We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

Distributed Particle Swarm Optimization for Structural Bayesian Network Learning

Ferat Sahin and Archana Devasia
Rochester Institute of Technology
USA

1. Introduction

Particle Swarm Optimization (PSO) was first introduced as a concept for a non-linear optimizer by Kennedy and Eberhart in 1995. Their seminal work articulates a technique of evolutionary computation, which has its origin in artificial intelligence and simplified social models such as bird flocking and fish schooling (Kennedy & Eberhart, Nov. 1995; Kennedy & Eberhart, Oct. 1995). Its early appeal lay in its use of only primitive mathematical operators and computational economy with regard to both memory and speed. The authors were influenced by the work of Reynolds, Heppner and Grenander in modeling bird flocking and recognized that the fundamental hypothesis to the development of PSO is that an evolutionary advantage is obtained by the social sharing of information among members of the same species. They stated that the simulation of the graceful but unpredictable choreography of a bird flock by collision-proof agents could be used as an effective optimizer for a wide range of functions.

1.1 PSO concept

The PSO technique involves casting a population of co-operative agents, randomly in the multidimensional search space. Each agent has an associated fitness value, which is evaluated by the fitness function to be optimized, and a velocity that directs its motion. Each agent can keep track of its solution that resulted in the best fitness as well as the solutions of the best performing agents in its neighborhood. The trajectory of each agent is dynamically governed by its own and its companions' historical behavior. Kennedy and Eberhart view this adjustment as conceptually similar to the crossover operation utilized by genetic algorithms (Kennedy & Eberhart, Nov. 1995). Such an adjustment maximizes the probability that the agents are moving toward a region of space that will result in a better fitness. At each step of the optimization, the agent is allowed to update its position by evaluating its own fitness and the fitness of the neighboring agents. The PSO algorithm is terminated when the specified maximum number of generations is reached or when the best particle position of the entire population cannot be improved further after a sufficiently large number of generations.

A simple pseudo code describing the functioning of the optimizer taken from (Taşgetiren & Liang, 2003) is shown below.

Source: Swarm Intelligence: Focus on Ant and Particle Swarm Optimization, Book edited by: Felix T. S. Chan and Manoj Kumar Tiwari, ISBN 978-3-902613-09-7, pp. 532, December 2007, Itech Education and Publishing, Vienna, Austria

Initialize parameters
Initialize population
Evaluate
Do{
    Find particlebest
    Find globalbest
    Update velocity
    Update position
    Evaluate
}While (Termination)

Kennedy and Eberhart realized that the behavior of the population of agents was more comparable to a swarm rather than a flock. This swarm behavior or swarm intelligence rests on five basic principles put forth by Millonas. These have been obtained from (Kennedy & Eberhart, Nov. 1995) and (Kennedy & Eberhart, Oct. 1995) and are listed as follows:

1. *Proximity principle*: The population should be able to carry out simple space and time computations.
2. *Quality principle*: The population should be able to respond to quality factors in the environment.
3. *Principle of diverse response*: The population should not commit its activities along excessively narrow channels.
4. *Principle of stability*: The population should not change its mode of behavior every time the environment changes
5. *Principle of adaptability*: The population must be able to change its behavior when its worth the computational price.

They found that the PSO concept seemed to be consistent with the checklist above. It could inherently carry out multidimensional space calculations over a series of time steps thus following the proximity principle. The agents could respond to quality factors such as their own best fitness values as well as the neighborhood best, in accordance with the quality principle. The algorithm could allocate responses between the individual best fitness value and the neighborhood best thus ensuring the fulfillment of the principle of diverse response. The population could change its mode of behavior only with a change in global best thereby suggesting stability. And finally, the change of state with a change in neighborhood best was in itself an indication of adaptability. The population was hence branded as a swarm. The authors called each agent of the swarm, a particle and hence the label particle swarm.

1.2 Mathematical formulation

The dynamic behavior of the swarm can be quantified as given in equation (1).

$$v(t+1) = v(t) + \phi_1\left(x - x_p\right) + \phi_2\left(x - x_n\right) \tag{1a}$$

$$x(t+1) = x(t) + v(t+1) \tag{1b}$$

Here, v is the particle velocity and x is the particle position which represents a test solution. In addition, ϕ_1 and ϕ_2 are uniform random variables which introduce an element of

uncertainty. The inclusion of such a stochastic factor facilitates an exhaustive search of the hyperspace under consideration thereby preventing the swarm from converging on to a local solution. Historically ϕ_1 and ϕ_2 are a combination of a positive constant and a random function. Thus (1a) becomes,

$$v(t+1) = v(t) + c_1 \times rand_1() \times (x - x_p) + c_2 \times rand_2() \times (x - x_n) \quad (2)$$

Here c_1 and c_2 are acceleration constants called cognition and social constants respectively, the functions $rand_1()$ and $rand_2()$ are random functions usually uniformly distributed between [0, 1]. The values of the constants determine the tension in the system (Shi & Eberhart, 2001). Low values allow the particles to roam far from the target regions before being pulled back, while high values result in abrupt movement toward or past target regions. Kennedy and Eberhart chose the both the acceleration constants to have a value of "2" in order to give the random factor a mean of "1" thereby causing the particles to overfly local optima and enable search in the region between local solutions. Variables ϕ_1 and ϕ_2 are clamped by an upper limit which is a parameter of the system.

The introduction of stochastic factors may cause the system to enter a state of explosion because of increased global exploration resulting in the particle velocities and positional co-ordinates tending to infinity. In order to prevent such a scenario, a maximum value of velocity $v_{\max}$ is usually defined. The second term in equation (1a) is the cognition part of the particle with the variable x_p representing the (previous) position of the particle that resulted in the best fitness so far. Kennedy and Eberhart referred to this as *simple nostalgia* (Kennedy & Eberhart, Nov. 1995). The last term of (1a) is the communal part which involves exchange of public knowledge. Here the variable x_n is the neighborhood position that resulted in the best fitness so far. Equation (1b) directs the new position of the particle based upon its current position and its new velocity.

1.3 Neighborhood size

In PSO, a neighborhood is defined for an individual particle as the subset of particles it is able to communicate with (Kennedy & Eberhart, April 2007). According to Bratton and Kennedy, since the earliest PSO model was a simulation of the social milieu, the neighborhood of choice was largely Euclidian. However it proved to be unwieldy and cumbersome in mathematical computations and hence was dispensed with to be replaced by topological neighborhoods. A number of neighborhood configurations have been discussed in literature. Some significant ones listed below are taken from (Kennedy, 1999; Guo et. al., July 2006) as shown in Fig. 1:

1. *Stars*: Every individual is connected to every other individual making the best performing individual the social source of influence.
2. *Circles*: Every individual is connected to only *K* of its immediate neighbors. This results in slower information propagation as compared to the stars topology. In this type of neighborhood, clusters are created that may converge onto different local optima. But due to neighbor to neighbor interaction, once the global solution is found, all the

clusters are pulled in towards it. For a $K = 2$ neighborhood (called a ring), it would take swarmsize/K number of steps for information about a new global best to be transmitted to the other side of the ring.

3. *Wheels*: One individual called the focal individual is connected to all the others and they are connected to only that one. The performance of the population is supervised by this central individual so as to determine the best and adjust its course according to it. If the adjustment results in improvement in the focal individual's performance then that improvement is communicated out to the rest of the population. This topology is faster than the ring topology.
4. *Random edges*: For N individuals, there are N random symmetrical connections between pairs of individuals.
5. *Von Neumann*: This topology is in the form of a 2-D lattice that wraps around itself as can be seen in Fig. 1(d).

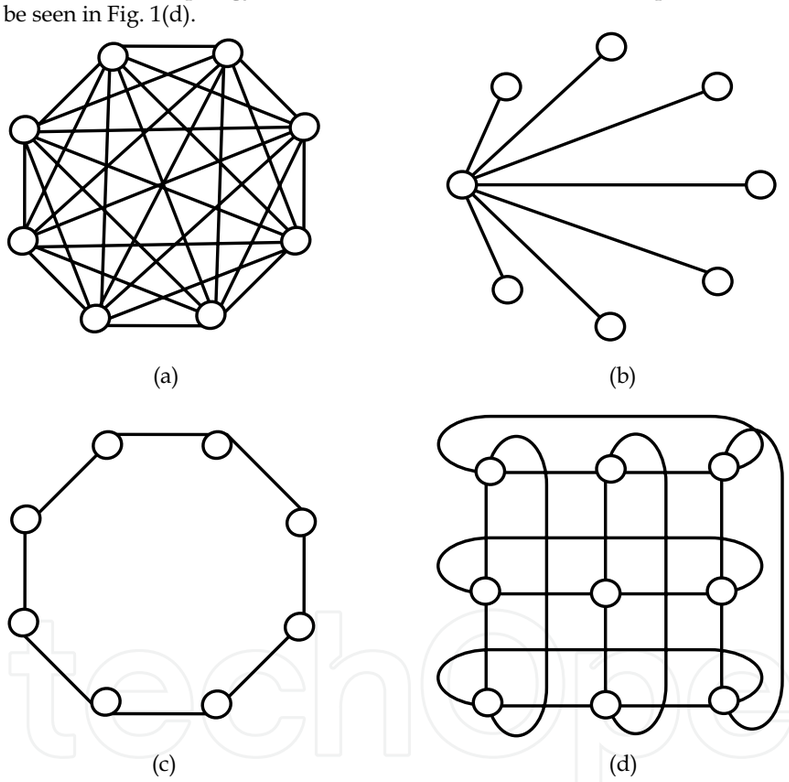


Figure 1. Neighborhood Topologies found in (Guo et. al, July 2006, Venayagamoorthy et. al, 2007): (a) Star, (b) Wheel, (c) Ring, (d) Von Neumann

1.3.1 Global neighborhood

A global neighborhood (also referred to as the GBEST model) has a star topology. The GBEST PSO algorithm as proposed in (Kennedy & Eberhart, Oct. 1995) is shown in Fig. 2.

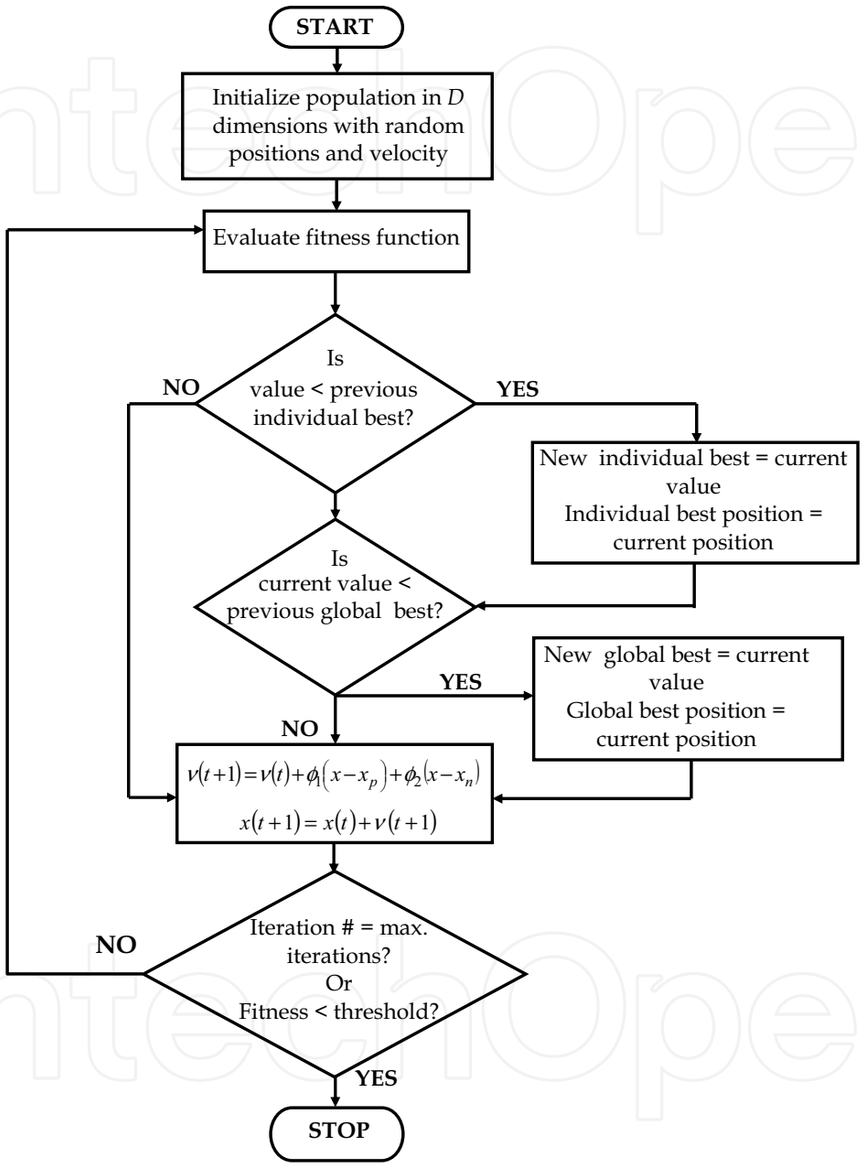


Figure 2. Flowchart of the GBEST PSO Algorithm

All the particles in the GBEST model try to reach the global solution. Hence even when a local solution is reached, all particles feel a tug in that direction. This may reduce the chances of the particles exploring the entire search space and may even cause the swarm to

converge at the local solution. However since every particle keeps track of every other particle in the swarm convergence rate is fast, which makes the GBEST approach an ideal candidate for uni-modal problems.

1.3.2 Neighborhood of *K*

This refers to the Circle topology. It is called the LBEST or local version of the PSO. The number of nearest neighbors is decided by the size of the neighborhood. As discussed previously, for a neighborhood of size *K*, each particle can communicate directly with only *K* other particles. Hence instead of moving toward the stochastic average of particle best and global best, the particles move toward the points defined by particle best and local best, which is the position of the particle with the best evaluation in the neighborhood (Kennedy & Eberhart, Oct. 1995). Kennedy and Eberhart found this local approach to be more flexible than the GBEST approach while trying to solve a three layer feed forward neural network designed to solve the XOR problem (Kennedy & Eberhart, Oct. 1995). They have attributed the insensitivity of this version to local solution to the fact that a number of groups of particles spontaneously separate and explore different regions. The LBEST ring model has been found to be suited for multi-model problems on account of its immunity to local optima convergence. The flipside to this limited interaction between swarm particles is the slower convergence rate in comparison to the GBEST model.

1.4 Other particle swarm parameters

In 2002, El Gallad *et al* have studied the various inputs required for working the particle swarm optimizer. Some of their findings are described below.

1) *Population of the swarm*: This factor depends upon the problem being optimized. Smaller swarms may be more successful for some problems while larger ones may be useful for others. However if the swarm size is too small it may result in convergence upon a local optimum while on the other hand very large swarms may increase computational time. Hence as suggested in (El Gallad *et al*, 2002) a balance has to be struck between the complexity of the algorithm and the risk of getting trapped in local optima by selecting a proper swarm size specific to the application at hand.
2) *Number of iterations*: The uncertainty in the velocity update equation introduced by the stochastic factors results in a global exploration of the search space that makes arriving at the global optimum extremely likely if the algorithm is run for a sufficiently long period of time. The use of the word *sufficient* is in itself indicative of the problem specific nature of this parameter. Indeed the permissible error margin, which strongly dictates the computational time, varies with the problem at hand. In cases where the time required to converge onto the global solution appears to be very long, it is more advantageous to run the algorithm for multiple short replications rather than running one very long replication. This is indeed a sound suggestion since it is possible that the time required for getting the particles out of local optima could be greater than the time required to reinitialize a new replication in (El Gallad *et al*, 2002). The stopping criterion for such multiple replications can be evaluated by observing if successive generations show any significant improvement or not.
3) *Velocity of particles*: This factor determines the fineness with which the hyperspace under consideration is searched. If the value of this parameter is too high, then the

particles may fly past the optimal solution and may even oscillate about a certain position. On the other hand if this value is too low, then the particles could get stuck at a local optimum. In order to circumvent this issue, an adaptive velocity technique can be applied. According to this approach, in the event that the solution found is oscillatory, the value of velocity is allowed to gradually decrease in a random fashion thereby helping the particles to get out of the oscillation and at the same time allowing the swarm to explore new areas.

1.5 Evolution of PSO through the ages

This section elucidates the development of PSO and details the various adjustments and modifications made to the original algorithm in order to maximize it performance.

1.5.1 Addition of inertia weight

Shi and Eberhart modified the PSO algorithm by introducing the concept of inertia weight. They argued that such a factor was necessary in order to bring a balance between global search and local search (Shi and Eberhart, 1998). Consider equation (1a). In the absence of the term representing the current velocity of the particle, the velocity would be memory less. If initially a particle, *i,* is at the best global position then it would be stationary at that position. The other particles would move toward the weighted centroid of their own best position and the global best causing the swarm to statistically contract toward the global best. This continues till another particle reaches a better global solution causing the particles to now statistically contract toward the new global best. The described scenario represents a search space that statistically shrinks over generations thus resembling a local search. Shi and Eberhart pointed out that in this case the global solution could be found only if it existed within the initial search space. Thus the search ability (i.e. global or local) could be varied by the presence or absence of the current velocity term in equation (1a). In order to fine tune the search ability, the inertia weight, w , was introduced which modified (1a) as follows.

$$v(t+1) = w\,v(t) + \phi_1\left(x - x_p\right) + \phi_2\left(x - x_n\right) \tag{3}$$

Shi and Eberhart used the problem of *Schaffer's f6* function to test this algorithm using w values ranging from 0 to 1.4. They found that the inertia weight in the range of [0.9, 1.2] resulted in a higher success rate of finding the global solution within a reasonable number of iterations as compared w values outside this range. They also experimented with time decreasing inertia weights and found that as w was linearly decreased from 1.4 to 0 from the first to the last iteration, the PSO showed significantly improved performance as regards success rate of finding the global optimum and number of iterations required to reach this optimum when compared to the case of using a fixed value of w . Further investigations were carried out in (Shi & Eberhart, 2000) using a linearly decreasing inertia weight starting at 0.9 and terminating at 0.4 on four benchmark functions viz. *spherical*, *Rosenbrock*, *Rastrigrin*, and *Griewank*. The mathematical expression for these functions can be found in Table 1. It was observed that the PSO algorithm converged quickly for all the four cases but reduced its convergence speed when reaching the optimum. Shi and Eberhart attributed this to the inability of the linearly decreasing inertia weighted PSO to perform a global search at the end of a run. If w_{int} and w_{fin} represent the initial and final values of w respectively,

MAX is the maximum number of optimization steps and *iter* represents the current iteration number, then a linearly decreasing w is defined in equation (4) (Iwamatsu, 2006),

$$w = (w_{\text{int}} - w_{fin}) \times (\text{MAX} - iter) / \text{MAX} + w_{fin} \tag{4}$$

Name	Mathematical Representation
Spherical	$f = \sum_{i=1}^{D} x_i^2$
Rosenbrock	$f = \sum_{i=1}^{D-1} \left\{ 100\left(x_{i+1} - x_i^2\right)^2 + \left(x_i - 1\right)^2 \right\}$
Rastrigrin	$f = \sum_{i=1}^{D} \left\{ x_i^2 - 10\cos(2\pi x_i) + 10 \right\}$
Griewank	$f = \frac{1}{4000} \sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$
Schaffer's f6	$f = 0.5 - \frac{\left(\sin\sqrt{x^2 + y^2}\right)^2 - 0.5}{\left(1 + 0.001\left(x^2 + y^2\right)\right)^2}$

Table 1. Benchmark functions used to test PSO in literature (Taşgetiren & Liang, 2003; Shi & Eberhart, 1998; Clerc & Kennedy, 2002)

1.5.2 Introduction of constriction coefficient

Clerc and Kennedy demonstrated that constriction coefficients could be used to prevent system explosion, which hitherto had been contained using $v_{\max}$ (Clerc & Kennedy, 2002). A constriction factor is defined as follows:

$$\chi = \frac{2}{\left| 2 - \phi - \sqrt{\phi^2 4\phi} \right|} \tag{5}$$

Here $\phi = \phi_1 + \phi_2$. The mathematical development leading to (5) is beyond the scope of this work but can be obtained from [13]. The constriction factor when inserted into the velocity update equation modifies equation (2) as follows,

$$v(t+1) = \chi\left[v(t) + c_1 \times rand_1() \times \left(x - x_p\right) + c_2 \times rand_2() \times \left(x - x_n\right) \right] \tag{6}$$

They also showed that for values of $\phi > 4$, the particles would quickly converge onto the global solution while for $\phi < 4$ the swarm would most likely get stuck at a local optimum. Such a behavior is similar to that exhibited by the inclusion of inertia weight, w, into the system response. This similarity spawned a study comparing the performance of a PSO

using a constriction factor with that using an inertia weight by (Eberhart & Shi, 2006). Five benchmark functions viz. *spherical*, *Rosenbrock*, *Rastrigrin*, *Griewank* and *Schaffer's f6* function were investigated during this performance analysis. It was found that even though it is not essential to specify the value of $v_{\max}$ in the constriction factor approach, limiting it to the dynamic range of each variable in each dimension (i.e. $x_{\max}$) of the system under consideration resulted in the fastest and most consistent way to obtain good results. The authors have shown that by setting $w = \chi$ and $\phi = c_1 + c_2$, the PSO algorithm using constriction factor can be considered as a special case of the PSO using inertia weight.

1.5.3 Use of adaptive scaling term

Sometimes situations are encountered wherein the evaluation of the objective function may not be feasible within a restricted time frame. In such cases the algorithm is limited to operate within an acceptable time resulting in a solution that is sub-optimal. The ideal choice here would be to accelerate the PSO in order to reduce convergence time and also increase the probability of finding the global optimum. This is the motivation for considering speed-up strategies for PSO. One such strategy proposed in (Fan, 2002) involved the use of an adaptive scaling term into the algorithm. As discussed previously the behavior of the swarm is modeled as shown in equation (1) and the necessary velocity limitations are applied as shown below,

$$\begin{aligned} v(t) &= V_{\max}, \text{ if } v(t) > V_{\max} \\ v(t) &= -V_{\max}, \text{ if } v(t) < -V_{\max} \end{aligned} \tag{7}$$

Fan explains that at the beginning of a search it is desirable that the particles be spread all over the search space in order to explore all possible regions to maximize the chances of finding the global solution. However as the search progresses, the searching scale should be reduced in order to allow the found solution to be refined. For this purpose he introduced a scaling term $\left(1-(t/T)^h\right)$ that revises (7) as,

$$\begin{aligned} v(t) &= \left(1-(t/T)^h\right)V_{\max}, \text{ if } v(t) > \left(1-(t/T)^h\right)V_{\max} \\ v(t) &= -\left(1-(t/T)^h\right)V_{\max}, \text{ if } v(t) < -\left(1-(t/T)^h\right)V_{\max} \end{aligned} \tag{8}$$

Here, t is the number of the current generation (i.e. optimization step), T is the maximum number of iterations and h is a positive constant chosen by trial and error. The velocity update and position update equations remain the same as shown in equation (1). Changes are effected only in setting the limits of velocity. Benchmark experiments revealed that this modified PSO performed better as compared to the original PSO on test functions such as spherical, Rosenbrock and Griewank's function. The modified PSO had a higher convergence rate than the original when used to solve these three function minimization problems. Fan found that the original PSO rapidly stagnated when no improvement was exhibited by its searched solution. However the modified PSO could still search progressively till the global solution was found indicating a higher reliability rate. Even with a fixed number of generations, the modified PSO exhibited better convergence reliability. It was also found that in case of the original PSO the parameter $v_{\max}$ strongly

influences the best function value, making the selection of $v_{\max}$ crucial. However in the case of the modified PSO, this parameter can be selected quite arbitrarily within a relatively large range. A preliminary study was also performed to examine the effect of the exponent h that controls the reducing speed of the searching scale on the algorithm. It was found that similar to $v_{\max}$; this parameter can also be arbitrarily selected over a wide range.

1.5.4 Inclusion of Boundary Conditions

In order to prevent the swarm searching outside the solution space, boundary conditions can be specified. These conditions are highly dependent upon factors such as problem dimensionality and the location of global optimum. The following list of boundary conditions has been taken from (Xu & Rahmat-Samii, 2007) who have also proposed two hybrids.

1. *Absorbing:* This is a type of restricted boundary condition in the sense that if a particle of the swarm flies outside the solution space in a particular dimension then it is tugged back to the boundary of the space of that dimension and its velocity is assigned a zero value. In 2007, Xu and Rahmat-Samii liken this situation to the energy of the errant particle being absorbed by a soft wall so that the particle is stuck on it, and eventually gets pulled back by its memory of best locations only.
2. *Reflecting:* This is another type of restricted boundary condition in which the deviant particle is pulled back to the boundary of the solution space of the dimension it overshot and the direction of its velocity in that dimension is altered. This is equivalent to the particle being reflected by a hard wall, and the energy driving it outside the boundary being totally reversed in order to accelerate it back toward the solution space.
3. *Damping:* This is the third type of restricted boundary condition and bears a resemblance to the reflecting boundary condition. In this case also the errant particle is drawn back into the solution space and is relocated at the boundary of the dimension under consideration where its velocity component is reversed and assigned a random number between 0 and 1. The only difference between a damping and a reflecting boundary condition is the introduction of this uncertainty factor, which makes the reflection imperfect.
4. *Invisible:* This is an unrestricted boundary condition in which the particle that leaves the solution space is not brought back but allowed to stay there. The fitness of that particle's position is not assessed and instead it is assigned a bad fitness value. In due course, the particle comes back into the solution space because of its inherent characteristic of setting its trajectory towards the weighted sum of global and individual best.
5. *Invisible/Reflecting:* This is the first of the two new unrestricted boundary conditions proposed in (Xu & Rahmat-Samii, 2007) and is a hybrid of the invisible and reflecting boundary conditions. In this case the errant particle is not pulled back to the solution space boundary and instead gets assigned a bad fitness score. Also, the direction of the velocity of the particle in the dimension under consideration is reversed because of which it accelerates back toward the solution space.
6. *Invisible/Damping:* This is the other new boundary condition proposed in (Xu & Rahmat-Samii, 2007) and is a combination of the invisible and reflecting boundary conditions. Again, the deviant particle is allowed to stay outside the solution space and gets assigned a bad fitness value while the direction of the velocity of the particle in that dimension is

reversed with a random factor between 0 and 1. As a result the particle comes back into the solution space.

1.6 Recent applications of PSO

Since its inception in 1995, the PSO algorithm has been used extensively; in some cases being tailored to suit the problem at hand and in other cases to solve issues that have not been attempted so far. In this section a brief description of some of the recent applications of the PSO algorithm have been described.

1. *Micro-PSO* (*μPSO*)*:* Recently a microparticle swarm optimizer (μPSO) is proposed for reconstructing the dielectric properties of normal and malignant breast tissues (Huang & Mohan, 2007). This is a type of high-dimensional microwave imaging which requires a large population of co-operative agents in order to find the global optimum for accurate image reconstruction. The population size adversely affects the computational effort required. Huang and Mohan have proposed an algorithm that utilizes a smaller population and implements a set of restart operations after the population has converged. If the population converges to a solution that is inferior or equal to the available best solution, the solution is blacklisted for future searches and all particles are prevented from converging onto the same solution again. They utilized the concept of the guaranteed convergence PSO and introduced a force of repulsion modeled on the lines of Coulomb's law of electrostatics between particles and blacklisted solutions. This repulsive force is inversely proportional to some power of the distance between the particle under consideration and a given blacklisted solution. The authors suggested the value of this power should be chosen in such a way that it cause enough force to repel the particles away from blacklisted solutions while at the same time allowing them to search spaces surrounding the blacklisted solutions. While selecting the value of the inertia parameter, the authors have employed an adaptive technique that sets the value of w depending on the quality of solutions found. As regards the type of neighborhood, since a μPSO typically consists of only 3-5 agents, the authors have suggested the use the GBEST topology.
2. Application to Electromagnetic Devices: The PSO is successfully applied for the purpose of optimizing the design of electromagnetic devices, particularly the problem of a super conducting magnetic energy storage (SMES) configuration with eight free parameters (Ho et al, 2006). In their attempt they have suggested certain enhancements in order to balance the exploration and exploitation capabilities of PSO. Stagnation may be introduced into the PSO algorithm due to sharing of information between the particles of the swarm. In order to boost up the diversity of the algorithm, the authors have proposed the introduction of an age variable, which is representative of the age of a global best, or an individual particle's best. If this age exceeds a certain threshold value then that particular solution is disposed and replaced by a new randomly generated solution thus improving global search ability. The authors also recommend that in order to further ensure the solution diversity of the particles, a Roulette wheel scheme should be adopted for selecting the individual and global bests from their respective sets. For the purpose of balancing personal and social experience as well as exploration and exploitation two new random factors are introduced by the authors. The former in this case is actually a combination of rand1() and rand2() (defined in equation (2)) set in such a way that the sum of rand1() and rand2() equals 1. Ho et al

also proposed the inclusion of an intensified search into the algorithm for accurately identifying the global optimum. They have explained this method as follows. When a global best is found, an intensification search is activated in the neighborhood around this point using only its speed vector with the cognitive and social influences being deliberately excluded in the velocity updating formula. In this iterative process, if a search is successful, the algorithm will keep the velocity vector unchanged while continuing its exploitation using this speed vector; otherwise, the algorithm will generate randomly a new speed vector to begin the next refinement search. The intensification search process will be repeated until the number of consecutive unsuccessful explorations around a new reaches a previously decided number.

3. Application to Circuit Partitioning: The PSO is applied to a circuit partitioning problem (Venayagamoorthy et al. 2007). Such a partitioning is essential in order to reduce the number of test vectors required to detect faults in VLSI circuits. The authors have compared the performance of a standard I-PIFAN (improved primary input and fanout based partitioning approach) algorithm in partitioning combinational CMOS circuits into a number of sub-circuits with that of a modified version employing PSO (called PSO-PIFAN). In the I-PIFAN, the circuit can be partitioned depending upon the combinations of the maximum node fan in size N and the minimum partitioning fanout value F. Venayagamoorthy et al showed that I-PIFAN's search is exhaustive and hence slow and is constrained within a pre-specified range of N and F combinations. The best result has to be selected from this range. Thus, if the optimal solution is outside the specified range of N and F values then it will not be found. In the case of the PSO-PIFAN, all combinations of N and F are searched simultaneously without necessitating a specified range. The authors have concluded that the PSO-PIFAN performs a directed search of the solution space and uses its memory to accelerate the PSO particles towards the global solution in a shorter time and will always converge to the optimal solution.
4. Application in Power Systems: Recently, there has been an attempt to demonstrate the feasibility and robustness of PSO in solving a transient stability constrained optimal power flow problem (TSCOPF) (Mo et al, 2007). They tested the algorithm on two test systems viz. the IEEE 30-bus system and the New England 39-bus systems with promising results. Comparison with GA revealed PSO to be better equipped for solving multi-contingency TSCOPF. In order to accelerate the process of computation, the authors have proposed the use of a parallel computing environment.

1.7 Binary PSO

In order to easily solve combinatorial problems such as scheduling and routing issues that involve ordering or arranging of discrete elements, Kennedy and Eberhart proposed a binary version of the PSO optimizer, which could operate on two valued functions (Kennedy & Eberhart, 1997). In this adaptation of the original PSO, the position of each particle is described either by a 0 or a 1 in each dimension. In this case, the velocity of the particle in a particular dimension represents the probability of the position of the particle in that dimension being 0 or 1. A sigmoid limiting transformation $\sigma(v(t+1))$ is used to update the position of the particle under consideration by comparing it to a random number ρ. This is expressed in the equation (9).

$$\text{If } \sigma(v(t+1)) > \rho \text{ then } x(t+1) = 1 \text{ otherwise } x(t+1) = 0 \tag{9}$$

The random number, ρ, is considered to be uniformly distributed in the range [0, 1].

The pseudo code for the discrete PSO developed by Kennedy, Eberhart and Shi and taken from (Guo et al, 2006) is described as follows:

```
Loop
    For i = 1 to N_p                                  // N_p is the number of particles
        If G(X^t_i) > G(P^t_i) then                   // G( ) evaluates the objective function, X^t_i is a potential solution i.e. X^t_i = (x^t_i1, x^t_i2,... x^t_iD), x^t_id {1,0}, D is the number of dimensions, t is the iteration number and P^t_i = (p^t_i1, p^t_i2,... p^t_iD) is the best solution that particle i has obtained until iteration t
            For d = 1 to D bits
                p^t_id = x^t_id                       //p^t_id is best so far
            Next d
        End if
        g = i                                         //arbitrary
        For j = indices of neighbors (or population)
            If G(P^t_j)>G(P^t_g) then g = j           // P^t_g = (p^t_g1, p^t_g2,... p^t_gD) is the best solution in the population or neighborhood at iteration t and g is index of best performer in neighborhood (or population)
        Next j
        For d = 1 to D
            v^t_id = v^t-1_id + c1r1(p^t_id - x^t_id) + c2r2(p^t_gd - x^t_id)
            v^t_id  [-V_max, + V_max]
            If random number < ρ(v^t_id) then
                x^t+1_id = 1
            else
                x^t+1_id = 0
        Next d
    Next i
Until criterion
```

1.7.1 Recent Applications of Binary PSO

Recently, the binary PSO approach is applied to the problem of polygonal approximation of digital curves (Yin, 2004). This problem is of significance since it is used in a number of image analysis tasks such as object recognition, image matching and target tracking. A polygon can be used to represent a shape in an image since the information of a shape is

mainly preserved at the corners that have the maximal measure of significance. The problem at hand is to approximate the curve along the corners as closely as possible while at the same time keeping the number of vertices at the corners of the polygon (called degree of the polygon) as small as possible. Yin employed the binary PSO technique developed in (Kennedy & Eberhart, 1007) with a slight modification. He introduced a local search heuristic in between successive generations of the discrete PSO so that once the algorithm found a good region within a given iteration, it could exploit that region thoroughly before moving onto another region. This hybrid PSO showed significantly improved performance in terms of the number of polygon vertices necessary for the same error and variations in results between different runs as compared to the original binary PSO.

Another work has showed the use of binary PSO in optimizing flowshop scheduling problems (Liao *et al,* 2007). They used a variant of the GBEST model to search for the best global solution. Instead of determining the global best for a given iteration from the individual bests of the individual particles up to that iteration, the global best was determined from the positions of the particles at the current iteration. Liao *et al* showed this technique to perform better than the conventional GBEST model. Even though it spent more time on converging, it increased the probability of not getting stuck at a local solution. In an attempt to further improve the PSO performance, Liao *et al* introduced a local search scheme to be carried out once, every fixed number of iterations within the PSO loop. The main idea was that given a current solution, the PSO mechanism would lead the solution to an intermediate solution. The local search would be applied to this intermediate solution in order to reach the global solution. The binary PSO method has been applied to define a preliminary short/medium range aircraft configuration, fully compliant with given requirements, that allows a minimum direct operating cost (Blasi & Del Core, 2007). In this work they tested two different boundary conditions viz. absorbing wall technique and reflecting wall. The authors found the latter technique to provide a slightly improved performance over the former. They also compared the PSO method with that of a previously studied genetic optimizer and found the PSO method to be quite promising.

2. Application of PSO to fault diagnosis of airplane engines

The work discussed in this work involves using PSO in conjunction with Bayesian Networks (BN) for diagnosing and predicting faults in airplane engines. A distributed Particle Swarm Optimization approach is explored in order to construct the best BN from a large dataset comprising of raw data taken from the sensors of airplane engines during actual flights. The inherent parallelism of the PSO technique has been exploited with the algorithm being implemented on a cluster of 48 processors using Message Passing Interface (MPI) in Linux. The seamless blend of graph theory and probability theory that makes uncertainty representation both instinctive and spontaneous is an inherent characteristic of Bayesian Networks and this makes it a highly appealing option for fault diagnosis. This work attempts to employ Bayesian Networks for the purpose of creating a fault diagnosis system. Initially no expert information is available as regards the relationship between the variables forming the network and it is discovered solely from the available engine data. After the network is conceptualized, expert information is incorporated for a more accurate modeling of the dependencies associated between fault and other system variables. The task of determining the Bayesian Network that best fits the data is accomplished by means of PSO.

As mentioned previously, the parallel behavior exhibited by the PSO technique is employed in fitness evaluation of the processed data on a cluster of 48 CPUs running parallel, using MPI in Linux. Such an arrangement serves to substantially reduce computational time.

2.1 Overview of Bayesian Networks

A Bayesian Network (BN) is a probabilistic network that provides a cogent and coherent depiction of the dependencies and independencies between the variables of interest. Such a network is a graphical model in the form of a directed acyclic graph (DAG), which has a causal semantics thereby enabling an effortless incorporation of causal prior knowledge. The strength of these causal relationships is encoded in the form of conditional independence assertions between the variables (Heckerman, 1995). Consider a domain of random variables given by $U = (X_1, X_2, \ldots, X_n)$. These signify the nodes of a network. Conditional dependencies are represented in the form of directed links between variables. An arrow from node X_1 to node X_2 indicates X_1 to be the parent of X_2. In order to quantify the effect of the parents on the node, a conditional probability distribution is associated with it defining its local semantics, e.g. each node X_i, has a conditional probability distribution $P(X_i \mid Parents(X_i))$. The product of these local conditional distributions evolves into global semantics of the problem at hand with the Chain rule being its mathematical manifestation. The Chain rule expresses the relationship between the unconditional probabilities $P(X_i)$, the conditional probabilities $P(X_i \mid e)$, where e is the evidence and the joint probability $P(U)$ as shown in equation (10). Here $P(U) = P(X_1, X_2, \ldots, X_n)$. An exponential enhancement in $P(U)$ is observed as the number of variables escalates.

$$P(U) = \prod_{i=1}^{n} P(X_i \mid Parents(X_i)) \tag{10}$$

2.2 Bayesian Learning

Incomplete knowledge spawns learning. It is a means of obtaining information through experience. Bayesian Learning uses hypotheses as intermediaries between data and predictions (Russell & Norvig, 1995). The main steps are:

- Estimating the probability of each hypothesis given the data
- Making predictions from the hypotheses, using the posterior probabilities of the hypotheses to weight the predictions

Four classes of *Bayesian Network Learning* arise based on whether the structure of the network is known or unknown and the variables are observable or hidden. These include the following:

1. *Known structure complete data:* This is the case where the network is specified and the data does not contain any missing values. It involves evaluation of the conditional probability tables for each node of the network from the complete data.
2. *Known structure incomplete data:* For this case the network is specified but the data is by no means complete and consists of missing values or hidden variables. The missing data can be estimated on the basis of the available data and the information about the missing data – an approach that is adopted by the *Expectation-Maximization* (EM)

algorithm (Friedman, 1995) and by *Gibbs' Sampling*. *Bound and Collapse* (BC) (Sebastiani & Ramoni, 2000) is another technique which can be explored given such a scenario.

3. *Unknown structure incomplete data:* Such a problem involves an unspecified network structure coupled with data having missing vales. Exact solutions are not viable and hence such problems call for sub optimal networks which can be determined using gradient based algorithms using *structural* EM and BC (Sebastiani & Ramoni, 2000).
4. *Unknown structure complete data:* The problem dealt with in this chapter belongs to this category. It attempts to learn the structure of the BN using the complete sensor data and on the basis of the developed structure endeavors to diagnose presence/absence of faults in airplane engines. Here the network topology has to be generated such that it fits the data the best. The number of structures grows super-exponentially as the number of variables multiplies, making such a problem computationally expensive. Thus applying distributed PSO could help greatly.

2.3 Structural Learning

In order to demonstrate the suitability of Bayesian Networks as an inference tool for predictive maintenance of airplane engines, the network has to be built first and this requires learning its topology using the available sensor data. This is structural BN learning. Given a training set *D*, the problem of learning a BN involves finding a network *B* that best matches *D* (Friedman, 1995). Structural BN learning can be addressed using either constraint based or score based learning. The former deals with conducting statistical tests on the given data and then determining a unique DAG that is consistent with the observed (in)dependencies. The latter approach focuses on optimization. It involves finding a network structure that maximizes a defined scoring function that represents how well each network structure fits the data. Less vulnerability to errors in individual tests gives score based methods an edge over constraint-based techniques. The approach in this work is score based. Literature provides an assortment of scoring functions which include *log-likelihood* (Heckerman, 1995) , the *minimal description length* (MDL) score (Lam & Bacchus, 2000), *Bayesian score* (Heckerman, 1995) etc. Of these, the *K2* scoring metric (based on a Bayesian approach) provided in (Cooper & Herskovits, 1992) has been found to be the most successful. The technique applied in the presented work is Bayesian score, which can be described as having the following form:

$$Score(B:D) = P(B \mid D) = \frac{P(D \mid B)P(B)}{P(D)} \tag{11}$$

$$P(D \mid B) = \int P(D \mid \theta_B, B)P(\theta_B \mid B)\mathrm{d}\,\theta_B \tag{12}$$

Here, *D* represents the data and *B* represents a network candidate. The network structure that maximizes $P(D \mid B)P(B)$, maximizes the score as well. The probability $P(D \mid B)$ is evaluated in the equation (12), where θ_B is a parameter of the network *B*.

As discussed previously, the goal of score based methods is to find the highest scoring network structure. This is accomplished by means of a search algorithm. This score + search approach is NP-hard (Chickering et al, 2004) justifying a heuristic approach (Djan-Sampson & Sahin, 2004). The most commonly used algorithm is a simple greedy hill-climbing

algorithm. However it suffers from the ills of local maxima and plateaus that have adverse effects on the score. Heuristic searches generally assume that ordering of variables is known and many do not scale well with networks having a large number (more than five) of variables. Additional scaling difficulties arise while dealing with large datasets such as gene and census data (Sahin et al, 2007; Yavuz et al, 2006). In order to avoid the pitfalls of heuristic searches we use a PSO based approach, as it is highly compatible with large datasets and large networks.

2.4 Applying Binary PSO

In this problem each particle of the swarm represents a BN. The position of each particle is made up of a string of 0s and 1s where each bit represents whether an edge exists between the nodes indexed by the bit. Assuming no node can be its own parent, the binary string will contain $n(n-1)$ bits. The fitness is calculated using the scoring function given below (Herskovits, 1992):

$$Fitness = 1/\log_{10}\left(\prod_{i=1}^{n}\prod_{j=1}^{q_i}\frac{(r_i-1)!}{(N_{ij}+r_j-1)!}\prod_{k=1}^{r_i}N_{ijk}\right) \tag{13}$$

Here r_i is the number of states for node i, the first product is over the nodes in the network, the second product is over the set of permutations of the parents of node i, and the third product is over the states of node. Also N_{ij} is defined as

$$N_{ij} = \sum_{k=1}^{r_i} N_{ijk} \tag{14}$$

Here, N_{ijk} is an entry in the conditional probability table for node i. The conditional probability table elements contain occurrences of joint instantiations of the parents, (each permutation is indexed with *j*) of node *i* for which node *i* is in state *k*. Hence, the sum N_{ij} is a total of a column of the conditional probability table, where each column enumerates occurrences of node i in each state for a specific instantiation set of parents. At each step of the optimization, equations (1a) and (9) are used to update the particle velocities and positions respectively.

3. Fault Diagnosis

Fault diagnosis using PSO based Bayesian Network learning is accomplished in two steps: preprocessing and network discovery. Preprocessing generates the input dataset. Network discovery is accomplished by the PSO algorithm that is run in a computer cluster. The output is a network that correctly models the system dependencies and serves as a tool for system diagnosis and monitoring as well as fault prediction.

3.1 Preprocessing

Extensive information pertaining to an assortment of airplane engine parameters viz. temperature, altitude, pressure, flight phase etc. is furnished by sensors connected to the airplane engines. MATLAB is used for the purpose of storing this raw data as structure files. Each structure file comprises of sensor information pertaining to a single flight. The raw data has to be suitably condensed in order to optimize computational efforts. The fact that oil related variables account for most number of airplane engine faults forms the basis of such a condensation. Hence, from each raw data file the features corresponding to only oil related variables are extracted. Another aspect necessitating preprocessing of raw data is data sampling. In order to allow all the oil related variables in a given file to have equal lengths, a sampling interval adjustment is vital. The necessity of sampling uniformity stems from the fact the sampling rate of different sensors is different.

In an effort to further reduce computational expense, focus is restricted to information obtained during the approach phase of the flight. The rationale behind the choice of flight phase is the fact that the sensors relevant for lube diagnosis record a broad range of values during the approach phase thereby allowing us to delineate distinct states for the BN nodes (Sahin et al, 2007). Such a choice also helps extend the coverage of flight data analysis since unlike take-off and cruise, the approach phase has not been studied as well (Sahin et al, 2007) . The adjusted data pertaining to a particular flight now constitutes equal sized arrays of sensor readings corresponding to only engine oil failure related variables, further narrowed down to include only approach phase readings. In a Bayesian Network, the maximum number of states corresponding to each node directly influences the total run time of the network structure learning algorithm. Hence it is crucial to reduce the variation of the values in the adjusted dataset. This is accomplished by means of an *equal frequency data binning algorithm*. Similar data are grouped together into bins while at the same time ensuring that that each bin contains a fairly equal number of elements. The data is tagged based on the bin numbers, which represent the probable states a given variable would be at a particular point in time. Thus a reduction in the maximum number of states associated with each node is brought about. In the presented work, each node is chosen to have four states (four bins). The equal frequency binning algorithm works as follows:

1. Initially a minimum number of elements (say n) are considered to be clustered together in one bin.
2. The first bin is filled up with the first n number of elements, the second bin by the next n number of elements and so on. This may cause the last bin to contain more or less than n number of elements.
3. To ensure that similar elements are in the same bin some elements are transferred to or from adjacent bins
4. Any resulting empty bins are discarded.
5. The bins are checked to see if similar elements are grouped together in the same bin. If not the control goes back to step (3)
6. The original data is represented by the bin number.

3.1.1 Addition of Fault

The binned data is classified as faulty or non faulty by introducing an additional column named Fault in the data. Information regarding the presence or absence of Fault is

determined from the maintenance records of the airplane engines. For example, a flight before an oil related repair on an engine is categorized as faulty while the very first flight after that maintenance is considered to have non faulty flight data. The entries of the Fault column are set to 1 or 0 respectively depending on whether the raw data file is faulty or not.

3.1.2 Packing data and compression

In order to reduce the size of the data, we have packed the data by combining data elements in bytes and applied compression techniques. The size reduction in the data file improved the performance of the distributed PSO algorithm since smaller data can be sent to the slaves faster in the cluster. Thus, less time was required to complete the algorithm. After packing and compression, it was possible to condense the original file by about 40-75 times. Details of this approach can be found in (Sahin et al, 2007).

3.2 Using Particle Swarm Optimization for Searching the best Bayesian Network

Parallelism is the hallmark of the PSO algorithm and this feature can be efficiently exploited for fitness calculation, as it is the most computationally demanding aspect of a BN search, especially when the problem at hand involves a large number of variables or large datasets. The following sections explore the characteristics of the implemented PSO. Fig. 3 shows the distributed PSO in master-slave framework.

3.2.1 Parallel Computing for Particle Swarm Optimization

The PSO algorithm was run on a cluster of 48 CPUs operating in parallel. An MPI (Message Passing Interface) having a master slave framework was implemented. The particle swarm was managed and initialized by the master. Each slave process received a particle from the master and was required to calculate its fitness and send it back. After all the fitness results for the swarm were received by the master, the algorithm was advanced by one step i.e. one iteration. The master again sent out the newly evolved particles to the slaves and the procedure was repeated.

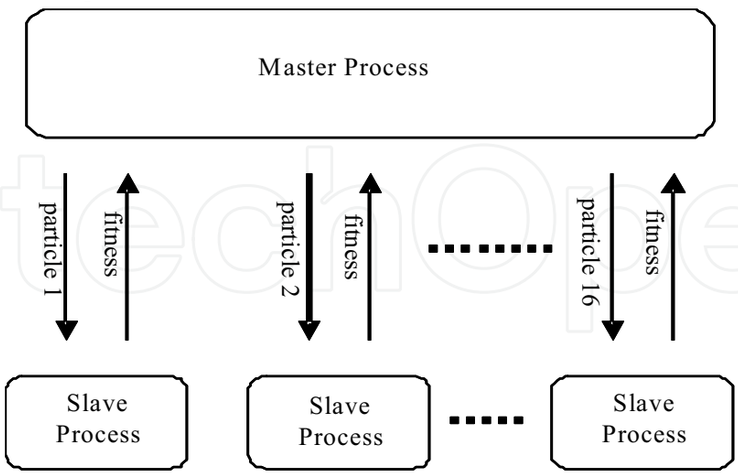


Figure 3. Parallel implementation of PSO algorithm

For dynamic evolution of the swarm, all the processes must wait for each other to complete their current fitness calculation. Such implementation architecture is termed as *synchronous PSO*. Efficient parallel implementation of the PSO algorithm was accomplished by keeping the number of slave processes equal to or greater than the number of particles. This is because with an adequate number of processes there is a high probability that all the processes up to the number of particles will compute fitness and return the values at approximately the same time resulting in a small idle process time.

3.2.2 Particle and Velocity Initialization

The particles in the swarm were heuristically initialized. If there were N nodes in the network, each particle was initialized to contain a randomly selected set of $N/2$ edges. If this resulted in a cyclic particle then it had to be axed and recreated. This was critical since the chosen fitness function was designed to handle only acyclic graphs. The maximum number of arcs was restricted to $2N$. Such a restriction did not impact the particle initialization. There was a possibility of encountering the problem of cyclic particles yet again when the particles were allowed to 'fly'. At such instances the cyclic particles were identified and rendered acyclic by repeated removal of edges. For velocity initialization, each component of each particle's velocity was randomly initialized on the interval

$$-v_{\max} \le v_o \le v_{\max} \tag{15}$$

This initialization lead to particles having approximately $N(N-1)/2$ arcs after they were moved for the first time. This ensured adequate initial exploration of the BN bit string particle swarm. Effectual exploration of the search space demanded intelligent selection of maximum velocity in order to prevent greediness from creeping in.

3.3 Training and testing

For the purpose of network generation and fault prediction, the PSO based structural BN learning code is developed in two modes: simulation and inference. The simulation mode is also referred to as training. This mode involved using a set of preprocessed data files (called training files) for exploring the optimal representation of the system dependencies by execution of the PSO algorithm. Other input parameters of significance included the number of PSO particles, type of neighborhood, and the number of optimization steps. This resulted in a BN that was representative of the input preprocessed (training) data. Inference mode is also called the testing mode. In this mode the accuracy of the generated BN in diagnosing faults in known and unknown datasets was investigated. A collection of preprocessed files different from those used for training was tested by using the BN. For this purpose a preprocessed training sample set, its corresponding BN realization and the set of files to be tested were fed into the inference engine. Correct diagnosis of known files served to validate the use of the BN for fault prediction in unknown datasets. Table 2 shows the list of engine oil failure variables under investigation that directly or indirectly influences Fault. The problem of coming up with the best BN that models the dependencies between the listed variables has been attempted in our previous work (Sahin et al, 2007; Yavuz et al, 2006). Here we incorporate expert information in order to make our model more accurate. This expert input is of two types. Firstly we tag certain variables to be independent of others as a result of which they show absence of parents in the resulting DAG. Secondly, we determine and discard those variables that have no influence on the system.

4. Experimental Tests and Results

Oil related variable	Symbol	Remarks
Pressure Altitude	ALT	-
Engine Cycle	ECYC	Independent variable
Engine Hours	EHRS	Independent variable
Exhaust Gas Temperature	EGT	-
Fuel Flow	FF	-
Mach	MACH	-
Fan Speed	N1	-
Core Speed	N2	-
Oil Pressure	OIP	-
Oil Temperature	OIT	-
Power Lever Angle	PLA	-
Total Pressure	PT	-
Total Air Temperature	TAT	Independent variable
Thrust Mode	TMODE	-
Engine Vibration	VIB	-

Table 2. List of oil related variables

4.1 Incorporation of independent variables

Based on experts at Honeywell Inc., three oil related variables viz. *Engine Cycle* (*ECYC*), *Engine Hours* (*EHRS*) and *Total Air Temperature* (*TAT*) are considered to be unaffected by others and hence are marked as independent variables. Initially a set of 10 files comprising of an equal number of faulty and non-faulty files are selected. After preprocessing, this data is fed into the simulation mode of the software that utilizes PSO to generate the required best BN. An accurate BN entails an appropriate PSO, the efficacy of which depends upon judicious selection of its parameters. To come up with the most efficient optimizer, four parameters viz. number of optimization steps, swarm size, maximum velocity of particles and type of neighborhood were investigated. These are enumerated in Table 3. The training data was subjected to an exhaustive series of simulations in order to study the inter-dependencies between the various PSO parameters and construct the best BN. As perceived from the Table 3, a total of 450 simulations were carried out.

PSO parameters	Values	Remarks
Number of optimization steps	1000, 2000, 3000, 4000, 5000	In a single run the PSO parameters take up specified values from column 2. Each run is repeated five times and the quality of the network generated by using those values for the parameters is assessed by considering the average fitness score of the five runs
Type of neighborhood	Global, neighborhood of 2	
Number of particles	8, 16, 24	
Maximum velocity of particles	6, 8, 10	

Table 3. List of PSO Parameters

Based on these 450 runs with different PSO parameter, we see that the behavior of a network generated using PSO is highly dependent upon the inter-relationship between the PSO parameters. Hence the choice of parameter values is always problem specific.

The quality of the generated BNs was evaluated on the basis of the fitness score and the number of parents to the Fault node. Networks with higher number (three or more) of parents to Fault were desirable since these provided better inference results (Sahin et al, 2007). Also networks with smaller fitness scores tended to differentiate faulty and non faulty files better (Sahin et al, 2007). As a result of the 450 different simulations carried, 23 networks having four or more parents to Fault were obtained. They are as listed in Table 4.

Network	Steps	No. of particles	Neigh-borhood	Velocity	No. of parents	Fitness Score	Percentage Fault	
							Faulty Test File	Non-Faulty Test file
Network1	3000	24	0	10	5	-3.6058E-06	92.61%	50.03%
Network2	5000	24	2	6	5	-3.4560E-06	96.65%	45.57%
Network3	5000	16	2	6	4	-3.4486E-06	49.99%	80.00%
Network4	3000	24	2	10	4	-3.4283E-06	99.90%	38.17%
Network5	5000	24	2	10	4	-3.4095E-06	58.50%	79.01%
Network6	5000	24	2	8	4	-3.3882E-06	88.26%	69.40%
Network7	5000	24	0	6	4	-3.3345E-06	60.28%	80.92%
Network8	5000	24	2	10	4	-3.3232E-06	58.54%	60.62%
Network9	4000	16	0	6	4	-3.2829E-06	98.00%	55.01%
Network10	2000	24	2	10	5	-3.2558E-06	67.59%	47.20%
Network11	4000	16	0	6	5	-3.2409E-06	52.43%	59.44%
Network12	2000	8	2	8	5	-3.2369E-06	49.99%	81.41%
Network13	2000	24	2	8	4	-3.2255E-06	78.73%	80.66%
Network14	3000	24	0	10	4	-3.2123E-06	33.60%	71.63%
Network15	3000	8	0	6	4	-3.1994E-06	87.59%	79.55%
Network16	4000	16	0	8	5	-3.1810E-06	63.27%	79.90%
Network17	5000	24	2	10	5	-3.1687E-06	81.30%	77.34%
Network18	2000	16	0	8	4	-3.1684E-06	96.13%	45.82%
Network19	2000	16	2	6	4	-3.1497E-06	50.08%	69.40%
Network20	4000	24	2	10	4	-3.1247E-06	99.50%	53.97%
Network21	3000	16	0	6	5	-3.1185E-06	77.55%	47.44%
Network22	3000	8	2	6	4	-3.0115E-06	98.53%	46.34%
Network23	1000	16	0	6	4	-3.0060E-06	67.87%	50.03%

Table 4. Simulation and inference results

Each network was tested on a set of seven known files in order to determine its diagnostic capability. The results are indicated in the final two columns of Table 4. Let us examine Network 2. It has five parents to Fault and a very good fitness score. It indicated a fault probability of about 97% and above for faulty files and a fault probability of about 46% and below for non-faulty files, thus exhibiting acceptable proficiency in fault diagnosis. Now consider Network 1. It also has five parents to Fault and in fact the best (i.e. lowest) fitness score as compared to the other networks. It was able to successfully diagnose faulty files as seen by the high value of fault probability for faulty files. However it demonstrated some ambiguity while diagnosing non-faulty files. This irregularity can be attributed to data over fitting. Increase in the number of parents to the Fault node does not always ensure

successful diagnosis. In fact overfitting may introduce excessive variance thereby reducing the prediction quality of the model. More than the number, it is *which* variables are parents to the Fault node is what is significant. Network 10 may also be the victim of such an overfitting as is indicated by the diminished capability of the network in diagnosing faulty files. The inferior diagnostic capability of Networks 11, 12, 16 and 17 can be attributed to poor fitness score in addition to overfitting. On the other hand, Network 4 with 4 parents to Fault exhibited excellent diagnostic capability even surpassing Network 2. Networks 5 through Network 8 have low (i.e. good) fitness scores but are weak representations of the system as observed by the excessively high fault probabilities predicted by these networks for non-faulty data. On the other hand, Networks 21 and 22 with relatively high (i.e. poor) fitness scores function as effective diagnostic tools. This may be considered as an indication towards the significance of the variables that affect Fault directly (i.e. are parents to Fault) as opposed to their number. Table 5 lists the parent variables to Fault for the networks discussed in Table 4.

Network	Fault Percentage		Parents to Fault
	Faulty Test File	Non-faulty Test File	
Network1	92.614632%	50.034897%	EHRS, MACH, PT, TMODE, VIB
Network2	96.649025%	45.570953%	ECYC, EGT, MACH, N2, TMODE
Network3	49.987072%	80.001564%	ALT, EGT, EHRS, OIP
Network4	99.900787%	38.174068%	ALT, EGT, EHRS, OIT
Network5	58.498676%	79.011688%	ECYC, EHRS, OIP, PLA
Network6	88.262665%	69.401489%	EGT, EHRS, OIP, PLA
Network7	60.278790%	80.917755%	OIP, PLA, PT, TAT
Network8	58.544533%	60.622322%	ALT, MACH, N1, TAT
Network9	98.004845%	55.009872%	EHRS, N2, PT, TMODE
Network10	67.592590%	47.199936%	ECYC, EGT, N1, OIT, TMODE
Network11	52.425045%	59.435143%	ALT, ECYC, EHRS, N1, VIB
Network12	49.987072%	81.405655%	ALT, ECYC, OIP, PLA, TAT
Network13	78.727547%	80.657501%	PT, TAT, TMODE, VIB
Network14	33.595486%	71.633659%	OIP, PLA, PT, TMODE
Network15	87.590485%	79.550301%	ECYC, OIP, PT, VIB
Network16	63.271446%	79.898872%	EGT, MACH, OIP, PT, TAT
Network17	81.295242%	77.337982%	ECYC, MACH, N1, OIP, TMODE
Network18	96.130951%	45.815529%	ALT, EGT, EHRS, PLA
Network19	50.079407%	69.395515%	FF, PLA, PT, TMODE
Network20	99.503967%	53.965019%	ECYC, PLA, TMODE, VIB
Network21	77.546539%	47.442135%	EHRS, N1, N2, OIT, TMODE
Network22	98.533943%	46.341629%	ECYC, N2, TAT, TMODE
Network23	67.869797%	50.034897%	EGT, N2, FF, PT

Table 5. Parents to Fault node

From Table 5 it is observed that all the networks with acceptable diagnostic capability viz. Networks 2, 4, 18, 21 and 22, include the variables ALT and/or N2 and/or TMODE as parents to Fault. Since the amount of data is limited for such a study no generalizations will be made. However it must be pointed out that the presence of these variables as well as that of others not identified here but which may very well appear repeatedly as parents to Fault

in further studies can be considered of certain consequence while deciding the suitability of a network for diagnosing or testing new data. In summary, while evaluating the quality of the BN for inference purposes, it is essential to consider the fitness score and, not only the number of parents but also the variables that act as parents to Fault.

4.2 Removal of irrelavent variables

Another effective way to enhance the accuracy of modeling and accelerate the algorithm is to determine and discard those variables that have no influence on the network. Such variables appear in the form of leaf nodes or islands. In order to obtain a visual representation of the networks generated from the simulations, a program called GraphViz was employed (GraphViz software). These graphical depictions were examined in order to ascertain the unnecessary variables. Three variables viz. EGT, MACH and VIB consistently appeared as leaf nodes in a number of networks. Fig. 4 (a) and (b) illustrate networks having these variables as leaf nodes.

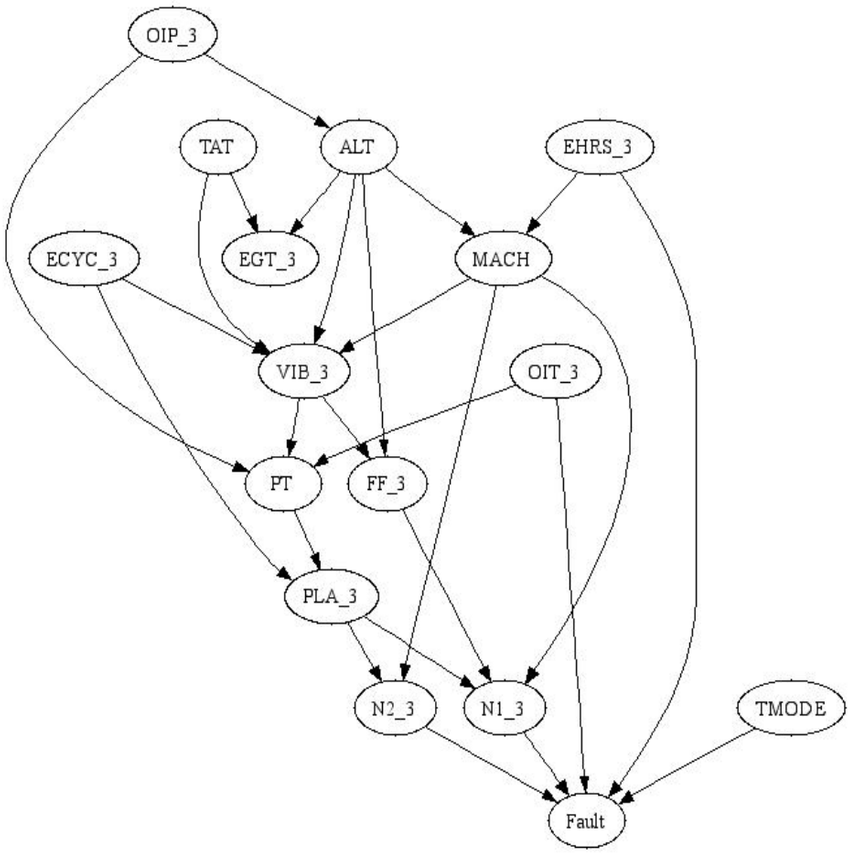


Figure 4(a). BN generated exhibiting variable EGT as a leaf node

Once the variables were identified, they were not included while pre-processing the raw data. An appropriate training set consisting of five faulty and five non-faulty files was selected and fed into the simulation mode of the software. The PSO parameters were chosen corresponding to those that resulted in the best diagnostic capability. The diagnostic proficiency of the resulting BNs was tested on a group of seven known files. The results are as presented in Table 6.

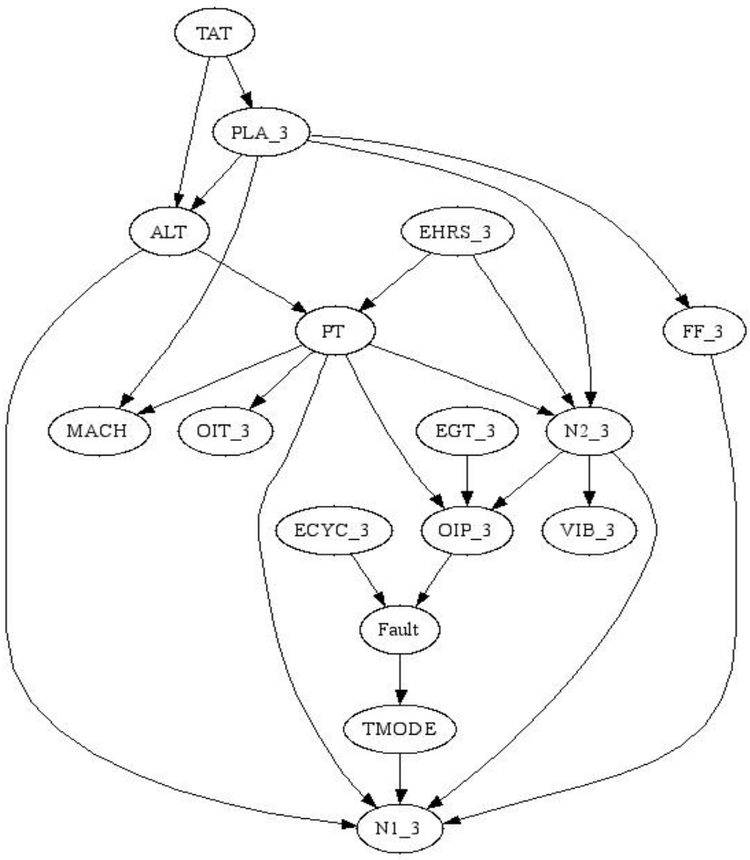


Figure 4 (b). BN generated using the proposed software exhibiting variable VIB as a leaf node

Five good networks were obtained by following the procedure indicated in Section 4.1. The values of the PSO parameters of these networks were selected while training the data with variables EGT, MACH and VIB removed. For each set of PSO parameters, four different runs were executed with an aim to obtain at least one network having three or more parents to Fault. As seen in Table 6, only one run out of 20 runs resulted in Fault having four parents. Three parents to Fault were found in nine runs. These 10 networks were then used to diagnose fault in a set of files consisting of two faulty and five non-faulty files. The results

are indicated in the last two columns of Table 6. Networks 2, 10 and 17 are able to successfully diagnose faulty files. Though there seems to be some uncertainty in diagnosing non-faulty files this approach looks promising. Further study is on to better the predictive capability for non-faulty files by performing extensive number of simulations and assessing the influence of the remaining variables on the network.

Network	Optimi-zation steps	No. of particles	Neighbor-hood	Velocity	Parents to Fault	Fitness Score	Fault Percentage	
							Faulty Test Files	Non Faulty Test Files
Network 1	5000	24	2	6	3	-	58.50%	53.10%
Network 2	5000	24	2	6	3	-4.0015E-06	99.97%	55.24%
Network 3	5000	24	2	6	1	-	-	-
Network 4	5000	24	2	6	2	-	-	-
Network 5	3000	24	2	10	1	-	-	
Network 6	3000	24	2	10	2	-	-	-
Network 7	3000	24	2	10	2	-	-	-
Network 8	3000	24	2	10	0	-	-	-
Network 9	2000	16	0	8	3	-4.2130E+06	58.50%	49.70%
Network 10	2000	16	0	8	3	-4.1461E-06	99.97%	51.96%
Network 11	2000	16	0	8	1	-	-	-
Network 12	2000	16	0	8	1	-	-	-
Network 13	3000	16	0	6	4	-4.4827E-06	62.37%	56.54%
Network 14	3000	16	0	6	3	-4.0790E-02	45.63%	60.02%
Network 15	3000	16	0	6	1	-	-	-
Network 16	3000	16	0	6	3	-4.0195E-06	0.00%	48.76%
Network 17	3000	8	2	6	3	-3.7565E-06	98.57%	52.61%
Network 18	3000	8	2	6	3	-3.8662E+06	47.86%	72.62%
Network 19	3000	8	2	6	3	-3.6141E-06	89.88%	76.83%
Network 20	3000	8	2	6	2	-	-	-

Table 6. Simulation and inference results with variable removal

5. Conclusion

This work involved the implementation of a highly successful technique for fault diagnosis and predictive maintenance of airplane engines. Some of the highlights of the discussed Bayesian Network approach include creation of the network without prior information and later incorporating expert information for better modeling, monitoring, and diagnosing faults in known systems, predicting faults in unknown systems, ability to handle large systems and the possibility of modifying the technique for diagnosing and distinguishing different types of faults. The presented Particle Swarm Optimization technique was effectual in reducing the computational complexity of the problem at hand by capitalizing on its innately parallel behavior thereby enabling the application of a cluster of 48 CPUs for faster network creation. Thus the developed software had several advantages of being generic, robust, scalable and modifiable.

6. References

Kennedy J. & Eberhart R. (Nov. 1995), Particle Swarm Optimization, *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, Nov 1995, pp. 1942-1948.

Kennedy J. & Eberhart R. (Oct. 1995), A New Optimizer using Particle Swarm Theory, *Proceedings of the 6th International Symposium on Micro Machine and Human Science,* Oct. 1995, pp. 39-43.

Taşgetiren M. F. & Liang Y.-C. (2003), A Binary Particle Swarm Optimization Algorithm for Lot Sizing Problem, *Journal of Economic and Social Research*, vol. 5, No. 2, 2003, pp. 1-20.

Shi Y. & Eberhart R. (2001), Particle Swarm Optimization: Developments, Applications and Resources, *Proceedings of the Congress on Evolutionary Computation*, vol. 1, 2001, pp. 81-86.

Kennedy J. & Eberhart R. (2007), Defining a Standard for Particle Swarm Optimization, *Proceedings of the IEEE Swarm Intelligence Symposium*, April 2007, pp. 120-127.

Kennedy J. (1999), Small Worlds and Mega-Minds: Effects of Neighborhood Topology on Particle Swarm Performance, *Proceedings of the Congress on Evolutionary Computation*, vol. 3, July 1999, pp. 6-9.

Guo Q.-J., Yu H.-B., & Xu A.-D. (2006), A Hybrid PSO-GD based Intelligent Method for Machine diagnosis, *Digital Signal* Processing, vol. 16, No. 4, July 2006, pp. 402-18.

El-Gallad A., El-Hawary M., Sallam A. & Kalas A. (2002), Enhancing the Particle Swarm Optimizer via Proper Parameters Selection, *IEEE Canadian conference on Electrical and Computer Engineering*, vol. 2, May 2002, pp. 792-797.

Shi Y. & Eberhart R. (1998), A Modified Particle Swarm Optimizer, *Proceedings of the IEEE International Conference on Evolutionary Computation*, May 1998, pp. 69-73.

Kennedy J. & Eberhart R. (1997), A Discrete Binary Version of the Particle Swarm Algorithm, *IEEE International Conference on Systems, Man and Cybernetics*, vol. 5, Oct. 1997, pp. 4104-4108.

Shi Y. & Eberhart R. (2000), Empirical Study of Particle Swarm Optimization, *Proceedings of the Congress on Evolutionary Computation*, vol. 1, July 2000, pp. 6-9.

Iwamatsu M. (2006), Locating All the Global Minima Using Multi-Species Particle Swarm Optimizer: The Inertia Weight and The Constriction Factor Variants, *Proceedings of the Congress on Evolutionary Computation*, vol. 3, July 2006, pp. 816-822.

Clerc M. & Kennedy J. (2002), The Particle Swarm – Explosion, Stability and Convergence in a Multidimensional Space, IEEE Transactions on Evolutionary Computation, vol. 6, No. 1, Feb 2002, pp. 58-73.

Eberhart R. & Shi Y. (2006), Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization, *Proceedings of the Congress on Evolutionary Computation*, vol. 3, July 2006, pp. 816-822.

Fan H. (2002), A Modification to Particle Swarm Optimization Algorithm, *Engineering Computations*, vol. 19, No. 8, 2002, pp. 970-989.

Xu S. & Rahmat-Samii Y. (2007), Boundary Conditions in Particle Swarm Optimization Revisited, *IEEE Transactions on Antennas and Propagation*, vol. 55, No. 3, March 2007, pp. 760-765.

Huang T. & Mohan A. S. (2007), A Microparticle Swarm Optimizer for the Reconstruction of Microwave Images, *IEEE Transactions on Antennas and Propagation,* vol. 55, No. 3, March 2007, pp. 568-576.

Ho S. L, Shiyou Y., Guangzheng N. & Wong H. C. (2006), A Particle Swarm Optimization Method with Enhanced Global Search Ability for Design Optimizations of Electromagnetic Devices, *IEEE Transactions on Magnetics,* vol. 42, No. 4, April 2006, pp. 1107-1110.

Venayagamoorthy G. K., Smith S. C. & Singhal G. (2007), Particle swarm-based optimal partitioning algorithm for combinational CMOS circuits, *Engineering Applications of Artificial Intelligence*, vol. 20, 2007, pp.177-184.

Yin P.-Y. (2004), A Discrete Particle Swarm Algorithm for Optimal Polygonal Approximation of Digital Curves *J. Vis. Comm. Image R.*, vol. 15, 2004, pp. 241-260.

Liao C.-J., Tseng C.-T. & Luarn P. (2007), A Discrete Version of Particle Swarm Optimization for Flowshop Scheduling Problems, *Computer & Operations Research*, vol. 34, 2007, pp. 3099-3111.

Blasi L. & Del Core G. (2007), Particle Swarm Approach in Finding Optimum Aircraft Configuration, *Journal of Aircraft*, vol. 44, No. 2, March-April 2007, pp 679-683.

Heckerman D. (1995), A Tutorial on Learning Bayesian Networks, Microsoft Research, 1995.

Russell S. & Norvig P. (1995), *Artificial Intelligence: A Modern Approach,* 1995, Prentice Hall.

Friedman N. (1995), Learning Belief Networks in the Presence of Missing Values and Hidden Variables, *Proceedings of the 14th International Conference on Machine Learning*, 1995, pp. 125-133.

Sebastiani P. & Ramoni M. (2000), Bayesian Inference with Missing Data using Bound and Collapse, *Journal of Computational and Graphical Statistics*, vol. 9, No. 4, Dec. 2000, pp.779-800.

Lam W. & Bacchus F. (1994), Learning Bayesian Belief Networks: An Approach Based on the MDL Principle, *Computational Intelligence*, vol. 10, 1994, pp. 269-293.

Cooper G. & Herskovits E. (1992), A Bayesian method for the Induction of Probabilistic Networks from Data, *Machine Learning*, vol. 9, No. 4, Oct. 1992, pp. 309-347.

Chickering D. M., Heckerman D. & Meek C. (2004), Large-Sample Learning of Bayesian Networks is NP-Hard, *Journal of Machine Learning Research*, vol. 5, 2004, pp. 1287-1330.

Djan-Sampson P. and Sahin F. (2004), Structural Learning of Bayesian Networks from Complete Data using Scatter Search Documents, *Proceedings of the IEEE Conference on Systems, Man and Cybernetics*, vol. 4, Oct 2004, pp. 3169-3624.

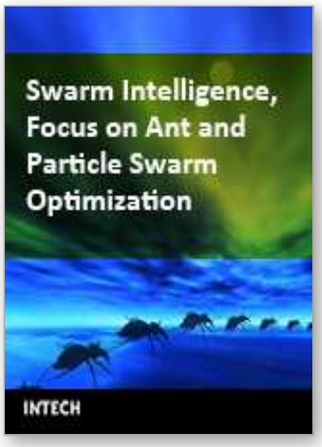
Sahin F., Yavuz M. C., Arnavut Z. & Uluyol O. (2007), Fault Diagnosis for Airplane Engines using Bayesian Networks and Distributed Particle Swarm Optimization, *Journal of Parallel Computing*, vol. 33, No. 2, March 2007, pp. 124-143.

Yavuz M. C., Sahin F., Arnavut Z. & Uluyol O. (2006), Generating and Exploiting Bayesian Networks for Fault Diagnosis in Airplane Engines, *Proceedings of the IEEE International Conference on Granular Computing,* April 2006, pp. 250-255.

Herskovits E. (1992), Computer-based probabilistic network construction, Stanford University, CA.

Mo N., Zou Z. Y., Chan K. W., & Pong T. Y. G. (2007), Transient stability constrained optimal power flow using particle swarm optimisation, *IET Generation, Transmission, and Distribution*, vol. 1, issue 3, May 2007, pp. 476-483.

GraphViz Software website. http://www.graphviz.org.



Swarm Intelligence, Focus on Ant and Particle Swarm Optimization
Edited by FelixT.S.Chan and Manoj KumarTiwari

ISBN 978-3-902613-09-7
Hard cover, 532 pages
Publisher I-Tech Education and Publishing
Published online 01, December, 2007
Published in print edition December, 2007

In the era globalisation the emerging technologies are governing engineering industries to a multifaceted state. The escalating complexity has demanded researchers to find the possible ways of easing the solution of the problems. This has motivated the researchers to grasp ideas from the nature and implant it in the engineering sciences. This way of thinking led to emergence of many biologically inspired algorithms that have proven to be efficient in handling the computationally complex problems with competence such as Genetic Algorithm (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), etc. Motivated by the capability of the biologically inspired algorithms the present book on "Swarm Intelligence: Focus on Ant and Particle Swarm Optimization" aims to present recent developments and applications concerning optimization with swarm intelligence techniques. The papers selected for this book comprise a cross-section of topics that reflect a variety of perspectives and disciplinary backgrounds. In addition to the introduction of new concepts of swarm intelligence, this book also presented some selected representative case studies covering power plant maintenance scheduling; geotechnical engineering; design and machining tolerances; layout problems; manufacturing process plan; job-shop scheduling; structural design; environmental dispatching problems; wireless communication; water distribution systems; multi-plant supply chain; fault diagnosis of airplane engines; and process scheduling. I believe these 27 chapters presented in this book adequately reflect these topics.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following:

Ferat Sahin and Archana Devasia (2007). Distributed Particle Swarm Optimization for Structural Bayesian Network Learning, Swarm Intelligence, Focus on Ant and Particle Swarm Optimization, FelixT.S.Chan and Manoj KumarTiwari (Ed.), ISBN: 978-3-902613-09-7, InTech, Available from: http://www.intechopen.com/books/swarm_intelligence_focus_on_ant_and_particle_swarm_optimization/distributed_particle_swarm_optimization_for_structural_bayesian_network_learning

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen