LoCost: a Spatial Social Network Algorithm for Multi-Objective Optimisation

LoCost: a Spatial Social Network Algorithm for Multi-Objective Optimisation

Andrew Lewis

***Abstract*—Particle Swarm Optimisation (PSO) is increasingly being applied to optimisation of problems in engineering design and scientific investigation. While readily adapted to single-objective problems, its use on multi-objective problems is hampered by the difficulty of finding effective means of guiding the swarm in the presence of multiple, competing objectives. This paper suggests a novel approach to this problem, based on an extension of the concepts of spatial social networks using a model of the behaviour of locusts and crickets. Comparison is made between neighbouring particles based on Pareto dominance, and a corresponding repulsion between particles added to previously suggested attractive forces. Computational experiments demonstrate that the new, spatial, social network optimisation algorithm can provide results comparable to a conventional MOPSO algorithm, and improved coverage of the Pareto-front.**

I. Introduction

Particle Swarm Optimisation (PSO) is a relatively new class of optimisation algorithms, introduced by Kennedy and Eberhart in 1995 [9]. This technique is based upon the social interaction of members of a population or swarm, usually referred to as particles. Applied to optimisation, the swarm of particles travels through the problem parameter space, each particle being given a computed velocity based upon its own performance and the performance of another selected *guide* particle. In most cases the guide particle is chosen to be the current best performing particle in the swarm.

These methods have been widely applied to problems involving a single objective [12], [7]. Recent efforts have been made in developing PSO algorithms for multi-objective optimization problems requiring the simultaneous solution of a multiple set of objectives [11], [4]. In the multi-objective PSO (MOPSO), choosing the guide particle has become more complex, as the output of a MOPSO algorithm consists of a set of solutions defining the trade-off between the different objectives. Several different methods of choosing guide particles have been investigated [10], [8] but it is believed no selection scheme will give adequate performance on all test problems.

This paper suggests a novel approach to this problem, based on an extension of the concepts of spatial social networks [6]. PSO algorithms are based on social interaction, modeling the swarming behaviour of some species of animals and insects. Individuals, or particles, implicitly communicate with each other, exchanging information about the objective function landscape in neighborhoods around the particles. In the general PSO this communication is via information about the guide particles. In social networks the information exchange within neighborhoods is explicit, and defined by the network structures. Communication is between neighbours within the structures, which are most commonly constructed on the basis of particle indices.

Suganthan proposed neighbourhoods be formed on the basis of Euclidian distance between particles in the parameter space [16]. Braendler and Hendtlass proposed a variation where particles move toward neighbouring particles that have found a good solution [2]. The proposed new algorithm, modelled on the behaviour of locusts and crickets, extends this idea to the multi-objective optimisation problem by making comparison between neighbouring particles based on Pareto dominance, and adding a corresponding repulsion to the previously suggested attraction. By confining influences on individual particles to Pareto-dominance interactions between nearest neighbours only, guide particles for the swarm are dispensed with altogether.

The remainder of this paper is organised as follows. Section II gives a brief summary of the mechanics of the general Particle Swarm Optimisation algorithm while Section III describes the interactions used in Spatial Social Networks. Section IV outlines the proposed new algorithm, its application to multi-objective optimisation problems and the extension based on locust behaviour modelling. The computational results of the new system are given in Section V and finally, the conclusions and future research directions are presented in Section VI.

II. PSO Algorithms

The PSO algorithm utilises a population of potential solutions, or particles, which move around the design parameter space with every iteration. The movement of these particles is governed by two equations:

$$v_{ij}(t+1) = wv_{ij}(t) + c_1 r_{1j}(t)[y_{ij}(t) - x_{ij}(t)] + c_2 r_{2j}(t)[\hat{y}_{ij}(t) - x_{ij}(t)] \quad (1)$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \quad (2)$$

Here, $v_{ij}(t)$ is the velocity of particle *i* in dimension *j* at time *t* and $x_{ij}(t)$ is the position of particle *i* in dimension *j* at time *t*. The velocity of a particle depends on both the best position that particle has found to time *t*, $y_{ij}(t)$, and the best solution the entire swarm has found has found to time time *t*, $\hat{y}_{ij}(t)$. The inertial component is scaled by constant *w*, and c_1 and c_2 are constants, usually defined between 0.5 and 3.0, used to control the impact of the local and global

Andrew Lewis is with the Institute for Integrated and Intelligent Systems, Griffith University, Queensland, Australia (email: a.lewis@griffith.edu.au).

978-1-4244-2959-2/09/$25.00 ©

components in velocity equation 1. The vector *r* is a vector of random numbers evenly distributed between zero and one generated for each particle at each time step *t*. Once new particle velocities have been calculated, the position of each particle is updated as in equation 2.

III. Spatial Social Networks

In the velocity equation of the general PSO, the motion of individual particles is determined by an inertial component, the cognitive component that contributes information about the best position a particle has found for itself, and the social component that contributes information about the best position the swarm as a whole has found. In social-based PSO algorithms some subset of the whole swarm defined by the social network structure shares information about a neighborhood, typically a subset of the space the whole swarm inhabits. The most common way information is shared is to replace the best solution for the swarm as a whole, $\hat{y}_{ij}$ in equation 1 with some fraction of the best solution for the set of neighbours, depending on the distance the particle is from this "local guide".

In standard social networks, neighborhoods are based on particle indices. For example, the neighborhood of a particle with index i might include the particles with indices $i-1$, i and $i+1$. This has the advantages that:

- no distance calculations are required to form the neighborhoods, so this approach is computationally inexpensive, and
- information is spread throughout the population irrespective of particle location in the search space.

In spatial social networks, neighborhoods are formed based on Euclidian distance in search space. The neighborhood of a particle with index i is defined as the set of (some given number of) particles with the smallest value of $\mathcal{E}(\vec{x_i}, \vec{x_{i'}})$, the Euclidian distance between particles i and i'. For fitness-based spatial neighborhoods this is amended to include a fitness term for the particles in the neighborhood, i.e. $\mathcal{E}(\vec{x_i}, \vec{x_{i'}}) \times f(\vec{x_{i'}})$.

Spatial social networks are computationally more expensive because at each iteration the Euclidian distances must be calculated. However, they have the advantage that neighborhoods can be overlapping and change dynamically. In many cases where optimisation is applied to real-world problems, the computation necessary to derive objective function values can far outweigh the distance calculation overheads, the computational cost of the optimisation algorithm becoming effectively negligible.

IV. A Multi-objective Model

The social network algorithms described so far are quite adequate for problems in which a single objective must be optimised, but when the problem involves multiple objectives the question of what constitutes a suitable guide for the swarm becomes problematic. To address this difficulty, it is proposed that a solution found in a neighborhood be considered "better" based on Pareto dominance. For solution vectors $\vec{x_i}$ and $\vec{x_{i'}}$, when the following conditions are met:

- $\vec{x_i}$ is at least as good as $\vec{x_{i'}}$ for all the objectives, and
- $\vec{x_i}$ is strictly better than $\vec{x_{i'}}$ for at least one objective

then $\vec{x_i}$ is said to "dominate" $\vec{x_{i'}}$ (denoted $\vec{x_i} \prec \vec{x_{i'}}$). In the case where $\vec{x_i}$ and $\vec{x_{i'}}$ dominate other solution vectors but not each other they are deemed mutually optimal solutions and referred to as Pareto-optimal.

It may be sufficient to adapt the fitness-based spatial social network by replacing the fitness term, $f(\vec{x_{i'}})$ with a term that takes into account Pareto dominance, but it is unclear how this might be scaled: the Pareto dominance gives no information about relative *degree* of dominance. A different approach might be more effective.

PSO algorithms are based on observation of animal behaviour, with simple rules abstracted from this observation giving rise to sophisticated emergent behaviour. Braendler and Hendtlass proposed that particles move *toward* neighbouring particles that have found good solutions. Simpson et al. [15] have observed cannibalism in cricket swarms and concluded that "band members must continually move to avoid being eaten by similarly [nutrient] deprived conspecifics". This suggest that a motion vector *away* from a particle that has not found a good solution may be just as important as one *toward* good solutions.

From this inspiration the vector equation 1 was reformulated:

$$v_{ij}(t+1) = \begin{cases} wv_{ij}(t) + c_1 r_{1j}(\vec{x_{i'}}(t) - \vec{x_i}(t)) : \vec{x_{i'}}(t) \prec \vec{x_i}(t) \\ wv_{ij}(t) - c_1 r_{1j}(\vec{x_{i'}}(t) - \vec{x_i}(t)) : \vec{x_{i'}}(t) \not\prec \vec{x_i}(t) \end{cases} \quad (3)$$

i.e. motion is toward a neighbour if it is Pareto-dominant, and away otherwise. Note that in the case the particles are mutually non-dominating the default is for generally dispersive motion. This is in direct contrast to algorithms incorporating a congregative function (e.g. [14].) The interactions considered are between nearest neighbour pairs only, i.e. between the particle with index i and its neighbouring particle, i', where $\mathcal{E}(\vec{x_i}, \vec{x_{i'}})$ is a minimum. An outline of the operation of the resulting algorithm is given in Algorithm 1.

While the use of neighbourhoods, of which this is the limiting case of size $n_{\mathcal{N}} = 2$, has been widely explored, this formulation of a repulsive motion vector is novel. The closest analogues are the repulsion phases of multi-phase PSO [1], [13] but these employ a time-varying switching of the sign of the c_1 and c_2 coefficients of equation 1. This causes the *whole* swarm, or sub-swarms, to move away from global best position, $\hat{y}_{ij}$, and sometimes also the local best position, y_{ij}, in order to promote swarm diversity. Note that these motions require these positions to be defined, a problematic exercise in multi-objective problems as has already been discussed.

V. Computational Experiments and Results

In order to evaluate the performance of the proposed algorithm, it was compared to a conventional MOPSO algorithm with Sigma update[10]. Each algorithm had a population of 50 and separate runs, with different random seeds, were performed for increasing numbers of iterations on standard test functions, to examine relative performance at different

Algorithm 1 LoCost

```
Initialise population
repeat
  for all particles in the population do
    Evaluate the particle fitness
  end for
  for all particles in the population do
    Find nearest neighbour particle
    if Neighbour is dominant then
      Adjust the motion vector toward neighbour, using Eq. 3
    else
      Adjust the motion vector away from neighbour, using Eq. 3
    end if
    Update the archive
    Update the particle positions, using Eq. 2
  end for
until Termination condition met
Return archive
```

TABLE I
TEST FUNCTIONS

Test	Function	Constraints
ZDT1	$g(x_2,\cdots,x_n) = 1 + 9(\sum_{i=2}^{n} x_i)/(n-1)$	$x_i \in [0,\ 1]$
	$h(f_1,g) = 1 - \sqrt{f_1/g}$	$i = 1,2,\ldots,n$
	$f_1(x_1) = x_1$	$n = 2$
	$f_2(\vec{x}) = g(x_2,\cdots,x_n)\cdot h(f_1,g)$	
ZDT2	$g(x_2,\cdots,x_n) = 1 + 9(\sum_{i=2}^{n} x_i)/(n-1)$	$x_i \in [0,\ 1]$
	$h(f_1,g) = 1 - (f_1/g)^2$	$i = 1,2,\ldots,n$
	$f_1(x_1) = x_1$	$n = 2$
	$f_2(\vec{x}) = g(x_2,\cdots,x_n)\cdot h(f_1,g)$	
ZDT3	$g(x_2,\cdots,x_n) = 1 + 9(\sum_{i=2}^{n} x_i)/(n-1)$	$x_i \in [0,\ 1]$
	$h(f_1,g) = 1 - \sqrt{f_1/g} - (f_1/g)\sin(10\pi f_1)$	$i = 1,2,\ldots,n$
	$f_1(x_1) = x_1$	$n = 2$
	$f_2(\vec{x}) = g(x_2,\cdots,x_n)\cdot h(f_1,g)$	
ZDT4	$g(x_2,\cdots,x_n) = 1 + 10(n-1)$	
	$+\sum_{i=2}^{n}(x_i^2 - 10cos(4\pi x_i))$	$x_1 \in [0,\ 1]$
	$h(f_1,g) = 1 - \sqrt{f_1/g}$	$x_i \in [-5,\ 5]$
	$f_1(x_1) = x_1$	$i = 2,\ldots,n$
	$f_2(\vec{x}) = g(x_2,\cdots,x_n)\cdot h(f_1,g)$	$n = 2$

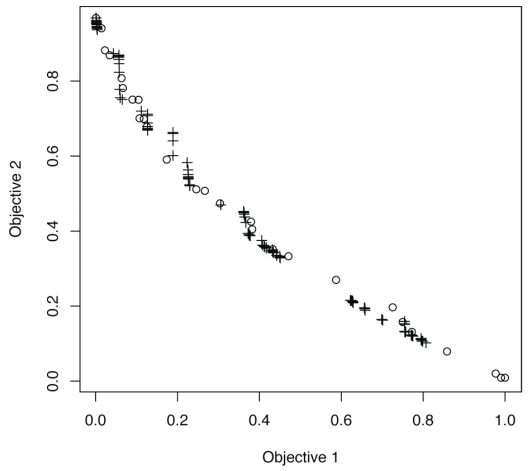

Fig. 1. Approximate Pareto-front for the ZDT1 test function at 40 iterations. (MOPSO shown as "+", LoCost as "o")

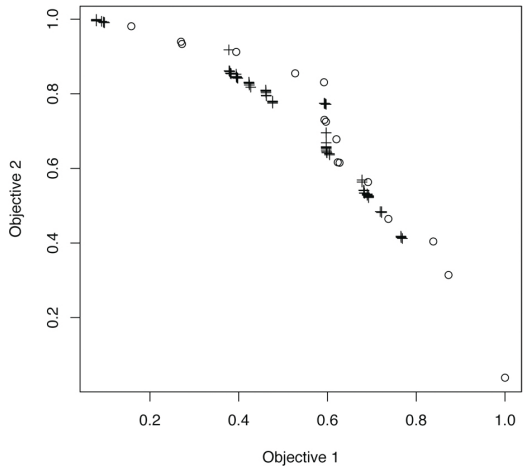

Fig. 2. Approximate Pareto-front for the ZDT2 test function at 40 iterations. (MOPSO shown as "+", LoCost as "o")

times during the progress of the optimisation. A number of well known test functions were used in this work, taken from [17]. The test functions used have a variety of convex, non-convex, discontinuous and multimodal Pareto-fronts and are shown in Table I. The dimension of parameter space, N, was 2 for all test functions. Following the arguments by Deb *et al.* [5] test functions with two objectives are sufficient to evaluate a multi-objective optimisation method and results from such tests are also valid for higher dimensional problems.

Figure 1 to 4 show the attainment surfaces, the approximations to the Pareto-front, achieved by the archive members of the proposed algorithm, LoCost, and the MOPSO algorithm after 40 iterations for ZDT1, ZDT2, ZDT3 and ZDT4 test functions.

By inspection of the figures it may be seen that the proposed, new LoCost algorithm has performed quite comparably to the conventional MOPSO algorithm. A qualitative comparison appears to show that LoCost has an appreciably greater coverage of the approximation to the Pareto-front than the MOPSO algorithm, "filling the gaps" in a number of places. In particular, LoCost has consistently found Pareto-optimal solutions for higher values of the second objective than MOPSO. Given the form of the test functions, where $f_1(\mathbf{x}) = x_1$, this series of functions is notorious for the ease with which Pareto-optimal results can be found in the region where f_1 approaches 0, but the difficulty of obtaining results with small f_2. LoCost appears to have a significant advantage in this region, compared to MOPSO.

From Figures 1 to 4 it would appear there is little to choose between the algorithms in terms of convergence: each algorithm reaches a similar degree of convergence at the same time. As has already been noted, however, there are subjective differences in the coverage achieved. To quantitatively evaluate the coverage of each method a new performance measure Ψ was introduced. For a two-dimensional problem, one of the objectives is selected and

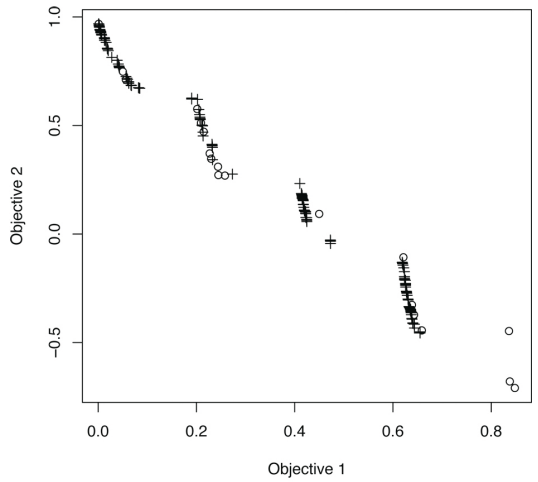

Fig. 3. Approximate Pareto-front for the ZDT3 test functions at 40 iterations. (MOPSO shown as "+", LoCost as "o")

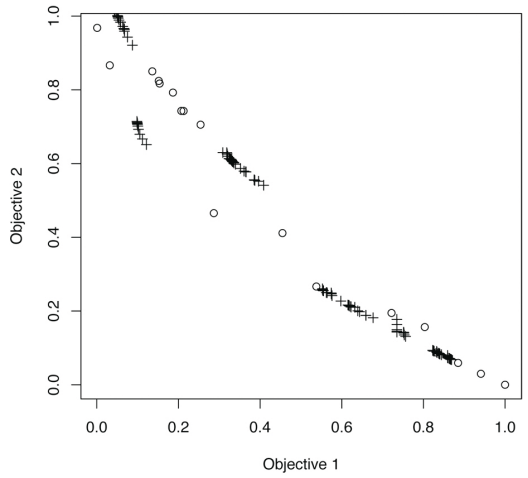

Fig. 4. Approximate Pareto-front for the ZDT4 test function at 40 iterations. (MOPSO shown as "+", LoCost as "o"

the respective dimension of the objective space is split into a number of equally sized intervals ψ_n, where the number of intervals is defined as the maximum of the size of the swarm and the size of the archive. Each of these intervals is inspected to see if a non-dominated solution lies between the upper and lower bounds of the interval. If at least one such particle exists, the interval is scored 1, otherwise its score is set to 0. The metric is given by the percentage of of "occupied" intervals. It is thus a measure of the total extent of the Pareto-front covered, and the uniformity of the coverage. For the purposes of the comparison between two algorithms, where the true Pareto-front is not assumed to be known, the coverage is measured relative to the total, aggregated approximation to the Pareto-front of the two archives.

The calculation of the metric can be formalised as:

$$\Psi = \frac{100}{\max\{|\mathcal{A}|, |\mathcal{S}|\}} \sum_{n=1}^{\max\{|\mathcal{A}|,|\mathcal{S}|\}} \psi_n \tag{4}$$

where $\mathcal{A}$ and $\mathcal{S}$ represent the size of the archive and the size of the swarm respectively. The individual "buckets" ψ_n are then defined as

$$\psi_n \begin{cases} 1, & \text{if } \exists \vec{f}(\vec{x}) \in \mathcal{PF}, \beta_{n-1} \leq f_1(\vec{x}) < \beta_n \\ 0, & \text{else} \end{cases} \tag{5}$$

with f_1 being one of the objectives and β_n defining the upper boundary for bucket n. Any objective can be chosen to be f_1 but the same objective has to be used for all the buckets for obvious reasons. The upper boundaries for the buckets β_n are defined as

$$\beta_n = L_1 + n \cdot \frac{U_1 - L_1}{\max\{|\mathcal{A}|, |\mathcal{S}|\}} \tag{6}$$

where U and L mark the upper and lower boundaries of the analysed portion of the Pareto-front based on the chosen objective f_1. The lower boundary for bucket n is defined as β_{n-1}.

The values of the coverage metric for each algorithm on each test function are shown in TableII. It can be seen that for test functions ZDT1 and ZDT2 LoCost did achieve significantly improved coverage. On the ZDT3 test case, MOPSO returns somewhat better coverage results, despite missing one of the Pareto-front segments entirely, the one for low values of f_2, which LoCost has found. For ZDT4 the MOPSO algorithm appears to deliver significantly better coverage but, as can be seen from Figure 4, it is at the expense of convergence, MOPSO having been trapped by the deceptive, multimodal fronts of this test function. On all test functions, while MOPSO may deliver more, densely-packed solutions on the attainment surface, LoCost has given better dispersion and coverage, albeit with sparsely scattered points..

TABLE II
COVERAGE METRIC, Ψ, FOR ALL TEST FUNCTIONS AT 40 ITERATIONS

	Test Function			
	ZDT1	ZDT2	ZDT3	ZDT4
LoCost	75	45	40	30
MOPSO	65	35	45	50

For the purposes of comparison, the hypervolume metric for each algorithm on all test functions is shown in Table III. For test functions ZDT1 and ZDT4, the performance of both algorithms as measured by the hypervolume is comparable. On ZDT2 LoCost has a significantly better performance than MOPSO, as might have been expected from inspection of Figure 2 where the extent of its coverage can be seen to be significantly better, particularly for solutions with low values of f_2. On ZDT3 MOPSO performs a little better, which may be attributable to the greater number of points it tends to accumulate along the attainment surface.

TABLE III
HYPERVOLUME METRIC FOR ALL TEST FUNCTIONS AT 40 ITERATIONS

	Test Function			
	ZDT1	ZDT2	ZDT3	ZDT4
LoCost	0.33	0.70	0.07	0.35
MOPSO	0.32	0.09	0.11	0.32

VI. CONCLUSIONS

The experiments described in this paper have demonstrated that a new, spatial, social network optimisation algorithm inspired by the swarming behaviour of crickets and locusts can provide results comparable to a conventional MOPSO algorithm, and improved coverage of the Pareto-front. It achieves these results without the use of any guide particles, thus avoiding the difficulties of selecting suitable particles for this purpose in multi-objective optimisation problems.

The results reported in this paper are preliminary, but promising. Future work will include detailed investigation of the behaviour of the algorithm and its dependence on a number of factors, particularly including the effect of swarm size. Buhl *et al.* have identified a critical density for the transition from disordered to ordered behaviour in locust swarms [3] which suggests that this may be an important factor in models of behaviour of this type. The statistical significance of the results obtained in this study should be verified, and the algorithm's performance on a greater range of test functions assessed. In addition, the scaling of motion vectors and randomisation coefficients need further investigation, and the algorithm should be compared with a wider variety of competing algorithms to investigate its relative performance.

REFERENCES

[1] B. Al-Kazemi and C. Mohan. Multi-phase generalization of the particle swarm optimization algorithm. In *Proceedings of the IEEE Congress on Evolutionary Computation 2002*, volume 2, pages 1057–1062, 2002.

[2] D. Braendler and T. Hendtlass. The suitability of particle swarm optimization for training neural hardware. In *Proc. 15th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, volume 2358 of *LNCS*, pages 190–199. Springer-Verlag, 2002.

[3] J. Buhl, D. Sumpter, I. Couzin, J. Hale, E. Despland, E. Miller, and S. Simpson. From disorder to order in marching locusts. *Science*, 312:1402–1406, 2006.

[4] C. Coello Coello and M. Lechuga. MOPSO: A proposal for multiple objective particle swarm optimization. In *Proc. IEEE World Congress on Computational Intelligence*, 2002.

[5] K. Deb. Multi-objective genetic algorithms: Problem difficulties and construction of test problems. *Evolutionary Computation*, 7(3):205–230, 1999.

[6] A. Engelbrecht. *Fundamentals of Computational Swarm Intelligence*. Wiley, 2005.

[7] S. Guru, S. Halgamuge, and S. Fernado. Particle swarm optimisers for cluster formation in wireless sensor networks. In *Proc. 2005 International Conference on Intelligent Sensors, Sensor Networks and Information Processing Conference*, pages 319–324, 2005.

[8] D. Ireland, A. Lewis, S. Mostaghim, and J. Lu. Hybrid particle guide selection methods in multi-objective particle swarm optimization. In *Proc. 2nd IEEE International Conference on eScience and Grid Computing*, 2006.

[9] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proc. International Conference on Neural Networks*, 1995.

[10] S. Mostaghim and J. Teich. Strategies for finding good local guides in multi-objective particle swarm optimization (MOPSO). In *Proc. IEEE Conference on Swarm Intelligence Symposium*, pages 26–33, 2003.

[11] K. Parsopoulos and M. Vrahatis. Particle swarm optimization methods in mutliobjective problems. In *Proc. of the 2002 ACM Symposium on Applied Computing*, pages 603–607, 2002.

[12] Y. Rahmat-Sammi, N. Jin, and S. Xu. Particle swarm optimization (pso) in electromagnetics: Let the bees design your antennas. In *22nd Annual Review on Progress in Applied Electromagnetics*, 2006.

[13] J. Riget and J. Vesterstrøm. A diversity-guided particle swarm optimizer - the ARPSO. Technical report, University of Aarhus, Department of Computer Science, 2002.

[14] S.He, Q. Wu, J. Wen, J. Saunders, and R. Paton. A particle swarm optimizer with passive congregation. *BioSystems*, 78:135–147, 2004.

[15] S. Simpson, G. Sword, P. Lorch, and I. Couzin. Cannibal crickets on a forced march for protein and salt. *Proc. of the National Academy of Sciences of the United States of America (PNAS)*, 103(11):4152–4156, 2006.

[16] P. Suganthan. Particle swarm optimiser with neighborhood operator. In *Proc. IEEE Congress on Evolutionary Computation*, pages 1958–1962, 1999.

[17] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms on test functions of different difficulty. In *Proc. 1999 Genetic and Evolutionary Computation Conference. Workshop Program*, pages 121–122, 1999.