A parallel implementation of particle swarm optimization using digital pheromones

Vijay Kalivarapu
Iowa State University, vkk2@iastate.edu

Jung Leng Foo
Iowa State University

Eliot H. Winer
Iowa State University, ewiner@iastate.edu

A parallel implementation of particle swarm optimization using digital pheromones

Abstract
A parallel implementation of Particle Swarm Optimization (PSO) using digital pheromones to coordinate the movements of the swarm within an n-dimensional design space is presented in this paper. Digital pheromones are models simulating real pheromones emitted by insects for communication to indicate a source of food or a nesting location. This principle of communication and organization between each insect in a swarm offers substantial improvement when integrated into a Particle Swarm Optimization algorithm. Digital swarms are used to search a design space with digital pheromones aiding communication within the swarm to improve search efficiency. With statistical analysis, the pheromone strength in a region of the design space is determined. The swarm then reacts accordingly based on the probability that this region may contain an optimum. When implemented in a parallel computing architecture, significant performance increases were observed. This paper presents the method development and results from several test cases.

Keywords
Virtual Reality Applications Center, algorithms, communication systems, computer simulation, parallel processing systems, probability, search engines, statistical methods, digital pheromones, Particle Swarm Optimization (PSO)

Disciplines
Computer-Aided Engineering and Design | Mechanical Engineering

Comments
This is a conference proceeding from *Collection of Technical Papers - 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference* (2006): AIAA 2006-6908, doi: 10.2514/6.2006-6908. Posted with permission.

11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference
6 - 8 September 2006, Portsmouth, Virginia

AIAA 2006-6908

A Parallel Implementation of Particle Swarm Optimization Using Digital Pheromones

Vijay Kalivarapu[*], Jung Leng Foo[†] and Eliot Winer[‡]
Iowa State University, Ames, IA, 50011, USA

A parallel implementation of Particle Swarm Optimization (PSO) using digital pheromones to coordinate the movements of the swarm within an n-dimensional design space is presented in this paper. Digital pheromones are models simulating real pheromones emitted by insects for communication to indicate a source of food or a nesting location. This principle of communication and organization between each insect in a swarm offers substantial improvement when integrated into a Particle Swarm Optimization algorithm. Digital swarms are used to search a design space with digital pheromones aiding communication within the swarm to improve search efficiency. With statistical analysis, the pheromone strength in a region of the design space is determined. The swarm then reacts accordingly based on the probability that this region may contain an optimum. When implemented in a parallel computing architecture, significant performance increases were observed. This paper presents the method development and results from several test cases.

I. Introduction

Heuristic optimization techniques such as Genetic Algorithms (GA) and Simulated Annealing (SA) are capable of exhaustively investigating design spaces to locate global optimal design points. The probabilistic nature of these heuristic methods gives distinct advantages over deterministic methods in finding a global optimum and hence are popular choices in solving multi-disciplinary optimization problems. A drawback to these methods is their computational expense and complexity.

PSO [1, 2] is a derivative free, population based heuristic method similar to GA and SA. Its inherent advantage is its simplicity in implementation due to a small number of parameters to adjust [3, 4]. In a traditional PSO, an initial randomly generated population swarm (a collection of particles) propagates towards the global optimum over a series of iterations. Each particle in the swarm explores the design space based on the information provided by previous best particles. A basic PSO uses this information to generate a velocity vector indicating a search direction towards a promising design point, and updates the locations of the particles. PSO is one of the recent additions to the global search methods [5], and the fact that it is evolutionary in nature makes it particularly suitable for a parallel implementation.

This paper focuses on the implementation of digital pheromones within a parallel PSO. Coupled with a statistical analysis on the pheromones, an efficient moveset is generated to update the search direction of each particle. Previous work by the authors [6] on implementing digital pheromones within PSO proved successful in decreasing the: a) number of function evaluations, b) number of iterations, c) solution time, and d) solution consistency. When implemented in parallel, the method produced further performance improvement when compared to both serial implementation as well as traditional PSO. This method is tested with a series of n-dimensional problems and the results are presented.

[*] Research Assistant, Department of Mechanical Engineering, Human Computer Interaction, Virtual Reality Applications Center, 2274 Howe Hall, Iowa State University, Ames, IA, 50011, USA, Student Member.
[†] Research Assistant, Department of Mechanical Engineering, Human Computer Interaction, Virtual Reality Applications Center, 2274 Howe Hall, Iowa State University, Ames, IA, 50011, USA, Student Member.
[‡] Assistant Professor, Department of Mechanical Engineering, Human Computer Interaction, Virtual Reality Applications Center, 2274 Howe Hall, Iowa State University, Ames, IA, 50011, USA, Member.

II. Background

A. Particle Swarm Optimization

PSO is a population based zero-order optimization method that portrays several evolutionary algorithm characteristics similar to Genetic Algorithms (GA) and Simulated Annealing (SA) – a) Initialization with a population of random solutions, b) Design space search for optimum through updating generations and c) Update based on previous generations [7]. The success of the algorithm has brought substantial attention among the research community in the recent past [8, 9]. The working of the algorithm is based on a simplified social model similar to the swarming behavior exhibited by a swarm of bees or a flock of birds. In this analogy, a bee (particle) uses its own memory and the behavior of the rest of the swarm to determine the suitable location of food (global optimum). The algorithm iteratively updates the direction of the swarm movement toward the global optimum. The mathematical formulation of the method is given in Equations (1) and (2).

$$V_{i+1} = w_i * V_i + c_1 * rand() * (pBest_i[] - X_i[]) + c_2 * rand() * (gBest[] - X_i[]) \tag{1}$$

$$X_{i+1} = X_i + V_{i+1} \tag{2}$$

$$w_{i+1} = w_i * \lambda_w \tag{3}$$

Equation (1), represents the velocity vector update of a traditional PSO method where *rand()* is a random number between 0 and 1. c_1 and c_2 are confidence parameters. *'pBest'* represents the best position attained by the swarm in the current iteration and *'gBest'* represents the best position attained by the swarm in the entire iteration history. w_i is called as the inertia weight [10, 11] and decreases in every subsequent iteration by a factor of λ_w, as represented in Equation (3). Equation (2) denotes the updated swarm location in the design space.

In addition to the originally developed PSO algorithm, significant enhancements have been proposed such as: a) mutation factors for better design space exploration [12, 13], b) methods for constraint handling [14, 15], c) parallel implementation [16, 17], d) methods for solving multi-objective optimization problems [18], e) methods for solving mixed discrete, integer and continuous variables [19].

B. PSO and Digital Pheromones

Pheromones are chemical scents produced by insects to communicate with each other to find a suitable food source, nesting location, etc. The stronger the pheromone, the more the insects are attracted to the path. A digital pheromone is analogous to an insect generated pheromone in that they are the markers to determine whether or not an area is promising for further investigation. One of the well-known applications of digital pheromones is its use in the automatic adaptive swarm management of Unmanned Aerial Vehicles (UAVs) [20, 21]. In this research, the UAVs are automatically guided towards a specific zone or target through releasing digital pheromones in a virtual environment, thereby reducing the requirement of humans physically controlling from ground stations. Other applications of digital pheromones include ant colony optimization for solving minimum cost paths in graphs [22, 23, 24], solving network communication problems [25]. The concept of digital pheromones is considerably new [26] and has not yet been explored to its full potential for investigating n-dimensional design spaces for locating an optimum.

In a basic PSO algorithm, the swarm movement obtains design space information from only two components – *pBest* and *gBest*. When coupled with an additional pheromone component, the swarm is essentially presented with more information for design space exploration that has a potential to reach the global optimum faster. This idea was previously tested and implemented by the authors with a substantial amount of success.

C. Parallelization

The primary requirement for parallelization is the ability of the method to decompose into segments for multi-processor operation. In addition, the two highly desirable characteristics for parallelization are: a) scalability – the ability to adapt to any number of processors with no/minimal changes and b) processor load balancing – use of the available number of processors to the full extent without any processor substantially running idle. Population based optimization methods such as GA and PSO are a natural fit for parallelization because the method parameters do not limit the number of processors that can be used for solving the problem.

Parallelization can be synchronous or asynchronous. Synchronous parallelization facilitates a step wise parallel execution of tasks. Coarse decomposition schemes are examples of synchronous parallelization where each processor has its own swarm exploring the design space. Solutions obtained from different processors are synchronized and gathered on a common processor (usually, the root processor) to evaluate the final global optimum. This is achieved through the use of a barrier function in the Message Passing Interface (MPI), the most commonly used interface for parallel programming. Asynchronous parallelization is the dividing of a sequential algorithm into autonomous tasks each of which can be carried out on different processors. Dependencies among the tasks are modeled by message passing or through shared memory [27], depending upon the hardware configuration.

The research presented in this paper explores the use of digital pheromones in PSO through implementing two parallelization schemes: a) synchronous coarse grain parallel implementation, and b) synchronous shared pheromone parallelization method using MPI (MPICH implementation) on a distributed memory system over a Myrinet connection.

III. Methodology

A. Overview of serial implementation of digital pheromones in PSO

Figure 1 summarizes the procedure for PSO, with steps involving digital pheromones highlighted in blue.

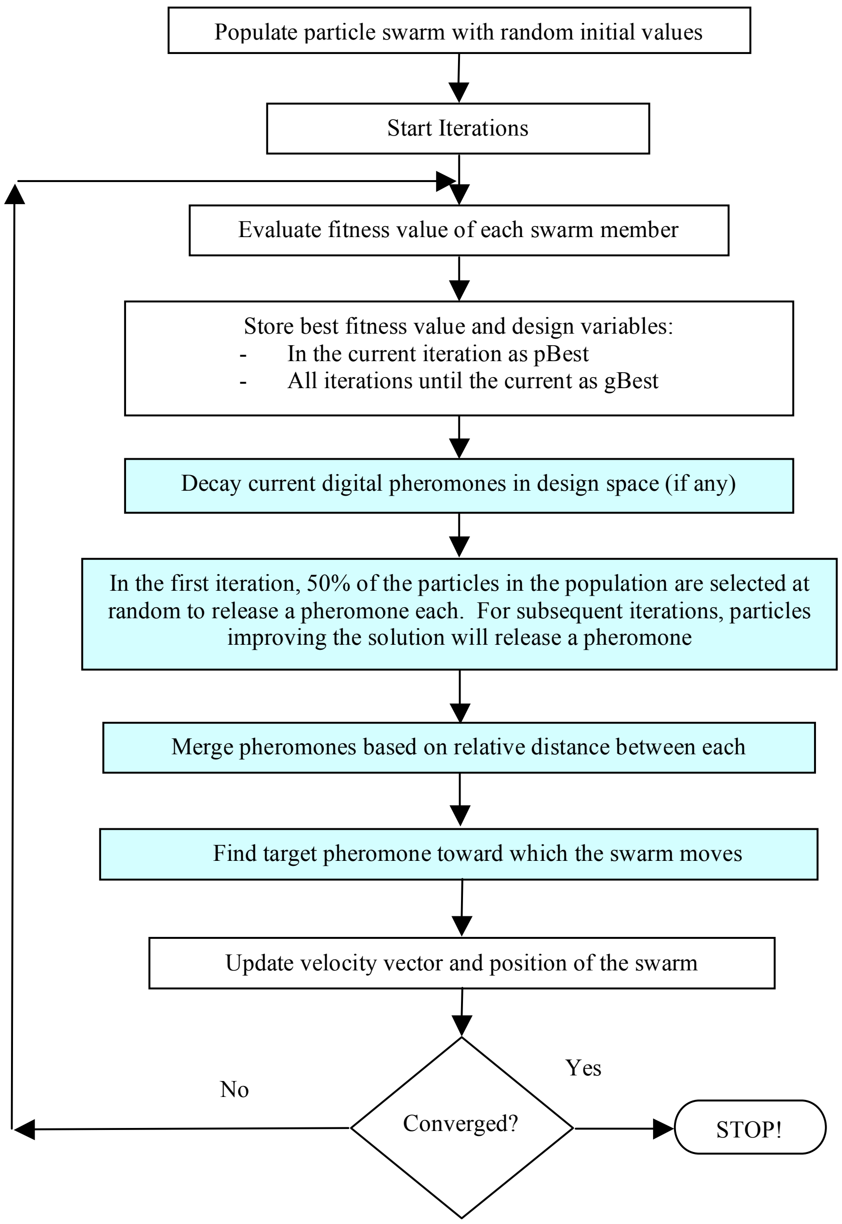
Populate particle swarm with random initial values
Start Iterations
Evaluate fitness value of each swarm member
Store best fitness value and design variables:
- In the current iteration as pBest
- All iterations until the current as gBest
Decay current digital pheromones in design space (if any)
In the first iteration, 50% of the particles in the population are selected at random to release a pheromone each. For subsequent iterations, particles improving the solution will release a pheromone
Merge pheromones based on relative distance between each
Find target pheromone toward which the swarm moves
Update velocity vector and position of the swarm
No
Yes
Converged?
STOP!

Figure 1. Flowchart of Particle Swarm Optimization with Digital Pheromones Algorithm.

The initialization of pheromone-based PSO is similar to a basic PSO except that a selected percentage of particles from the swarm that find a better solution release pheromones within the design space in the first iteration. For subsequent iterations, each swarm member that finds a better objective function releases a pheromone. Pheromones (from current as well as past iterations) that are close to each other in terms of design variable values are merged into a new pheromone location. This effectively creates a pheromone pattern across the design space while still keeping the number of pheromones manageable. Based on the pheromone level and its position relative to a particle, a probability is then used in a ranking process to select a target pheromone for each particle in the swarm. The target position for each particle will be an additional component of the velocity vector update in addition to *pBest* and *gBest*. Following this, the objective value for each particle is recalculated and the entire process continues until the convergence criteria is satisfied.

Digital Pheromones and Merging

In order to populate the design space with a user-defined initial set of digital pheromones (default is 50% of the particles in the population) are randomly selected to release pheromones, regardless of the objective function value. This is done so as to ensure a good design space exploration by the particle swarm in the initial stages of the optimization process. For subsequent iterations, the objective function value for each particle in the population is evaluated and only particles finding an improvement in the objective function value will release a pheromone. Any newly released pheromone is assigned a level *P*, with a value of 1.0. Just as natural pheromones produced by insects decay in time, a user defined decay rate, λ_P, defaulting to 0.995, is assigned to the pheromones released by the particle swarm. Digital pheromones are decayed as the iterations progress forward to allow the swarm to propagate toward a better design point instead of getting attracted to an older pheromone which may not be a good design point.

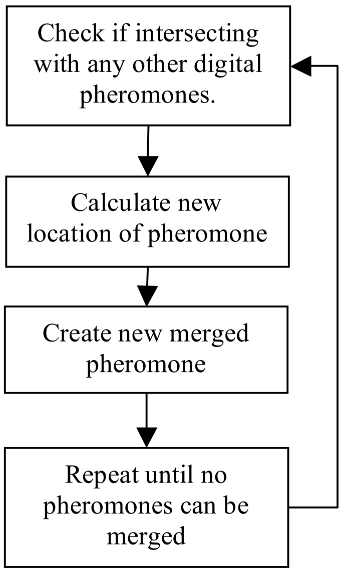
Check if intersecting with any other digital pheromones.
Calculate new location of pheromone
Create new merged pheromone
Repeat until no pheromones can be merged

Figure 2. Flowchart of digital pheromones merging process.

Every particle that finds a solution improvement releases a pheromone potentially making the pheromone pool unmanageably large. Therefore, an additional step to reduce them to a manageable number, yet retaining the functionality, is implemented. Pheromones that are closely packed within a small region of the design space are merged together. To check for merging, each pheromone is associated with an additional property called 'Radius of Influence' (*ROI*). For each design variable of a pheromone, an *ROI* is computed and stored. The value of this *ROI* is a function of the pheromone level and the bounds of the design variables. Any two pheromones for a design variable less than the sum of the ROIs are merged into one. This is analogous to two spheres merging into one if the distance between them is less than the sum of their radii. A resultant pheromone level is then computed for the merged pheromones. Through this approach, regions of the design space with stronger resultant pheromone levels will attract more particles and therefore, pheromones that are closely packed would indicate a high chance of optimality. Also similar to the pheromone level decay, the *ROI* also has its own decay factor, λ_{ROI}, whose value is set equal to

λ_P as a default. This is to ensure that both the pheromone levels and the radius of influence decay at the same rate. Figure 2 illustrates the pheromone merging process.

Attraction to a Target Digital Pheromone

With numerous digital pheromones generated within the design space, a swarm member needs to identify which pheromone it will be attracted too most. The criteria for generating this target pheromone are: a) small magnitude of distance from the particle and b) high pheromone level. To rank which digital pheromone from the pheromone pool fits this criteria, a target pheromone attraction factor *P'* is computed. The value of *P'* is a product of the normalized distance between that pheromone and the particle, and its pheromone level. Also, the attraction factor must increase when the pheromones are closer to the particles. Therefore, the attraction factor is computed as shown in Equation (5). Equation (6) computes the distance between the pheromone and each particle in the swarm. Figure 3 shows an example scenario of a particle being attracted to a target pheromone.

$$P' = (1-d)P \tag{5}$$

$$d = \sqrt{\sum_{1}^{k}\left(\frac{Xp_k - X_k}{range_k}\right)^2}, k = 1:n \quad \#\text{ of design variables} \tag{6}$$

Xp – Location of pheromone

X – Location of particle

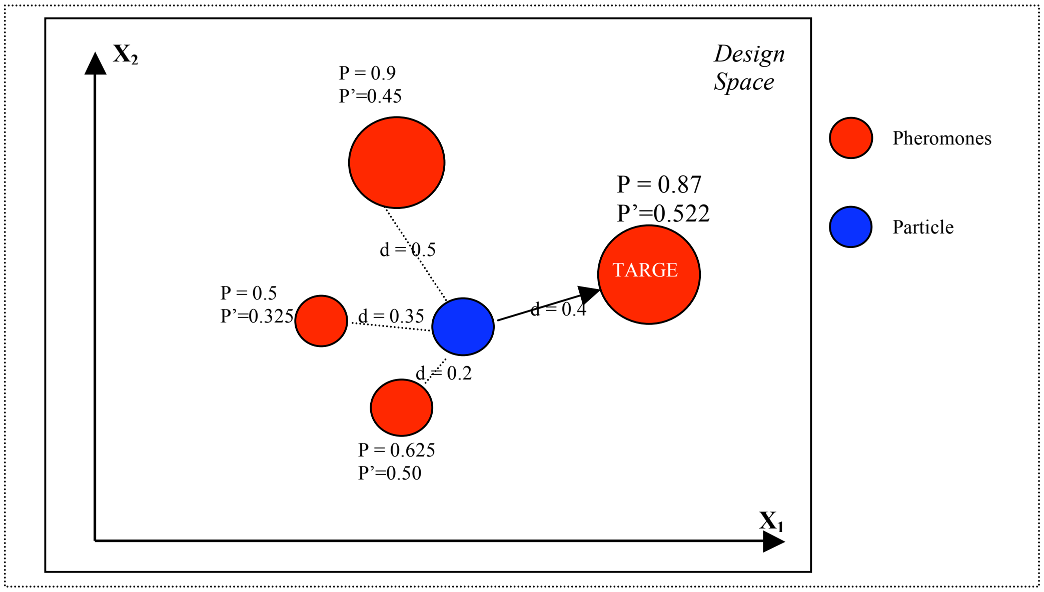

Figure 3. Illustration of target pheromone selection.

In the figure, the particle will be more attracted to a pheromone with a higher *P'* value, as opposed to pheromones that are closer but with a lower *P'* value.

Velocity Vector Update

The velocity vector update implements the pheromone component as a third term in addition to the *pBest* and *gBest* components in a traditional PSO. This is shown in Equation (7).

$$V_{i+1} = w_i * V_i + c_1 * rand() * (pBest_i[] - X_i[]) + c_2 * rand() * (gBest[] - X_i[]) + c_3 * rand() * (Target[] - X_i[]) \tag{7}$$

c_3 is the confidence parameter for the pheromone component of the velocity vector, and is set to be larger than c_1 and c_2. This is done in order to increase the influence of pheromones in the velocity vector. From experimentation, it was found that a default value of 10.0 sufficed for most problems.

Move Limits, *ML*

The additional pheromone term in the velocity vector update, especially with a large c_3 value, can considerably increase the computed velocity. To avoid this value from becoming unmanageably large, a move limit is imposed. The move limit is set to an initial value and reduced gradually as the iterations progress forward. This ensures a fair amount of freedom in exploration in the beginning and as the method approaches a solution, a smaller move limit exploits the current design point of a particle for a more constrained search towards an optimum. Although this is a user defined parameter, an initial set value of 10% of the design space for the move limit showed good performance characteristics. A default decay factor, λ_{ML} of value 0.995 was used.

B. Synchronous Coarse Grain Parallel Implementation

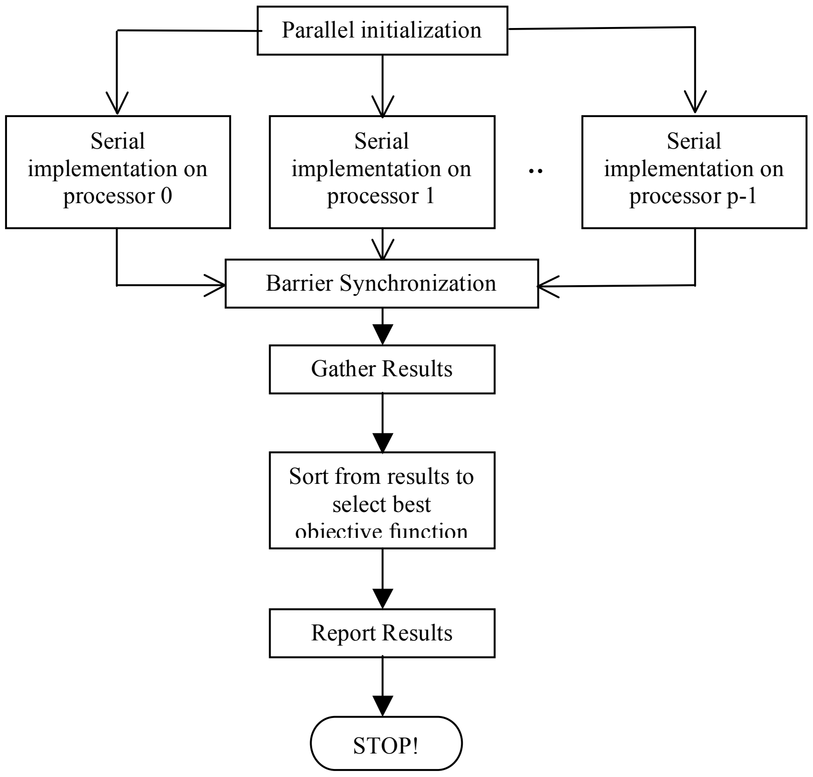
Parallel initialization
Serial implementation on processor 0
Serial implementation on processor 1
..
Serial implementation on processor p-1
Barrier Synchronization
Gather Results
Sort from results to select best objective function
Report Results
STOP!

Figure 4. Schematic of synchronous coarse grain parallel implementation of digital pheromones in PSO

The schematic shown in Figure 4, details the various steps involved in the coarse grain synchronous parallelization scheme employed. In this decomposition approach, each processor proceeds with its own copy of the serial PSO code with its own randomized population swarm. Upon calculating the velocity vector and particle position update, each processor checks for its own convergence criteria and then arrives at the optimal point. Using barrier synchronization, optimal points from all the processors are gathered on the root processor and the overall best objective function value and its corresponding design variable values are sorted and selected.

The larger the number of processors used the greater the chances of finding the global optimum. In addition, data communication between the processors takes place only toward the end when gathering results from each processor, avoiding network latencies – the primary bottleneck in parallelization. While this is a desirable feature in the performed coarse grain parallelization, it is also true that each processor is unaware of the progress of each other processor. A potential good pheromone location pointed out by processor ‘A’ is completely obscure to the particle swarm in processor ‘B’. A communication of some sort between the processors during an iteration could improve the quality of the search direction, and hence the chances of finding the global optimum. This idea is explored through the implementation of shared pheromones across processors.

C. Shared Pheromone Parallel Implementation

Figure 5 shows the schematic of various steps involved in the method.

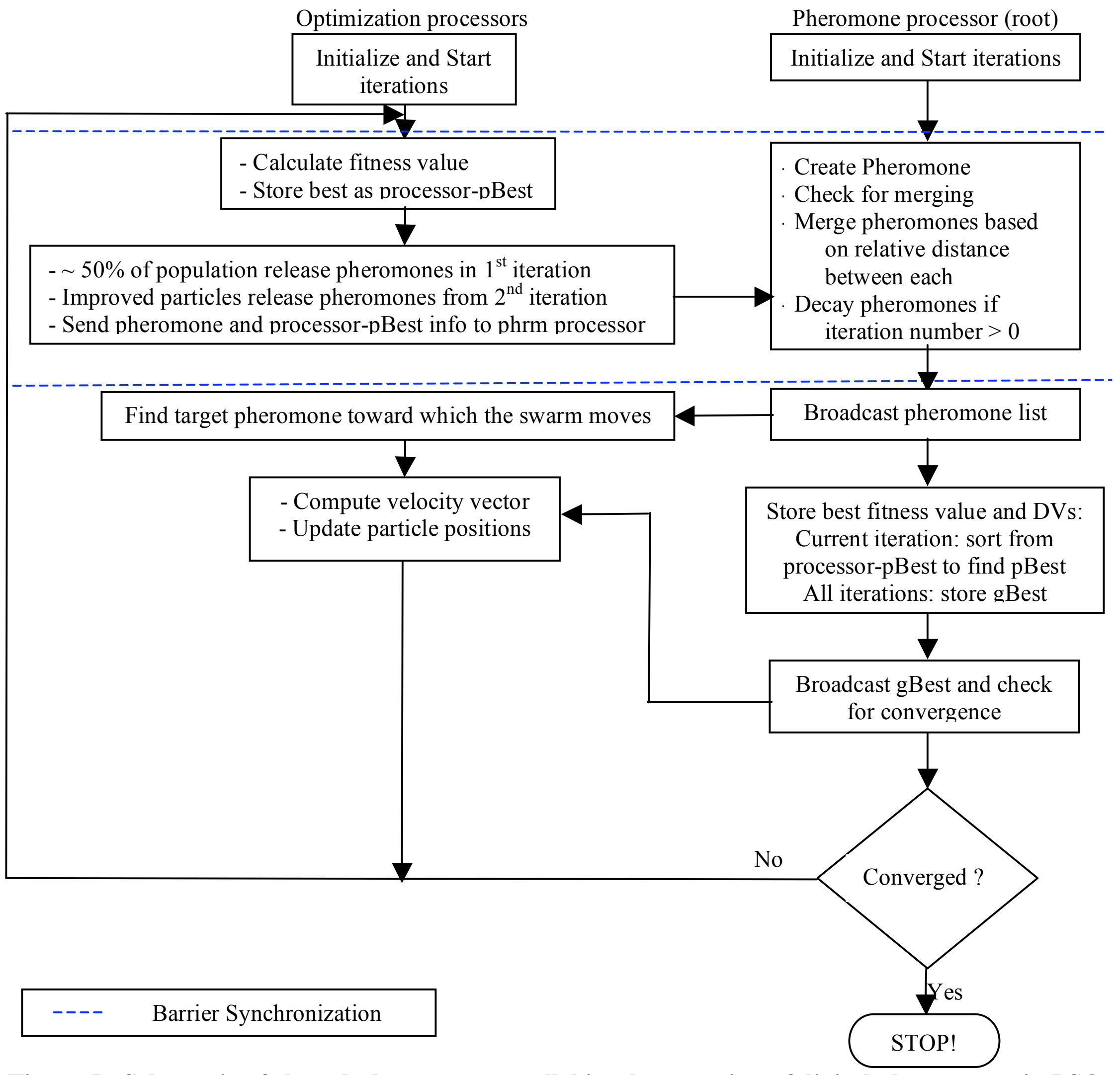
Optimization processors
Pheromone processor (root)
Initialize and Start iterations
Initialize and Start iterations
- Calculate fitness value
- Store best as processor-pBest
- ~ 50% of population release pheromones in 1st iteration
- Improved particles release pheromones from 2nd iteration
- Send pheromone and processor-pBest info to phrm processor
· Create Pheromone
· Check for merging
· Merge pheromones based on relative distance between each
· Decay pheromones if iteration number > 0
Find target pheromone toward which the swarm moves
Broadcast pheromone list
- Compute velocity vector
- Update particle positions
Store best fitness value and DVs: Current iteration: sort from processor-pBest to find pBest All iterations: store gBest
Broadcast gBest and check for convergence
No
Converged ?
Yes
STOP!
Barrier Synchronization

Figure 5. Schematic of shared pheromone parallel implementation of digital pheromones in PSO

In this method, the available processors are divided into two categories based on their assigned functions, and are designated as the optimization processor(s) and a pheromone processor. Each of the optimization processors perform: 1) random population swarm generation, 2) fitness value evaluation, 3) pheromone release, 4) calculation of target pheromones, 5) calculation of velocity vector, and 6) particle position update. However, access to the common pheromone list, *pBest* and *gBest* is obtained only through communication with the pheromone processor. The optimization processors are scalable to any number while the pheromone computations are currently performed on a single processor. This pheromone computations are:

a. Create, merge and decay pheromones as and when released by the optimization processors.
b. Maintain a global pheromone list made available to the optimization processors through a broadcast.
c. Gather *processor-pBest* (*pBest* on each processor) values from the optimization processors and determine the overall *pBest* and *gBest*.
d. Perform convergence check and broadcast *pBest*, *gBest* to the optimization processors.

This scheme ensures that a promising design point located by an optimization processor is transparent to the rest of the processors through communication with the pheromone processor. Since the pheromone and the optimization processors perform two distinctly different tasks, they need to be synchronized when accessing and sending information such as the global pheromone list for target pheromone calculation, *pBest* and *gBest* values for velocity vector calculation and proceeding with subsequent iterations. This is achieved through the use of a barrier synchronization, depicted as blue dotted lines in Figure 5. Since *pBest* and *gBest* are available on the pheromone processor, the convergence check is performed on the pheromone processor and the outcome of whether or not subsequent iterations are necessary is broadcast to the optimization processors.

Pheromones are created and the global list is updated every iteration on the pheromone processor. As the iterations progress toward convergence, the pheromone activity increases substantially. Given that the third component of the velocity vector is still active and with a default c_3 value of 10.0, the magnitude of the velocity vector becomes substantially large thereby causing the particle position update location to move away from the global optimum instead of converging towards it. Therefore, the value of c_3 was decayed as the number of iterations increased. This serves two purposes – 1) a high value of c_3 provides a greater spread over the design space in the initial stages of optimization, causing a better design space exploration 2) a low value of c_3, and hence a lower velocity vector magnitude, towards the close of convergence reduces the spread of the particle swarm thereby propagating toward the global optimum instead of moving away from it. Although a default decay value of 0.95 provided best results when solving the test cases, the factor can be user defined depending upon the problem parameters.

IV. Results

A. Test Cases

Three unconstrained problems of varying dimensionality are used as test cases to evaluate the performance of PSO with digital pheromones in a parallel computing environment. The swarm size for each test case was experimentally determined to be 10 times the number of design variables. This parameter is user-defined and can be adjusted to any value. The test cases used are described below.

1. Six-hump camelback function

This is a multi-modal optimization problem with two design variables with six local minima, two of which are global minima. The optimization problem statement is:

Minimize:

$$F(x_1,x_2) = \left(4 - 2.1x_1 + \frac{x_1^{\ 4}}{3}\right)x_1^{\ 2} + x_1x_2 + \left(-4 + 4x_2^{\ 2}\right)x_2^{\ 2}$$

$$-3 \le x_1 \le 3 \quad \text{and} \quad -2 \le x_2 \le 2$$

Published solution:

$$F_{\min}(x_1,x_2) = -1.031628$$

$$(x_1,x_2) = (0.0898,-0.7126), \quad (-0.0898,0.7126)$$

This problem was solved using a swarm size of 20 particles per processor on both coarse grain implementation ands shared pheromone implementation. Although this is just a two design variable problem, it was chosen to evaluate the scalability characteristics of the developed method. The solutions obtained from the parallelization were tested for 1% accuracy with the published solution values.

2. Ackley's path

The problem statement for solving the Ackley's path function is as follows:

Minimize:

$$F(x) = -a \cdot e^{-b\cdot\sqrt{\frac{\sum_i^5 x_i^2}{5}}} - e^{\frac{\sum_i^5 \cos(c\cdot x_i)}{5}} + a + e^1$$

$$a = 20; \quad b = 0.2; \quad c = 2 \cdot PI; \quad i = 1:n;$$

$$-32.768 \le x_i \le 32.768$$

Published solution:

$$F_{\min}(x) = 0.0, \quad x_i = 0.0$$

The problem is scalable to any number of dimensions. Figure 6 illustrates a 2D Ackley's path function. The figure on the left implicates that the function is uni-modal but a closer look at the design space shows that it is a multi-modal problem (figure on the right). Two cases were chosen to test the developed method – a 5 design variable problem with a swarm size of 50 per processor and a 20 design variable problem with a swarm size of 200 per processor. The published minimum value for Ackley's path function is 0.0. Therefore, a percentage accuracy cannot be used for evaluating the solutions obtained from parallelization. Hence, the solutions obtained were regarded as accurate if the minimum objective function values were less than 0.1.

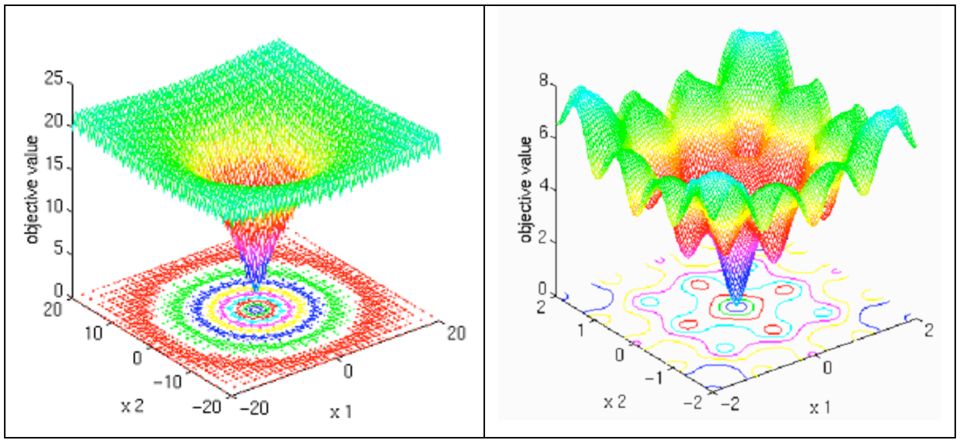

Figure 6. Illustration of a two-dimensional Ackley's Path function. Bounds of design space: Left image [-20, 20], Right image [-2, 2].

The following parameters were used for solving the optimization problem for all test cases:

Table 1. Default values of parameters used for test cases

Parameter	Default value
c_1	2.0
c_2	2.0
c_3	10.0
Size of initial move limit, *ML*	0.1*range of design variables
Move limit decay factor, λ_{ML}	0.995
Inertia weight initial value, *w*	1.0
Inertia weight decay factor, λ_w	0.995
Pheromone level decay factor, λ_P	0.995
Pheromone radius of influence decay factor, λ_{ROI}	0.995
c_3 decay (shared pheromone implementation only)	0.5% decrease per iteration

Both the parallel implementations were coded in C++ and the results were noted from performing 100 runs for each test case. Due to the unavailability of identical computing platforms, the evaluation of coarse grain and shared pheromone parallelization was made on two different systems and the computing environment specifications are described below in table 2.

Table 2. Computing platforms for Coarse Grain and Shared Pheromone parallelization

Coarse Grain Parallelization:	**Shared Pheromone Parallelization:**
- Intel Pentium 4, Xeon-MP processor, 2.8 GHz - Red Hat Enterprise Linux 4 - 2GB Memory/node with 2 CPUs per node	- Intel Pentium 4, Xeon, 3.20 GHz (hyper-threaded) - Red Hat Enterprise Linux 4 - 2GB system Memory

B. Results from the two schemes of parallelization

1. Six-hump Camel Back Function

Table 3 summarizes the results from solving the six-hump camelback function using coarse grain approach.

Table 3. Summary of results for Six-hump Camel Back Function using Coarse Grain approach

	Number of proc.	Solution Accuracy	Iteration			Duration (seconds)			Seconds/ Iteration
			Average	Min	Max	Average	Min	Max	
Basic PSO	1	100%	96.14	54	123	1.9244	1.0798	2.4596	0.020017
	4	99.75%	98.09	52	125	2.0096	1.0598	2.5392	0.020018
	8	99.125%	97.56	51	129	2.0069	1.0798	2.5796	0.020015
PSO with Digital Pheromones	1	100%	66.03	52	79	1.3256	1.0408	1.5847	0.020076
	4	100%	64.21	51	81	1.3019	1.0888	1.6657	0.020081
	8	100%	64.92	51	83	1.3347	1.0628	3.4915	0.020359

The solution accuracy for both basic and pheromone implementation were fairly equal regardless of the number of processors used. However, the average number of iterations for the pheromone implementation was considerably lower compared to a basic PSO method. The overhead due to pheromones was not substantial as evident from the time taken per iteration in both the basic and pheromone implementation. There is no significant difference in the average performance data from using various numbers of processors, since each processor is performing the same functions in the same code. Given that the problem is two-dimensional, savings due to lesser number of iterations through using digital pheromones was not readily evident.

Table 4. Summary of results for Six-hump Camel Back Function using Shared Pheromones

	Number of proc.	Solution Accuracy	Iteration			Duration (seconds)			Seconds/ Iteration
			Average	Min	Max	Average	Min	Max	
PSO with Digital Pheromones	2	100%	111.04	57	187	2.7851	1.3760	4.6112	0.025081
	4	100%	94.66	54	175	2.9530	1.5833	5.3269	0.031196
	8	100%	88.44	57	140	4.3551	2.6347	6.8219	0.049243

Table 4 shows the results obtained from the shared pheromone implementation. The solution accuracy is the same as that used for the coarse grain implementation. At least one optimization processor and one pheromone processor are required to solve the problem thereby making the minimum number of processors two. Although the difference in the number of seconds per iteration can be partly attributed to different computing systems used, the additional processing involved in the information flow between the pheromone and the optimization processors is an additional cause for the increased time per iteration.

2. Ackley's Path – Five Design Variable

Table 5 summarizes the results of solving the Ackley's 5 design variable problem using the coarse grain approach.

Table 5. Summary of results for Ackley's Path – Five Design Variable using Coarse Grain approach

	Number of proc.	Solution Accuracy	Iteration			Duration (seconds)			Seconds/ Iteration
			Average	Min	Max	Average	Min	Max	
Basic PSO	1	82.00%	176.69	64	237	8.841	3.199	11.879	0.050035
	4	78.00%	177.07	149	223	8.920	7.599	11.179	0.050037
	8	77.38%	176.74	142	244	8.941	7.599	12.198	0.050037
PSO with Digital Pheromones	1	100.00%	189.56	169	207	9.834	8.730	10.828	0.051879
	4	99.75%	187.79	146	215	9.766	8.693	11.131	0.051785
	8	100.00%	187.78	157	210	9.813	8.563	10.869	0.051802

The results indicate that the solution accuracy is superior to the solutions obtained from basic PSO although there is an increase in the number of seconds per iteration in the third decimal place. This increase can be attributed to the overhead due to the pheromone activity before reaching the solution. The solution accuracy from using the shared pheromone approach, as shown in table 6, is almost 100% in all the cases of using 2, 4 and 8 processors although the number of seconds per iteration increased. This increase is partly attributed to the increased information flow between the optimization and pheromone processors. An additional reason is due to the barrier synchronization that causes all processors to wait before proceeding to the subsequent iteration.

Table 6. Summary of results for Ackley's Path – Five Design Variable using Shared Pheromones

	Number of proc.	Solution Accuracy	Iteration			Duration (seconds)			Seconds/ Iteration
			Average	Min	Max	Average	Min	Max	
PSO with Digital Pheromones	2	100%	217.96	162	349	20.236	14.813	32.126	0.092843
	4	98%	234.76	173	312	21.818	16.174	29.154	0.092938
	8	100%	219.84	59	303	21.739	5.806	29.819	0.098884

3. Ackley's Path – 20 Design Variable

This problem is the same as test case #2, but with 20 design variables. The results for the coarse grain approach are summarized in table 7.

Table 7. Summary of results for Ackley's Path – 20 Design Variable

	Number of proc.	Solution Accuracy	Iteration			Duration (seconds)			Seconds/ Iteration
			Average	Min	Max	Average	Min	Max	
Basic PSO	1	0.00%	236.42	202	278	48.280	41.223	56.785	0.204213
	4	0.00%	233.27	188	305	47.999	41.838	62.270	0.204144
	8	0.00 %	233.89	180	325	48.277	41.850	66.336	0.204144
PSO with Digital Pheromones	1	85.00%	191.57	157	224	42.535	34.819	50.419	0.222089
	4	83.25%	190.21	152	226	42.569	34.581	50.698	0.222089
	8	84.38%	191.12	154	226	42.727	35.037	50.969	0.222231

The advantage of using digital pheromones with PSO is also evident in this test case. While the basic PSO failed to find a solution within the pre-set accuracy tolerance, the PSO with digital pheromones solved the problem about 83% of the time. The number of iterations decreased to about 190 while the high average for basic PSO was about 233 (without locating the solution). The overhead to attain this accuracy level is approximately an additional 0.02 seconds per iteration. Table 8 represents the results from solving the 20 design variable problem using the shared pheromone approach. Within the pre-set tolerance limits, the accuracy attained was 100% with an overhead of 0.1 seconds more per iteration when compared to PSO with and without pheromones.

Table 8. Summary of results for Ackley's Path – 20 Design Variable using Shared Pheromones

	Number of proc.	Solution Accuracy	Iteration			Duration (seconds)			Seconds/ Iteration
			Average	Min	Max	Average	Min	Max	
PSO with Digital Pheromones	2	100%	216.16	179	260	63.392	52.492	76.288	0.293264
	4	100%	197.42	165	226	60.621	50.608	69.406	0.307067
	8	100%	186.10	163	215	61.054	53.545	70.461	0.328073

C. Varying Objective Functions Computation Time

The test cases thus far are academic in nature and they do not properly scale to the type of problems solved in industrial settings. Evaluating objective functions with longer computational times was performed in an attempt to model these types of problems. This situation was simulated by adding sleep times when evaluating objective functions. For coarse grain parallelization, three different sleep times 5, 10 and 20 milliseconds were introduced while calculating the objective function. The results are summarized in table 9. The results show that there is a significant improvement in solution times as the lengths of a single objective function evaluation were increased. This means that coarse grain parallel PSO with digital pheromones provides significant time savings when solving problems with complex objective functions that take considerable amount of computing time.

Table 9. Summary of results for Ackley's Path function (20 Design variables) with variable sleep times.

Sleep time (milliseconds)	Duration – Basic (seconds)	Duration – Pheromones (seconds)	% improve
0.0	48.28006	42.53528	+11.915
5.0	322.49980	273.61885	+15.157
10.0	578.08528	457.97666	+20.777
20.0	1032.26224	829.54794	+19.638

Figure 7 compares the performance of basic PSO and PSO with digital pheromones. When applied to realistic objective functions, the benefits of using PSO with digital pheromones will be more noticeable. Since coarse grain parallelization is similar on multiple processors, results from using only 1 processor are presented in Table 9.

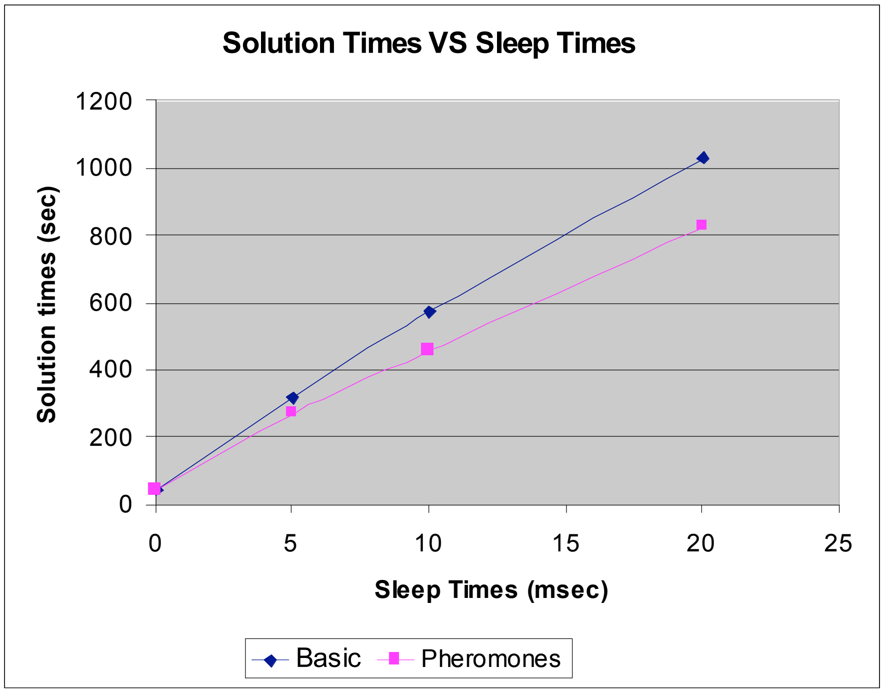

Figure 7. Plot of resulting time to solve for an optimum, with varying 'sleep times' are introduced. Solved using one processor.

In addition to the above, the Ackley's 20 design variable problem was solved using the shared pheromone parallelization scheme with a 10 millisecond sleep time, and is portrayed in figure 8. The figure shows that, when sleep time is introduced, the solution time decreased with increased number of processors. This demonstrates that the shared pheromone approach will improve the solution times for a problem with longer objective function evaluation times.

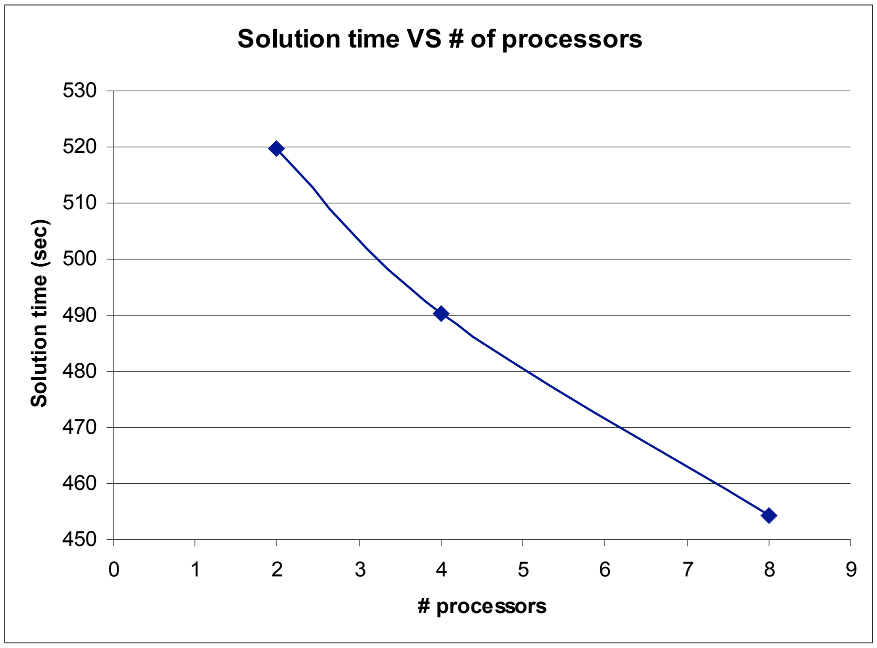

Figure 8. Plot of resulting time to solve for an optimum using Parallel PSO with shared pheromones, varying the number of processors.

V. Conclusion

This paper presents two different parallelization schemes for implementing digital pheromones in PSO. From the test results presented, it is evident that the developed methods consistently found the global optimum solution. The solutions showed improved accuracy, especially when the complexity of the problem increases. The substantial increase in performance can be attributed to the increased transparency of the solution progress in each processor. That is, through accessing the global pheromone list in each iteration, the particle swarm is directed to a better location in the design space. Having common overall *pBest* and *gBest* values also improved the quality of the solution as opposed to each processor solving the PSO without any sort of communication as in the case of coarse grain parallelization.

The scalability issue was addressed by not limiting the number of processors that can be used for solving the optimization problem, with the exception that at least two processors are required for the shared pheromone parallelization scheme. The advantages of the proposed methods became quite significant when the complexity of the objective function increased. This was demonstrated when artificial complexities were simulated, by adding ‘sleep times’ during objective function evaluation. PSO with digital pheromones performed significantly faster, while also providing more accurate solutions.

References

[1] Kennedy, J., and Eberhart, R. C., "Particle Swarm Optimization", *Proceedings of the 1995 IEEE International Conference on Neural Networks*, Vol. 4, Inst. of Electrical and Electronics Engineers, Piscataway, NJ, 1995, pp. 1942-1948.

[2] Eberhart, R. C., and Kennedy, J., "A New Optimizer Using Particle Swarm Theory", *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, Inst. of Electrical and Electronics Engineers, Piscataway, NJ, 1995, pp. 39-43.

[3] J.F. Schutte. Particle swarms in sizing and global optimization. Master’s thesis, University of Pretoria, Department of Mechanical Engineering, 2001.

[4] A. Carlisle and G. Dozier. An off-the-shelf pso. In Proceedings of the Workshop on Particle Swarm Optimization, 2001, Indianapolis.

[5] Russell C. Eberhart and Yuhui Shi, “Particle swarm optimization: Developments, applications, and resources”, *In Proceedings of the 2001 Congress on Evolutionary Computation 2001*, 81–86.

[6] Kalivarapu, V., Foo, J. L., Winer, E. H., “Implementation of Digital Pheromones for Use in Particle Swarm Optimization”, 47th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, 2nd AIAA Multidisciplinary Design Optimization Specialist Conference, Newport, RI, 1-4 May 2006.

[7] Hu X H, Eberhart R C, Shi Y H., "Engineering Optimization with Particle Swarm", *IEEE Swarm Intelligence Symposium*, 2003: 53-57.

[8] G. Venter and J. Sobieszczanski-Sobieski, "Multidisciplinary optimization of a transport aircraft wing using particle swarm Optimization", *In 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization 2002*, Atlanta, GA.

[9] P.C. Fourie and A.A. Groenwold, "The particle swarm algorithm in topology optimization", *In Proceedings of the Fourth World Congress of Structural and Multidisciplinary Optimization 2001*, Dalian, China.

[10] Shi, Y., Eberhart, R., "Parameter Selection in Particle Swarm Optimization", Proceedings of the 1998 Annual Conference on Evolutionary Computation, March 1998

[11] Shi, Y., Eberhart, R., "A Modified Particle Swarm Optimizer", *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation*, pp 69-73, Piscataway, NJ, IEEE Press May 1998

[12] Natsuki H, Hitoshi I., "Particle Swarm Optimization with Gaussian Mutation", *Proceedings of IEEE Swarm Intelligence Symposium*, Indianapolis, 2003:72-79.

[13] Hu, X., Eberhart, R., Shi, Y., "Swarm Intelligence for Permutation Optimization: A Case Study of n-Queens Problem", *IEEE Swarm Intelligence Symposium 2003, Indianapolis, IN, USA*

[14] Venter, G., Sobieszczanski-Sobieski, J., "Particle Swarm Optimization", *AIAA Journal*, Vol.41, No.8, 2003, pp 1583-1589

[15] Hu, X., Eberhart, R., "Solving Constrained Nonlinear Optimization Problems with Particle Swarm Optimization", 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2002), Orlando, USA

[16] Schutte, J., Reinbolt, J., Fregly, B., Haftka, R., George, A., "Parallel Global Optimization with the Particle Swarm Algorithm", *Int. J. Numer. Meth. Engng*, 2003.

[17] Koh, B, George A. D., Haftka, R. T., Fregly, B., "Parallel Asynchronous Particle Swarm Optimization", International Journal For Numerical Methods in Engineering", *International Journal of Numerical Methods in Engineering*, 67:578-595, 2006, Published online 31 January 2006 in Wiley InterScience, DOI: 10.1002/nme.1646.

[18] Hu, X., Eberhart, R., Shi, Y., "Particle Swarm with Extended Memory for Multiobjective Optimization", *Proceedings of 2003 IEEE Swarm Intelligence Symposium,* pp 193-197, Indianapolis, IN, USA, April 2003, IEEE Service Center

[19] Tayal, M., Wang, B., "Particle Swarm Optimization for Mixed Discrete, Integer and Continuous Variables", 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Albany, New York, Aug 30-1, 2004.

[20] Walter, B., Sannier, A., Reiners, D., Oliver, J., "UAV Swarm Control: Calculating Digital Pheromone Fields with the GPU", *The Interservice/Industry Training, Simulation & Education Conference (I/ITSEC),*Volume 2005 (Conference Theme: One Team. One Fight. One Training Future).

[21] Gaudiano, P, Shargel, B., Bonabeau, E., Clough, B., "Swarm Intelligence: a New C2 Paradigm with an Application to Control of Swarms of UAVs", *In Proceedings of the 8th International Command and Control Research and Technology Symposium*, 2003.

[22] Colorni, A., Dorigo, M., Maniezzo, V., "Distributed Optimization by Ant Colonies", *In Proc. Europ. Conf. Artificial Life,* Editors: F. Varela and P. Bourgine, Elsevier, Amsterdam, 1991.

[23] Dorigo, M., Maniezzo, Colorni, A., "Ant System: Optimization by a Colony of Cooperating Agents", *In IEEE Trans. Systems, Man and Cybernetics*, Part B, Vol. 26, Issue 1, pp 29-41, 1996.

[24] Montgomery, J., "Towards a Systematic Problem Classification Scheme for Ant Colony Optimization", *Technical Report tr02-15*, School of Information Technology, Bond University, Australia, 2002.

[25] White, T., Pagurek, B., "Towards Multi-Swarm Problem Solving in Networks", *icmas*, p. 333, Third International Conference on Multi Agent Systems (ICMAS'98), 1998.

[26] Parunak, H., Purcell M., O'Conell, R., "Digital Pheromones for Autonomous Coordination of Swarming UAV's". *In Proceedings of First AIAA Unmanned Aerospace Vehicles, Systems, Technologies, and Operations Conference*, Norfolk, VA, AIAA, 2002.

[27] Itzigehl, P., R., "A Method for Asynchronous Parallelization", International Conference on Software Engineering, Proceedings of the 10th International Conference on Software Engineering, Singapore, pp.4-9, ISBN: 0-89791-258-6, 1988.