A New Continuous Optimization Algorithm Based on Sociological Models

Mathew Mithra Noel and Thomas C. Jannett

Abstract - **Genetic algorithms (GAs) have a wide variety of applications in control. However, GAs may suffer from slow convergence rates, and require the user to make difficult choices of ranking and scaling schemes and subpopulations that may lead to complexities in implementation. A new computationally inexpensive alternative to GAs, the continuous adaptive culture model (CACM), is proposed in this paper. This new optimization algorithm is inspired by sociological models of culture dissemination and uses operators that act directly on vectors of real numbers to avoid the computation associated with binary encoding and decoding in GAs. The new algorithm does not use global information sharing which makes it amenable to parallel implementation since computational bottlenecks are avoided. The De Jong test suite of optimization problems is used to test the new optimization algorithm. Effects of various parameters on the performance of the algorithm are investigated through simulations.**

I. Introduction

GLOBAL search algorithms like genetic algorithms (GAs) have a wide variety of control applications like design of fuzzy systems, adaptive and nonlinear control, neural network training, estimation and routing. However, GAs sometimes exhibit slow convergence and require a difficult choice of ranking and scaling schemes and subpopulations that may lead to complexities in implementation. When using GAs in problems requiring optimization of real-valued functions of real variables, the binary representation of tentative solutions that is needed for crossover and mutation is not natural. In this paper, a new continuous adaptive culture model (CACM) optimization algorithm that is computationally inexpensive and easy to implement is proposed as an alternative to GAs. The CACM algorithm is inspired by sociological models of culture dissemination and uses operators that act directly on vectors of real numbers.

Manuscript received September 27, 2004. This work was supported in part by the Office of Naval Research under Grant N00014-03-1-0751.

Mathew Mithra Noel is pursuing the Ph. D. in Computer Engineering at The University of Alabama at Birmingham, Birmingham, Al 35294 USA (e-mail: mathew.mithra@ gmail.com).

Thomas C. Jannett is Professor of Electrical and Computer Engineering at The University of Alabama at Birmingham, Birmingham, Al 35294 USA (e-mail: tjannett@uab.edu).

The new CACM algorithm employs a random search, but instead of generating a sequence of random points in the solution space as proposed in [1], the new algorithm uses a population of potential solutions that is evolved to generate better solutions. The evolution mimics the way animal societies composed of simple individuals solve complex optimization problems, and is based on the adaptive culture model (ACM) that was published in 1997 by Robert Axelrod [2] as a model of the dissemination of culture. In the CACM algorithm, the population is organized into neighborhoods and individuals move towards the best solution found in their particular neighborhood. This is similar to the way an ant swarm finds the shortest path to food sources. The ant system algorithm [3]-[7] is an optimization algorithm that uses rules copied from the behavior of real ants to solve routing problems. There is no centralized control or global sharing of information and individuals act based on local information alone. Another example of an optimization algorithm based on models of animal behavior is the particle swarm optimization (PSO) algorithm [8]-[9] that was inspired by the behavior of flocking birds.

In the ACM the global fitness of a population rises as each individual interacts with its neighbors. The ACM serves as the basis for the classical ACM optimization scheme in which tentative solutions (represented as strings) are organized in a rectangular array. Since the individuals are arranged in a rectangular array the individuals at the boundary will have only two neighbors instead of four; this can be avoided by assigning to each boundary individual another individual which is directly opposite on the opposite boundary as its neighbor. From a geometrical point of view, this amounts to folding a rectangle into a cylinder and then folding the resulting cylinder into a toroid. For example, to solve the traveling salesman problem (TSP) tentative solutions could be strings representing cities to be visited. Each individual is compared to its neighbors (above, below and on either side) and a non-matching character in the neighboring string is adopted if the neighbor is fitter. The process of adopting non-matching characters from fitter neighbors is repeated for each individual in the population. The individuals are updated row-wise. Thus, when each row is updated, good features (characters) from strings in rows above and below are copied to the individuals in that row resulting in higher fitness. The classical ACM algo-

rithm can be used to solve only combinatorial optimization problems like the TSP since tentative solutions are represented as strings. However, many fundamental problems in engineering like model fitting, estimation, classification, training neural networks and controller design require optimization of real-valued functions of real variables. Thus, we consider the following generic function optimization problem:

$$\text{minimize } f(x_1, x_2, ..., x_N) \tag{1}$$

where $f : R^N \to R$.

Tentative solutions to this problem will be real vectors of length *N*. In the classical GA implementation, real vectors are encoded in binary to perform crossover and mutation in order to evolve a new population that is fitter than the original population on the average. Since solutions close to good solutions are likely to be good as well, mutation of a given binary encoded individual should in general produce a new individual that is close (Hamming distance) to the original individual. This is because most functions that arise in practical applications are discontinuous on only a small subset of the solution space. Thus if $\vec{x}_1$ is close to $\vec{x}_2$, then f($\vec{x}_1$) will be close to f($\vec{x}_2$) in most parts of the solution space. Also, recombination of two individuals should in general produce a new individual that is close to either of the parent individuals with higher probability; this is because an individual that is midway between two fit individuals will in general not be fit.

In this paper, we introduce new operators that offer the desired behavior described above while operating directly on real vectors. The new operators avoid computation associated with binary coding and decoding by acting directly on vectors of real numbers. The new operators are used to generalize the classical ACM algorithm resulting in the CACM algorithm that is useful for continuous optimization**.** The De Jong test suite of optimization problems is used to test the new optimization algorithm. Effects of various tuning parameters on the performance of the algorithm are investigated through simulation.

II. Operators that Act Directly on Vectors of Real Numbers

A. *A New Crossover Operator*

Consider two vectors $\vec{x}_1$ and $\vec{x}_2$. The following crossover operator is proposed to achieve recombination of $\vec{x}_1$ and $\vec{x}_2$ to produce $\vec{x}$:

$$\vec{r} = \vec{U} - 0.5\vec{I} \tag{2}$$

$$\Delta\vec{x} = \sigma_r \vec{r}$$

IF $u < 0.5$

$$\vec{x} = \vec{x}_1 + \Delta\vec{x}$$

ELSE

$$\vec{x} = \vec{x}_2 + \Delta\vec{x}$$

END

Where

$\vec{r}$ -Random vector with each component distributed independently and uniformly between -0.5 to 0.5

$\vec{U}$ -Vector of dimension N with each component uniformly distributed between 0 and 1

u -Uniform random variable between 0 and 1

$\vec{I}$ -Vector of dimension N with each component 1

σ_r -Scaling constant which controls size of recombination (distance from parent).

The behavior of the crossover operator (2) can be understood as follows. Given two vectors $\vec{x}_1$ and $\vec{x}_2$, choose one with probability 0.5 (both are equally likely). Then a new vector close to the selected parent vector is computed by adding to the selected vector a random vector which can point in all directions with equal probability and whose length is controlled by σ_r. Thus, the new individual generated by recombination will be close to either of the parents with a high probability. A uniform probability distribution was used to create $\vec{r}$ because it is symmetric and has the highest entropy. A symmetric distribution is used since there is no reason to expect individuals on any side to be better. A large value of σ_r means that the new individual has a higher probability of being far from a parent; thus higher values can be used to perform a more random search since information contained in the parents is ignored. However, computational time will be wasted if a pure random search is made so an intermediate value has to be chosen.

Note that the new operator (2) will be referred to as a "crossover" operator although a portion of each parent individual is not copied directly to the resultant individual (offspring); the new operator takes two parent individuals and produces another individual that is close to either of the parents in a probabilistic sense just like the GA crossover operator. Given the parents, as long as the offspring produced by the new operator (2) and the classical crossover operator have the same probability distributions, there are no differences between the new and classical operators with respect to convergence properties.

B. *A New Mutation Operator*

Mutation is a device to search locally around a given point in the solution space. An individual mutation should produce a new individual that is close to the parent individual. Given a real vector $\vec{x}$, a new vector that is close probabilistically can be computed as follows:

$$\vec{r} = \vec{U} - 0.5\vec{I} \quad (3)$$

$$\Delta\vec{x} = \sigma_m \vec{r}$$

$$\vec{x} = \vec{x} + \Delta\vec{x}$$

Where

$\vec{r}$ - Random vector with each component distributed independently and uniformly between -0.5 to 0.5

$\vec{U}$ - Vector of dimension N with each component uniformly distributed between 0 and 1

$\vec{I}$ - Vector of dimension N with each component 1

σ_m - Scaling constant which controls size of mutation.

Larger values of σ_m allow the new individual to be distant from its parent.

III. A New Continuous Optimization Algorithm

This paper uses the new continuous mutation operator (3) to generalize the classical ACM algorithm for continuous optimization. The new CACM algorithm is similar to ACM algorithm in that the average fitness of the population increases as a result of local interactions of individuals with their neighbors. Tentative solutions are directly represented as real vectors arranged in a rectangular array as in the classical ACM algorithm. The population of tentative solutions is evolved as follows. Each vector is either mutated with probability P_{mut} or it is replaced with probability P_{xov} by a mutated copy of its most fit neighbor. Replacing an individual by a mutated version of its best neighbor will be referred to as fitness adoption. The σ associated with each mutation operation (see (3)) will be referred to as σ_m and σ_f, respectively. Consider a population of size P of tentative solution vectors arranged as a rectangular array of size ROWS*COLUMNS = P. The vector (tentative solution) at row i and column j is denoted by $\vec{x}_{ij}$. The new continuous optimization algorithm proposed is as follows:

The CACM Algorithm

Initialize P_{mut}, P_{xov}, σ_m and σ_f. (4)

FOR i = 1 to ROWS
  FOR j = 1 to COLUMNS
    Randomly initialize $\vec{x}_{ij}$ (choose a point in the solution space with uniform probability).
  END
END

Do the following until convergence or maximum iteration is reached.

FOR i = 1 to ROWS
  FOR j = 1 to COLUMNS
    Step 1: Mutate $\vec{x}_{ij}$ using (3) with probability P_{mut} and sigma equal to σ_m in (3).
    Step 2: Find the best individual $\vec{b}_{ij}$ in the neighborhood containing $\vec{x}_{ij}$.
    Step 3: Do the following with probability P_{xov}. Replace $\vec{x}_{ij}$ by a mutated copy of $\vec{b}_{ij}$ found in step 2 with sigma equal to σ_f in (3).
    END
END

In the above algorithm the neighborhood of $\vec{x}_{ij}$ refers to individuals in positions (*i, j*), (*i+1, j*), (*i-1, j*), (*i, j+1*) and (*i, j-1*). Thus when the individual at (*i, j*) is updated in step 3, its fitness increases since it moves closer to the best individual in that neighborhood. Since the population is updated row-wise the next individual to be updated after (*i, j*) will be (*i, j+1*). Thus the individual at (*i, j+1*) has a chance to learn (be updated) from individuals on its left and on top (on previous row) which have already been updated and hence are fitter than the individual at (*i, j+1*). In this way, as the population is updated row-wise, each individual profits from the individuals in its neighborhood that have already been updated. This positive feedback mechanism, which is present in many complex biological systems, is responsible for faster convergence compared to GAs. In GAs two good solutions are recombined to produce an individual that is more fit than the parent individuals on the average. However, in the CACM algorithm an individual being updated has a chance to learn from all previously updated individuals leading to faster convergence.

Thus, it is unlikely that an individual being updated is the best individual in its neighborhood because of the presence of other individuals that have already been updated. However, if an individual is the best individual in its neighborhood, it cannot learn from other individuals in its neighborhood; the algorithm converges prematurely (to a local minimum) if the best individual is not allowed to change. In (4) if $\vec{x}_{ij}$ happens to be the best individual in its neighborhood, it will be replaced with probability P_{xov} by a mutated version of itself.

In GAs, "elitism" refers to the strategy of allowing good solutions to stay unchanged from one iteration to the next

to prevent good solutions from getting lost. The following experiment was done to study the effects of using elitism in the CACM algorithm. Algorithm (4) was modified so that when $\vec{x}_{ij}$ is the best individual in its neighborhood it is replaced by a mutated copy of itself only with a certain probability, $P_{elitist}$. For the sphere test function in the De Jong test suite, the average cost over 100 runs for different values of $P_{elitist}$ is shown in Fig. 1. It was found that convergence was slowed for $P_{elitist} < 1$. Thus, use of elitism is not beneficial in case of the CACM algorithm.

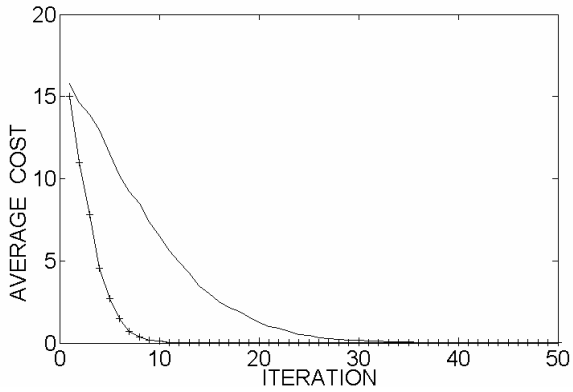

Fig. 1. Effect of $P_{elitist}$ on the convergence rate of the CACM algorithm for sphere test function. Convergence is slowed if locally best solutions are not allowed to change. $P_{elitist}$ is the probability that the best individual is changed (solid line: $P_{elitist}$ = 0.3 and plus: $P_{elitist}$ = 1).

A P_{xov} of 0.9 was found to work well for all test functions.

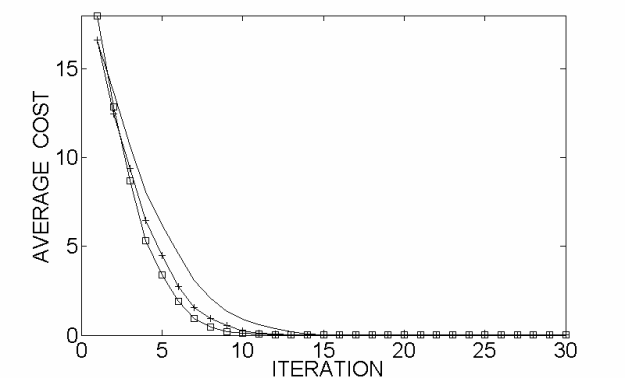

Fig. 2. Effect of P_{mut} on the convergence rate of the CACM algorithm for sphere test function. The average cost was taken over 100 independent runs. P_{mut} higher than 0.3 resulted in slower convergence. A choice of 0.3 for P_{mut} was found to work for all test functions (solid: P_{mut} = 0, plus: P_{mut} = 0.1, and square: P_{mut} = 0.3).

The mutation probability P_{mut} has a larger effect on performance for the CACM algorithm than for GAs; this is because better solutions found by mutation are adopted while bad solutions are rejected on the average. Thus, the CACM algorithm provides a better way of incorporating the mutation operator than GA. In the new CACM algorithm, larger values of P_{mut} (P_{mut} = 0.3) were needed to achieve high convergence rates than are used in GAs. Fig. 2 shows the average cost over 100 runs of the CACM algorithm with different mutation rates for the sphere test function in the De Jong suite.

It was found that convergence as well as the quality of the final solution was improved by letting σ_f go to zero linearly. Also, a linearly decreasing σ_f was found to yield better results than a random or constant σ_f (Fig. 3). The following formula was found to yield good results:

$$\sigma_f(k) = -k / K_{max} + 1 \qquad (5)$$

where

k - Iteration number

K_{max} - Maximum iterations allowed.

Thus, the parameters σ_f and σ_m control the amount of movement towards best neighboring solution and mutation, respectively. For functions with many local minima, a larger σ_m is needed to avoid getting trapped in local minima.

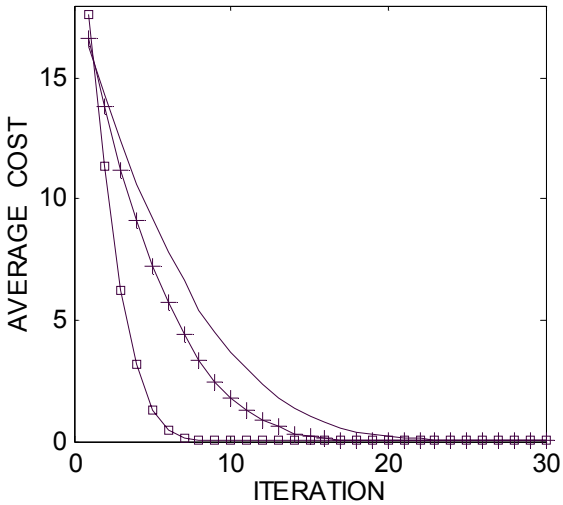

Fig. 3. The effects of various choices of σ_f on convergence of the CACM algorithm for the sphere test function. The average cost was taken over 100 independent runs. A linearly decreasing σ_f results in faster convergence and better quality of the final solution than a constant or random σ_f. (solid: σ_f = 0.1, min = 2.2819e-004; plus: σ_f = 0.5u, min = 8.3487e-004; square: σ_f = -k/50+1, min = 7.3796e-005).

The nature of the cost function is not usually known beforehand, so a random selection has to be made for σ_m.

This allows the amount of mutation to vary from iteration to iteration. The following selection of σ_m gave good results for all test functions.

$$\sigma_m(k) = 2U + 3|N| \tag{6}$$

Where

k - Iteration number
U - Uniform random variable between 0 and 1
N - Normal random variable with mean 0 and variance 1.

IV. Performance of CACM algorithm on Benchmark Cost Functions

The CACM algorithm was found to converge in significantly less iteration than classical GAs for functions in the De Jong test suite. For functions with few local minima convergence to 0.1 accuracy was achieved within 10 iterations. Since the CACM algorithm avoids the fundamental problem of binary encoding this represents a significant reduction in flops. The CACM algorithm avoids the ranking problem inherent in GAs. Thus good solutions are kept in the population without sacrificing diversity due to use of local neighborhoods alone instead of global information sharing. The performance of the new CACM algorithm for some benchmark test functions found in the De Jong suite is shown in Fig. 4-6. The test functions have a global minimum of zero. Fig. 4 shows the performance of the CACM algorithm on a test function with flat regions (Rosenbrock's valley test function).

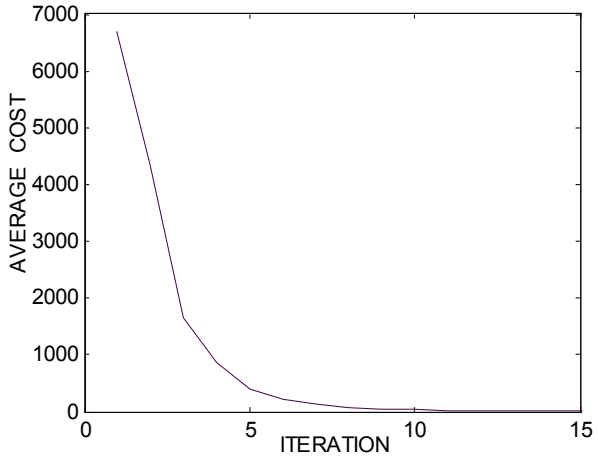

Fig. 4. Performance of CACM algorithm on a test function with a flat region surrounding the global minimum. The average cost was taken over 100 independent runs. Flat regions pose a challenge to optimization algorithms; since the gradient is zero, there are no good search directions. Consequently, a random walk is the best approach to search flat regions.

Flat regions are difficult to search since there are no good search directions like the gradient. However, the CACM algorithm is able to find the global minimum embedded in a flat region by performing a random walk when it encounters flat regions. The performance of the CACM algorithm on test functions with large numbers of local minima is shown in Fig. 5 and 6.

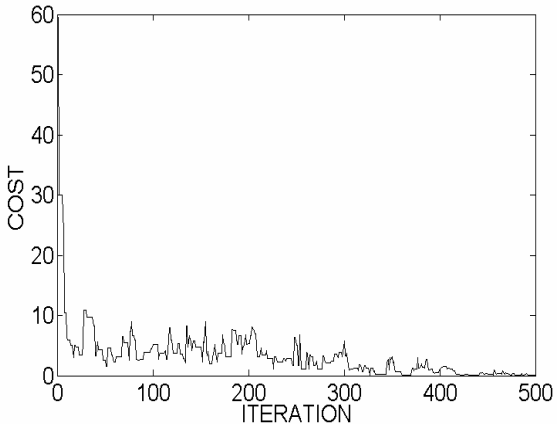

Fig. 5. Performance of CACM algorithm on a function with a large number of uniformly distributed local minima (Rastrigin's function). Average cost is not used because of the large number of iterations required to find the minimum.

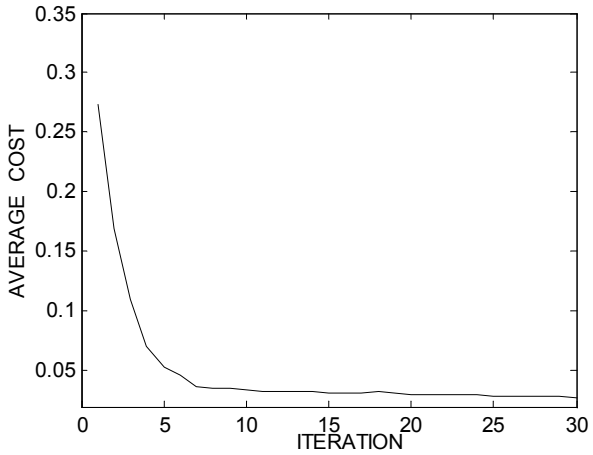

Fig. 6. Performance of the CACM algorithm on a test function with large numbers of local minima (Griewangk's function). The average cost was taken over 100 independent runs.

The best individuals in the population are close to various local minima. The global minimum is found by a stochastic search around the local minima. Local minima pose serious challenges to GAs since individuals near local minima have higher fitness and have a higher tendency to get reproduced. Thus the population might converge to a local minimum leading to stagnation of the evolutionary process. For the CACM algorithm, since each individual is updated based on neighboring individuals alone, information about the best solution found so far takes some time before it reaches all individuals. Since during this delay random changes can occur due to movement towards the best

neighbor, mutation, and change in the best individual itself, traps due to local minima are avoided.

V. Comparison of GA and ACM Algorithm

Figure 7 shows the performance of the ACM algorithm and a classical GA. A four dimensional Rastrigin function was used as the test function since it has multiple local minima. The GA used 40 bits accuracy per variable and a single population. It is seen that the CACM algorithm converges faster on the average than the classical GA.

For the CACM algorithm, since each individual is updated based on neighboring individuals alone, information about the best solution found so far takes some time before

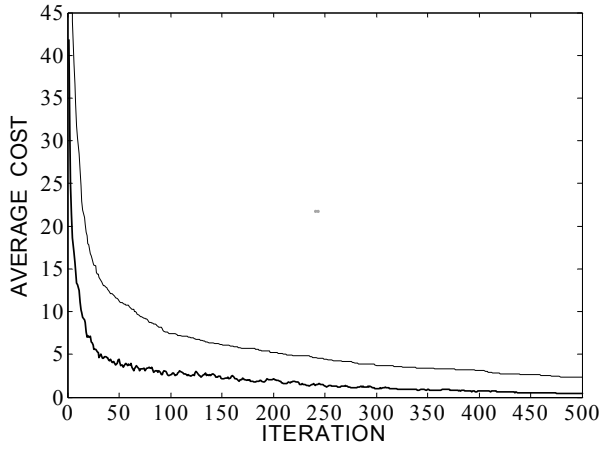

Fig. 7. Performance of GA (upper) and CACM algorithm (lower) for a test function with large numbers of local minima (Rastrigin function). The GA used 40 bits per variable. The average cost was taken over 50 independent runs.

it reaches all individuals. During this delay random changes occur in the population due to movement towards the best neighbor, mutation, and change in the best individual itself. This phenomenon helps avoid premature convergence and traps due to local minima. Also, use of local neighborhoods alone without use of global neighborhoods preserves the diversity of the population; global competition among individuals might result in the loss of less fit individuals resulting in loss of diversity. When using GAs, problems due to local minima can be alleviated by using multiple populations and sharing individuals periodically [10]. However, for the CACM algorithm this is achieved with less computation by updating each individual based only on its neighbors. Thus the more complex multipopulation approach and associated information sharing problems are avoided.

VI. Conclusion

In this paper a new computationally inexpensive alternative to GAs referred to as the CACM algorithm has been proposed. The CACM algorithm uses new operators that act directly on real vectors and generalize the classical operators in a geometrically intuitive way. This approach is more natural and avoids the disadvantages of binary encoding and decoding. In GAs two good solutions (parent solutions) are recombined to produce two offsprings that are fitter on the average. In the CACM algorithm, when an individual is updated it has a chance to learn from all previously updated individuals; this positive feedback effect leads to higher convergence rates compared to GAs. The effect of various parameters on the performance of the CACM algorithm was studied. Also, the CACM algorithm does not require global information sharing thus avoiding computational bottlenecks and facilitating parallel implementations. The new operators introduced in this paper can also be used in GAs to avoid binary coding.

VII. Acknowledgement

Parts of this effort were sponsored by the Department of the Navy, Office of Naval Research. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Office of Naval Research.

References

[1] F. J. Solis and J. B. Wets, "Minimization by random search techniques," *Mathematics of Operations Research*, vol. 6, pp. 19 -30, 1981.

[2] R. Axelrod, "The dissemination of culture: A model with local convergence and global polarization," *Journal of Conflict Resolution,* vol. 4, pp. 203-226, 1997.

[3] M. Dorigo, V. Maniezzo, and A. Colorni, "The ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics*, Part B, vol. 26, no. 1, pp. 29-41, 1996.

[4] S. Johnson, *Emergence: The Connected Lives of Ants, Brains, Cities, and Software*. New York: Scribner, 2001.

[5] Wang Lei and Wu Qidi, "Performance evaluation of ant system optimization processes," *Proceedings of the 4th World Congress on Intelligent Control and Automation*, vol. 3, 10-14, pp. 2546 – 2550, June 2002.

[6] Wang Lei, Xiao-Ping Wang and Wu Qidi, "Ant System algorithm based Rosenbrock function optimization in multi-dimension space," *Proceedings of the International Conference on Machine Learning and Cybernetics*, vol. 2, pp. 710 – 714, 4-5 Nov. 2002.

[7] Wang Lei and Wu Qidi, "Further example study on ant system algorithm based continuous space optimization*," Proceedings of the 4th World Congress on Intelligent Control and Automation*, 2002, vol. 3, pp. 2541 – 2545, 10-14 June 2002.

[8] J. Kennedy and R. Eberhart, *Swarm Intelligence.* Morgan Kaufmann Academic Press 2001.

[9] Xiaohui Hu, R. C. Eberhart and Yuhui Shi "Engineering optimization with particle swarm," Proceedings of the Swarm Intelligence Symposium, *IEEE 24-26,* pp. 53 – 57, April 2003.

[10] B. Carse, A. G. Pipe and O. Davies, "Parallel evolutionary learning of fuzzy rule bases using the island injection genetic algorithm," Systems Man and Cybernetics, *IEEE International Conference on Computational Cybernetics and Simulation*, vol. 4, 12-15, pp. 3692 – 3697, Oct. 1997.