A Multi-Cluster Grid Enabled Evolution framework for Aerodynamic Airfoil Design Optimization

Hee-Khiang Ng[1], Dudy Lim[1], Yew-Soon Ong[1], Bu-Sung Lee[1], Lars Freund [2], Shuja Parvez[2], Bernhard Sendhoff[2]

[1]School of Computer Engineering
Nanyang Technological University, Nanyang Avenue, Singapore 639798
{mhkng, dlim, asysong, ebslee}@ntu.edu.sg
[2]Honda Research Institute Europe Gmbh
Carl-Legien-Strasse 30, 63073 Offenbach
shshgs01@fht-esslingen.de,
{lars.freund , bernhard.sendhoff]}@honda-ri.de

Abstract. Advances in grid computing have recently sparkled the research and development of Grid problem solving environments for complex design. Parallelism in the form of distributed computing is a growing trend, particularly so in the analysis and optimization of high-fidelity computationally expensive real world design problems in science and engineering. In this paper, we present a powerful and inexpensive grid enabled evolution framework based on Globus and NetSolve toolkits for facilitating embarrassingly parallelism in hierarchical parallel evolutionary algorithms. By exploiting the grid evolution framework and a multi-level parallelization strategy of hierarchical parallel GAs, we present the evolutionary optimization of a realistic 2D aerodynamic airfoil structure. Further, we study the utility of hierarchical parallel GAs on two potential grid enabled evolution framework and analysis how it fares on a grid environment with multiple heterogeneous clusters, *i.e.*, clusters with differing specifications and processing nodes. From the results, it is possible to conclude that a grid enabled hierarchical parallel evolutionary algorithm is not mere hype but offers a credible alternative, providing significant speed-up to complex engineering design optimization.

1 Introduction

Genetic Algorithms (GA) represents one of the well-known modern stochastic search techniques inspired by the Darwinian theory of natural selection and evolution [1]. By emulating the process of natural evolution, GAs has been employed with great success for solving many complex engineering design problems including turbine blade design [2], multi-disciplinary rotor blade design [3], aircraft wing design [4], military airframe preliminary design [5] and large flexible space structures design [6]. Its popularity lies in the ease of implementation and the ability to arrive close to the global optimum design. Another well-known strength of GA is that sub-linear improvements

in the search efficiency may be easily achieved by incorporating parallelism. Many studies on the parallelism of GA have been made over the last decade [7-10], with many strategies introduced to date. In general, these strategies to achieve parallelism in evolutionary algorithm (EA) may be categorized as master-slave, fine-grained, or coarse-grained parallel EAs.

Recently, there has been a new paradigm shift in science and engineering towards the utilization of increasingly high-fidelity and accurate analysis codes in the design analysis and optimization processes. In many application areas such as photonics, electromagnetics, aerospace, biomedical, micro-electro-mechanical systems and coupled-field multidisciplinary system, the design process generally requires a Computational Structural Mechanics (CSM), a Computational Fluid Dynamics (CFD) or a Computational Electronics & Electromagnetics (CEE) simulation procedure. Here, a single analysis of the design involving CFD, CSM or CEE could take up many minutes to hours or even days of supercomputing time [11-13]. The high computational costs associated with the use of high-fidelity simulation models thus poses a serious impediment to the successful application of evolutionary algorithms (EAs) to engineering design optimization since EAs typically require many thousands of function evaluations to locate a near optimal solution. Hence, when computationally expensive high-fidelity simulation models are used for predicting design improvements, the use of EAs may be computationally prohibitive. Moreover, solving computationally expensive design optimization problems using a parallel EA may be regarded as impractical since this often requires a huge amount of computational power that are extremely costly for any single organization to take full ownership of. Hence, the use of Grid computing presents a viable and cost effective option to large-scale and computationally expensive design optimization problems.

Recent technologies in Grid computing [14-16] has therefore offered a fresh solution to this problem by enabling collaborative computing on an unprecedented scale via leveraging from geographically distributed computing resources. Here, we harness the idea of employing heterogeneous computing resources distributed in different design teams at disparate geographical locations as a powerful and inexpensive technology to facilitate the embarrassingly parallelism in evolutionary optimization. Due to the large design spaces often considered, usually stochastic optimization algorithms such as parallel GA and its variants are employed in the aerodynamic search in order to arrive at a near optimum design efficiently.

The use of Grid technologies in optimization can be found in [17-19]. The Grid Enabled Optimization and Design Search for Engineering (GEODISE) [17] of the e-science group, UK, represent one of the recent initiatives of Grid computing for engineering design search and optimization. Other works includes using Grid computing to demonstrate optimization speed-up in data-driven reservoir studies [18], and earth system modeling [19]. However, many existing studies on Grid optimization frameworks were not targeted on multi-clusters within a distributed Grid environment, but are rather limited to single cluster.

In this paper, we present a scalable parallel evolutionary optimization framework for engineering design problems in a Grid infrastructure which we refer to as Grid Enabled Evolution (GEE). In particular, we consider the parallel evolutionary design optimization of 2D aerodynamic airfoil using the proposed GEE, where an optimal

solution is sought for a particular configuration of flight speed given by the Mach number M_∞, and the angle of attack (AOA). The 2D aerodynamic airfoil design problem represents one of the most frequently tackled computationally expensive design problems in aeronautics.

One major feature of GEE is the ability to harness computing clusters that spans across international boundaries, *i.e.*, computing clusters in Asia and Europe may be used simultaneously in the GEE. This is achieved by using standard Globus [20] and NetSolve [21] toolkits. In the GEE, the parallel evolution of multiple subpopulations are conducted across all computing clusters available on the Grid. The use of multiple subpopulations not only facilitates possible embarrassingly parallelism in the EA search, it at the same time generates greater diversity in the final design solutions.

The rest of this paper is organized as follows. In section 2, we present a brief overview on parallel GAs. Section 3 provides a brief description of the aerodynamic airfoil design problem we consider in this work while section 4 describes the GEE framework. The empirical study of GEE for hierarchical parallel evolutionary optimization of a realistic 2D aerodynamic airfoil structure is presented in section 5. Analyses of the multi-cluster GEE for hierarchical parallel evolutionary design optimization based on the result obtained from the experiments are also presented in the section. Finally section 5 concludes this paper. The preparation of manuscripts which are to be reproduced by photo-offset requires special care. Papers submitted in a technically unsuitable form will be returned for retyping, or canceled if the volume cannot otherwise be finished on time.

2 Parallel Genetic Algorithm

A well-known strength of GAs is the ease of extensions to incorporate parallelism. For instance, parallel GA represents an extension of the canonical GA (also known as simple or standard GA). Since the algorithm works with sets of populations, instead of a single individual, the basic concept of parallel GA is a simple division of the tasks in the GA across different processors. The other benefit of parallel GA is that it facilitates speciation, a process where subpopulations evolve in different directions simultaneously. They have been shown to speed up the search process as well as to obtain higher quality solutions when dealing with complex design problems. In general, the various types of parallel GAs may be classified into three main categories [8-9], i.e. the global single-population master-slave, single population fine-grained, and multi-population coarse-grained parallel GAs.

2.1 Master-slave PGA

In master-slave PGAs, it is assumed that there is only a single panmictic population, i.e., a simple GA. Like the simple GA, each individual compete and reproduce with any other in the master-slave PGA. However, unlike the simple GA, evaluations of individuals are distributed by scheduling fractions of the population among the processing slave nodes. In addition, master-slave PGA uses parallel computing to speed up

the operation of the simple GA without changing the basic operations of the sequential GA. Such a model has the advantage of ease of implementations and does not alter the searching in the canonical GA, i.e., the existing theory of simple GA still applies. Further, it poses as an efficient method of parallelization when evaluation of the fitness functions is computationally expensive. A motivating example for us is aerodynamic wing design, where one function evaluation involving the solution of the Navier–Stokes equations can take many hours of computer time [4-5].

2.2 Fine-grained PGA

Fine-grained parallel GA consists of only a single population, which is spatially structured. It is designed to run on closely-link massively parallel processing system, i.e. a machine consisting of large number of processing elements and connected in a specific high-speed topology. For instance, the population of individuals in a fine-grained PGA may be organized as a 2-Dimensional grid, since many massively parallel computers have processing elements that are connected using this topology. Consequently, selection and mating in a fine-grained parallel GA are restricted to small groups. Nevertheless, groups overlap to permit some interactions among all the individuals so that good solutions may disseminate across the entire populations. Sometimes, fine-grained parallel GA is also termed as the cellular model.

2.3 Multi-population PGA

Multiple population (or deme) GA is more sophisticated, as it consists of several subpopulations that exchange individuals occasionally. This exchange of individuals is called migration and it is controlled by several parameters. Hence, the important characteristics of multi-population GA are in the use of multiple subpopulations and migration. Multi-population PGAs are known by different names. Besides, since multi-population PGA resembles the "island model" in population genetics that considers relatively isolated demes, it is often also known as "Island GA".

Here in the GEE proposed, we consider a hybrid of the multi-population coarse-grained and master-slave type which we call it PHGA in short. In particular, we consider a multi-population coarse-grained GA model at the first level of the hybrid, where the multiple subpopulations are deployed across the pool of computing clusters available on the Grid. Subsequently, we consider the master-slave model at the second level, i.e., the subpopulation level, where all individuals in each subpopulation are farmed across all processing nodes onto the cluster where evolution of the subpopulation resides.

3 Grid Enabled Evolutionary Framework

In this section, we present the architecture of the proposed GEE for complex engineering design optimization. Like any Grid computing setups, it would be necessary to

first enable the software components as grid services so that they may be accessed within the Grid environment. Here, two grid services are created using our extended GridRPC technologies proposed in [22] for 'gridifying' existing applications. The first 'subpopulation-evolution' service is a composition of the standard GA evolutionary operators for evolving a GA subpopulation. On the other hand, the other 'airfoil-analysis' grid service is the gridified aerodynamic airfoil analysis code or the objective function of the GA for evaluating the subpopulation of chromosomes. Further for security reasons, we restrict the 'subpopulation evolution' service to be executed only on the master node of each cluster. This implies that the 'subpopulation-evolution' is developed as a Globus grid services capable of remote execution across unlimited computing clusters. In contrast, we consider the 'airfoil-analysis' as a NetSolve services that resides on all processing nodes of the clusters. This ensures all evaluations of the chromosomes are evaluated within the cluster of processing nodes where the 'subpopulation-evolution' executes in.

As described briefly in, previous section, we employ two levels of parallelism in the GA search: the first level is in the parallelism of subpopulations *i* onto the computing cluster *i* (here we consider the case where the number of subpopulations is defined to be equal to the number of computing clusters), while the second level involves the parallelism of all evaluations of chromosomes in subpopulation *i* across the processing nodes in cluster *i* only. A PHGA algorithm using the GEE framework is outline in Figure 1. Before the search starts, the services are deployed onto the clusters on the grid and registered with the resource agent. This enables the latter to search for the available computing resources and 'airfoil-analysis' service. The workflow of the Grid enabled evolutionary optimization framework is also depicted in Figure 2.

```
START PHGA
Initialize GA
while termination condition not met
   for each subpopulation
        Globus function call of 'subpopulation-reproduction'
        start "subpopulation-reproduction"
        for each chromosome
              NetSolve function calls of 'airfoil-analysis'
        end for
            GA specific operations
        end 'subpopulation-reproduction'
   end for
   chromosomes migration
end while
END PHGA
```

Fig. 1. PHGA algorithm using GEE framework.

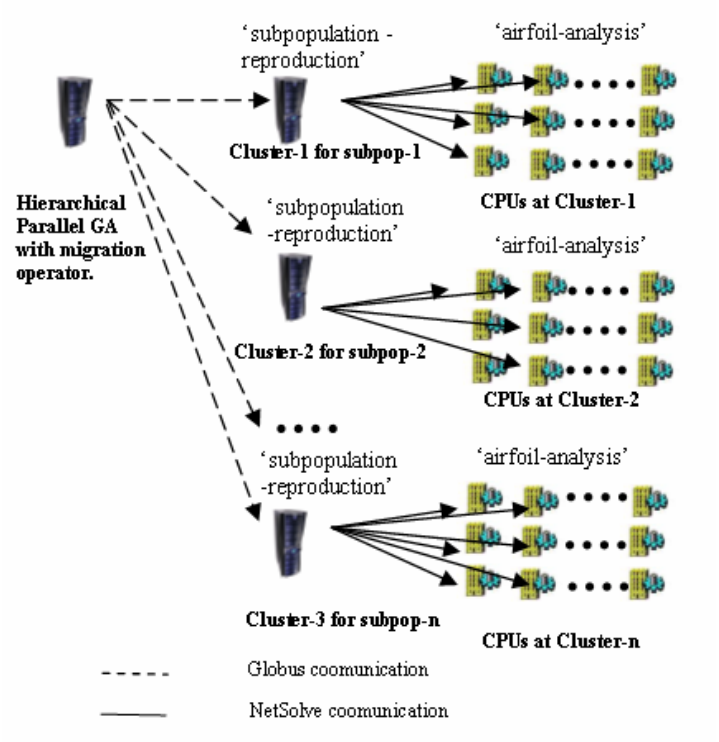


Fig. 2. Workflow of the Grid enabled evolutionary optimization framework

The detail workflow of the GEE can be outlined as nine crucial stages and is depicted in Figure 3. The nine stages are described as follows:

1) Prior to the start of the evolutionary search, the grid enabled PHGA contacts the meta-scheduler, requesting for services and resources necessary required for conducting the evolution of the GA subpopulations.
2) The metascheduler [22] then obtains a list of the available resources together with their status of availability. Such status information is acquired from services provided by the Globus Monitoring and Discovery Service (MDS) and Ganglia monitoring toolkit [22-24]. The populations of design analysis requests may then be farmed out to the available grid resources accordingly to the workload and resource information provided by the monitoring services. It is worth noting that whenever a new computing resource or software service, for instance the 'airfoil-analysis' service, is added, they get reflected in the monitoring services automatically, since mechanisms are provided to ensure any new resources are registered to the Globus MDS before they are allowed to join the Grid environment.

3) These resources information and services are then provided to allow the PHGA to proceed with the parallel evolutionary search.
4) Upon obtaining the information in relation to the resources and services, the Grid Security Infrastructure (GSI) [25] credentials are subsequently generated. This forms the authentication or authority to use the computing resources available in the system.
5) The GridFTP [26] mechanism provided in the GEE then transfers the subpopulations in the form of ASCII data files to the computing clusters that has the correct services to perform genetic evolution and fitness evaluation.
6) Parallel evolution of the multiple subpopulations is then started at the remote computing clusters using the Globus job submission protocol.
7) Whenever the Globus Resource Allocation Manager (GRAM) [27] gatekeeper of a cluster receives a request to start the 'subpopulation-evolution' service, an instance of this service get instantiated on the master node of the cluster. Subsequently, the nested set of 'airfoil-analysis' service requests within the 'subpopulation-evolution' service can then be farmed across the processing nodes within the cluster using any cluster level local scheduler, for example, NetSolve, Sun Grid Engine [28], Condor [29] or others. The responsibility of the local agent in each cluster is to perform local scheduling and resource discovery across the processing nodes within a cluster.
8) When the 'airfoil-analysis' services completed execution, the fitness values of the chromosomes are conveyed back to the 'subpopulation-evolution' service so that standard GA operations such as mutation, crossover and selection can take place.
9) Similarly when the 'subpopulation-evolution' services deployed across the remote clusters completes, the resultant evolved subpopulations are then marshaled back to the main PHGA program using the Global Access to Secondary Storage (GASS) [30]. The migration operation of the PHGA then proceeds. The process repeats until the termination condition is met.

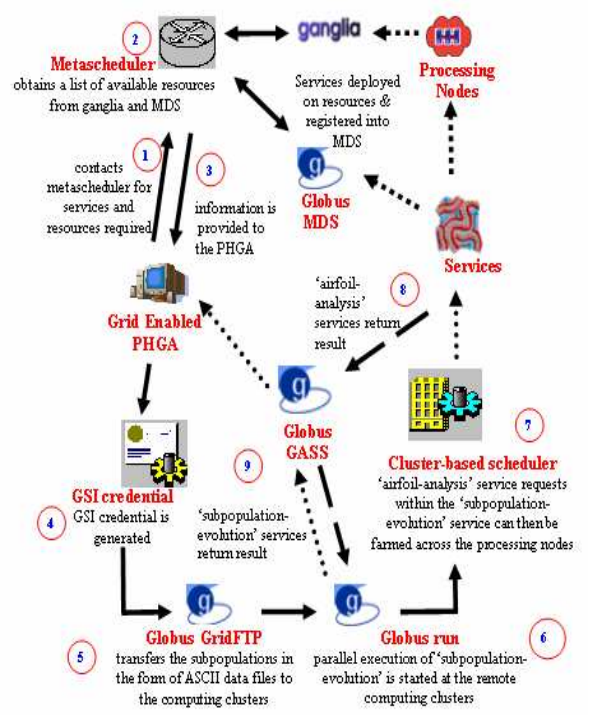


Fig. 3. GEE Workflow.

4 Aerodynamic Airfoil Design Problem

In this section, we present a 2D aerodynamic airfoil design problem, particularly, the subsonic inverse pressure design problem used in our present study. The target pressure profile is generated from the NACA 0012 airfoil, which itself is the baseline shape. The airfoil geometry is characterized using 24 design variables as depicted in Figure 4. Hence, there exists for this problem a global solution corresponding to $z_1 = \ldots = z_{24} = 0$. The free-stream conditions in this problem are subsonic speed of Mach 0.5, and zero angle of attack (AOA), corresponding to symmetric pressure profiles on the upper and lower walls.

It is worth noting that the inverse problem constitutes a good test problem for validating the convergence property of GEE, since the optimal design is known in advance. At the same time, it facilitates our study on complex engineering design optimization problems of variable-fidelity. Secondly, the inverse design problem also has a practical purpose, as the designer generally has an idea of the desired pressure

profile that yields good aerodynamic performance. For example, in transonic design, a shock front on the upper surface generally leads to undesirably high pressure drag that degrades the efficiency of the airfoil. A typical approach to inverse pressure design is to 'smoothen' the pressure distribution on the upper-surface in a way that maintains the area under the curve, so as to maintain the lift force generated by the airfoil. Thus, the inverse pressure design problem can be formulated as a minimization problem of the form:

$$I(w,S) = \frac{1}{2}\int_{\text{wall}} (p - p_d)^2 d\sigma \tag{1}$$

subject to constraints.

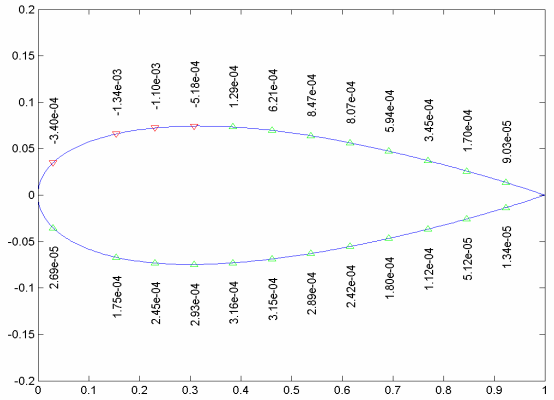


Fig. 4. A 2D airfoil geometry characterised using 24 design variables with the NACA 0012 as baseline.

5 Empirical Study

In this section we present an empirical study of the PHGA using the GEE framework for complex engineering design, particularly, aerodynamic airfoil design optimization.

5.1 Experimental Setup

The control parameter of the PHGA are configured as follows: population size for every subpopulation is 80, crossover probability of 0.9, mutation probability of 0.1, migration period of 10 generations with 1 chromosome per migration phase, linear

fitness scaling, elitism, and termination upon maximum number of generation 100. Further, three computing clusters used in our study and are listed in Table 1. Here we provide the processing power of these computing clusters measured based on the commonly used Linpack benchmark problem [31].

Table 1. Specifications of the clusters used.

Cluster Name	No. of CPUs	CPU Clock	Memory	MFLOPS (average)
pdcc	28	PIV Xeon 3.6GHz	10G	920
pdpm	20	PIV Xeon 2.6GHz	10G	800
surya	21	PIII 450MHz PIII 550MHz PIII 733MHz	6G	150

The respective Million Floating Point Operations (MFLOPS) of the computing clusters are also tabulated in Table 1. It can be observed that all the three clusters we considered here have very different MFLOPS values. The *pdcc* cluster has a significantly higher MFLOPS than the *surya* cluster. Clearly they are heterogeneous clusters and *pdcc* and *pdpm* are much more powerful clusters than *surya*.

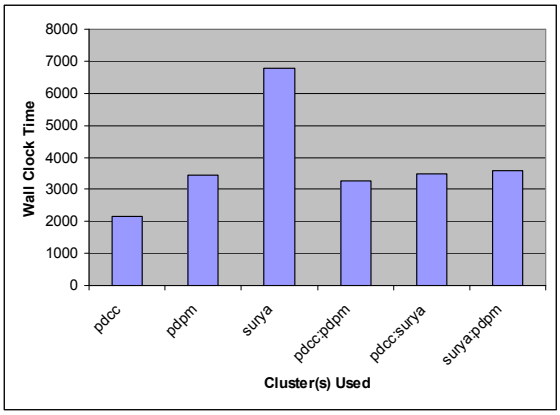
5.2 Experimental Results and Analysis

Using 24 design variables and the cost function in equation 1, the PHGA with the GEE framework is applied for the optimization of the subsonic inverse pressure design problem described in section 4. Further we consider two separate analysis codes or variable-fidelity in our study. The low-fidelity and moderate-fidelity analysis codes considered here represent realistic computationally inexpensive and expensive design problems, respectively. The exact wall clock time for a single airfoil analysis on the three heterogeneous is summarized in Table 2. A single moderate-fidelity analysis of the airfoil geometry using an Euler CFD solver takes around 110 seconds on a Pentium III processor, while a low-fidelity takes around 10 seconds. From Table 2, it may also be observed that the time taken for each clusters to complete an analysis is clearly significantly different on the moderate-fidelity analysis code.

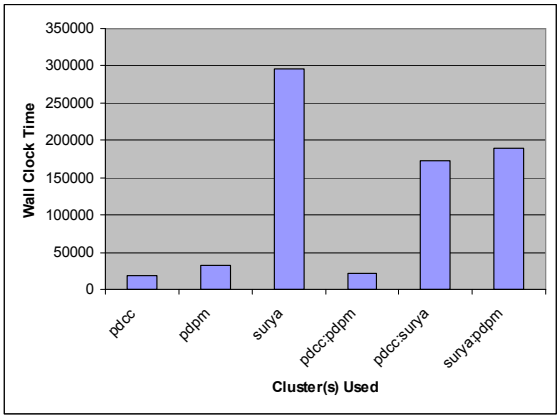
Table 2. Wall clock time to conduct a single of the variable-fidelity airfoil code on clusters on the clusters.

Variable Fidelity	Low-fidelity Analysis code			Moderate-fidelity Analysis code		
Cluster	*surya*	*pdpm*	*pdcc*	*surya*	*pdpm*	*pdcc*
Wall Clock Time	10 s	9 s	8 s	110 s	54 s	37 s

For each set of experimental study, 10 PHGA runs using the GEE framework were conducted and the average of the runs are reported. The average wall clock time taken by the PHGA to complete a maximum of 100 generations on a GEE with single cluster or multiple clusters for 2 subpopulations is depicted in Figure 5. From these results, the PHGA with 2 subpopulations and a GEE a single *pdcc* cluster appears to complete the maximum of 100 generations much earlier than all other single cluster or two-clusters combinations. Specifically, all two-cluster combinations of the three clusters have longer wall clock time than using the *pdcc* alone to complete the evolution of two subpopulations in parallel. This is a consequence of our present restriction on the GEE which enforces a one-to-one mapping between subpopulations and computing clusters. This implies that we assign the same workload to all clusters regardless of their specifications, *i.e.*, CPU clock, number of processing nodes, memory, MFLOP and etc. The effect is more evident on the Figure 5(b) than (a), due to the larger differences on the execution time to perform a single analysis of the moderate-fidelity analysis code on the three clusters. In addition, the extreme poor performance of the PHGA-two subpopulations optimization runs on the *surya* cluster, *i.e.*, 10 times slower than *pdcc* alone, is a result of overloading due to mismatch between the low memory specification of *surya* as opposed to the heavy memory requirement of many moderate-fidelity analysis code executing in parallel.



(a) Low-fidelity analysis code



(b) Moderate-fidelity analysis code

Fig. 5. Average wall clock time for PHGA-2 subpopulations using the GEE framework for variable-fidelity analysis code.

Besides, due to the large differences in completion time by the clusters to complete the evolution of a subpopulation, all subpopulations have to wait for the slowest cluster to complete evaluations of all chromosomes and evolution before any migration operation may take place and proceed with the next generation. Hence, it appears proper parallelism and division of the subpopulations and chromosomes evaluations is crucial to the performance of the PHGA and GEE when operating on a heterogeneous computing cluster environment such as the grid. It may not always be the case that using greater computing resources would provide significant speed-up on the optimization search of the PHGA. Clearly, a GEE that enforces a one-to-one mapping between subpopulations and computing clusters limits the potential of the PHGA to attain high performance in optimization efficiency. Here, to fully utilize the grid computing cluster resources for complex engineering design of computationally expensive optimization problems, dynamic bundling of chromosomes is proposed. Here we pool all chromosomes in the subpopulations together and submit chromosomes to clusters according to their specifications, for instance based on their MFLOPS and CPU numbers. In this way, more chromosomes are sent to the high-end clusters than to its lower-end counterparts. In our case, as the number of CPUs are all almost the same, i.e. varying from 20-28 CPUs for the three clusters (see Table 1), we consider only MFLOPS as the criterion for dynamic bundling. For instance, using the MFLOPS of the three heterogeneous clusters defined in Table 1, dynamic bundling is carried out as follows:

$$R_i = \frac{C_i}{\sum_{i=1}^{n} C_i} \times Pop \quad (2)$$

where R_i = Ratio of chromosomes sent to the cluster i.
C_i = MFLOPS of cluster i.
n = Total number of clusters to be used.

Using the GEE with dynamic bundling, the PHGA with 2 subpopulations is once again used for optimizing the airfoil design problem. Note that all other parameters are kept the same as previous experiments. Using two-clusters, *pdcc:surya* and *pdcc:pdpm*, the chromosomes are bundled as tabulated in Table 3 using equation (2).

Table 3. Chromosomes distribution ratio based on the MFLOPS of the clusters for 2 subpopulations (a total of 160 chromosomes).

Cluster Name	MFLOPS Ratio	Chromosomes Distribution Ratio
pdcc : *surya*	920:150	138:22
pdcc : *pdpm*	920:800	86:74

The average wall clock time of the experiments are depicted in Figure 6. It can be observed that with the use of the dynamic bundling GEE for PHGA optimization, the wall clock time is significantly improved on all the two-cluster combinations. This is because chromosomes are now sent to more powerful clusters for evaluations than their less powerful counterparts on the grid. In effect, it is possible to conclude that the use of the grid and hence the proposed GEE for facilitating embarrassingly parallelism in PHGA can provide significant speed-up on the optimization search.

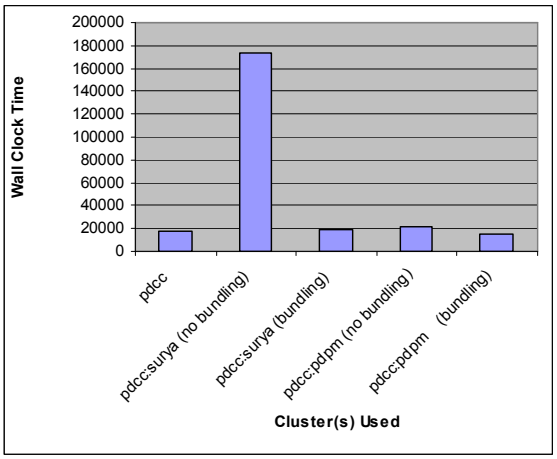


Fig. 6. Average wall clock time for PHGA-2 subpopulations using the dynamic bundling GEE framework for moderate-fidelity analysis code.

6 Conclusion

In this paper, we have presented the Grid Enabled Evolution framework, which employs Grid computing technologies for facilitating embarrassingly parallelism in multi-population parallel GA optimization of computationally expensive design problems. Using a multi-level parallelisation strategy of hierarchical parallel GAs in a Grid environment, we present the evolutionary optimization of a realistic 2D aerodynamic airfoil structure. Based on the experimental results obtained, an assessment and analysis of the GEE is performed. The negative consequences using of heterogeneous clusters in a realistic grid environment based on a GEE with one-to-one mapping between subpopulations and computing clusters is discussed. Further, dynamic bundling based on the MFLOPS metric of the clusters is also proposed and demonstrated to provide significant speed-up in the PHGA optimization search. From our analysis, it is possible to conclude that a grid enabled hierarchical parallel evolutionary algorithm is not mere hype but does offers as a credible alternative for providing significant speed-up to complex engineering design optimization.

Acknowledgments

The authors would like to thank the Parallel and Distributed Computing Centre at the School of Computer Engineering, Nanyang Technological University for providing support and computing resources to this work.

References

[1] Goldberg D.E., "Genetic Algorithms in Search, Optimization and Machine Learning", 1989.

[2] M. Olhofer, T. Arima, T. Sonoda and B. Sendhoff, "Optimization of a stator blade used in a transonic compressor cascade with evolution strategies", *Adaptive Computing in Design and Manufacture (ACDM)*, Springer Verlag, pp. 45-54, 2000.

[3] P. Hajela, J. Lee., "Genetic algorithms in multidisciplinary rotor blade design", *Proceedings of 36th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Material Conference*, New Orleans, pp. 2187-2197, 1995.

[4] Y. S. Ong and A.J. Keane, "Meta-Lamarckian in Memetic Algorithm", *IEEE Trans. Evolutionary Computation*, Vol. 8, No. 2, pp. 99-110, April 2004.

[5] I. C. Parmee., D. Cvetkovi., A. H. Watson, C. R. Bonham, "Multi objective satisfaction within an interactive evolutionary design environment", *Evolutionary Computation*, Vol. 8, No. 2, pp. 197-222, 2000.

[6] P. B. Nair and A. J. Keane, "Passive Vibration Suppression of Flexible Space Structures via Optimal Geometric Redesign", *AIAA Journal* 39(7), pp. 1338-1346, 2001.

[7] Baluja S., "The Evolution of Genetic Algorithms: Towards Massive Parallelism", *Machine Learning: Proceedings of the Tenth International Conference*, 1993.

[8] Mariusz Nowostawski, Riccardo Poli, "Parallel Genetic Algorithm Taxonomy", *Proceedings of the Third International conference on knowledge-based intelligent information engineering systems (KES'99)*, pages 88-92, Adelaide, August 1999. IEEE.

[9] Cantu-Paz E., "A Survey of Parallel Genetic Algorithms" , *Calculateurs Paralleles, Reseaux et Systems Repartis* vol. 10 No. 2 pp. 141-171, 1998.

[10] Baluja S., "The Evolution of Genetic Algorithms: Towards Massive Parallelism", *Machine Learning: Proceedings of the Tenth International Conference*, 1993.

[11] Huyse L. and Lewis R.M., "Aerodynamic Shape Optimization of Two-dimensional Airfoils Under Uncertain Operating Conditions", Hampton, Virginia: ICASE NASA Langley Research Centre, 2001.

[12] Padula S.L. and Li W., "Robust Airfoil Optimization in High Resolution Design Space", Hampton, Virginia: ICASE NASA Langley Research Centre, 2002.

[13] Ong Y.S., Lum K.Y., Nair P.B., Shi D.M. and Zhang Z.K., "Global Convergence of Unconstrained and Bound Constrained Surrogate-Assisted Evolutionary Search in Aerodynamic Shape Design Solvers", *IEEE Congress on Evolutionary Computation*, Special Session on Design Optimization with Evolutionary Computation", 2003.

[14] Foster I. and Kesselman C., editors, "The Grid: Blueprint for a New Computing Infrastructure," Morgan Kaufman Publishers,1999.

[15] Foster I., Kesselman C., and Tuecke S., "The Anatomy of the Grid: Enabling Scalable Virtual Organizations" , *International J. Supercomputer Applications*, vol. 15, no. 3, 2001.

[16] Baker M., Buyya R., Laforenza D., "The Grid : International Efforts in Global Computing", *International Conference on Advances in Infrastructures for Electronic Business, Science, and Education on the Internet*, 2000.
[17] Cox J., "Grid enabled optimisation and design search for engineering (Geodise)", in: NeSC Workshop on Applications and Testbeds on the Grid, 2002.
[18] Parashar M. et. al., "Application of Grid-enabled Technologies for Solving Optimization Problems in Data-Driven Reservoir Studies", submitted to Elsevier Science, 2004.
[19] Price A.R. et. al., "Tuning GENIE Earth System Model Components using a Grid Enabled Data Management System", School of Engineering Sciences, University of Southampton, UK.
[20] Foster I., "The Globus Toolkit for Grid Computing", *Proceedings of the 1st International Symposium on Cluster Computing and the Grid*, 2001.
[21] Agrawal S., Dongarra J., Seymour K., Vadhiyar S., "NetSolve: past, present, and future; a look at a grid enabled server", 2002.
[22] Ho Q.T., Cai W.T., and Ong Y.S., "Design and Implementation of An Efficient Multi-cluster GridRPC System", Cluster and Computing Grid, 2005.
[23] Globus: Information Services/MDS, [online] http://www-unix.globus.org/toolkit/mds.
[24] Massie M., Chun B., and Culler D., "The Ganglia Distributed Monitoring System: Design, Implementation, and Experience", Technical report, University of California, Berkeley, 2003.
[25] Tuecke S., "Grid Security Infrastructure (GSI) Roadmap", Internet Draft Document: draft-gridforum-gsi-roadmap-02.txt, 2001.
[26] The Globus Project, "GridFTP Universal Data Transfer for the Grid", The Globus Project White Paper, 2000.
[27] The Globus Resource Allocation Manager (GRAM) [online] http://www-unix.globus.org/ developer/resource-management.html.
[28] Geer D., "Grid Computing Using the Sun Grid Engine", Technical Enterprises, Inc., 2003.
[29] Frey J., Tannenbaum T., Livny M., Foster I., Tuecke S., "Condor-G: A Computation Management Agent for Multi-Institutional Grids", *Proceedings of the Tenth IEEE Symposium on High Performance Distributed Computing (HPDC10)*, 2001.
[30] Samwel B., "Data Management in Computational Grid," [online] http://www.liacs.nl/home/llexx/gc/dm/pdf.
[31] Dongarra J.J., Luszczek P., Petitet A., "The LINPACK Benchmark: Past, Present, and Future.", University of Tennessee, Department of Computer Science, Knoxville, USA, 2002.