Available online at www.sciencedirect.com
ScienceDirect
Future Generation Computer Systems 23 (2007) 658–670
FGCS
FUTURE GENERATION COMPUTER SYSTEMS
www.elsevier.com/locate/fgcs

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

Received 2 August 2006; received in revised form 3 October 2006; accepted 15 October 2006
Available online 28 November 2006

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 the Grid environment, and (2) a metascheduler for seamless resource discovery and selection. To assess the practicality of the framework, a theoretical analysis of the possible speed-up offered is presented. An 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 a significant speed-up to evolutionary design optimization in science and engineering.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Grid computing; Parallel Genetic Algorithms

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 [1–5]. 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 their 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 GAs. 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 GAs is their 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 have been created for exploiting the explicit parallelism of GAs. The reader is referred to [9,10] for some excellent expositions of parallel GAs.

While numerous researches 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 [11–13]. Their focus has been on small-scale dedicated computing resources, and they 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 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 site licenses for commercial analysis packages, for example,

* Corresponding address: Nanyang Technological University, Emerging Research Laboratory, School of Computer Engineering, Block N4, 2b-39, Nanyang Avenue, Singapore 639798, Singapore.
E-mail addresses: dlim@ntu.edu.sg (D. Lim), asysong@ntu.edu.sg (Y.-S. Ong), yaochu.jin@honda-ri.de (Y. Jin), bernhard.sendhoff@honda-ri.de (B. Sendhoff), ebslee@ntu.edu.sg (B.-S. Lee).

0167-739X/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2006.10.008

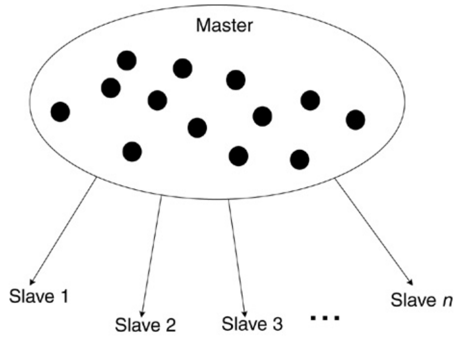
ANSYS [14], FLUENT [15], SYSNOISE [16], etc. It is for these reasons that the recent advent of what is termed Grid computing [17,18] has gained widespread attention, as it sets about the notion of establishing a set of open standards for distributed resources and ubiquitous services. Advances in Grid computing have also fueled the research and development of a Grid problem solving environment for complex design in science and engineering [19–22]. While many open issues remain when using Grid computing across geographically disparate laboratories, scheduling, resource management [23], and benchmarking Grids [24] 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 a 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 Grid-enabled parallel genetic searching can generate significant speed-ups for evolutionary design optimization.

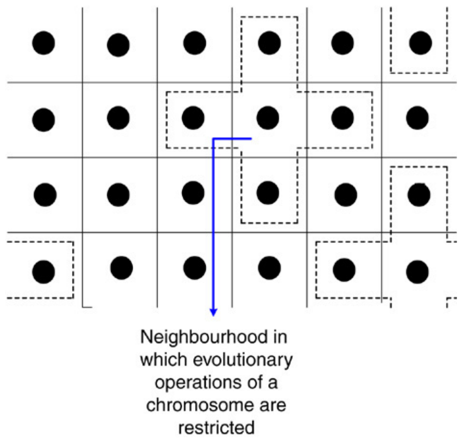
The rest of this paper is organized as follows. In Section 2, we present a brief overview on parallel GAs, as one of the successful design optimization methodologies used in science and engineering. Section 3 describes the proposed Grid-Enabled Hierarchical Parallel Genetic Algorithm (GE-HPGA) optimization framework. A theoretical analysis of 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.

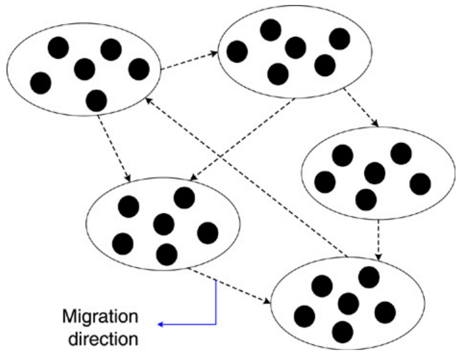
2. Parallel Genetic Algorithms

Genetic Algorithms [6] are probabilistic meta-heuristic methods inspired by the ‘survival of the fittest’ principle of the neo-Darwinian theory of evolution. Artificial creatures are created and put into competition in a struggle for life and only the survivors are able to reproduce. A new population is thus created using biologically inspired operators such as crossover, mutation, and selection, and the process repeats until some search termination criteria are reached. A GA without any structure is usually referred to as a panmictic GA. Parallel Genetic Algorithms (PGAs) are extensions of panmictic GAs. 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 to complex design problems [25–27]. In addition to the parallel panmictic GA, two popular parallel structured GAs include the island and cellular GAs [9,10]. In this section, we present a brief review of these algorithms. The three basic models of PGA are as illustrated in Fig. 1.



(a) Single population master–slave PGA.


Neighbourhood in which evolutionary operations of a chromosome are restricted
(b) Cellular PGA.



(c) Island PGA.

Fig. 1. Basic models of Parallel Genetic Algorithms.

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 canonical GA, evaluations of individuals are distributed by scheduling fractions of the population among the processing slave nodes. Such a model has the advantage of 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 a closely-linked massively parallel processing system, i.e., a computing system consisting of 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 individuals, so that good solutions may be disseminated across the entire populations. Sometimes, the fine-grained parallel GA is also termed as the cellular model PGA.

Multi-population or multi-deme or island PGA. A 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 the upper level) and a fine-grained PGA or master–slave PGA (which we consider throughout the development of the framework in this paper), or even another level of island PGAs (at lower levels). 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. A theoretical analysis of 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. The Globus Toolkit[1] [28], The Commodity Grid Kit (CogKit)[2] [29], Ganglia monitoring tool[3] [30], and NetSolve[4] [31] 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 for the selection of Grid resources. As a result, users are naturally required to perform manual look-ups and select resources when assigning new tasks onto the Grid's computing resources. This is clearly impractical, since a 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 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) [32] and Ganglia [30] 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 [33,21], based on the Commodity Grid Kit (CoGKit) for 'gridifying' new or existing analysis/simulation codes or objective/fitness functions as Grid services. This choice is down to its simplicity of implementation 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 [21] for further exposition. Other Grid technologies utilized in the present work include the Globus Grid Security Infrastructure (GSI) [34] for secure and authenticated access over the Grid. For data flow, the Globus Grid File Transfer Protocol (GridFTP) [35] 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 consider a two-level HPGA in the present study, i.e., Level 1: Island model PGA and Level

[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 deployed in 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 the servers registered to an agent.

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

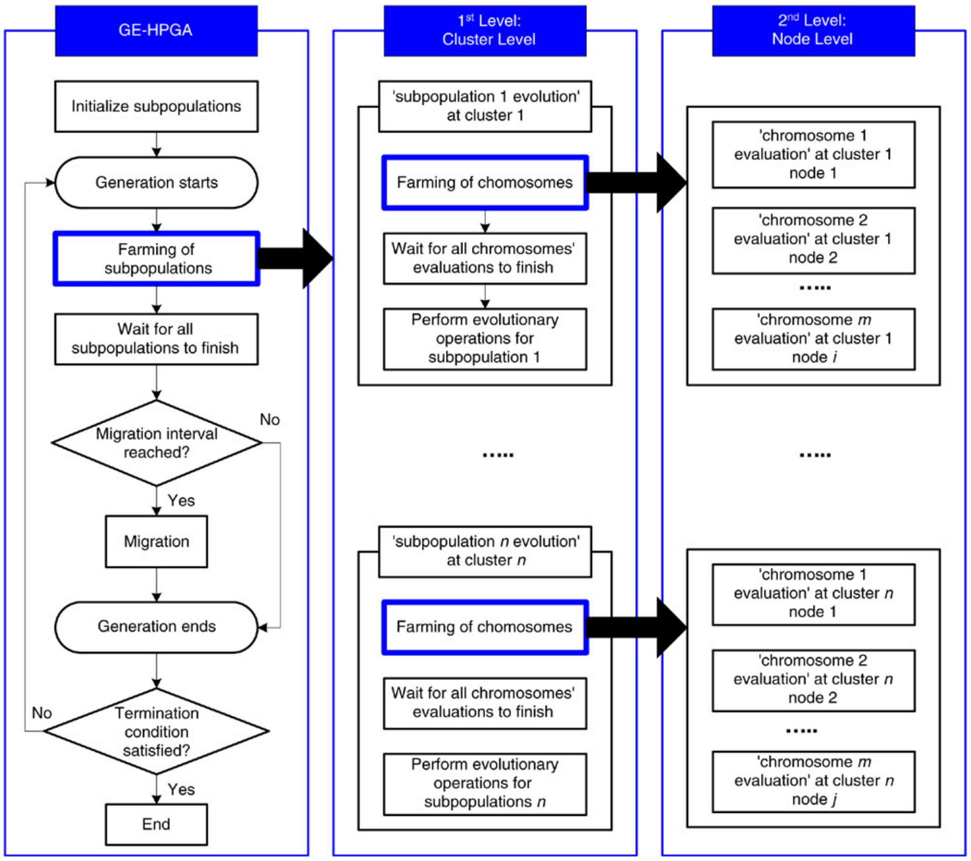

GE-HPGA
Initialize subpopulations
Generation starts
Farming of subpopulations
Wait for all subpopulations to finish
Migration interval reached?
No
Yes
Migration
Generation ends
Termination condition satisfied?
No
Yes
End
1st Level: Cluster Level
'subpopulation 1 evolution' at cluster 1
Farming of chomosomes
Wait for all chromosomes' evaluations to finish
Perform evolutionary operations for subpopulation 1
'subpopulation n evolution' at cluster n
Farming of chomosomes
Wait for all chromosomes' evaluations to finish
Perform evolutionary operations for subpopulations n
2nd Level: Node Level
'chromosome 1 evaluation' at cluster 1 node 1
'chromosome 2 evaluation' at cluster 1 node 2
'chromosome m evaluation' at cluster 1 node i
'chromosome 1 evaluation' at cluster n node 1
'chromosome 2 evaluation' at cluster n node 2
'chromosome m evaluation' at cluster n node j

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 evaluation.

2: Master–Slave PGA. The two levels of the hierarchical PGA are ‘gridified’ to form the ‘subpopulation evolution’ and ‘chromosome evaluation’ Grid services respectively, as depicted in Fig. 2. In each cluster, a ‘subpopulation evolution’ Grid service is put in place for remote invocation using the Globus job submission protocol. The ‘chromosome evaluation’ invocation, on the other hand, may be realized using any local cluster scheduler, for example NetSolve [31], the Sun Grid Engine [36], Condor [37], or others. The detailed workflow of the GE-HPGA can be outlined as nine crucial stages, and is depicted in Fig. 3.

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 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) The Grid computing resources and service information maintained by the Globus MDS and Ganglia are then provided to the GE-HPGA master program to proceed with its 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 the 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 subpopulations of ‘chromosome evaluation’ service requests that nest within the ‘subpopulation evaluation’ service are subsequently farmed across the field of processing nodes that are 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 the chromosomes is marshaled back to the ‘subpopulation evolution’ service

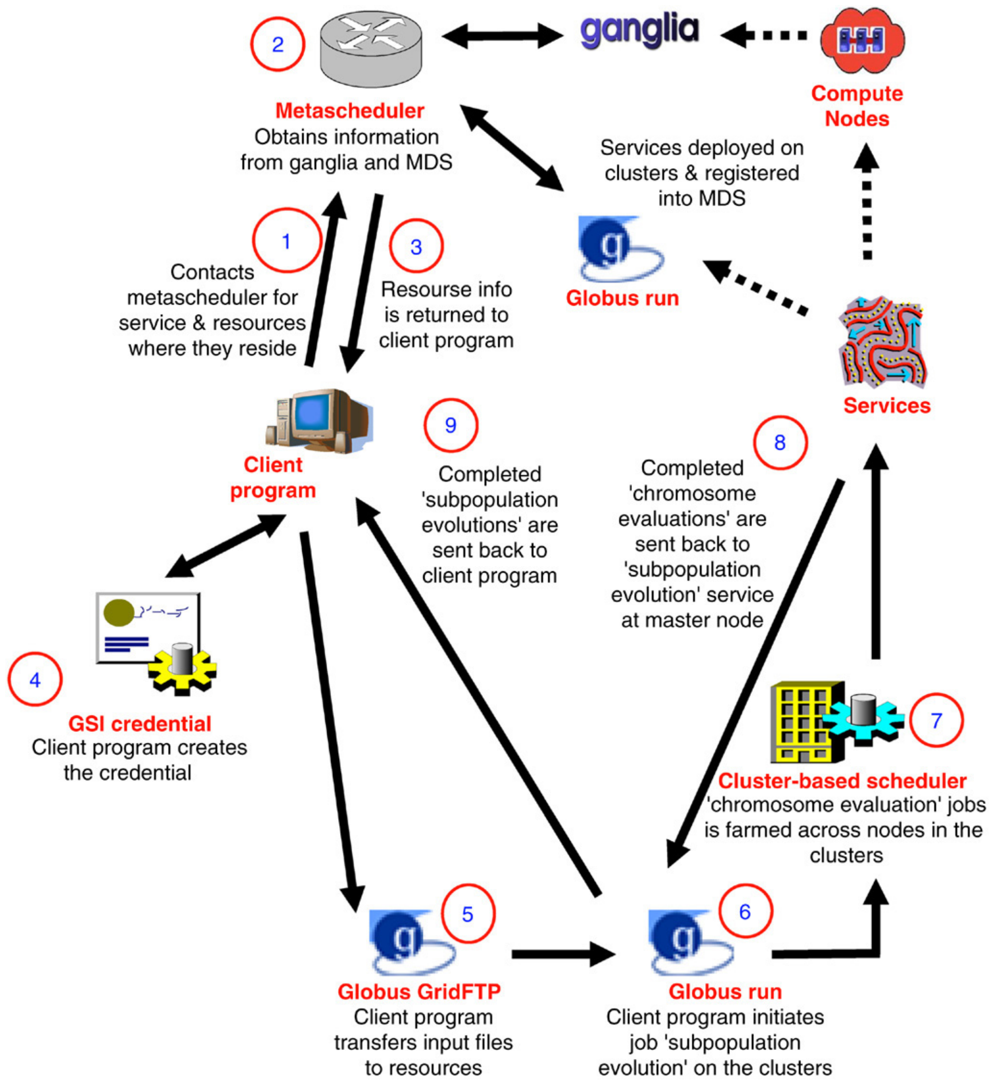

2
Metascheduler
Obtains information from ganglia and MDS
ganglia
Compute Nodes
Services deployed on clusters & registered into MDS
1
Contacts metascheduler for service & resources where they reside
3
Resourse info is returned to client program
Globus run
Services
Client program
9
Completed 'subpopulation evolutions' are sent back to client program
8
Completed 'chromosome evaluations' are sent back to 'subpopulation evolution' service at master node
4
GSI credential
Client program creates the credential
7
Cluster-based scheduler
'chromosome evaluation' jobs is farmed across nodes in the clusters
5
Globus GridFTP
Client program transfers input files to resources
6
Globus run
Client program initiates job 'subpopulation evolution' on the clusters

Fig. 3. Workflow of the GE-HPGA framework.

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 of GE-HPGA*

One of the major performance issues when running a parallel algorithm is how much speed-up it can offer as compared to a sequential run of the same algorithm [38]. This speed-up (S) measurement can be defined by:

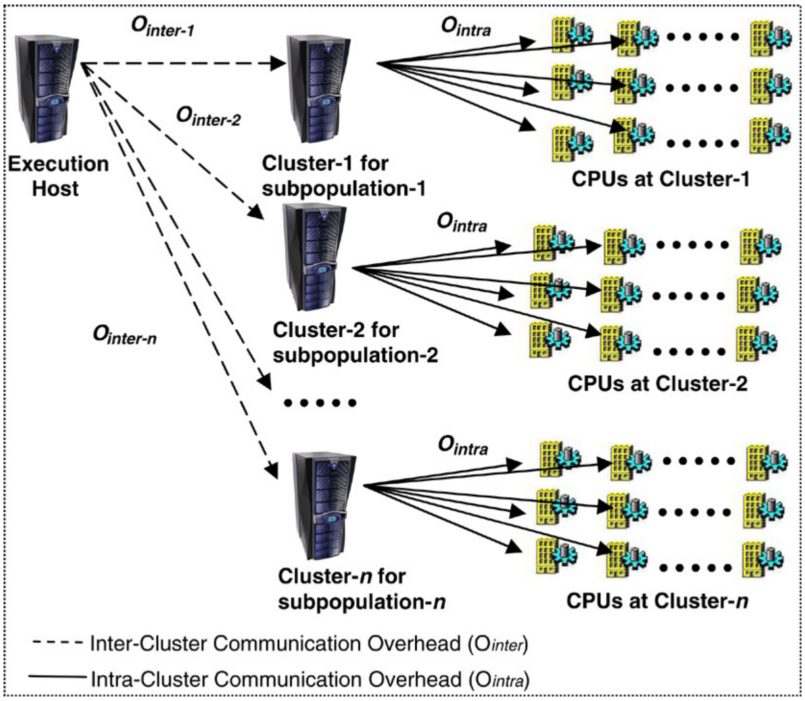
$$S = \frac{T_s}{T_p} \tag{1}$$

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 overhead. For a problem of size n, and number of processors p, it is possible to derive from Eq. (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 the computation is divided equally among the processors. The further simplification of the upper bound of maximum speedup that may be obtained by using *Amdahl's Law* [39], as shown in Eq. (2), becomes:

$$\begin{aligned} S(n,p) &= \frac{\lambda(n)+\gamma(n)}{\lambda(n)+\frac{\gamma(n)}{p}+O(n,p)} \le S^{\max}(n,p) \\ &= \frac{\lambda(n)+\gamma(n)}{\lambda(n)+\frac{\gamma(n)}{p}}. \end{aligned} \tag{3}$$



$O_{inter-1}$
$O_{inter-2}$
$O_{inter-n}$
O_{intra}
Execution Host
Cluster-1 for subpopulation-1
Cluster-2 for subpopulation-2
Cluster-*n* for subpopulation-*n*
CPUs at Cluster-1
CPUs at Cluster-2
CPUs at Cluster-*n*
Inter-Cluster Communication Overhead (O_{inter})
Intra-Cluster Communication Overhead (O_{intra})

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

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 Eqs. (1)–(3), we analyze the theoretical execution time of the GE-HPGA. From Fig. 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.

The inter-cluster communication overhead represents the time incurred for Globus remote job execution and transferring of the 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 of a cluster using Netsolve. To study the impact of the communication overheads on search efficiency, we analyze the computational complexity of an 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-subpopulation HPGA on a single cluster Grid environment. This models the case where the evolutions of PGA subpopulations are conducted in a sequential manner. Hence there is only a single level of parallelism, consisting only of the level 2 HPGA depicted in Fig. 2, where the subpopulation of chromosomes is transferred onto the processing nodes of a single cluster for fitness evaluations. Assuming that the execution host is also used for subpopulation evaluations, the total wall clock time for such a process can be derived as:

$$T_s = nG(C\,O_{\text{intra}} + \alpha F) \tag{4}$$

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 the n subpopulations of the 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:

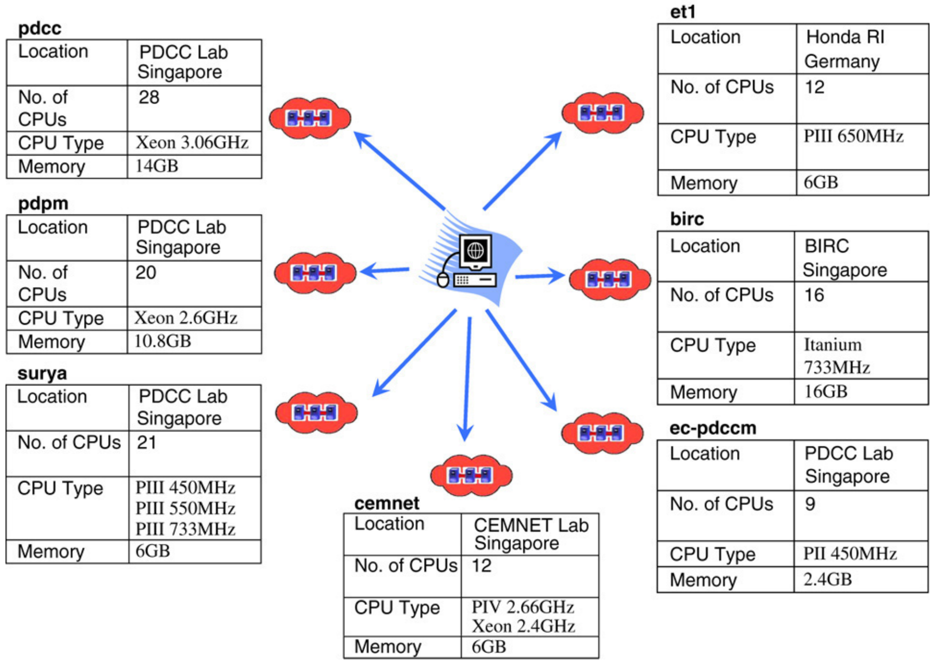
$$T_p = G\left(\left(\sum_{i=1}^{n} O_{\text{inter}}^{i}\right) + C\,O_{\text{intra}} + \alpha F\right) \tag{5}$$

where $\sum_{i=1}^{n} O_{\text{inter}}^{i}$ represents the total inter-cluster communication overhead to transfer n PGA subpopulations onto n computing clusters in the Grid environment. Without loss of generality, Eq. (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) the higher parallelism factor, α and (2) the lower inter-cluster communication overheads, O_{inter}, since the subpopulations are executed on the same cluster in parallel.

3.3.2. *Speed-up due to the GE-HPGA*

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

$$t = C\,O_{\text{intra}} + \alpha F. \tag{6}$$



pdcc

Location	PDCC Lab Singapore
No. of CPUs	28
CPU Type	Xeon 3.06GHz
Memory	14GB

pdpm

Location	PDCC Lab Singapore
No. of CPUs	20
CPU Type	Xeon 2.6GHz
Memory	10.8GB

surya

Location	PDCC Lab Singapore
No. of CPUs	21
CPU Type	PIII 450MHz PIII 550MHz PIII 733MHz
Memory	6GB

cemnet

Location	CEMNET Lab Singapore
No. of CPUs	12
CPU Type	PIV 2.66GHz Xeon 2.4GHz
Memory	6GB

et1

Location	Honda RI Germany
No. of CPUs	12
CPU Type	PIII 650MHz
Memory	6GB

birc

Location	BIRC Singapore
No. of CPUs	16
CPU Type	Itanium 733MHz
Memory	16GB

ec-pdccm

Location	PDCC Lab Singapore
No. of CPUs	9
CPU Type	PII 450MHz
Memory	2.4GB

Fig. 5. Nanyang Campus Grid.

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

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

where

$T_s^{\max}$	Maximum total wall clock time to execute the HPGA serially.
$T_p^{\min}$	Minimum total wall clock time to execute the 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.
$O_{\text{inter}}^{\min}$	Minimum inter-cluster communication overhead, incurred by the parallel GE-HPGA.

While Eq. (7) provides an upper bound for the 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)–(11) must hold:

$$S^{\min} > 1$$

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

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

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

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

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

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

In summary, the GE-HPGA is able to offer an 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 of the GE-HPGA for distributed and heterogeneous Grid computing environments. In particular, we are interested in how GE-HPGA fares under diverse Grid environments having different communication protocols, cluster sizes, computing nodes, and geographically (across network border) disparate clusters. Fig. 5 depicts some of the resources available in our collaborative Grid environment (Nanyang Campus Grid).

One of the clusters is physically located at Honda Research Institute (Honda RI) Europe in Germany, and the rest the in Parallel and Distributed Computing Centre (PDCC), Centre 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 the heterogeneous clusters commonly found in real world settings, where one resource usually differs from the 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 the resources of design teams are often geographically disparate.

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

Cluster	Number of CPU(s)	CPU type	Memory (GB)
pdcc	28	Xeon 2.8 GHz	14
pdpm	20	Xeon 2.6 GHz	10.8
surya	21	PIII 450 MHz	6
		PIII 550 MHz	
		PIII 733 MHz	
*et*1	12	PIII 650 MHz	6
birc	16	Itanium 733 MHz	16

4.1. Computationally cheap optimization problem

First, we consider use of 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 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. Uniform mutation and single-point crossover operators with 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 performance of the GE-HPGA for solving the Rastrigin problem is examined 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
- a 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 Figs. 6 and 7. The numerical results are reported for the averages of 10 independent runs. The GE-HPGA is observed in Fig. 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-cluster Grid environments by referring to Fig. 7. Interestingly, the results obtained indicate that the multi-cluster GE-HPGAs 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 a GE-HPGA.

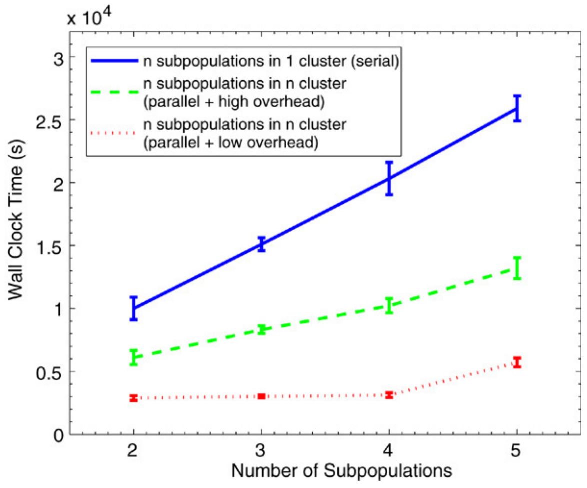


Fig. 6. Average wall clock time of S-HPGA and GE-HPGA in single and multiple clusters on the Rastrigin function. Note: High overhead implies the use of secure communication protocol and vice versa.

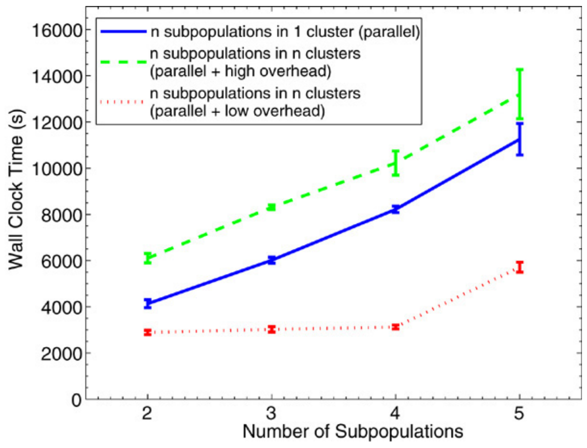


Fig. 7. Average wall clock time of GE-HPGA in single and multiple clusters on the Rastrigin function. Note: High overhead implies the use of secure communication protocol and vice versa.

4.2. Real world problem: Aerodynamic airfoil design

In this section, we extend our study of the GE-HPGA to a complex real world engineering problem, specifically the efficient evolutionary design of aerodynamic airfoil shapes. In particular, we consider the parametric design optimization of a 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 also provides the baseline shape. The airfoil's geometry is characterized using 24 design variables, with the NACA 0012 airfoil as the baseline, as shown in Fig. 8. The free-stream conditions in this problem are a subsonic speed of Mach 0.5, and a 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

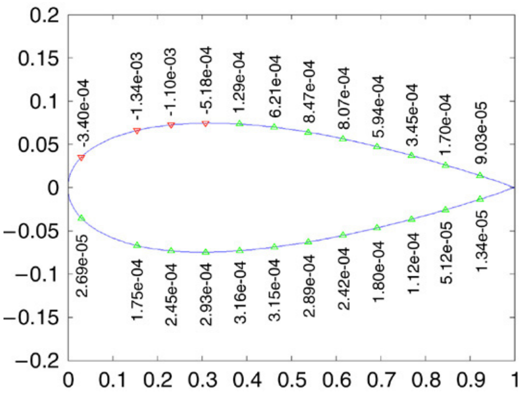


Fig. 8. Airfoil geometry characterized using 24 design variables with the NACA 0012 as baseline.

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

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

with the desired pressure profile P_d of the NACA 0012 airfoil. In aerodynamic shape optimization problems, if w is the flow variable and S the shape design variable, the inverse pressure design problem can be formulated as a minimization problem of the form:

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

Using 24 design variables and the fitness function defined in Eq. (13), the GE-HPGA is used for optimizing a subsonic inverse pressure design problem of moderate fidelity.[6] In our 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.

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

The average wall-clock times 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., the 2, 3, or 4 subpopulations GE-HPGAs, for optimizing the inverse pressure design problem is then reported in Fig. 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. as in the 4-cluster GE-HPGA.

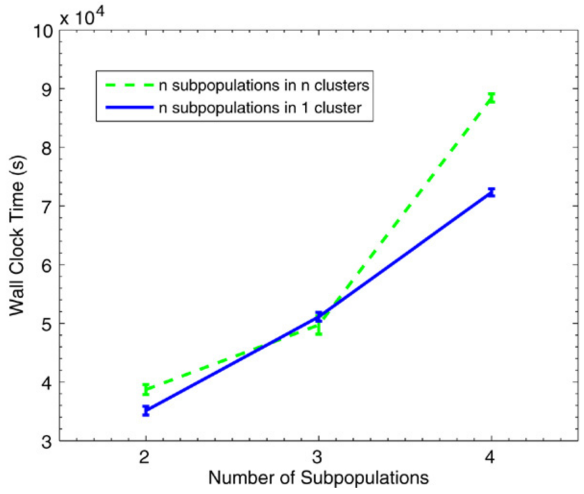


Fig. 9. Average wall clock time of GE-HPGA in single and multiple clusters on the airfoil design problem.

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)

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

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 Eq. (8).

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 standard behavior of a synchronous island PGA, and it is also against the philosophy of Grid computing. Since the search behavior could be unpredictable under such an approach, it is advisable to be more conservative by maintaining uniform subpopulation sizes. A preferable solution should still

[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.

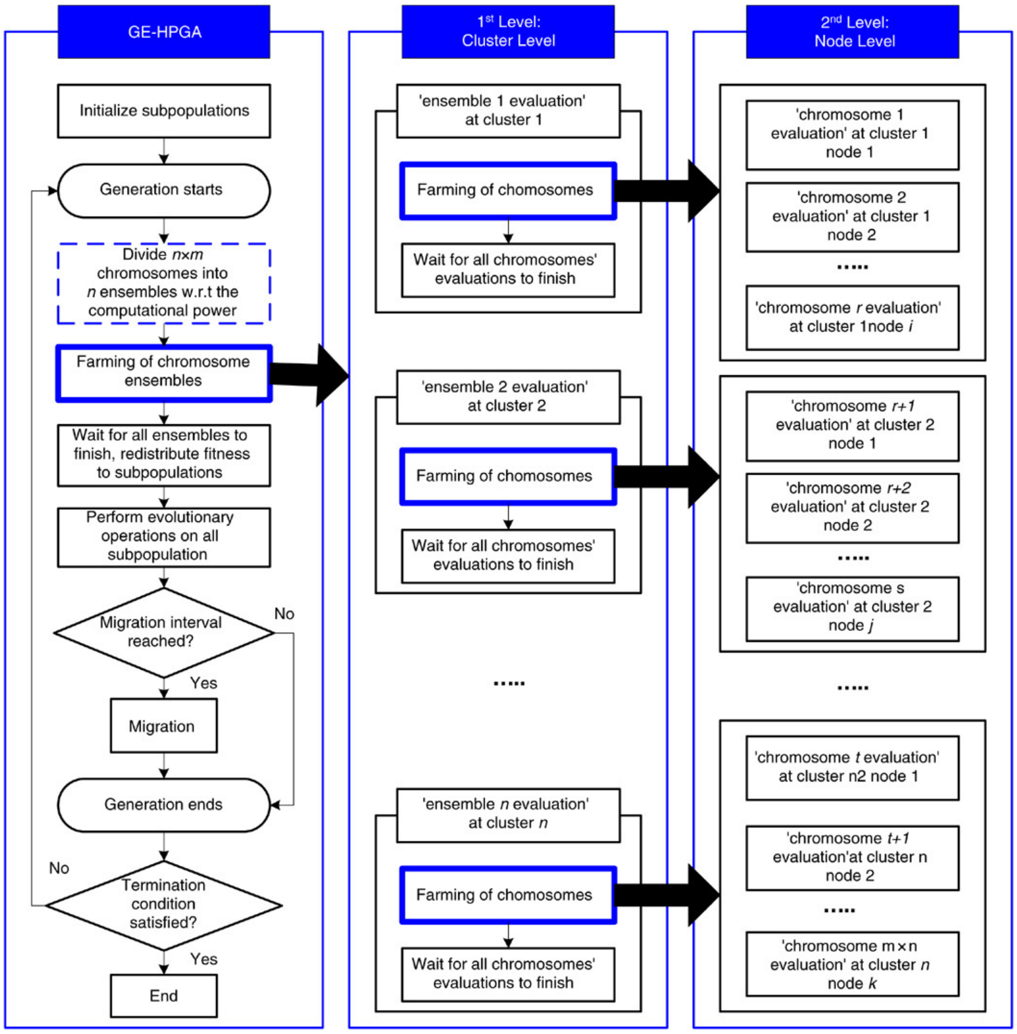

GE-HPGA
Initialize subpopulations
Generation starts
Divide nxm chromosomes into n ensembles w.r.t the computational power
Farming of chromosome ensembles
Wait for all ensembles to finish, redistribute fitness to subpopulations
Perform evolutionary operations on all subpopulation
Migration interval reached?
No
Yes
Migration
Generation ends
No
Termination condition satisfied?
Yes
End
1st Level: Cluster Level
'ensemble 1 evaluation' at cluster 1
Farming of chomosomes
Wait for all chromosomes' evaluations to finish
'ensemble 2 evaluation' at cluster 2
'ensemble n evaluation' at cluster n
2nd Level: Node Level
'chromosome 1 evaluation' at cluster 1 node 1
'chromosome 2 evaluation' at cluster 1 node 2
'chromosome r evaluation' at cluster 1node i
'chromosome r+1 evaluation' at cluster 2 node 1
'chromosome r+2 evaluation' at cluster 2 node 2
'chromosome s evaluation' at cluster 2 node j
'chromosome t evaluation' at cluster n2 node 1
'chromosome t+1 evaluation'at cluster n node 2
'chromosome m×n evaluation' at cluster n node k

Fig. 10. The decoupled GE-HPGA.

provide a 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 Fig. 10.

The core idea lies in the decoupling of the 'subpopulation evolution' Grid service in GE-HPGA (see Fig. 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 the purpose of fitness evaluation, 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 chromosomes for fitness evaluations.

To minimize the idling time of the 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 ith cluster for evaluations is estimated by:

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

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

The average wall-clock time and optimum shape obtained when using the GE-HPGA and DGE-HPGA to search on the subsonic inverse pressure design problem for a maximum of 100 generations are reported in Figs. 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 Eq. (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 ($pdpm : cemnet : surya : ec - pdccm = 100 : 45 : 35 : 20$).

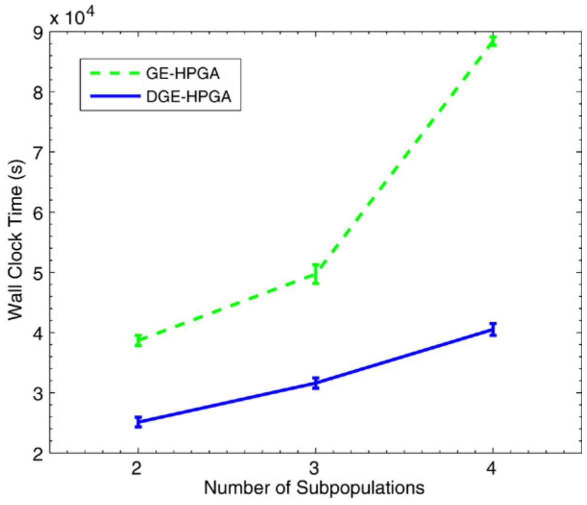


Fig. 11. Average wall clock time of GE-HPGA and DGE-HPGA in 2, 3, and 4-subpopulation runs in optimizing the aerodynamic airfoil design problem.

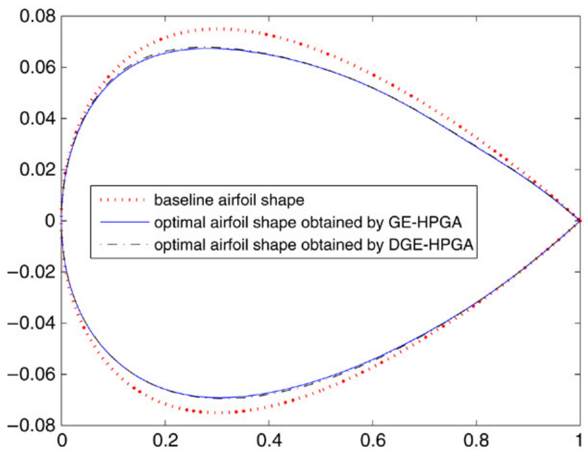


Fig. 12. Optimal airfoil shape obtained after 100 generations of the 4-subpopulation GE-HPGA and DGE-HPGA.

More importantly, the results in Figs. 11 and 12 show that the DGE-HPGA converges to the same optimal design airfoil shape as the GE-HPGA significantly faster in terms of wall-clock time spent.

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 the efficient parallel evolutionary design of problems with computationally expensive fitness functions, by providing novel features that conceal 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, a theoretical analysis of the maximum speed-up deriving from the GE-HPGA, as well as the practical conditions that must be fulfilled for a speed-up to be achieved, have been reported. Empirical results using a benchmark problem and then a realistic airfoil design problem further confirm that a speed-up can be attained as long as the noted bounds on fitness function cost, cluster size, and communication overheads of the Grid environment are satisfied.

Acknowledgements

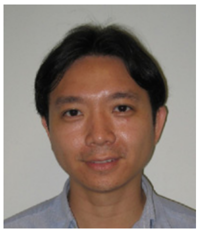
D. Lim and Y.S. Ong would like to thank Honda Research Institute Europe for sponsoring the research on the topic, and members of Nanyang Technological University and Honda Research Institute Europe for providing support. Without them, this project would not have been a successful one.

References

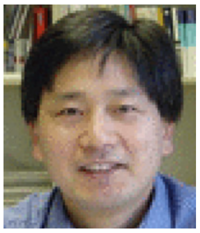
[1] Y.S. Ong, P.B. Nair, A.J. Keane, Evolutionary optimization of computationally expensive problem via surrogate modeling, American Institute of Aeronautics and Astronautics Journal 41 (4) (2003) 687–696.
[2] M. Olhofer, T. Arima, T. Sonoda, B. Sendhoff, Optimization of a stator blade used in a transonic compressor cascade with evolution strategies, in: Adaptive Computing in Design and Manufacture, ACDM, Springer Verlag, 2000, pp. 45–54.
[3] H.T. Kim, B.Y. Kim, E.H. Park, J.W. Kim, E.W. Hwang, S.K. Han, S. Cho, Computerized recognition of Alzheimer disease-EEG using genetic algorithms and neural network, Future Generation Computer Systems 21 (7) (2005) 1124–1130.
[4] Y. Gao, H. Rong, J.Z. Huang, Adaptive grid job scheduling with genetic algorithms, Future Generation Computer Systems 21 (1) (2005) 151–161.
[5] M. Li, B. Yu, M. Qi, PGGA: A predictable and grouped genetic algorithm for job scheduling, Future Generation Computer Systems 22 (5) (2006) 588–599.
[6] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, Reading, MA, 1989.
[7] E. Alba, A.J. Nebro, J.M. Troya, Heterogeneous Computing and Parallel Genetic Algorithms, Journal of Parallel and Distributed Computing 62 (2002) 1362–1385.
[8] E. Alba, J.M. Troya, Synchronous and asynchronous parallel distributed genetic algorithms, Future Generation Computer Systems 17 (4) (2001) 451–465.
[9] M. Nowostawski, R. Poli, Parallel genetic algorithm taxonomy, in: Proceedings of the Third International Conference on Knowledge-Based Intelligent Information Engineering Systems, KES'99, Adelaide, 1999, pp. 88–92.
[10] E. Cantu-Paz, A survey of parallel genetic algorithms, Calculateurs Paralleles, Reseaux et Systems Repartis 10 (2) (1998) 141–171.
[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, in: HPC on the Information Superhighway, HPC-Asia, 1997, p. 456. hpcasia.
[12] F.J. Villegas, T. Cwik, Y. Rahmat-Samii, M. Manteghi, A parallel electromagnetic Generic-Algorithm Optimization (EGO) application for patch antenna design, IEEE Transactions on Antennas and Propagation 52 (9) (2004) 2424–2435.
[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, C. Kesselman (Eds.), 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, in: International Conference on Advances in Infrastructures for Electronic Business, Science, and Education on the Internet, 2000.

[19] Y.S. Ong, A.J. Keane, A domain knowledge based search advisor for design problem solving environments, Engineering Applications of Artificial Intelligence 15 (1) (2002) 105–116.
[20] G. Xue, W. Song, S.J. Cox, A.J. Keane, Numerical optimisation as grid services for engineering design, Journal of Grid Computing 2 (3) (2004) 223–238.
[21] Q.T. Ho, W.T. Cai, Y.S. Ong, Design and Implementation of an efficient multi-cluster GridRPC system, in: IEEE Cluster Computing and Grid Conference, vol. 1, 9–12 May 2005, pp. 358–365.
[22] Special section: Complex problem-solving environments for grid computing, Future Generation Computer Systems 21 (6) (2005).
[23] 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) (2002) 1061–1074.
[24] A. Iosup, D.H.J. Epema, GrenchMark: A framework for analyzing, testing, and comparing grids, in: 6th IEEE/ACM International Symposium on Cluster Computing and the Grid, CCGrid'06, IEEE Press, 2006, pp. 313–320.
[25] 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 20 (7) (2004) 1119–1129.
[26] J. Cui, T.C. Fogarty, J.G. Gammack, Searching databases using parallel genetic algorithms on a transputer computing surface, Future Generation Computer Systems 9 (1) (1993) 33–40.
[27] 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 17 (4) (2001) 477–488.
[28] I. Foster, The globus toolkit for grid computing, in: Proceedings of the 1st International Symposium on Cluster Computing and the Grid, 2001.
[29] CoG Kit Wiki, [online]. http://www.cogkit.org.
[30] M. Massie, B. Chun, D. Culler, The ganglia distributed monitoring system: Design, implementation, and experience, Technical Report, University of California, Berkeley, 2003.
[31] S. Agrawal, J. Dongarra, K. Seymour, S. Vadhiyar, NetSolve: Past, present, and future; a look at a Grid enabled server, 2002.
[32] Globus: Information Services/MDS, [online]. http://www-unix.globus.org/toolkit/mds.
[33] H. Nakada, S. Matsuoka, K. Seymour, J. Dongarra, C. Lee, H. Casanova, GridRPC: A remote procedure call API for Grid computing, in: Grid Computing — Grid 2002, in: LNCS, vol. 2536, 2002, pp. 274–278.
[34] S. Tuecke, Grid Security Infrastructure (GSI) Roadmap, Internet Draft Document: draft-Gridforum-gsi-roadmap-02.txt, 2001.
[35] The Globus Project, GridFTP Universal Data Transfer for the Grid, The Globus Project White Paper, 2000.
[36] D. Geer, Grid computing using the sun grid engine, Technical Enterprises, Inc., 2003.
[37] J. Frey, T. Tannenbaum, M. Livny, I. Foster, S. Tuecke, Condor-G: A computation management agent for multi-institutional grids, in: Proceedings of the Tenth IEEE Symposium on High Performance Distributed Computing, HPDC10, 2001.
[38] M.J. Quinn, Parallel Programming in C with MPI and OpenMP, McGraw Hill, 2004.
[39] G. Ahmdal, Validity of the single processor approach to achieving large scale computing capabilities, in: AFIPS Conference Proceedings, vol. 30, Thompson Books, Washington DC, 1967, pp. 483–485.

Dudy Lim received his B. Eng. (Hons) degree in 2004 from the School of Computer Engineering, Nanyang Technological University (SCE, NTU) Singapore. He is now with the Emerging Research Laboratory, SCE, NTU Singapore, and a visiting researcher at Honda Research Institute Europe while pursuing his Ph.D. His research interests include robust evolutionary engineering designs under uncertain and dynamic environments, complex engineering designs harnessing Grid technology, Grid economy, algorithms, and neural networks. Previously, Dudy Lim worked as a Grid Engineer at the Grid Operation and Training Centre at Parallel and Distributed Computing Centre, SCE, NTU.



Yew-Soon Ong received his Bachelor's and Master's degrees in Electrical and Electronics Engineering from Nanyang Technology University, Singapore, in 1998 and 1999, respectively. He then joined the Computational Engineering and Design Center, University of Southampton, UK, where he completed his Ph.D. degree in 2002. Yew-Soon is currently an Assistant Professor with the School of Computer Engineering, Nanyang Technological University, Singapore. His research interests lie in computational intelligence, spanning: memetic algorithms, evolutionary design, and grid computing. He has been guest editor of IEEE Transactions on Systems, Man and Cybernetics, Genetic Programming and Evolvable Machines Journal, and has co-edited a volume on Advances in Natural Computation published by Springer Verlag, and is editing an upcoming book dedicated to Evolutionary Computation in Dynamic And Uncertain Environments in the Springer Series on Computational Intelligence. He is also a member of the editorial board for the International Journal of Computational Intelligence, an Emergent Technologies technical committee member, and an Evolutionary Computation in Dynamic and Uncertain Environments technical committee member of the IEEE Computational Intelligence Society. Dr. Ong is currently an Associate Editor of the IEEE Transactions on Systems, Man and Cybernetics — Part B.



Yaochu Jin received the B.Sc. and M.Sc. degrees from Zhejiang University, Hangzhou, China, in 1988 and 1991, respectively, and Ph.D. degrees from Zhejiang University in 1996 and Ruhr-Universitaet Bochum in 2001. He joined the Electrical Engineering Department, Zhejiang University, in 1991, where he became an Associate Professor in 1996. From 1996 to 1998, he was with the Institut fuer Neuroinformatik, Ruhr-Universitaet Bochum, first as a Visiting Scholar and then as a Researcher. He was a Postdoctoral Associate with the Industrial Engineering Department, Rutgers University, New Brunswick, NJ, from 1998 to 1999. Since 1999, he has been with the Division of Future Technology Research, Honda R&D Europe, Offenbach/Main, Germany. His research interests include control, optimization, and self-organization of complex systems using soft computing techniques. Dr. Jin is currently an Associate Editor of the IEEE Transactions on Control Systems Technology. He was a Guest Editor of the IEEE Transactions on Industrial Electronics Special Issue on Soft Computing Techniques in Intelligent Vehicle Systems. He has served on the Program Committees of several international conferences. He received the Science and Technology Progress Award from the Ministry of Education of China in 1995. He is a member of ISGEC.

Bernhard Sendhoff studied physics at the Ruhr-Universitaet Bochum, Bochum, Germany, and the University of Sussex, Brighton, UK. In 1993, he received the Diploma degree and in 1998 the Doctorate degree in physics from the Faculty of Physics and Astronomy, Ruhr-Universitaet Bochum, Bochum, Germany. He is Chief Scientist and Deputy Division Manager at the Future Technology Research Division, Honda R&D Europe GmbH, Offenbach/Main, Germany, in addition to being Head of the Evolutionary and Learning Technology Group. From 1994 to 1999, he was a researcher and postdoctoral member of the Institute of Neuroinformatik, Ruhr-Universitaet Bochum. His current research interests include the design and structure optimization of adaptive systems based on neural networks, fuzzy systems, and evolutionary algorithms. Dr. Sendhoff is a member of the ENNS and the DPG.

Bu-Sung Lee received his B.Sc. (Hons) and Ph.D. from the Electrical and Electronics Department, Loughborough University of Technology, UK in 1982 and 1987, respectively. A/Prof. Lee Bu Sung joined NTU in 1987 and is currently an Associate Professor and Vice-Dean (Research) with the School of Computer Engineering, Nanyang Technological University. He was the Director of Network Technology of the Asia Pacific Advance Network Consortium (APAN) from 2000–2003. In November 2003 he was elected the first president of the Singapore Research & Education Networks (SingAREN) society.

The objectives of SINGAREN are to advocate and champion advanced network applications and technology in Singapore; to be a platform for collective representation of the Singapore research and education networks community, as well as to facilitate the cost-competitive adoption of advanced Internet technologies for the research and education community. SingAREN is a partner of the TEIN-2 project, which connects European countries directly to ASEM member countries. He has been contributing actively in moving the organisations in the research direction, and also administratively. He was a member of the BioMedical Grid project from 2001–2002. He is presently the Group leader for the National Grid Network Working Group, which is coordinating the Grid research activities in Singapore. He is also a founding member and Chairman of the Nanyang Campus Grid, which is supporting the deployment of a campus wide Grid environment in Nanyang Technological University. He is an active member of Pacific Rim Middleware and Application Association (PRAGMA). In 2005, he and his team successfully deployed the Multi-organisation Grid Accounting System software, which they developed, across the PRAGMA grid test-bed covering over 10 sites in the region. His research interests are in network management, mobile and broadband networks, distributed networks, Network QoS, and Grid computing. He has published over 100 refereed articles in various research areas.