Optimizing thermal-elastic properties of multi-phase and multi-layer composites by using iterative MapReduce guided genetic algorithm

TAO YOU[a,*], YINGJIE XU[b], CHENGLIE DU[a]
[a]*Institute of Computer Testing, Control and Simulation, School of Computer Science, Northwestern Polytechnical University, Xi'an, 710072, China*
[b]*Engineering Simulation and Aerospace Computing, Key Laboratory of Contemporary Design & Integrated Manufacturing Technology, Northwestern Polytechnical University, Xi'an, 710072, China*

The objective of this work is to optimize the thermal-elastic properties of multi-phase and multi-layer (MPML) composites by controlling the interfaces and matrix layers thicknesses. Aiming at the traditional genetic algorithm (GA) including parallel genetic algorithm (PGA) faces efficiency, scalability, and programming difficulty to solve this kind of optimization problems, we propose an iterative MapReduce guided genetic algorithm (IMGA). This method brings bond into MapReduce, and reasonably allocates various stages of GA to the map and reduce operator, then completes target optimization through multi-step iteration of map and reduce. The IMGA is interfaced with finite element code to find an optimal design for minimizing the coefficient of thermal expansion (CTE) of the MPML unidirectional fiber reinforced composite with constraints of elastic modulus and fiber volume fraction. Satisfactory results are obtained by comparing IMGA and GA.

(Received October 11, 2014; accepted January 21, 2015)

Keywords: Multi-phase and multi-layer, Thermal-elastic properties, Genetic Algorithm, Bond, Iterative MapReduce

1. Introduction

Multi-phase and multi-layer (MPML) composite offers a high potential application in aerospace structures, in which the matrix consists of multiphase materials. In fact, this multi layer architecture can effectively resist the diffusion of oxidizing medium in composite to prevent the oxidation of fibers, and the interfaces between multi material phases can greatly prevent the crack expansion in matrix to improve the toughness of the composite [1]. Typically, continuous carbon fiber reinforced carbon/silicon carbide composite (C/C-SiC) [2-3] and ceramic matrix composite-multi-layer self-healing (CMC-MS) [4-5] are such kind of composite. Fig. 1 is the microstructure photograph of a certain CMC-MS [6]. As can be seen, the multi-layer matrices consisted of $B_{13}C_2$/SiC/$B_{13}C_2$/SiC are distributed around the fiber. With the existence of multi-phase material interfaces, they have the advantage of high oxidation resistance and high toughness. This is why MPML composite has its attractive advantages in aerospace applications.

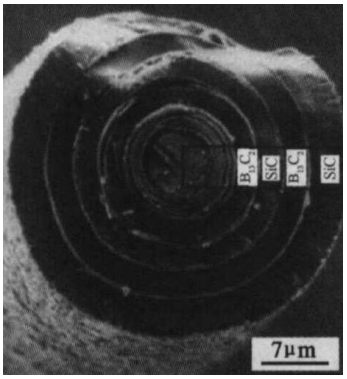


Fig. 1. MPML microstructure of CMC-MS.

MPML composite are usually obtained by using the chemical vapor infiltration (CVI) process to alternately infiltrate the multi-layer material phases. During the CVI process, the thicknesses of layers are controllable. The variation of the thickness of each layer will lead to the geometrical change of the material microstructure, and affect the effective properties of composite as well. How to design the layers thicknesses to obtain composite with the best possible properties is significant for the engineering applications. Under high-temperature environment, the primary concern is to use materials with

low thermal expansion behaviors and high elastic properties. Motivated by this situation, the present paper is directed at optimizing the thermal-elastic properties of MPML composites by using finite element computation and genetic algorithm (GA) with parallel computing scheme.

Optimization methods based on GA have demonstrated the potential to overcome many of the problems associated with gradient-based methods. Even though GA is most effective when the design space is large, high computational time and storage requirement often forces us to work with a reduced design space and thus limiting the efficacy of GA for achieving global optimal solutions. A solution to the above dilemma is to implement GA in a parallel computing environment, where full advantage can be taken of the low communication requirements of GA, and to use specialized models allowing sufficiently detailed representation without excessive computational requirements.

It is well known that writing efficient parallel and distributed applications is complex. Current parallel GA (PGA) [7] either offers high-level programming abstractions, but is not scalable, or achieves scalability by hand-coding models using low-level parallel frameworks. Furthermore, PGAs also face the common development difficulties in distributed environments, such as communication and synchronization between distributed components. Now, due to the increase of cloud computing [8], PGAs have to solve more challenging problems common in data centers, such as heterogeneity and frequent failures. Many existing models for PGAs are based on message passing interface (MPI), which is not designed for cloud computing. Thus, it is important to explore a more suitable solution for performing distributed GAs in data centers.

So the improved parallel PGA algorithms should avoid scientists to write complex and error-prone parallel code, and provide automatically distributing agents to achieve scalability. Recently, Google proposed the MapReduce [9] programming framework to address this complexity, which provides efficiency, scalability and easy-programming ways to solve parallel problems. Many real world tasks can be expressed in this model. The success of the more procedural map-reduce programming model, and its associated scalable implementations on commodity hardware bring lots of advantage to scientific calculation. However, the map-reduce paradigm is too low-level and rigid, and can not be combined with other computation model such as GA algorithm directly. Motivated by this situation, the MapReduce is extended to a new iterative MapReduce in the present work, which is then combined with GA algorithm to develop an iterative MapReduce guided GA (IMGA).

The paper is organized as follows. Modeling study of the thermal-elastic properties of the composite is investigated by finite element method. Then, the optimization problem of minimizing thermal expansion coefficient of the MPML unidirectional fiber reinforced composite is considered with constraints of elastic modulus and fiber volume fraction. Finally, an attempt is made for the first time to use IMGA along with finite element method for carrying out optimal design.

2. Finite element analysis of the MPML unidirectional composite

2.1. Unit cell model

The architectures of MPML unidirectional composite consist of arranged fibres. The components of the multi-layered interfaces and matrix are infiltrated within the porous fibre preforms, according to the CVI process. In the present study, hexagonal fiber arrays are used to model the unidirectional CMCs. Two layers of interfaces and tow layers of matrix are distributed around the fibers. Fig. 2 shows the transverse cross-section of the MPML unidirectional composite. In the longitudinal direction, the fiber axes have been assumed to be parallel and of equal lengths.

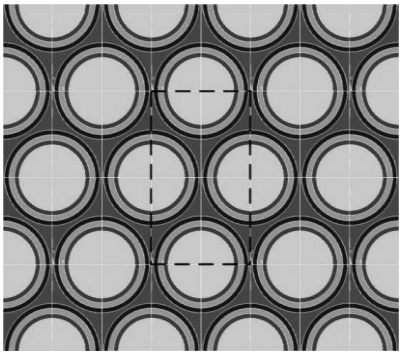


Fig. 2. Cross section of the MPML composite.

The unit cell of composite (as seen in Fig. 3) is used in the present finite element analysis. Characteristic geometric parameters of the unit cell model are given: ϕ^f is fibre diameter, $d_1 \sim d_4$ are thicknesses of the interface and matrix layers.

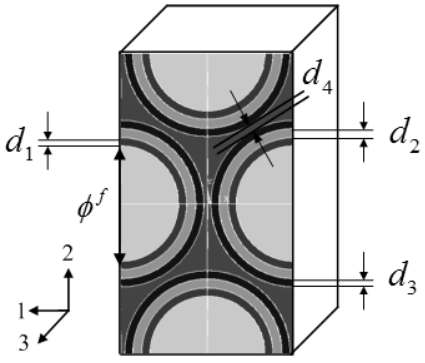


Fig. 3. Geometric parameters of unit cell.

The unit cell model is meshed using the 3D twenty-node, thermal-structural coupled element (SOLID 96) of ANSYS finite element software [10], as depicted in Fig. 4. The number of elements and nodes is 3,840 and 3,986, respectively.

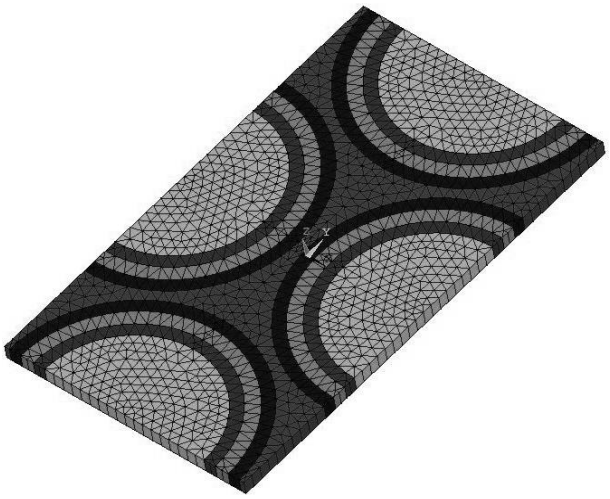


Fig. 4. Finite element model of unit cell.

2.2. Finite computation of the effective properties

2.2.1. Computation of the CTE

The advantages of finite element method are its flexibility in modeling complex shapes, spatial variation of material properties, and its simplicity in numerical implementation. The CTE can be determined by finite element analysis of the thermo-elastic behavior of the composite's unit cell.

The unit cell model is assumed as a perfect elastic body without plastic deformation. The structural and thermal boundary conditions are given as follows: Along the planes $x_1 = 0$, $x_2 = 0$, and $x_3 = 0$, the model is restricted to move in the x_1, x_2, and x_3 directions. Planes $x_1 = l_1$, $x_2 = l_2$, and $x_3 = l_3$ are free to move but have to remain planar in a parallel way to preserve the compatibility with adjacent cells. Suppose the deformation of the unit cell is caused by a temperature rise of ΔT. During the deformation, $x_i = l_i$ becomes $x_i = l_i + \Delta l_i$ and the displacement, Δl_i, can be determined from the finite element analysis. The CTE in direction i then corresponds to

$$\alpha_i = \frac{\Delta l_i}{l_i \Delta T} \tag{1}$$

2.2.2. Computation of the elastic modulus

The specified displacement boundary conditions, such as the tensile and shear displacements, are imposed in the finite element model of unit cell. The elastic properties of the composites can be finally determined by the finite element analysis results of the stresses and strains of the unit cell model. Besides, the periodical displacement boundary conditions are needed to be imposed on the finite element model in order to satisfy forces continuity and displacements compatibility on the opposite faces of the unit cell. More details about the boundary conditions and computation procedure are not included herein for saving the text space, but can be obtained in our previous work [11].

3. Optimization problem

In the present study, the CTE of the MPML unidirectional composite is minimized with constraints of elastic modulus and fiber volume fraction. The material used is T300 carbon fiber/pyrolytic carbon-silicon carbide matrix (C/C-SiC). The unit cell model depicted in Fig. 5 is concerned for optimization, which consists of two layers of interfaces and two layers of matrices made up of alternate pyrolytic carbon and silicon carbide. The fiber diameter ϕ^f is $10.0\mu m$. Properties of each material phase can be found in Table 1 [12].

Table 1. Properties of the constituents.

	E_{11} GPa	E_{33} GPa	G_{12} GPa	G_{23} GPa	v_{12}	v_{23}	α_{11} $10^{-6}/°C$	α_{22} $10^{-6}/°C$	α_{33} $10^{-6}/°C$
T300 carbon fiber	22	220	7.75	4.8	0.42	0.12	8.85	-0.7	-0.7
pyrolytic carbon	12	30	2.0	4.3	0.4	0.12	1.8	-	-
silicon carbide	350	-	145.8		0.2	-	4.5	-	-

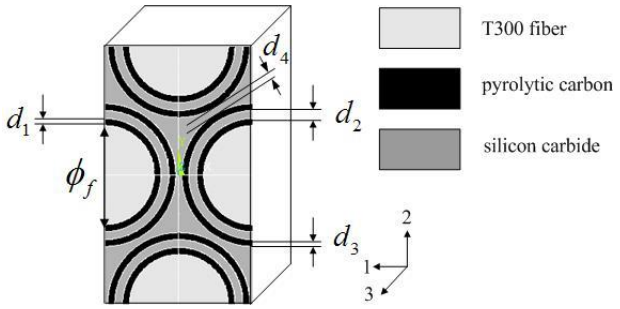


Fig. 5. Unit cell model for optimization.

The design variables are the thicknesses of the interfaces and matrix layers. The optimization problem can be formulated as the follows:

Minimize: $\alpha(d_1, d_2, d_3, d_4)$

Subject to:

$$E(d_1, d_2, d_3, d_4) - E^l \geq 0$$

$$V_f - V_f^0 = 0$$

and

$$d_i^l \leq d_i \leq d_i^u,\ i = 1, ..., 4$$

where $\alpha(d_1, d_2, d_3, d_4)$ and $E(d_1, d_2, d_3, d_4)$ are the extensional thermal expansion coefficient and elastic modulus of MPML composite, respectively. E^l is the given elastic modulus. V_f is the volume fraction of fiber and constrained to be a constant V_f^0. d_i denotes the thickness of ith matrix layer. The values d_i^l and d_i^u are the lower and upper bounds of the thickness of ith matrix layer. The values of constrains and all the variables limits are listed in Table 2.

Table 2. Values of constrains and all the variables limits.

Constraints		Limits of the design variables			
E^l (GPa)	V_f^0	$d_1^l,\ d_2^l$ (μm)	$d_1^u,\ d_2^u$ (μm)	$d_3^l,\ d_4^l$ (μm)	$d_3^u,\ d_4^u$ (μm)
200.0	0.4	0.5	1.0	0.5	1.5

The non-differentiable and nonlinear natures of the above optimization problem induce difficulty in using classical deterministic approaches for solutions. To solve this optimization problem a genetic algorithm is used. GA abstracts the problem space as a population of individuals, and explores the optimum individual through a loop of operations. Usually the individual is represented by a string of symbols, and each step of the loop produces a new generation with reproduction, mutation, evaluation and selection operations. Given a generation of individuals as ancestors, the reproduction operation generates their offspring by combining several ancestors and the mutation operation performs simple stochastic variations on each offspring to generate a new version of it. The evaluation operation evaluates the offspring according to an objective function and the selection operation chooses the best one from the population for next generation. This process repeats until the optimum individual is found.

4. Iterative MapReduce guided genetic algorithm

The implementation of GA to solve the optimization problem of MPML composite involves large numbers of finite element computations for obtaining the final solution. The computational time would be too long if the program is running on a single-processor computer.

Therefore, an efficient distributed and parallel computing strategy is developed in this paper. A new iterative MapReduce that extends the popular MapReduce programming model to iterative computation is proposed and combined with GA to develop an efficient IMGA strategy. By using this approach, large numbers of computations are effectively merged to map process across lots of computing nodes and the computation time is decreased dramatically.

4.1. Iterative MapReduce

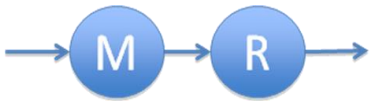
4.1.1. MapReduce programming model

In this section the MapReduce framework is described. Its implementation and refinements were originally

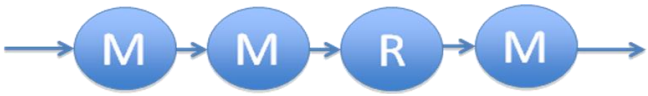
proposed in [13]. In MapReduce framework, the processing is divided into two steps: map and reduce. Map take key/value pair as input and generate intermediate key/value pairs. Reduce merge all the pairs associated with the same (intermediate) key and then generates output. Abstractly, the two phases can be presented as follows. Obviously, MapReduce is a one-step computing model.

$$\begin{aligned} map(K_1,V_1) &\to list(K_2,V_2) \\ reduce(K_2,list(V_2)) &\to (K_3,V_3) \end{aligned} \quad (2)$$

Unfortunately, the map-reduce model has its own set of limitations. Its one-input, two-stage data flow is extremely rigid. To perform tasks having a different data flow, e.g. joins, n stages or iteration, inelegant workarounds have to be devised. Fig. 6(a) shows a classical model that can be processed by MapReduce while Fig. 6(b) displays a chain can not be processed by MapReduce directly. So the procedure of GA, a kind of processing chain, can not be joined with MapReduce directly.



(a) MapReduce



(b) MapReduce Chains

Fig. 6. MapReduce and MapReduce chains.

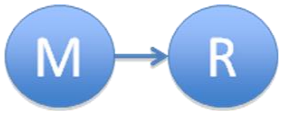
4.1.2. Bring bond to MapReduce

It is well known that MapReduce abstraction comes from the map and reduce primitives present in Lisp [13]. In Lisp and many other functional languages, any functions can be bonded. For example, if function *double* is described as $double(x)=x\times 2$, the function *quadruple* can be described as $quadruple(x)=double(double(x))$. In this example *double* bonds *double* to build a *quadruple* function. Processing two or more functions together is called "bond". So, bond introduces an iterative mechanism to functional languages.

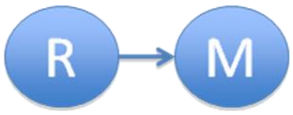
Based on the foundation of bond semantics and as a reaction to the iteration requirements in GA, we designed a new abstraction, iterative MapReduce.

4.1.3. Iterative MapReduce

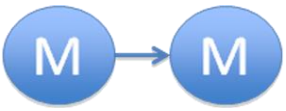
Similar to MapReduce abstraction is inspired by the map and reduces primitives, the design of iterative MapReduce is based on map, reduce and bond primitives. By combining bond with the map/reduce operation, the map/reduce operations can be connected. During the bond step, it can make several different processing results. Fig. 7 shows several kinds of iterative MapReduce. The natural bond between Map and Reduce in classical MapReduce is illustrated in Fig. 7(a).



(a) Natural bond between map and reduce



(b) Bond between reduce and map



(c) Bond between map and map

Fig. 7. Several kinds of bond for iterative MapReduce.

When we bond reduce and map (as shown in Fig. 7 (b)), map take key/value pair from reduce as input and generate intermediate key/value pairs. The programming model can be shown as

$$\begin{aligned} reduce(K_2,list(V_2)) &\to (K_3,V_3) \\ map(K_3,V_3) &\to list(K'_2,V'_2) \end{aligned} \quad (3)$$

As we bond map and map (as shown in Fig. 7 (c)), map1 take key/value pair as input and generate intermediate key/value pairs while map2 take intermediate key/value pair from map1 as input and generate other intermediate key/value pairs. The programming model can be shown as

$$\begin{aligned} map1(K_1,V_1) &\to list(K_2,V_2) \\ map2(K_2,V_2) &\to list(K'_2,V'_2) \end{aligned} \quad (4)$$

In this paper, we use the bond between reduce and map as well as natural bond. Through these two kinds of iterations, we can allocate various stages of the GA to iterative map and reduce operators easily.

4.2. IMGA programming model

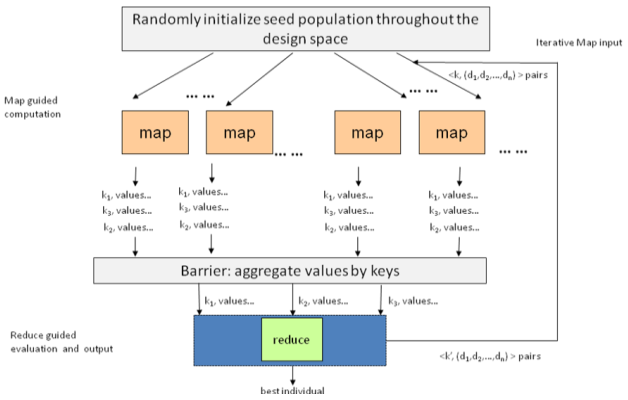


Fig. 8. IMGA procedure.

Based on the basic GA algorithm explained above, the IMGA procedure can be obtained as shown in Fig. 8. The control flow of execution consists of the following stages:

1) The initialization generates offspring and performs mutation.

2) The offspring are split into m pieces respectively for m map tasks. The value of m is chosen so as to maximize parallelism for map tasks. Generally this value is larger than the number of machines.

3) Each piece of offspring is sent to a map task. The map task executes the fitness function for each individual and generates intermediate results.

4) Reduce task merges all the pairs associated with the same (intermediate) key to generate output. When stopping criterion is achieved, reduce task outputs the best solution and ends the procedure. Otherwise, reduce task generates <k', value' > pairs as input for map tasks and begin a new iteration.

The above stages are repeated until the optimum individuals meet the specified requirements. It can be seen that each fitness computation are involved in map procedure, and the various stages of the GA algorithm are combined with iterative MapReduce easily. The pseudo-code for the initialization, map and reduce functions in IMGA algorithm is listed in Table 3.

Table 3. Initialization, map and reduce functions for IMGA procedure.

Initialization()
{
Randomly initialize seed population throughout the design space;
}

map(key,value) (key:the number of individual; value: individual $\alpha\left(d_1, d_2, ..., d_n\right)$ *)*
{
Fitness(key)=Evaluation(value);//Calculate the fitness for individual
Emit(Fitness(key),value); // Emit intermediate output
}

Evaluation(value){ //Calculate α *in some condition as fitness*
if ($E\left(d_1, d_2, ..., d_n\right) - E^l \geq 0 \,\&\, V_f - V_f^0 = 0 \,\&\, d_i^l \leq d_i \leq d_i^u,\ i = 1, 2, ..., n$ *)*
return $\alpha\left(d_1, d_2, ..., d_n\right)$;
}

reduce(key,value_list) (key: fitness of individual; value: individual $\alpha\left(d_1, d_2, ..., d_n\right)$ *)*
{
if fitness > fitnessbest then
    fitnessbest ← fitness; // Fitnessbest is current best fitness value
    vbest ← individual; //vbest is the individual with best fitness value
end if
if vbest do not change for 30 generations then
    Write best individual;
else//Generate offspring
    Do selection, mutation;
    Emit(key', value') for map;
end if
}

5. Numerical tests

5.1. Setup

5.1.1. Architecture of IMGA

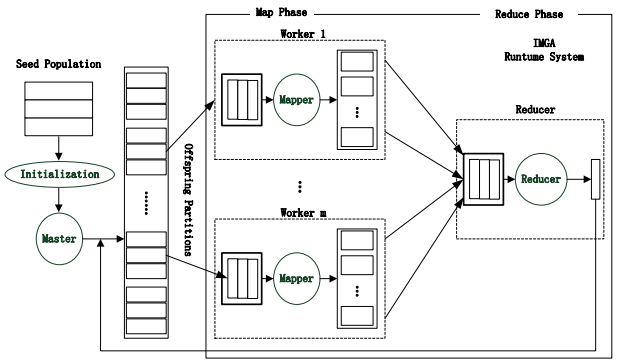


Fig. 9. IMGA architecture overview.

Haloop [14] as an extension to Hadoop [15] is the most popular open source implementation of iterative MapReduce. Fig. 9 shows our architecture of iterative MapReduce, which is based on Haloop. The architecture consists of one master, and multiple mapper and reducer workers. Mapper workers are responsible for executing the map function and reducer workers execute the reduce function (in Table 3), while the master schedules the execution of parallel tasks.

5.1.2. Environment and implementation

Our numerical tests are based on a cluster, composed of 50 worker nodes. A MapReduce cluster can cache the invariant data in the first iteration, and then reuse them in later iterations. Each node has 2 single-core AMD Operon64 2.2GHz CPUs, 4GB DDR RAM, and is connected by 1 Gbps Ethernet. All of the machines are in the same hosting facility and therefore the round-trip time between any pair of machines is less than a millisecond. The whole system is based on Haloop latest version.

The map and reduce functions are built firstly and the fitness computation procedure based on ANSYS are then interfaced with the map function.

5.2. Optimization results

The optimization problems introduced in section 3 are implemented by using the above IMGA. In the optimization process, we have considered the size of the population equal to 100 individuals. Fig. 10 provides a convergence rate of the optimization procedure. It can be seen that the algorithm achieves the best solution after 128 iterations. The extensional CTE has been decreased to $4.08\times10^{-6}/°C$. The final optimized interfaces and matrix layers thicknesses are: 0.64 μm, 0. 58 μm, 0.79 μm,1.49 μm.

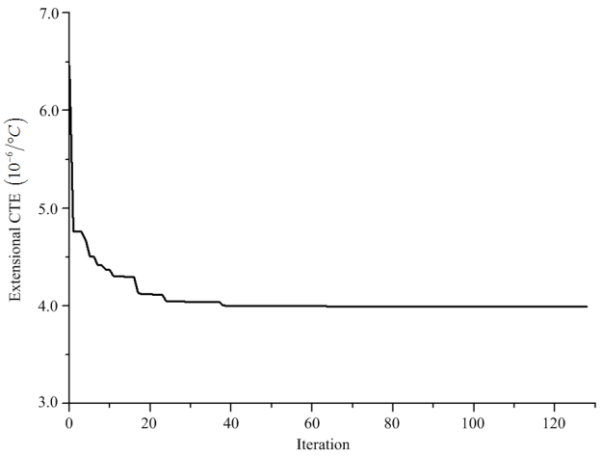


Fig. 10. Convergence rate for the optimization.

To illustrate the correctness and efficiency for IMGA we obtained the results by IMGA and the direct GA which is displayed in Table 4. As can be seen, the agreement between the two solutions is satisfactory, and IMGA improves greatly the efficiency and takes only 6.6% of the execution time required for the conventional single node GA processing.

Then, the scalability of IMGA is tested by scale the number of worker nodes from 10 to 50. We can see that the efficiency is around 91% in Fig. 11.

Table 4. The comparation for IMGA in 20 nodes and conventional GA in single node.

Item	GA	IMGA
Number of individuals	100	100
Number of machines	1	20
Number of generations	131	128
Execution time(min)	2153	143
Best fitness value($10^{-6}/°C$)	4.08	4.08

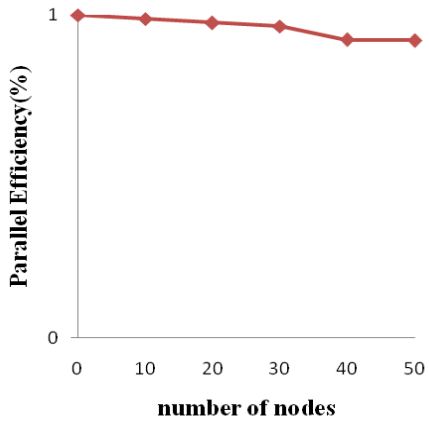


Fig. 11. The parallel efficiency of the IMGA.

6. Conclusion

Optimal design of MPML unidirectional composite with respect to the thermal-elastic properties is obtained by the use of a new IMGA algorithm described in the paper. The unit cell finite element model of MPML unidirectional composite is generated and finite element analysis is realized to determine the CTE and elastic modulus. A new iterative MapReduce that extends the popular MapReduce programming model to iterative computations is proposed and combined with GA method to develop the IMGA. The IMGA is finally used to minimize CTE of the MPML unidirectional composite with constraints of elastic modulus and fiber volume fraction. In addition, the computation efficiency and scalability of the proposed IMGA algorithm are studied respectively. Results indicate that the present IMGA algorithm provides an efficient tool for the large-scale optimization problems.

Acknowledgement

This article is supported by "The Fundamental Research Funds for the Central Universities NO. 3102014JSJ0008", "The National Natural Science Funds NO. 61303225", "The aviation science funds NO. 20135553034".

References

[1] L. T. Zhang, L. F. Cheng, Y. D. Xu, et al, Journal of Aeronautical Materials, **26**, 226 (2006).
[2] W. Krenkel, Cfi-Ceramic Forum International, 80(8), 31 (2003).
[3] J. F. Schulte, J. Schmidt, R. Tamme, et al, Materials Science and Engineering A, **386**(1-2), 428 (2004).
[4] Aslain R, Pailler R, Bourrat X, et al, Solid State Ionics, **141-142**, 541 (2001).
[5] T. Aguchi, T. Nazawa, N. Iagawa, Journal of Nuclear Materials, **329-333**, 572 (2004).
[6] F. Aumouroux, S. Bertrand, R. Paller, et al, Composites Science and Technology, **59**, 1073 (1999).
[7] E. Cant´u-Paz. Efficient and Accurate Parallel Genetic Algorithms. Springer, 2000.
[8] A. Weiss. Computing in the Clouds. netWorker, **11**(4), 16 (2007).
[9] J. Dean, S. Ghemawat, In OSDI'04: Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation. USENIX Association, Berkeley, CA, USA, 10-10 (2004).
[10] ANSYS 10.0 User's Manual", ANSYS Inc., Canonsburg, PA, USA.
[11] Y. J. Xu, W. H. Zhang, H. B. Wang, Materials Science and Technology, **24**(4), 435 (2008).
[12] X. F. Han, Master Degree Thesis, Northwestern Polytechnical University, Department of Material Science, 2006.
[13] J. Dean, S. Ghemawat, Mapreduce: simplified data processing on large clusters. Commun. ACM 51, **1**, 107 (2008).
[14] Bu Y, Howe B, Balazinska M, Ernst M. Haloop, Proceedings of 38th International Conference on Very Large Databases, **3**(1-2), 285 (2010).
[15] The Apache Hadoop Project. http://hadoop.apache.org.

*Corresponding author: youtao@nwpu.edu.cn