Hindawi Publishing Corporation
Applied Computational Intelligence and Soft Computing
Volume 2010, Article ID 505194, 10 pages
doi:10.1155/2010/505194

Research Article

A Distributed Bio-Inspired Method for Multisite Grid Mapping

I. De Falco,[1] A. Della Cioppa,[2] U. Scafuri,[1] and E. Tarantino[1]

[1] *Institute of High Performance Computing and Networking, National Research Council of Italy, Via P. Castellino 111, 80131 Naples, Italy*
[2] *Natural Computation Laboratory, DIIIE, University of Salerno, Via Ponte don Melillo 1, 84084 Fisciano (SA), Italy*

Correspondence should be addressed to I. De Falco, ivanoe.defalco@na.icar.cnr.it

Received 31 July 2009; Revised 8 January 2010; Accepted 20 March 2010

Academic Editor: Chuan-Kang Ting

Computational grids assemble multisite and multiowner resources and represent the most promising solutions for processing distributed computationally intensive applications, each composed by a collection of communicating tasks. The execution of an application on a grid presumes three successive steps: the localization of the available resources together with their characteristics and status; the mapping which selects the resources that, during the estimated running time, better support this execution and, at last, the scheduling of the tasks. These operations are very difficult both because the availability and workload of grid resources change dynamically and because, in many cases, multisite mapping must be adopted to exploit all the possible benefits. As the mapping problem in parallel systems, already known as NP-complete, becomes even harder in distributed heterogeneous environments as in grids, evolutionary techniques can be adopted to find near-optimal solutions. In this paper an effective and efficient multisite mapping, based on a distributed Differential Evolution algorithm, is proposed. The aim is to minimize the time required to complete the execution of the application, selecting from among all the potential ones the solution which reduces the use of the grid resources. The proposed mapper is tested on different scenarios.

1. Introduction

A grid [1] is a decentralized heterogeneous multisite system which aggregates geographically dispersed and multiowner resources (CPUs, storage system, network bandwidth, etc.). From user's perspective, a grid is a collaborative computationally intensive problem-solving environment in which users execute their distributed jobs. Each job, made up of a collection of separate cooperating and communicating tasks, can be processed on the available grid resources without user's knowledge on where they are or even who owns them.

It is noted that the execution times of distributed applications and the throughput of parallel multicomputer systems are heavily influenced by the task mapping and scheduling which, in case of large and disparate set of grid resources, become still more impractical even for experienced users. In fact, grid resources have a limited capacity and their characteristics vary dynamically as jobs change and randomly arrive. Since, in many cases, single-site resources could be inefficient for meeting job requirements, multisite mapping must be adopted to provide all the possible benefits. Obviously, this latter concern further complicates the mapping operation.

On the basis of these considerations, it is clear that an efficient mapping is possible only if it is supported by a fully automated grid task scheduler [2].

Naturally when a new job is submitted for execution on a grid, the dynamical availability and the pertaining workload of grid resources imply that, to select the appropriate resources, the grid task scheduler has to know number and status of the resources available in that moment. Hence such a scheduler, hereinafter referred to as Metascheduler, is not simply limited to the mapping operation, but must act in three successive phases: resource discovery, mapping or task/node allocation and job scheduling [3].

The resource discovery phase, which has to determine the amount, type, and status of the available resources, can obtain this information either by specific tables based on statistical estimations in a particular time span or gathered tracking periodically and forecasting dynamically resource conditions [4, 5]. For example, in Globus Toolkit [6], which is the middleware used for building grids, global information gathering is performed by the Grid Index Information Service which contacts the Grid Resource Information Service to acquire local information [7].

In the mapping phase, the Metascheduler has to select, in accordance with possible user requirements, the nodes which opportunely match the application needs with the available grid resources.

Finally, in the last phase the Metascheduler establishes the schedule timing of the tasks on the nodes. To have that all the tasks will be promptly coscheduled, our Metascheduler selects, in line with job requirements, resources conditions and knowledge of the different local scheduling policies, only the nodes, even belonging to different sites, which in that moment are able to coschedule the tasks assigned to them. This last assumption avoids to perform the job scheduling phase. It is noted that, if locally supported, an alternative to attain the coscheduling could be to make advance reservations. However, this approach, which requires that resource owners have a good planning on their own tasks, presents difficulties to be employed in a shared environment.

As concerns the resource discovery phase, the Metascheduler here implemented determines the available nodes considering historical information pertaining the workload as a function of time, and the characteristics of each node by using specific tables.

In this paper, the attention is focused only on the mapping phase. Since mapping algorithms for traditional parallel and distributed systems, which usually run on homogeneous and dedicated resources, for example, computer clusters, cannot work adequately in heterogeneous environments [1], other approaches have been proposed to cope with different issues of the problem [8–12].

Generally the allocation of jobs to resources is performed respecting one or more optimization criteria like minimal makespan, minimal cost of assigned resources, or maximal throughput and so on. Here, in contrast to the classical approach [13–15] which takes into account the grid user's point of view and aims at minimizing the completion time of the application task, we deal with the multisite mapping problem from the grid manager's point of view. Thus, our aim is to find the solution which minimizes execution time and communication delays, and optimizes resource utilization using at the minimum the grid resources it has to exploit at the most.

Unfortunately, the mapping problem, already known as NP-complete for parallel systems [16, 17], becomes even more difficult in a distributed heterogeneous environment as in grid systems. Moreover, in the future, grids will be characterized by an increasing number of sites and nodes per site, so as to meet the ever growing computational demands of large and diverse groups of tasks. Hence, it has seemed natural to devote attention to the development of mapping tools based on heuristic optimization techniques, as, for example, evolutionary algorithms. Several evolutionary-based techniques have been used to face the task allocation in a heterogeneous or grid environment [10, 13–15, 18–22].

Within this paper, a distributed version of Differential Evolution (DE) [23, 24] approach is proposed. This technique is attractive because it requires few control parameters, it is relatively easy to implement, effective and efficient in solving practical engineering problems. Unlike all the other existing evolutionary approaches which simply search for mapping the job onto just one site [21], we deal with a multisite approach.

Then, differently from other methods which face the problem of mapping in a heterogeneous environment for applications developed according to a specific paradigm, as, for example, the master/slave model in [25, 26], we do not make hypotheses about the application graph. Moreover, as a further distinctive issue with respect to other approaches in literature [12], we consider the nodes making up the sites as the lowest computational unit taking into account its actual load.

Paper structure is as follows: Section 2 illustrates our evolutionary approach to the mapping problem. Section 3 describes the distributed DE algorithm, while Section 4 reports on the test problems faced and outlines the results achieved. Lastly, Section 5 contains conclusions and future works.

2. Differential Evolution for Mapping

2.1. The Technique. Differential Evolution is a stochastic and reliable evolutionary optimization strategy which presents noticeable performance in optimizing a wide variety of multidimensional and multimodal objective functions in terms of final accuracy and robustness, and overcomes many of the already existing stochastic and direct search global optimization techniques [27–29]. In particular, given a minimization problem with q real parameters, DE faces it starting with a population of $\mathcal{M}$ randomly chosen solution vectors each made up by q real values. At each generation, new vectors are generated by a combination of vectors randomly chosen from the current population. The outcoming vectors are then mixed with a predetermined target vector. This operation is called recombination and produces the trial vector. Many different transformation schemes have been defined by the inventors to produce the candidate trial vector [23, 24]. To explicit the strategy they established a sensible naming-convention for each DE technique with a string like DE/*x*/*y*/*z*. In it, DE stands for Differential Evolution, x is a string which denotes the vector to be perturbed (*best* = the best individual in current population, *rand* = a randomly chosen one, *rand-to-best* = a random one, but the current best participates in the perturbation too), y is the number of difference vectors taken for perturbation of x (either 1 or 2), while z is the crossover method (*exp* = exponential, *bin* = binomial). We have chosen the *DE/rand/1/bin* strategy throughout our investigation. In this model, a random individual is perturbed by using one difference vector and by applying binomial crossover. More specifically, for the generic ith individual in the current population three integer numbers r_1, r_2, and r_3 in $\{1, \ldots, \mathcal{M}\}$ differing one another and different from i are randomly generated. Furthermore, another integer number s in the set $\{1, \ldots, q\}$ is randomly chosen. Then, starting from the ith individual a new trial one i' is generated whose generic jth component is given by

$$x_{i'_j} = x_{r_{3_j}} + F \cdot \left(x_{r_{1_j}} - x_{r_{2_j}} \right) \tag{1}$$

provided that either a randomly generated real number ρ in $[0.0, 1.0]$ is lower than a value CR (parameter of the DE, in the same range as ρ) or the position j under account is exactly s. If neither is verified, then a simple copy takes place: $x_{i'_j} = x_{i_j}$. F is a real and constant factor which controls the magnitude of the differential variation $(x_{r_{1j}} - x_{r_{2j}})$, and is a parameter of the algorithm.

This new trial individual $x_{i'}$ is compared against the ith individual in the current population and is inserted in the next population if fitter. This basic scheme is repeated for a maximum number of generations g.

2.2. Definitions and Assumptions. In this work, we refer to a grid as a system constituted by one or more sites, each containing a set of nodes, while to a job as a set of distributed tasks, each with various requirements [8, 30–33]. In absence of virtual or dedicated links, sites generally communicate by means of internet infrastructure.

In each site, single node and multinode systems are present. With single node we intend a standalone computational system provided by one or more processors and one or more links, while with multinode we refer to a parallel system. Moreover, we assume that the node is the elementary computation unit and that the proposed mapping is task/node. Each node executes the tasks arranged in two distinct queues: the local queue (L_q) for the locally submitted tasks and the remote queue (R_q) for those presented via grid. The tasks in R_q can be executed only if there are not ready tasks in L_q. While the tasks in L_q will be scheduled on the basis of the locally established policy, a First-Come-First-Served (FCFS) strategy with priority must be adopted for those in R_q. According to this scheduling policy, to perform the mapping process both the current local and grid workloads are taken into account.

To focus the mapping problem in the premised grid we need information on the number and on the status of both accessible and demanded resources. Consequently, we assume to have a grid application subdivided into P tasks (demanded resources) to be mapped on n nodes (accessible resources) with $n \in \{1, \dots, N\}$, where P is fixed *a priori* and N is the number of grid nodes.

We have to know node capacities (the number of instructions computed per time unit), network bandwidth and load of each grid node in a given time span. In fact, the available power of each node varies over time due to the load by the original users in shared-resource computing. In particular, we need to know *a priori* the number of instructions α_i computed per time unit on node i. Furthermore, we assume to have cognition of the communication bandwidth β_{ij} between any couple of nodes i and j. It should be noted that β_{ij} is the generic element of an $N \times N$ symmetric matrix $\boldsymbol{\beta}$ with very high values on the main diagonal, that is, β_{ii} is the bandwidth between two tasks on the same node. We suppose that this information is contained in tables based on statistical estimations in a particular time span.

In general, grids address nondedicated resources since they have their own local workloads. This affects the availability of local performance. Thus we must consider these load conditions to evaluate the expected computation time. There exist several prediction models to face the challenge of nondedicated resources [34, 35]. For example, as attains the computational power, we suppose to know the average load $\ell_i(\Delta t)$ of the node i at a given time span Δt with $\ell_i(\Delta t) \in [0.0, 1.0]$, where 0.0 means a node completely discharged and 1.0 a node locally loaded at 100%. Hence $(1 - \ell_i(\Delta t)) \cdot \alpha_i$ represents the fraction of power at node i available for executing grid tasks.

As an example, if the resource is a computational node, the conditions collected could be the fraction of CPU which can be destined to the execution of the newly started processes, and the fraction of bandwidths which could be different in conformity with the remote hosts involved in the communication.

As regards the resources requested by the job, we assume to know for each task k the respective number of instructions γ_k to be executed and the number of communications ψ_{km} between the kth and the mth task for all $m \neq k$. Obviously, ψ_{km} is the generic element of a $P \times P$ symmetric matrix $\boldsymbol{\psi}$ with all null elements on the main diagonal.

All this information can be obtained either by a static program analysis, or by using smart compilers or by other emerging tools which automatically generate them. For example, the Globus Toolkit includes the Resource Specification Language which constitutes an XML format to define application requirements [7].

2.3. Encoding. In general, any mapping solution should be represented by a vector $\boldsymbol{\mu}$ of P integers in the set $\{1, \dots, N\}$. To obtain $\boldsymbol{\mu}$, the real values provided by DE in the range $[1, N + 1[$ are truncated before evaluation. The truncated value $\lfloor \mu_i \rfloor$ denotes the node onto which the task i is mapped.

As long as the mapping is considered by characterizing the tasks by means of their computational needs γ_k only, this is an NP-complete optimization problem, in which the allocation of a task does not affect that of the other ones, unless one attempts to load more tasks on the same node. If, instead, also communications ψ_{km} are taken into account, the mapping problem becomes by far more complicate. In fact, the allocation of a task on a given node can cause that the optimal mapping needs that also other tasks must be allocated on the same node or in the same site, so as to decrease their communication times and thus their execution times, taking advantage of the higher communication bandwidths existing within any site compared to those between sites.

Such a problem is a typical example of *epistasis*, that is, a situation in which the value taken on by a variable influences those of other variables. This situation is also *deceptive*, since a solution $\boldsymbol{\mu}_1$ can be transformed into another with better fitness $\boldsymbol{\mu}_2$ only by passing through intermediate solutions, worse than both $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$, which would be discarded. To overcome this problem we have introduced a new operator, named *site mutation*, applied with a probability p_m any time a new individual must be generated. When this mutation is to be carried out, a position in the current solution $\boldsymbol{\mu}$ is randomly chosen. Let us suppose its value refers to a node belonging to a site C_i. This value is equiprobabilistically modified into another one which is related to a node of another cluster, say C_j. Then, any other task assigned to C_i in the current solution is let randomly migrate to a node of C_j by inserting into the related position a random value within the bounds for C_j. If *site mutation* does not take place, the classical transformations typical of DE must be applied.

2.4. Fitness. The two major parties in grid computing, namely, resource consumers who submit various applications, and resources providers who share their resources, usually have different motivations when they join the grid. Currently, most of objective functions in grid computing are inherited from traditional parallel and distributed systems. As attains applications, grid users and providers of resources can have different demands to satisfy. As an example users could be interested in the total cost to run their application, while providers could pay more attention to the throughput of their resources in a particular time interval. Thus objective functions can meet different goals.

In our case, the fitness function calculates the summation of the execution times of the set of all the tasks on the basis of the specific mapping solution.

Use of Resources. Denoting τ_{ij}^{comp} and τ_{ij}^{comm}, respectively, the computation and the communication times requested to execute the task i on the node j it is assigned to the generic element of the execution time matrix $\boldsymbol{\tau}$ is computed as

$$\tau_{ij} = \tau_{ij}^{\text{comp}} + \tau_{ij}^{\text{comm}}. \qquad (2)$$

In other words, τ_{ij} is the total time needed to execute task i on node j and is evaluated on the basis of the computation power and of the bandwidth which remain available once deducted the local workload. Let τ_j^s be the summation on all the tasks assigned to the jth node for the current mapping. This value is the time spent by node j in executing computations and communications of all the tasks assigned to it by the proposed solution. Of course, it does not consider the time intervals in which these tasks are idle waiting for communicating, so that tasks dependency does not influence the results of the mapping proposed. Clearly, τ_j^s is equal to zero for all the nodes j not included in the vector $\boldsymbol{\mu}$, that is, all the nodes which do not have assigned tasks.

Considering that all the tasks are coscheduled, the time required to complete the application execution is given by the maximum value among all the τ_j^s. Then, the fitness function is

$$\Phi(\boldsymbol{\mu}) = \max_{j\in\{1,\ldots,N\}} \left\{\tau_j^s\right\}. \qquad (3)$$

The goal of the evolutionary algorithm is to search for the smallest fitness value among these maxima, that is, to find the mapping which uses at the minimum, in terms of time, the grid resource it has to exploit at the most.

If during the DE generation of new individuals the offspring has the same fitness value as its parent, then it is selected the individual for which

$$\Phi^*(\boldsymbol{\mu}) = \sum_{j=1}^{N} \tau_j^s \qquad (4)$$

is smaller. This quantity represents the total amount of time dedicated by the grid to the execution of the job. Obviously, such a mechanism takes place also for the selection of the best individual in the population. This choice aims at meeting the requirements of resource providers, favouring mappings which exploit best the shared resources.

It should be noted that, though the fitness values of the proposed mapping are not related to the completion time of the application, Φ and Φ^* can be seen, respectively, as the lower and the upper bound of the job execution time.

The pseudocode of our DE for mapping is shown in the following Algorithm 1.

3. The Distributed Algorithm

Our Distributed DE (DDE) algorithm is based on the classical coarse-grained approach to Evolutionary Algorithms, widely known in literature [36]. It consists in a locally linked strategy, the *stepping stone-model* [37], in which each DE instance is connected to d instances only. If, for example, we arrange them as a folded torus, then each DE instance has exactly four neighbouring subpopulations as shown in Figure 1, where the generic DE algorithm is shown in black, and its neighbouring subpopulations are indicated in grey. The subpopulation under examination is, thus, “isolated” from all the other ones, shown in white, and it can communicate with them in an indirect way only, through the grey ones. Moreover every M_I generations (*Migration Interval*), neighbouring subpopulations are allowed to exchange individuals. The percentage of individuals each subpopulation sends to its neighbours is called *Migration Rate* (M_R).

A design decision is the quality of the elements to be sent; they might be the best ones or randomly chosen ones. Another decision must be taken about the received individuals; they might anyway replace the worst individuals in the population or substitute them only if better, or they might finally replace any individual (apart from the very best ones, of course). It is known from literature that the number of individuals sent should not be high, nor should the exchange frequency, otherwise the subsearch in a processor might be very disturbed by these continuously entering elements which could even be seen as noise [36]. This mechanism allows to achieve both *exploitation* and *exploration*, which are basic features for a good search. Exploration means to wander through the search space so as to consider also very different situations, looking for the most hopeful (favourable) areas to be intensively sampled. Exploitation means that one area is thoroughly examined, so that we can be confident in being able to state whether this area is promising. By making use of this approach, good solutions will spread within the network with successive diffusions, so more and more processors will try to sample that area (exploitation), and, on the other hand, there will exist at the same time clusters of processors which will investigate different subareas of the search space.

Within this general framework, we have implemented a distributed version for DE, which consists of a set of classical DE schemes, running in parallel, assigned to different

```
begin
randomly initialize population $X = (x_1, \dots, x_{\mathcal{M}})$
evaluate fitness $\Phi$ for all the individuals $x_i$
while (maximal number of generations $g$ is not reached) do
    begin
     for $i = 1$ to $\mathcal{M}$ do
        begin
          choose a random real number $p_{sm} \in [0.0, 1.0]$
          if $(p_{sm} < p_m)$
           apply site mutation
        else
          begin
            choose three integers $r_1, r_2$ and $r_3 \in \{1, \dots, \mathcal{M}\}$, with $r_1 \neq r_2 \neq r_3 \neq i$
            choose an integer number $s$ in $\{1, \dots, q\}$
            for $j = 1$ to $q$ do
                begin
                  choose a random real number $\rho \in [0.0, 1.0]$
                  if $((\rho < CR)$ OR $(j = s))$
                      $x_{i'_j} = x_{r_{3j}} + F \cdot (x_{r_{1j}} - x_{r_{2j}})$
                  else
                    $x_{i'_j} = x_{i_j}$
                end
            if $\Phi(x_{i'}) \leq \Phi(x_i)$
                insert $x_{i'}$ in the new population
            else
              insert $x_i$ in the new population
          end
       end
    end
end
```

Algorithm 1

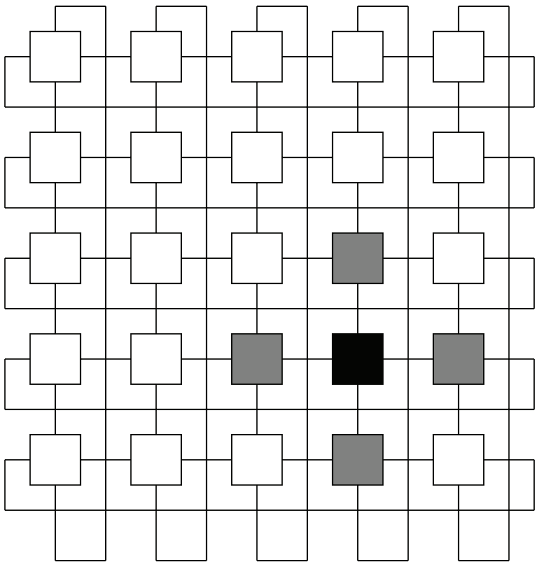


Figure 1: The folded torus topology.

processing elements arranged in a folded torus topology, plus a master. The master process acts as an interface to the user: it simply collects the current local best solutions of the "slave" processes and saves the best among them at each generation. Besides, this latter is compared with the overall best found so far and, if fitter, becomes the new overall best and is shown to the user.

4. Experiments and Findings

Before effecting any kind of experiment the structure of the available resources and the features of the machines belonging to each site must be known. Generally, sites of a grid architecture have different number of systems (parallel machines, clusters, supercomputers, dedicated systems, etc.) with various characteristics and performance. To perform a simulation, we assume to have a grid composed of $N = 58$ nodes subdivided into five sites denoted with A, B, C, D, and E with 16, 8, 8, 10, and 16 nodes, respectively. This grid structure is outlined in Figure 2 while an example of the site B, made up by four single nodes and a four-node cluster, is shown in Figure 3.

Hereinafter, we will denote the nodes by means of the numbers shown in Figure 2, so that, for example, 20 is the fourth node in site B, while 37 is the fifth node in site D.

Without loss of generality, we suppose that all the nodes belonging to the same site have the same power α expressed in terms of millions of instructions per second (MIPS) as shown in Table 1.

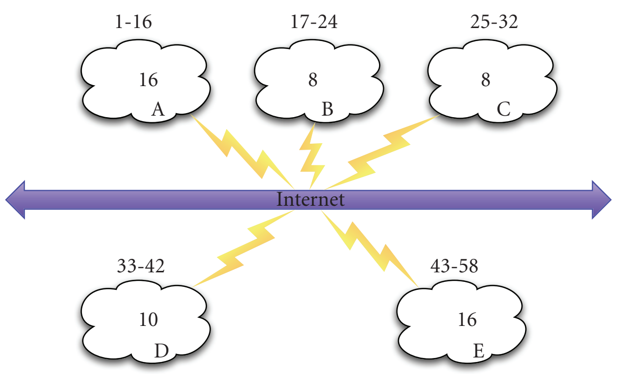


Figure 2: The grid architecture.

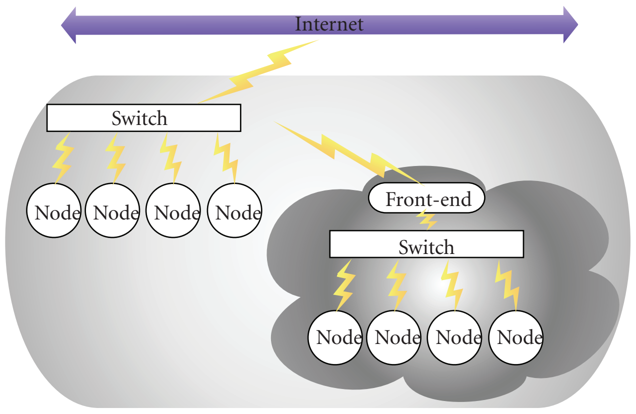


Figure 3: An example of site B.

Table 1: Power of nodes for each site expressed in MIPS.

Sites	A	B	C	D	E
α	500	900	2000	1700	700

For the sake of simplicity, we have hypothesized for each node three communication bands. The first is the bandwidth β_{ii} available when tasks are mapped on the same node (*intranode communication*), the second is the bandwidth β_{ij} between the nodes i and j belonging to the same site (*intrasite communication*), and the third is the bandwidth β_{ij} when the nodes i and j belong to different sites (*intersite communication*). Besides, we presume that all the β_{ii}s have the same very high value (10 Gbit/s) so as to yield the related communication time negligible with respect to intrasite and intersite communications.

For each site, the bandwidth of the output link is supposed equal to that of the input link. In our case, the intersite bandwidths are reported, with the addition of the intrasite bandwidths, in Table 2.

Moreover we assume to know the average load of available grid resources for the time span of interest.

A generally accepted set of heterogenous computing benchmarks does not exist and the detection of a representative set of such benchmarks remains a current and unresolved challenge. To evaluate the effectiveness of our DDE-based

Table 2: Intersite and intrasite bandwidths expressed in Mbit/s.

	A	B	C	D	E
A	10				
B	2	100			
C	6	3	1000		
D	5	10	7	800	
E	2	5	6	1	100

approach we have decided to investigate different application tasks with particular attention to both computation-bound and communication-bound tasks as the load of grid nodes varies.

After a very preliminary tuning phase, the parameters of each DDE have been set as follows: $\mathcal{M} = 30$, $g = 1000$, $CR = 0.3$, $F = 2.0$, $p_m = 0.2$, $M_I = 50$, and $M_R = 1$. This set of parameters is left unchanged for all the experiments carried on.

Our DDE can be profitably used for mapping of message passing applications. Here we have used the Message Passing Interface (MPI) [38] which is a widely used standard library which makes the development of grid applications more accessible to programmers with parallel computing skills. Actually, many MPI library implementations, as MPICH-G2 [39], MagPIe [40], MPI_Connect [41], MetaMPICH [42] and so on, allow the execution of MPI programs on groups of multiple machines potentially based on heterogeneous architectures. However, all these libraries require that users must explicitly specify the resources to be used and they may have enormous difficulties to select, at the best, the appropriate resources for their works in grid environments.

The DDE algorithm has been implemented in C language and all the experiments have been effected on a cluster of 17 (1 master and 16 slaves) 1.5 GHz Pentium 4 interconnected by a FastEthernet switch.

For each test problem 20 DDE executions have been carried out, so as to investigate the dependence of the results on the random seed. Each execution has required 13s for a total of 260*s* for each set of experiments. It should be noted that if the situation described at the end of Section 2.4 takes place when comparing the results of the different runs, the same tie-break mechanism is adopted.

Once defined the evolutionary parameters and the grid characteristics, different scenarios must be designed to demonstrate the effectiveness of the approach over a broad range of realistic conditions. To ascertain the degree of optimality, different tests are conceived to allow a simple comparison between a manual calculation and the solution provided by the mapping tool. Note that, for the sake of simplicity, in the experiments reported, we suppose that the local load of a node is constant during all the execution time of the application task allocated to it. Obviously, a variable load would require only a different calculation but it would not invalidate the approach proposed. In the following, we show the mapping results attained for these experiments.

The first experiment has regarded an application of $P = 12$ tasks with $\gamma_k = 90$ Giga Instructions (GI), $\psi_{km} = 0$ for

Table 3: Findings for each experiment.

Exp. no.	1	2	3	4	5	6	7	8
Φ	52.94	52.94	128.57	139.19	180.00	271.73	484.77	128.57
Φ^*	571.76	587.64	2571.42	2744.59	5387.14	8693.28	5817.29	1781.22
n_b	20	15	20	15	3	4	3	20
Φ_{av}	52.94	62.20	128.57	156.64	218.25	298.50	939.97	128.57
σ	0.00	16.46	0.00	31.01	16.48	13.73	255.29	0.00

all $k, m \in \{1, \ldots, P\}$, and $\ell_i(\Delta t) = 0$ for all the nodes. The mapping solution found by our DDE is:

$$\boldsymbol{\mu} = \{25, 26, 27, 28, 29, 30, 31, 32, 41, 42, 35, 36\}. \quad (5)$$

As expected, the mapping procedure has allocated all the tasks on the most powerful available nodes, eight belonging to the site C and four to site D.

In the second experiment, all the parameters remain unchanged except the load. In particular, we have supposed $\ell(\Delta t) = 0.7$ on the two nodes 31 and 32 and $\ell(\Delta t) = 0.5$ on the three nodes 40, 41, and 42. In this hypothesis, the mapping solution found is

$$\boldsymbol{\mu} = \{25, 34, 27, 28, 37, 30, 39, 38, 33, 29, 36, 26\}. \quad (6)$$

As it can be observed the solution again involves the most powerful nodes (six belonging to C and six to D), discarding correctly the loaded nodes in those sites.

In the third experiment, we have $P = 20$ with $\gamma_k = 90$ GI, $\psi_{km} = 0$ for all $k, m \in \{1, \ldots, P\}$ and $\ell(\Delta t) = 0.9$ for all the nodes of the sites B and D, while for the site C we assume $\ell_i(\Delta t) = 0.8$ for $i \in \{25, \ldots, 28\}$ and $\ell_i(\Delta t) = 0.6$ for $i \in \{29, \ldots, 32\}$. The mapping solution discovered by our DDE is

$$\begin{aligned}\boldsymbol{\mu} = \{&43, 44, 45, 46, 47, 29, 49, 50, 51, 52, 53, 54, 55, 56,\\ &57, 58, 48, 30, 31, 32\}.\end{aligned} \quad (7)$$

It is worth noticing that in this load conditions the mapping procedure has chosen once again the most powerful nodes: 4 of C with $\ell_i(\Delta t) = 0.6$ which are those with a minor load and 16 of E.

The same solution has been obtained in the fourth experiment where we have just introduced the communications $\psi_{km} = 10$ Mbit for all $k, m \in \{1, \ldots, P\}$.

In the fifth experiment, we have left unchanged both the load conditions and the number of instructions that each task k has to effect ($\gamma_k = 90$ GI). Simply we have considered $P = 36$ and removed all the communications among the tasks. The allocation is outlined in the following

$$\begin{aligned}\boldsymbol{\mu} = \{&1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,\\ &47, 50, 44, 57, 46, 31, 29, 54, 49, 51, 55, 53, 30,\\ &32, 45, 52, 43, 58, 56, 48\}.\end{aligned} \quad (8)$$

This solution, according to the load conditions, has mapped 16 tasks on the 16 nodes of A, 16 on all the nodes of E, and 4 on the 4 nodes of C which present the lowest load (0.6).

In the sixth experiment, we have merely added a communication $\psi_{km} = 10$ Mbit for all $k, m \in \{1, \ldots, P\}$. The result is:

$$\begin{aligned}\boldsymbol{\mu} = \{&1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 43,\\ &45, 51, 28, 58, 49, 46, 52, 47, 48, 56, 27, 57, 50, 30,\\ &29, 25, 32, 31, 26\}.\end{aligned} \quad (9)$$

Such a solution provides 16 tasks on the 16 nodes of site A, 12 on the site E, and 8 on all the nodes of C. It can be noted that the mapping proposed has selected four nodes of C which are loaded at 0.8, and therefore less powerful than the other discharged nodes of E, to exploit the major bandwidth among nodes allocated on the site C with respect to the intersite bandwidth between C and E.

The influence of the communications is highly evidenced in the successive experiment where, leaving unchanged all the other conditions, the communication ψ_{km} has been set to 100 Mbit for all $k, m \in \{1, \ldots, P\}$. The mapping proposed has allocated all the 36 tasks on the 16 nodes of site E. In fact, the time requested to perform the communications becomes relevant compared to the computation time and thus it is advantageous to allocate more tasks on each node of site E rather than to subdivide them on nodes of different sites. The solution is

$$\begin{aligned}\boldsymbol{\mu} = \{&53, 47, 43, 44, 47, 48, 45, 49, 50, 46, 46, 48, 49, 50, 43,\\ &44, 53, 51, 51, 45, 52, 52, 54, 54, 56, 57, 57, 55, 55, 56,\\ &58, 58, 43, 47, 53, 55\}.\end{aligned} \quad (10)$$

As an example of the behavior shown by our tool, Figure 4 reports the evolution of the best run achieved for this last test. Namely, we depict the best, average and worst fitness values among those sent to the master by the 16 slaves at each generation. Since the initial generation the average, the best and the worst fitness values decrease over generations, and this continues until the end of the run. Every now and then several successive generations take place in which no improving solutions are found, and this results in best, average and worst values becoming more and more similar. Then, a new better solution is found and the three values become quite different. The described behavior implies that good solutions spread only locally among linked

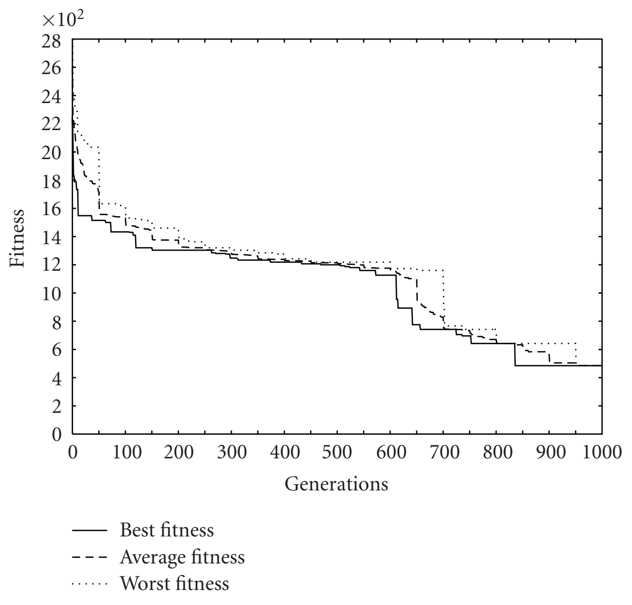


Figure 4: Behavior of fitness as a function of the number of generations for the best run of experiment 7.

subpopulations without causing premature convergence to the same suboptimal solution on all the slaves, which is a positive feature of the system.

The final experiment has attained a job with $P = 36$, $\gamma_k = 90$ GI for $k \in \{1, \dots, 12\}$, $\gamma_k = 9$ GI for $k \in \{13, \dots, 36\}$, $\psi_{km} = 0$ for all $k, m \in \{1, \dots, 24\}$ and $\psi_{km} = 10$ Mbit for all $k, m \in \{25, \dots, 36\}$, while the load conditions are the same of the previous experiment. The mapping found is

$$\boldsymbol{\mu} = \{43, 44, 45, 46, 47, 48, 31, 32, 51, 52, 53, 30, 55, 56, 57, 58, 29, 57, 58, 31, 58, 57, 58, 57, 29, 29, 29, 29, 29, 29, 29, 30, 29, 29, 29, 32\}. \tag{11}$$

From the mapping proposed, it can be observed that 17 tasks are placed on C and 19 are allocated on E. In particular, three of the tasks with $\gamma_k = 90$ GI have been mapped on three nodes of C with $\ell(\Delta t) = 0.6$ (nodes 30, 31 and 32) and the remaining 9 with the same computational requirements on 9 nodes of site E, while the fourth node of C with $\ell(\Delta t) = 0.6$ (node 29) has been used to allocate 10 tasks each with $\gamma_k = 9$ GI and $\psi_{km} = 10$ Mbit for all $(k, m) \in \{25, \dots, 36\}$.

In Table 3, for each experiment (Exp. no) the best fitness values for Φ and Φ^* are outlined and, for all the 20 runs, the number of occurrences (n_b) of the best result, the average fitness values (Φ_{av}), and the standard deviations σ are shown.

The tests performed have evidenced a high degree of efficiency of the proposed model in terms of both goodness of the solutions provided and convergence times. In fact, efficient solutions have been quickly provided independently of work conditions (heterogenous nodes diverse in terms of number, type, and load) and kind of jobs (computation or communication bound).

5. Conclusions and Future Works

This paper faces the multisite mapping problem in a grid environment by means of Differential Evolution. In particular, the goal is the minimization of the degree of use of the grid resources by the proposed mapping. The results show that a Distributed Differential Evolution algorithm is a viable approach to the important problem of grid resource allocation. A comparison with other methods is impossible at the moment due to the lack of approaches dealing with this problem in the same operating conditions as ours. In fact, some of these algorithms, such as Min-min, Max-min, and XSuffrage [12], are related to independent tasks and their performances are affected in heterogenous environments. In case of dependent tasks, the classical approaches apply the popular model of Direct Acyclic Graph (DAG) differently from our approach in which no assumptions are made about the communications among the processes since we have hypothesized tasks coscheduling.

Future works will include an investigation of the different DE schemes, together with a wide tuning phase for parameter sets, to experiment their effectiveness in facing the problem under exam.

A dynamic measure of the load of grid nodes will be examined. Furthermore, we have supposed that the cost per MIPS and Mbit/s is the same for all the grid nodes. Since nodes with different features have different costs, in the future these costs will be added to the other parameters considered in the mapping strategy.

Finally, since Quality of Service (QoS) assumes an important role for many grid applications, we intend to enrich our tool so it will be able to manage multiple QoS requirements as those on performance, reliability, bandwidth, cost, response time, and so on.

References

[1] F. Berman, "High-performance schedulers," in *The Grid: Blueprint for a Future Computing Infrastructure*, I. Foster and C. Kesselman, Eds., pp. 279–307, Morgan Kaufmann, San Francisco, Calif, USA, 1998.

[2] G. Mateescu, "Quality of service on the grid via metascheduling with resource co-scheduling and co-reservation," *International Journal of High Performance Computing Applications*, vol. 17, no. 3, pp. 209–218, 2003.

[3] J. M. Schopf, "Ten actions when grid scheduling: the user as a grid scheduler," in *Grid Resource Management: State of the Art and Future Trends*, pp. 15–23, Kluwer Academic Publishers, Norwell, Mass, USA, 2004.

[4] S. Fitzgerald, I. Foster, C. Kesselman, G. von Laszewski, W. Smith, and S. Tuecke, "A directory service for configuring high-performance distributed computations," in *Proceedings of the 6th IEEE International Symposium on High Performance Distributed Computing*, pp. 365–375, IEEE Computer Society, Portland, Ore, USA, August 1997.

[5] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman, "Grid information services for distributed resource sharing," in *Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing*, pp. 181–194, San Francisco, Calif, USA, August 2001.

[6] I. Foster, “Globus toolkit version 4: software for service-oriented systems,” in *Proceedings of IFIP International Conference on Network and Parallel Computing (NPC ’05)*, vol. 3779 of *Lecture Notes in Computer Science*, pp. 2–13, Beijing, China, November-December 2005.

[7] L. Adzigogov, J. Soldatos, and L. Polymenakos, “EMPEROR: an OGSA grid meta-scheduler based on dynamic resource predictions,” *Journal of Grid Computing*, vol. 3, no. 1-2, pp. 19–37, 2005.

[8] R. F. Freund, “Optimal selection theory for super concurrency,” in *Supercomputing*, pp. 699–703, IEEE Computer Society, Reno, Nev, USA, 1989.

[9] M. M. Eshaghian and M. E. Shaaban, “Cluster-m programming paradigm,” *International Journal of High Speed Computing*, vol. 6, no. 2, pp. 287–309, 1994.

[10] T. D. Braun, H. J. Siegel, N. Beck, et al., “A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems,” *Journal of Parallel and Distributed Computing*, vol. 61, no. 6, pp. 810–837, 2001.

[11] K.-H. Kim and S.-R. Han, “Mapping cooperating grid applications by affinity for resource characteristics,” in *Proceedings of the 13th International Conference on AIS*, vol. 3397 of *Lecture Notes in Artificial Intelligence*, pp. 313–322, 2005.

[12] F. Dong and S. G. Akl, “Scheduling algorithms for grid computing: state of the art and open problems,” Tech. Rep. 2006-504, School of Computing, Queens University, Kingston, Canada, 2006.

[13] H. Singh and A. Youssef, “Mapping and scheduling heterogeneous task graphs using genetic algorithms,” in *Proceedings of Heterogeneous Computing Workshop*, pp. 86–97, IEEE Computer Society, Honolulu, Hawaii, USA, 1996.

[14] P. Shroff, D. W. Watson, N. S. Flan, and R. F. Freund, “Genetic simulated annealing for scheduling data-dependent tasks in heterogeneous environments,” in *Proceedings of Heterogeneous Computing Workshop*, pp. 98–104, IEEE Computer Society, Honolulu, Hawaii, USA, 1996.

[15] L. Wang, H. J. Siegel, V. P. Roychowdhury, and A. A. MacIejewski, “Task matching and scheduling in heterogeneous computing environments using a genetic-algorithm-based approach,” *Journal of Parallel and Distributed Computing*, vol. 47, no. 1, pp. 8–22, 1997.

[16] O. H. Ibarra and C. E. Kim, “Heuristic algorithms for scheduling independent tasks on non identical processors,” *Journal of Association for Computing Machinery*, vol. 24, no. 2, pp. 280–289, 1977.

[17] D. Fernandez-Baca, “Allocating modules to processors in a distributed system,” *IEEE Transactions on Software Engineering*, vol. 15, no. 11, pp. 1427–1436, 1989.

[18] Y.-K. Kwok and I. Ahmad, “Efficient scheduling of arbitrary task graphs to multiprocessors using a parallel genetic algorithm,” *Journal of Parallel and Distributed Computing*, vol. 47, no. 1, pp. 58–77, 1997.

[19] A. Abraham, R. Buyya, and B. Nath, “Nature’s heuristics for scheduling jobs on computational grids,” in *Proceedings of the 8th International Conference on Adavanced Computing and Communication*, pp. 45–52, 2000.

[20] S. Kim and J. B. Weissman, “A genetic algorithm based approach for scheduling decomposable data grid applications,” in *Proceedings of the International Conference on Parallel Processing (ICPP ’04)*, pp. 406–413, Montreal, Canada, August 2004.

[21] A. Bose, B. Wickman, and C. Wood, “MARS: a metascheduler for distributed resources in campus grids,” in *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing (GRID ’04)*, pp. 110–118, IEEE Computer Society, Pittsburgh, Pa, USA, November 2004.

[22] S. Song, Y.-K. Kwok, and K. Hwang, “Security-driven heuristics and a fast genetic algorithm for trusted grid job scheduling,” in *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS ’05)*, p. 65, Denver, Colo, USA, April 2005.

[23] K. Price and R. Storn, “Differential evolution,” *Dr. Dobb’s Journal*, vol. 22, no. 4, pp. 18–24, 1997.

[24] R. Storn and K. Price, “Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces,” *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[25] G. Shao, F. Berman, and R. Wolski, “Master/slave computing on the grid,” in *Proceedings of the 9th Heterogeneous Computing Workshop*, pp. 3–16, IEEE Computer Society, Cancun, Mexico, 2000.

[26] N. Ranaldo and E. Zimeo, “An economy-driven mapping heuristic for hierarchical master-slave applications in grid systems,” in *Proceedings of the 20th International Parallel and Distributed Processing Symposium (IPDPS ’06)*, Rhodes Island, Greece, 2006.

[27] S. Das, A. Abraham, and A. Konar, “Particle swarm optimization and differential evolution algorithms: technical analysis, applications and hybridization perspectives,” in *Studies in Computational Intelligence*, Y. Liu, et al., Ed., vol. 116, pp. 1–38, Springer, Berlin, Germany, 2008.

[28] A. Nobakhti and H. Wang, “A simple self-adaptive differential evolution algorithm with application on the ALSTOM gasifier,” *Applied Soft Computing Journal*, vol. 8, no. 1, pp. 350–370, 2008.

[29] S. Das, A. Abraham, U. K. Chakraborty, and A. Konar, “Differential evolution using a neighborhood-based mutation operator,” *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 3, pp. 526–553, 2009.

[30] R. F. Freund and H. J. Siegel, “Heterogeneous processing,” *IEEE Computer*, vol. 26, no. 6, pp. 13–17, 1993.

[31] A. Khokhar, V. K. Prasanna, M. Shaaban, and C. L. Wang, “Heterogeneous computing: challenges and opportunities,” *IEEE Computer*, vol. 26, no. 6, pp. 18–27, 1993.

[32] H. J. Siegel, J. K. Antonio, R. C. Metzger, M. Tan, and Y. A. Li, “Heterogeneous computing,” in *Parallel and Distributed Computing Handbook*, A. Y. Zomaya, Ed., pp. 725–761, McGraw-Hill, New York, NY, USA, 1996.

[33] V. S. Sunderam, “Design issues in heterogeneous network computing,” in *Proceedings of the Workshop on Heterogeneous Processing*, pp. 101–112, IEEE Computer Society, Beverly Hills, Calif, USA, 1992.

[34] R. Wolski, N. T. Spring, and J. Hayes, “Network weather service: a distributed resource performance forecasting service for metacomputing,” *Future Generation Computer Systems*, vol. 15, no. 5, pp. 757–768, 1999.

[35] L. Gong, X.-H. Sun, and E. F. Watson, “Performance modeling and prediction of nondedicated network computing,” *IEEE Transactions on Computers*, vol. 51, no. 9, pp. 1041–1055, 2002.

[36] E. Cantú-Paz, “A summary of research on parallel genetic algorithms,” Tech. Rep. 95007, University of Illinois, Urbana-Champaign, Ill, USA, July 1995.

[37] H. Mühlenbein, “Evolution in time and space—the parallel genetic algorithm,” in *Foundation of Genetic Algorithms*, pp. 316–337, Morgan Kaufmann, San Francisco, Calif, USA, 1992.

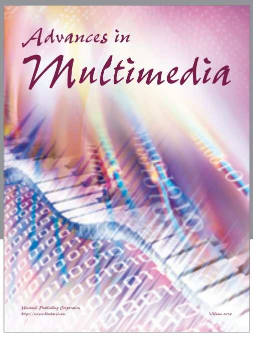
[38] M. Snir, S. Otto, S. Huss-Lederman, D. Walker, and J. Dongarra, *MPI: The Complete Reference, Vol. 1—The MPI Core*, MIT Press, Cambridge, Mass, USA, 1998.

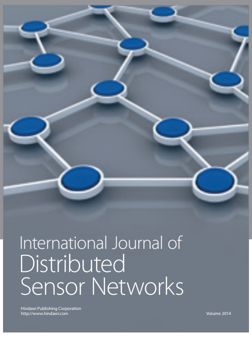
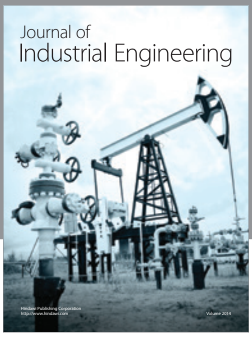
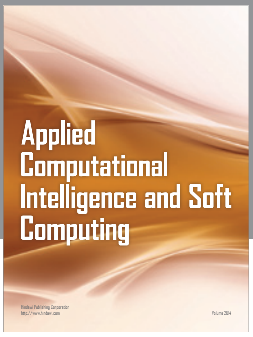
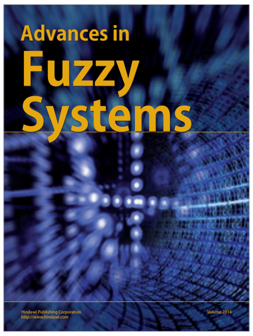
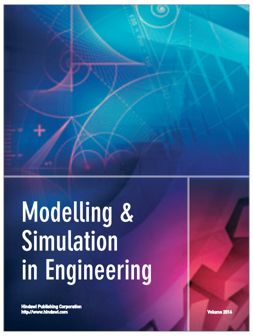
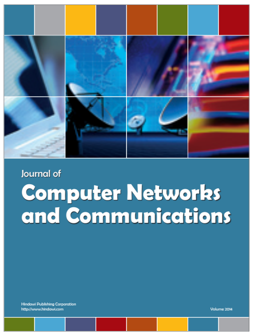
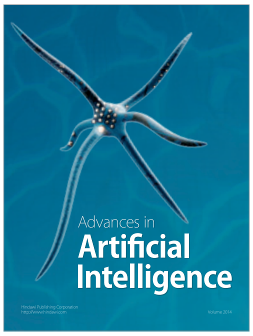
[39] N. T. Karonis, B. Toonen, and I. Foster, "MPICH-G2: a grid-enabled implementation of the Message Passing Interface," *Journal of Parallel and Distributed Computing*, vol. 63, no. 5, pp. 551–563, 2003.

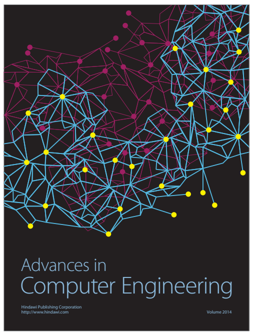
[40] T. Kielmann, H. E. Bal, J. Maassen, et al., "Programming environments for high-performance grid computing: the Albatross project," *Future Generation Computer Systems*, vol. 18, no. 8, pp. 1113–1125, 2002.

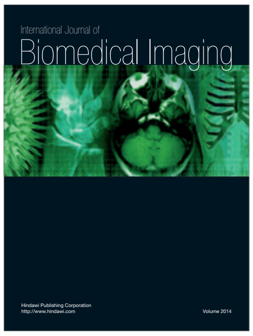
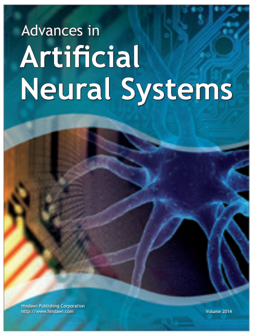
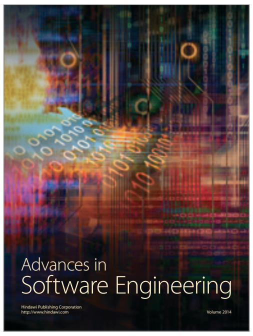
[41] G. E. Fagg, K. S. London, and J. J. Dongarra, "MPI connect: managing heterogeneous MPI applications interoperation and process control," in *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, vol. 1497 of *Lecture Notes in Computer Science*, pp. 93–96, Springer, New York, NY, USA, 1998.

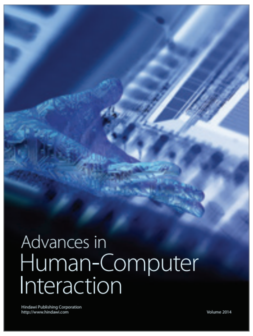
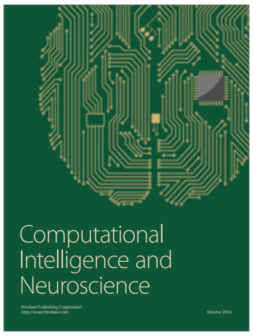
[42] B. Bierbaum, C. Clauss, T. Eickermann, et al., "Reliable orchestration of distributed MPI-applications in a UNICORE-based grid with MetaMPICH and MetaScheduling," in *Proceedings of the 13th European PVM/MPI User's Group Meeting*, vol. 4192 of *Lecture Notes in Computer Science*, pp. 174–183, Bonn, Germany, September 2006.


Advances in
Multimedia

The Scientific World Journal

International Journal of
Distributed Sensor Networks

Journal of
Industrial Engineering

Applied Computational Intelligence and Soft Computing

Advances in
Fuzzy Systems

Modelling & Simulation in Engineering

Journal of
Computer Networks and Communications

Advances in
Artificial Intelligence


Advances in
Computer Engineering

International Journal of
Computer Games Technology

International Journal of
Biomedical Imaging

Advances in
Artificial Neural Systems

Advances in
Software Engineering

Journal of
Robotics

Advances in
Human-Computer Interaction

Computational Intelligence and Neuroscience

International Journal of
Reconfigurable Computing

Journal of
Electrical and Computer Engineering