2006 IEEE Congress on Evolutionary Computation
Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada
July 16-21, 2006

Enhancing the robustness of a speciation-based PSO

Stefan Bird, *Member, IEEE*, and Xiaodong Li, *Member, IEEE*

***Abstract*— Speciation encourages an evolutionary algorithm to locate multiple solutions in multimodal environments. Speciation algorithms often require a user to specify a parameter to define the species radius, which can be a major drawback since this knowledge may not be available a priori. This paper proposes a technique using a time-based convergence measure to overcome this problem. The proposed method is used to enhance the performance of a speciation-based PSO (SPSO) and has been shown to be robust over a wide range of values for this user-specified parameter.**

I. Introduction

Evolutionary Algorithms (EA) have been shown to be very effective in solving difficult optimisation problems. EAs were initially designed to solve unimodal problems, that is problems with only one peak. However, there is a large class of problems that have more than one optimum. These are known as multimodal problems. They may have many local optima, many global optima or both. EAs can have difficulty solving problems with many local optima as they can become trapped on a local peak. For problems with many global optima, the simple EAs will generally only converge on one of the optima. Often it is desirable to locate most or all of the optima so the user can be presented with a choice.

Several techniques exist to provide better performance on multimodal problems. These include derating [1], restricted tournament selection [2], crowding [3], [4], fitness sharing [5] and speciation [6], [7], [8], [9], [10].

Speciation, also known as niching, improves EA performance on multimodal problems by allowing different areas of the problem space to be explored independently. Individuals are grouped into species based on their proximity; the algorithm either discourages or prohibits interactions between different species. Speciation reduces the risk of premature convergence by having several populations converging on different areas simultaneously. Even if one species becomes trapped in a local peak, chances are that another will locate a global optimum. The EA is also able to locate multiple global optima simultaneously as the populations on each optimum are not affected by each other.

One speciation technique, speciation-based PSO (SPSO) [8], has shown to be effective on a variety of multimodal problems. It has a drawback however in that it requires the user to specify the radius r of each species. The algorithm is quite sensitive to the value this parameter is set to [8]. Setting it too small causes many particles to become trapped in local optima. Setting it too large prevents the algorithm from locating nearby optima - peaks within r of an already-found peak will not be seen. This paper presents an enhanced version of SPSO, ESPSO. This new algorithm is able to locate individual optima, even if they are within r of each other.

Section II presents an outline of existing research. Section III describes the enhancements to SPSO, and our testing methods for this new algorithm will be explained in Section IV. The effectiveness of the enhancements will be discussed in Section V, followed by some concluding remarks in Section VI.

II. Prior Work

In this section, we will give an introduction to particle swarms. We will then describe how SPSO enhances particle swarms, improving performance in multimodal environments.

A. Particle Swarms

Particle Swarm Optimisation (PSO) is a stochastic search strategy. It maintains a population of "particles" which travel around the search space. Each particle is able to remember a single location - the best point it has found so far, known as its *personal best*. It is also connected to several neighbours with which it can communicate the fitness and location of its personal best.

To decide where to move next, each particle chooses a point somewhere along the line running between its personal best and the fittest personal best of any of its neighbours, known as the *neighbourhood best*. While the particle will aim for this point, it has momentum, meaning it will generally overshoot the point and have to turn around and return. By continually overshooting, the particle is able to thoroughly explore the area, hopefully finding an even better point. When a particle finds a good location, its neighbours become aware of and are attracted to the point. Once they reach the area and discover fitter points for themselves, they attract their neighbours, who attract their neighbours and so forth. Assuming that the particles have found the best known optimum, eventually the whole population will converge on the peak.

We have used Clerc's constriction coefficient PSO variation in our testing [11], [12] as it guarantees convergence [11]. It is mathematically described thus:

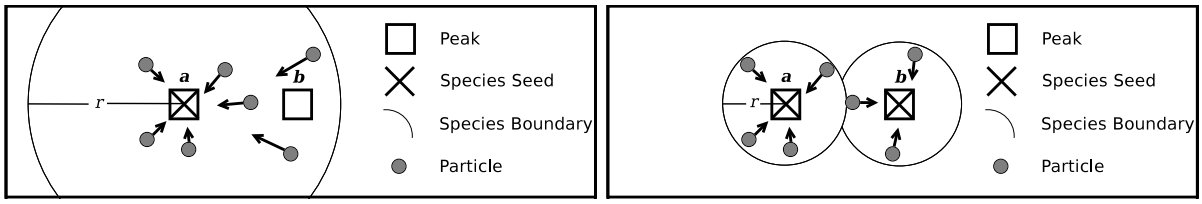
$$\vec{v}_{(i,t+1)} = \chi(\vec{v}_{(i,t)} + \varphi_1(\vec{p}_{(i,t)} - \vec{x}_{(i,t)}) + \varphi_2(\vec{p}_{(g,t)} - \vec{x}_{(i,t)})) \quad (1)$$

$$\vec{x}_{(i,t+1)} = \vec{x}_{(i,t)} + \vec{v}_{(i,t+1)} \quad (2)$$

Stefan Bird is with the School of Computer Science and Information Technology, RMIT University, Melbourne, Victoria, Australia (email: stbird@seatiger.org).

Xiaodong Li is with the School of Computer Science and Information Technology, RMIT University, Melbourne, Victoria, Australia (email: xiaodong@cs.rmit.edu.au).

0-7803-9487-9/06/$20.00/©2006 IEEE



(a) Peak ***a*** has been located, but ***b*** is unlikely to be found because r is set too large; particles approaching peak ***b*** are attracted to ***a***.

(b) The swarm is able to locate both peaks because r has been set to the ideal value for this problem. Determining the correct value for r often requires knowledge of the problem domain.

Fig. 1. The effects of setting r too large in SPSO

where:

$$\varphi_1 = c_1 r_1, \quad \varphi_2 = c_2 r_2,$$
$$\chi = \frac{2\kappa}{\left|2 - c - \sqrt{c^2 - 4c}\right|} \tag{3}$$

Equations (1) and (2) are run at every step t. The velocity of the particle at step t is represented as $\vec{v}_{(i,t)}$, and its current location $\vec{x}_{(i,t)}$. The particle's personal and neighbourhood best locations are denoted by $\vec{p}_{(i,t)}$ and $\vec{p}_{(g,t)}$ respectively. In equation (2) the particle's new velocity is added to its current position to calculate the next location for the particle. The constriction factor χ, calculated in Equation (3), is used to dampen the velocity. This prevents violent oscillations around an optimum, allowing the particles to converge. c_1 and c_2 are constants, typically set to 2.05. $c = c_1 + c_2$. κ is also a constant, usually set at 1. r_1 and r_2 are uniform random numbers between 0 and 1.

B. SPSO

Pétrowski [10] developed a clearing procedure for genetic algorithms. The individuals are sorted in descending order of fitness and placed into a list. For each particle in the list, the algorithm scans all of the particles that come after it. All subsequent particles that are closer than the user-set distance parameter σ of the current particle have their fitness set to 0. By doing this, each peak is represented by a single individual.

Brits et. al. created an algorithm called NichePSO [6]. Unlike a normal PSO, in NichePSO the particles use a cognitive model and do not communicate. The system tracks the fitness of each particle over the last 3 steps. If the variance of the fitness is less than the user-set parameter δ, a subswarm is created between that particle and its nearest topological neighbour. Particles that subsequently move to a point within the species join that species.

In [13] Kennedy presents a speciation technique that uses a clustering algorithm to allocate particles to species. The species centre is used as the particle's personal best. The clustering algorithm used requires the number of clusters to be set a priori. Kennedy doesn't report how different values of this parameter affect performance.

In [8] Li extended Pétrowski's algorithm to develop SPSO. Li's algorithm allocates each particle to a species based on that particle's proximity to a better particle. At each iteration, the particles are sorted by the fitness of their current location, so that the fittest particles are at the front of the list. The species list is then reset so it is empty. The list is traversed and each particle is checked in turn:

- If the particle is not within the radius r of any existing species, create a new species containing only that particle. This particle is set as the *species seed* and the species added to the species list.
- Otherwise the particle is placed in the first species in the species list whose seed is within r of the particle.

The particle's neighbours are all of the particles in its species; there is no interaction between species.

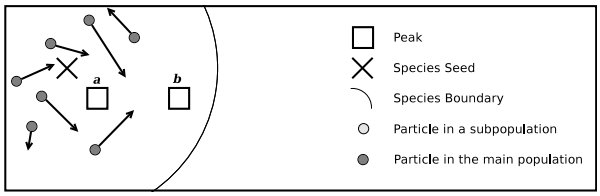
III. Enhancing SPSO

The main disadvantage of SPSO is its dependence on the radius parameter, r. For most problems, the ideal setting is the largest value that does not cause nearby optima to interfere with each other. Any set of optima within $2r$ of each other can potentially interfere if a significant portion of one optimum's catchment area lies within r of another. Interference occurs when particles seeking an unrepresented optimum get "captured" by a species representing a neighbouring optimum. If the optima are within r of each other it becomes impossible for SPSO to differentiate them at all (see Figure 1).

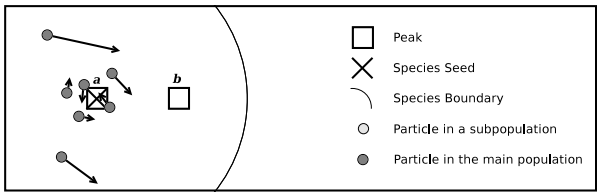
The purpose of the r parameter is to allow particles to interact only with others that are in their local area and not be distracted by a potential solution on the other side of the search space. However once a peak has been located, it is desirable to prevent the species representing it from capturing new particles. This allows nearby optima to be located without interference from existing species.

A. Detecting Convergence

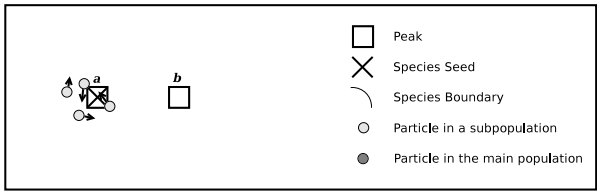
The "enhanced" parts of ESPSO are only triggered once a species has converged. The simplest method of detecting convergence is to require all particles to be within a certain distance of the seed. This distance is generally very small relative to r. Although much easier to set than r, this parameter is still dependant on the scale of the problem space, meaning there is not a single value that will work for all problems. Instead ESPSO uses a time-based measure



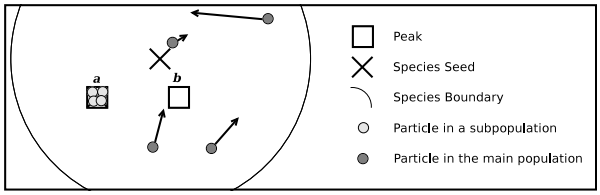
(a) The particles have approximately located peak ***a***. The species seed (see Section II-B) will keep moving as the particles hone in on the optimum.



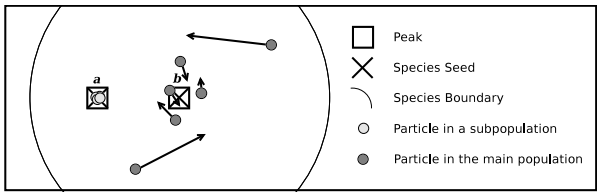
(b) Peak ***a*** has been located and the species seed has not moved for s steps. The best particles in the species will be placed into a sub-population that doesn't interact with the rest of the swarm. The other particles will be reinitialised with random positions and velocities and their personal best memory set to their new location.



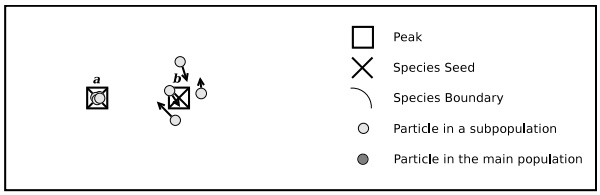
(c) Immediately after the sub-population is created. The particles that were kept still have the same location, velocity and personal best, and will continue to converge over time. As there is no longer a species in the area, other particles are free to come in and explore, eventually locating peak ***b***.



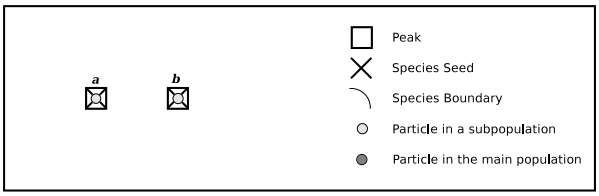
(d) Other particles move in and start locating peak ***b***. As these particles do not interact with the particles on peak ***a*** in any way, there is no interference from the existing optimum.



(e) As happened with peak ***a***, the best particles in the species are about to be placed in a sub-population.



(f) Immediately after the sub-population is created. The particles on peak ***b*** will continue to converge.



(g) The particles on both optima are now converged. Other particles are free to explore the area looking for any other peaks that might exist.

Fig. 2. Time-lapse diagram of ESPSO locating two nearby peaks within radius r.

- a species is said to have converged if the personal best location of its seed has not moved for s steps. This measure is invariant to the scale of the problem space. This process is illustrated in Figure 2 parts (b) and (e).

In SPSO, species groupings are determined by the current location of each particle. ESPSO uses the personal best location instead, as it more accurately reflects the true location of the peak. The personal best is also far more stable; it would be impractical to use the time-based convergence measure with the particle's current location - the seed particle would have to have stopped moving completely before a species is considered to have converged.

TABLE I

EVALUATION FUNCTIONS. THE OPTIMAL VALUE OF r FOR SPSO IS DETERMINED AS SOMEWHERE BETWEEN 50% AND 90% OF THE DISTANCE BETWEEN THE CLOSEST TWO OPTIMA FOR THAT FUNCTION.

Function	**Function Name**	**Range**	**Optimal r (SPSO)**	**Comments**
F1	Brainin RCOS [14]: $F1(x,y) = (y - \frac{5.1x^2}{4\pi^2} + \frac{5x}{\pi} - 6)^2 + 10(1 - \frac{1}{8\pi})cos(x) + 10$	$-5 \leq x \leq 10$; $0 \leq y \leq 15$	4	3 global optima
F2	Six-Hump Camel Back [14]: $F2(x,y) = -4[(4-2.1x^2+\frac{x^4}{3})x^2+xy+(-4+4y^2)y^2]$	$-1.9 \leq x \leq 1.9$; $-1.1 \leq y \leq 1.1$	1	2 global optima and 4 local optima
F3	Deb's 1st Function [15]: $F3(x) = sin^6(5\pi x)$	$0 \leq x \leq 1$	0.15	5 equally spaced global optima
F4	Himmelblau [1]: $F4(x,y) = 200 - (x^2 + y - 11)^2 - (x + y^2 - 7)^2$	$-6 \leq x, y \leq 6$	3	4 global optima
F5	Shubert 2D [7]: $F5(x,y) = \sum_{i=1}^{5} i\cos[(i+1)x + i] \sum_{i=1}^{5} i\cos[(i+1)y + i]$	$-10 \leq x, y \leq 10$	0.75	18 global optima in 9 clusters, many local optima

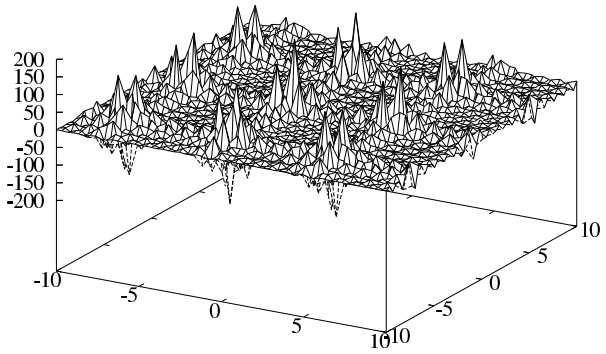


Fig. 3. The Shubert 2D test function has 18 global optima located in 9 pairs, as well as many local optima. SPSO performs poorly on this function due to the proximity of the global optima and multitude of local peaks.

B. Preventing Future Interactions

When a species has converged, a sub-population containing m particles is formed (See Figure 2 parts (c) and (f)). The best m particles from the original species are placed in the sub-population; all other particles in the species are reinitialised with a random location and velocity and their personal best memories set to their new position. This is similar to the P_{max} method presented by Parrott and Li [9], where they placed an upper limit on the number of particles that are allowed to join each species. If there are fewer than m particles in the species, new particles are created near the seed. These particles have a random location and velocity, however the new particles will be closer than the furthest existing particle in the species. Their velocity is also limited so that it is smaller than the distance between the furthest particle and the seed. The species then becomes a separate sub-population[1] and does not interact with any other particles. If the original species had only one particle, the distance and velocity limits are set to r.

[1]Note that a sub-population is different from a species. Particles are free to join and leave species as they move around the decision space. Particles in a sub-population cannot leave and do not interact with particles outside that sub-population.

C. Removing Duplicate Species

A potential problem with ESPSO is duplicate species. It is possible for a species to locate an optimum, be converted to a sub-population, then have another species locate the same optimum. The result is multiple sub-populations located at the same point. To counter this, once a seed has been at the same location for greater than s steps and is within the Euclidean distance in the search space δ of a fitter seed, the species or sub-population is killed. δ is extremely easy for the user to set - it represents the smallest difference between optima that the user cares about. If the population size after removing duplicate species is smaller than the desired population size, new particles with random positions and velocities will be created to meet the shortfall.

IV. PERFORMANCE MEASUREMENT

The primary motivation for this work is to reduce the sensitivity of SPSO to large values of r. The performance of ESPSO will be compared to SPSO on 5 test functions, shown in Table I. These functions were chosen as they represent a variety of different problem types:

- Brainin RCOS (F1) and Himmelblau (F4) both have peaks with large catchment areas. The two right-hand peaks of Himmelblau can cause SPSO problems if r is set too large, even if it is not set large enough to encompass both - the peak that is first located has a tendency to "steal" particles that are exploring the other. This tendency worsens as r is increased.
- Six-Hump Camel Back (F2) has two global optima with relatively large catchment areas, however there are also 4 local optima for the particles to become trapped in.
- Deb's 1st Function (F3) is quite simple to solve even though there is little separation between peaks.
- Shubert 2D (F5) is the most difficult as it is highly multimodal. There are 18 global optima, however their

catchment areas are extremely small. The optima are located in pairs which are evenly distributed throughout the search space. The optima in each pair are very close, meaning that for SPSO to find them all r has to be set very small. However with this setting most of the particles become trapped in the many local optima (see Figure 3).

For F1 through F4, a population size of 50 has been used. F5, being far more multimodal and challenging, uses a population of 500.

ESPSO introduces two new parameters, s and m. For the enhancements to be worthwhile, we need to show that the parameters are robust across a range of problem types. We will analyse the effects of different values for these parameters. For s, we will test values between 1 and 90; m will be tested with values between 2 and 30. We will also look at the effect of the total population size - for F1 through F4 we will test with populations from 10 to 100, F5 will be tested with populations from 100 to 1000. The population tests were conducted with $m = 8$ and $s = 3$. r was set as 2.5 times the approximate ideal radius for SPSO for each problem.

To test the robustness of r, it is set to $\frac{1}{2}$, $\frac{3}{4}$, 1, 2.5, 5, 7.5 and 10 times the approximate ideal radius for SPSO on each function (see Table I). The ideal r value for SPSO is somewhere between 50% and 90% of the distance between the closest two optima in the particular problem. This allows the largest catchment area for each species without causing undue interference between species. The comparisons between SPSO and ESPSO are made with $m = 8$ and $s = 3$, and SPSO has been modified to use the personal best location for species seeds in order to make the results comparable to ESPSO. All tests are repeated 50 times with δ set to 0.1. A run is only considered successful if all of the optima are located to within ϵ = 0.00001; if this isn't achieved within 2000 steps the run is marked as a failure.

V. Results

The results section is divided into several subsections. The first three subsections will analyse the effects of varying s, m and the population size respectively, and the fourth will show the robustness of r. The final subsection will give a comparison between the performance of SPSO and ESPSO.

A. The effect of s

Figure 4 shows the effect of s on the number of evaluations required to find all of the optima. On all of the functions, small values allow ESPSO to locate the optima with fewer evaluations. With an s value of 90, all of the functions took an order of magnitude more evaluations to complete. There is little difference in performance for values between 1 and 5 on any of the test problems; we recommend a value of 3 for all problems.

The success rate of ESPSO is less affected by s (Figure 5). Again small values provide the best success rate on all of the problems tested. On Shubert 2D, the success rate was far lower when s is set larger than 20. Since knowledge of the

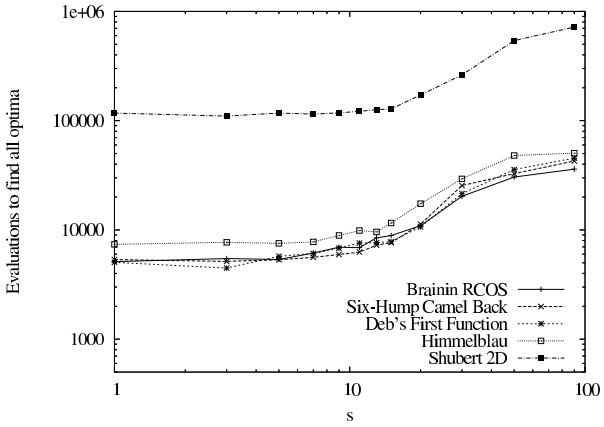


Fig. 4. A log-log graph showing the number of evaluations required for ESPSO to locate all optima with different values of s. r is set to 2.5 times the optimum value for SPSO for each problem, m = 8.

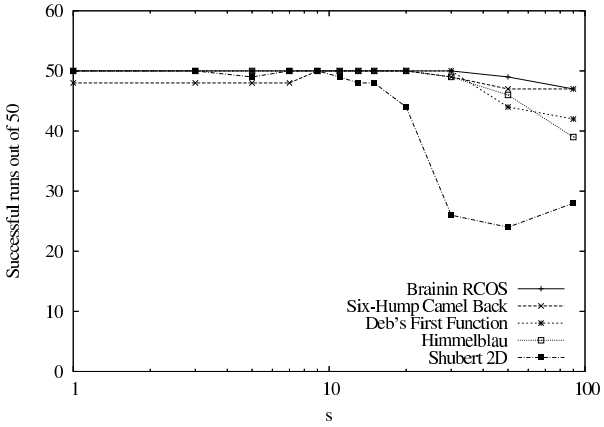


Fig. 5. The number of successful runs with different values of s. r is set to 2.5 times the optimum value for SPSO for each problem, m = 8.

cluster of 2 peaks is lost from the main population when a sub-population is formed, each peak cluster must be found at least twice[2]. As peaks within r of each other cannot be discovered at a rate any greater than once every s steps, large values of s increase the time needed to locate nearby peaks, thus it is probable the algorithm simply ran out of time trying to find all of the peaks.

This parameter s can be seen as an "aggressiveness" control. Large values make ESPSO behave almost identically to SPSO - a species seed has to be still for a very long time before the enhancements are activated. As ESPSO becomes more like SPSO, its efficiency when using large values for r reduces. Eventually, for very large values of s, ESPSO will be unable to locate all peaks because the species are never converted to sub-populations. Similarly, small values of s make ESPSO very aggressive in converting species to sub-populations.

[2]We say at least because it is possible for the same peak to be discovered multiple times.

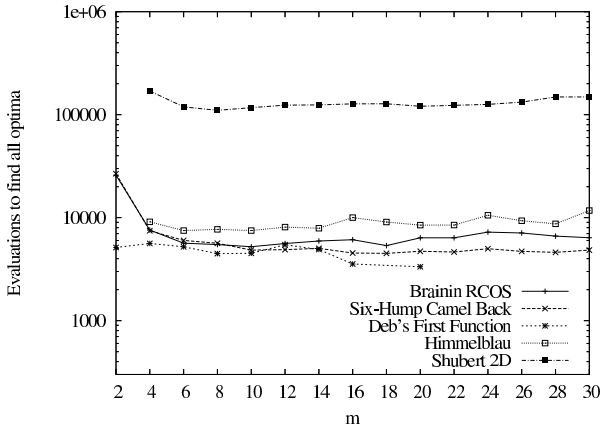


Fig. 6. The number of evaluations required for ESPSO to locate all optima with different values of m. r is set to 2.5 times the optimum value for SPSO for each problem, s = 3.

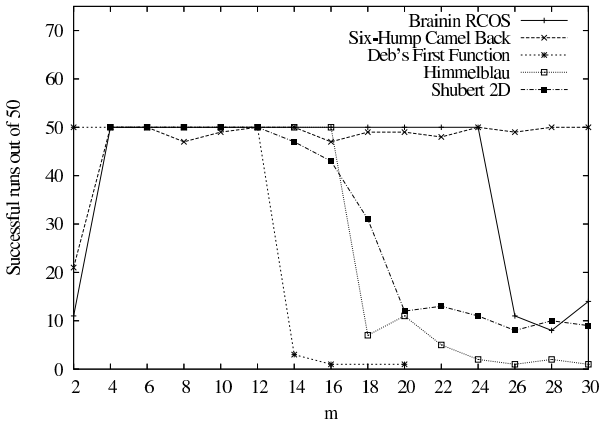


Fig. 7. The number of successful runs with different values of m. r is set to 2.5 times the optimum value for SPSO for each problem, s = 3.

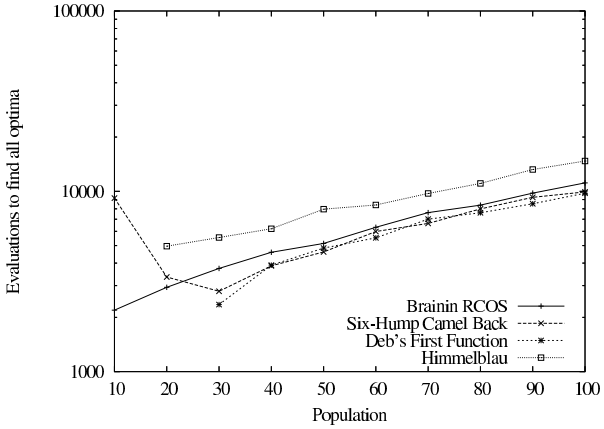


Fig. 8. The number of evaluations required for ESPSO to locate all optima with different population sizes. r is set to 2.5 times the optimum value for SPSO for each problem, $m = 8$, s = 3.

B. The effect of m

From Figure 6 it can be seen that m does not have a large effect on the number of evaluations required to locate all of the optima. Values smaller than 6 cause an increase in the time taken, this is because the sub-populations formed do not have enough search power to quickly locate a peak. As the ideal value of s is very small, the sub-populations are created when the particles are still largely in exploratory mode - they have not converged very far. A larger number of particles is likely to locate the peak more quickly, allowing them to rapidly hone in.

Figure 7 shows that m has a much larger effect on the success rate of the algorithm. The rapid drops in success rate are caused by ESPSO running out of particles. Because the population size for F1 to F4 is 50, large sub-population sizes can quickly exhaust the population limit. If these sub-populations have converged on a local peak, there may not be enough particles left in the main population to locate the global peaks. A more gradual curve can be seen on Shubert 2D. Because Shubert is extremely multimodal and r is still relatively small (1.875) compared to the search space, many sub-populations are formed on local optima. As the number of particles in the sub-populations increases it takes fewer sub-populations on local optima to use up all the particles. We recommend setting m to 8. This provides sufficient search power, but does not force the user to use large population sizes to locate all optima.

Six-Hump Camel Back proved slightly unreliable with most values of m, although the minimum success rate for $4 \leq m \leq 24$ was still 94%. For $m \geq 10$ this can be explained as sub-populations being formed on too many local optima, however we do not have an explanation for values of m less than this. Future research could focus on why ESPSO has difficulty on this particular test function.

C. The effect of population size

The population size has an effect on the evaluations required to solve the problems - Figure 8 shows that larger populations use more evaluations; adding unneeded particles hinders optimisation performance. For every problem except Brainin RCOS and Shubert 2D, the best performance was achieved with a population size of 30. Shubert 2D achieved its best performance with a population size of 300, although the 100% reliability is only achieved with at least 400 particles. Results for Shubert 2D are shown in Table III.

Population size does not have a large effect on the reliability (Figure 9), provided there are enough individuals to cover all of the global optima. Since $m = 8$, this means the system required approximately 8 particles for each optimum, although the system achieved 100% reliability using slightly less than this on most of the fitness functions. The reason for this is that there are usually a few particles left in the main population after most of the optima are discovered; these were able to locate the last optimum.

As the optimum number of particles differs for each test function we cannot offer a population size that would work for all, however it may be possible to make this property adaptive by adding new individuals when the number remaining in the main population becomes too low. This

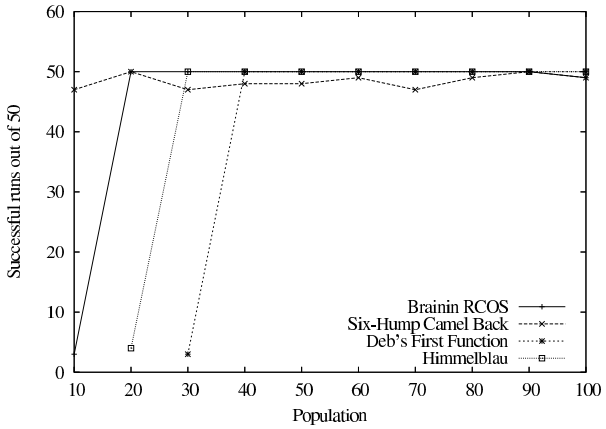


Fig. 9. The number of successful runs with different population sizes. r is set to 2.5 times the optimum value for SPSO for each problem, $m = 8$, $s = 3$.

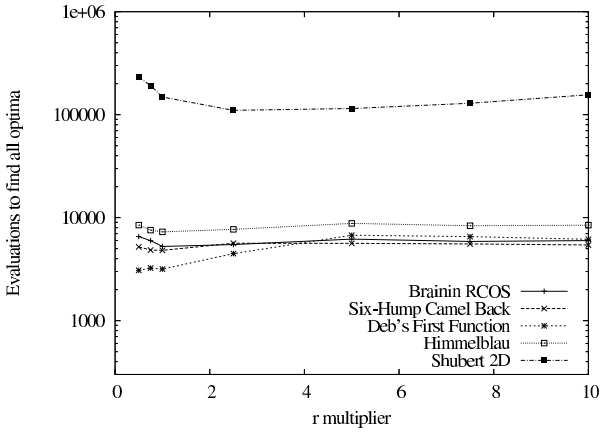


Fig. 10. The number of evaluations required for ESPSO to locate all optima with different values of r. To aid comparison between test functions, r has been shown as a multiple of SPSO's optimal r for each test problem.

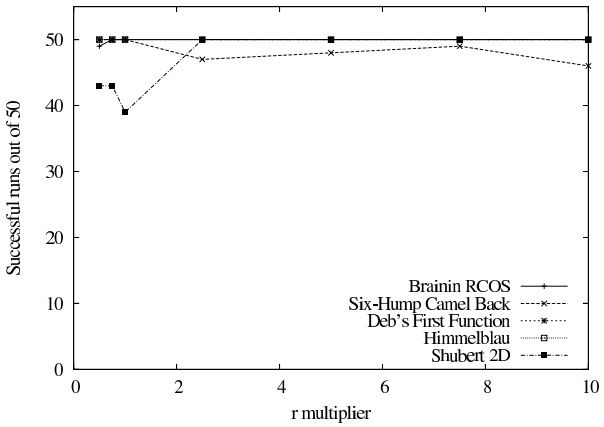


Fig. 11. The number of successful runs with different values of r.

condition indicates there are too many optima for the number of particles and that it may be possible to locate more optima. To prevent a population explosion it would probably be necessary to "reclaim" particles from under-performing subpopulations before creating new individuals. This could be a focus of future research.

D. The effect of r

Figures 10 and 11 show the value of r has very little effect on performance. While there is benefit to setting it close to SPSO's optimal value, it is certainly not a requirement. In the graphs, an r multiplier of 10 corresponds to a radius larger than the search space on all of the problems except Shubert 2D. This shows it is possible to run ESPSO without using the r parameter at all; in which case it can be thought of as acting similarly to sequential speciation algorithm (although it is still inherently parallel). As r is still a problem-dependant parameter (even if only limited in effect), we are unable to recommend a value that will be optimal on a majority of test functions. If the distance between optima is unknown though, we recommend setting it to a large value as being the safest option. Larger values of r encourage more cooperation between particles, allowing individuals to explore the most promising areas.

From Figure 11 it appears that small values of r are somewhat unreliable on Shubert 2D. This is because there were several runs where ESPSO was unable to locate all of the optima. Even on its worst run it still located 16 of the 18 optima, however for the purposes of the graph, this was still a failure. Figure 13 gives a better representation of performance on this function when finding every peak is not a requirement.

E. Comparison to SPSO

Figures 12 and 13 compares the average number of optima located by SPSO and ESPSO for different values of r on different test problems. As can be seen, SPSO has a definite sweet spot where it is able to find most of the optima, most of the time. ESPSO had no difficulty locating all of the optima when using values of r larger than SPSO's sweet spot.

Table II compares the relative performance of ESPSO and SPSO when r is set to the optimal value for SPSO. It can be seen that on most functions, SPSO required approximately 10-20% fewer evaluations to locate all of the optima. On Shubert 2D, SPSO required a quarter of the evaluations that ESPSO did, however SPSO only located every peak 30% of the time. ESPSO appears to be ineffecient at this r value; increasing it allowed it to find all of the peaks with a similar number of evaluations to SPSO (see Figure 10). SPSO was able to find all optima on Six-Hump Camel Back and Himmelblau 96% of the time and it was successful every time on Deb's First Function and Brainin RCOS. ESPSO was successful on every run for all of the functions.

VI. Conclusion

ESPSO enhances SPSO by greatly increasing the robustness of the r parameter - to the point that the algorithm is still effective even if it isn't used at all. It introduces three new parameters, s, m and δ. The first two have been shown to be robust across the test functions we used. The last is problem

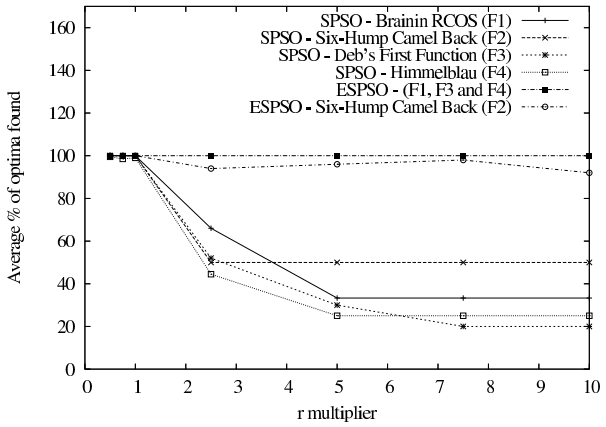


Fig. 12. Comparing the number of optima found by SPSO and ESPSO on test functions F1 to F4 for different values of r. For ESPSO, s = 3 and m = 8. SPSO fails to locate all of the optima as soon as r increases much beyond the optimal value, whereas ESPSO is still effective even with very large r values.

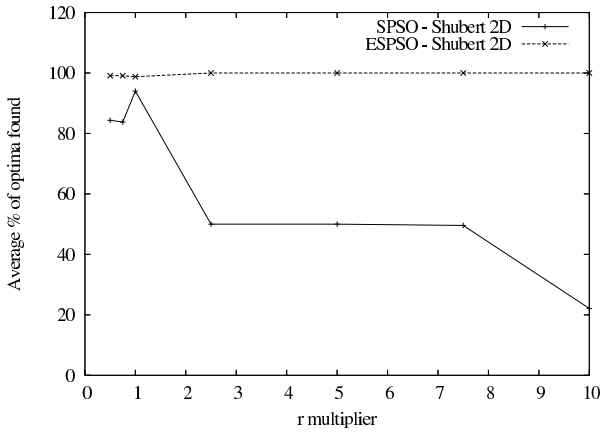


Fig. 13. Comparing the number of optima found by SPSO and ESPSO on the Shubert 2D function for different values of r (s = 3, m = 8).

dependant, however it is intuitive and easy to set - it can be presented to the user as "Do not show solutions closer than δ". Further research could investigate whether ESPSO is effective on a wider variety of problems, including ones with higher dimensionality.

TABLE II

NUMBER OF EVALUATIONS REQUIRED TO FIND ALL GLOBAL PEAKS (MEAN AND STANDARD DEVIATION).

Function	Pop.	ESPSO	SPSO
F1	50	5250 ($\pm$1184)	4387 ($\pm$943)
F2	50	4863 ($\pm$1151)	3716 ($\pm$951)
F3	50	3167 ($\pm$904)	2886 ($\pm$737)
F4	50	7279 ($\pm$1260)	6080 ($\pm$864)
F5	500	415256 ($\pm$509870)	103533 ($\pm$18956)

TABLE III

ESPSO PERFORMANCE ON SHUBERT 2D WITH DIFFERENT POPULATION SIZES.

Population	Number of evaluations	Reliability
100	43255 ($\pm$0)	2%
200	57130 ($\pm$7918)	14%
300	81612 ($\pm$20254)	62%
400	99045 ($\pm$20669)	100%
500	117100 ($\pm$16643)	100%
600	133147 ($\pm$18873)	100%
700	162782 ($\pm$23952)	100%
800	173864 ($\pm$28417)	100%
900	193309 ($\pm$19985)	100%
1000	218354 ($\pm$36092)	100%

REFERENCES

[1] D. Beasley, D. Bull, and R. Martin, "A sequential niche technique for multimodal function optimization," *Evolutionary Computation*, vol. 1, no. 2, pp. 101–125, 1993. [Online]. Available: citeseer.ist.psu.edu/beasley93sequential.html

[2] G. Harik, "Finding multimodal solutions using restricted tournament selection," in *Proceedings of the Sixth International Conference on Genetic Algorithms*. Morgan Kaufmann, 1995, pp. 24–31. [Online]. Available: citeseer.ist.psu.edu/harik95finding.html

[3] K. D. Jong, "An analysis of the behavior of a class of genetic adaptive systems." Ph.D. dissertation, University of Michigan, 1975.

[4] S. W. Mahfoud, "Crowding and preselection revisited," in *Parallel problem solving from nature 2*. Amsterdam: North-Holland, 1992, pp. 27–36. [Online]. Available: citeseer.ist.psu.edu/mahfoud92crowding.html

[5] D. Goldberg and J. Richardson, "Genetic algorithms with sharing for multimodal function optimization," in *Proceedings of the Second International Conference on Genetic Algorithms*, J. Grefenstette, Ed., 1987, pp. 41–49.

[6] R. Brits, A. Engelbrecht, and F. van den Bergh, "A Niching Particle Swarm Optimizer," in *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAL'02)*, vol. 2, 2002, pp. 692–696.

[7] J.Li, M. Balazs, G. Parks, and P. Clarkson, "A species conserving genetic algorithm for multimodal function optimization," *Evolutionary Computation*, vol. 10, no. 3, pp. 207–234, 2002.

[8] X. Li, "Adaptively choosing neighbourhood bests using species in a particle swarm optimizer for multimodal function optimization," in *Proceedings of Genetic and Evolutionary Computation Conference 2004 (GECCO'04) (LNCS 3102)*, 2004, pp. 105–116.

[9] D. Parrott and X. Li, "A particle swarm model for tracking multiple peaks in a dynamic environment using speciation," in *Congress on Evolutionary Computation (CEC2004)*, vol. 1, 2004, pp. 98–103.

[10] A. Pétrowski, "A clearing procedure as a niching method for genetic algorithms," in *Proceedings of the 3rd IEEE International Conference on Evolutionary Computation*, 1996, pp. 798–803.

[11] M. Clerc and J. Kennedy, "The particle swarm - explosion, stability, and convergence in a multidimensional complex space," in *IEEE Transactions on Evolutionary Computation*, vol. 2, 2002, pp. 58–73.

[12] J. Kennedy and R. Eberhart, *Swarm intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001.

[13] J. Kennedy, "Stereotyping: improving particle swarm performance with cluster analysis," in *Congress on Evolutionary Computation (CEC2000)*, vol. 2, 2000, pp. 1507–1512.

[14] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. New York: Springer-Verlag, New York, 1996.

[15] K. Deb and D. Goldberg, "An investigation of niche and species formation in genetic function optimization," in *Proceedings of the Third International Conference on Genetic Algorithms*, J. Schaffer, Ed., 1989, pp. 42–50.