Engineering Applications of Artificial Intelligence 25 (2012) 1611–1627

Contents lists available at SciVerse ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

MPI-based parallel synchronous vector evaluated particle swarm optimization for multi-objective design optimization of composite structures

S.N. Omkar [a,]*, Akshay Venkatesh [b], Mrunmaya Mudigere [a]

[a] Department of Aerospace Engineering, Indian Institute of Science, Bangalore-560012, India
[b] National Institute of Technology Karnataka, Surathkal-575025, India

ARTICLE INFO

Article history:
Received 1 July 2011
Received in revised form
31 March 2012
Accepted 27 May 2012
Available online 10 July 2012

Keywords:
Multi-objective optimization
Vector evaluated particle swarm optimization (VEPSO)
Message passing interface (MPI)
Composite laminated plates
Peer-to-peer communication
Vector evaluated genetic algorithm (VEGA)

ABSTRACT

This paper presents a decentralized/peer-to-peer architecture-based parallel version of the vector evaluated particle swarm optimization (VEPSO) algorithm for multi-objective design optimization of laminated composite plates using message passing interface (MPI). The design optimization of laminated composite plates being a combinatorially explosive constrained non-linear optimization problem (CNOP), with many design variables and a vast solution space, warrants the use of non-parametric and heuristic optimization algorithms like PSO. Optimization requires minimizing both the weight and cost of these composite plates, simultaneously, which renders the problem multi-objective. Hence VEPSO, a multi-objective variant of the PSO algorithm, is used. Despite the use of such a heuristic, the application problem, being computationally intensive, suffers from long execution times due to sequential computation. Hence, a parallel version of the PSO algorithm for the problem has been developed to run on several nodes of an IBM P720 cluster. The proposed parallel algorithm, using MPI's collective communication directives, establishes a peer-to-peer relationship between the constituent parallel processes, deviating from the more common master-slave approach, in achieving reduction of computation time by factor of up to 10. Finally we show the effectiveness of the proposed parallel algorithm by comparing it with a serial implementation of VEPSO and a parallel implementation of the vector evaluated genetic algorithm (VEGA) for the same design problem.

1. Introduction

Material history has traversed through the stone, the bronze and the iron ages and at present we are in the "composite age". Superior mechanical characteristics of composite materials such as high stiffness to weight ratio compared to conventional metals and their inherent amenability towards the tailoring of their properties have made them indispensable. With their low cost and weight benefits, composite laminated plates, a class of composite materials, are extensively used in mechanical, automobile, aerospace, marine, biomedical, civil and other branches of engineering. Composite laminates are a series of lamina or plies of varying thicknesses and various fiber orientations stacked in a certain order to obtain desired directional stiffness and strength properties as required for an acceptable design. Diverse material behaviors, tailored to specific structural needs, can be obtained by slightly altering the properties of these composites. With such characteristics at dispense the choice or the configuration of such materials is left to the manufacturer and is influenced by the application of the composite.

Composite design optimization typically involves identifying the optimal laminate stacking sequence, ply thickness and the optimal number of plies. Such design variables generally have very vast ranges which contribute to the enormity of the number of solutions offered. Hence manufacturers are often burdened with the laborious process of selecting the right combination of elements from the many choices available to them. Hence, optimization of the available solutions becomes necessary.

Composite design optimization involves tackling combinatorial problems determined by their design variables with very vast solution spaces, rendering the problem under the class of constrained non-linear optimization problems (CNOP). Of these solutions those with the least weight and cost are preferred over the rest. Specifically in aerospace applications, weight minimization and reducing manufacturing cost has always been a priority in the manufacturing industry owing to benefits such as reduced fuel consumption and an increased payload. Therefore, the focus is on combined minimization of manufacturing cost and structural weight. This multi-objective nature of the problem and the difficulty in selecting the right values out of a large range of constrained design variables makes mathematical optimization a

* Corresponding author. Tel.: +91 080 229 32873: fax: +91 080 236 00134.
E-mail addresses: omkar@aero.iisc.ernet.in (S.N. Omkar), akshaydirefloyd@gmail.com (A. Venkatesh).

0952-1976/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.engappai.2012.05.019

natural tool for the design of laminated composite structures (Gurdal et al., 1999).

Optimum fiber orientations of laminated composite plates, for the maximum strength taken as the objective function, using state space based methods under multiple in-plane loading conditions with Tsai-Wu failure criterion for several test problems are carried out by Kim et al. (1997)). Adali et al. (1996) have discussed the weighted average method of multi-objective design of symmetrically laminated plates for different criteria like maximum strength, stiffness and minimum mass. Topal and Uzmana (2010) have developed a modified feasible direction (MFD) method for the multi-objective optimization of symmetrical angle-ply, square laminated plates subjected to biaxial compressive and uniform thermal loads in order to maximize the buckling load.

The emergence of heuristics such as Genetic Algorithm, Ant colony Optimization, Tabu Search, etc in solving functional optimization problems has brought in its wake the possibility of solving many problems like Traveling Salesman Problem, Quadratic Assignment and Graph problems, network routing, clustering, data mining, job scheduling and problems of NP-complete nature (Garey and Johnson, 1979) and proves to be a viable alternative to traditional mathematical tools which suffer from drawbacks such as local minimum trapping and single-path searching among others. The multi-objective design optimization of composite laminated structures, also being an NP-complete natured problem, has been solved using various nature-inspired algorithms (Ghasemi and Ehsani, 2007; Pelletier and Senthil, 2006; Deka, 2005; Boyang et al., 2000; Luersen and Holdorf Lopez, 2009). Many researchers have developed different approaches to minimize the weight and cost of laminated composite structures. Of the many heuristic techniques available PSO has become increasingly popular and has found many applications.

Particle swarm optimization (PSO) developed and introduced by Kennedy and Eberhart (1995), is an effective stochastic, non-gradient based global optimization algorithm derived by simulating social behavior depicted by a flock of birds, where each individual gleans from its own as well as the whole flock's discoveries. It is applicable for a wide range of non-linear function optimization problems. PSO's global search capability and insensitivity to scaling design variables (Schutte et al., 2005) makes it particularly suitable for problems (Kovacs et al., 2004) similar to that considered in this paper. PSO supersedes Genetic algorithm (Hassan et al., 2005) in the context of ease of implementation and proves computationally superior to traditional gradient-based algorithms (Snyman, 2004).

Vector evaluated particle swarm optimization (VEPSO) (Parsopoulos and Vrahatis, 2002), a multi-swarm variant of the PSO, an algorithm that adopts and modifies the main ideas of vector evaluated genetic algorithm (VEGA) (Schaffer, 1985) is particularly suited for multi-objective (MO) optimization problems (Goel et al., 2007; Deb, 2001) where often the objectives are competing, incommensurable and need simultaneous optimization. VEPSO algorithm is based on the principle where each swarm is exclusively evaluated with one of the objective functions, but, information coming from other swarm(s) is used to influence its motion in the search space. The best position attained by each particle separately and the global best positions from a different swarm are the main guidance mechanisms of the swarm. The information exchange among swarms enables optimizing all the objective functions involved.

In the proposed case of the composite design optimization problem, because the minimization of weight does not necessarily ensure the minimization of the corresponding cost, because of the disparate nature of the objective functions, VEPSO has been adopted and fitness evaluations based on the objectives of minimizing the weight and cost are carried out by two separate swarms deployed in a vast solution space where mutual exchange of information between swarms governs them towards a convergence point—indicating movement of the swarms towards a near optimal solution.

Despite its applicability and simplicity, PSO and its variant VEPSO take long durations for the completion of the optimization process due to the application problem's complexity and the nature of algorithms adopted which look to simulate socio-cognitive behavior by pursuing a sequential approach to phenomena that are innately parallel in nature.

Since the inception of computer science as a discipline, computer scientists and engineers have realized that a possible route to accelerating the execution of a computational task is to exploit the parallelism inherent in its program flow. Recent advancements in scientific computing techniques and the emergence of parallel programming platforms have helped in achieving quicker execution of complex problems through parallelization of serial algorithms. Such related work can be seen in literature (Oh et al., 2009; Yua et al., 2007; Parsopoulos et al., 2004).

Of the many available paradigms, Message-passing on cluster computers is one of the main programming paradigms used for high performance scientific computing these days. Message passing interface (MPI) is a specification standard defined by a broadly based group of parallel computer vendors, library writers, and applications specialists (William Gropp et al., 1996; The MPI Forum, 1995). MPI is a de-facto standard for message-passing used for developing high-performance portable parallel applications (Matsuda et al., 2008; Hempela and Walkerb, 1999). MPI standard defines a library of routines that implement the message-passing model (Gropp et al., 1994). The function of MPI, as the name implies, is to help several concurrently computing processes communicate by passing messages between them (William Gropp et al., 1996). Many researchers have carried out master-slave paradigm based parallel implementations using the message passing model (Coello Coello and Sierra, 2004; Schutte et al., 2004; Dubreuil et al., 2006).

Long execution time owing to the computational complexity of the problem, despite the use of popular heuristics like PSO and its variant VEPSO, and the availability of parallel programming environments, and the need to test the suitability of the relatively unused peer-to-peer paradigm for this line of research, were the driving forces behind the present work leading to parallelization of the VEPSO algorithm for the composite design optimization problem. The novelty of this parallelization lies in the use of MPI's collective operations, such as *bcast* and *allreduce* (Bova and Carey, 2000), which enable the development of a decentralized/peer-to-peer relationship between communicating nodes to solve the composite problem, as opposed to point-to point operations such as *send* and *receive* (Houzeaux and Codina, 2003), which support master-slave architecture (Gorlatch, 2002).

To summarize, for this work, a novel parallel VEPSO algorithm is presented, based on decentralized/peer-to-peer paradigm, to optimize the design of composite plates, based on the principles of classical laminate plate theory (CLPT)—to determine the stresses at each layer subjected to Uniformly Distributed Load and Point load. To check for failure two failure criteria are used—Maximum Stress (Narayana Naik et al., 2011) failure criterion and Tsai-Wu (Narayana Naik et al., 2011) failure criterion. This computation problem has been decomposed into parallelizable parts and run concurrently on a Linux-based IBM P720 cluster computer in tandem with MPICH v.1.2.7, a high performance portable implementation of MPI. The decomposition is such that the computation roles of individual particles of the swarms, which solve the optimization problem, are assumed by the different nodes of the cluster computer and to enable communication among these particles/nodes, emulating swarm communication behavior, MPI's collective communication primitives are made use of. Thus particles configured on different nodes, initialized with stochastically generated design configurations, perform calculations iteratively and in parallel to finally arrive at an optimum or near optimum design configuration. The results show a considerable decrease in the execution time of the parallel algorithm compared to the serial one. To further compare this parallel approach with a more commonly used heuristic for this problem domain, a parallel vector evaluated genetic algorithm has been implemented using MPI and executed on the same platform. Effective execution time comparisons of the algorithms are presented in the results and discussion sections. Execution time of the present parallel approach for VEPSO is found to be comparatively less than its GA counterpart. This approach, at the time of writing the paper, is the first to use MPI's collective communication to develop a peer-to-peer based parallel VEPSO for composite plate optimization. This work is primarily a follow-up to a previous work (Omkar et al., 2008) in our efforts to find faster ways of solving the composite optimization problem and confirming the efficacy of new methods (such as peer-to-peer parallelization) for the problem domain, starting with a smaller problem used here. The scalability and the speedup that this novel decentralized parallel technique offers shows promise in application to problems of higher complexity in the field and hence our future efforts will consider longer execution problems.

The paper is organized as follows. Section 2 includes the brief explanation of basics of the multi-objective problems. Section 3 discusses the emergence of PSO and VEPSO. Section 4 elucidates the details of the problem and its formulation and outlines the optimization process. Section 6 explains the necessity of VEPSO. Section 7 explains MPI's role in the parallelization used for the algorithms and its implementation to the optimization problem. Sections 8 and 9 includes the results, discussion, comparisons, and conclusions, respectively.

2. Multi-objective optimization (MO)

Let $S \subseteq \mathbb{R}$ be an n-dimensional search space and

$f_i(x) : S \to \mathbb{R}, i = 1,\ldots,k$

be objective k functions defined over S. Let,

$g_i(x) \le 0, j = 1,\ldots,m$

be m inequality constraints, then MO problem is nothing but finding the vector $x^T = (x_1^T,\ldots,x_n^T) \in S$which satisfies the m constraints and optimizes the function

$F(x) = [f_1(x),\ldots,f_k(x)] : \mathbb{R}^n \to \mathbb{R}^k$

In most cases the objective functions may be in conflict, thus, it is not possible to obtain the global minimum at the same point for all the objectives. The goal of such multi-objective optimization problems is to provide a set of Pareto optimal solutions (Goel et al., 2007; Deb, 2001).

If $u=(u_1,\ldots,u_k)$ and if $v=(v_1,\ldots,v_k)$, then u dominates v if and only if $u_i \le v_i$ $i=1$, $u_i \le v_i$, $i=1,\ldots,k$ (in the case where optimization means finding the global minimum) and $u_i \le v_i$ for at least one component. This property is known as Pareto dominance and it is used to define the Pareto optimal points. Thus, a solution x of the MO problem is said to be Pareto optimal if and only if there does not exist another solution y, such that $F(y)$ dominates $F(x)$.

The set of all Pareto optimal solutions of an MO problem is called Pareto optimal set and it is denoted as P^T and $PF^T = ((f_1(x),\ldots,f_k(x)) | x \in P^T$ is called Pareto front.

3. Particle swarm optimization (PSO)

Particle swarm optimization (PSO) is an evolutionary optimization algorithm proposed by Eberhart and Kennedy while attempting to simulate the motion of bird flocks as part of a socio-cognitive study investigating the phenomenon of 'collective intelligence' in biological populations. In Particle swarm optimization (PSO) each swarm unit/particle explores a possible solution depending on the point in the search space where it exists. Its trajectory is influenced by its own as well as the entire swarm's learning. The personal best position is the best solution that is found by the particle in the course of flight. The best position of the whole flock is the global best solution. The former and latter are called personal best and global best, respectively. Every unit/particle of the swarm continuously updates itself through the aforementioned best solutions. Thus, particles move in the defined solution space to finally converge at a possible optimal point. In practice, the fitness function, which is determined by the optimization problem, determines the quality of solutions at various points in the search space of the problem at hand.

If the population of swarm is N, then the current position of the ith, ($\forall i=1,2,3,\ldots,N$) particle is expressed as $X_i(t)$. The current best position discovered by the ith particle is expressed as $pBest_i(t)$. The global best position of the whole swarm is expressed as $gBest(t)$. Therefore, the ith swarm particle updates its own speed and position according to the following equations:

$$V_i(t+1) = w \times V_i(t) + \{C_p \times r_1 \times (pBest_i(t) - X_i(t))\} + \{C_g \times r_2 \times (gBest(t) - X_i(t))\} \quad (1)$$

$$X_i(t+1) = X_i(t) + V_i(t+1) \quad (2)$$

where Cp is the Cognitive learning rate and Cg is the Social learning rate. The factors r_1 and r_2 are randomly generated within the range {0, 1} and w is the inertia factor. Eq. (2) updates the position of the particle while Eq. (1) updates the velocity and is composed of three components. The first component is the former speed, $V_i(t)$of the swarm particle, which shows its present state; the second component is the cognition modal, which expresses the thought of the individual swarm particle itself; the third component is the social modal. These three parts together determine the solution space searching ability. The first component balances the whole and searches local region. The second component empowers the swarm particle to search the whole search space and avoid local minimum. The third component is a reflection of the global influence on the particle. Under the influence of these three components, the swarm has good coverage and tends to converge at globally optimum positions. There are numerous methods for solving multiple objective problems using the PSO algorithm (Parsopoulos and Vrahatis, 2002; Hu and Eberhart, 2002; Hu et al., 2003; Coello Coello and Lechuga, 2002; Pulido and Coello Coello, 2004). One of the most common approaches is to aggregate all the objective functions with appropriate weights into a single objective function (Parsopoulos and Vrahatis, 2002). This requires problem analysis to assign appropriate weights for each of the objective functions. The nature of the current problem and the objectives being considered compels separate evaluation of these objectives. Hence, we've used Vector Evaluated PSO proposed by Parsopoulos and Vrahatis, 2002.

3.1. Vector evaluated PSO (VEPSO)

VEPSO is a multi-swarm variant of PSO, is an approach to multi-objective optimization (Reyes-Sierra and Coello Coello, 2006), which is inspired by the vector evaluated genetic algorithm (VEGA) (Schaffer, 1985). In VEPSO, two or more swarms are employed to probe the search space and information is exchanged among them (Parsopoulos and Vrahatis, 2002). The key issue in these co-evolutionary algorithms is that the fitness of an individual in a population depends on the individuals of a different population. Searching capabilities of the algorithm using co-evaluation techniques enhance the capability of the VEPSO algorithm to better explore the search space (Parsopoulos and Vrahatis, 2002; Shang-Jeng Tsai et al., 2010). The salient features of VEPSO are explained in detail below:

The VEPSO method assumes that M swarms $S_1,\ldots,S_M$ each of size N aim to optimize simultaneously M-objective functions. Each swarm is exclusively evaluated according to one of the objective functions. Let ${}^{[j]}S_i(t+1)$ be the new updated position, ${}^{[j]}V_i(t+1)$ the new updated velocity, ${}^{[j]}P_{i(t)}$ the current personal best position, ${}^{[j]}S_{i(t)}$ and ${}^{[j]}V_i(t)$ the previous position and velocity of the ith particle in the jth swarm, respectively, at a given time t. Let ${}^{[k]}P_{gb}(t)$ be the global best position of the kth swarm at the same time t. Then the VEPSO swarms are updated according to Eqs. (3) and (4)

$$\begin{aligned}{}^{[j]}V_i(t+1) = {}^{[j]}k \times \Big[& {}^{[j]}w_i \times {}^{[j]}V_i(t) + {}^{[j]}C_p \times r_1 \times \{{}^{[j]}P_i(t) - {}^{[j]}S_i(t)\} \\ & + {}^{[j]}C_g \times r_2 \times \{{}^{[k]}P_{gb}(t) - {}^{[j]}S_i(t)\}\Big] \end{aligned} \tag{3}$$

$${}^{[j]}S_i(t+1) = {}^{[j]}S_i(t) + {}^{[j]}V_i(t+1) \tag{4}$$

where the superscripts represent the PSO parameters for the jth swarm. Cp is the Cognitive learning rate and Cg is the Social learning rate. The factors r_1 and r_2 are randomly generated within the range {0, 1} and w_i is the inertia factor. The VEPSO assumes that the search behavior of a swarm is affected by a neighbouring swarm—kth swarm. The parameter k is determined by the information migration scheme between the multiple swarms.

4. Application of VEPSO for design optimization

In the present work, the design of a composite plate, simply supported on all four edges, subjected to uniformly distributed load and point load is considered. The design is governed by the arrangement of constituent plies that make up the composite plate, the stacking sequence, and the stresses and strains developed on each ply of the composite plate are obtained from classical laminate plate theory (CLPT). Using these stresses and strains, failure of a particular design configuration is checked using Maximum Stress and Tsai-Wu failure criteria. The solution to the application problem involves selecting that stacking sequence which minimizes both the weight and cost of the designed composite plate and at the same time passes the above mentioned failure check. This design configuration being composed of many variables with wide ranges renders the solution space enormous. To reduce the time taken to find the optimum or near optimum solution, VEPSO is made use of. Two different swarms, with objectives of minimizing weight and cost, respectively, are deployed in the search space. Initially, the particles of both the swarms are assigned a randomly generated design configuration. With this, the particles communicate with other particles of their own swarm as well as certain particles from the other swarm, progressively exploring the solution space, until all of them converge at a point which coincides with the globally optimum solution. In the remaining part of this section the experimental setup, how the CLPT, in tandem with failure criteria, is used to check for the feasibility of designing composite plates and how VEPSO benefits the process of selecting a cost and weight effective plate configuration among many feasible configurations has been discussed.

4.1. The composite laminate plate design

The Composite laminate plate considered in the present work is composed of a Carbon/Epoxy FRP. The material properties of which are listed inTable 2.

The mathematical model, similar to the one used in Omkar et al. (2008), for a plate simply supported on all four edges is discussed in Section 4.1.1 with a detailed description of the structural analysis in Section 4.1.2. From this mathematical formulation we obtain the deflections for different types of loading conditions from which we get the stresses required to evaluate if the design satisfies various failure criteria.

4.1.1. Simply supported rectangular plate

A thin rectangular composite laminated plate of length a in *x-direction,* width b in the y-direction, and thickness h in the z-direction as shown in Fig. 1 is considered in the present work.

The plate is assumed to be constructed of arbitrary number, n, of linearly elastic orthotropic layers. Each layer consists of homogeneous fiber reinforced composite material and the plate is simply supported on all four edges. A rectangular Cartesian coordinate system x, y and z is used to describe the infinitesimal deformations of an n-layer laminated composite plate. The laminate consists of n plies with the individual thicknesses h_i for the layer orientation angles $\theta_i (i=1,2,3\ldots,n)$ as shown in Fig. 1. Total thickness of the plate is h and bottom and top surfaces are located at $z=-h/2$ and $z=h/2$, respectively. We assume that the middle surface of the undeformed plate coincides with the *xy-plane.* The principal fibre direction is oriented at an angle θ to the x-axis.

4.1.2. Composite plate structural analysis

Classical lamination plate theory (CLPT) is used to develop an analytical solution for a specially orthotropic carbon/epoxy composite laminate plate with Stress Strain relations for a symmetric angle ply laminate. Navier methods are employed for designing of rectangular composite plate with all four edges of the plate as simply supported. The mathematical model and the equations governing static bending in absence of in-plane and thermal forces are

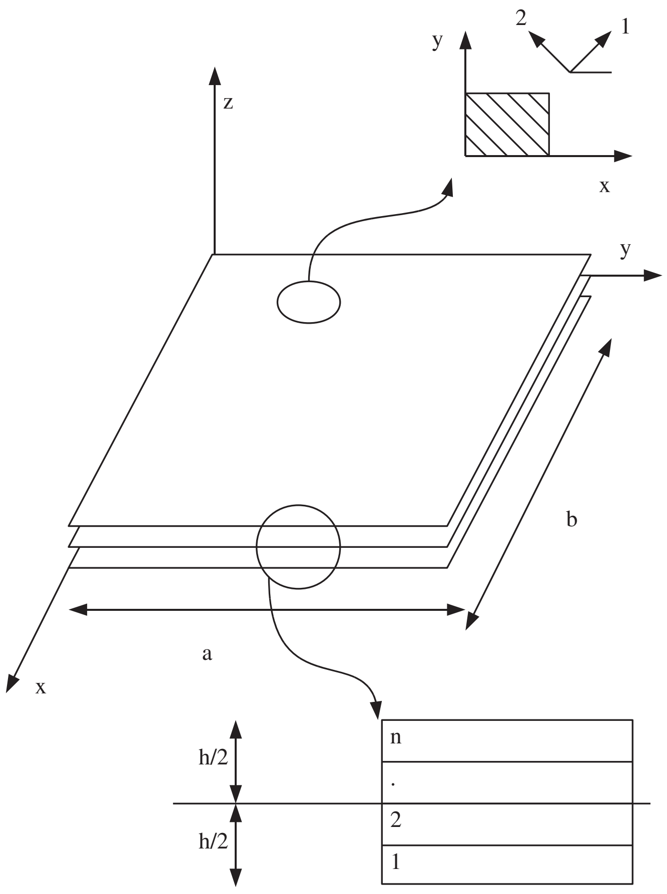

Fig. 1. Geometry of a simply supported thin laminated plate.

elaborately discussed in Robert Millard (1999) and relations between mechanical properties of the plates (for example—fibre orientation) and the stresses and strains induced in them are established.

4.2. *Types of loading*

The Rectangular composite laminate plate used design for this work is subjected to two types of transverse loading conditions; uniformly distributed and point load and the deflections it undergoes under these loads yield the stresses that are need to evaluate their feasibility through failure criteria.

4.2.1. *Uniformly distributed load*

As show in Fig. 2 under this type of setting, the load is uniformly distributed over the surface are of the composite plate and is defined by the function $q(x, y)$. The deflections which the plate undergoes is again discussed in[Robert Jones, Mechanics of composite materials (Section 5.3.1)].

4.2.2. *Point load*

For a setup shown in Fig. 3 the load is concentrated at a single point on the surface and is given by the point load function $q(x, y)$. The deflections which the plate undergoes is discussed in Robert Millard, (1999).

4.3. *Failure criteria*

The Laminate stress analysis of Sections 4.2 combined with lamina failure criteria predicts failure in the laminate. Laminate failure is the eventual result of progressive failure processes taking place in the constituent laminas under loading. Failure envelopes are predicted by linear laminate analysis, providing an extremity for the solution, and defining a specified boundary and solution space ensuring a failsafe design. Various failure criteria proposed by different researchers have been used as an effective tool in evaluation of stress strain behavior and design of composite laminates (Shi and Eberhart, 1998) and for this work's analysis Maximum Stress and Tsai-Wu failure criteria are used.

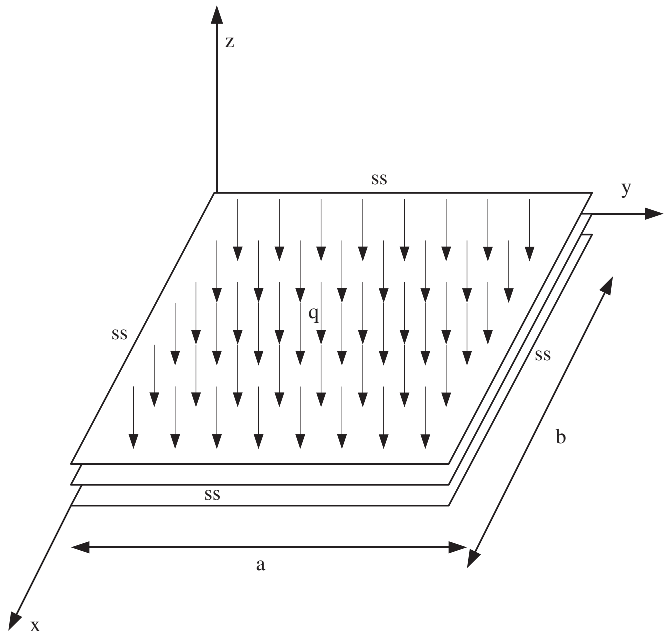

Fig. 2. Orthotropic composite plate subjected to uniformly distributed load (UDL).

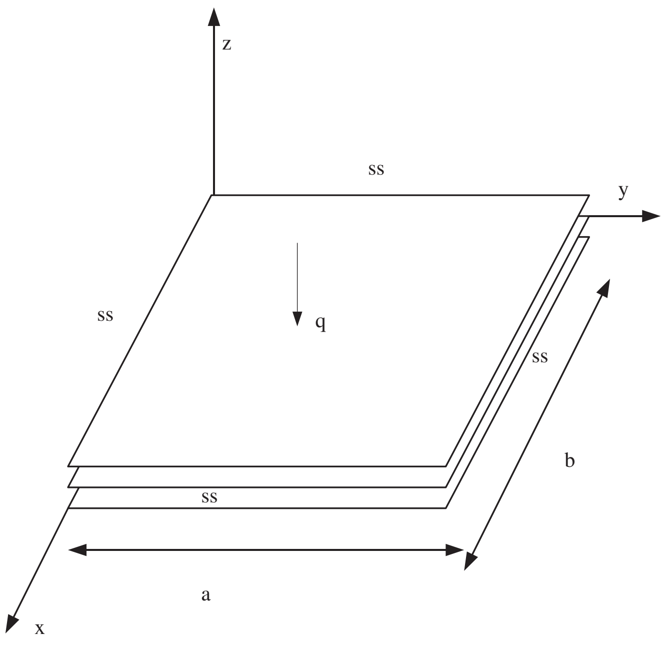

Fig. 3. Orthotropic composite plate subjected to point load (PL)

4.3.1. *Maximum stress failure criteria*

The composite plate ply fails when one of the following conditions is violated (Omkar et al., 2008).

$$X_c \geq \sigma_{xx} \leq X_t, Y_c \geq \sigma_{yy} \leq Y_t, -S \geq \sigma_{xy} \leq S \tag{5}$$

where X_T, X_C, Y_T, Y_C are strengths of the lamina in X–Y directions. S-shear strength of the lamina and σ_{xx}, σ_{yy} and σ_{xy} are the stresses induced in the principal direction. Eq. (5) stipulates the condition for non-failure for any particular ply of composite laminated plate.

4.3.2. *Tsai-Wu failure criterion*

Tsai-Wu brings in more complexity from a computational perspective and further reinforces Maximum Stress Failure criterion's evaluation. This failure theory is based on the total strain energy failure theory of Beltrami. This is a generic model proposed by Tsai-Wu et al. Narayana Naik et al., 2011. The failure theory states that the lamina failure occurs when the following condition is satisfied.

$$F_{LL}\sigma_L^2 + F_{TT}\sigma_T^2 + 2F_{LT}\sigma_L\sigma_T + F_L\sigma_L + F_T\sigma_T + F_{SS}\sigma_{LT}^2 > 1 \tag{6}$$

$$F_{LL} = \frac{1}{X_L X_C}; F_{TT} = \frac{1}{Y_T Y_C}; F_L = \frac{1}{(X_T - X_C)};$$
$$F_T = \frac{1}{(Y_T - Y_C)}; \quad F_{SS} = \frac{1}{S^2}; \quad F_{LT} = -0.5(F_{LL}F_{TT})^{0.5}$$

where, F_{LL}, F_{TT}, F_{LT}, F_L, F_T, F_{SS} are the strength parameters. X_c, X_t, Y_c and Y_t are the laminate compressive and tensile strengths in X, Y directions, respectively. S is the shear strength of the component in the X–Y plane. This criterion predicts the immanency of failure but not the failure modes.

4.4. *Optimization problem*

Once the failsafe set of design configurations are obtained the problem progresses into finding those configurations that yield minimum weight and cost for the composite plate. To do so the design variables which make up the solution space, the constraints imposed on these sets of variables, the objective functions the govern the selection of a set of values for these variables, the algorithm that dictates the selection and progression of design configurations are discussed in following sections.

4.4.1. *Design variables*

The Design Variables in the present problem are the number of plies in each orientation, the number of layers, and thickness of each layer. The number of plies placed at different orientation angles with specific order of arrangement is known as stacking sequence. In our problem, a variable stacking sequence consisting of fiber orientation angles within the range of $(-90^\circ, +90^\circ]$ in steps of 15° is considered, leading to 12 different possible values of orientation angle θ

$$\theta=[-75^\circ/-60^\circ/-45^\circ/-30^\circ/-15^\circ/0^\circ/15^\circ/30^\circ/45^\circ/60^\circ/75^\circ/90^\circ]$$

Here, the number of layers present at each of the different fiber orientation angles are considered as the actual design variables for the optimization process. The thickness of the lamina, a control variable in the plate optimization is also considered, within the range of 0.05 mm to 0.5 mm. In this paper, it is assumed that each layer is of the same thickness t, retaining the practicality of the solution. The number of plies in each orientation generally varies between 1 and 200 which effectively translates to well over 200^{12} different possible configurations rendering a huge search space.

4.4.2. *Objective functions*

Multi-objective optimization indicates the use of two objective functions for the current problem as minimization of weight and total cost of the composite laminate. A conflict of objectives may arise where variation in one may affect the other owing to the formulation of these objective functions. This nature is purely to be based on the parameters used in the governing equations of the objectives. The design variables considered are number of plies, thickness of plies and orientation of the plies.

4.4.2.1. The weight function. The Weight of the laminate is found out by Eq. (7) with the total weight of the laminate being directly proportional to the number of layers and layer thickness t. Total height h of the composite plate of thickness t mm is given by:

$$h=\left[\left\{\sum_{i=1}^{12} n_{\theta_i}\right\}\times t\right] \quad Weight,\ W_t=\rho\times h\times a\times b \tag{7}$$

where, ρ—density of the material of the composite plate

4.4.2.2. The cost function. The composite plate is optimized with respect cost function 'g', which is formulated using the cost function developed by Kovacs et al., (2004) for carbon-fiber-reinforced plastic (CFRP) sandwich-like structure with aluminum (Al). The cost in the design optimization of a composite structure is of major importance attributed to the high costs of the composite materials available in the market. The material cost attributes to the raw materials used for the composite plates. The manufacturing cost is a direct function of time associated with manufacturing of the composite plate, which includes the time lost in press form preparation, layer cutting, layer sequencing and final working. The various costs incurred in the manufacture of a composite laminate in reality are to be considered. The total cost of the laminate is given by Eq. (8) below,

$$Total\ cost=Material\ Cost+Manufacturing\ Cost$$

$$g=g_{matl}+g_{manufact}$$

$$g(x)=[g_{matl}\{W_t\}+g_{maf}\{(\sum n_{\theta_i}\times 20)+(\sum n_{\theta_j}\times 120)+140\}] \tag{8}$$

Considering cost due to manufacturing costs associated with non standard orientation ply angles then the total manufacturing cost is given by,

$$Manufacturing\ cost=\mu\times(standard\ orientation\ plies)+\gamma\times(non{-}standard\ orientation\ plies)$$

The indices g_{matl} and $g_{manufact}$ are determined based on the material being used and the type of manufacturing process employed. For our work we have used $\gamma/\mu=6$. This large fraction is deliberately used to make sure that during optimization the heuristic shows reduced inclination towards selecting non-standard plies.

4.5. *Mathematical representation*

Let the set of permutations in the search space be represented by,

$$P=(n_{\theta_1},n_{\theta_2},\ldots,n_{\theta_{12}})$$

For different values of n_{θ_i} from the search space such that $0<n_{\theta_i}\le 200$.

If $W(p)$ and $C(p)$ are the weight and cost objective functions of permutation $p\in P$, then in summary, the problem can be seen mathematically as finding that permutation S, such that

$S\in P$
$W(S)\le W(p)\forall p\in P.$
$C(S)\le C(p)\forall p\in P.$

And the strength of S is greater than the minimum allowed, i.e., the conditions mentioned in Section 4.3 are not violated.

5. The need for VEPSO

The problem at hand looks to obtain a composite laminate which satisfies certain physical constraints such that it bears an applied load without failure. While doing so we deploy a heuristic that looks to minimize the weight and corresponding cost of laminate obtained. These objective functions can in simple terms be summarized as follows:

1. Weight function ($W=h^*\times a^*\times b^*\times\rho$ where ρ=density $h=\sum\theta_i{}^*t$ and t=thickness of each of the plies)
 a. weight $\propto$ number of plies.
 b. weight $\propto$ thickness of plies.
2. Cost Function (C=(cost of each ply)$\times$number of plies$+$ constant$\times$(weight of material))
 a. cost $\propto$ number of plies.
 b. cost $\propto$ net weight of the composite laminate.

The results obtained by using PSO, assuming that cost gets minimized when weight gets minimized, to optimize weight of the composite laminate shows the following observations for two trials as seen in Table 1.

Overall for the many trials we conducted, we observed that the optimum weight obtained for the problem ranges between approx. 81–85 units while corresponding costs ranges between approx. 700,000–400,000 units. This large variation in cost can be explained by the variation of the number of the plies, required to withstand the same applied load, of varying thicknesses, varying between 0.05 mm to 0.5 mm (Table 2).

Table 1
Optimum weights and corresponding costs for PSO.

Trial number	Optimum weight	Corresponding cost	Thickness
1	81.046	673,909	0.05
2	80.985	450,418	0.30

Table 2
Carbon/Epoxy FRP material properties.

Elastic moduli of laminate in (GPa)			Lamina poisson's ratio			Rigidity moduli of laminate (GPa)		
E_{xx}	E_{yy}	E_{zz}	V_{xz}	V_{xy}	Vyz	G_{xy}	G_{yz}	G_{xz}
Longitudinal	Transverse	Normal	*xz* direction	*xy* direction	*yz* direction	*xy* direction	*yz* direction	*xz* direction
126	11	11	0.28	0.28	0.4	6.6	3.93	6.6

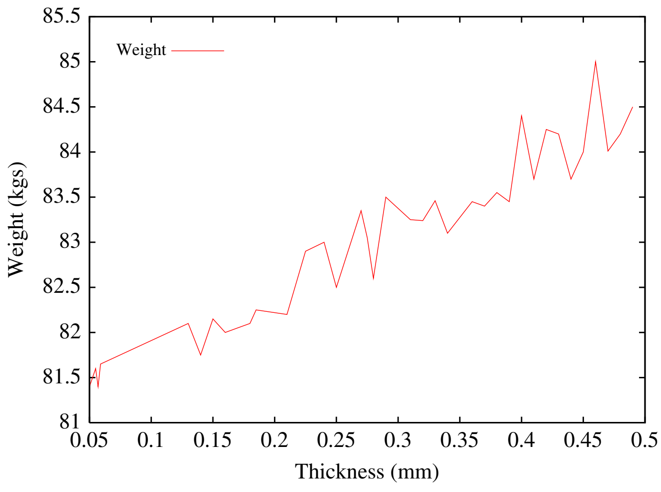

Fig. 4. Variation of cost (corresponding to optimum weight) with thickness.

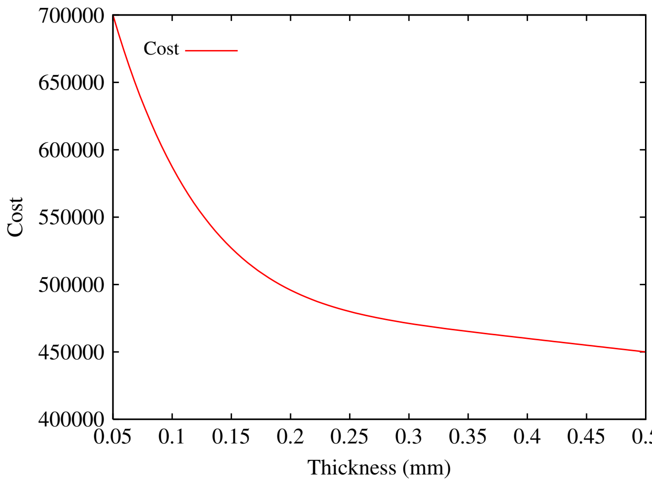

Fig. 5. Variation of optimum weight with thickness.

To observe the variation of cost on thickness basis, we have conducted trials keeping the thickness constant for each trial. The results of these trials for different thicknesses have been show in Figs. 4 and 5:

As one can see from these plots, the variation of the cost corresponding to optimum weight can vary quite drastically depending on the thickness of the plates. This essentially means that minimizing weight does not necessarily minimize weight. In fact there is a likelihood that any of the cost values in the range of 400,000–700,000 units may accompany the optimum weight obtained. This can be explained by cost being composed of the material component and the manufacturing component and this manufacturing component depends on the number of plies. As thickness decreases, the number of plies required to bear the applied load increase, which increases the manufacturing component of the cost function thus increasing the total cost.

These results are the impetus behind us choosing VEPSO to separately evaluate the cost and weight functions and minimize them. Results have shown that choosing VEPSO has worked well because we have obtained minimized cost and minimized weight with certainty for the trials conducted.

6. The optimization process using VEPSO

Vector evaluated particle swarm optimization in the current work has been modified for constrained non-linear optimization problems with discrete design variables unlike previous works carried out for optimizing systems with continuous variables (Gies and Rahmat-Samii, 2004; Vlachogiannis and Lee, 2005). The key point in the constrained optimization process is dealing with the constraints associated with decision variables. In the current work, the constraints are effectively handled to preserve the feasibility of the solutions evolved. In order to constrain the optimum solution to the failure criteria constrained solution space, each particle is made to search the solution space keeping track of only the feasible solutions. Further, to increase the likelihood of finding more of such feasible solutions the particles are initialized within the feasible solution space. This process, however, does consume quite some time. The design variables involved in the current optimization problem are discrete in nature. The twelve variables corresponding to the number of layers at each of the twelve different fiber orientation angles are integers specified range making them discrete and finite in nature. Further, the layer thickness (t) is also considered to be a discrete variable, capable of taking values between the specified ceiling and floor limits with a least count of 0.001 mm. This consideration has been taken in view of retaining the practicality of the evolved solution in terms of limitations posed on fabrication.

VEPSO employs two swarms to probe the search space and information is exchanged among them. Each swarm is exclusively evaluated with one of the objective functions, but, information coming from other swarm(s) is used to influence its motion in the solution space. Specifically, in this case since there are two objective functions, two swarms (X_1, X_2) of N particles each are used. X_1 evaluates the weight objective function and X_2 evaluates the cost objective function. There is no necessity for a complicated information migration scheme between the swarms as only two swarms are employed. Each swarm is exclusively evaluated according the respective objective function. The best particle of the second swarm (X_2) is used for calculation of the new velocities of the first swarm's (X_1) particles and accordingly the best particle of the first swarm (X_1) is used for calculation of the new velocities of the second swarm (X_2). The particles' velocity and position update Eqs. (9) and (10) for the first swarm—X_1

$$\begin{aligned}{}^{[X_1]}V_i(t+1) &= {}^{[X_1]}k \\ &\times [{}^{[X_1]}w_i \times {}^{[X_1]}V_i(t) + {}^{[X_1]}C_p \times r_1 \times \{{}^{[X_1]}P_i(t) - {}^{[X_1]}S_i(t)\} \\ &\quad + {}^{[X_1]}C_g \times r_2 \times \{{}^{[X_2]}P_{gb}(t) - {}^{[X_2]}S_i(t)\}]\end{aligned} \tag{9}$$

$${}^{[X_1]}S_i(t+1) = {}^{[X_1]}S_i(t) + {}^{[X_1]}V_i(t+1) \tag{10}$$

The particles' velocity and position update Eqs. (11) and (12) for the first swarm—X_2

$$^{[X_2]}V_i(t+1) = {}^{[X_2]}k \times [{}^{[X_2]}w_i \times {}^{[X_2]}V_i(t) + {}^{[X_2]}C_p \times r_1 \times \{{}^{[X_2]}P_i(t) - {}^{[X_2]}S_i(t)\} + {}^{[X_1]}C_g * r_2 * \{{}^{[X_1]}P_{gb}(t) - {}^{[X_2]}S_i(t)\}] \tag{11}$$

$$^{[X_2]}S_i(t+1) = {}^{[X_2]}S_i(t) + {}^{[X_2]}V_i(t+1) \tag{12}$$

The particles of both the swarms (X_1, X_2) move in solution space according to the above mentioned equations. After an empirically derived maximum number of iterations or upon convergence at a point or small region in the solution space by a majority of the swarm, the optimization process is terminated and the results of the best particles of both swarms are reported. This maximum number is dependent on the number of particles employed for optimization process. The greater the number of particles the fewer iterations required for them to converge.

The performance of the PSO is very sensitive to the control parameter choices (Shi and Eberhart, 1998; Engelbrecht, 2005). For our work, which uses VEPSO, we have tried different variations of these parameters and we have used those values which produces better convergence in terms of speed and quality. The number of swarm particles is decided empirically based on the limiting number of particles that make a difference in the quality of the solution obtained. The increase in the number of particles deployed is stopped when the increase makes no difference to the solution. Twice the number of dimensions of the problem is taken as the number of particles for each swarm. As the current problem is 13-Dimensional, 26 swarm particles are used for both the swarms as this number has been empirically found to be sufficient in effectively converging on the near optimum solution, found thus far in the trials conducted, in a relatively short time. During initialization, it is ensured that all the particles are within the failure criteria constrained solution space. So initialization itself may take a longer time if the population size is too large. Hence a lower population size significantly lowers the computational time. Further the remaining parameters; the inertia weight w, cognition learning rate Cp and the social learning rate Cg are also fixed based various trials which allow better convergence rate and greater coverage of solution space. The same PSO parameters as in (Shi and Eberhart, 1998) have been the used for each swarm and for all simulation runs in this VEPSO work too. Here, the inertia weight parameter w_i is adjusted dynamically during the optimization, as suggested by Shi and Eberhart (1998). A starting value of $w_i=1$ is used to initially accommodate a more global search and is dynamically reduced to $w_i=0.4$. The idea behind this approach is to terminate the PSO algorithm with a more local search. The w_i value is adaptively allocated as per Eq. (13).

$$w_i = w_{max} - \left[\{\frac{w_{max} - w_{min}}{i_{max}}\} \times i\right] \tag{13}$$

where w_{max} is the initial weight factor, w_{min} is the final weight factor, 'i' is the current iteration number and i_{max} is the maximum number of iterations. The initial higher value may result in greater population diversity in the beginning of the optimization, whereas at a later stage lower values are favoured, causing a more focused exploration of the search space.

The parameters shown in Table 3 have been used for the same reasons as in Omkar et al. (2008).

Table 3
The VEPSO parameters.

Individualistic factor	**p_incr=1.5**
Socialistic factor	**g_incr=1.5**
Inertia factor	**w_i=[1,…,0.4], *adaptively allocated*** (decreasing from 1 to 0.4 with each iteration)
Number of Swarm Particles	***N*=26**
Maximum number of iterations	**Max_it=100 iterations**
End condition (Number of iterations without update in the best values)	**20 iterations**

7. Message passing interface (MPI) and parallelization

MPI (message passing interface) is a specification for a standard library which is used for message passing between concurrent processes on distributed systems. The MPI standard defines only one API (or three to be more precise, one each for FORTRAN, C, C++ and C#). Every super-computer manufacturer offers its own implementation, optimized for its own hardware. MPI forms the basis of a standard high level communication environment featuring collective communication, point-to-point communication.

The current work involves decomposing the serial computation, which mainly consists of running computations of particles of the swarm one after another in a serial fashion, into parallelized chunks on basis of particles of the swarm. Hence the computations pertaining to each particle is run in parallel on different nodes of cluster computer. To enable communication and synchronization among them made particles/nodes MPI's collective communication calls are made use of.

7.1. Synchronization

Particle swarm optimization algorithm requires results from different particles to be assessed and decisions for the ensuing calculations are based on them. To allow results from calculations performed different particles on different nodes of the cluster of a particular iteration to be completed and to allow temporary cessation of calculations until certain results are obtained some means of synchronization is required. Essentially, this ensures that before the swarm moves to the updating phase the fitness evaluations of all the particles are completed and assessed. This is taken care by the MPI collective routine MPI_ALLREDUCE which temporarily stops the coordinating node from proceeding with the next swarm iteration until all of the computational nodes have responded with a fitness value. This, however, implies that the time required for a single parallel swarm fitness evaluation will be dictated by the slowest fitness evaluation in the swarm.

7.1.1. Use of MPI_ALLREDUCE for synchronization and broadcast of global best

MPI_ALLREDUCE combines multiple values from all processes and distributes the result of the operation on all these values specified in the call back to all processes. When the personal best of all the particles of the swarms are computed, we need to calculate the global best and this global best value should be made available to all. This is possible through the use of MPI_ALLREDUCE with a minimization operator to help find the minimum of the personal bests and distribute this value to all the processes. To make available the entire configuration of the global best position, by distributing all the co-ordinates of that position the following procedure has been employed.

1. Agree on the global best within the swarm using MPI_REDUCE (If used with a minimization operator, this function call allows each process to receive a single, minimum of all data sent by different processes).
2. Obtain this global best and compare with particle's current best.

3. If global best does not match the particle's current best then change all values of the particle's copy of global to a very large value else retain values.
4. Now perform MPI_ALLREDUCE on this set of global bests of the particles.
5. The end result is that every particle is left with a copy of the current iteration's global best.

7.2. *Summary of the algorithms*

Outline of the VEPSO Algorithm used for the composite design optimization problem

1. For each particle of the weight and cost swarms assign random values sequentially for the number of plies each of the 12 orientations and a fractional random value (between 0.05 and 0.5) for the thickness.
2. Modify these values until all the particles of both swarms have all been initialized with a configuration that meets design constraints and satisfy failure criteria.
3. Perform fitness evaluations on the particles of both swarms.
4. Update particle's personal best on comparison with current evaluation.
5. From the above set, extract the best results of the personal bests of all particles of both swarms and disseminate this information among all particles of its peer swarm.
6. Each particle updates its positions on obtaining best values from neighboring swarms after the mutual exchange of information between swarms.
7. If termination conditions are met then go to step 8 else step 3.
8. Report and exit.

7.2.1. *Serial implementation*

```
I Initialization
  1. $i=1$ where 'i' is the particle index.
  2. $j=1$ where 'j' is the dimension index.
  3. Randomly initialize particle positions ($x_{ij}\in X_1$ and $y_{ij}\in X_2|x_{ij}$, $y_{ij}\in N$) for the weight and cost variables, respectively.
  4. Randomly initialize thickness for each particle ($x_{ij}\in X_1$ and $y_{ij}\in X_2|x_{ij}$, $y_{ij}\in[0.05,0.5]$) if $j=13$.
  5. if $j<14$ increment j and go to step 3.
  6. if $i<M$ (particle population) increment i and go to step 2.
II Meeting design constraints
  1. $i=1$.
  2. Evaluate fitness of particle at $x_i$ and $y_i$. If f and g, the the cost and weight fitness at these points is such that strength of the configuration < minimum allowable strength then go to step 3 else step 6.
  3. $j=1$.
  4. Increment $x_{ij}$ and $y_{ij}$ suitably.
  5. If $j<14$ then increment j and go to 4.
  6. If $i<M$ then increment i and go to step 2.
III Optimization
  1. $j=1$ (iteration index), $i=1$ (particle index)
  2. Evaluate fitness of $f_{ij}$ and $g_{ij}$ for positions $x_i$ and $y_i$, respectively.
  3. If $j=1$ then evaluate $f_{i1}$ and $g_{i1}$ and $f_{ibest}$ and $g_{ibest}$ are $f_{i1}$ and $g_{i1}$, respectively else it is the minimum of the best and current evaluations, i.e., if $f_{ij}<f_{ibest}$ then $p_i=x_i$ and if $g_{ij}<g_{ibest}$ then $q_i=y_i$ where p and q personal bests of weight and cost swarm particles.
  4. If $i<N$ then increment i and go to step 2.
  5. $i=1$.
  6. Update $x_i$ and $y_i$ from the best information of neighboring swarm.
  7. If $i<N$ then increment i and go to step 6.
  8. If termination conditions not met then increment j and go to step 2.
IV Report results and terminate
```

7.2.2. *Parallel implementation*

```
III. Initialization
  1. 'i' is the particle index so concurrently for $i=1,2,\ldots,M$
         do
         {
  2. $j=1$ where 'j' is the dimension index.
  3. Randomly initialize particle positions ($x_{ij}\in X_1$ and $y_{ij}\in X_2|x_{ij}$, $y_{ij}\in N$) for the weight and cost variables, respectively
  4. Randomly initialize thickness for each particle ($x_{ij}\in X_1$ and $y_{ij}\in X_2|x_{ij}$, $y_{ij}\in[0.05,0.5]$) if $j=13$
  5. If $j<14$ increment j and go to step 2
     }
III. Meeting design constraints
  1. Concurrently for $i=1,2,\ldots,M$ evaluate fitness of particle at $x_i$ and $y_i$. If f and g, the fitness at these points is such that strength of the configuration < minimum allowable strength then go to step 2 else go to step 5
         do
         {
  2. $j=1$
  3. increment $x_{ij}$ and $y_{ij}$ suitably
  4. if $j<14$ then increment j and go to 3
     }
  5. Synchronize
III. Optimization
  1. $j=1$(iteration index)
  2. Concurrently for $i=1,2,\ldots,M$ evaluate fitness of $f_{ij}$ and $g_{ij}$ for positions $x_i$ and $y_i$, respectively
         do
         {
  3. if $j=1$ then evaluate $f_{i1}$ and $g_{i1}$ and $f_{ibest}$ and $g_{ibest}$ are $f_{i1}$ and $g_{i1}$, respectively else it is the minimum of the best and current evaluations, i.e., if $f_{ij}<f_{ibest}$ then $p_i=x_i$ and if $g_{ij}<g_{ibest}$ then $q_i=y_i$ where p and q personal bests of weight and cost swarm particles
     }
  4. Synchronize
  5. Concurrently for $i=1,2,\ldots,M$ update $x_i$ and $y_i$ from the best information of neighboring swarm.
  6. If termination conditions not met then increment j and go to step 2.
III. Report Results and terminate
```

The flow diagrams of the serial and parallel algorithms have been shown in Figs. 6 and 7.

8. Results and discussion

This work is primarily concerned with design of a parallel VEPSO algorithm for the multi-objective design optimization of laminated composite plates problem and to test its efficacy, in terms of execution time and coherence, it has been compared with sequential VEPSO. To further test how our work fares against other popular parallel heuristics we have compared it with parallel vector evaluated genetic algorithm (PVEGA) designed for the same problem. The structural problem for our work is similar to a previous work (Omkar et al., 2008) but has two different variations, which is in the use of uniform distributed load and point load.

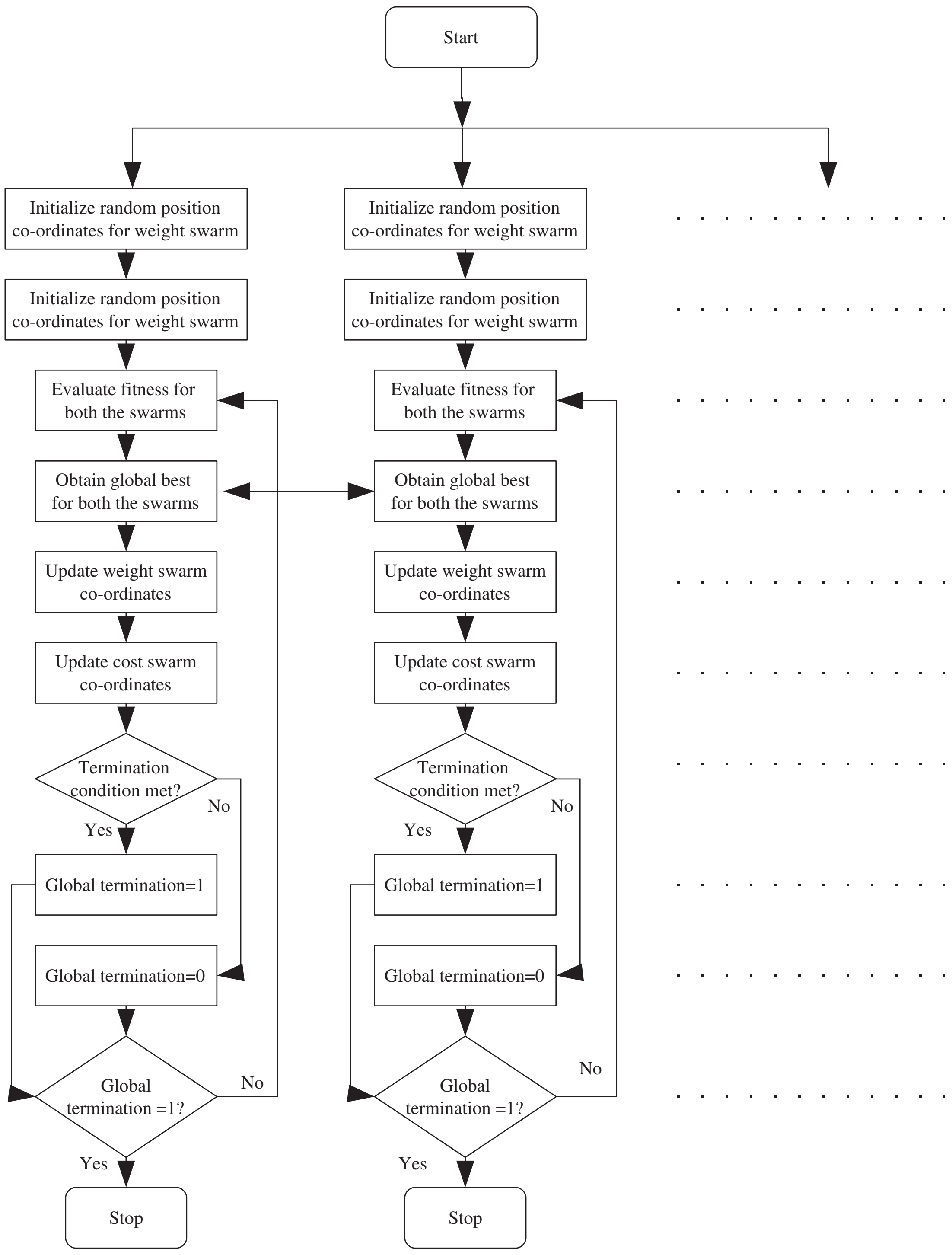
Start
Initialize random position co-ordinates for weight swarm
Initialize random position co-ordinates for weight swarm
Evaluate fitness for both the swarms
Obtain global best for both the swarms
Update weight swarm co-ordinates
Update cost swarm co-ordinates
Termination condition met?
Yes
No
Global termination=1
Global termination=0
Global termination =1?
Yes
No
Stop

Fig. 6. Parallel implementation of the optimization problem.

8.1. *Experimental platform*

To obtain these results, the parallel algorithm described in Section 7 has been executed on an IBM 720 Cluster with the following specifications:

i. Sixty four 4-way SMP nodes (256 processors) P720 open power systems.
ii. IBM Power-5 systems operating at 1.65 GHz.
iii. 4-GB main memory per node with a total of 256 GB for the cluster.
iv. Dual Gigabit network with Nortel 5510 gigabit switches.
v. SUSE Enterprise Linux 9.0 operating system.

In tandem with the hardware mentioned above, MPICH v.1.2.7, a high performance portable implementation of MPI (Coello Coello and Sierra, 2004), has been used as the software platform to implement our parallel algorithm. As a metric of comparison we have exclusively used speedup. The speedup *s* is defined as the ratio of the serial execution time to the parallel execution time which gives us an indication of how much faster the parallel is than the serial approach.

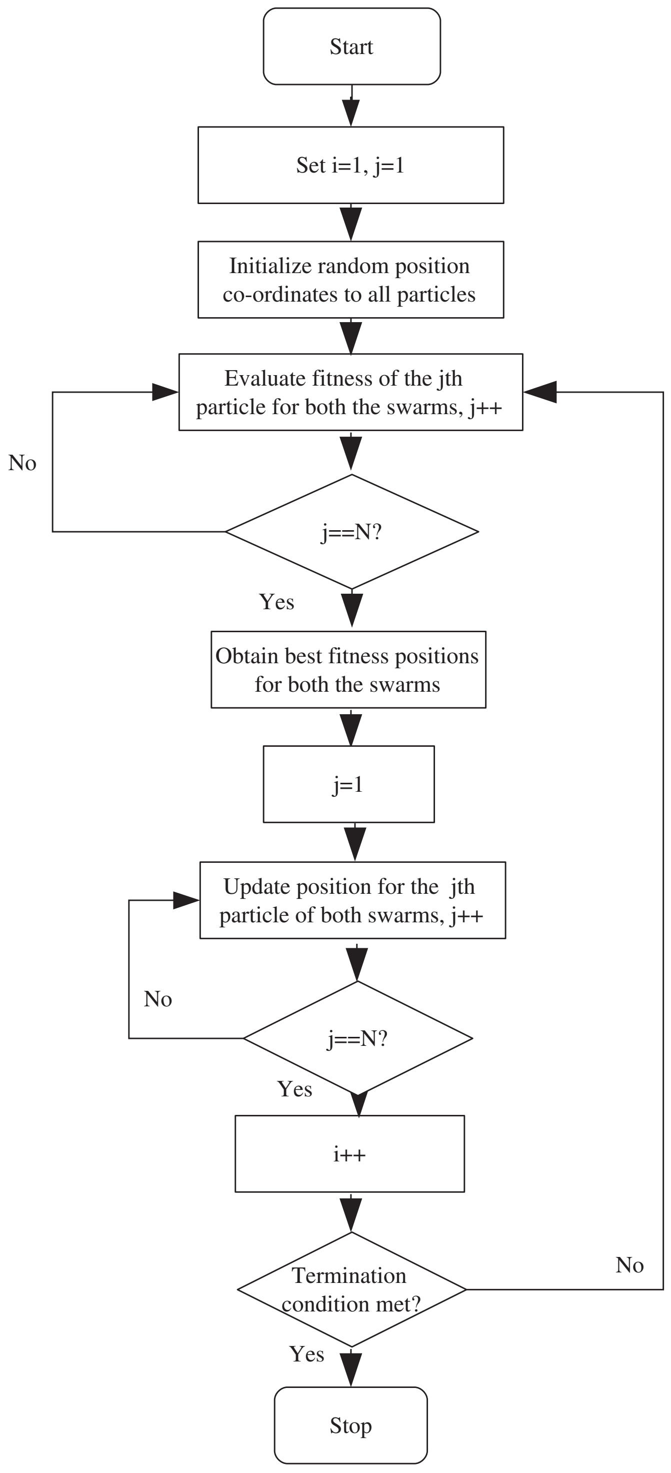
Start
Set i=1, j=1
Initialize random position co-ordinates to all particles
Evaluate fitness of the jth particle for both the swarms, j++
No
j==N?
Yes
Obtain best fitness positions for both the swarms
j=1
Update position for the jth particle of both swarms, j++
No
j==N?
Yes
i++
Termination condition met?
No
Yes
Stop

Fig. 7. Serial implementation of the optimization problem.

$s = T_{Serial}/T_{Parallel.}$
Where, T_{serial} is the time taken for serial execution.
$T_{Parallel}$ is the time taken for parallel execution.

8.2. Comparison of parallel VEPSO and sequential VEPSO

We would like to emphasize that, despite the relatively short execution time of the serial algorithm, the intent of this work is primarily to seek further improvement in execution time by parallelization and to explore the suitability of the peer-to-peer paradigm with MPI collectives. This paradigm is relatively unused for the problem domain and the results show its efficacy by showing how the model scales with particle size which motivates its application for more expensive problems in the domain.

For the serial version of the multi-objective design optimization of laminated composite plates using VEPSO we have used the same hardware platform as the parallel one but the program is run on a single node on a single processor whereas the parallel algorithm has used all of the available processors when necessary. The codes developed for both the algorithms use the same governing mathematical equations and models to optimize the solution. On the software front, however, we have used an *xlc* compiler for the serial version and *mpcc* compiler, for the parallel program.

The key strategy employed by the sequential approach is that the computations pertaining to each particle of the swarm, involved in the optimization process, in executed one after another obviating the needs of set synchronization points. This is in stark contrast to the strategy employed by the parallel approach which leverages on the available parallel processors as described in Section 7. As expected we found marked reduction in the time taken for the parallel algorithm and have managed speedups of up to **10x**. Figs. 8–11 show coherence results and Figs. 12–20 show the plots for execution times and speedup curves for comparison of parallel

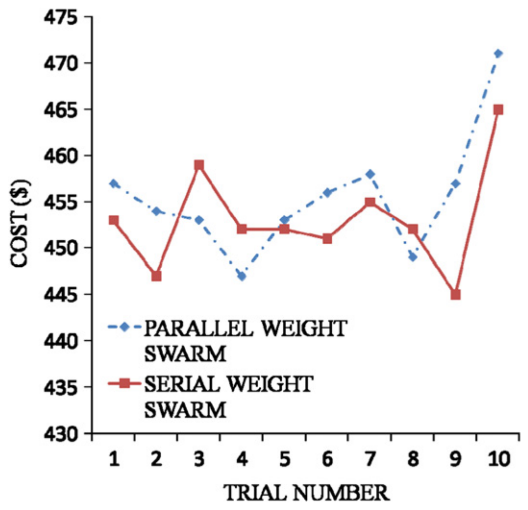

Fig. 8. Optimum cost obtained by serial and parallel versions of the weight swarm.

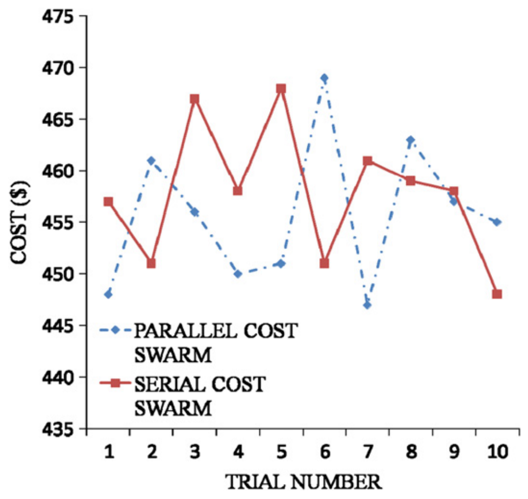

Fig. 9. Optimum cost obtained by serial and parallel versions of the cost swarm.

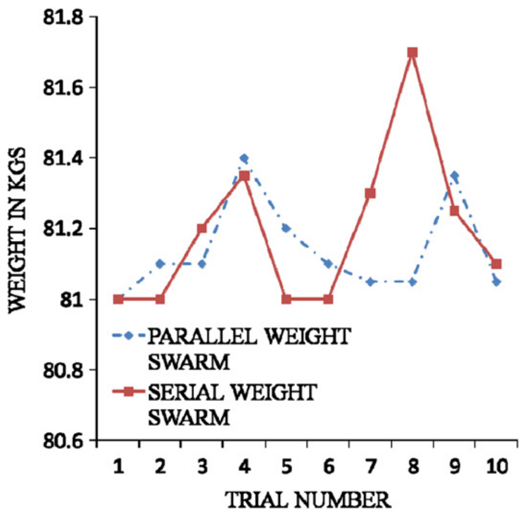

Fig. 10. Optimum weight obtained by serial and parallel versions of the weight swarm.

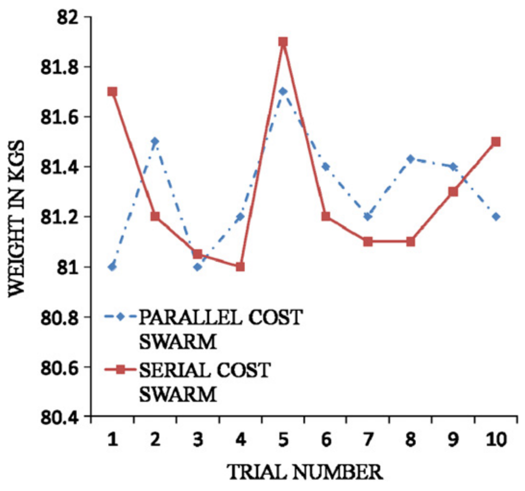

Fig. 11. Optimum weight obtained by serial and parallel versions of the cost swarm.

VEPSO with sequential vector evaluated approaches, respectively. The cases have been briefly discussed below with an emphasis on coherence first and then on speedup.

8.2.1. Coherence

MPI is an API which is defined in *C* and standard *C* libraries are compatible for use with MPI libraries. The essence of this is that instruction sets and accuracies of operations remain the same. The serial version also being written in *C* makes it furthermore feasible to achieve absolute coherence. Although the parallel algorithm does not parallelize serial algorithm verbatim owing to programming paradigm shifting from the procedural with the introduction of MPI, the essence of the algorithm logic is preserved and parallel results show nearly complete coherence with serial version.

Figs. 8–11 depicts the variation in optimum cost and the weight values of the serial and parallel implementations for a test case of the composite laminate problem subjected to uniformly distributed load evaluated with maximum stress failure criterion, respectively. The curves in the figures represent the values obtained for parallel and the serial implementations, respectively. The curve profile indicates small variations in both the optimal weight values and cost values obtained for both serial and parallel implementations. In the cost evaluation there is a maximum variation of approximately 4.25% between the parallel and serial results for the trials conducted for both cost and weight swarms. In evaluating weight there is an even smaller variation of approximately1.25% between the parallel and serial results. This variation can be considered negligible for all practical purposes and they become insignificant when a mix of large number of trials are considered as the parallel and serials results become more coherent on an average.

8.2.2. Performance measurements

Case 1. Uniformly distributed load with maximum stress failure criterion.

Fig. 12 show both the execution times of serial and parallel implementation for the composite laminate problem subjected to a Uniformly Distributed Load (UDL) of 0.5 N/mm^2 with maximum stress failure criterion. The variation of the serial and the parallel running time plots indicate the latter's execution time reduction and Fig. 13 speedup curve verifies the same.

Case 2. Point load with maximum stress failure criterion.

Fig. 14 show the execution time of both serial and parallel implementation for the composite laminate plate problem subjected to a Point Load (PL) of 3000 N/mm^2 with maximum stress failure criterion. We observe results almost similar to the previous case with the parallel outperforming the serial in almost all cases. The corresponding speedup can be seen in Fig. 15.

Case 3. Uniformly distributed load with Tsai-Wu failure criterion.

Fig. 16 show the comparison of the serial and parallel execution times for the optimization of composite laminate plate subjected to Uniformly Distributed Load (UDL) with Tsai-wu failure criterion. Similar results have been obtained with exception of the actual computation times, which has generally increased owing to increased computational complexity brought in by Tsai-Wu failure criterion. The speedup is shown in Fig. 17.

Case 4. Point load with Tsai-Wu failure criterion.

Figs. 18 and 19 show the execution times of serial and parallel implementation for the composite laminate plate subjected to a Point Load (PL) with Tsai-wu failure criterion and the corresponding speedup curve.

8.3. Comparison of parallel VEPSO with parallel VEGA

Comparison of the serial versions of both the nature inspired optimization techniques—*VEPSO* and *VEGA* along with the comparison of different failure criteria evaluated for different loading conditions for multi-objective design optimization of composite laminated structures evaluated using different failure criteria is presented by Narayana Naik et al. (2011) and Omkar et al. (2008). An effective comparison of serial versions of PSO and GA is carried out by Eberhart and Shi (1998). PSO has proven a flexible and well-balanced mechanism to enhance and adapt to the global and local exploration and exploitation abilities within a short calculation time (Eberhart and Shi, 1998). Our previous work regarding the comparison between serial PSO and GA (Narayana Naik et al., 2011) prompted the comparison of their parallel counterparts for this application and hence in this section, the proposed parallel approach of VEPSO is compared with a parallel version of Vector Evaluated Variant of the GA employed to solve the composite laminate problem under uniformly distributed load conditions

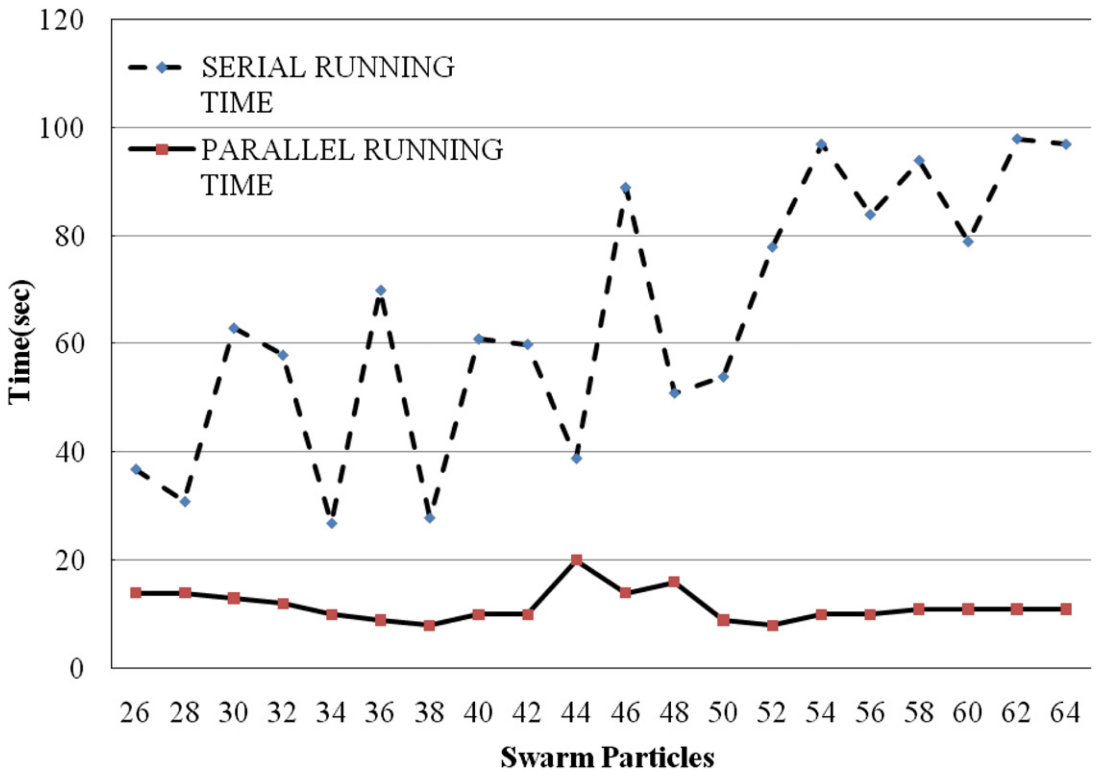

Fig. 12. Serial and parallel running time for uniformly distributed load with maximum stress failure criterion.

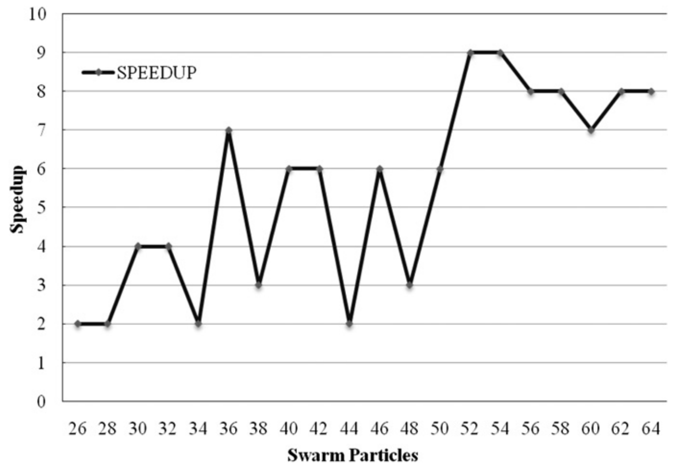

Fig. 13. Speedup for uniformly distributed load with maximum stress failure criterion.

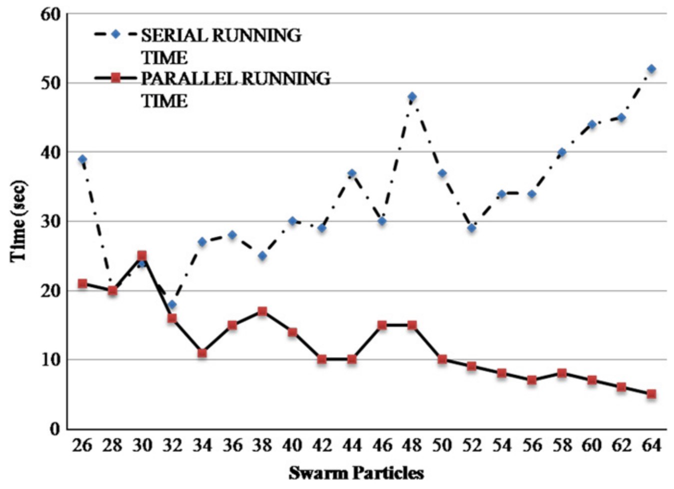

Fig. 14. Serial and parallel running time for point load with maximum stress failure criterion.

evaluated with Maximum Stress Failure criterion. The parameters that govern the nature of the VEGA are consistent with those used in Narayana Naik et al. (2011). Again the logical aspects of GA are perfectly preserved but some modifications have been made to accommodate MPI's programming model.

8.3.1. *Outline of the parallel vector evaluated genetic algorithm*

The parallel VEGA algorithm developed for this application behaves for most part like a traditional genetic algorithm except that two populations are deployed to minimize the weight and cost. Hence, at the end of each generation, there is a high probability of choosing the design configurations of the two candidates with best fitness of particular population are used by the other population for ensuing generation and vice versa. This process is repeated until sufficient convergence is observed.

MPI's MPI_Allgather collective communication primitive has been used instead of MPI_ALLREDUCE. This is because in genetic algorithm there is a chance that even the design configuration of the worst candidate may be chosen as the parent for a particular offspring of the next generation, however unlikely such an event is although be it a highly improbable event. Hence design configurations of the whole population are important in VEGA as opposed to just the best candidate, as in the case of VEPSO. MPI_Allgather enables synchronization and dissemination of information of all candidates to each process and hence each pair of processes uses the same two distinct parents from the whole population using the information available locally to generate

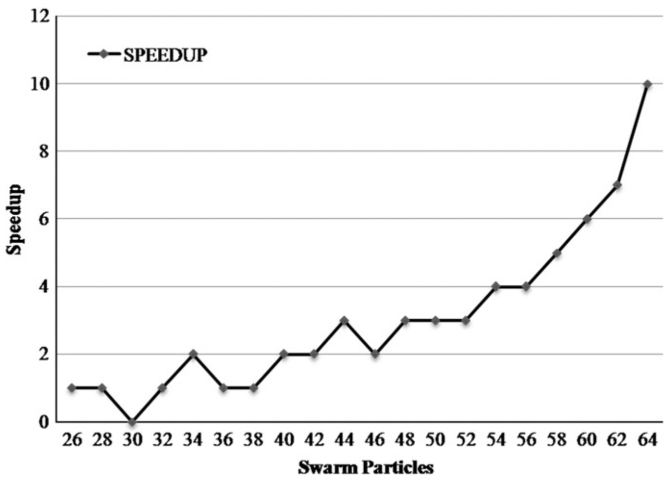

Fig. 15. Speedup for point load with maximum stress failure criterion.

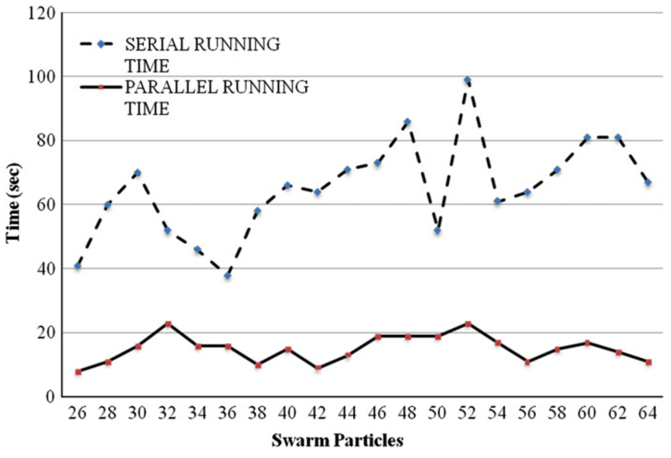

Fig. 16. Serial and parallel running time for uniformly distributed load with Tsai-Wu failure criterion.

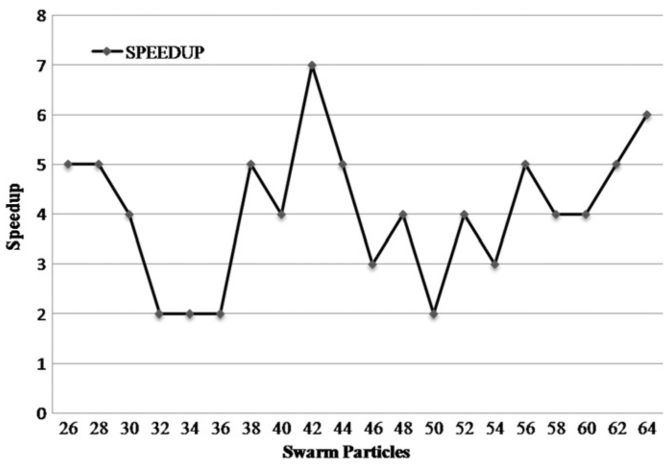

Fig. 17. Speedup for uniformly distributed load with Tsai-Wu failure criterion.

offspring of the next generation. Fig. 20 presents this idea more clearly. For convenience we will refer to each individual in a population as particle. The algorithm's flowchart is presented in Fig. 20.

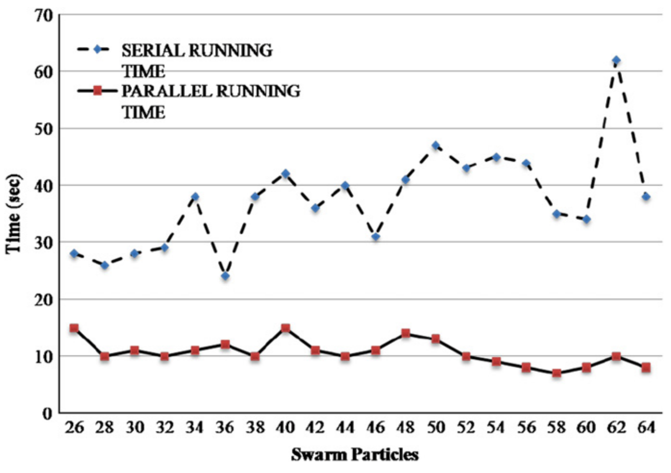

Fig. 18. Serial and parallel running time for point load with Tsai-Wu failure criterion.

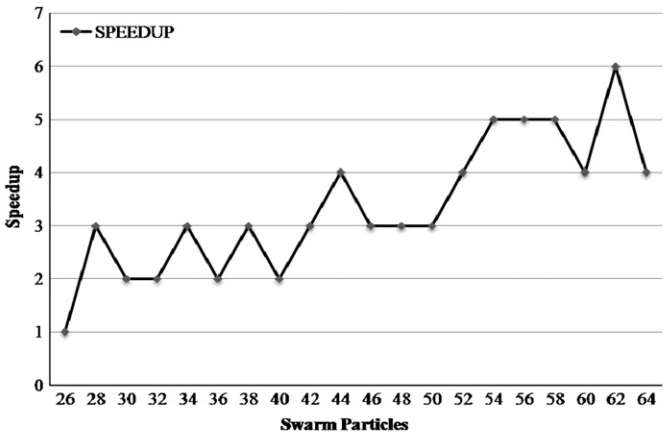

Fig. 19. Speedup for point load with Tsai-wu failure criterion.

```
I  Initialization
   1. 'i' is the particle index so concurrently for $i=1,2,\ldots,M$
          do
          {
   2. $j=1$ where 'j' is the dimension index.
   3. Randomly initialize particle positions ($xij$ $X_1$ and $yij$ $X_2|xij$, $yij$
      $N$) for the weight and cost variables, respectively.
   4. Randomly initialize thickness for each particle ($xij$ $X_1$ and $yij$
      $X_2|xij$, $yij$ [0.05,0.5]) if$j=13$.
   5. If $j<14$ increment j and go to step 2
      {
II Meeting design constraints
          Concurrently for $i=1,2,\ldots,M$ evaluate fitness of particle at $x_i$ and $y_i$. If $f$ and $g$, the fitness at these points
          is such that strength of the configuration < minimum allowable strength then go to step 2 else go to step 5
          do
          {
   1. $j=1$
   2. increment $x_{ij}$ and $y_{ij}$ suitably
   3. if $j<14$ then increment j and go to 3
      }
   4. Synchronize
III Optimization
   1. $j=1$(iteration index)
   2. Concurrently for $i=1,2\ldots..,M$ evaluate fitness of $f_{ij}$ and $g_{ij}$ for
      positions $x_i$ and $y_i$, respectively
```

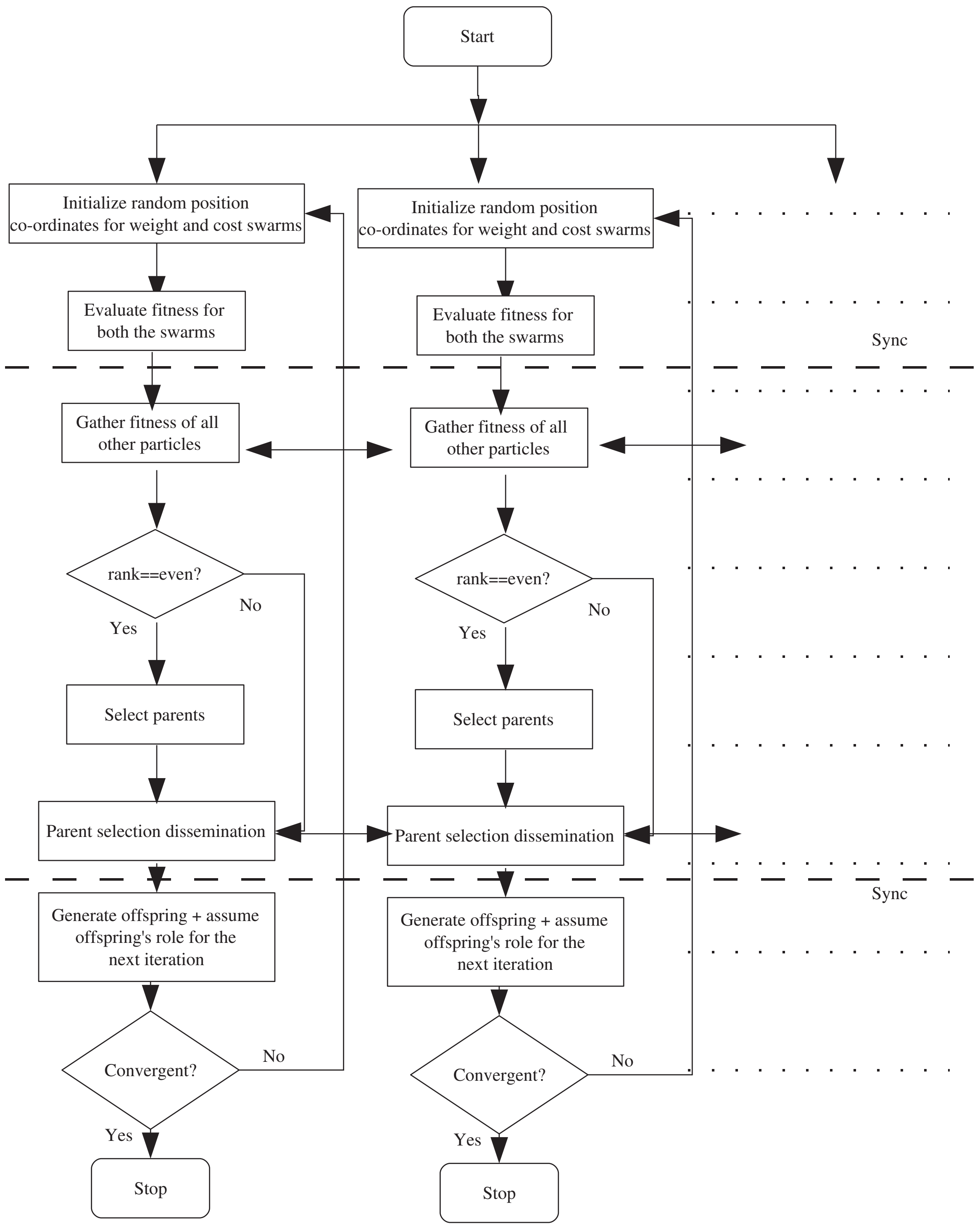
Start
Initialize random position co-ordinates for weight and cost swarms
Evaluate fitness for both the swarms
Sync
Gather fitness of all other particles
rank==even?
Yes
No
Select parents
Parent selection dissemination
Sync
Generate offspring + assume offspring's role for the next iteration
Convergent?
No
Yes
Stop

Fig. 20. Parallel implementation of VEGA.

do

{

3. *if $j=1$ then evaluate f_{i1} and g_{i1} and f_{ibest} and g_{ibest} are f_{i1} and g_{i1}, respectively else it is the minimum of the best and current evaluations,* i.e., *if $f_{ij} < f_{ibest}$ then $p_i = x_i$ and if $g_{ij} < g_{ibest}$ then $q_i = y_i$ where p and q personal bests of weight and cost swarm particles*

}

4. *Synchronize*
5. *Gather fitness and design configuration data of each particle (using MPI_Allgather)*
6. *If the rank of the particle is even then choose parents from the other swarm with the probability of choosing parents influenced by their fitness values. (Odd ranked nodes do nothing here).*
7. *Disseminate parent selections using MPI_Allgather (This lets each particle know their parents' design configurations). (Alternatively each even rank can send patent information to*

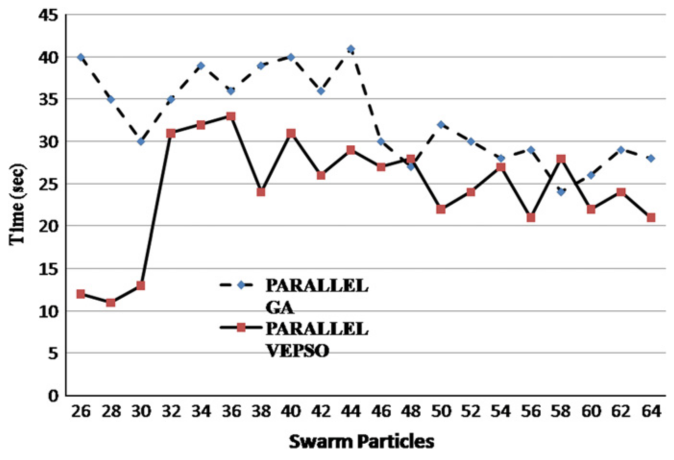

Fig. 21. Comparison of parallel VEGA and parallel VEPSO for uniformly distributed load with maximum stress failure criterion.

corresponding odd rank but this has to be followed by a barrier. There is little difference in the two methods in terms of execution time).

8. *Use these parent configurations and generate offspring with a degree of mutation.* (*Here the two ranks transform from being parents to being the next generation's offspring*).
9. *if termination conditions* (*number of iterations*) *not met then increment j and go to step* 2.

IV **Report Results and terminate**

Fig. 21 depicts the execution times for both parallel VEPSO and VEGA, respectively. The curve profile of both the parallel VEPSO and parallel VEGA indicates that in most cases parallel VEPSO fares slightly better than VEGA in terms of their execution times which is explained by the simplicity in the nature of VEPSO algorithm. The GA employing fairly heavier communication, which involves each particle getting to know about every other particle's fitness and parent selections, results in fairly significant difference in execution times when smaller populations are used but as the population size is increased the communication overhead becomes the dominating factor in execution time of both algorithms and the execution time of the two algorithms seem to converge.

9. Conclusions

In our work we have developed a novel parallel approach to VEPSO algorithm, which captures the essence of the peer-to-peer paradigm model of communication and synchronous evaluations, for the design optimization of composite structures using MPI parallel programming platform. MPI being widely available allows its collective communication protocol to be used for a range of problems where peer-to-peer paradigm is intended to be used. The results show reduction in the running time by a considerable extent, achieving speedups of up to 10x while maintaining the same quality of solutions that the sequential approach yields. Our approach has also shown reduced execution time compared to a parallel approach Vector evaluated GA for a single case of the composite problem. Parallel VEPSO has fared better than sequential VEPSO and parallel VEGA in the cases we have investigated. The approach provides for faster selection of optimum stacking sequence corresponding to the design of composite laminates with the objectives of minimizing weight and cost. The parallel approach clearly indicates the increase in speedup for adequate numbers of processors allocated. The parallel algorithm developed has shown to be scalable for increased particle sizes and populations. From the results obtained, we can conclude that, if each particle of the swarm is allocated to a dedicated processor then the parallel program's execution time remains nearly invariant. For increased swarm populations, the parallel algorithm remained almost invariant in execution time while the serial approach tended to linearly increase indicating the algorithm's scalability. Our future attempts will be in the direction of trying out parallel algorithms which incorporate higher degrees of parallelism enabling parallelization of each particle of the swarm using NVIDIA's Compute Unified Device Architecture (CUDA) platform and also adopting this work's methods for longer execution problems in the domain.

References

Adali, S., Walker, M., Verijenko, V.E., 1996. Multi-objective optimization of laminated plates for maximum pre-buckling, buckling and post-buckling strength using continuous and discrete ply angles. Compos. Struct. 35 (1), 117–130.

Bin Yua, Zhongzhen Yang, Chuntian, Cheng, 2007. Optimizing the distribution of shopping centers with parallel genetic algorithm. Eng. Appl. Artif. Intell. 20 (2), 215–223.

Bova, S.W., Carey, G.F., 2000. A distributed memory parallel element-by-element scheme for semiconductor device simulation. Comput. Meth. Appl. Mech. Eng. 181 (4), 403–423.

Boyang, Liu, Haftka, Raphael T., Akgün, Mehmet A, Akira, Todoroki, 2000. Permutation genetic algorithm for stacking sequence design of composite laminates. Comput. Meth. Appl. Mech. Eng. 186 (2–4), 357–372.

Coello Coello C.A., Lechuga M.S., 2002. MOPSO: a proposal for multiple objective particle swarm optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC); pp. 1051–1056.

Coello Coello, C.A., Sierra, M.R.., 2004. A study of the parallelization of a coevolutionary multiobjective evolutionary algorithm. In: Ranroy, Gustavo Arroyo-Figueroa, Luis Enrique Sucar, Humberto Sossa (Eds.), Proceedings of the Third Mexican International Conference on Artificial Intelligence (MICAI'2004), Lecture Notes in Artificial Intelligence, vol. 2972, pp. 688–697.

Deb, K., 2001. Multi-objective Optimization Using Evolutionary Algorithms. John Wiley and Sons Ltd.

Deka, D.J., 2005. Multiobjective optimization of laminated composites using finite element method and genetic algorithm. J. Reinf. Plast. Compos. 24, 273–285.

Dubreuil, Marc, Gagn, Christian, Parizeau, Marc, 2006. Analysis of a master-slave architecture for distributed evolutionary computations. IEEE Trans. Syst. Man Cybern. Part B 36 (1), 229–235.

Eberhart R.C., Shi Y., 1998. Comparison between genetic algorithms and particle swarm optimization, in Proc. IEEE Int. Conf. Evol. Comput., 611–616.

Engelbrecht, A.P., 2005. Fundamentals of Computational Swarm Intelligence. John Wiley and Sons.

Garey, M.R., Johnson, D.S., 1979. Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, San Francisco.

Ghasemi, M.R., Ehsani., A., 2007. A hybrid radial-based Neuro-GA multiobjective design of laminated composite plates under moisture and thermal actions. World Acad. Sci. Eng. Technol. (28), 356–364.

Gies D., Rahmat-Samii. Y., 2004. Vector evaluated particle swarm optimization (VEPSO): optimization of a radiometer array antenna. In the proceedings of Antennas and Propagation Society—IEEE International Symposium, (3), pp. 2297–2300.

Goel, Tushar, Vaidyanathan, Rajkumar, Haftka, Raphael.T, Shyy, Wei, Queipo, Nestor.V., 2007. Response surface approximation of Pareto optimal front in multi-objective optimization. Comput. Meth. Appl. Mech. Eng. 196 (4–6), 879–893.

Gorlatch., Sergei, 2002. Message passing without send-receive. Future Gener Comp. Syst. 18, 797–805.

Gropp, W., Lusk, E., Skjellum, A., 1994. Using MPI: Portable Parallel Programming with the Message-Passing Interface. MIT Press, Cambridge, MA.

Gurdal, Z., Haftka, R.T., Hajela, P., 1999. Design and Optimization of Laminated Composite Materials. Wiley-Interscience.

Hassan R., Cohanim B.K., Weck O.D., Venter G., 2005. A comparison of particle swarm optimization and the genetic algorithm. Proceedings of the 46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, Austin, TX.

Hempela, Rolf, Walkerb, David W., 1999. The emergence of the MPI message passing standard for parallel computing. Comp. Stand. Interfaces 21 (1), 51–62.

Houzeaux, Guillaume, Codina., Ramon, 2003. A Chimera method based on a Dirichlet/Neumann(Robin) coupling for the Navier–Stokes equations. Comput. Meth. Appl. Mech. Eng. 192 (31–32), 3343–3377.

The MPI Forum, 1995. The MPI Message-Passing Interface Standard. 〈http://www.mcs.anl.gov/mpi/mpi-report/mpi-report.html〉.

Hu, X., Eberhart, R.C., 2002. Multi-objective optimization using dynamic neighbourhood particle swarm optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC), pp. 1677–1681.

Hu, X., Eberhart, R.C., Shi, Y., 2003. Particle swarm with extended memory for multi-objective optimization. In: Proceedings of the IEEE Swarm Intelligence Symposium; pp. 193–197.

Kennedy, J., Eberhart, R., 1995. Particle swarm optimization. IEEE International Conference on Neural Networks, Perth, WA, Australia, pp. 1942–1948.

Kim, C.W., Hwang, W., Park, H.C., Han, K.S., 1997. Stacking sequence optimization of laminated plates. Compos. Struct. 39 (3–4), 283–288.

Kovacs, G., Groenwold, A.A., Jarmai, K., Farkas, J., 2004. Analysis and optimum design of fibre-reinforced composite structures. Struct. Multidiscip. Optim. 28 (2–3), 170–179.

Luersen, M.A., Holdorf, Lopez, R., 2009. Optimization of laminated composite materials using a genetic algorithm". Workshop on Computational Approaches to Material Modeling and Optimization—WCAMMO 2009 Joinville.

Matsuda, Motohiko, Kudoh, Tomohiro, Kodama, Yuetsu, Takano, Ryousei, Ishikawa, Yutaka, 2008. The design and implementation of MPI collective operations for clusters in long-and-fast networks. Cluster Comput. 11 (1), 45–55.

Narayana Naik, G., Omkar, S.N., Dheevatsa, Mudigere, Gopalakrishnan, S., 2011. Nature inspired optimization techniques for the design optimization of laminated composite structures using failure criteria. Expert Syst. Appl., 2489–2499.

Omkar, S.N., Mudigere, Dheevatsa, Narayana Naik, G., Gopalakrishnan, S., 2008. Vector evaluated particle swarm optimization (VEPSO) for multi-objective design optimization of composite structures. Comput. Struct. 86, 1–14.

Parsopoulos, K.E., Vrahatis, M.N., 2002. Recent approaches to global optimization problems through particle swarm optimization. Nat. Comput. 1, 235–306.

Parsopoulos, K.E., Vrahatis, M.N., Particle swarm optimization method in multi-objective problems. Proceedings of the ACM 2002 Symposium on Applied Computing (SAC 2002), pp. 603–607.

Parsopoulos, K.E., Tasoulis, D.K., Vrahatis, M.N.,2004. Multiobjective optimization using parallel vector evaluated particle swarm optimization. In Proceedings the IASTED International Conference on Artificial Intelligence and Applications, as Part of the 22nd IASTED International Multi-Conference on Applied Informatics, Innsbruck Austria.

Pelletier, Jacob L., Senthil, S.Vel, 2006. Multi-objective optimization of fiber reinforced composite laminates for strength, stiffness and minimal mass. Comput. Struct. 84, 2065–2080.

Pulido, G.T., Coello Coello C.A., 2004. Using clustering techniques to improve the performance of a multi-objective particle swarm optimizer. In: Proceedings of the Genetic and Evolutionary Computation Conference, Seattle.

Reyes-Sierra, Margarita, Coello Coello, Carlos A., 2006. Multi-objective particle swarm optimizers: a survey of the state-of-the-art. Int. J. Comput. Intell. Res. 2 (3), 287–308.

Robert Millard, Jones, 1999. Mechanics of Composite Materials, second ed., pp. 279–301.

Schaffer, J.D., 1985. Multiple objective optimization with vector evaluated genetic algorithms. In Genetic Algorithms and their Applications: Proc. Int. Conf. on Genetic Algorithms, pp. 93–100.

Schutte, J.F., Reinbolt, J.A., Fregly, B.J., Haftka, R.T., George, A.D., 2004. Parallel global optimization with the particle swarm algorithm. Int. J. Numer. Methods Eng. 61 (13), 2296–2315.

Schutte, Jaco F., Koh, Byung-I, Reinbolt, Jeffrey A., Fregly, Benjamin J., Haftka, Raphael T., George, Alan D., 2005. Evaluation of a particle swarm algorithm for biomechanical optimization. J. Biomech. Eng. 127 (3), 465–475.

Shang-Jeng, Tsai, Tsung-Ying, Sun, Chan-Cheng, Liu, Sheng-Ta, Hsieh, Wun-Ci, Wu, Shih-Yuan, Chiu, 2010. An improved multi-objective particle swarm optimizer for multi-objective problems. Expert Syst. Appl. 37 (8), 5872–5886.

Shi, Y.H., Eberhart, R.C., 1998. Parameter selection in particle swarm optimization. Evolut. Prog. VII, Lecture Notes Comput. Sci., 591–600.

Snyman, J.A., 2004. Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-based Algorithms. Kluwer Academic Publishers, Dordrect, The Netherlands.

Sung-Kwun, Oh, Han-Jong, Jang, Witold, Pedrycz, 2009. The design of a fuzzy cascade controller for ball and beam system: A study in optimization with the use of parallel genetic algorithms. Eng. Appl. Artif. Intell. 22 (2), 261–271.

Topal, Umut, Uzmana, Umit, 2010. Multiobjective optimization of angle-ply laminated plates for maximum buckling load. Finite Elem. Anal Des. 46 (3), 273–279.

Vlachogiannis, J.G., Lee, K.Y., 2005. Determining generator contributions to transmission system using parallel vector evaluated particle swarm optimization. Power Syst. IEEE Trans. 204, 1765–1774.

William Gropp, Ewing, Lusk, Nathan, Doss, Anthony Skjellum, 1996. A high-performance, portable implementation of the MPI message passing interface standard. Parallel Comput. 22, 789–828.