A hybrid, massively parallel implementation of a genetic algorithm for optimization of the impact performance of a metal/polymer composite plate

Kiran Narayanan, Angel Mora, Nicholas Allsopp and Tamer El Sayed

International Journal of High Performance Computing Applications published online 17 July 2012
DOI: 10.1177/1094342012451474

THE INTERNATIONAL JOURNAL of HIGH PERFORMANCE COMPUTING APPLICATIONS

The International Journal of High Performance Computing Applications
1–11

DOI: 10.1177/1094342012451474
hpc.sagepub.com
SAGE

A hybrid, massively parallel implementation of a genetic algorithm for optimization of the impact performance of a metal/polymer composite plate

Kiran Narayanan[1], Angel Mora[1], Nicholas Allsopp[1,2] and Tamer El Sayed[1]

Abstract

A hybrid parallelization method composed of a coarse-grained genetic algorithm (GA) and fine-grained objective function evaluations is implemented on a heterogeneous computational resource consisting of 16 IBM Blue Gene/P racks, a single x86 cluster node and a high-performance file system. The GA iterator is coupled with a finite-element (FE) analysis code developed in house to facilitate computational steering in order to calculate the optimal impact velocities of a projectile colliding with a polyurea/structural steel composite plate. The FE code is capable of capturing adiabatic shear bands and strain localization, which are typically observed in high-velocity impact applications, and it includes several constitutive models of plasticity, viscoelasticity and viscoplasticity for metals and soft materials, which allow simulation of ductile fracture by void growth. A strong scaling study of the FE code was conducted to determine the optimum number of processes run in parallel. The relative efficiency of the hybrid, multi-level parallelization method is studied in order to determine the parameters for the parallelization. Optimal impact velocities of the projectile calculated using the proposed approach, are reported.

Keywords

parallel genetic algorithm, computational steering, optimization, impact, ductile fracture, composite

1 Introduction

High-performance computing (HPC) can solve large-scale, complex engineering problems in a significantly shorter time compared with single-processor machines (Adeli, 2000). Simulation tools used for finite-element (FE) analysis of boundary value problems in engineering can be coupled with optimization methods to determine optimum values of input parameters required for the FE analysis, thus facilitating computational steering towards optimal designs. Obtaining high-fidelity simulation results often involves increased computational costs and has led to a quest for faster optimization methodologies (Venkataraman and Haftka, 2004). Parallel implementations of genetic algorithms (GAs) have been the subject of extensive research aimed at reducing the processing time needed to reach acceptable solutions for optimization problems (Cantu-Paz and Goldberg, 2000). In the present study, we couple a GA iterator for single objective minimization (Adams et al., 2009) with an in-house FE analysis code to facilitate guided computation of the optimal impact velocities of a projectile colliding with a polyurea/structural steel composite plate. The GA iterator is used with algorithmic coarse-grained parallelism, which involves the parallelization of multiple, independent function evaluations. Each function evaluation is, in itself, a dynamic, non-linear FE analysis and is implemented in parallel using the Message Passing Interface (MPI). This constitutes function evaluation fine-grained parallelism; cf. Eldred and Hart (1998). The combination of parallelization strategies at the GA iterator level as well as within objective function evaluations leads to a hybrid parallelization method that

[1] King Abdullah University of Science and Technology, Thuwal, Kingdom of Saudi Arabia
[2] Cray Computing, Stuttgart, Germany

Corresponding author:
Tamer El Sayed, Physical Sciences and Engineering Division, King Abdullah University of Science and Technology, Building 4, Room 3222, 4700 KAUST, Thuwal 23955-6900, KSA
Email: tamer.elsayed@kaust.edu.sa

effectively uses available processing power to reduce the time to solution.

We briefly review research involving the solution of optimization problems on multiprocessor machines, wherein a GA is coupled with a code to perform FE analyses of a mechanical structure. Such a coupling is used to solve an optimization problem in structural mechanics and predominantly involves the search for optimal design variables by solution of the governing equations of the structure, i.e. the balance of linear momentum subject to constraints. The FE model includes descriptions of the material's constitution such as fiber angles in the case of plys, weight distributions, etc. The variables being optimized may be associated with the material's constitution, the geometry of the structure or the externally applied loads. Solution of the governing equations of the structure by FE analysis involves discretization of the physical domain into finite elements over which the solution is approximated as polynomials. We note that the parallel implementation of the FE method and GAs, are each, separate and vast areas of research. A detailed review of these topics are not presented in this paper. Reviews of the application of high-performance computing (HPC) methods to FE analyses in structural mechanics are presented by Sotelino (2003) and Fahmy and Namini (1994). A review of the parallel implementations of GAs is presented by Adeli et al. (1993). Using HPC methods to coupled GA/FE analyses to obtain optimal design parameters with reduced computational time and better efficiency has been adopted by several authors. This is the focus in this paper.

The optimization of the response of a composite laminate subjected to impact using an in-house GA coded in FORTRAN coupled to a commercial FE package was presented in Yong et al. (2008). The authors studied two cases: (1) low-velocity impact of a slender laminate strip in which the peak deflection was the objective function for the GA to minimize; (2) high-velocity impact of a rectangular plate in which the penetration was minimized. The optimization variable in both cases was the layup angle for the unidirectional carbon/epoxy composite plies. The performance of the GA was compared with that of the parallel implementation of the response surface methodology by the commercial package LS OPT. The GA performed better on the high-velocity problem in which the search space for the optimization variable was non-linear and larger as opposed to the low-velocity case. The improvement in calculated rebound velocities of the cylindrical impactor for an increased number of average FE evaluations was significantly greater for the GA as compared with LS OPT. The GA led to better results with lower computational costs in problems involving a large number of non-linear optimization variables.

The reduction of time to solution as well as the improvement in design achieved in large-scale structural optimization problems via computational steering using a GA coupled to a FE analysis was demonstrated by Papadrakakis et al. (2003). The function evaluations of the GA, which are FE analyses are parallelized. The GA iterator was used to solve the sizing problem for truss systems. For a truss system with 5269 degrees of freedom and a population size in the range 100–1000 for the GA, the average speed up factor of the parallelization was 8.9. With a similar population size range, an average speed up of 8.6 was observed for the problem of simultaneous shape, topology and sizing optimization of the structure. The authors report computational times and optimal solutions achieved using combinations of the parallelization strategies adopted, such as the optimizer in serial and the FE analysis in parallel and vice versa (cf. Tables VII, XV of Papadrakakis et al. (2003)).

Optimization of fiber-reinforced plastic laminate plates composed of stacked plys and subjected to transverse impact loading using the island model parallel GA coupled with a FE code is presented by Rahul et al. (2005). Several competing sub-populations are used in the island model to search for optimal design variables, including ply thickness, ply material density and the cost of the material per unit, with respect to cost and weight. The optimization was carried out for each objective function separately as well as for a multi-objective case. Faster convergence to optimal solutions and hence an increase in efficiency of computation with an increase in the number of processors used in parallel was reported. For the weight minimization problem, computational time reduction of 90% was reported.

A GA was coupled to FE analysis to obtain optimal designs for protective packaging of household electronic products in Lye et al. (2004). The objective function for the GA was a function of the difference between a target impact load factor (G) value and the simulated G value. The weight distribution, the number of sensitive areas and the number of support areas were the optimization variables. The GA used a FE dynamic drop test simulation for objective function evaluations. The magnitudes of the largest reaction forces encountered by the model were translated into the highest G value the package design could sustain. Optimal designs for packaging were determined for electronic items such as washing machines, television sets, computers and refrigerators. For each item, the G value of the optimal design was compared with the target values, as well as to those from experimental drop tests. Variances in the range of 0.62–11.8% were reported which are ostensibly acceptable by industry standards, thus validating the approach.

The studies reviewed here, except for Papadrakakis et al. (2003), involved a GA implemented in parallel, coupled to a FE analysis code that has been implemented in serial. In this study, both the GA and the FE analysis are implemented in parallel, albeit using different methodologies. In the following sections, we describe the implementation of our optimization methodology followed by a discussion of results obtained.

2 Impact of projectile on a metal/polymer composite plate

Polymer coating of structural members has been used as a means to mitigate the effect of impact loading on a

structure, for example, shop floors, truck undersides and ship hulls. In the present study, we consider the problem of impact of a projectile traveling at high speed on a polyurea/steel composite plate. The mechanical behaviors of polyurea and steel have been modeled using constitutive models of porous plasticity for ductile fracture by void growth (cf. Weinberg et al., 2006; El Sayed et al., 2008). The model for polyurea included the effect of thermal softening, thus allowing for the formation of shear bands. The variational formulation of constitutive updates (cf. Ortiz and Stainier, 1999) has been adopted to implement constitutive models within the FE code. Furthermore, the strain localization elements to capture adiabatic shear bands and contact potential used to model the impact forces are adapted from Yang et al. (2005) and El Sayed et al. (2009), respectively.

We present a brief description of the constitutive model formulation in this section. The model characterizes the mechanical behavior of polymers subject to large deformation by decomposing the material's mechanical response into equilibrium and non-equilibrium components, represented by an elastoplastic network and several viscoelastic mechanisms. Let $\mathbf{F}$ denote the deformation gradient at an arbitrary point of the material, and let

$$\mathbf{F} = \mathbf{F}^e\mathbf{F}^p = \mathbf{F}_1^e\mathbf{F}_1^v = \cdots = \mathbf{F}_M^e\mathbf{F}_M^v \tag{1}$$

be its multiplicative decomposition, where M is a positive integer that defines the number of viscoelastic (Maxwell-type) relaxation networks that the model possesses, which act in parallel with an elastoplastic equilibrium network; $\mathbf{F}^e$, $\mathbf{F}_1^e, \ldots, \mathbf{F}_M^e$ are the elastic parts of $\mathbf{F}$; $\mathbf{F}^p$ is the plastic deformation gradient; and $\mathbf{F}_1^v, \ldots, \mathbf{F}_M^v$ are the viscous deformation gradients. With recourse to the variational formulation (cf. Ortiz and Stainier, 1999), the free energy is assumed to have the following additive structure

$$\begin{aligned} &A^{ep}(\mathbf{F}, \mathbf{F}^p, \mathbf{Z}^p, T) + A^{ve}\left(\mathbf{F}^p, \mathbf{F}_i^v, \mathbf{Z}_i^v\right) \\ &\quad = W^e\left(\mathbf{F}\mathbf{F}^{p-1}, T\right) + W^p(\mathbf{Z}^p, T) + \sum_{i=1}^{M} W_i^e\left(\mathbf{F}\mathbf{F}_i^{v-1}, T\right) \\ &\quad + \rho C_v T\left(1 - log\frac{T}{T_0}\right) \end{aligned}$$

where W^e is the elastic strain-energy density, W^p is the plastic stored energy, W_i^e $(i = 1, \ldots, M)$ are the elastic strain-energy densities corresponding to the viscous relaxation mechanisms; ρ_0 is the mass density per unit undeformed volume; C_v is the specific heat per unit mass at constant volume; and T_0 is the reference temperature. The variables $\mathbf{F}^p$,$\mathbf{Z}^p$ and $\mathbf{F}_i^v$, $\mathbf{Z}_i^v$ are related to each other by means of the flow rules. The time integration of the constitutive equations is effected by recourse to an incremental variational update implemented via the use of a predictor-corrector algorithm for the incremental logarithmic strain (cf. Weinberg et al., 2006). Parameters used for the porous plasticity constitutive model for steel and soft-tissue model for polyurea are listed in Tables 1 and 2, respectively. These parameters were obtained from El Sayed et al. (2009) where the parameters were validated and verified against experimental data for identical loading conditions of the materials. A FE mesh comprising of 25,605 fully integrated 10-node composite tetrahedral elements was constructed in adherence to the dimensions given in Table 3. For a more detailed description of the FE model, the reader should refer to El Sayed et al. (2009).

Table 1. Porous plasticity parameters for steel (cf. El Sayed et al., 2009).

ρ (kg/m^3)	7700
E (GPa)	214.0
ν	0.28
ρ_0 (Pa)	150×10^6
ε_0^p	0.004
n	7.5
$\dot{\varepsilon}_0^p$ (s^{-1})	0.007
m	50.0
T_0 (K)	294
T_m (K)	1371
C_v (J kg^{-1} K^{-1})	438.0
α (K^{-1})	11.7×10^{-6}
l	0.75
β	0.9
Void radius $a_0(m)$	5.2×10^{-9}
Void density N_v (voids/m^3)	1.0×10^{22}

Table 2. Material parameters for polyurea (cf. El Sayed et al., 2009).

ρ (kg/m^3)	1070.0
E (GPa)	2.42
ν	0.499
ρ_0 (Pa)	5.2575×10^6
ε_0^p	0.12
$\dot{\varepsilon}_0^p$ (s^{-1})	7198.88
m	0.411345
n	1.0
l	1.0
C_v (J kg^{-1} K^{-1})	1200.0
T_0 (K)	294.0
T_m (K)	473.0
α (K^{-1})	1.0×10^{-4}
Void radius a_0 (m)	100.0×10^{-6}
Void density N_v (voids/m^3)	1.0×10^{10}
β	0.3
N	2

Table 3. Dimensions and parameters of the FE model.

Impact mass (g)	145
Impact hardness (RC)	36.0
Steel target plate mass (g)	692.2
Steel target plate diameter (mm)	154.2
Steel target plate average thickness (mm)	4.75
Polymer mass (g)	230.1
Polymer diameter (mm)	154.2
Polymer/steel thickness ratio	2.335

For a given value of projectile velocity during initiation of contact, the FE analysis code computes the projectile velocity values as time progresses by simulating impact/contact with the polyurea/steel composite plate. Other mechanical response variables such as strains and stresses within the material are also available as output for postprocessing. For the purposes of the present study, the time duration for the dynamic simulation is determined by any one of the following two termination criteria. First, if the difference between the magnitudes of the velocities of the projectile between subsequent iterations is less than 10 m/s over the course of 50,000 analysis iterations, a termination signal is generated. Second, if, during an analysis iteration the projectile velocity is greater than zero, i.e. positive, the analysis terminates, for this corresponds to the projectile being deflected back without penetrating the plate. The termination criteria, along with the exchange of data and control between the GA and the FE code are implemented by means of the coupling script described in Section 5.

3 Genetic algorithm

A GA starts with an initial population of randomly selected individuals (i.e. design points) in parameter space. A 'genetic string', analogous to DNA in a biological system, is constructed from the values of the design parameters. The 'genetic string' uniquely represents each individual in the population. The GA then searches the parameter space for an optimal design point by following a sequence of 'generations'. The best individuals in the population of each generation (i.e. those having low objective function values) are considered to be most 'fit' and are allowed to survive and reproduce. The mathematical analogues of processes such as natural selection, breeding and mutation are then employed to obtain subsequent 'fitter' generations of individuals. Ultimately, the GA identifies a design point that minimizes the objective function of the optimization problem. An extensive discussion of GAs may be found in Goldberg (1989).

The lack of explicit derivatives necessitated the use of a GA in our study. The flowchart in Figure 1 shows the sequence of steps implemented by the GA. The Single Objective Genetic Algorithm (cf. Adams et al., 2009) was used to find optimal values of projectile velocity onto a polyurea/steel composite plate. The fitness function for the GA, which is the magnitude of the velocity of the projectile at the end of the FE analysis, is minimized. This definition for the fitness function, together with the termination criteria for the FE analysis described in Section 2 renders the GA iterator to search for the impact velocity of the projectile which results in the projectile being stopped by the plate.

At the beginning of the GA, the population size, i.e. the number of individuals that comprise one generation, is specified. Each individual of the first generation is initialized using unique random seeding, i.e. individuals with random variable values are created with recourse to a uniform number distribution while rejecting any duplicates. The seed size defines the starting seed value for the random number generator. Objective function evaluation of the initial population members is then performed and each individual is assigned a fitness value, which, in our case, is the value of the objective function. 'Shuffle random' crossover is then performed on this initial generation to create offspring by choosing individuals at random from a specified number of parents enough times that the requested number of children is produced. The crossover rate specifies the probability of a crossover operation being performed to generate new offspring. In this study, a crossover rate of 0.8 is used. The number of crossovers is equal to the rate multiplied by the population size. Next, offspring are subjected to mutation as a final step in creating the new generation of individuals. Mutation is performed using 'uniform replacement' in which a replacement value for an individual is determined by using uniformly distributed values over the total range. The number of mutations is the product of the mutation rate and the population size. The new generation of individuals is then evaluated and their fitness assessed. Fitness assessment determines how strongly differences in fitness are weighted in the process of selecting 'parents' for crossover (i.e. producing the next offspring). In our study, fitness assessment is based on a merit function that makes use of a proportional scaling of the probability of selection based on the relative value of each individual's objective function in the population. Once the fitness assessment is completed, the population is replaced with members selected to continue in the next generation. The potential members are sourced out of the current population and the new generation. The GA uses the fitness of each member and replaces the current population based on a unique roulette wheel selection criteria. Each individual is conceptually allotted a portion of a wheel proportional to its fitness relative to the fitness of the other individuals. Portions of the wheel are then chosen at random and individuals occupying those portions are duplicated in the next population. This ensures that an individual is selected only once. The sequence of operations, namely crossover, mutation, evaluation, fitness assessment and replacement, are performed until one of the two termination criteria is satisfied. The stopping criteria is based on the total number of function evaluations. The convergence criteria is based on average fitness; the GA stops if the average fitness value for the population does not change more than 0.1% over 10 generations. Upon termination, the GA reports the member with the best fitness (i.e. lowest objective function value) as the solution to the optimization problem.

4 Parallelization method

The hybrid parallelization method adopted in the present study is schematically illustrated in Figure 2. The figure illustrates the usage of the different components of the heterogenous computational resource that consists of a

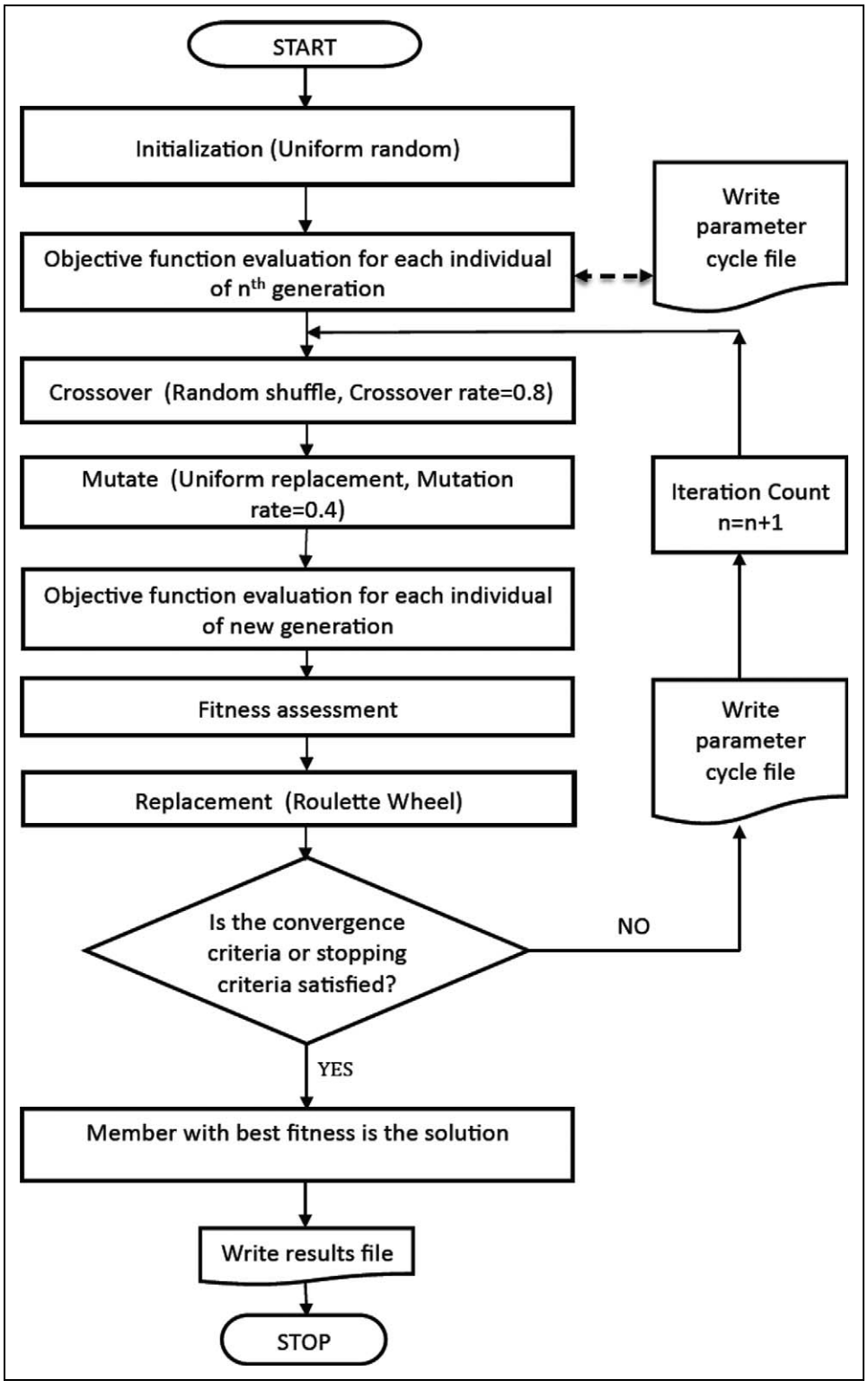

Figure 1. Flowchart of the genetic algorithm.

single x86 node and 16 IBM Blue Gene/P (BG/P) racks. Both the master x86 node and BG/P machine utilize a common directory structure via a general parallel file system (GPFS). The GPFS facilitates the passing of data between the GA iterator running on the master node and objective function evaluations running on the BG/P as described below.

4.1 Parallelization at GA iterator level

Algorithmic *coarse-grained* parallelism (cf. Eldred and Hart, 1998) based on the master–worker single program multiple data (SPMD) model is used at the GA iterator level. The GA iterator is executed on a master x86 node and

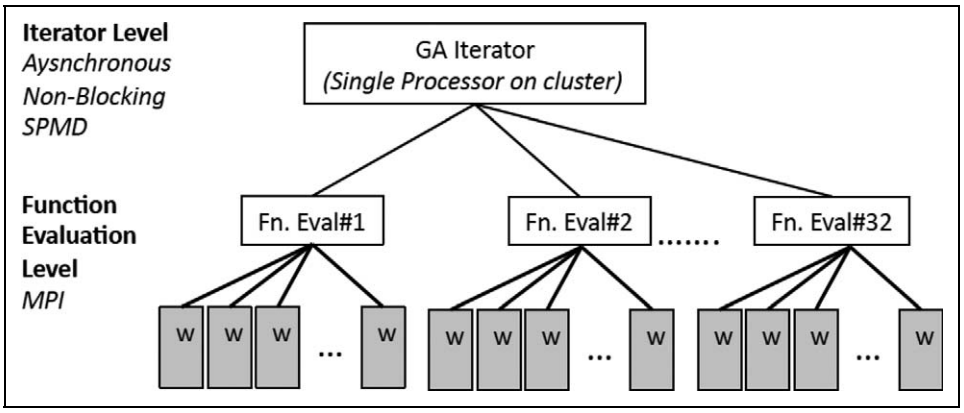

Figure 2. Parallelization method

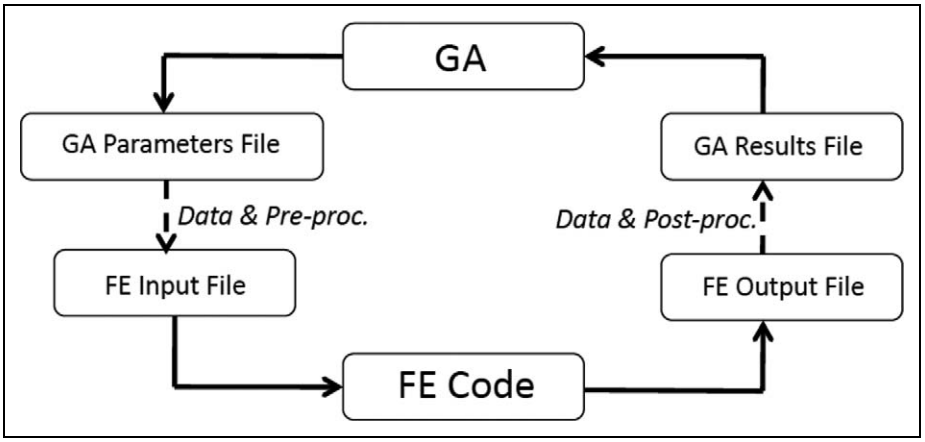

Figure 3. Schematic of the GA/FE code coupling.

launches multiple objective function evaluations in parallel onto the worker processors on the BG/P. The master executes the iterator code while the workers execute the FE code used for objective function evaluation. The master–worker SPMD model eliminates resource contention since the master is responsible for all I/O (cf. Eldred and Hart, 1998). Asynchronous local schedulers are used for managing concurrent function evaluations with non-blocking synchronization. The function evaluations themselves are invoked using a fork simulation interface. Proper maintenance of the file systems was ensured to guarantee efficient scheduler execution between the x86 Master and BG/P files.

4.2 Parallelization at function evaluation level

The FE analyses that constitute the objective function evaluations are executed in parallel on the BG/P multi-core system using *fine-grained* parallelism by the use of MPI. The in-house FE code uses a domain decomposition technique to assign subdomains of the composite plate between processors. For a fixed plate size, as the number of processors used increases, the size of the subdomain assigned to each processor decreases. Thus, there is an optimal number of processors for calculating the fracture characteristics of the composite plate depending upon the size of the plate and the amount of communication required between processors. To assess the optimal size of each subdomain, a fixed composite plate size was used and the number of MPI processors was changed using a load leveler script to obtain strong scaling data which are discussed in Section 6.1.

5 GA/FE coupling

The GA iterator is executed on a single master x86 cluster node and launches concurrent job evaluations onto the worker processors of the BG/P, each of which, in turn, launch portions of the FE analysis onto worker processors using MPI. The method of coupling the GA iterator to the FE code depends on both residing on a common file system. A shell script was written to couple the GA iterator to the FE analysis code using the scheme shown in Figure 3. The GA iterator running on the single master node runs the coupling script during function evaluations. The flowchart of the coupling script is shown in Figure 4.

The coupling script performs the following tasks:

1. Create a temporary work directory for each instance of the coupling and copy necessary input files into it, thus permitting the use of fixed names for intermediate files.

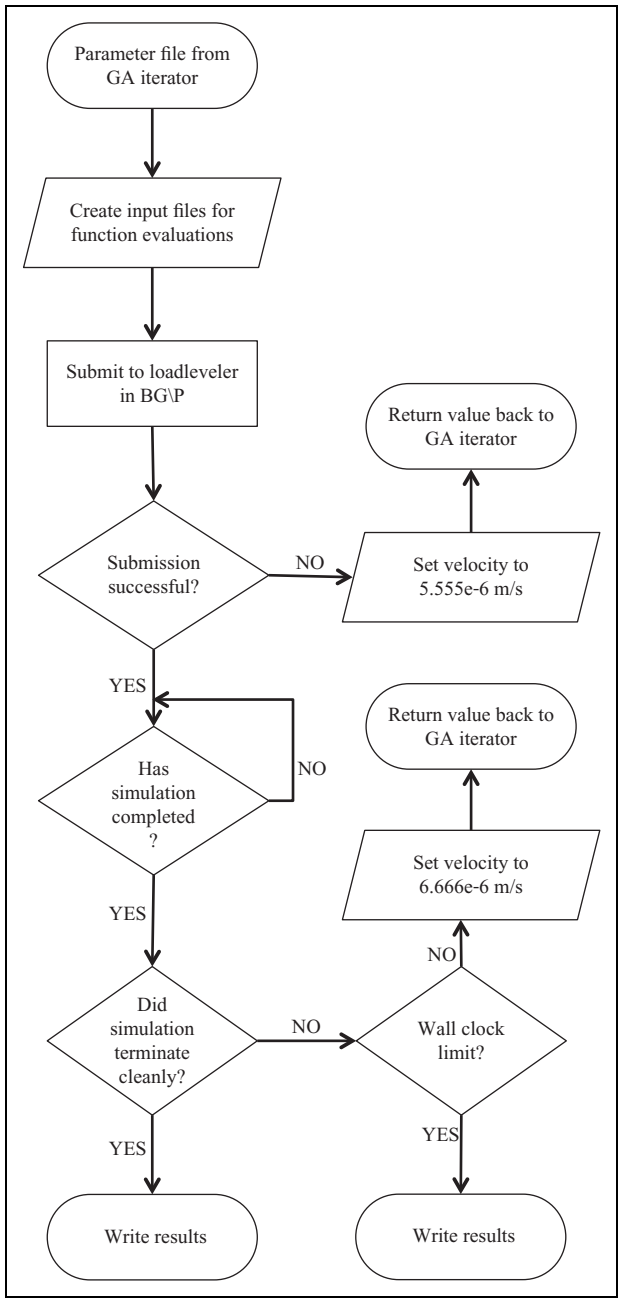

Figure 4. Flowchart for the coupling script.

2. Use *dprepro*, a parsing utility, to extract the velocity value for the current individual of the population from the parameter cycle file that the GA creates and combine it with a template file to create input for FE analysis.
3. Execute the load leveler script for the FE analysis, which in turn submits MPI-enabled jobs onto worker processors. If the job submission is unsuccessful, the velocity value is set to a very small default value close to zero, which is used as a flag to detect unsuccessful submissions. If the submission is successful, check for job completion every 60 seconds.
4. Once the objective function evaluation is complete, check for the existence of an error file. If the FE analysis failed, the velocity value is set to a default small number. If the evaluation is completed without errors or stopped because the wall clock limit was reached, the final velocity is passed back to the GA iterator.
5. Remove the temporary work directory after moving the required output files.

Table 4. Strong scaling data: dynamic simulation times at the wall clock limit.

Number of processors	Simulation time [$\times 10^{-5}$ seconds]	% Scaling
32	1.46	—
64	2.08	71
128	4.18	72
256	1.83	16

6 Results and discussion

6.1 Strong scaling study of parallel function evaluations

A strong scaling study of the parallel implementation of the FE code using domain decomposition and MPI was performed in two stages. In the first stage, for a given projectile impact velocity, the FE code was allowed to run until it reached the wall clock limit and the dynamic simulation times were estimated by varying the number of processors assigned to each subdomain. The value of the dynamic simulation time upon termination of the FE analysis is indicative of the speed of computation. The results listed in Table 4 show that the optimal number of processors for parallel function evaluations using MPI is 128. An increase in computational cost is observed when the number of processors assigned to each subdomain is increased to 256, owing to an increase in the communication time between processors. In the second stage of the strong scaling study, for a given projectile impact velocity, the FE code was executed until the velocity of the projectile reached a specified value. The number of processors assigned to each subdomain was varied and the corresponding simulation times were plotted as shown in Figure 5. The simulation run time data was fitted to a simple mathematical model of the form

$$T(p') = A + B/p' \tag{2}$$

where p' is the number of processors assigned to each subdomain, $T(p')$ is the corresponding simulation run time and A, B are constants. The value of the constants A and B obtained using least squares fit were 10,072 and 2,479,318 seconds, respectively. In our study, the minimum number of processors used for parallel function evaluations ($p'_{\min}$) was 32. The number of processors used during the strong scaling study was not reduced below this number in order to avoid the significantly higher computational

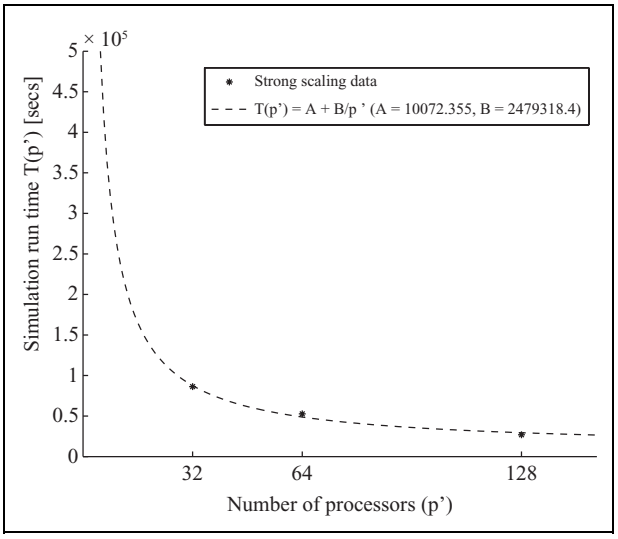

Figure 5. Strong scaling plot for the FE code ($A = 10,072$ seconds, $B = 2,479,318$ seconds).

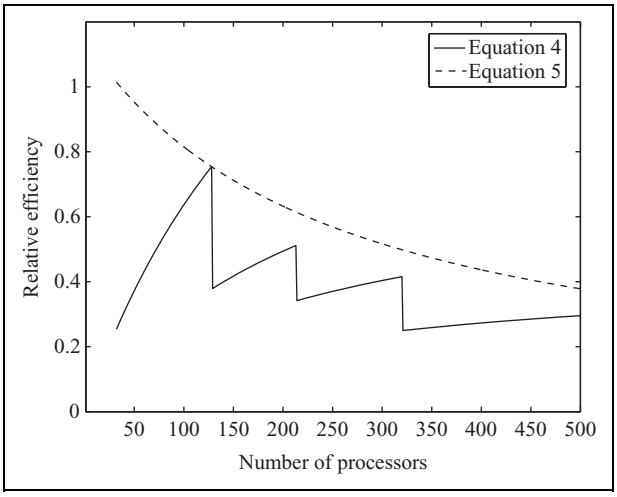

Figure 6. Relative efficiency of the parallelization (concurrency = 5).

cost associated with such a run due to the long simulation times observed during our simulations. Using the strong scaling data, the optimal number of worker processors used for each subdomain (p'_{opt}) was estimated to be 128.

6.2 Relative efficiency of hybrid, parallel, and coupled GA/FE simulations

We investigated the relative efficiency of the multi-level parallel implementation of the coupled GA/FE simulations using the results from the strong scaling study presented in Section 6.1 and a mathematical model adapted from Eldred and Hart (1998). The relative efficiency $E(p)$ of the parallelization which accounts for the efficiency of the GA iterator as well as the parallelized function evaluation, is (cf. Eldred and Hart, 1998)

$$E(p) = \frac{(p'_{\min}+1)\hat{T}(p'_{\min}+1)}{p\hat{T}(p)} = \frac{(p'_{\min}+1)\left(T_{\text{serial}} + \alpha\kappa\left(2T_{\text{comm}} + T(p'_{\min})\right)\right)}{p(T_{\text{serial}} + \alpha\kappa(2T_{\text{comm}} + T(p')))}$$

for $p \geq p'_{\min} + 1$. Here p is the number of processors used, $p'_{\min}$ is the smallest number of processors in which the function evaluations can be executed, $T(p')$ is the simulation run time for one function evaluation on p' processors obtained from Equation (2), $\hat{T}(p'_{\min}+1)$ is the simulation run time when using the smallest number of processors to run the master-worker algorithm, i.e. $p_{\min} = p'_{\min} + 1$ and $\hat{T}(p)$ is the run time on p processors. Furthermore, α is the number of cycles to convergence of the GA, κ is the number of evaluations per iteration of the GA, i.e. the evaluation concurrency, and T_{serial}, T_{comm} are the execution time for the serial part of the GA and the communication time between master and worker processors, respectively.

It is worthy to note that the analysis of relative efficiencies presented here is in general applicable to simulation-based optimization schemes, i.e. in situations where an optimizer has been coupled with a "black-box" simulation code. Most parallel algorithms satisfy the assumption that $T(p')$ is convex. Further, simulation-based optimization problems satisfy the assumption that the smallest $T(p')$ is large relative to the communication time between the master and worker processors as well as to the execution time of the serial portion of optimizer, i.e. in our case, the GA. We can thus approximate $E(p)$ as

$$E(p) \approx \frac{(p'_{\min}+1)\kappa T(p'_{\min})}{p\beta_p T(p')} \tag{4}$$

where $\beta_p = \frac{\kappa}{\left\lfloor \frac{p-1}{p} \right\rfloor}$ is the maximum number of function evaluations any worker performs in each GA iteration, and $p = 1 + \kappa p'_{\text{opt}}$.

Eliminating β_p rids us of the non-smooth operations in Equation (4) and we have

$$E(p) \approx \frac{(p'_{\min}+1)\left(\kappa p'_{\text{opt}}\right)T(p'_{\min})}{(1+\kappa p'_{\text{opt}})p'T(p')}. \tag{5}$$

Figures 6, 7 and 8 show the curves for Equations (4) and (5), for evaluation concurrency values of 5, 16 and 32, respectively. The value of $p'_{\min}$ in our study is 32. The non-smooth expression for $E(p)$ in Equation (4) is less accurate when the round-off error arising due to the floor or ceiling operations to estimate β_p are significantly high. This prediction is confirmed by the plots that show better approximations as κ is increased, since an increase in κ signifies a β_p value closer to the ideal value of 1. We also observe that a relative efficiency of 75.5% is achieved for the p'_{opt} value of 128 obtained in the strong scaling study. From the analysis presented in this section it is clear that when the minimum number of processors used during the parallelization of independent function evaluations, i.e. $p'_{\min} \succ 1$, the efficiency of the hybrid parallelization is always higher than a single level parallelism constituted of fine-grained function evaluations alone.

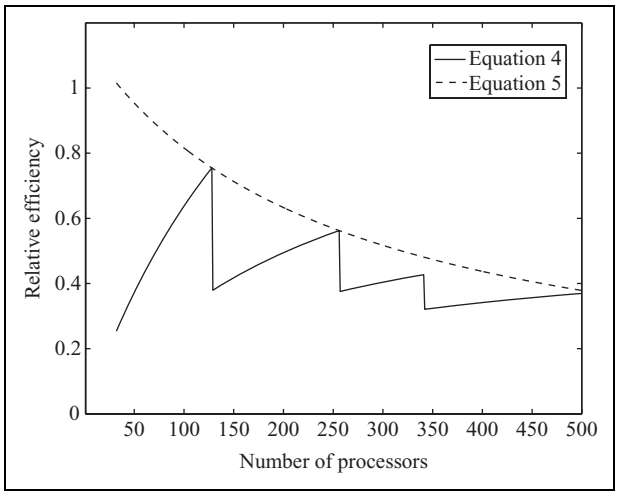

Figure 7. Relative efficiency of the parallelization (concurrency = 16).

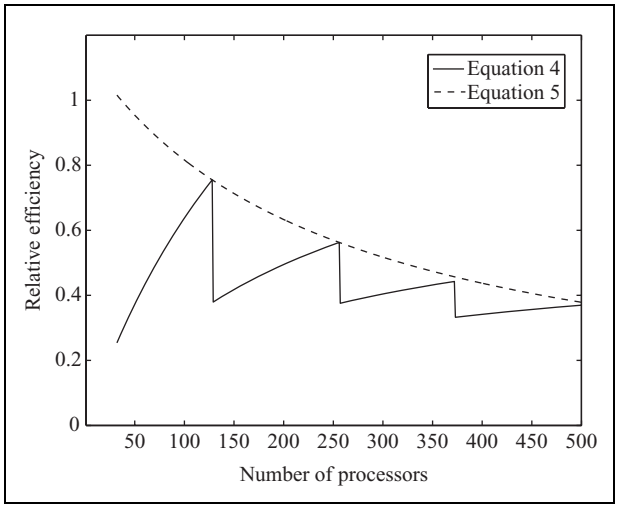

Figure 8. Relative efficiency of the parallelization (concurrency = 32).

6.3 Optimal projectile velocity

The optimal value of the impact velocity of the projectile (v_{opt}) was determined to be 302.3 m/s. Table 5 lists the calculated values of impact velocity (v_0) that are closest to the optimal value for the different values of κ and the maximum number of evaluations $Z_{\max}$ used in our study. The time–velocity plot for the FE simulation of projectiles for these cases are shown in Figure 9 and 10, which correspond to κ values of 5 and 16, respectively.

Furthermore, the impact velocity that is closest to the optimal value for each generation of the GA and the corresponding computational times are plotted in Figures 11 and 12, respectively. It is observed that for a concurrency value of 16, the time to solution was considerably less compared with the case where a concurrency of 5 was used. This illustrates how carefully selected parameters for the parallelization of the GA can help reduce the computational costs associated with a large number of function evaluations.

Table 5. Calculated values of impact velocity v_0 for different values of κ and Z_{max}.

κ	$Z_{\max}$	v_0 (m/s)
5	25	300.3
5	25	299.2
5	25	283.3
5	25	282.4
5	25	272.7
16	32	302.3
16	32	286.0
16	32	285.4
16	32	261.9
16	32	225.6

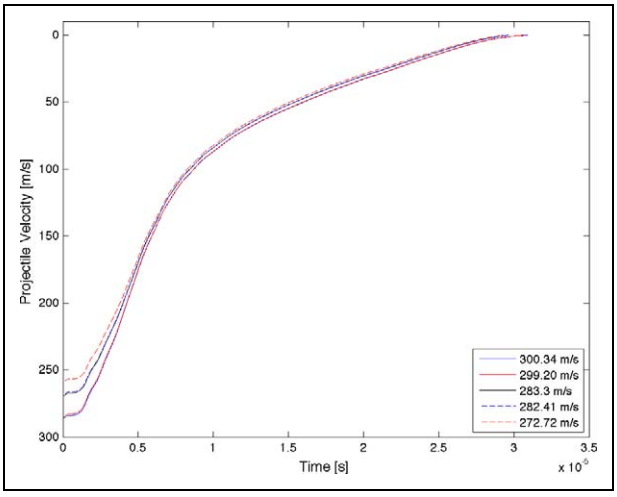

Figure 9. Time velocity plot for designs closest to the optimum (concurrency = 5).

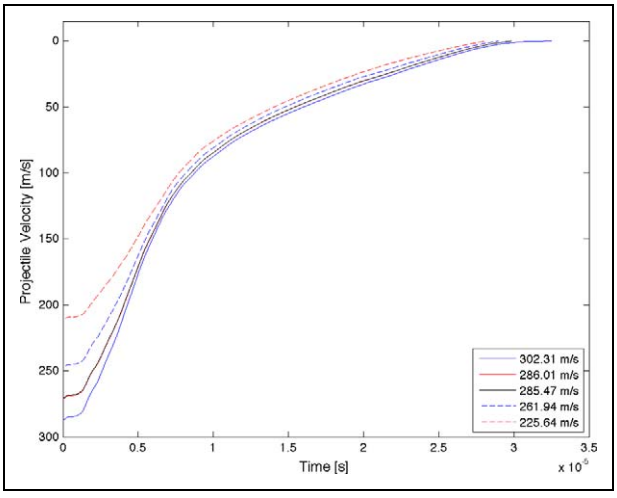

Figure 10. Time velocity plot for designs closest to the optimum (concurrency = 16).

7 Conclusions

A hybrid massively parallel and coupled GA/FE approach was used to calculate optimal impact velocities of a

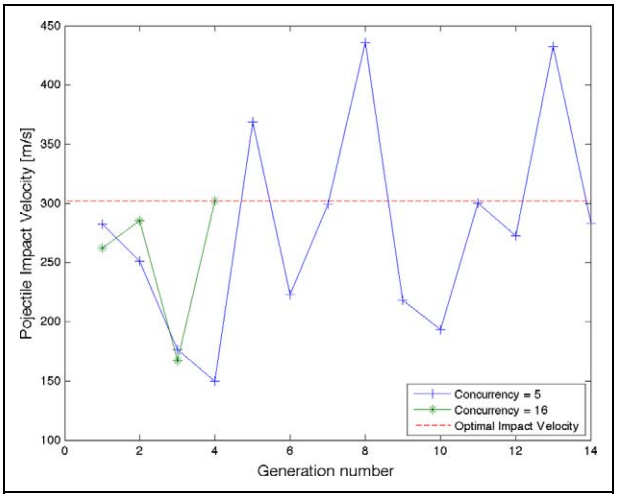

Figure 11. Design points closest to optimum for each generation of the GA.

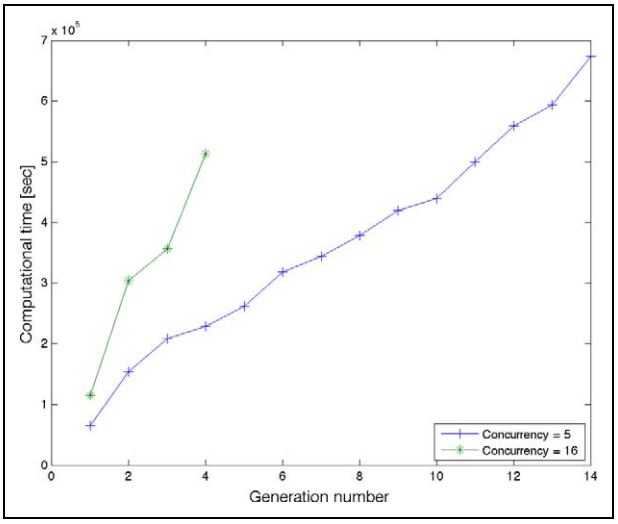

Figure 12. Computational time to arrive at designs closest to optimum for each generation of the GA.

projectile on a polyurea/steel composite plate. Significant reductions in the time to solution were achieved by using the hybrid parallelization strategy. In addition, careful selection of the parameters of the GA further contributed to the savings in computational cost. The hybrid approach presented in this study is, in general, useful for simulation-based optimization problems as evident from the analysis of efficiencies presented in Section 6.2. It is worthy to note that although a change in the scale of the FE problem would affect the strong scaling of the parallelized function evaluations across processors, the method of selecting optimal parameters for the parallelization of the function evaluations and, more importantly, of the GA would remain the same. In future work, we will test the potential of this approach towards development of a computational uncertainty quantification (UQ) framework that calculates the bounds for the uncertainties in performance measures.

Funding

This work was fully funded by the KAUST baseline fund.

Acknowledgements

The first and second authors were both equal contributors to this work. The authors would like to thank the reviewers for their comments and suggestions. For computer time, this research used the resources of the Supercomputing Laboratory at King Abdullah University of Science & Technology (KAUST) in Thuwal, Saudi Arabia.

References

Adams B, Bohnhoff W, Dalbey K, et al. (2009) DAKOTA, A multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis: Version 5.0 user's manual. Sandia Technical Report SAND2010-2183.

Adeli H (2000) High-performance computing for large-scale analysis, optimization, and control. *Journal of Aerospace Engineering* 13: 1–10.

Adeli H, Kamat MP, Kulkarni G and Vanluchene RD (1993) High-performance computing in structural mechanics and engineering. *Journal of Aerospace Engineering* 6: 249–267.

Cantu-Paz E and Goldberg D (2000) Efficient parallel genetic algorithms: theory and practice. *Computer Methods In Applied Mechanics and Engineering* 186: 221–238.

El Sayed T, Mock J, Mota AW, Fraternali F and Ortiz M (2009) Computational assessment of ballistic impact on a high strength structural steel/polyurea composite plate. *Computational Mechanics* 43: 525–534.

El Sayed T, Mota A, Fraternali F and Ortiz M (2008) A variational constitutive model for soft biological. *Journal of Biomechanics* 41: 1458–1466.

Eldred M and Hart W (1998) Design and implementation of multilevel parallel optimization on the intel teraflops. In *Proceedings of the 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, AIAA-98-4707, pp. 44–54.

Fahmy M and Namini A (1994) A survey of parallel nonlinear dynamic analysis methodologies. *Computers and Structures* 53: 1033–1043.

Goldberg DE (1989) *Genetic Algorithms in Search, Optimization, and Machine Learning*. New York: Addison-Wesley.

Lye S, Lee S and Chew B (2004) Virtual design and testing of protective packaging buffers. *Computers in Industry* 54: 209–221.

Ortiz M and Stainier L (1999) The variational formulation of viscoplastic constitutive updates. *Computer Methods In Applied Mechanics and Engineering* 171: 419–444.

Papadrakakis M, Lagaros N and Fragakis Y (2003) Parallel computational strategies for structural optimization. *International Journal For Numerical Methods in Engineering* 58: 1347–1380.

Rahul Chakraborty D and Dutta A (2005) Optimization of FRP composites against impact induced failure using island model parallel genetic algorithm. *Composites Science and Technology* 65: 2003–2013.

Sotelino E (2003) Parallel processing techniques in structural engineering applications. *Journal of Structural Engineering—ASCE* 129: 1698–1706.

Venkataraman S and Haftka R (2004) Structural optimization complexity: what has Moore's law done for us? *Structural and Multidisciplinary Optimization* 28: 375–387.

Weinberg K, Mota A and Ortiz M (2006) A variational constitutive model for porous metal plasticity. *Computational Mechanics* 37: 142–152.

Yang Q, Mota A and Ortiz M (2005) A class of variational strain-localization finite elements. *International Journal For Numerical Methods In Engineering* 62: 1013–1037.

Yong M, Falzon B and Lannucci L (2008) On the applications of genetic algorithms for optmising composites against impact loading. *International Journal of Impact Engineering* 35: 1293–1302.

Author Biographies

Kiran Narayanan joined the Computational Solid Mechanics Laboratory at KAUST as a PhD student in September 2010. His research interests include finite-element methods, constitutive modeling of polycrystalline solids, polymers and biomaterials, and HPC methods for uncertainty quantification and optimization. He graduated as a gold medalist with a bachelor's degree in mechanical engineering from the University of Madras, India in May 2004, after which he was a graduate student at the non-linear mechanics lab at Texas A&M University from Fall 2004 to Spring 2008. Prior to joining KAUST, he has worked as a research fellow at the computational nanomechanics lab in the Defense Institute of Advanced Technology in Pune, India as well as project engineer for DeepSea Engineering & Management, Houston, USA.

Angel Mora was born in Cuernavaca, Mexico, the 'city of eternal spring'. He received his BS degree in Mechatronics Engineering in December of 2007 from ITESM Cuernavaca Campus, also known as Monterrey Tec. After graduation, he worked for 6 months at Nissan in Toluca, Mexico as Design Engineer until December of 2008. He received his MSc degree in Mechanical Engineering from KAUST in 2010. He joined CSML as a PhD student in January 2011. His research interests include finite element methods and constitutive modeling of thermo-mechanical response of materials.

Nicholas Allsopp obtained his doctorate in theoretical condensed matter physics from Lancaster University in the United Kingdom. He then joined the performance group in IBM (UK) where he spent 10 years specializing on the optimization and scaling of applications onto high-end supercomputers. He recently joined Cray Computing (Germany) and is presently working at HLRS on Europe's largest academic supercomputer as an application specialist. Prior to his appointment at Cray, he was an application specialist as part of the HPC group at KAUST (Saudi Arabia). His main interest lies in porting, tuning applications onto new hardware and make them scale as far as possible.

Tamer El Sayed works at KAUST and prior to relocating to Saudi Arabia, he served as a Visiting Associate in Aeronautics at the California Institute of Technology (Caltech). His latest work is involved with formulating constitutive models for polymers and biological tissues for the purpose of predicting their damage behavior under dynamic impact. Prior to joining the KAUST faculty, he held the position of Senior Research Scientist at nanoPrecision Products Inc. (nPP) in Los Angeles, California, where he conducted research on computational nano-precision manufacturing, uncertainty quantification and non-convex optimization. In 2007, he was appointed Postdoctoral Scholar at Caltech in the Graduate Aeronautical Laboratories where he established computational capability to determine ballistic critical impact velocities on polymer-reinforced composite structures and formulated constitutive models for soft materials. His work also focused on the formulation of highly predictive computational models for head trauma that can aid in understanding the physiological dysfunction associated with traumatic brain injuries. He received his doctoral and master's degrees in Mechanical Engineering from Caltech. He graduated as valedictorian and summa cum laude with a bachelor's degree in Mechanical Engineering from California Polytechnic University (Cal Poly), Pomona.