, K other agents", so the probability is

$$p_{SPSO} = 1 - \left(1 - \frac{1}{S}\right)^K \tag{1.1}$$

where S is the population size. Note that this set of information links is modified after an iteration only if there has been no global improvement (i.e. the best position known by the swarm has not been improved). The Stochastic Star of Miranda et al. (2008) uses exactly the opposite criterion, that is, the set of information links is modified after an iteration only if there has been a global improvement, but the two methods seem to be equivalent. At least, there are no

significant differences on the test functions that they use in their paper. In fact, what seems to be important is to not modify the topology (the set of information links) too often.

However, there are a lot of other cooperation mechanisms in real life, and it would be interesting to apply some of them to PSO, in order to see if they are better in the specific case of optimisation.

2 Five cooperation strategies

In Nowak (2006), Nowak defines five mechanisms for the evolution of cooperation in a population. Loosely inspired by this paper, we define here five cooperation rules in a population based optimiser. Each rule given below is complete in itself and is independent of the others.

1. Reciprocity Rule: An agent A may help an agent B if B has helped A in the past
2. Vicinity Rule: An agent A may help an agent B if B is a neighbour of A
3. Kin Rule: An agent A may help an agent B if B is similar to A
4. Reputation Rule: An agent A may help an agent B if A has a good "reputation"
5. Anybody Rule: An agent A may help an agent B if B is any agent

3 Implementation in PSO

In order to implement these rules in PSO, we need algorithmic definitions that can be coded. Below, we give a few such possible definitions. We use the word "possible" as these are non-unique; one may come up with alternative definitions. The definitions given here are based on a *help matrix* $H\,(S \times S)$, and/or a reputation vector $R\,(S)$, where S is the swarm size.

When a particle i has to move, it tries to get some information from the particles j (i.e. the best position known by j), according to a probability vector $P\,(S)$ defined as follows:

- if the help matrix is used, P is the result of a normalisation of the $H\,(*, i)$ column, seen as vector;
- if the reputation vector is used, P is the result of a normalisation of the reputation vector.

Each cooperation mechanism uses a specific normalisation method. Then the rule is:

if $rand\,(0,1) < P\,(j)$, i indeed receives information from j

In this study, we apply the standard PSO method to use the information sent by the informants: particle i builds the list of the particles j that send information to it, and select only the best one, to use it in the velocity update formula. Let us see now how H or R can be defined for our five cases. All the rules below are quite rudimentary and can of course be replaced by more sophisticated ones.

3.1 Reciprocity

At the very beginning, all elements of the help matrix have the same initial value h_{init}. After that:

- during each iteration, if j indeed helps i, then $H\,(i,j)$ is incremented by h_{incr};
- after each iteration, all elements of H are decremented by h_{decr} (but with zero as lower bound), in order to simulate a kind of oblivion phenomenon. Note that even for a particle$j$ that has helped particle i in an iteration, and was therefore incremented in $H\,(i,j)$, we decrement the value;
- the rules of the normalisation are defined below:

$$\begin{cases} H\,(j,i) > h_{init} & \Rightarrow & P\,(j) = 1 \\ & else & P\,(j) = h_{min} \end{cases} \tag{3.1}$$

Actually, some parameters are common to several mechanisms. In the tests below, we have used the following set:

$$\begin{cases} h_{init} &=& 0.5 \\ h_{incr} &=& 2 \\ h_{decr} &=& 1 \\ h_{min} &=& adaptive \end{cases} \tag{3.2}$$

Some preliminary tests show that a constant h_{min} is not good for any swarm size and any problem. That is why we use an adaptive h_{min}, which is computed by the following empirical formula:

$$h_{min} = \max\left(0, \frac{1}{S - s_i}\right) \tag{3.3}$$

where s_i is the number of times $H\,(i,j)$ is greater than h_{init}. This means that while counting s_i, we ignore the past and only compare the current values. Thus, at every step, we simply check if $H\,(i,j) > h_{init}$, and if so, s_i is incremented by 1.

3.2 Vicinity

Let D be the dimension of the search space. A particle receives information from the D nearest ones. If D is bigger than S, all particles are possible informants. Therefore, if j is one of the D nearest neighbours of i, then $P(j) = 1$, otherwise $P(j) = h_{min}$. The adaptive h_{min} is defined in a similar way as in Reciprocity:

$$h_{min} = \max\left(0, \frac{1}{S-D}\right) \tag{3.4}$$

When h_{min} is not null, it means that even particles that are not the nearest ones have a slight chance to be informants.

3.3 Kin

This is the most complicated one, for it needs mutation and selection. Though it is commonly said that the position and velocity of a particle structurally define it, strictly speaking, it is not correct. Instead, the confidence coefficients that a particle uses in the classical velocity update formula of PSO define its structure rigorously. Here, we do assume that these coefficients can be different for each particle (thus, each particle can be structurally different). So, every particle has three "genes":

- w, the inertia weight
- c_1, the cognitive confidence coefficient
- c_2, the social confidence coefficient

With the "genes"defined, let us start the formulation of the process. First, we define the initial values: at the very beginning, all particles have the same genes $(w_{init}, c_{1,init}, c_{2,init})$. Second, we define a common mutation mechanism for each gene g, which could be w, c_1, or c_2

$$g \leftarrow g + g\rho\,(1 - 2rand\,(0,1)) \tag{3.5}$$

where ρ is the mutation rate in $]0,1]$ As we can see, g can either slightly increase or decrease.

Third, we define a "genetic" distance measure between two triplets $(w_i, c_{1,i}, c_{2,i})$ and $(w_j, c_{1,j}, c_{2,j})$ associated with two particles x_i and x_j

$$d_g\,(x_i, x_j) = \sqrt{\left(\frac{w_i - w_j}{w_{init}}\right)^2 + \left(\frac{c_{1,i} - c_{1,j}}{c_{1,init}}\right)^2 + \left(\frac{c_{2,i} - c_{2,j}}{c_{2,init}}\right)^2} \tag{3.6}$$

Fourth, we define a selection rate ς in $]0, 0.5]$.

Then the process is the following:

- as said, at the very beginning, all triplets of "genes" are the same. Therefore a particle may inform any other;

- after each iteration, the ς fraction of best particles generate "mutated" ones, which replace the ς fraction of worst particles;

- for each particle i, and for the D genetically closest particles j, $P(j)$ is set to 1. It is set to 0 for all other particles j. Here "genetically closest" means according to the d_g distance measure.

3.4 Reputation

The *reputation* of a particle j is a memorised estimate of how many times it has successfully helped another particle. At the very beginning, all elements of the reputation vector R have the same initial value r_{init}. After that:

- during each iteration, if j indeed helps i, and if then i improves its position, then $R(j)$ is incremented by r_{incr} ;

- after each iteration, all elements of R are decremented by r_{decr} (but with zero as lower bound), in order to simulate a kind of oblivion phenomenon. Note that, as in Reciprocity, we apply this decrement even to a particle j that has helped i during an iteration and was therefore incremented during the iteration.

- the rules of normalisation are defined similarly as in Reciprocity:

$$\begin{cases} R(j) > r_{init} & \Rightarrow & P(j) = 1 \\ & else & P(j) = r_{min} \end{cases}$$

For simplicity, the parameter r_{init}, r_{incr}, r_{decr} are set to the same values as h_{init}, h_{incr}, h_{decr}. And r_{min} is defined as h_{min} in Reciprocity:

$$hr_{min} = \max\left(0, \frac{1}{S - s_i}\right) \tag{3.7}$$

where s_i is the number of times $R(j)$ is greater than r_{init}. As in Reciprocity, while counting s_i we ignore the past and simply check if $R(j)\, r_j > r_{init}$. If so, s_i is incremented by 1.

3.5 Anybody

This is the easiest one. Any particle has the same (constant) probability to help any other one, which is here simply set to 1.

4 Experimentation

The above methods have been coded by starting from the C version of SPSO 2011 (PSC). Only the cooperation mechanism has been modified, everything else is the same, for fair comparison. The resulting CooPSO code is freely available on my PSO site Clerc. It has been tested on 13 functions of dimensions between 2 and 42 (see table 1). The exact definitions of the functions are not given here, but they can be found in the C code. The pseudo-random number generator used is KISS PRNG, by Marsaglia, and is included in the code.

4.1 Comparisons

What is important in this study is not the results themselves, but the comparisons between six cooperation mechanisms (including the one originally used in SPSO 2011). As SPSO 2007 is still often used for its simplicity, I also present the results with this variant. Note that SPSO 2007 makes use of the same cooperation mechanism as SPSO 2011, but the velocity update formula is different (for more details, see Standard PSO Descriptions on Clerc). For all variants the swarm size is $S = 40$, and the values of $(w_{init}, c_{1,init}, c_{2,init})$ are (0.721, 1.193, 1.193). For Kin, these are just the initial values, the mutation rate is $1/S$, and the selection rate is 0.5.

Let us comment on table 2. As usual, we can classify the problems into two groups: the ones for which the seven methods are more or less equivalent, and the ones for which some methods are pretty good, and some others very bad. The first group contains Tripod, Network, Perm, Sphere, Rastrigin, and Ackley, and the second group has the rest. On the whole, SPSO 2007 is the worst (mainly because of its bad performance on Step and Rosenbrock that are not compensated by excellent ones on some other problems). The cooperation scheme Anybody is also not very good.

Kin is the best by a very slight margin, and Reciprocity comes next; and Reciprocity is simpler. However, the differences are not very significant (it is even more obvious if you consider the results after 1000 runs, see 7.1, figure 7.3).

For the different mechanisms , the weaknesses are not always on the same problems. For example, Kin is rarely better than both Vicinity and Reciprocity. So defining some hybrids may be fruitful. Let us try that.

Table 1: Test functions.

Name	Search space	Max. eval.	Objective (adm. error)	Comment
Tripod	$[-100, 100]^2$	10000	0 (0.0001)	Two local optima
Network	$\{0, 1\}^{38} \times [0, 20]^4$	5000	0 (0)	Partly binary
Step	$[-100, 100]^{10}$	2500	0 (0)	Biased
Lennard-Jones	$[-2, 2]^{15}$	65000	-9.103852 (10^{-6})	Five atoms
Gear Train	$\{12, 13, \ldots, 60\}^4$	20000	$2.7 \times 10^{-12}(10^{-13})$	Discrete
Perm	$\{-5, -4, \ldots, 5\}^5$	10000	0(0)	Discrete
Compression Spring	$\{1, 2, \ldots, 70\}$ $\times [0.6, 3]$ $\times \{0.207, 0.208, \ldots, 0.5\}$	20000	$2.6254214578(10^{-10})$	Partly discrete
Sphere	$[-100, 100]^{30}$	300 000	-450(10^{-6})	Shifted CEC (2005)Unimodal
Rosenbrock	$[-100, 100]^{10}$	100 000	390(0.01)	Shifted CEC (2005)
Rastrigin	$[-5.12, 5.12]^{30}$	300 000	-330(0.01)	Shifted CEC (2005)
Schwefel	$[-100, 100]^{10}$	100 000	-450(10^{-5})	Shifted CEC (2005)
Griewank	$[-600, 600]^{10}$	100 000	-180(0.01)	Shifted CEC (2005) Not rotated
Ackley	$[-32, 32]^{10}$	100 000	-140(10^{-4})	Shifted CEC (2005) Not rotated

Table 2: Success rate and Mean best over 100 runs. Swarm size $S = 40$. For Kin, $\rho = 1/S$, and $\varsigma = 0.5$

	SPSO 2007	SPSO 2011	Reciprocity	Vicinity	Kin	Reputation	Anybody
Tripod	63%	79%	90%	90%	86%	48%	48%
	0.31	0.14	0.1	0.09	0.08	0.73	0.71
Network	0%	0%	0%	0%	0%	0%	0%
	106	109	151.8	163.1	160.7	159.2	150.7
Step	3%	99%	100%	100%	100%	100%	100%
	4.53	0.01	0	0	0	0	0
Lennard-Jones	68%	50%	100%	100%	100%	99%	100%
	0.077	0.168	7.09E-7	7.37E-7	7.46E-7	0.007	7.27E-7
Gear Train	16%	58%	40%	31%	53%	14%	4%
	2.47E-10	0.19E-10	0.49E-10	0.45E-10	0.14E-10	8.22E-10	12.1E-10
Perm	46%	36%	26%	26%	26%	10%	12%
	292	309	379.3	350.2	394.4	618.7	731.9
Compression Spring	72%	81%	73%	70%	72%	19%	14%
	0.0019	0.0032	0.002	0.0027	0.0043	0.069	0.063
Sphere	100%	100%	100%	100%	100%	100%	100%
	9.00E-7	0	9.01E-7	9.15E-7	9.18E-7	9.13E-7	9.18E-7
Rosenbrock	9%	50%	62%	55%	61%	69%	67%
*	*1.8*	*57.7*	*178.8*	102.31	*68.5*	*40.83*	*44.90*
Rastrigin	0%	1%	0%	0%	0%	0%	0%
	38.9	5.4	118.4	121.9	113.8	124.61	118.98
Schwefel	100%	100%	100%	100%	100%	100%	100%
	8.75E-5	0	0.87E-5	0.88E-5	0.87E-5	0.86E-5	0.86E-5
Griewank	18%	9%	1%	2%	9%	1%	1%
	0.030	0.021	0.076	0.073	0.041	0.099	0.123
Ackley	98%	100%	80%	75%	97%	63%	70%
	0.019	0	0.028	0.359	0.039	0.577	0.510
Mean success rate	**45.6%**	54.7%	**59.4%**	**57.6%**	**61.8%**	**47.9%**	**47.4%**

* For Rosenbrock, the Mean best is meaningless, for its estimation does not converge as number of runs increases. From time to time, a bad run occurs, with a very high final value.

Table 3: Results for some hybrids. For easier comparison, only the success rates are given. No hybrid is significantly better than all the "pure" mechanisms.

	Reciprocity ↔Vicinity	Reciprocity↔ Kin	Reciprocity ↔Reputation	Vicinity ↔Kin	Vicinity ↔Reputation	Kin ↔Reputation
Tripod	94	84	61	80	61	44
Network	0	0	0	0	0	0
Step	100	100	100	100	100	100
Lennard-Jones	100	100	100	100	100	100
Gear Train	34	45	21	49	30	18
Perm	17	31	29	28	22	16
Compression Spring	76	73	37	68	46	36
Sphere	100	100	100	100	100	100
Rosenbrock	57	56	66	54	60	74
Rastrigin	0	0	0	0	0	0
Schwefel	100	100	100	100	100	100
Griewank	2	7	0	11	3	7
Ackley	75	86	72	85	70	83
Mean success rate	**58.1**	**60.1**	**52.8**	**59.6**	**53.2**	**52.1**

5 Hybrids

There is a very simple way to define some hybrids: if a given variant does not give a global improvement for one iteration, try another one for the next iteration. Let us apply this method, by trying all unordered pairs chosen from the four best cooperation mechanisms (as Anybody is clearly not very good). As we can see from table 3, no such hybrid is significantly better. For a more synthetic view, we can sort our 17 mechanisms by increasing order of the mean success rate over the benchmark, and plot the results. Figure 5.1 shows that the mechanisms that use Vicinity, Reciprocity, Kin, or their hybrids are more or less equivalent, although Kin alone seems to be the best by a small margin.

6 Discussion

The cooperation method used in SPSO (2006, 2007, 2011, see PSC) can be compactly and roughly described as follows: "Each agent informs at random a few other agents". If, after a while, there is no improvement, another random selection is performed." It performs quite well (and, in passing, it may be interesting to

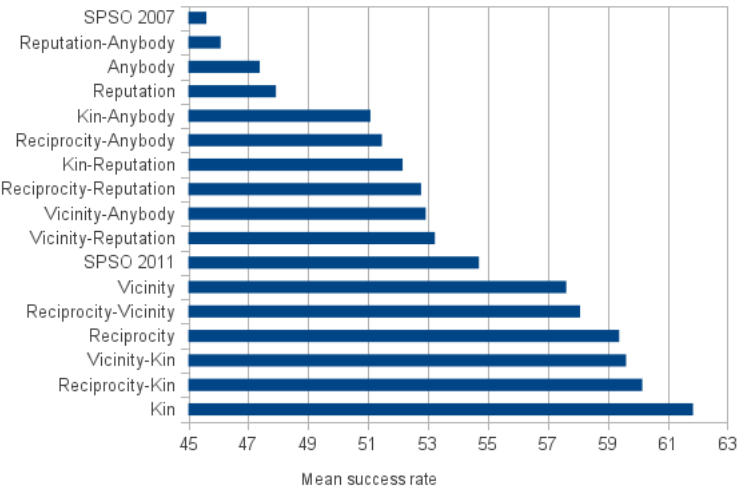

Figure 5.1: Comparison of the 17 mechanisms. SPSO 2007 is clearly the worst. Kin seems to be the best by a small margin, but in fact the three mechanisms Vicinty, Reciprocity, Kin, and their hybrids are more or less equivalent.

simulate the behaviour of a social system based on this kind of cooperation), but this study shows that some mechanisms coming from real life can improve the optimisation process, particularly Kin and Reciprocity. The first one needs to define two additional parameters, for mutation and selection. The second one is simpler, and on the whole almost equivalent. On the other hand, precisely because Kin has more parameters, it may be possible to find "good" values of these parameters to improve the performance even more (see 7.3).

Some concepts may need further investigation. In particular:

- the "genome" of a particle represented by the parameter values that it uses to compute its displacement, and not, as is usually done, by its position;

- the valued topology, which is a generalisation of the classical one, and which can be represented by a valued graph or, equivalently, by a "help matrix". In particular, the method presented here to build and update such a matrix is very rudimentary, and can certainly be improved.

7 Appendix

7.1 About the success rate

This study suggests that some cooperation mechanisms are better than some others, but the comparisons are not very rigorous. There is no statistical analysis like Wilcoxon, Friedman or t-test. In particular, how much can the success rates over 100 runs be trusted? After all, when you plot the curve "success rate" vs "number of runs", you always get some oscillations, before reaching a level that can be reasonably said to be "stable". Actually, it is easy to see that between r and $r+1$ runs, the "jump" of the estimated success rate can be as big as $100/r$. In other words, after one run, the success rate is necessarily either 0% or 100%, after two runs, only the three values 0%, 50%, 100% are possible, etc.

As we can see from figure 7.1, even with 100 runs such a stabilisation is not completely certain. In passing, note that a lot of studies make use of 30 runs or even less. In such a case, the estimation of the success rate may be quite bad. On the example here, it would be 73% after 30 runs, while the true value is about 61%.

In the literature, a standard deviation or a confidence interval is usually given with the mean best value, so that the reader may have an idea of how precise the estimate is. However, it is rarely done when the success rate is used as the comparison criterion. Let us try to do it.

We can perform 100 runs 100 times. Then, we get 100 success rate values, from which we can have an idea of its distribution, compute a mean value, and a standard deviation. For the Rosenbrock function, and the Reciprocity mechanism, the result is given in figure 7.2. Here, the mean is 61.7%, and the standard deviation 4.3%. A complete analysis is possible, for the estimated success rate follows a binomial law, but it is out of the scope of this paper. However, it is worth nothing if we want a standard deviation not greater than 1%. In such a case, we must perform at least 1000 runs (with only 30 runs, the standard deviation would be greater than 6%). Fortunately, as we can see from figure 7.3, for the four best mechanisms, the order is *almost* the same (the last two exchange places, but the standard deviation is only one percent, and thus the difference is not significant), even if the mean success rate values are smaller.

Also note that the performance criteria, including the success rate, may largely depend on the random number generator that is being used. That is why the RNG must be seen as a part of the algorithm Clerc (2012).

7.2 About the Reciprocity mechanism

To initialise the mechanism, we have to assign a non null probability h_{init} to any particle so that it can be an informant. After that, though, because of the decay process, this probability may become h_{min}. What happens if h_{min} is equal to zero, i.e. what happens if a particle tends to inform only the ones that have already

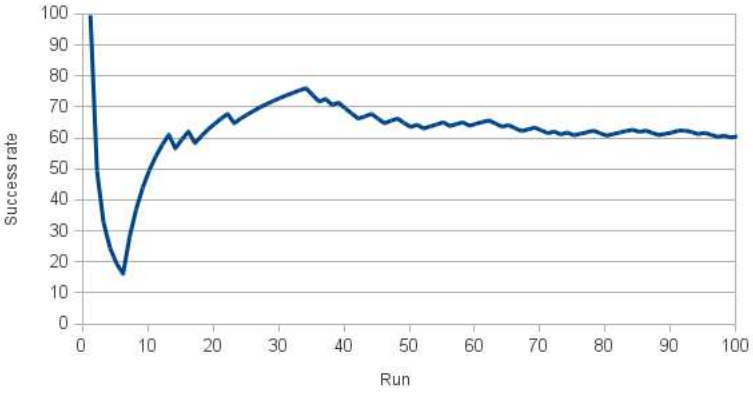

Figure 7.1: Evolution of the success rate over 100 runs (Rosenbrock, with Kin). After 30 runs, the success rate is 73%, but in fact the true value is about 61%.

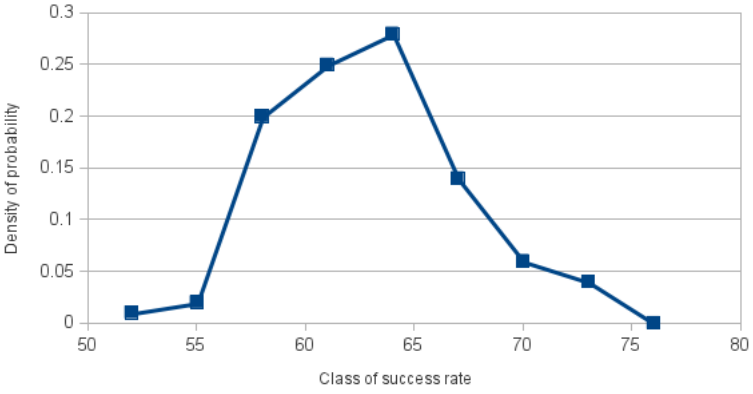

Figure 7.2: Rosenbrock with Reciprocity. Distribution of the success rate over 100x100 runs.

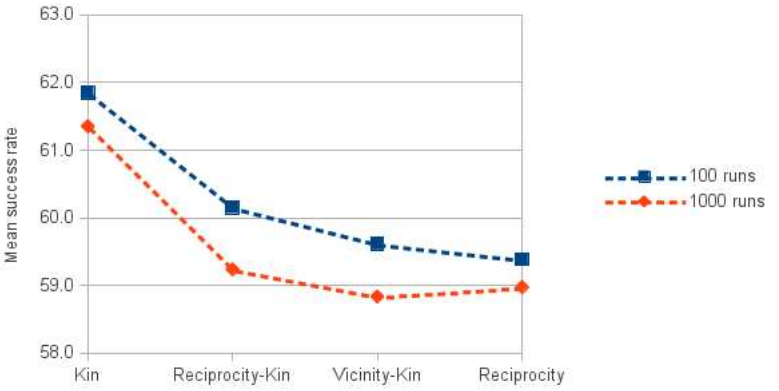

Figure 7.3: Mean success rate for 100 runs and 1000 runs. Comparison for the four best mechanisms. The improved estimates, after 1000 runs, are always a bit smaller than after 100 runs. The two best mechanisms keep their rank. For the two others, the order changes, but the difference is not very significant.

informed it? As we can see from table 4, the mean performance then is not as good, and on some problems it is very poor .

More generally, if h_{min} is kept constant the performance depends on its value. Is there an optimal one? Let us try different values, and plot the curve "Mean success rate" vs h_{min}. There is indeed a best value, near 0.024, which leads to a global performance very near to the optimal one (at least on average for the 13 benchmark functions used here), both for a swarm size $S = 20$, and $S = 40$, as we can see from figure 7.4.

For all swarm sizes, a constant $h_{min} = 0$ would be the worst choice. From a "sociological " point of view: helping *only* people that have helped you (except at the very beginning) is not very efficient, and with just a bit of altruism, the efficiency increases.

Anyway, using the same h_{min} value is just a compromise, for the true optimal value in fact depends on the swarm size, and on the problem being solved. That is why we use adaptive formulae here. Even if they are quite simple, the mean performance is improved. For example, for Reciprocity, and $S = 40$, we have seen it reaches 60.1%, whereas with a constant h_{min} its maximum value is 59.1%.

7.3 About the Kin mechanism

As we have seen, Kin is globally the best though slightly so, and the most robust. This is probably due to the permanent adaptation of the parameters (w, c_1, c_2) , but to what extent? The diagrams in figure7.5 for the Tripod problem show that

Table 4: Pure Reciprocity, i.e. with $h_{min} = 0$. The cooperation mechanism is then almost always less efficient.

	Success rate over 100 runs	Mean best
Tripod	100	7.04E-5
Network	0	161.6
Step	100	0
Lennard-Jones	100	7.45E-7
Gear Train	9	1.05E-9
Perm	8	454.81
Compression Spring	31	0.016
Sphere	100	9.14E-7
Rosenbrock	54	70.49
Rastrigin	0	118.95
Schwefel	100	8.77E-6
Griewank	6	0.059
Ackley	69	0.498
Mean success rate	**52.1**	

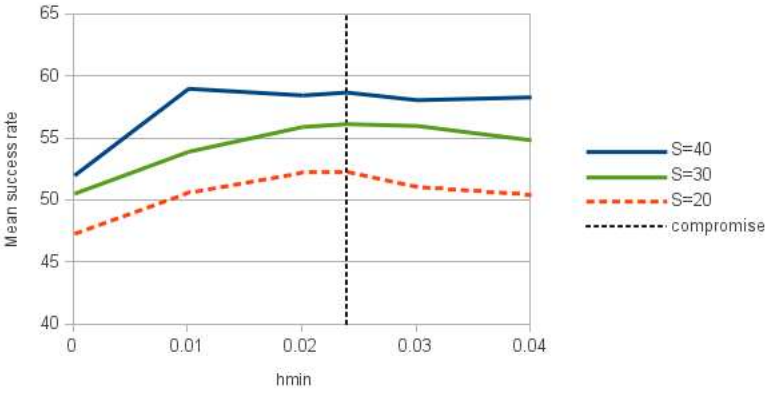

Figure 7.4: If h_{min} is kept constant, the average mean performance (here for Reciprocity) depends on its value and on the swarm size S, and there is an optimum for each swarm size. A h_{min} value near to 0.024 leads to nearly optimal performances, but is of course just a compromise. Using an adaptive h_{min} is a better approach.

Table 5: Parameter tuning for Kin. A small mutation rate ρ_w for the inertia weight w is enough, but by using a bigger one ρ_c for the cognitive and social coefficients c_1 and c_2, the performance can be slightly improved. The selection rate is kept constant (0.5).

w	c_1	c_2	ρ_w	ρ_c	Mean success rate over the 13 test functions
0.721	1.481	1.481	0.025	0.025	59.3%
0.721	1.193	1.193	0.025	0.025	61.8%
0.721	1.193	1.193	0.025	0.1	61.6%
0.721	1.193	1.193	0.025	0.2	**63.3%**
0.721	1.193	1.193	0.025	0.3	60.4%
0.4	0.4	0.4	0.5	0.5	34.8%
0.4	0.4	0.4	0.5	1	39.7%
39.70.4	0.4	0.4	1	1	37.7%

they are not largely modified during a run, just oscillating for w, which suggests that the initial values are already pretty good. For the best particle (the one that finds a solution first), the coefficients stabilize very quickly, but the final values of c_1 and c_2 are both smaller than the initial ones. It probably means that for this specific problem, the initial values chosen are slightly high. The initial values are $(0.721348, 0.721348, 0.721348)$, and the means of all final values are $(0.720712, 1.196748, 1.172209)$. Moreover, if we use these new values as initial ones, the success rate improves only slightly, in a way that is not statistically significant.

However, it is also interesting to see what happens if we start from apparently bad parameter values, say $(0.4, 0.4, 0.4)$. If we keep the same small mutation rate, i.e. 0.025, then the performance is very poor, for example, 3% for Tripod (instead of 86%). However, if we use a high mutation rate, say 0.5, the results are significantly improved, though they are not as good as when starting from the initial parameter values chosen before. We get 57% for Tripod (instead of 86%), and 35.8% in average, instead of 61.8%. In this case, the final mean values are $(0.63, 0.38, 0.46)$. So w has increased to a "good" value, but neither c_1 nor c_2 has improved well. It suggests that it may be better to define at least two mutation rates: a small one for w, say ρ_w, and a bigger one for c_1 and c_2, say ρ_c. Table 5 summarises a few experiments. It shows that the global performance can indeed be a bit improved this way, but not enough if the initial values are too bad.

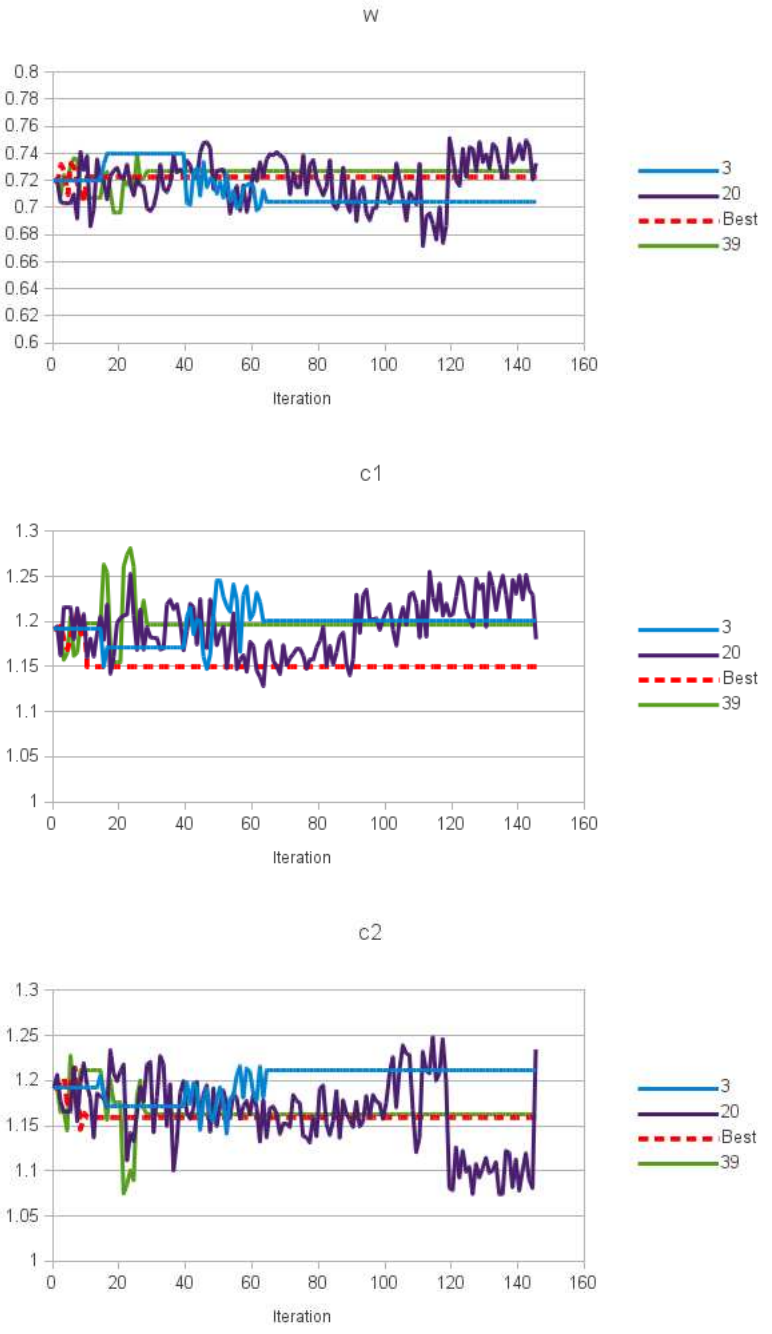

Figure 7.5: Evolution of (w, c_1, c_2). Every colour denotes a particular particle. Tripod function, successful run.

7.4 Spreading the word

When a particle is aware of a good position, how long does it take to inform others? Or, said differently, what is the probability that a given other particle is informed, directly or not, after t time steps? If we consider only the information links that exist at each time step, we can speak of a *potential spreading*. However, not all information is used, only the information given by the best informant is taken into account (in SPSO, it is different; for example in FIPS Mendes et al. (2004)). So, if we consider what really happens, we can speak of the *real spreading*.

7.4.1 Potential spreading

For SPSO, the potential spreading is easy to compute. Formula 1.1 gives the probability of being informed, for a given particle, at a given time step. So the probability of being informed at time t or before is simply

$$p_{spread,SPSO}(t) = 1 - (1 - p_{SPSO})^t \tag{7.1}$$

For the Anybody strategy, it is even easier, for we obviously have

$$p_{spread,Anybody}(t) = 1 \tag{7.2}$$

Unfortunately, for the other cooperation strategies studied here, the potential spreading is problem dependent, so we can just estimate some lower bounds (the upper bound is always 1). For Reciprocity, Vicinity, and Reputation the probability that particle j does inform the particle i is at least h_{min}. Therefore, we have

$$p_{spread}(t) \geq 1 - (1 - h_{min})^t \tag{7.3}$$

For Kin, each particle is informed by the D nearest ones (according to the "genetic" distance), but a given particle may perfectly be never informed by another one. So, the lower bound is simply 0.

7.4.2 Real spreading

Now, we want to know the real probability. It depends on the problem to be solved, so we can approximate it by experiments. The method that has been used here is based on *chains of information*. Such a chain is a temporal sequence of particles $\{i_1, i_2, \ldots, i_T\}$ so that at time t, the particle i_{t+1} has really used the information sent by i_t. In practice, for SPSO or similar variants, it means that at time t there is an information link from i_t to i_{t+1} and that i_t is the best informant of i_{t+1}.

Then, for a given problem, and a given cooperation mechanism, we estimate the real spreading as follows:

- randomly select two particles, i and j;

- on a complete run, save the distribution of the lengths of the chains of information between i and j, i.e. for each length T how many times a chain of this length is generated;
- repeat the above two steps many times, and compute the average distribution.

So, for each problem, and each kind of cooperation, we can plot the cumulative average distribution (see figure7.6). The details of these distributions are not important, and, anyway, difficult to read (particularly if you have a black and white copy of this paper!). However, two points are obvious:

- for the worst kind of cooperation (Anybody), all chains are very short, at most two particles;
- the best mechanism (Kin), tends to produce long chains for some problems, and short chains for some other ones. There may be a relationship with the "difficulty" of these problems, but nothing is very clear. So this point needs further investigation.

Anyway, a good cooperation mechanism will certainly induce a distribution of the chain lengths with neither too many short values nor too many high values. How to define such a good compromise is still an open question. For example, it is probably possible to force the information chain lengths to be between a minimal and a maximal value.

7.5 Help matrix and valued topology

In the PSO context, the topology at time t is the graph of information links. The nodes are the particles, and there is a link from j to i if j is an informant of i. Such a graph can be represented by a binary $S \times S$ matrix. The help matrix is a generalisation of this. Instead of having 0 (j does not inform i) or 1 (j does inform i), we have a probability (j may inform i, with the probability $H(j,i)$).

Conversely, the help matrix can be seen as a representation of a valued graph, which can therefore be named a *valued topology*. So, we can see a clear evolution of the cooperation mechanisms that are used in the context of PSO:

- fully connected graph (often called global best topology). Example: the very first PSO version Kennedy and Eberhart (1995);
- static local best, connected graph. Examples: Ring, Von Neumann, Four Clusters, etc. (see Mendes (2004) for a quite complete list);
- variable graph (informants selected from time to time). Clerc (2005)Janson and Middendorf (2005)Miranda et al. (2008), etc.;
- variable valued graph (informants selected from time to time, and with a probability to use them). Example: the present study.

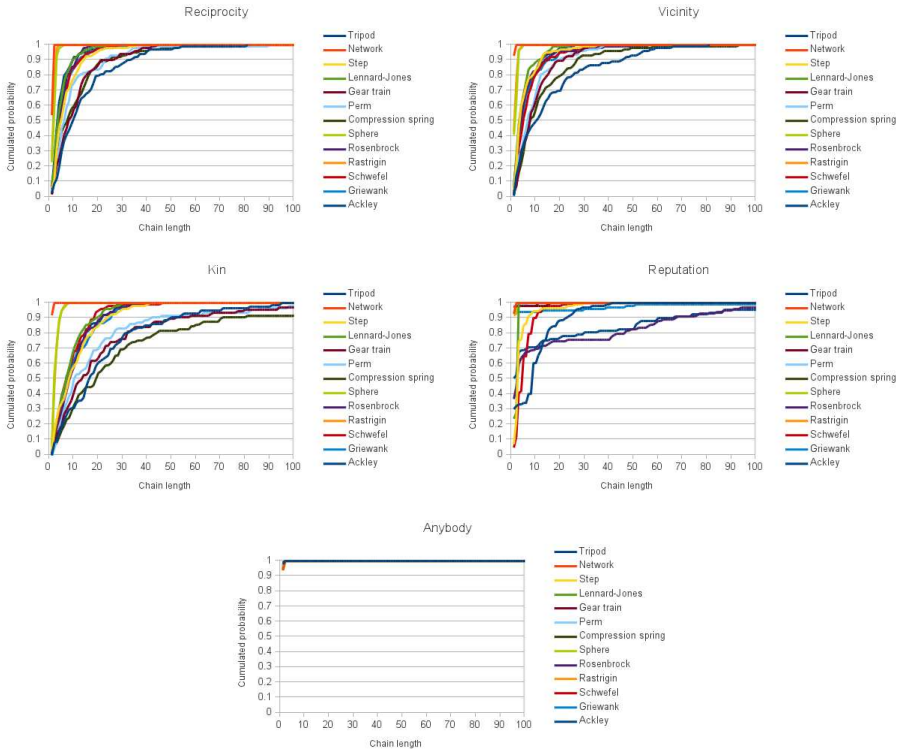

Figure 7.6: Cumulative probability of the length of the chains of information. The worst mechanism (Anybody) induces very short chains. The best one (Kin) tends to induce long chains for some problems. However, there is no clear relationship with the "difficulty" of the problem. Indeed, Reputation also sometimes induces long chains, but is nevertheless not very good on average.

As a next step, an obvious and even more general mechanism could be the same as the last one given above, but with all parameters being adaptive (see in particular 3.2).

Acknowledgements

I am particularly grateful for the assistance given by Abhi Dattasharma, who has carefully read the draft of this chapter, and helped me to write it in a more readable English.

References

CEC (2005). Congress on Evolutionary Computation Benchmarks,http://www3.ntu.edu.sg/home/epnsugan/.

Clerc, M. Math Stuff about PSO, http://clerc.maurice.free.fr/pso/.

Clerc, M. (2005). *L'optimisation par essaims particulaires. Versions paramétriques et adaptatives*. Hermés Science.

Clerc, M. (2006). *Particle Swarm Optimization*. ISTE (International Scientific and Technical Encyclopedia).

Clerc, M. (2012). Randomness matters, http://clerc.maurice.free.fr/pso/randomness Technical report.

Janson, S. and Middendorf, M. (2005). A hierarchical particle swarm optimizer and its adaptive variant. *IEEE Trans Syst Man Cybern B Cybern*, 35(6):1272–1282.

Kennedy, J. and Eberhart, R. C. (1995). Particle Swarm Optimization. In *IEEE International Conference on Neural Networks*, volume IV, pages 1942–1948, Perth, Australia. Piscataway, NJ: IEEE Service Centeer.

Mendes, R. (2004). *Population Topologies and Their Influence in Particle Swarm Performance*. PhD thesis, Universidade do Minho.

Mendes, R., Kennedy, J., and Neves, J. (2004). Fully Informed Particle Swarm: Simpler, Maybe Better. *IEEE Transactions of Evolutionary Computation*, 8:204–210.

Miranda, V., Keko, H., and Duque, A. J. (2008). Stochastic Star Communication Topology in Evolutionary Particle Swarms (EPSO). *International Journal of Computational Intelligent Research*.

Nowak, M. A. (2006). Five rules for the evolution of cooperation. *Science*, 314, n^o 5805:1560–1563.

PSC. Particle Swarm Central, http://www.particleswarm.info.