Efficient Hierarchical Parallel Genetic Algorithms Using Grid Computing

Dudy Lim [a] Yew-Soon Ong [a,*] Yaochu Jin [b] Bernhard Sendhoff [b]
Bu-Sung Lee [a]

[a] *School of Computer Engineering, Nanyang Technological University*
Nanyang Avenue, Singapore 639798

[b] *Honda Research Institute Europe GmbH*
Carl-Legien Strasse 30, 63073 Offenbach

Abstract

In this paper, we present an efficient Hierarchical Parallel Genetic Algorithm framework using Grid computing (GE-HPGA). The framework is developed using standard Grid technologies and has two distinctive features, 1) an extended GridRPC API to conceal the high complexity of Grid environment, and 2) a metascheduler for seamless resource discovery and selection. To assess the practicality of the framework, theoretical analysis on the possible speed-up offered is presented. Empirical study on GE-HPGA using a benchmark problem and a realistic aerodynamic airfoil shape optimization problem for diverse Grid environments having different communication protocols, cluster sizes, processing nodes, at geographically disparate locations also indicates that the proposed GE-HPGA using Grid computing offers a credible framework for providing significant speed-up to evolutionary design optimization in science and engineering.

Key words: Grid computing, parallel Genetic Algorithms
PACS: 01.30.−y

* Corresponding author.
Email addresses: dlim@ntu.edu.sg (Dudy Lim), asysong@ntu.edu.sg (Yew-Soon Ong), yaochu.jin@honda-ri.de (Yaochu Jin), bernhard.sendhoff@honda-ri.de (Bernhard Sendhoff), ebslee@ntu.edu.sg (Bu-Sung Lee).

1 Introduction

Evolutionary Algorithms (EA) as a family of computational models inspired by the natural process of evolution, have been applied with a great degree of success to complex design optimization problems [1][2][3][4][5]. In Genetic Algorithms (GA) [6], a subclass of EA, potential solutions are encoded into a simple chromosome-like data structure, and recombination and mutation operators are repeatedly applied to a population of such potential solutions until a certain termination condition is reached. Their popularity lies in their ease of implementation and the ability to locate designs close to the global optimum. However, thousands of calls to the analysis codes are often required to locate a near optimal solution in most conventional GA. The increasing use of time-consuming high-fidelity analysis codes in science and engineering for studying the effect of altering key design parameters on product performance has further led to even longer and intractable design cycle times. Fortunately, another well-known strength of GA is the ability to partition the population of individuals among multiple computing nodes. Doing so allows sublinear speedups in computation and even super-linear speedups [7][8] if possible algorithmic speed-up is also considered. Over the last decade, many variants of parallel GAs exist for exploiting the explicit parallelism of GAs. The reader is referred to [9][10] for some excellent expositions of parallel GAs.

While numerous research on parallel GAs for various distributed computing technologies have since been reported, most studies and applications of parallel GAs have been on using dedicated and homogeneous computing nodes [11][12][13]. The focuses have been on small-scale dedicated computing resources and are not easily extendable towards harnessing computing resources that span across laboratories or even organizations at disparate geographical locations. The issue of standards is one major challenge, as many existing technologies do not have the common interfaces and methods of doing things. In addition, the parallel evaluations in PGAs are often constrained by the limited commercial licenses a design team may access seamlessly for analyzing the designs at once. This is primarily due to the high costs associated with the site licenses of commercial analysis packages, for example, ANSYS [14], FLUENT [15], SYSNOISE [16], etc. It is for these reasons that the recent advent of what is termed Grid computing [17][18] has gained widespread attention, as it sets about the notion of establishing a set of open standards for distributed resources and ubiquitous services. Advances in Grid computing have also fueled the research and development of Grid problem solving environment for complex design in science and engineering [19][20][21]. While many open issues remain when using Grid computing across geographically disparate laboratories, scheduling, resource management [22], and benchmarking Grids [23] are among some of the more frequently discussed topics.

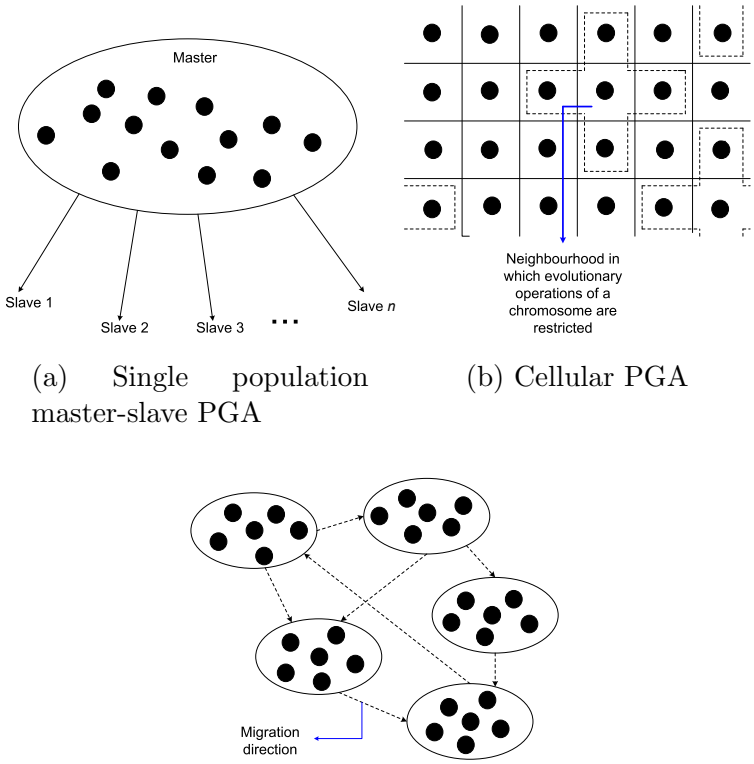
To complement existing works on parallel GAs, we present an efficient hierarchical parallel genetic algorithm framework using Grid computing technologies in this paper. The framework provides unique features that include 1) an extended GridRPC element for concealing the high complexity of Grid computing environment from the users and 2) a meta-scheduler for seamless resource discovery and selection in a Grid environment. Subsequently, experiments are carried out to assess the efficacy of the proposed framework for parallel evolutionary optimization under diverse Grid environments. Finally, we show that the Grid-enabled parallel genetic search generates significant speed-up for evolutionary design optimization.

The rest of this paper is organized as follows. In Section 2, we present a brief overview on parallel GAs, as one of the successful design optimization methodology used in science and engineering. Section 3 describes the proposed Grid-Enabled Hierarchical Parallel Genetic Algorithm (GE-HPGA) optimization framework. Theoretical analysis on speed-up by GE-HPGA is also presented in the section. Section 4 presents an empirical study on GE-HPGA in Grid computing environments containing mixtures of diverse computing clusters that are geographically disparate, using a benchmark problem and a realistic aerodynamic airfoil shape design problem. Finally, Section 5 concludes this paper with a brief summary.

2 Parallel Genetic Algorithms

Genetic Algorithms [6] are probabilistic meta-heuristic methods inspired by the 'survival of the fittest' principle of neo-Darwinian theory of evolution. Artificial creatures are created and put into competition in a struggle for life and only the survivors are allowed to reproduce. A new population will be created using biologically inspired operators such as crossover, mutation, and selection, and the process repeats until some search termination criteria are reached. A genetic algorithm without any structure is usually referred to as a panmictic GA. The Parallel Genetic Algorithms (PGAs) are extensions of the panmictic GA. The well-known advantage of PGAs is their ability to facilitate speciation, a process by which different subpopulations evolve in diverse directions simultaneously. They have been shown to speed up the search process and to attain higher quality solutions on complex design problems [24][25][26]. In addition to the parallel panmictic GA, two popular parallel structured GAs include the island and cellular GA [9][10]. In this section, we present a brief review on these algorithms. The three basic models of PGA are as illustrated in Figure 1.

Master-slave PGA. In master-slave PGAs, it is assumed that there is only a single panmictic population, i.e., a canonical GA. However, unlike the



(a) Single population master-slave PGA

(b) Cellular PGA

(c) Island PGA

Fig. 1. Basic models of Parallel Genetic Algorithms.

canonical GA, evaluations of individuals are distributed by scheduling fractions of the population among the processing slave nodes. Such a model has the advantage for ease of implementation and does not alter the search behavior of a canonical GA.

Fine-grained or Cellular PGA. Fine-grained PGA consists of only a single population, which is spatially structured. It is designed to run on closely-linked massively parallel processing system, i.e., a computing system consisting a 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 two-dimensional grid. 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 PGA.

Multi-population or Multi-Deme or Island PGA. Multiple population (or deme) PGA may be 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. Multi-population PGAs are also known by various names. Since they resemble

the 'island model' in population genetics that considers relatively isolated demes, it is also often known as the island model PGA.

Hierarchical PGA. The various PGA models may also be hybridized to produce other new Hierarchical PGA (HPGA) models. For instance, one may form a hierarchical PGA that combines a multi-population PGA (at upper level) and a fine-grained PGA or master-slave PGA (as what we consider throughout the development of the framework in this paper), or even another level of island PGA (at lower level). Basically, any combination of two or more of the three basic forms of PGA is an HPGA.

3 Grid-enabled Hierarchical Parallel Genetic Algorithm (GE-HPGA) Framework

In this section, we present a Grid-enabled HPGA, which we call GE-HPGA in short. Various Grid enabling technologies have been considered in developing the GE-HPGA framework and these are discussed in Section 3.1. The detailed workflow of GE-HPGA is then discussed in Section 3.2. Theoretical analysis on the speed-up offered by the proposed framework is also considered in Section 3.3.

3.1 Grid Enabling Technology

In this section, we discuss some of the key Grid technologies used in developing the GE-HPGA. Globus Toolkit [1] [27], Commodity Grid Kit (CogKit) [2] [28], Ganglia monitoring tool [3] [29], and NetSolve [4] [30] are some of the core Grid technologies used in developing the GE-HPGA. From a survey of the literature, it is possible to establish that the existing GridRPC standard lacks mechanisms for automatic resource discovery and selection of Grid resources. As a result, users are naturally required to perform manual look-up and select resources when assigning new tasks onto the Grid computing resources. This is clearly impractical since large amount of computing resources exist on the Grid and are often dynamic in practice. Furthermore, most optimization problems have a large number of evaluation tasks that require many identical

[1] **Globus** is the de-facto Grid Middleware, which provides the Grid infrastructures for security, data management, resource management, and information service.

[2] **CogKit** is the API for Globus.

[3] **Ganglia** is a distributed monitoring system for high-performance computing systems such as clusters and Grids, which has been in deployed on over 500 clusters in the world.

[4] **NetSolve** is a Grid tool, based on the agent-client-server model that enables the clients to access any services provided by servers registered to an agent.

computations of different analysis parameter sets represented in the form of chromosomes. It is therefore extremely inefficient if the interactions between the client (master) and resources (slaves) are repeated many times for the same remote procedure call. The uniqueness of our GE-HPGA framework is therefore an extended GridRPC API with the inclusion of a meta-scheduler.

The meta-scheduler performs discovery, bundling and load balancing using online information gathered from the computing clusters and Grid services [5] that exist on the Grid. In our GE-HPGA framework, the Globus Monitoring and Discovery Service (MDS) [31] and Ganglia [29] have been used to provide the online information. The MDS maintains a database of available resource information and acts as a centralized directory service keeping track of the resources, their locations on the Grid and how they may be consumed. Ganglia, on the other hand, monitors and provides workload information about the available clusters, computing nodes and Grid services.

In any Grid computing setup, it is necessary to first enable the software components as Grid services so that they can live in a Grid environment. Hence, our GE-HPGA is equipped with an extended GridRPC API [32][20] based on the Commodity Grid Kit (CoGKit) for 'gridifying' new or existing analysis/simulation codes or objective/fitness function as Grid services. This choice is down to its simplicity in implementations and its ability to offer high-level abstraction, thus concealing the high-level of complexity of Grid computing environments from the end users. For the sake of brevity, we will not describe the implementation details of our extended GridRPC API here but refer the readers to [20] for further exposition. Other Grid technologies utilized in the present work include the Globus Grid Security Infrastructure (GSI) [33] for secure and authenticated access over the Grid. For data flow, the Globus Grid File Transfer Protocol (GridFTP) [34] is used for conducting all forms of data transfer.

3.2 GE-HPGA Workflow

In this section, we outline the workflow of the GE-HPGA framework. In particular, we considered a two-level HPGA in the present study, i.e., Level 1: Island model PGA and Level 2: Master-Slave PGA. The two levels of the hierarchical PGA are 'gridified' to form the 'subpopulation evolution' and 'chromosome evaluation' Grid services as depicted in Figure 2. In each cluster, a 'subpopulation evolution' Grid service is put in place for remote invocation using the

[5] Here, 'Grid service' refers to any shared software components that is wrapped to live on the Grid environment. In the context of optimization in science and engineering, the Grid services are wrapped forms of analysis/simulation codes or objective/fitness functions.

Globus job submission protocol. The 'chromosome evaluation' invocation, on the other hand, may be realized using any local cluster scheduler, for example NetSolve [30], Sun Grid Engine [35], Condor [36] or otherwise. The detailed workflow of the GE-HPGA can be outlined as nine crucial stages and is depicted in Figure 3.

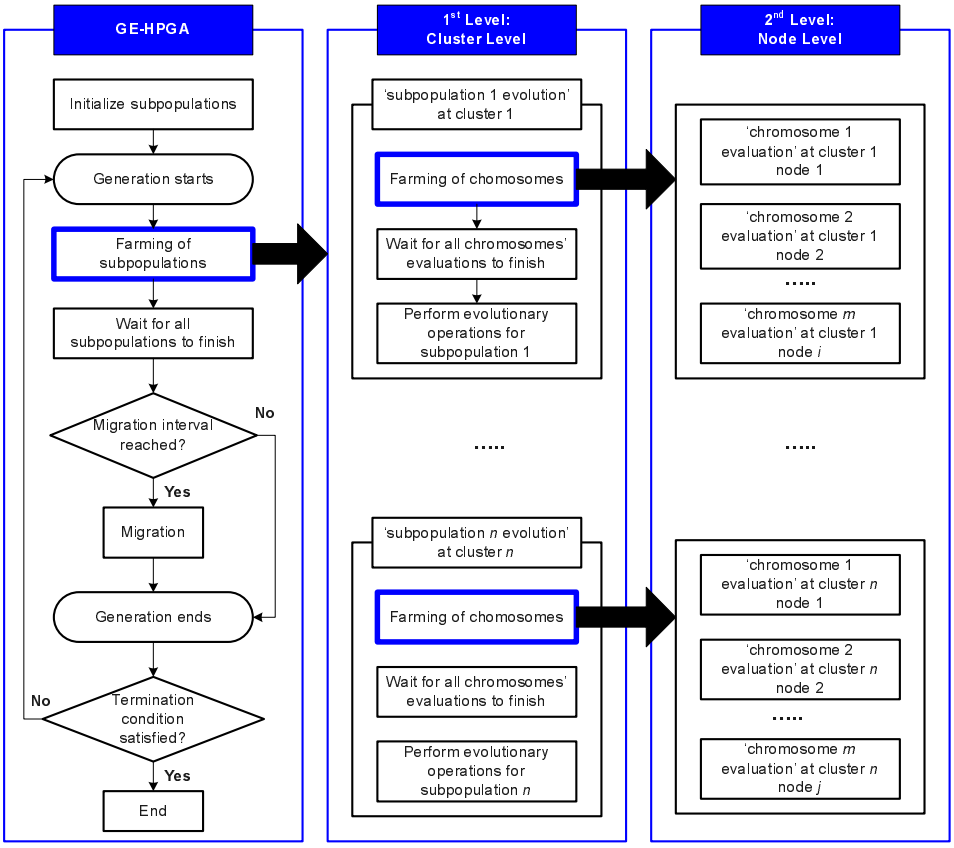


Fig. 2. The two levels of parallelism in GE-HPGA. *1st Level*: Farming of subpopulations to the computing clusters for genetic evolution. *2nd Level*: Farming of chromosomes onto computing nodes in the cluster for fitness evaluations.

These stages are described as follows:

(1) Prior to the start of the evolutionary search, the GE-HPGA master program contacts the meta-scheduler to request for suitable computing nodes and the 'subpopulation evolution' and 'chromosome evaluation' Grid services.
(2) The meta-scheduler then obtains a list of available resources together with their status of availability. Such status information is acquired from the Globus MDS and Ganglia. It is worth noting that mechanisms to automatically reflect new computing clusters and processing nodes or software services are provided to ensure proper registrations to Globus MDS whenever they join the Grid.
(3) Grid computing resource and service information maintained by the Globus

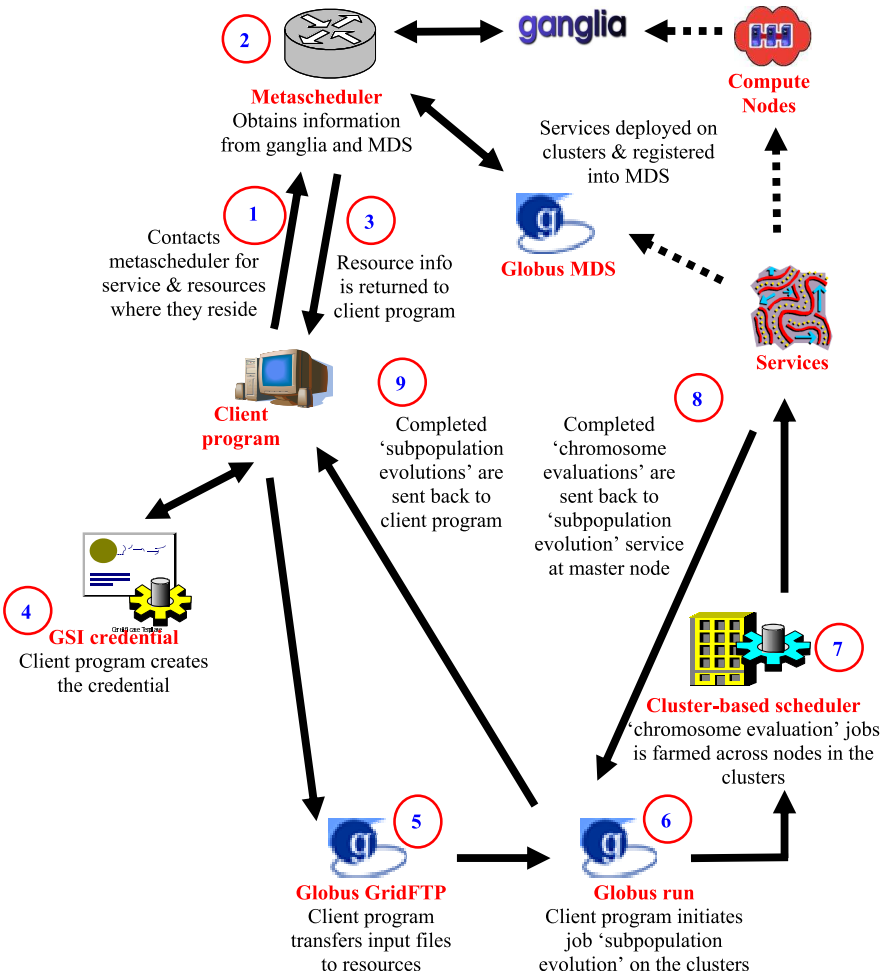


Fig. 3. Workflow of GE-HPGA framework.

MDS and Ganglia are then provided to the GE-HPGA master program to proceed with parallel evolutionary search.

(4) To access Grid resources, Grid Security Infrastructure (GSI) credentials are subsequently generated. This forms the authentication or authority for consuming any form of resources living in the Grid environment.

(5) Represented in the form of ASCII or XML data files, the HPGA subpopulations are then transferred onto the identified remote computing clusters that offer the required 'subpopulation evolution' and 'chromosome evaluation' services.

(6) Parallel evolution of the multiple subpopulations then commences at the selected remote computing clusters using Globus job submission protocol. Whenever the cluster receives a request to launch the 'subpopulation evolution' service, an instance of this service gets instantiated only at the main (master) node of the respective cluster.

(7) The subpopulation of 'chromosome evaluation' service requests that nest within the 'subpopulation evaluation' service are subsequently farmed across the field of processing nodes that is available in the cluster through

NetSolve. The role of Netsolve is to conduct scheduling and resource discovery in a single computing cluster.

(8) Once all 'chromosome evaluation' service requests are completed, the obtained fitness of chromosomes is marshaled back to the 'subpopulation evolution' service to undergo evolutionary operations involving genetic mutation, crossover and selection.

(9) The evolved subpopulations are then marshaled back to the GE-HPGA master to proceed with the migration operation. This process repeats until the search termination criteria are met.

3.3 Theoretical Analysis on GE-HPGA

One of the major performance issues when running a parallel algorithm is how much speed-up they can offer compared to a sequential run of the same algorithm [37]. This speed-up (S) measurement can be defined by:

$$S = \frac{T_s}{T_p} \tag{1}$$

where T_s and T_p denote the execution time when the algorithm is executed in serial and parallel, respectively. The whole computation of a parallelizable algorithm can be divided into three major parts, i.e. sequential computation (λ), parallelizable computation (γ), and parallelization overheads (O). In many cases, the communication overhead incurred represents the most dominating parallelization overheads. For a problem of size n and number of processors p, it is possible to derive from equation (1) that

$$S(n,p) = \frac{\lambda(n) + \gamma(n)}{\lambda(n) + \frac{\gamma(n)}{p} + O(n,p)} \tag{2}$$

which provides an upper bound for the maximum speed-up achievable by a parallel computer having p processors when computation is divided equally among the processors. Further simplification to the upper bound of maximum speedup may be obtained by using *Amdahl's Law* [38] as shown in equation (2) becomes:

$$S(n,p) = \frac{\lambda(n) + \gamma(n)}{\lambda(n) + \frac{\gamma(n)}{p} + O(n,p)} \leq \ S^{max}(n,p) = \frac{\lambda(n) + \gamma(n)}{\lambda(n) + \frac{\gamma(n)}{p}} \tag{3}$$

This provides a theoretical bound on the maximum speed-up that can be achieved by the parallel algorithm.

3.3.1 Execution Time of GE-HPGA

Based on equations (1),(2), and (3), we analyze the theoretical execution time of the GE-HPGA. From Figure 4, two forms of communication overhead may be observed in GE-HPGA. These are the inter-cluster and intra-cluster communication overheads and are denoted here as O_{inter} and O_{intra}, respectively.

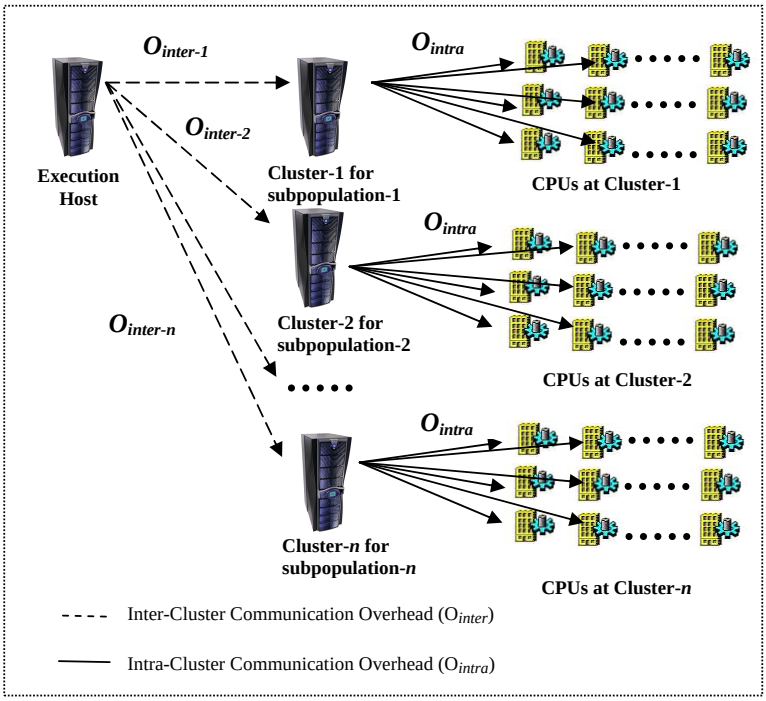


Fig. 4. Communication overheads of GE-HPGA framework.

The inter-cluster communication overhead represents the time incurred for Globus remote job execution and transferring of GA subpopulation details onto resource clusters using the GridFTP protocol. Intra-cluster communication overheads on the other hand, are attributed to parallelizing a subpopulation of chromosomes onto the processing nodes in a cluster using Netsolve. To study the impact of the communication overheads on search efficiency, we analyze the computational complexity of n-subpopulation GE-HPGA for single cluster and n-cluster Grid environments separately.

Case 1. n Subpopulations on Single Cluster. We first analyze the n subpopulations HPGA on a single cluster Grid environment. This models the case where evolutions of PGA subpopulations are conducted in sequential manner. Hence there is only a single level of parallelism consisting only of level 2 HPGA depicted in Figure 2, where the subpopulation of chromosomes is transferred onto the processing nodes of a single cluster for fitness evaluations. Assuming the execution host is also used for subpopulations evaluations, the total wall

clock time for such a process can be derived as:

$$T_s = nG(CO_{intra} + \alpha F) \tag{4}$$

where G : Number of generations
n : Number of subpopulations or clusters
C : Number of chromosomes in a subpopulation
α : Parallelism factor of a cluster, which is a function of the population size and CPU specifications
O_{intra} : Intra-cluster communication overhead to parallelize a chromosome within a cluster
F : Wall clock time to complete a single fitness evaluation

Case 2. n Subpopulations across n Clusters. Next, we model the case where n subpopulations HPGA are evolved across n number of clusters in parallel, i.e., the subpopulation size is assumed to be equal to cluster size here. The total wall clock time in this case becomes:

$$T_p = G\left(\left(\sum_{i=1}^{n} O_{inter}^{i}\right) + CO_{intra} + \alpha F\right) \tag{5}$$

where $\sum_{i=1}^{n} O_{inter}^{i}$ represents the total inter-cluster communication overhead to transfer n PGA subpopulations onto n computing clusters in the Grid environment. Without loss of generality, equation (5) remains valid for the case where the n-subpopulation GE-HPGA is available only at a single cluster, while the subpopulations are evolved in a parallel manner. The differences lie in 1) higher parallelism factor, α and 2) lower inter-cluster communication overheads, O_{inter}, since the subpopulations are executed on the same cluster in parallel.

3.3.2 Speed-up of GE-HPGA

Here, we analyze the possible speed-up of GE-HPGA. For brevity, let t denotes the wall clock time to evolve a single subpopulation in each generation. From equations (4) and (5), we can define:

$$t = CO_{intra} + \alpha F \tag{6}$$

The maximum speed-up offered by the framework, can then be derived as:

$$S^{max} = \frac{T_s^{max}}{T_p^{min}} = \frac{nGt^{max}}{G\left(nO_{inter}^{min} + t^{min}\right)} = \frac{nt^{max}}{nO_{inter}^{min} + t^{min}} \tag{7}$$

where T_s^{max} : Maximum total wall clock time to execute HPGA serially.

T_p^{min} : Minimum total wall clock time to execute GE-HPGA in a parallel mode.

t^{max} : Maximum wall clock time incurred by the slowest cluster to evolve a single subpopulation in each generation.

t^{min} : Minimum wall clock time incurred by the fastest cluster to evolve a single subpopulation in each generation

O_{inter}^{min} : Minimum inter-cluster communication overhead, incurred by the parallel GE-HPGA.

While equation (7) provides an upper bound for possible speed-up in the proposed framework, in practice one would expect a speed-up of less than S^{max}. Further, for any possible speed-up to happen, i.e. $S^{min} > 1$, inequalities (9), (10) and (11) must hold:

$$S^{min} > 1$$

$$\Rightarrow \frac{T_s^{min}}{T_p^{max}} > 1$$

$$\Rightarrow T_p^{max} < T_s^{min} \tag{8}$$

$$\Rightarrow nO_{inter}^{max} + t^{max} < nt^{min}$$

$$\Rightarrow t^{min} > O_{inter}^{max} + \frac{t^{max}}{n} \tag{9}$$

$$or \Rightarrow n > \frac{t^{max}}{t^{min} - O_{inter}^{max}} \tag{10}$$

$$or \Rightarrow O_{inter}^{max} < t^{min} - \frac{t^{max}}{n} \tag{11}$$

In summary, the GE-HPGA is able to offer any speed-up if the practical conditions on fitness function compute time t, cluster size n, and the communication overheads O_{inter} are satisfied.

4 Empirical Study

In this section, we present an empirical study on GE-HPGA for distributed and heterogeneous Grid computing environments. In particular, we are interested in how GE-HPGA fares under diverse Grid environment having different

communication protocols, cluster sizes, computing nodes, and geographically (across network border) disparate clusters. Figure 5 depicts some of the resources available in our collaborative Grid environment (Nanyang Campus Grid).

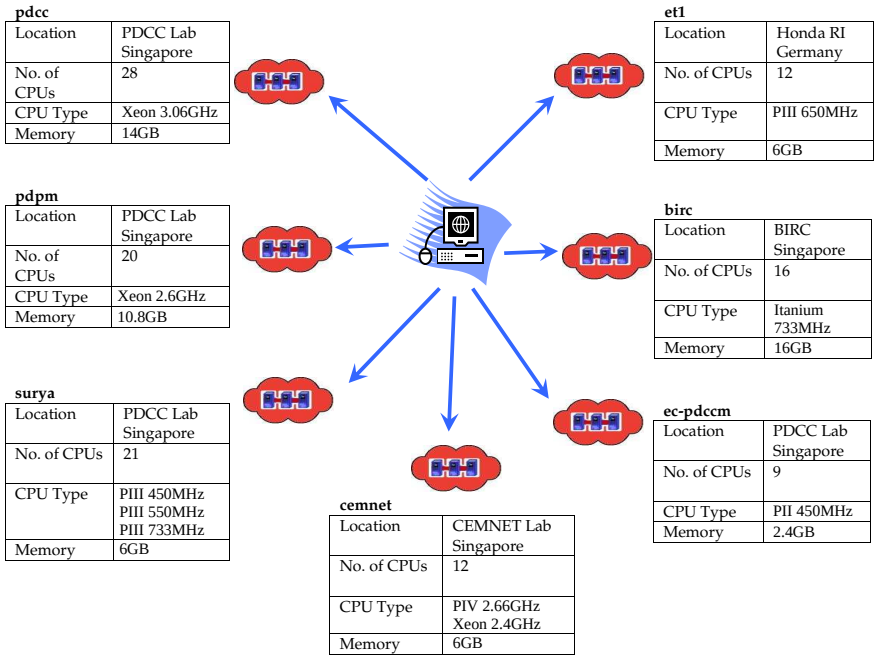


Fig. 5. Nanyang Campus Grid.

One of the clusters is physically located at Honda Research Institute (Honda RI) Europe in Germany, and the rest in Parallel and Distributed Computing Centre (PDCC), Central for Multimedia and Network Technology (CEMNET) and BioInformatic Research Centre (BIRC) situated physically at Nanyang Technological University in Singapore. They are chosen to represent the existence of heterogeneous clusters commonly found in real world settings where one resource usually differs from others in terms of processing speed, number of processors, and average intensity of workload. It is worth noting that our decision to consider clusters that are geographically distributed across continents also better emulates a realistic Grid environment where resources of design teams are often geographically disparate.

4.1 Computationally Cheap Optimization Problem

First, we consider the GE-HPGA for solving a computationally cheap optimization problem using the multi-modal Rastrigin benchmark function defined as:

$$f(x) = 20 + \sum_{i=1}^{j} x_i^2 - 10 \sum_{i=1}^{j} \cos(2\pi x_i), \tag{12}$$

where j denotes the problem dimension. A 10-dimensional Rastrigin function is considered here in the present study. The configurations of the GE-HPGA are given as follows: The subpopulation size and search termination criterion are configured as C=80 chromosomes and G=100 maximum search generations, respectively. A uniform mutation and single-point crossover operators at probabilities of P_m=0.1 and P_c=0.9, respectively, are employed while the migration interval is configured at I=5. A linear ranking algorithm is used for selection and elitism is also enforced. The performances of GE-HPGA for solving the Rastrigin problem are experimented for the following factors using the five clusters detailed in Table 1:

- high and low security Grid environment,
- varying number of subpopulations,
- a single cluster Grid environment, and
- multi-cluster Grid environment.

The computational efforts incurred by the serial HPGA (S-HPGA) and GE-HPGA for optimizing the Rastrigin problem under diverse grid environments are reported in Figures 6 and 7. The numerical results are reported for averaged of 10 independent runs. The GE-HPGA is observed in Figure 6 to be computationally more efficient than the serial HPGA (S-HPGA) counterpart with either the low or high security communication protocol. S-HPGA refers to an HPGA where all the subpopulations are executed sequentially in a cluster. Next, we discuss the performance of the GE-HPGA for single cluster and n multi-clusters Grid environments by referring to Figure 7. Interestingly, the results obtained indicate that the multi-clusters GE-HPGA do not offer any benefits in search efficiency over the single cluster GE-HPGA. This is the effect of the high communication protocol overheads and low computational cost of the objective/fitness function which results in the loss of any advantages in using multiple clusters in the GE-HPGA, which agrees with our theoretical analysis in Section 3.3. This implies that it is not always beneficial to consider using more clusters in an GE-HPGA.

Cluster	Number of CPU(s)	CPU Type	Memory
pdcc	28	Xeon 2.8GHz	14GB
pdpm	20	Xeon 2.6GHz	10.8GB
surya	21	PIII 450MHz PIII 550MHz PIII 733MHz	6GB
et1	12	PIII 650MHz	6GB
birc	16	Itanium 733MHz	16GB

Table 1
Summary of the Grid environment considered for optimizing the Rastrigin problem.

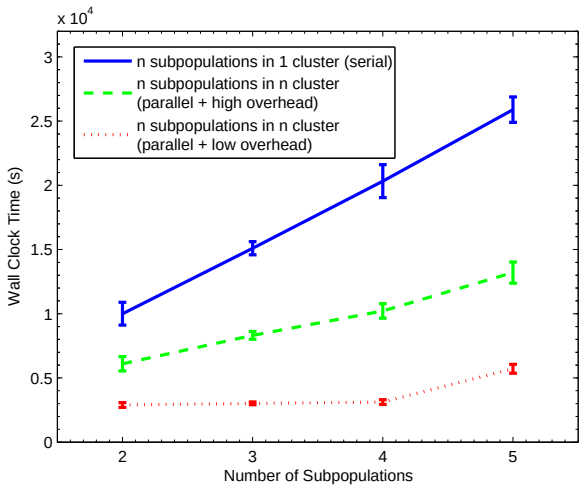


Fig. 6. Average wall clock time of S-HPGA and GE-HPGA in single and multiple clusters on Rastrigin function. *Note: High overhead implies the use of secure communication protocol and vice versa.*

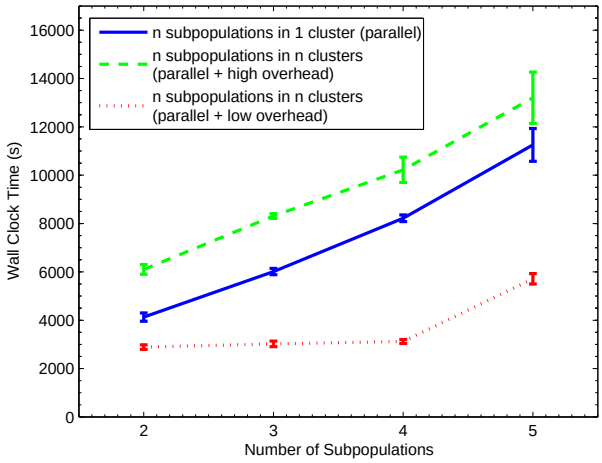


Fig. 7. Average wall clock time of GE-HPGA in single and multiple clusters on Rastrigin function. *Note: High overhead implies the use of secure communication protocol and vice versa.*

4.2 Real World Problem: Aerodynamic Airfoil Design

In this section, we extend our study of the GE-HPGA for complex real world engineering problem, specifically the efficient evolutionary design of aerodynamic airfoil shapes. In particular, we consider the parametric design optimization of 2D airfoil structure using a subsonic inverse pressure design problem. A typical approach to inverse pressure design is to 'smoothen' the upper-surface pressure curve in a way that maintains the area under the curve, so as to maintain the lift force generated by the airfoil. The target pressure profile is

generated from the NACA 0012 airfoil, which itself is also the baseline shape. The airfoil geometry is characterized using 24 design variables with the NACA 0012 airfoils as the baseline, as shown in Figure 8. The free-stream conditions in this problem are subsonic speed of Mach 0.5, and 2.0 angle of attack (AOA), corresponding to symmetric pressure profiles on the upper and lower walls. To proceed, we consider the inverse design problem, which consists in minimizing the difference between the surface pressure P of a given airfoil with the desired pressure profile P_d of the NACA 0012 airfoil. In aerodynamic shape optimization problems, if w is the flow variables and S the shape design variables, the inverse pressure design problem can be formulated as a minimization problem of the form:

$$I(w, S) = \frac{1}{2} \int_{wall} (P - P_d)^2 \, d\sigma \tag{13}$$

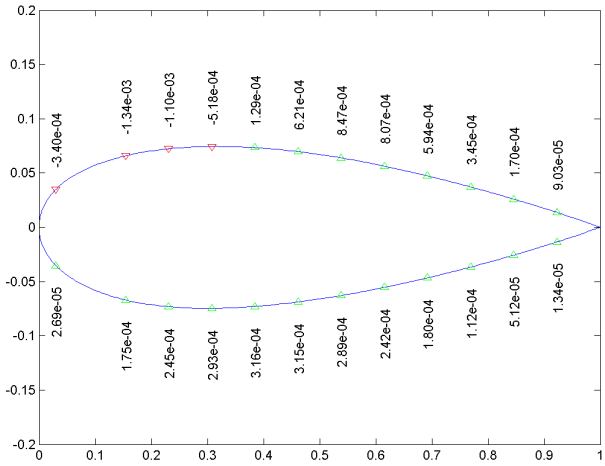


Fig. 8. Airfoil geometry characterized using 24 design variables with the NACA 0012 as baseline.

Using 24 design variables and the fitness function defined in equation (13), the GE-HPGA is used for optimizing a subsonic inverse pressure design problem of moderate fidelity [6]. In the experimental study, we consider the following configurations: single-point crossover probability P_c=0.9, uniform mutation probability P_m=0.1, subpopulation size C=50, maximum generation count G=100, and migration interval I=5. Unless otherwise specified, all results are taken from 10 independent runs.

[6] The moderate-fidelity model is obtained by reducing the accuracy of the exact subsonic inverse pressure airfoil model which takes lesser computational efforts to compute than the original.

Here we consider a Grid environment consisting of the four clusters summarized in Table 2.

Cluster	Number of CPU(s)	CPU Type	Memory
pdpm	8	8 × Xeon 2.6GHz	4GB
cemnet	5	4 × Xeon 2.4GHz 1 × PIV 2.66GHz	2GB
surya	7	7 × PIII 733MHz	3.2GB
$ec - pdccm$	8	8 × PIII 650MHz	2.2GB

Table 2
Summary of the Grid environment considered for optimizing the airfoil design problem.

The average wall-clock time of each computing cluster for evaluating a subpopulation of 50 GA chromosomes in parallel are given in Table 3. Note the significant difference in the computational efforts required by these heterogeneous clusters. The search performances of the n-cluster GE-HPGAs, i.e., 2, 3, or 4 subpopulations GE-HPGA, for optimizing the inverse pressure design problem is then reported in Figure 9. To obtain a more conservative result, whenever the single cluster setting is considered, the fastest cluster will be used. Due to the heterogeneity of the Grid environment considered, the slowest *surya* cluster has become the bottleneck of the GE-HPGA since it uses a synchronous migration model which waits for all the 'chromosome evaluation' and 'subpopulation evolution' services in a GE-HPGA generation to complete before the migration operation and subsequent search generations may proceed. This results in the poorer search efficiency of the multi-cluster environment, especially when more clusters are used, i.e. in the 4-cluster GE-HPGA.

The results obtained can be easily explained as follows. Consider the 4-subpopulation GE-HPGA run on 4 clusters, it can be estimated from Table 3 that the required computational effort is significantly higher than in a single cluster GE-HPGA, i.e., $852.34 > 4 \times 164.13 = 656.52$, hence subsequently violating equation (8).

Cluster	Average wall clock time (of 10 independent runs) for evaluating 50 individuals
pdpm	164.13 s
cemnet	366.48 s
surya	465.79 s
$ec - pdccm$	852.34 s

Table 3
Computational efforts required to evaluate a subpopulation of 50 designs using the moderate-fidelity airfoil analysis code (inclusive of the communication overhead incurred).

To compromise with the heterogeneity of the computing resources, a simple solution to maintain the benefit of parallelization is to configure the subpopulation size according to the computational capabilities of the clusters. Nevertheless, such an approach has the disadvantage of possibly altering the

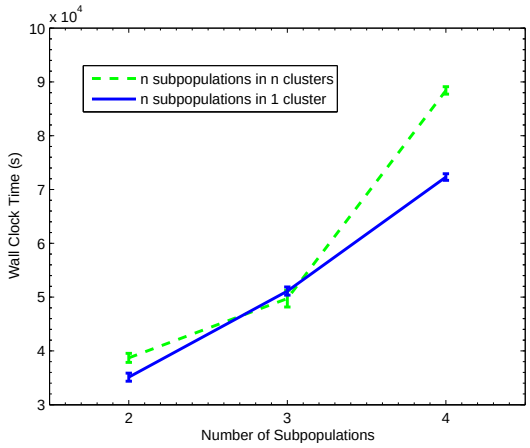


Fig. 9. Average wall clock time of GE-HPGA in single and multiple clusters on the airfoil design problem.

standard behavior of a synchronous island PGA and is also against the philosophy of Grid computing. Since the search behavior could be unpredictable in such approach, it is advisable to be more conservative by maintaining uniform subpopulation sizes. A preferable solution should still provide the speed-up regardless of the heterogeneity in the Grid environment while preserving the standard behavior of the parallel genetic search. To achieve such a solution, we present a Decoupled GE-HPGA (DGE-HPGA) as depicted in Figure 10.

The core idea lies in the decoupling of the 'subpopulation evolution' Grid service in GE-HPGA (see Figure 2) into two separate services, namely, the 'evolutionary operations' at the GE-HPA client(master) and 'chromosome ensemble' at each computing cluster. In this way, the computing clusters are solely meant for fitness evaluation purpose and all evolutionary operations proceed at the client side. This is slightly different from the original GE-HPGA where both evolutionary operations and fitness evaluations are coupled in the 'subpopulation evolution' service at each computing cluster. By doing so, the uniformity of the subpopulation size can be maintained at the client side while the computing clusters are actually allocated non-uniform ensembles of chromosome for fitness evaluations.

To minimize the idling time of processing nodes in fast clusters while waiting for synchronization in the DGE-HPGA, we hope to obtain a well-balanced execution time in each cluster and for each HPGA search generation. In each n-subpopulation DGE-HPGA search generation, each non-uniform ensemble of individuals, C_i, assigned to the i-th cluster for evaluations is estimated by:

$$C_i = \frac{t_i^{-1}}{\sum_{j=1}^{n} t_j^{-1}} \times n \times C \tag{14}$$

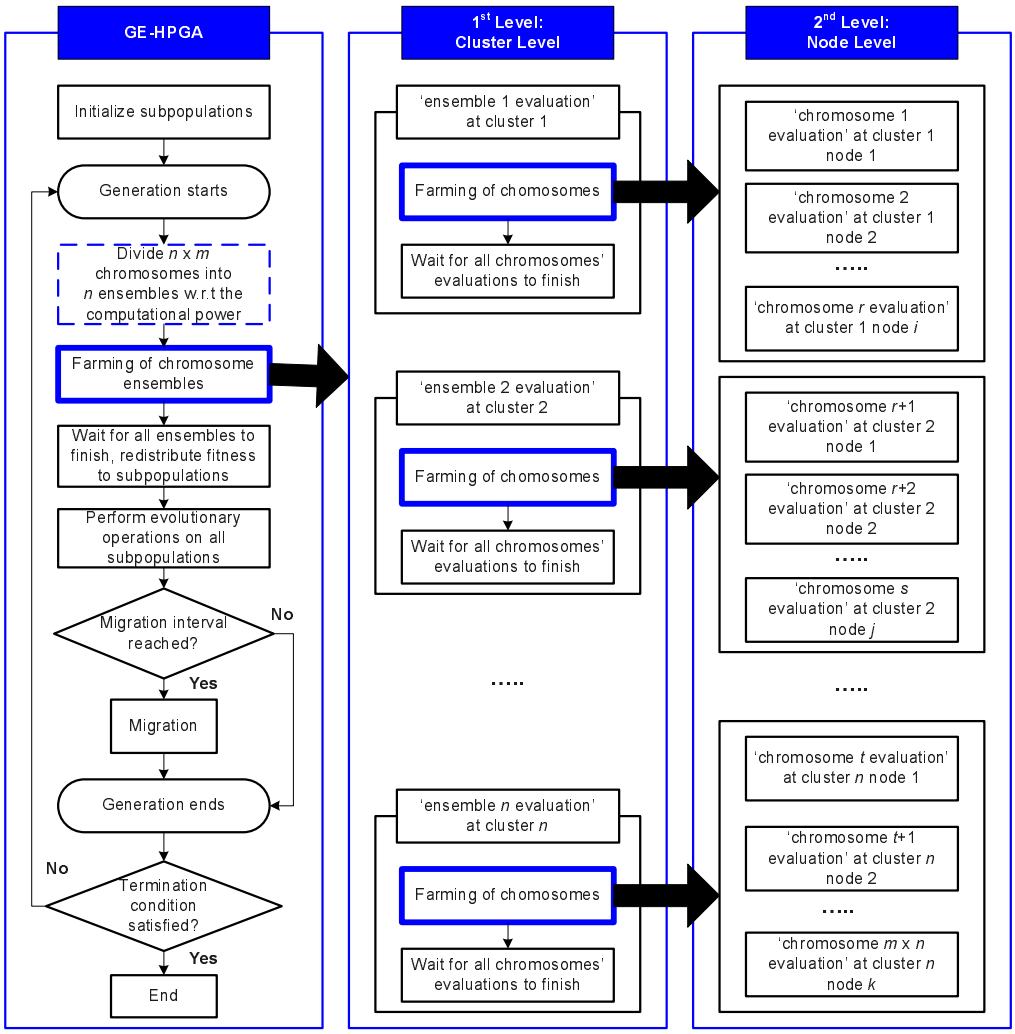


Fig. 10. The Decoupled GE-HPGA.

where C is the subpopulation size, n is the number of subpopulations(or clusters), and t_i is the time required by the i-th cluster to evolve a subpopulation in one generation of GE-HPGA.

The average wall-clock time and optimum shape obtained when using GE-HPGA and DGE-HPGA to search on the subsonic inverse pressure design problem for a maximum of 100 generations are reported in Figures 11 and 12, respectively. Note that the n-subpopulation GE-HPGA has a uniform subpopulation size of 50. In contrast, the size of the ensembles of individuals are defined using equation (14) in the n-subpopulation DGE-HPGA, i.e., 2 subpopulations ($pdpm : cemnet = 69 : 31$), 3 subpopulations ($pdpm : cemnet : surya = 83 : 37 : 30$), and 4 subpopulations ($pdpm : cemnet : surya : ec - pdccm = 100 : 45 : 35 : 20$).

More importantly, the results in Figure 11 and 12 show that the DGE-HPGA converges to the same optimal design airfoil shape as the GE-HPGA at sig-

nificantly lesser amount of wall-clock time spent.

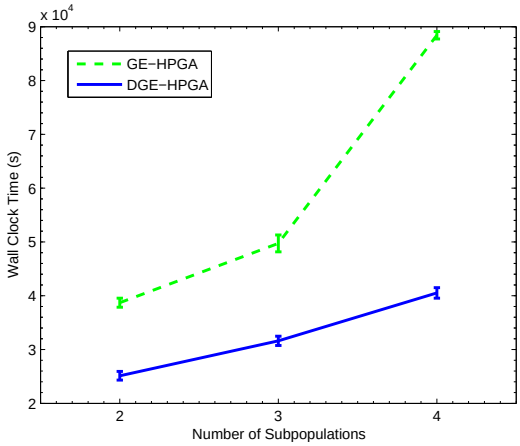


Fig. 11. Average wall clock time of GE-HPGA and DGE-HPGA in 2, 3, and 4-subpopulation runs in optimizing the aerodynamic airfoil design problem.

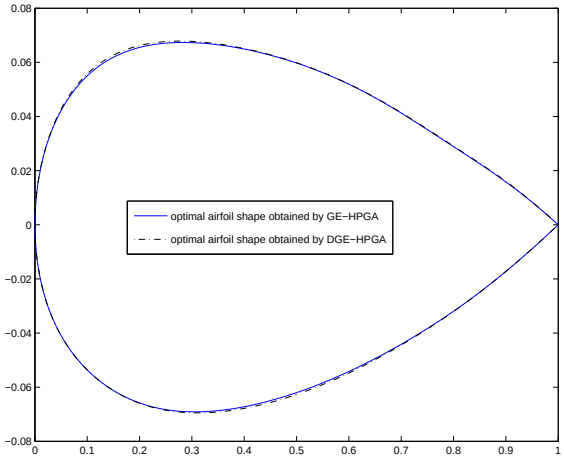


Fig. 12. Optimal airfoil shape obtained after 100 generations of the 4-subpopulation GE-HPGA and DGE-HPGA.

5 Conclusions

In this paper, we have proposed and presented a Grid-Enabled Hierarchical Parallel Genetic Algorithm framework (GE-HPGA) based on standard Grid technologies. The framework offers a comprehensive solution for efficient parallel evolutionary design of problems with computationally expensive fitness functions, by providing novel features that conceals the complexity of a Grid environment through an extended GridRPC API and a metascheduler for automatic resource discovery. To assess the effectiveness of the framework,

theoretical analysis on maximum speed-up of GE-HPGA and the practical conditions that must be fulfilled for any speed-up has been reported. Empirical results using a benchmark problem and a realistic airfoil design problem further confirm that speed-up can be attained as long as the bounds on fitness function cost, cluster size, and communication overheads of the Grid environment are satisfied.

6 Acknowledgement

D. Lim and Y.S. Ong would like to thank Honda Research Institute Europe for sponsoring the research on the topic, and members of Nanyang Technological University and Honda Research Institute Europe for providing the support. Without them, this project would not have been a successful one.

References

[1] Y.S. Ong, P.B. Nair, and A.J. Keane, "Evolutionary Optimization of Computationally Expensive Problem via Surrogate Modeling," *American Institute of Aeronautics and Astronautics Journal*, Vol. 41, No. 4, pp. 687-696, 2003.

[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] H.T. Kim, B.Y. Kim, E.H. Park, J.W. Kim, E.W. Hwang, S.K. Han, and S. Cho, "Computerized recognition of Alzheimer disease-EEG using genetic algorithms and neural network," *Future Generation Computer Systems*, Volume 21, Issue 7, pp. 1124-1130, 2005.

[4] Y. Gao, H. Rong, and J.Z. Huang, "Adaptive grid job scheduling with genetic algorithms," *Future Generation Computer Systems*, Volume 21, Issue 1, pp. 151-161, 2005.

[5] M. Li, B. Yu, and M. Qi, "PGGA: A predictable and grouped genetic algorithm for job scheduling," *Future Generation Computer Systems*, Volume 22, Issue 5, pp. 588-599, 2006.

[6] D.E. Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning," Addison-Wesley, Reading, Massachusetts, 1989.

[7] E. Alba, A.J. Nebro, J.M. Troya, "Heterogeneous Computing and Parallel Genetic Algorithms," *Journal of Parallel and Distributed Computing* 62, pp. 1362-1385, 2002.

[8] E. Alba, J.M. Troya, "Synchronous and Asynchronous Parallel Distributed Genetic Algorithms," *Future Generation Computer Systems*, 17(4):451-465, January 2001.

[9] M. Nowostawski, R. Poli, "Parallel Genetic Algorithm Taxonomy," *Proceedings of the Third International conference on knowledge-based intelligent information engineering systems* (KES'99), pp. 88-92, Adelaide, 1999.

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

[11] H.Y. Foo, J. Song, W. Zhuang, H. Esbensen, E.S. Kuh, "Implementation of a Parallel Genetic Algorithm for Floorplan Optimization on IBM SP2," hpcasia, p. 456, *HPC on the Information Superhighway*, HPC-Asia, 1997.

[12] F.J. Villegas, T. Cwik, Y. Rahmat-Samii, and M. Manteghi, "A parallel electromagnetic Generic-Algorithm Optimization (EGO) application for patch antenna design," *IEEE Transactions on Antennas and Propagation*, Vol. 52, No. 9, pp. 2424-2435, 2004.

[13] D. Abramson, J. Abela, "A Parallel Genetic Algorithm for Solving the School Timetabling Problem," *Technical Report*, Division of Information Technology, C.S.I.R.O, Melbourne, 1991.

[14] Ansys Inc., [online] http://www.ansys.com.

[15] CFD Flow Modeling Software and Services from Fluent Inc., [online] http://www.fluent.com.

[16] Sysnoise, [online] http://ludit.kuleuven.be/software/packages/sysnoise.html.

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

[18] M. Baker, R. Buyya, D. Laforenza, "The Grid: International Efforts in Global Computing," *International Conference on Advances in Infrastructures for Electronic Business, Science, and Education on the Internet*, 2000.

[19] G. Xue, W. Song, S.J. Cox, A.J. Keane, "Numerical optimisation as Grid Services for Engineering Design," *Journal of Grid Computing*, Vol.2, No.3, pp. 223-238, 2004.

[20] Q.T. Ho, W.T. Cai, and Y.S. Ong, "Design and Implementation of An Efficient Multi-cluster GridRPC System," *IEEE Cluster Computing and Grid Conference*, pp. 358-365, vol. 1, 9-12 May 2005.

[21] Special section: Complex problem-solving environments for grid computing, *Future Generation Computer Systems*, Vol. 21, Issue 6, Elsevier Science Publisher, Amsterdam, 2005.

[22] D. Abramson, R. Buyya, J. Giddy, "A computational economy for grid computing and its implementation in the Nimrod-G resource broker", *Future Generation Computer Systems*, 18(8): 1061-1074, 2002.

[23] A. Iosup and D.H.J. Epema, "GrenchMark: A Framework for Analyzing, Testing, and Comparing Grids", *6th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid'06)*, pp. 313-320, IEEE Press, 2006.

[24] Z. Liu, A. Liu, C. Wang, Z. Niu, "Evolving neural network using real coded genetic algorithm(GA) for multispectral image classification," *Future Generation Computer Systems*, Vol. 20, Issue 7, pp. 1119-1129, 2004.

[25] J. Cui, T.C. Fogarty, and J.G. Gammack, "Searching databases using parallel genetic algorithms on a transputer computing surface," *Future Generation Computer Systems*, 9(1):33-40, 1993.

[26] G.A. Sena, D. Megherbi, G. Isern, "Implementation of a parallel genetic algorithm on a cluster of workstations: travelling salesman problem, a case study," *Future Generation Computer Systems*, Vol. 17, Issue 4, pp. 477-488, 2001.

[27] I. Foster, "The Globus Toolkit for Grid Computing," *Proceedings of the 1st International Symposium on Cluster Computing and the Grid*, 2001.

[28] CoG Kit Wiki, [online] http://www.cogkit.org.

[29] M. Massie, B. Chun, and D. Culler, "The Ganglia Distributed Monitoring System: Design, Implementation, and Experience," Technical report, University of California, Berkeley, 2003.

[30] S. Agrawal, J. Dongarra, K. Seymour, S. Vadhiyar, "NetSolve: past, present, and future; a look at a Grid enabled server," 2002.

[31] Globus: Information Services/MDS, [online] http://www-unix.globus.org/toolkit/mds.

[32] H. Nakada, S. Matsuoka, K. Seymour, J. Dongarra, C. Lee, H. Casanova, "GridRPC: A remote procedure call API for Grid computing," *Grid Computing - Grid 2002*, LNCS 2536, pp. 274-278, 2002.

[33] S. Tuecke, "Grid Security Infrastructure (GSI) Roadmap," Internet Draft Document: draft-Gridforum-gsi-roadmap-02.txt, 2001.

[34] The Globus Project, "GridFTP Universal Data Transfer for the Grid," The Globus Project White Paper, 2000.

[35] D. Geer, "Grid Computing Using the Sun Grid Engine," Technical Enterprises, Inc., 2003.

[36] J. Frey, T. Tannenbaum, M. Livny, I. Foster, S. Tuecke, "Condor-G: A Computation Management Agent for Multi-Institutional Grids," *Proceedings of the Tenth IEEE Symposium on High Performance Distributed Computing (HPDC10)*, 2001.

[37] M.J. Quinn, "Parallel Programming in C with MPI and OpenMP," McGraw Hill, 2004.

[38] G. Ahmdal, "Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities," *AFIPS Conference Proceedings*, Vol. 30, pp. 483-485, Thompson Books, Washington D.C., 1967.