16

EA-based Problem Solving Environment over the GRID

Mohamed Wahib[1], Asim Munawar[1], Masaharu Munetomo[2]
and Kiyoshi Akama[2]
[1] *Graduate School Of Information Science and Technology*
[2] *Division of Large Scale Computing Systems, Information Initiative Center*
Hokkaido University
Sapporo, Japan

1. Introduction

EA-based problem solving environments have progressively evolved in the last two decades from explicit one-problem serial solvers to multi-solvers platforms running on vast distributed heterogeneous resources. Significant efforts in the literature were devoted towards designing EA-based problem solving environments. Those research efforts were mainly directed to innovating new EAs with a parallel implementation (Cantu-Paz, 2000), and the counterpart for those research efforts were directed towards designing and constructing parallel computing environments (Weise, 2007) that could host parallel and distributed implementations of EAs. Still for the evolution of the problem solving paradigms, problem solving environments have not fully shifted to parallel and distributed models, and even up till today practices of serially implementing EAs problems of medium complexity are still noticeable. These practices prevailed in part due to the continuous increase in clock speeds, multicore processors, and problem nature.

Yet, in the past few years, the significant increase in distributed resources, high bandwidth/ low latency networks and cheap data storage along with the wide expansion in problem scope and addressing new problem types that were not attainable before, all combined together strongly motivated to rethinking the strategy of designing EA-based problem solving environments. Various distributed computing paradigms were used as platforms for EA-based problem solving environments, (Munawar et al., 2008) gives a brief illustration of those paradigms. In this chapter we concentrate on a modern distributed computing paradigm, namely grid computing (Foster & Kesselman, 1999). In the recent years, grid computing acquired widespread attention from both research and industrial institutions, as it provides contextual establishment of open standard platforms for distributed computing (more details in section 2.1)

Constructing an EA-based problem solving environment requires two main streams of working, one is the algorithm design and the other is the challenges associated with constructing a Grid based platform. The algorithm design is significantly affected when using distributed technologies, therefore many points should be taken into account when designing algorithms for distributed environments: fault tolerance, support of

Source: Advances in Evolutionary Algorithms, Book edited by: Witold Kosiński, ISBN 978-953-7619-11-4, pp. 468, November 2008, I-Tech Education and Publishing, Vienna, Austria

interconnection for loosely coupled resources, support of late binding and dynamic migration. The other main stream which is the challenges accompanied with the grid computing environments both that are general (i.e. grid computing traditional problems) and specific (i.e. challenges related to EA-based solvers deployment).

In this chapter we present MHGrid (Meta Heuristics Grid), a service-oriented grid application that provides easy to use robust environment for meta heuristics optimization solvers, including EAs, over a grid. The objective of MHGrid is to offer a framework, using which a user can solve complex global optimization problems using EAs over a grid with minimal effort. MHGrid is designed in a service-oriented fashion and offers the following services to the user:

1. Allow the user to use any of the solvers registered with MHGrid to solve a problem with minimal input and in a black box manner.
2. Allow solver developers to write and register a new EA-based solver with MHGrid.
3. Allow solver developers to write and register a new objective function with MHGrid.
4. Ability to control the parallelization model of the solver and objective function for high complexity problems.
5. Provide all the preceding services at both the application layer and middleware layer.

This chapter is intended for a reader interested in the implementation of grid based problem solving environments of EAs. The reader is expected to have the basic background about EAs so the chapter scope will be focused on the grid computing problem solving environment and the effect of using the grid on the algorithms (i.e. parallelization and solver–to-objective function relation). We have tried not to overload the chapter with details by providing a very brief summary of the most notable and significant related work. So the chapter is focusing the discussion on the MHGrid platform and not devoted to being a comprehensive overview or survey of the previous work done in the area.

The chapter is organized as follows, section 2 discusses grid-based EAs problem solving environments, it briefly investigates the related work of grid-based EAs. Section 3 discusses the design, architecture and implementation of MHGrid as a problem solving environment. Section 4 presents a close-up, from the service orientation perspective, to the SOA (Service Oriented Architecture) that MHGrid encompasses and also the modelling of MHGrid solvers as services. Section 5 illustrates a full test case starting from a user registering his service to using the registered service. Finally, section 6 concludes the whole chapter and gives an insight for the future work.

2. Grid-based problem solving environment in EAs

This section will give a very brief introduction of grid computing and why use grid computing with EA followed by showing the impact of the grid on algorithm design. Also a revision of the related work is discussed.

2.1 Grid computing

The most commonly used definition to abstractly define a grid is: “Coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organization” (Foster & Kesselman, 1999). The most common among the categories of grid are:

- *Computational Grids:* Grids that basically aggregate computational resource to offer transparent computational power to the applications that use them.

- *Data Grids:* Used to manage and control access to huge distributed data stored on heterogeneous storage devices.
- *Utility Grid:* A market-oriented Grid that applies utility computing concepts in designing the grid.

EAs problem solving environments when associated with grid fall under the category of computational grids. Yet in some problem solving environments that require extensive data handling, techniques that are basic components in data grids such as data replication and staging are introduced in the computational grid. Now almost every production level computational grid has support to what is known as workflow (transfer of data and files across the grid). Nonetheless, grid computing when addressed in EAs conventionally means computational grids.

The grid architecture as shown in figure 2 is a revised version of the traditional grid architecture. The traditional grid architecture is composed of three layers only, the resources, the middleware layer and the application layer. The middleware is a software layer that resides between an application and the underlying platform, in grids the middleware hides the underlying low-level details and complexities from the application layer. Yet, practically in grids, a big semantic gap lies between the middleware and the application layer, so (Abramson, 2006) revised the traditional architecture and modified it by splitting the middleware layer into two layers, the upper middleware layer and the lower middleware layer. This architecture was adopted in MHGrid due to its enhanced subjective representation and ease of modelling.

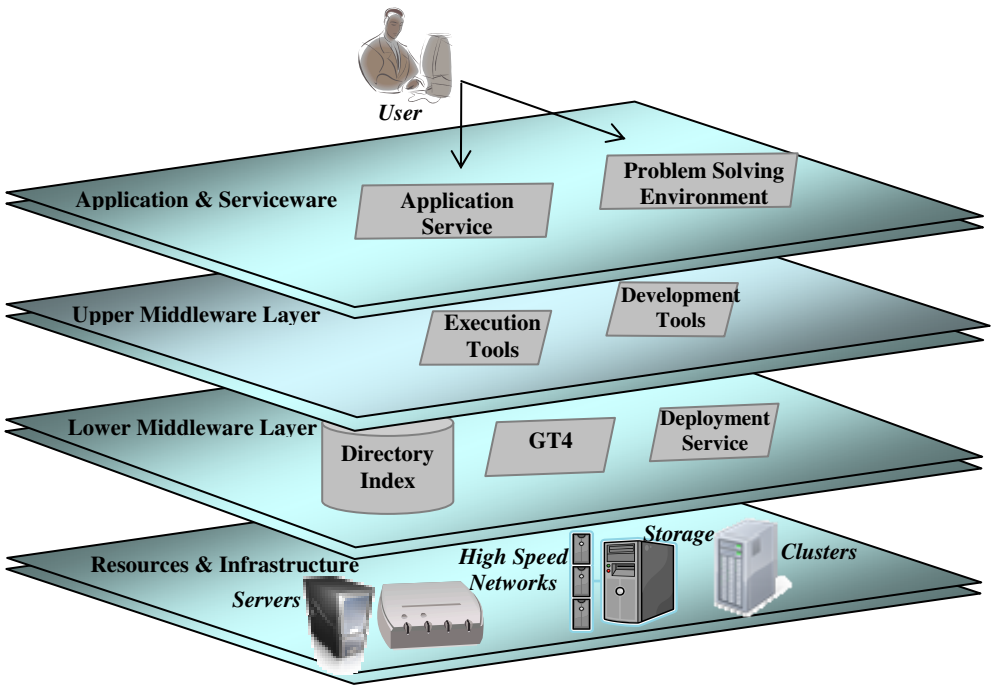

Fig.2. General revised architecture of MHGrid as a computational grid.

2.2 Why grid computing for EAs

An often repeatedly aroused question is why use grid computing for EAs, as it naturally adds a significant overhead to the performance compared to other technologies such as cluster computing. Also designing and implementing a problem solving environment over the grid involves much more complexity than compared to other techniques. The answer to that question lies in a three point checklist by Ian Foster (Foster, 2002), that is when satisfied, classifies the distributed computing framework as a grid. The checklist is:

- Resources are not administered centrally.
- Open standard, general-purpose interfaces and protocols are used.
- Non-trivial quality of service is achieved.

From the checklist above, considering the non-trivial quality of service, grid will be a good choice as a distributed computing paradigm. The major non trivial quality of service is the grid application hosting environment. As the grid application can be available over the Internet and accessed through a Web portal (this is the case in MHGrid), so the hosting environment in this case is the Internet, and the user could be any person accessing the portal and having a valid grid certificate. Other parallel computing paradigms on the other hand (e.g. cluster computing and supercomputing) are available locally in the scale of a LAN, and thus the users in this case, are users having direct access to the resources. This feature of grid computing (i.e. availability over Internet and Intranets) is a basic advantage that attracts developers in the case of applications that are intended to be accessed widely with remote resources.

Other non trivial qualities of service include availability, latency and throughput. A more detailed study on quality of service metrics and aspects in grid is at (Daniel & Emiliano, 2004). The handling and presentation of those metrics could be through defining utility functions (Chunlin & Layuan, 2007) or by defining the provided functionalities as services and thus have a SLA (Service Level Agreement) for each service. One more case that will be most suitable to adopt grid technology with EAs and that is the case of using grid to aggregate resources to provide a huge underlying computational power that enables addressing new complex and relatively expensive problems that were not addressed before due to resource limitation. One fine example to this case is (Chrabakh & Wolski, 2006) in which the authors were able to solve problems that were not solved before due to resource limitation. (Chrabakh & Wolski, 2006) is mainly designated for SAT problems but it still gives a clear evidence of how the grid can be used to address problems of higher complexity compared to other distributed computing paradigms.

Summarizing the need of using grid for EAs; the ability to use non-trivial quality of service metrics rather than speedup, and the ability to use the application over the Internet rather than direct local access is particularly the most important non trivial quality of service. Another reason will be the ability to address new problem of high order complexity and cost depending on the grid ability to aggregate heterogeneous geographically dispersed resources.

2.3 Impact of grid on algorithms

A common practice of running EAs over grid is to use legacy EAs that were written to run on another parallel computing paradigm and running it intactly on the grid. This practice for some algorithms will not be suitable and will be error-prone (i.e. an algorithm that is

tightly coupled with out being able to tolerate communication delays will have very significant performance degradation.). From the other side, if the algorithm design did not take into account the nature of the grid it will not benefit most from using a grid and will at best expectations run without any degradation in performance. Therefore the following points should be taken into account when designing EAs for a grid:

- The algorithm should be designed and implemented in a manner that supports interconnection of loosely coupled entities.
- The algorithm should be able to tolerate communication delays for up to 100's milliseconds without significant performance degradation.
- The algorithm should have interfaces allowing for late binding to allow a space for dynamic scheduling and workflows.
- The algorithm should be able to rely on remote data sources as copying the data locally before executing might not be feasible.
- The Algorithm should be fault tolerant.

2.4 Related work

Projects using EAs over grids or EAs problem solving environmets over grid are numerous. Table 1 summarizes some of the notable efforts in this direction and also projects trageted to optimization problem solving environments in general. The table has a comparison of MHGrid with different projects, of different scopes and using different technologies, it gives a close-up to the relation of optimization problems with grids.

NEOS (Czyzck, 1998)*:* The only non-grid based project among the other projects in the table, yet it later motivated using grids for the similar functionality. NEOS is simply a client-server system that is dedicated to solving optimization problems by allowing the user to submit his optimization problems as well as allowing the user to add a solver of his own through NEOS management. The user has no control over the solver parallelization.

Folding@Home (Larson, 2003): This project is categorized as what is called desktop grids, utilizing processor cycles of distributed non-dedicated normal PCs, it was designed to perform computationally intensive simulations of protein folding and other molecular dynamics, it involves GROMACS optimization, and it does not allow user interaction with the job running, the user just installs the client and offer his resources for usage. Folding@Home has not provided optimization solving problem solving environment, yet it is a well known example of how aggregated resources when combined can address new problem scope.

Nimrod/O (Abramson et al., 2000): A very significant project as the authors not only designed the problem solving environment but they also added and modified the grid middleware to adapt with the grid application. Nimrod/ O offers namely 4 optimization solving packages solving non-linear optimization problems, but it doesn't allow the user to add his own solver and limits him/ her to the provided solvers. Further to mention, Nimrod/ O uses an ontology based module to guide the user to the best solver considering his/ her problem.

GEODISE (Cox et al., 2002): Specific to optimization problems in computational fluid dynamics, it uses Application Service Provision (ASP) and offers the services through a custom Matlab toolbox, it was designed for production and like Nimrod/ O and had a commercial version.

Table 1. Comparison of Different projects providing optimization solving environments.

Project	Scope	Black-Box	Arch-itecture	Middleware	Info. Exchange	Adding New Solver By User	Security	Solv el
NEOS	Any Optim-ization	Not Supp-orted	Trivial Web Applic-ation	Non-Standard	Plain Sockets	Allowed(Thr-ough Managem-ent)	No	Pre
Folding @Home	GROMACS Optim-ization	Not Supp-orted	Distri-buted	SMP Client	Client-Server Sockets	Not Allowed	Yes (2048 bit Digital Signat-ure)	Ind Wo
Nimrod/O	Non-linear Optimization	Not Supp-orted	Event Driven	OGSA Compliant	Active Sheets	Not Allowed	Yes (GSI)	Pre
GEODISE	Fluid Dyn-amics	Not Supp-orted	ASP	OGSA Compliant	Environment API	Not Allowed	Yes (GSI)	Pre
OSP	Decision Supp-ort System	Not Supp-orted	ASP	Aggregated Components	AMPL/ M PL	Not Allowed	Yes	Pre
GE-HPGA	Island-model Genetic Algorithm	Supp-orted	Trivial Distribut-ed Appli-cation	OGSA Compliant	Environment API	Not Allowed	Yes (GSI)	Fix
MW	Combinator-ial optimization	Not Supp-orted	ASP	OGSA Compliant	Java Interfaces	Alowed(pa rtially)	Yes (GSI)	Str De
MHGrid	Global Optimization with metaheuristics (GA,SA,EDA ,...)	Supp-orted	Service orient-ed grid applic-ation	OGSA Compliant	MHML (XML-based)	Allowed (Through Portal)	Yes (GSI)	Solv def own Dis

OSP (Optimization Service Provider, www.osp.org): A recent EU funded project using ASP for solving decision support systems optimization problems. It was later extended to another project (WEBOPT, www.webopt.org) that uses the E-service model instead. Both of them intend to offer a web-based DSS optimization solving environment.
GE-HPGA (Lim et al., 2007): It is similar to MHGrid in offering black-box optimization, the framework is limited to only one solver and the main target was to offer speedup compared to other distributed models. To achieve it's target, GE-HPGA used the island model GA that splits the population into sub-populations to minimize the program inter-communication as much as possible and thus minimize the grid overhead as much as possible.
MW (Glankwamdee & Linderoth, 2006): A framework that is targeting to offer combinatorial optimization solvers over the grid, MW has a very interesting feature for solver and task definition where through MW API (Java interfaces), the user can implement the interfaces to define his task, and also his solver. This technique solves the problem of solver deployment but on the other hand enforces the user to use Java language which is relatively slow, yet it eases the usage of MW by defining flexible interfaces.
MHGrid is a service oriented grid-based framework compliant with OGSA, Open Grid Services Architecture (Foster et al., 2005). It offers various solvers to global optimization problems. All solvers belong to the meta heuristics family of solvers (meta heuristics is a wide category containing EAs and other solver types like search heuristics). Solvers that are meta heuristics based support black box optimization in which the user provides the input and receives the output without knowledge of the underlying computation, black box optimization is a highly desirable feature in optimization solvers to relief the user from involvement in too much details. As for the user interface, the user could use MHGrid's web portal or directly use the Web services of MHGrid. Information interchange between the user and the system is maintained through MHML (Meta Heuristics Mark up Language), Details for MHML are in section 3.4.

3. MHGrid: A grid-based global optimization problem solving environment

MHGrid is a framework dedicated for solving optimization problems over Grid. The main target of the framework is global optimization problems (global optimization is a branch of applied mathematics and numerical analysis that deals with the optimization of a function or a set of functions to some criteria). The framework is intended for the solvers based on heuristic or meta heuristic searching methods.
MHGrid targets general purpose global optimization problems, a major challenge is that according to the No Free Lunch theorem, NFL, (Wolpert & Macready, 1995), no single optimization algorithm will give good results will all problems. The strategy that MHGrid uses to overcome this part is by offering diverse techniques for global optimization covering a wide range of problem type, and also offering mediation between the problem-solver pairs to assure that the solver used is the most adequate to the problem in hand. The strategies enforced by MHGrid to overcome the NFL problem are discussed later in sections 4.1, 4.2.
MHGrid provides the following functions to the user:

- Allows the usage of a solver registered with MHGrid to solve a problem in hand, this is done with a minimal input.
- Enables solver developers to write a new solver that is integrated with MHGrid sing MHAPI, and register it.
- Enables solver developers write a new objective function and register it.

- Do all the previous either through MHGrid's web portal or by directly consuming MHGrid's Web services.

The key contribution is combining the computational power offered by grid technology along with the optimization efficiency of meta heuristics algorithms to give an easy to use general purpose Problem Solving Environment (PSE) for global optimization problems. All MHGrid Web services are WSRF complaint web service to enable the user to use the services directly or through the portal. We have used a unique hybrid parallelization technique that employs GridRPC (Symour et al., 2002) + GridMPI (Ishikawa et al., 2005) approach to dynamically adapt to the grain size of the solver. We have also developed an XML based mark up language, MHML, which acts as an interface between the user and MHGrid Web services.

3.1 MHGrid architecture

Figure 3 gives an overview of MHGrid's architecture, it shows the services that are directly or indirectly used by MHGrid. As the figure shows, the base layer is a high performance grid network, on the top of that runs our Web services in a globus GT4 container (Foster, 2006). All other technologies and services are either build on top of globus or they use globus in one way or the other. Globus Toolkit Monitoring and Discovery Service (MDS) are used by the Condor-G scheduler (Frey et al., 2001) to collect information about the current state of the dynamically changing Grid environment. This information is used by the Condor-G based scheduler to negotiate SLA (Service Level Agreement) with the web service and also to manage and schedule the jobs in a better way.

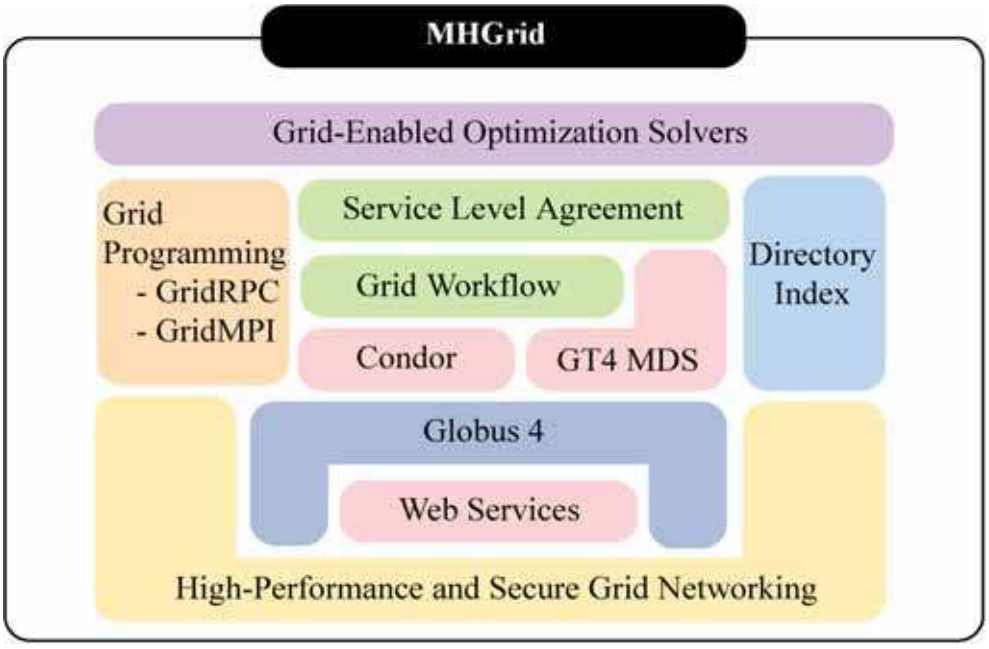

Fig.3. MHGrid architecture at an abstract level.

GridRPC (Symour et al., 2002) - MHGrid uses Ninf-G (Tanaka et al., 2003) implementation of GridRPC- and GridMPI (Ishikawa et al., 2005) are also built on top of the Grid technologies, they are Grid variants of the famous Remote Procedure Call (RPC) and Message Passing Interface (MPI) technologies respectively and their use is almost similar to that of their non-Grid counterparts. Next is the Directory index, which is responsible for storing the logs and maintaining the indexes for the solvers and objective functions. A Workflow management module is needed for managing data staging in case of solvers requiring remote datasets. Service Level Agreement (SLA) layer is used for controlling the negotiations between the resource broker (i.e. Condor-G Central Manager) and the users submitting jobs. On top of all these layers come the solvers that run on the Grid to solve global optimization problems.

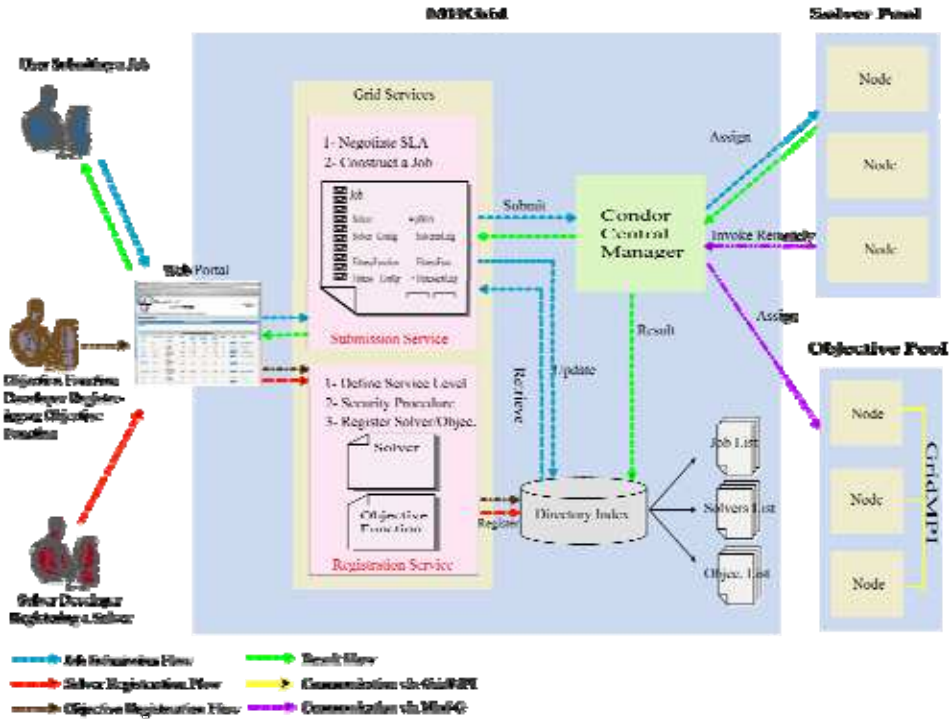

Fig.4. A close-up to MHGrid internals.

Figure 4 gives an insight to the internals of MHGrid and the flow of information inside MHGrid. The arrows with short dashed show the information flow for a user submitting a job, while the dashed-single dotted show an objective function developer registering an objective function and the long dashed are of a solver developer registering a solver. Different modules and functionalities provided by the framework are visible from the figure. The modules of the framework are as follows:

Web Portal: A 2nd generation portal using Gridshpere (Novotny, 2004) as a portlet container. Custom JSR compliant portlets are added to enable the user to use MHGrid with minimal effort. The portal is simply a client application consuming MHGrid's Web services on behalf of the user.

MHGrid's Web services: Runs in a globus container and are the core of MHGrid connecting all components together. Three main services exist, one for retrieving the list of solvers and objective functions registered, one for adding a new solver or objective function to MHGrid and the last is for job submission.
Directory Index: A database that consists of all the objective functions and solvers registered with the framework. It maintains a list of all the jobs and is also responsible for keeping a log of all the previous runs along with the obtained results.
Condor-G based Scheduler: A simple scheduler that is responsible for scheduling jobs to appropriate resources in the grid.

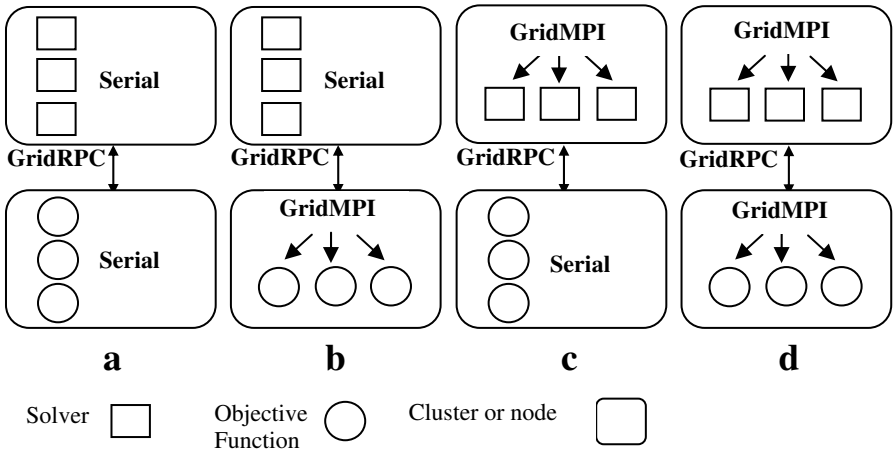

Fig. 5.Different grain size depending on parallelization combination. a) A solver running in serial fashion and objective function computing also running in serial, simplest scenario with no parallelization. b) The solver running serial but the objective computing is running parallel in another cluster, Master-slave GAs are an example that will use this scenario. The Master here is the solver process running in serial and the GridMPI objective function processes are the slaves. c) The solver running in parallel while objective function calculation is serial, a solver like parallel BOA will use this scenario where the objective function calculation is not heavy while the solver involves heavy computation (candidate selection). In this scenario one of the GridMPI solver processes is a controlling node that will call upon objective function calculation. d) Both solver and objective function are running in parallel on different clusters. This scenario will have one of the GridMPI solver processes acting as a controlling node that will be acting as a master for the GridMPI objective function processes.

3.2 Dynamic grain size in MHGrid

Parallelization in meta heuristics in general differs depending on the algorithm communication/ computation ratio. To offer an environment that will host a variety of solvers, there is a necessity of having a mechanism that allows the usage of different parallelization technologies to be used within the solvers and objective functions. MHGrid uses a hybrid of two technologies, GridMPI and GridRPC (MHGrid uses Ninf-G, a wrapper for GridRPC). Figure 5 shows how the mixed use of GridMPI and GridRPC can offer

different parallelization models providing the solver developer with flexibility in designing his solver. This unique parallelization technique employing GridRPC and GridMPI was first used in (Takemiya et al, 2006) for a specific problem. MHGrid deploys this technique as a general model for dynamic grain size definition.
The deployment of solvers and objective functions in such a way to provide those parallelization models is a complicated process that uses both Ninf-G and Condor-G deployment techniques. Detailed method of objective function deployment is discussed in (Munawar et al., 2008).

3.3 Solver developing and integration to MHGrid

When a user requires adding a solver to MHGrid, he is required to provide two things, the first is the solver source files and the other is an MHML file including the SLD of the solver to be added (the SLD part of MHML usage will be explained later in section 4.2). On the other hand for the user to be able to integrate his solver with MHGrid, he/ she needs to use MHAPI. MHAPI is an API provided by MHGrid that includes a set of functions that allow the user to run and deploy his solver on MHGrid. As shown in figure 6 the solver developer writes the solver and uses the APIs in MHAPI for the following:

- Reading the input and configuration data from the job's MHML file.
- Calling the objective function calculation whenever needed.
- Initialize the deployment of the objective function. Then MHGrid will transparently deploy the objective function on behalf of the user.

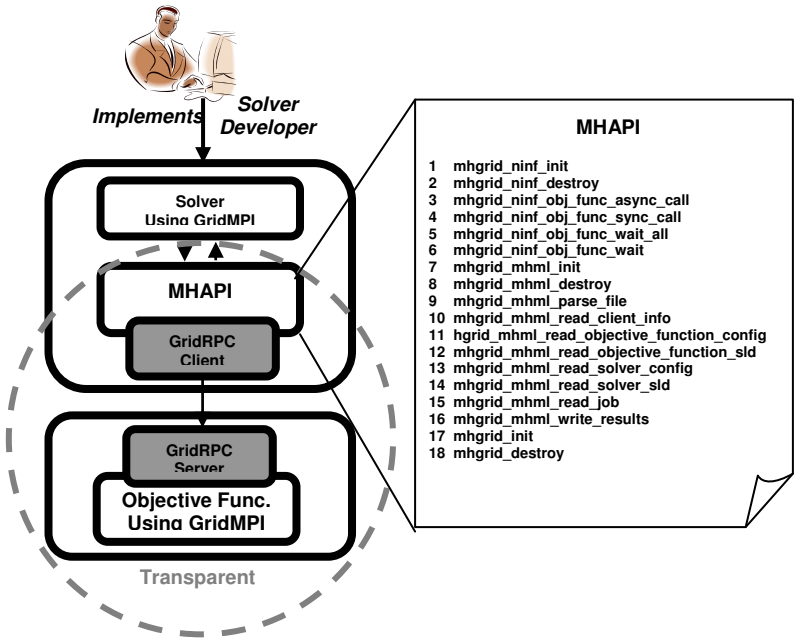

Fig.6.Main functionalities provided by MHAPI. Note that every thing is kept transparent from the solver developer.

Two points to note here about objective function calling and objective function deployment. For objective function calling, the writer of the objective function is usually different from the writer of the solver, so for an objective function to be used by solvers in MHGrid, it must comply with a predefined Ninf IDL. This IDL defines the interfacing between the solver and any objective function that will be used with it with eyes on the different problem encodings that can be used (e.g. binary, real, combinatorial … etc). Figure 7 shows how a simple Ninf-IDL file looks like.

```
// Sample IDL file

Module obj;
Define obj-func(IN int in_length_of_chromosome, IN float in_chromosome[length], OUT float *out_fitness)
"sga on rpc"
Required "obj_func.o"
{
        Extern float obj_func(float length, float *x);
        *out_fitness = obj_func(int in_length_of_chromosome, in_chromosome);
}
```

Fig. 7.Simple sample of a Ninf-IDL file.

3.4 MHML

MHML is an XML-based language providing all the functionalities required from a language to describe meta heuristics information interchange. Full details about MHML is beyond the scope of this chapter, MHML language is fully demonstrated in (Munawar et al., 2007), we will only summarize why the need to use MHML and the basic features of MHML.

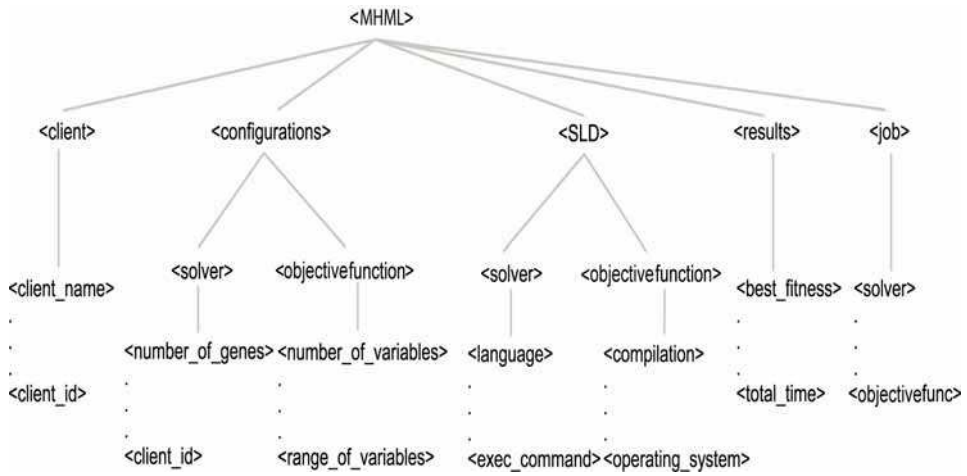

Fig.8.Top level hierarchy of MHML.

The rationale behind MHML was the need for standardizing the communication interface. Standardizing the communication interface not only enables a flexible design, but also eases the process of extendibility and interoperability. XML was chosen as it appears the most promising information interchange language, and its wide dominance in the area of web-based information interchange.
MHML basically is an extension/ modification to an earlier attempt by (Alba et al., 2003). (Alba et al., 2003) proposed a language to configure optimization algorithms as XML DTD. Yet, it failed to address important issues considering the configuration of optimization algorithms. MHML offers many advantages compared to (Alba et al., 2003), from the top-level hierarchy of MHML shown in figure 8, it is clear that MHML has the capability to represent: *Jbb configuration, Solver description and configuration, Objective function description and configuration, submitting client information* and *job results*

4. Service orientation aspect in MHGrid

Creating a general framework for global optimization problem solving is challenged with two major problems that will compromise the generality-to-performance trade-off; the first problem is that if the set of available solvers is fixed then the overall scope of the framework will be limited to the solvers in hand. The second problem is the reduced efficiency due to week or non existing relation between the solver and the problem using the solver. Added to the complexity of the second problem is that the nature and availability of the underlying resources is dynamically changing in Grid-based systems. Another complexity added to the second problem is the compound nature of meta heuristics based solvers, as Meta heuristic based solvers constitute of the main solver code and the objective function which is a computationally independent, cost expensive and repeatedly called function. Thus, the need to formulate the interaction between solver and objective function counterparts.
MHGrid tackles these two problems by adopting service oriented architecture (SOA), this SOA is attained in MHGrid by applying a set of strategies in both the vertical and horizontal direction. And by applying these set of strategies that melt down MHGrid in a SOA frame, the performance of MHGrid as a framework is leveraged to the desired level of being a general framework (i.e. addressing problems of different scope.) while still offering a reasonable performance to the problems submitted. Figure 9 shows three different models with different problem type to performance relations. The Narrow scope-High Quality model is the typical case of optimization problem solvers according to NFL (Wolpert & Macready., 1995). The Wide scope low quality model is a model having a set of robust solvers. This model targets average performance for wide scope of problem types. The last model, MHGrid, targets a wide problem scope with performance that is high above the average by modelling MHGrid in a SOA through applying strategies to expand in the horizontal and vertical directions.
This section will give a close-up to the SOA of MHGrid by discussing the strategies used to model MHGrid into a SOA. An important point to note here is that MHGrid doesn't embrace SOA by just using OGSA and Web services in the middleware layer, as normally in SOA context, modelling a framework to fit into a SOA implies using Web services. This is not the case in MHGrid, as Web services – though used in all modules of MHGrid – are just tools in the middleware layer. The SOA referred to here is effective at the application layer (i.e. solvers as services), section 4.2 discusses this point in details. The next sections will discuss the horizontal expansion strategies, vertical expansion strategies and finally the impact of those expansions on the adaptation of MHGrid into a SOA.

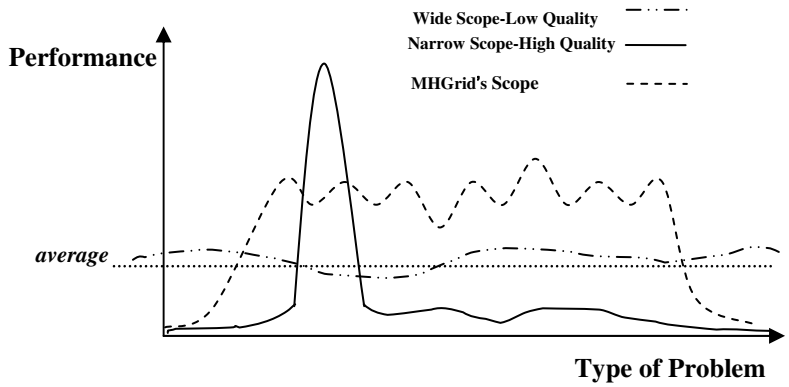

Fig.9. MHGrid scope according to problem-type space.

4.1 Horizontal expansion strategies

Expanding MHGrid in the horizontal direction is mainly directed to widen the solvers base. The strategies that MHGrid use to expand horizontally can be summarized in two points:

- Offer a variety of state-of-art robust solvers that make the framework suitable for different problem types.
- Allow the user to add his own solver(s) and objective function(s).

For the first point, a set of robust solvers developed by the information systems design laboratory at the information initiative center, Hokkaido university are to be used in MHGrid platform. These solvers are the fuel of MHGrid that provide the ability to address a wide variety of problems. The second point is as mentioned before in section 3.3, providing a mechanism to allow the solver developers to add their solvers and objective functions.

4.2 Vertical expansion strategies

The vertical expansion strategies are much more complicated as they are mainly concerned with increasing the semantics of the solver to problem relation. The following are the strategies:

- Solvers and objective functions are represented as services in MHGrid, thus binding a Service Level Description (SLD) with each solver/ objective function to describe the service level offered by the solver/ objective function. MHML has two main sections one for solvers and the other for objective functions. The SLD part should be submitted with newly added solvers/ objective functions. The SLD section contains information like what problem type is the solver targeting, problem encoding and what model of parallelization is used (e.g. Island model GA will use any parallel model while master/ slave pBOA requires the solver to run in parallel on the same cluster). The SLD information is later used to guide the user for which solver to select to the problem in hand and to check if the grid resources will support the parallelization model required.
- Having an M-N relation between the solvers and objective functions registered with MHGrid, where the user can run the same solver against many objective functions and vice versa. This strategy is handled through the Ninf-IDL interface described in section 3.3.

- Allowing the solver developer to control the parallelization model in the solver / objective function he writes. The solver developer can choose the parallelization model and thus the grain size as mentioned in section 3.2.
- Offering two SAPs (Service Access Points) for the user of MHGrid, one of them is the web portal and the other is by consuming the MHGrid's Web services directly. Accessing MHGrid services through the portal will be shown in the test case of section 5, also there is another SAP that can be used in case the user wants to avoid the overhead in using the web portal and also to use MHGrid's services automatically in case he needs that.
- Having a Service Level Agreement (SLA) for each job submitted to MHGrid. Initially upon job submission and after the user chooses the solver/ objective function pair, the scheduler checks the state of the available resources, then the SLA manager using the state of resources along with the solver/ objective function SLD informs the user with the expected scenario that rises from running the selected solver/ objective function running on the current available resources. The SLA in the case of MHGrid is at the application layer and not the middleware layer, and therefore refraining from the expected SLA procedure at middleware (i.e. SLA based scheduling). SLA at the application layer guarantee to the user that his problem is well matched to a solver, while if at middleware layer will be targeting QoS metrics such as time, cost and resources availability. The current SLA implementation is rather trivial, but different options are now being investigated and it is anticipated that SLA mechanism will later use e-contracts at the application level.

4.3 MHGrid as a grid application benefiting from SOA

Grid applications are combined with SOA and service fundamentals in many projects, and often the grid application that are modelled after SOA are referred to as *service-oriented grid applications*. The case of MHGrid despite being a service-oriented application, yet it used a different approach to combine SOA with grid technology. MHGrid as a framework is designed to be a *general* framework for global optimization, yet this goal was challenged with the NFL theorem, and so the expansion in both directions was thought of in order to enable more generality for MHGrid. This expansion design for MHGrid was clearly consistent with SOA fundamentals and concepts, for example the following are the SOA projections mapped to the vertical expansion strategies:

- Solvers as services with SLDs. Mapping: A well known practice of SOA, where every service in a SOA model should have a description of what it is doing in order to be used later for QoS process. Analogous to WSDL associated with Web services.
- M-N solver to objective function relation (one solver can be associated to many objective functions and vice versa). *Mapping: Service interoperability is a main concept in SOA*.
- Solver developer control over the parallelization model. Mapping: From SOA perspective, this is providing strong semantics for inter-services relations.
- Offering two SAPs. *Mapping: The two service access points for the solvers in MHGrid comes in favour of ease of use, this polymorphic interfacing to the services is indeed a merit from SOA perspective.*
- Having an SLA for each job submitted. *Mapping: A straightforward SOA pillar.*

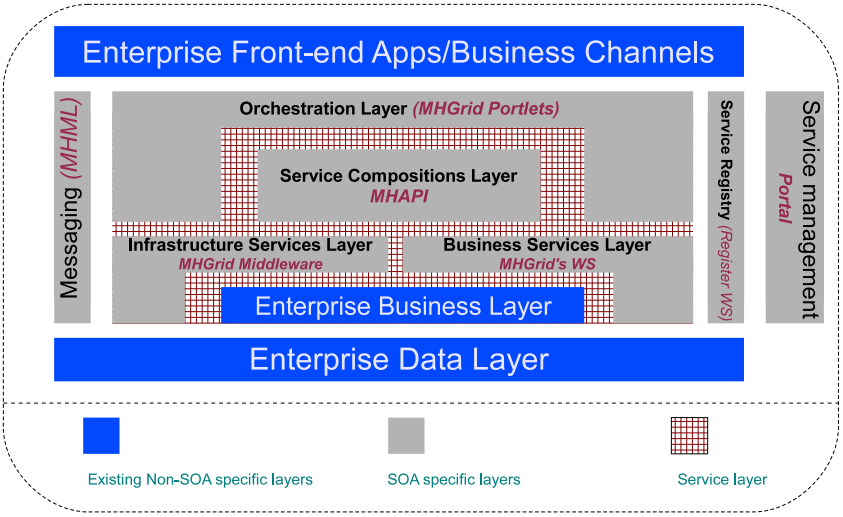

Fig.10. MHGrid modules mapped to a typical SOA layout.

The point of concern that can be concluded from merging MHGrid as a grid application with SOA, is that MHGrid was not designed as a SOA compliant model in order to benefit from the typical advantages of SOA such as ease of extensibility, but MHGrid was framed into a SOA model to achieve the basis of having a general problem solving framework in terms of wide problem type support. Figure 10 shows a mapping of MHGrid modules to a typical SOA layout.

5. Test case of MHGrid from user perspective

This section illustrates a test case example for MHGrid from the user perspective. The illustration will start by a solver developer registering a solver he wrote for MHGrid, then as a user retrieving the list of solvers and objective functions and finally submitting a job to MHGrid.

- Solver registration: The solver developer will initially write his solver that uses MHAPI, and then write the MHML file with the SLD section of the solver. Then the solver developer logins to the portal opens the solver registration portlet and uploads both the solver tar ball and the MHML file. The solver developer will be notified though his e-mail registered with the portal. Figure 11 middle snapshot shows the solver registration portlet while registering a solver.
- Job submission: The job submission is done in two steps, first the user uses the retrieve portlet to get a list of all the registered solvers and the registered objective functions. For each solver and objective function displayed, the information for the corresponding SLD is displayed to give guidance to the user. Figure 11 top snapshot shows the retrieve portlet where the user can view the solvers and objective functions before deciding which one to use. The next step where the user actually submits the job, the user will switch to the job submission portlet and choose a solver/ objective function pair, and then the portlet will give the user an indication of how the current available resources are coherent with the SLDs of the solver and objective function. After the user decides which solver/ objective function pair to use, he has to supply the MHML job file, and he'll later get the MHML result file on his e-mail. Figure 11 bottom snapshot shows the job submission portlet while in submission process.

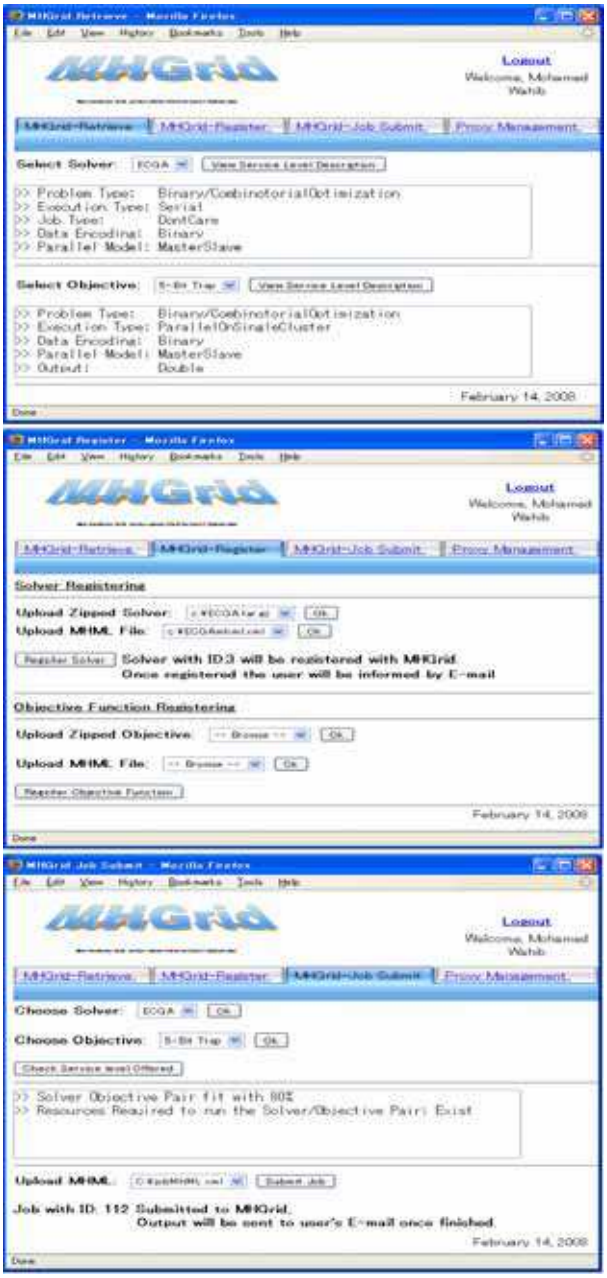

Fig.11. Top Snapshot: A user registering a solver with MHGrid through the web portal. Middle snapshot: A user retrieving information about the solvers and objective functions registered with MHGrid through the web portal. Bottom snapshot: A user submitting a job to MHGrid through the web portal

This was a simple example case just to acknowledge the reader with how MHGrid is viewed from the user perspective, nevertheless, MHGrid can still be accessed directly from the Web services, but illustration for that was skipped to refrain the user from details outside the scope of the book.

6. Conclusions and future work

This chapter presented a grid based problem solving environment that uses EAs and other algorithms all falling under the meta heuristics category to offer black box global optimization for the user. The chapter first highlighted the grid computing technology and then discussed with reasons behind using the grid for MHGrid, Meta Heuristics Grid, and the benefits of the grid technology compared to other distributed paradigms.

Then a comparison of MHGrid with related work was discussed, to imply the concepts behind the design of optimization solving grid applications. The design and implementation of MHGrid was explained, including the layered architecture, the workflow inside the framework and explanation of MHAPI, a library that allows the solver developers to integrate their solvers with MHGrid.

MHGrid as a model was expanded in both the vertical and horizontal directions in order to widen the base of MHGrid to be a general framework rather than being tailored to one problem type. The expansion strategies reformed the architecture of MHGrid into a SOA, the main impact for MHGrid adopting SOA was the representation of solvers and objective functions as services and thus having the service oriented grid application mostly affecting the application layer whilst using OGSA and Web services at the middleware layer. A sample example case was demonstrated to acknowledge the reader with the user perspective of MHGrid.

For the future work, modifications and extensions will cover different aspects. Major points will include adopting a more sophisticated SLA mechanism, defining new interfaces that allow one solver to use another solver, for example pBOA algorithm can internally use Tabu search for candidate offspring selection, and one more important point is to conduct more study on the dynamic grain size in EAs to reach the best formulation of parallelization models adopted. Other minor points will include enchantments on the portlets to auto generate the MHML files on behalf of the users.

7. References

Abramson, D. ; Lewis, A. & Peachy, T. (2000). Nimrod/ O: A Tool for Automatic Design Optimization, *Proceedings of 4th International Conference on Algorithms & Architectures for Parallel Processing; ICA3PP 2000*, pp. 90-98, ISBN, Hong Kong, December 2000, World Scientific Publishing

Abramson, D. (2006). Applications Development for the Computational Grid, *Proceedings of APWEB 2006*, pp. 1-12, China, January 2006, Springer, Harbin

Alba, E.; Garc-Nieto, J. & Nebro, A. (2003). *On the Configuration of Optimization Algorithms by Using XML Files,* Technical report, http:/ / neo.lcc.uma.es/ publications/ Publi2003

Cantu-Paz, E. (2000). *Efficient and Accurate Parallel Genetic Algorithms,* Springer, 0792372212, Chicago IL.

Chrabakh, W. & Wolski, R. (2006). GridSAT: Design and Implementation of a

Computational Grid Application. *Journal of Grid Computing,* Vol. 4, No. 2, (June 2006) pp. 177-193. 1570-7873

Chunlin, L. & Layuan, L. (2007). Utility Based Multiple QoS Guaranteed Resource Scheduling Optimization in Grid Computing. *Proceeding of the International Conference on Computing: Theory and Applications, 2007. ICCTA apos;07,* pp 165-169, India, October 2007, Kolkata.

Cox, S.; Chen, L.; Campobasso, S.; Duta, M.; Eres, M.; Giles, M.; Goble, C.; Jiao, Z.; Keane, A.; Pound, G.; Roberts, A.; Shadbolt, N.; Tao, F.; Wason, J. & Xu, F. (2002). *Grid Enabled Optimization and Design Search (GEODISE),* Technical report, www.geodise.org

Czyzck, J; Mesnier, M. & More, J. (1998). The NEOS Server. *IEEE Journal on Computational Science and Engineering,* Vol. 5, No. 3, (May 1998) pp. 68-75.

Daniel, M. & Emiliano, C. (2004). Quality of Service Aspects and Metrics in Grid Computing. *Proceeding of Computer Measurement Group Conference,* pp 102-111, USA, December 2004, Las Vegas, NV.

Foster, I. & Kesselman, C. (1999). *The Grid : Blueprint for a New Computing Infrastructure,* Morgan Kaufmann Publishers, 1558604758, Chicago IL.

Foster, I. (2002). What is the Grid? A three point checklist. *GRIDtoday,* Vol. 1, No. 6, (February 2002)

Foster, I.; Kishimoto, H..; Sava, A.; Berry, D.; Djaoui, A.; Grimshaw, A.; Horn, B.; Maciel, F.; Siebenlist, F.; Subramaniam, R.; Treadwell, J. & Reich, J. (2005). *The Open Grid Services Architecutre Version 1.0,* GGF informational document Global Grid forum, www.globalgridforum.org

Foster, I. (2006). Globus toolkit version 4: Software for service oriented systems, *Proceedings of IFIP International Conference on Network and Parallel Computing,*, pp. 2-13, Japan, October 2006, Sprinder-Verlag LINCS 3779, Tokyo

Frey, J; Tannenbaum, T.; Foster, I.; Livny, M. & Tuecke, S. (2001). Condor-G: A Computation Management Agent for Multi-institutional Grids, *Proceedings of 10th IEEE symposium on High Performance Distributed Computing*, pp. 7-19, USA, August 2001, Morgan Kaufmann Publishers, San Francisco, California

Glankwamdee, V. & Linderoth, J. (2006). MW: A Software Framework for Combinatorial Optimization on Computational Grids, In: *Parallel Combinatorial Optimization,* Talbi, E., pp. 239-256, Wiley-Interscience, 0471721018

Ishikawa, Y.; Kaneo, Y.; Edamoto, M.; Okazaki, F.; Koie, H.; Takano, R.; Kudoh, T. & Kodama. Y., (2005). Overview of GridMPI Version 1.0, *Proceedings of SWoPP'05*, pp. 116-127, Japan, October, Tokyo

Larson, S.; Snow, C. & Pande, V. (2003). *Folding@Home and Genome@Home: Using Distributed Computing to Tackle Previously Intractable Problems in Computational Biology*, R. Grant, Horizon Press

Lim, D.; Ong, Y.; Jin, Y.; Sendhoff, B. & Lee, S. (2007). Efficient Hierarchical Parallel Genetic Algorithms using Grid Computing. *Future Generation Computational systems,* Vol. 4, No. 23, (May 2007) pp. 658-670.

Munawar, A.; Wahib, M.; Munetomo, M. & Akama, K. (2007). Optimization Problem Solving Framework Employing GAs with Linkage Identification over a Grid Environment, *Proceedings of CEC2007: IEEE Congress on Evolutionary Computation*, pp. 3659-3661, Singapore, September 2007

Munawar, A.; Wahib, M.; Munetomo, M. & Akama, K. (To Appear). Parallel GEAs with Linkage Analysis over Grid, In: *Linkage in Genetic and Evolutionary Algorithms*, Springer.

Novotny, J.; Russell, M. & Wehrens, O. (2004). Gridsphere: A Portal Framework for Building Collaborations, *Concurrent Computing: Practices and Exercises,* Vol. 5, No. 16, (June 2004) pp. 503-513

Symour, K.; Nakada, H.; Matsuoka, S.; Dongarra, J; Lee, C. & Casanova, H. (2002). Overview of GridRPC: A Remote Procedure Call API for Grid Computing, *Proceedings of 3rd International Workshop of Grid Computing*, pp. 274-278, USA, November 2002, Morgan Kaufmann Publishers, Baltimore, Maryland

Takemiya, H.; Tanaka, Y.; Sekiguchi, S.; Ogata, S.; Kalia, R.; Nakano, A. & Vashishta, P. (2006). Sustainable Adaptive Grid Supercomputing: Multiscale Simulation of Semiconductor Processing Across the Pacific, *Proceedings of the 2006 ACM/IEEE conference on Super Computing; SC'06*, pp. 106-118, USA, November 2006, Morgan Kaufmann Publishers, New York, NY

Tanaka, Y.; Nakada, H.; Sekiguchi, S.; Suzumura, T. & Matsuoka, S. (2003). Ninf-G: A Reference Implementation of RPC based Programming Middleware for Grid Computing, *Journal of Grid Computing,* Vol. 3, No. 7, (June 2003) pp. 41-51

Wahib, M.; Munawar, A.; Munetomo, M. & Akama, K. (2007). MHGrid: Towards an Ideal Optimization Environment for Global Optimization Problems using Grid Computing, *Proceedings of Parallel and Distributed Computing, Applications and Technologies; PDCAT2007*, pp. 217-220, Australia, December 2007, Morgan Kaufmann Publishers, Adelaide

Weise, T. (2007). *Global Optimization Techniques and Genetic Programming Applied to Distributed Computing,* Thomas Weise, Online as e-book.

Wolpert, H. & Macready, G. (1995). *No Free Lunch Theorems for Search,* Technical report, SFI-TR-95-02-010 Santa Fe, NM

Advances in Evolutionary Algorithms

Edited by Xiong Zhihui

ISBN 978-953-7619-11-4
Hard cover, 284 pages
Publisher InTech
Published online 01, November, 2008
Published in print edition November, 2008

With the recent trends towards massive data sets and significant computational power, combined with evolutionary algorithmic advances evolutionary computation is becoming much more relevant to practice. Aim of the book is to present recent improvements, innovative ideas and concepts in a part of a huge EA field.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Mohamed Wahib, Asim Munawar, Masaharu Munetomo and Kiyoshi Akama (2008). EA-based Problem Solving Environment over the GRID, Advances in Evolutionary Algorithms, Xiong Zhihui (Ed.), ISBN: 978-953-7619-11-4, InTech, Available from: http://www.intechopen.com/books/advances_in_evolutionary_algorithms/ea-based_problem_solving_environment_over_the_grid

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821