Multi-swarm Optimization in Dynamic Environments

Tim Blackwell[1] and Jürgen Branke[2]

[1] Department of Computing, Goldsmiths College, University of London
New Cross, London SE14 6NW, U.K.
t.blackwell@gold.ac.uk

[2] Institute AIFB, University of Karlsruhe,
D-76128 Karlsruhe, Germany
branke@aifb.uni-karlsruhe.de

Abstract. Many real-world problems are dynamic, requiring an optimization algorithm which is able to continuously track a changing optimum over time. In this paper, we present new variants of Particle Swarm Optimization (PSO) specifically designed to work well in dynamic environments. The main idea is to extend the single population PSO and Charged Particle Swarm Optimization (CPSO) methods by constructing interacting multi-swarms. In addition, a new algorithmic variant, which broadens the implicit atomic analogy of CPSO to a quantum model, is introduced. The multi-swarm algorithms are tested on a multi-modal dynamic function – the moving peaks benchmark – and results are compared to the single population approach of PSO and CPSO, and to results obtained by a state-of-the-art evolutionary algorithm, namely self-organizing scouts (SOS). We show that our multi-swarm optimizer significantly outperforms single population PSO on this problem, and that multi-quantum swarms are superior to multi-charged swarms and SOS.

1 Introduction

Particle Swarm Optimization (PSO) is now established as an efficient optimization algorithm for static functions in a variety of contexts [18]. PSO is a population based technique, similar in some respects to evolutionary algorithms, except that potential solutions (particles) move, rather than evolve, through the search space. The rules, or particle dynamics, which govern this movement, are inspired by models of swarming and flocking [17]. Each particle has a position and a velocity, and experiences linear spring-like attractions towards two attractors:

1. The best position attained by that particle so far (particle attractor), and
2. The best of the particle attractors (swarm attractor),

where best is in relation to evaluation of an objective function at that position. The swarm attractor therefore enables information sharing between particles, whilst the particle attractors serve as individual particle memories.

The optimization process is iterative. In each iteration, the acceleration vectors of all the particles are updated based on the position of the corresponding attractors.

G.R. Raidl et al. (Eds.): EvoWorkshops 2004, LNCS 3005, pp. 489–500, 2004.

Then, this acceleration is added the velocity vector, after which the velocity is 'constricted' so that the particles progressively slow down, and this new velocity is used to move the individual from the current to the new position. The constriction factor acts like friction, slowing the particles, so that finer exploration is achieved.

While most of the optimization problems discussed in the scientific literature are static, many real-world problems change over time, i.e. they are dynamic. In those cases, the optimization algorithm has to track a moving optimum as closely as possible, rather than just finding a single good solution. It has been argued [9] that evolutionary algorithms (EAs) may be a particularly suitable candidate for that type of problems. Recently, the application of PSO to dynamic problems has been explored [18, 15, 12, 1].

Although similar to EAs, PSO needs to be adapted for optimal results on dynamic optimization problems. This is due to diversity loss and linear collapse. If the swarm is converging, the attractors will be close to the optimum position and the swarm will be shrinking at a rate determined by the constriction factor and by the local environment at the optimum. For functions with spherical symmetric local neighborhoods, a theoretical analysis and an experimental verification suggest that the shrinkage (and hence diversity loss) is scale invariant [3, 4, 5]. If the optimum shifts within the collapsing swarm, then re-optimization will be efficient. However, if the optimum shift is significantly far from the swarm, the low velocities of the particles will inhibit tracking, and the swarm can even oscillate about a false attractor and along a line perpendicular to the true optimum (linear collapse) [2].

Various adaptations to PSO have been suggested. Hu and Eberhart [15] list a number of these, which all involve randomization of the entire, or part of, the swarm. This is either in response to function change, or at some pre-determined interval. Function change can be detected by a re-evaluation of the objective function at one or several of the attractors. These techniques are somewhat arbitrary, and suffer from the fact that randomization implies loss of information gathered during the search so far. As an alternative adaptation, Blackwell and Bentley [2] introduced charged swarms with the aim to maintain diversity throughout the run. In charged PSO (CPSO), mutually repelling particles orbit a nucleus of neutral particles. This nucleus is, in fact, a conventional PSO swarm. The picture is reminiscent of classical pictures of the atom [2], although the orbits are chaotic rather than elliptical. The idea is that the charged subswarm maintains population diversity, at least within the spatial extent of the charged orbits, so that function change can be quickly (and automatically) registered, and the swarm can adapt. Meanwhile the neutral swarm can continue to explore the neighborhood of the optimum in increasing detail. CPSO has been applied to a number of uni- and bi-modal test functions of high change frequency and spatial severity, and has been shown to work well, outperforming conventional PSO [1].

In the area of EAs, multi-population approaches like the self-organizing scouts developed by Branke [9] have shown to give excellent results. The goal there is to have a number of sub-populations watching over the best local optima. For that purpose, a part of the population is split off when a local optimum is discovered, and remains close to this optimum for further exploration. The remainder of the population continues to search for new local optima, and the process is repeated if any more local optima are found. This technique is expected to work well for a class of dynamic

functions consisting of several peaks, where the dynamism is expressed by small changes to the peaks' locations, heights and widths. These have been argued to be more representative of real world problems [8]. To track the optimum in such an environment, the algorithm has to be able to follow a moving peak, and to jump to another peak when the peak heights change in a way that makes a previously non-optimal peak the highest peak.

Inspired by self-organizing scouts, we investigate here whether a multi-population version of CPSO (multi-CPSO) might also be beneficial in dynamic environments. Again, the underlying idea is to place a CPSO swarm on each local optimum in a multi-modal environment. Whilst the neutral sub-swarms continue to optimize, the surrounding charged particles maintain enough diversity to track dynamic changes in location of the covered peaks.

There has been some work on parallel niching PSO sub-swarms for static problems [11]. The approach of the authors in [11] is to create a 2 particle sub-swarm from a particle and its nearest spatial neighbor, if the variance in that particle's fitness is less than a threshold. Sub-swarms may merge, and they also absorb particles from the main swarm. The nichePSO was able to optimize successfully some static multi-modal benchmark functions, but is not adaptable to the dynamic problem in an obvious way. This is because the algorithm, as the authors point out, depends on a very uniform distribution of particles in the search space, and on a training phase.

The multi-swarm approach developed here has already been proposed in a non-optimization context [6]. Two multi-swarm models are proposed here. One of these, multi-CPSO, is a multi-population version of CPSO. The other, multi-Quantum Swarm Optimization (multi-QSO) uses quantum swarms, which are based on a quantum rather than classical picture of an atom. This is proposed here in order to cure some deficiencies in CPSO (see below) and for comparative purposes.

This paper continues with an explanation of multi-swarm algorithms. Then, an experiment to test multi-swarms on the moving peaks benchmark is described and results are presented. The results are discussed and conclusions are drawn in the final section.

2 Swarm Algorithms

2.1 General Considerations

Multi-PSO swarms do not immediately generalize, since the swarms do not interact (i.e. the dynamics governing the position and velocity updates of a particle in a particular swarm are specified by parameters belonging to that swarm only). Multi-PSO's may interact if swarms have access to attractors from other swarms; however, this reduces to a single swarm with different information sharing topologies [16].

A multi-swarm is, however, easily constructed as combination of CPSO swarms. Multiple CPSO swarms will interact since charged particles repel charged particles from any swarm, including their own. However a few difficulties are apparent and these will now be discussed.

It is hoped that the repulsions between the charged swarms will enable explorations of different regions of the search space by different swarms. However, since the neutral particles do not repel, a situation may arise where the attractors from a number of swarms are on a single peak. This might be remedied by implementing repulsions between neutral particles from different swarms, but this would not prevent an equilibrium arising where a number of swarms surround a single peak and the attractions to the swarm attractors are in balance with the inter-swarm repulsions. In such a case, no single swarm would be able to move closer to the peak and optimization would cease.

In order to prevent this, we use a simple competition between swarms that are close to each other. The winner is the swarm with the best function value at its swarm attractor. The loser is expelled, and the winner remains. Swarms can be considered to be close to each other when their swarm attractors are within an 'exclusion radius' r_{excl}. Although a force law could be used for the expulsion suffered by the loser, a simple randomization of the swarm in the search space is considered in this paper.

A further difficulty is that CPSO is difficult to control, in the sense that the spatial extent of the charged swarm is unpredictable due to the chaotic nature of the orbits [5]. This might have drawbacks in multi-CPSO where it is desired to find swarms on and around localized optima. Another problem is that CPSO suffers from $O(N^2)$ complexity, arising from the Coulomb repulsion. It is probable that the success of CPSO over PSO in the dynamic context is due to the increased diversity around the contracting PSO swarm, rather than due to charged particles finding better solutions within the contracting nucleus. If this is true, then the exact dynamics are not important as long as diversity is maintained. In which case, simple randomization of the charged particles in a region surrounding the neutral swarm may be sufficient, negating the need for expensive $O(N^2)$ computations. This is the basis of the quantum swarm.

The quantum swarm, which builds on the atomic picture of CPSO, uses a quantum analogy for the dynamics of the charged particles. In the quantum model of the atom, the orbiting electrons are replaced by a 'quantum cloud'. This cloud is actually a probability distribution governing where the electron will be found upon measurement. The electron does not follow a classical trajectory in between 'measurements'. The probability distribution depends on the energy (and various other quantum numbers) of the electron. For example, electron positions in the lowest energy state of the hydrogen atom are distributed according to $p(r)\,dr = (\text{const})r^2e^{-2r}$ [14].

Quantum Swarm Optimization (QSO) is therefore a hybrid technique, since the contracting nucleus of the PSO sub-swarm follows deterministic dynamics whereas the enveloping charged cloud follows stochastic dynamics. By stretching the quantum analogy still further, a 'measurement' corresponds to a function evaluation. At this point the charged particles are randomized within a ball of radius r_{cloud} centered on the swarm attractor. This provides a very simple update rule and corresponds to a uniform (and very unphysical) probability distribution. The velocity of the charged particle is irrelevant (it is indeterminate in the quantum picture) and the charged particles are not repelled from other charged particles or attracted to any attractor. However, any good location that they do find by virtue of their random positioning around the swarm attractor may still be useful to the neutral swarm due to information sharing.

The multi-QSO is therefore an assembly of QSO swarms. The only interaction between these swarms occurs when swarms collide and the exclusion principle is applied.

2.2 Proposed Algorithm

After initialization, the proposed multi-swarm algorithm iterates a main loop with four stages: Test for function change, particle update, attractor update and exclusion. These stages are described below.

INITIALIZE. Randomize positions and velocities of each particle in search space, set all attractors to randomized particle positions, set swarm attractor to particle attractor 1 and set all stored function values to function floor.
REPEAT.
  FOR EACH swarm n
    // Test for Change.
    Evaluate function at swarm attractor of swarm n.
    IF new value is different from last iteration **THEN**
      Re-evaluate function values at each particle attractor.
      Update swarm attractor.
      Store function values.
    FOR EACH particle k of swarm n
      //Update Particle
      Apply equations (1) – (7) to particle k of swarm n.
      //Update Attractor.
      Evaluate function at updated position and store value.
      IF new value better than particle attractor value **THEN**
        Particle attractor k := position and value of particle k.
      IF new value better than swarm attractor value **THEN**
        Swarm attractor := position and value of particle k.
    FOR EACH swarm $m \neq n$
      //Exclusion.
      IF swarm attractor p_n is within r_{excl} of p_m **THEN**
        Randomize the swarm with the worse swarm attractor value.
        (re-set particle attractors, evaluate f at each new position, store these values, and set attractor of swarm to the position of its best particle).
UNTIL number of function evaluations performed > max

For updating particle k in swarm n, the following equations are used:

1. For neutral particles:

$$v_{nk} = \chi[\, v_{nk} + c_1\, \varepsilon\,(p_n - x_{nk}) + c_2 \varepsilon\,(p_{nk} - x_{nk})] \quad (1)$$

$$x_{nk} = x_{nk} + v_{nk} \quad (2)$$

2. For charged particles in QSO:

$$x_{nk} \in B_n(r) \quad (3)$$

3. For charged particles in CPSO:

$$v_{nk} = \chi[\, v_{nk} + c_1\, \varepsilon\,(p_n - x_{nk}) + c_2 \varepsilon\,(p_{nk} - x_{nk})] + a^{+}_{nk} \quad (4)$$

$$x_{nk} = x_{nk} + v_{nk} \quad (5)$$

$$a^{+}_{nk} = \sum_{m=1}^{M} \sum_{l=1}^{N_m^+} a_{nk,ml}(1 - \delta_{nk,ml}) \qquad \delta_{nk,ml} = \begin{cases} 1 & nk = ml \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

where

$$a_{nk,ml} = \begin{cases} \dfrac{Q_{ml} Q_{nk}}{\left| x_{core} \right|^2} \dfrac{x_{nk} - x_{core}}{\left| x_{nk} - x_{core} \right|} & \left| x_{nk} - x_{ml} \right| < r_{core} \\ 0 & \left| x_{nk} - x_{ml} \right| > r_{\lim it} \\ \displaystyle\sum_{\mathrm{m}=1}^{\mathrm{M}} \sum_{l=1}^{N_m^+} \frac{Q_{ml} Q_{nk}}{\left| x_{nk} - x_{ml} \right|^3} \left(x_{nk} - x_{ml} \right) & \text{otherwise} \end{cases} \quad (7)$$

In these equations, x_{nk}, $v_{nk,}$ p_{nk} are d-dimensional position, velocity and attractor vectors of particle k from swarm n. The PSO parameters are the spring constants $c_{1,2}$, and the constriction factor $\chi<1$. ε is a random number drawn from a uniform distribution.

The stored function values f_{nk} arise from evaluations $f(p_{nk})$ at time t and, with g(n) as the index of the best attractor (the swarm attractor), $f_{ng(n)} = \max\{ f_{nk}\}$. The test for change therefore compares $f_{ng(n)}$ with $f(p_{ng(n)})$.

A convenient representation of the multi-swarm *configuration* is $M(N_n + N_n^+)$ where N_n and N_n^+ are the numbers of neutral and charged particles in swarm n and M as the number of swarms in the multi-swarm. $M(N_n + N_n^+)$ also evaluates to the total number of particles. From an information-theoretic viewpoint, swarm n is actually the data set $S_n = \{x_{nk}, v_{nk,} p_{nk}, f_{nk}, g(n)\}$, $k = 1,\ldots, N_n + N_n^+$.

The multi-swarm is therefore a colony of M interacting swarms. Neutral particles in any swarm always follow a pure PSO position and velocity update rule. Charged classical particles obey the neutral dynamics, but are also mutually repelled from other charged particles in any swarm. A charged quantum particle, on the other hand, does not follow a classical rule and, upon measurement (a function evaluation), is to be found in a d-dimensional ball $B_n(r_{cloud})$ of radius r_{cloud} centered on $p_{ng(n)}$.

3 Experiments

The experiment was designed to investigate the effect of different multi-swarm configurations on the optimization of a dynamic benchmark function. In order to compare the multi-swarms with the results of evolutionary techniques reported in [8], population size N was fixed at 100 particles. Many different configurations of 100

particles are possible. The number of swarms, M, can range from 1 (where the multi-swarms reduce to PSO, CPSO and QSO) to 100 (where the concept of a swarm is lost, and the model becomes a 'gas' of interacting particles). However it is expected that optimal configurations will lie in between these extremes. As far as possible, symmetrical configurations of 2 to 50 swarms were tested, along with the extremes. The effects of different near-symmetrical configurations was also tested for M = 6 and 7 multi-swarms.

Table 1. PSO and CPSO Parameters.

$c_{1,2}$	κ	r_{core}	r_{limit}	r_{cloud}	Q_{nk}	r_{excl}
2.05	0.729843788	1.0	10.0	10.0	1.0	10.0

The parameter settings for all algorithms were fixed at standard values (cf. [13] for PSO and [2, 1] for CPSO) and are listed in Table 1. The standard PSO parameters $c_{1,2}$ and κ have been well tested by many authors and have been shown to lead to convergence for non-interacting particles, and for interacting particles close to symmetric optima [3, 4]. The single QSO parameter, the exclusion radius r_{cloud}, was set to the perception limit of the equivalent CPSO. This choice was made for comparative purposes: the intention is that charged particles can move freely in a hypersphere of radius $r_{cloud} = r_{limit}$ (although there are large fluctuations in CPSO). The final parameter in both algorithms is the exclusion radius r_{excl}. It would be important to study the dependence of the algorithms on this free parameter, but in this study r_{excl} is fixed at a value chosen to be commensurate with the peak width (between 1 and 12) and the charged swarm size (10). It can be conjectured that a much larger values of r_{excl} would lead to one swarm covering many peaks, and a much smaller value would lead to many swarms trying to optimize the same peak.

For performance evaluation, we used the publicly available moving peaks benchmark [10] which consists of a number of peaks, moving around in the search space, while also changing heights and width. The benchmarks parameter settings correspond to the Scenario 2 as specified on the benchmark web site [10]. The search space has five dimensions $[0, 100]^5$ with 10 peaks. Peak heights can vary randomly in the interval [30, 70], width can vary within [1, 12]. Scenario 2 specifies a family of benchmark function since the initial location, height and width of the peaks, and their subsequent development is determined by a pseudorandom number generator. Furthermore, some of the parameters of Scenario 2 are only defined to within a range of values. Of these, the following choices were made to facilitate comparisons with the experiments of reference [7]: Change severity vlength = 1.0, correlation lambda = 0.0, and peak change frequency = 5000.

Each experiment consists of a particular function instance and initial particle distribution. The termination condition for each experiment is 500000 function evaluations. The experimental conditions are altered by changing the random seed of the benchmark generator and the initial positions and velocities of the particles. The primary performance measure is the offline error [7] which is recorded for each experiment and then averaged, for each algorithm, over 50 experiments with different ran-

dom seeds (which also means different instances of the class of benchmark problems). This measure is always greater or equal to zero and would be zero for perfect optimization. The results for the offline error are tabulated in Table 2, together with the standard error. The configurations marked by a star in this table have 98 rather than 100 particles. For easier comparison, the data is also visualized in Fig. 1.

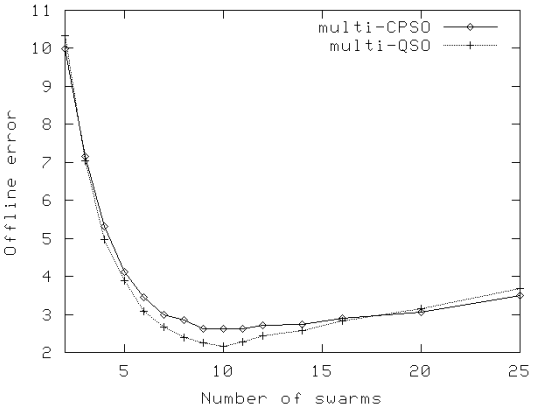

Fig. 1. Performance of multi-CPSO and multi-QSO depending on the number of swarms.

4 Discussion

In the limiting case of a single swarm, configurations 1(100+0) and 1(50+50), the multi-swarm models reduce to the three single particle swarm algorithms, PSO, CPSO and QSO. The PSO is marginally superior, with an offline error of 16.5076. The slightly better performance of the swarm with the smallest diversity is presumably due to its better optimization on a single peak. Clearly, the dynamics of the peak function is too severe for the two simple diversification measures (charged and quantum sub-swarms) to be effective. The very similar offline errors of the single swarm algorithms is quite curious, and to gain insight into this phenomenon, animations of the particle motions were viewed in two dimensional slices through the 5 dimensional space. These animations revealed that in each case the single swarm becomes attracted to, and remains on, a single peak, irrespective of whether this is the maximum peak. Interestingly, the offline error for the single swarms is comparable to the results for a standard EA of the same population size, and for the same experimental conditions, cf. Fig. 4 of [7]. The standard EA and the EA with memory have offline errors > 18 so the single swarm algorithms are performing better than these evolutionary techniques, although neither is particularly good.

However, when the standard EA is augmented with a diversification strategy, the offline error is considerably improved, with a best result of approx. 4.6 for the self-organizing scouts model. The effect of replacing the single swarm with increasingly numerous multi-swarms can be seen by reading down Table 2. The offline error dips below 4.6 at 5 swarms and rises above 4.6 for $M > 25$, demonstrating that multi-

Table 2. Mean offline error ± standard error for multi-CPSO/QSO and various configurations of 100 particles.

Number of Swarms, M	Configuration $M(N_n + N_n^+)$	multi-CPSO	multi-QSO
1	1(100+0)	16.51 ± 0.56 (PSO)	16.51 ±0.56 (PSO)
	1(50+50)	16.72 ± 0.54 (CPSO)	16.99 ±0.51(QSO)
2	2(25 + 25)	9.98 ± 0.35	10.33 ± 0.33
3	3(17+16)*	7.15 ± 0.27	7.04 ± 0.27
4	4(13 + 12)	5.32 ± 0.18	4.97 ± 0.18
5	5(10 + 10)	4.12 ± 0.15	3.89 ± 0.15
6	4(8+8)+2(8+10) 4(8+8)+2(10+8) 4(8+8) + 2(9+9)	3.45 ± 0.14 3.45 ± 0.14 3.42 ± 0.12	3.09 ± 0.10 3.05 ± 0.10 3.11 ± 0.10
7	6(7+7) + 1(9+7) 6(7+7) + 1(7+9) 6(7+7) + 1(8+8) 7(7+7)*	3.03 ± 0.12 3.07 ± 0.12 3.06 ± 0.11 3.00 ± 0.12	2.74 ± 0.10 2.74 ± 0.10 2.65 ± 0.08 2.66 ± 0.11
8	7(6+6)+1(8+8)	2.85 ± 0.11	2.38 ± 0.07
9	8(6+5)+1(6+6)	2.62 ± 0.11	2.25 ± 0.07
10	10(5+5)	2.63 ± 0.10	2.16 ± 0.06
11	10(5+4)+1(5+5)	2.62 ± 0.09	2.27 ± 0.06
12	11(4+4)+1(6+6)	2.72 ± 0.11	2.44 ± 0.07
14	14(4+3)*	2.73 ± 0.09	2.57 ± 0.07
16	14(3+3)+2(4+4)	2.90 ± 0.10	2.82 ± 0.07
20	20(3+2)	3.06 ± 0.08	3.16 ± 0.07
25	25(2+2)	3.50 ± 0.10	3.69 ± 0.08
50	50(1+1)	16.07 ± 0.39	5.72 ± 0.15
100	100(1+0)	44.33 ± 1.57	44.33 ± 1.57
100	100(0+1)	44.34 ± 1.58	8.44 ± 0.17

swarms perform very well when compared to diversity-enhanced EA's in this environment. The best result (2.1566) is for the 10 swarm multi-QSO.

The optimum configurations for multi-CPSO and multi-QSO are 11 and 10 swarms respectively. This is comparable to the maximum number of peaks in the dynamic function. (Although a ten peak scenario is chosen, it is possible that a lower peak may move under a higher peak, thus effectively removing one of the local maxima.) This result is consistent with the motivating picture of each swarm shadowing a peak.

The multi-QSO performs better than the equivalent multi-CPSO for most configurations. The suspicion that the Coulomb dynamics are not important for adding diversity to a neutral swarm seems to be supported by this set of experiments. It is known that the Coulomb force is responsible for some very large fluctuations in the charged

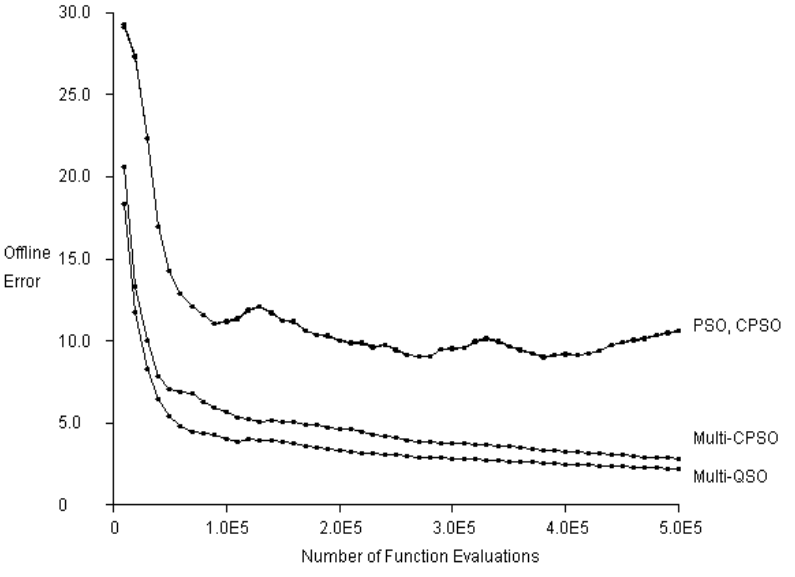

Fig. 2. Convergence plot for the offline error over time. Multi-CPSO and Multi-QSO use the 10(50+50) configuration, respectively.

swarm size [4, 5]. On the other hand, the quantum cloud approach of QSO means that the diversity increasing sub-population is concentrated close to the optimum discovered by the neutral swarm. These results suggest that this strategy is more efficient, presumably because it is less wasteful; the quantum particles are always close to where they will be needed at peak change.

As the number of swarms increases away from the optimum number of 10 swarms towards the extreme configurations 20(3+2) and 25(2+2), the two algorithms converge in terms of performance. Two and three particle neutral swarms are the smallest for which the PSO algorithm makes any sense and the performance of each neutral swarm is at its lowest. When there are many more swarms than peaks, it is expected that 'free' swarms (i.e. those that are not associated with peaks) will be constantly 'patrolling' the search space. This is because patrolling swarms will be attracted to peaks, but will be expelled by any swarms which are already optimizing that peak. If this picture is correct then diversity lies in the multi-swarm as a whole, and the single swarm diversity of CPSO or QSO is of lesser importance. Once more, this conjecture was explored qualitatively by observing animations, and this picture was observed to be broadly correct.

The 100(1+0) and 100(0+1) configurations are very distorted and the concept of a swarm is lost. In the 100(1+0) model, every particle is neutral and feels an acceleration towards its best position. The model could be viewed as 100 independent local optimizers, or a single non-interacting gas of free particles since there is no informa-

tion sharing amongst the particles. Unsurprisingly, the offline error for multi-CPSO and multi-MSO is very poor (and identical, because there are no charged particles) and worse than single swarms. However, in the 100(0+1) configuration, each particle is charged and although there is no information sharing, there will be interactions (in multi-CPSO) since each particle is repelled from every other. The difference this makes to the offline error is however very small. But, if every free particle is a quantum particle with an updated position in the cloud surrounding its best position, the offline error is reduced from 44.3 to 8.4, a stunning performance considering the simplicity of the model and the results of the standard EA (offline error ~19).

The results for the $M = 6$ and 7 configurations suggest that small differences to N_n and N_n^+ make little difference. The most symmetrical configurations, 4(8+8) + 2(9+9) and 7(7+7) are slightly better.

Figure 2 shows the evolution of the offline error over time (starting at evaluation 10000 for clarity). As can be seen, the multi-PSO outperforms the single population PSO or CPSO right from the beginning.

5 Conclusions

This paper has proposed two multi-swarm generalizations of particle swarms and compared their performance on a benchmark dynamic multi-modal environment. The first of these models, multi-CPSO, is an assembly of charged particle swarm optimizers. It is already known that surrounding a neutral or conventional PSO sub-swarm with an orbiting sub-swarm of mutually repelling particles increases swarm diversity. This improves pure PSO results in dynamic optimization of uni- and bi-modal functions of high severity [1,2,4,5]. The results presented here suggest that in a multi-modal environment a multi-CPSO is preferable to a single swarm, and that the optimum number of swarms is commensurate with the number of optima.

The second model, multi-QSO, was introduced to see if the particular dynamics (inverse square repulsions) of the charged particles in the CPSO are important, or whether a diversity increasing measure would by itself be sufficient. The results tend to suggest the latter. In QSO, the charged particles are simply randomized within a hyper-sphere centered on the swarm attractor at each update, and multi-QSO outperforms multi-CPSO for almost every configuration, and has the best overall result. A further advantage of QSO is that it is scaleable, being of linear complexity.

A comparison has also been made to experiments applying evolutionary techniques to the same environment [7]. Single swarm PSO and CPSO are broadly comparable (in fact slightly better) with an EA and a memory-enhanced EA optimization. However, the multi-swarms manage to at least halve the offline error of the diversity enhanced multi-population EA's (which represent the best evolutionary approach found in these experiments).

While the reported experiments are very encouraging, future tests should examine the sensitivity to parameter settings and also look at benchmark problems with other characteristics, e.g. where the number of peaks is significantly higher than the number

of particles. Furthermore, it would be worthwhile to develop self-adaptation mechanisms for the number of swarms and the exclusion radius r_{excl}.

References

1. Blackwell, T.M.: Swarms in Dynamic Environments. Proc Genetic and Evolutionary Computation Conference (2003) 1-12
2. Blackwell, T.M. and Bentley, P.J.: Dynamic search with charged swarms. Proc Genetic and Evolutionary Computation Conference (2002) 19-26
3. Blackwell, T.M.: Particle Swarms and Population Diversity I: Analysis. GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems (2003) 103-107
4. Blackwell, T.M.: Particle Swarms and Population Diversity II: Experiments. GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems (2003) 108-112
5. Blackwell, T.M.: Particle Swarms and Population Diversity. Soft Computing (submitted)
6. Blackwell, T.M.: Swarm Music: Improvised Music with Multi-Swarms. Proc. AISB '03 Symposium on Artificial Intelligence and Creativity in Arts and Science (2003) 41-49
7. Branke, J. and Schmeck H.: Designing Evolutionary Algorithms for Dynamic Optimization Problems. S. Tsutsui and A. Ghosh, editors, Theory and Application of Evolutionary Computation: Recent Trend*s*. Springer (2002) 239-262.
8. Branke, J.: Memory Enhanced EA for Changing Optimization Problems. Congress on Evolutionary Computation CEC´99. 3 (1999) 1875-1882.
9. Branke, J.: Evolutionary optimization in dynamic environments. Kluwer (2001)
10. Branke, J.: The moving peaks benchmark website. Online, http://www.aifb.uni-karlsruhe.de/~jbr/MovPeaks
11. Brits, R., Engelbrecht, A.P., van den Bergh, F.: A Niching Particle Swarm Optimizer. Fourth Asia-Pacific Conference on Simulated Evolution and Learning. Singapore (2002) 692-696
12. Carlisle, A. and Dozier, G.: Adapting Particle Swarm Optimization to Dynamic Environments. Proc of Int Conference on Artificial Intelligence. (2000) 429-434
13. Clerc, M. and Kennedy, J.: The Particle Swarm: Explosion, Stability and Convergence in a Multi-Dimensional Complex Space. IEEE Transactions on Evolutionary Computation. 6 (2002) 158-73
14. French, A.P. and Taylor, E.F.: An Introduction to Quantum Physics. W.W. Norton and Company (1978)
15. Hu, X. and Eberhart, R.C.: Adaptive particle swarm optimisation: detection and response to dynamic systems. Proc Congress on Evolutionary Computation (2002) 1666-1670.
16. Kennedy, J. ad Mendes, R.: Population Structure and Particle Swarm Performance. Congress on Evolutionary Computation (2002) 1671-1676
17. Kennedy, J. and Eberhart, R.C.: Particle Swarm Optimization. Proceedings of the 1995 IEEE International Conference on Neural Networks. IV (1995) 1942-1948.
18. Parsopoulos, K.E. and Vrahatis, M.N.: Recent Approaches to Global Optimization Problems through ParticleSwarm Optimization. Natural Computing 1 (2002) 235-306