ENCODING MINIMUM REQUIREMENTS OF INTER-CONNECTED GRID VIRTUAL ORGANISATIONS USING GENETIC ALGORITHMS

Stelios Sotiriadis, Nik Bessis, Paul Sant, Carsten Maple
Department of Computer Science and Technology
University of Bedfordshire, UK
(stelios.sotiriadis, nik.bessis, paul.sant, carsten.maple) @beds.ac.uk

ABSTRACT

The work herein continues the effort of achieving inter-cooperation between several Virtual Organisations (VOs) by utilising a heuristic genetic algorithm for resource discovery within such environments. The discovery process is based on an ad hoc strategy where each participant acts as an individual, detached from any centralized topologies articulated by VOs. The method is extended to an inter-collaborative model wherein discovery is derived from data extracted from a member snapshot profile. The primary challenging goal in building an ad hoc grid is supplying each grid member with specific directions for continuously maintaining information related to each community participant. Such information is stored at each VO member public profile and is available for advertising on the resource discovery process. The proposed approach will be able to define a collaborative model within a community domain and extend it to an inter-communication environment. To define these models it is essential to share a common understanding of the structure among community members. A way to achieve it is by utilizing Genetic Algorithms which offer a way to correlate VO members' roles and actions into a genetic chromosome of information and be able to be composed in a common format and be shared among VO participants. Consequently, by mimicking the same processes that nature uses, we deliver a case scenario of genetic algorithms in order to assist the resource discovery among inter-cooperated VOs.

KEYWORDS

Open Grid, Ad hoc grid, Self Led Critical Friends, Genetic Algorithms

1. INTRODUCTION

The ad hoc grid is a spontaneous organisation of cooperative heterogeneous nodes without fixed infrastructure and central administration [1] and provides computing resources on demand. In this way community members are capable of defining their own locally administrative rules for communication and delegation within an inter-cooperative grid. However, the accessibility factor of certain resources makes VO members mistrustful of the quality of neighbouring participants. The problem gets worst when several members have already been defined with local policies and characteristics. As a result, it is essential that current VOs need to be redefined under a common scheme with a specific structure according to requirements based on member interactions. However, due to the huge number of grids, each one has been developed using different standards and structures thus, it is necessary to unify the design of such an environment. Nowadays, a new design gets the attention of the academic community; the fully decentralized models in which every VO participant needs to embrace and invoke any partner in order to provide a more autonomous solution. In this direction [3] suggests that the grid is best described in terms of what it brings to the individuals and communities and the middleware design is built upon their needs.

Typically within a VO, members are able to share jobs which are inherently bounded by various factors such as the adopted information system or other agreed constraints [5]. Problems raised by such limitations are thus related to finding a way to enable interoperation between nodes from different environments. So it is essential that decentralization of several grids could be achieved by utilizing a common model of knowledge of each ad hoc grid system. By considering the primary goal of ad hoc grids, that inner capabilities must be shared among community members, participant potentials have to be incorporated into a metadata snapshot profile of internal attributes. By analyzing roles and actions of a member, we conclude that participant possibilities are derived from the minimum requirements that need to be addressed in order to achieve a successful interaction [9]. This information refers to internal procedures of a member such as policy authorization, internal knowledge, physical resources and time constraints, advertised through a metadata snapshot profile within a public domain. The aforementioned standards compose a blueprint of each participant analogous to an organism which is built from tiny blocks of life. The effort here aims to encode VO participant attributes into a chromosome of specific traits. Therefore by utilizing genetic algorithms we may then support that resource discovery in ad hoc grids could be a step forward in the direction of inter-cooperated grids. In the following sections we present a study of applying genetic algorithms to a collaborative environment by starting with the inspiration of the main concept, the definition of a metadata snapshot profile and the case scenario of the genetic algorithms. The aforementioned part consists of a brief clarification of GAs principles and by analysing their functionality we aim to encode grid members' roles and actions into a genetic resource discovery problem. The final solution of optimized paths within a collaborative environment aims to demonstrate the employment of the heuristic algorithm to a typical VO member interaction.

2. MOTIVATION

An open grid predisposes the existence of several members of a huge collection of VOs which are interconnected forming the grid. The method of self led critical friends (SFC) based on the above idea, introduced by [4] defines a novel model as the next step to achieve the inter-cooperation of multi institutional VOs. They suggest that SCF are intermediate stations in the communication between multiple grids by composing an extended environment. In essence, each SCF plays the role of a mediator by reflecting the information to different members of the organization as much as possible in order to maximize the grid members. The prospects opened up by the use of this technique, allows the realization of communication between different grids, without having to be controlled by a single point manager. The method is straightforward and each VO member will have the opportunity beyond the internal boundaries of the organisation; to perform jobs on interconnected trusted members belonging to undisclosed VOs. On the other hand, the concept of an ad hoc grid can assist in the consolidation of several grids, particularly in cases where the discovery of resources uses the SCF approach. By definition, an ad hoc grid provides a flexible environment for the members, minimizing the scalability and offering a decentralized model without a fixed structure [6]. This can be put into practice by any member of any VO with only minimal administrative requirements and within the safety of a flexible and secure environment. Although in some cases grids can be mistrustful of the above method because they are affiliated to specific tactics and securities articulated by a central administrator, so the use of SCF in an ad hoc grid can lead to the realization of an open grid. Consequently cooperation between centralized and decentralized VOs can be effective; especially if a handshake communication between members and administrators is legitimate. The information of each member is reflected in a profile and is available to members upon request. Essentially, the profile will be published to all members during the course of the resource discovery process. The recorded information extracted from the characteristics of each participant and is the identity of VO or SCFs referred to the agreed protocols, the level of knowledge and a list of available jobs.

3. CASE SCENARIO: THE GENETIC ALGORITHM

It is a challenging project to gather prerequisites in order to meet the resource requirements of large scale grids. These kinds of characteristics make the resource discovery and scheduling a complex optimization problem. A discussion about Genetic Algorithm (GA) has been presented by [2] which supports the viewpoint that GAs have been widely used to solve these problems efficiently. However, authors in [1] discuss that the standard genetic algorithm is too slow when used in a realistic scheduling scenario due to its time consuming iteration and are not guaranteed to reach a conclusion or can get stuck in local optima. So we present a realistic approach in which each interaction between members is based on the least minimum prerequisites for a successful interaction by composing a blueprint of traits for each interaction. More specifically by translating minimum requirements into a chromosome we may then improve the scheduling of jobs as well as the snapshot profile information. The GA is applied at the resource discovery stage whilst the data are extracted from the metadata snapshot profile. Finally, the evolutionary operations are selected according to the job scheduling description.

3.1 The principle

All information stored within a metadata snapshot profile can be described as the characteristics of a VO member. We define the internal capabilities of each member as attributes and these are encoded into genes in correlation to an organism, which sequentially are connected together into long strings called chromosomes [8]. Hence, in an analogous way, each community member's characteristics behave as a gene by representing these as specific traits within a chromosome. These genes and their attributes are referred to as an organism genotype. The physical expression of the genotype which is the entire organism is called the phenotype. During an interaction among members, a procedure starts in which each participant contributes by sharing their internal characteristics, which we call genes. In real life when two organisms share their genes the consequential offspring may be created by inheriting genes from both parents. This process is called recombination or crossover [8] and in a typical grid VO, interactions are similar to real life examples by combining roles and actions for an improved result. Very infrequently a gene may be mutated, and normally will affect the participant phenotype. The new offspring member will be expressed in the organism as a new trait. In this way, each member attribute will be affected after a successful interaction and new data will be stored to each node public profile. The new evolved offspring member has been mutated and new actions and roles have been stored as new genes in real life. In this case, the completely new trait of the node is comparable to a mutated chromosome of DNA. The success of the offspring members to internal interactions with other participants will be measured by their fitness value.

3.2 The Algorithm

The algorithm begins with a set of possible solutions which are represented by the chromosomes – at this point the internal roles and actions of a VO member – which represents the population for the GA. The aim is that solutions from one population will be used to a form a new population that exhibits better characteristics than the old one called offspring. However, potential new solutions could be worse so the new population needs to be selected according to their fitness value. This mechanism will be able to decide which members should be present in the successor population. Before utilizing a genetic algorithm to solve a problem, we require a large set that can represent the so-called solution space. In analogy, a typical grid member's attributes should be extracted and encoded as a chromosome. This could be achieved by the conversion of an attribute to a binary bit string; the chromosome. Attributes have been defined by [9] as the minimum prerequisites that a node should achieve before an interaction occurs. Moreover, they propose that it is crucial to identify minimum requirements primary including policy management control, knowledge base pairing and physical and time constraints. In fact, above standards could be converted into a chromosome bit and constitute the population of the member organism. Minimum requirements are identified as follows:

Step 1: *Policy management* is the authorization procedure which will be carried out by a member. This includes a login process by utilizing accounts and contract validations by using agreements.

Step 2: *Knowledge base pairing* provide information concerning actions and capabilities of each node.

Step 3: *Physical resource constraints* as a record of a computer physical standards such as CPU, Memory Space, Hard Disk, Operating System, etc. and heuristic information of network devices.

Step 4: *Time constraints* as the due time of a job, including execution and communication times.

The above prerequisites constitute the attributes that each participant should include in the form of a chromosome. By encoding each step we should be able to define each attribute as a unique gene.

3.3 Parallelism of terms

The following section presents an association between ad hoc grid and genetic terminologies. In a genetic algorithm an individual is a possible solution to the problem, equivalent to a job delegation within a VO. The candidate group of members responsible for problem solving is called the population and there are looking for all possible solutions to the search space. As Chromosome we identify the sum of minimum prerequisite that nodes must retain before a successful interaction, comparable to the blueprint of an individual – here any possible solution. [7, 8] describe that the possible aspects of an individual are called traits and the possible settings for a trait is called allele. They also suggest that as locus is defined the position of a gene within a chromosome and finally the collection of all chromosomes for an individual is called genome.

3.4 Encoding minimum requirements

Before solving our problem it is essential to encode any potential solution. The association of attributes minimum requirements will finally be presented within the chromosome of a VO member. Figure 1 demonstrates the encoding of VO minimum requirements into a DNA of specific genes and each gene into a detailed trait of alleles with specific locus within the chromosome.

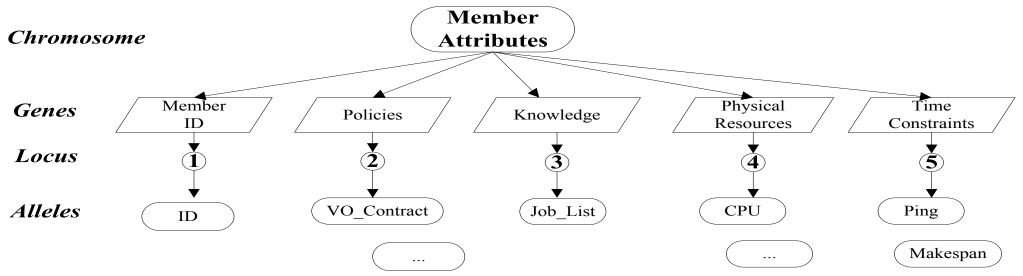

Figure 1: Encoding of minimum requirements

Allele's values will specify each member's capabilities and will be playing a vital role in the resource discovery scheduler. In fact, each time the algorithm initiates each VO member it will be transformed into a chromosome based on information stored in each public profile. In this stage it is crucial to assign each allele a value in order to compose the chromosome. The new structured chromosome consists of specific locus as the positions of each gene in the chromosome and data from the alleles that constitute the genetic information.

3.5 Resource discovery method

At this time we assume that within a VO, a common policy and knowledge background is shared among participants so locus 2 and 3 are subject to a pairing procedure in the discovery process because they are already predefined. Physical resources and time constraints are selected according to each member's requirements. So in that case, the chromosome is constructed by locus 4 and 5. On the other hand, within a public domain, resource discovery may be complicated because the pairing of knowledge and jobs description could not be successful. Difficulties raised from such limitations may be overcomed by utilizing genetic algorithms and by generating a new and better offspring profile of internal capabilities. Accordingly, the metadata profiles could be improved in locus (position of data within a chromosome) 2 and 3.

Currently, the above procedure assumes that information is already predefined and paired successfully and it will be discussed in future papers. While locus 2 and 3 are selected for pairing purposes, locus 4 and 5 can be the means to achieve the resource discovery. Each chromosome can be represented by a composition of physical and time constraints. In order to start the resource discovery process, we need to encode physical resources and times in a chromosome. By using genetic algorithms, we can decide the best weighted path of each graph edge based on physical resource and time information. According to [9] the physical resource value is calculated according to the following formula:

$$Ranking_i = 1/\,(CPU_i * CPU\_Coefficient_i + Memory_i * Memory\_Coefficient_i + HD_i * HD\_Coefficient_i)$$

in which CPU_Coefficient, Memory_Coefficient and HD_Coefficient are values assigned as a measure of required values for each physical resource. For instance, if a node requests high CPU power the CPU Coefficient will be 1, in other case will be from 0,1 to 0,9. Times are calculated by the following formula:

$$T_i = Ping_i * 2 * Makespan_i$$

where ping is the communication time needed to connect to a member and makespan the expected execution time of a job [2] according to information stored in the public profile. The makespan time is calculated as the time difference between the start and finish of a sequence of jobs or tasks. So the weight of each path is measured by the following formula:

$$W_i = Ranking_i * T_i$$

According to the job description we assume that the default parameters for the coefficient values are: CPU Coefficient: 1, Memory Coefficient: 1, H/D Coefficient: 1. So the algorithm is initiated as follows:

The algorithm first retrieves a population of VO members and calculates the weight of each path as follows:

$$W_i = Ranking_i * T_i$$

The following algorithm aims to find the best available member for task execution. The substantial goal is to improve the general performance of a particular VO in the way of discovering the best neighbouring member.

A. *Start:* The algorithm first retrieves a population of specific VO members which are in particular the chromosomes suitable for solving the problem.

B. *Fitness utility:* Evaluate a fitness function of each generated member - chromosome and decides the best solution that fits to the problem.

C. *New population:* Generate a new population of members by repeating the following steps until the new population is complete:

a. *Reproduction:* Select two parent chromosomes from a population according to their fitness score specified by the fitness function.

b. *Crossover:* Decide a crossover probability and cross over the parent chromosome of the previous step in order to form a new offspring.

c. *Mutation:* Mutate the new offspring by deciding mutation probability and locus – position of mutation within a chromosome.

d. *Recombination:* Accept the generated offspring as the new population and use the new offspring for further runs of the algorithm.

e. *Return:* Return the best solution in the current population

f. *Loop:* Go to step 2

3.6 Solution of optimized paths using SCF’s

In the following situation we aim to optimise an interaction within a VO in order to schedule a job into any cooperated members.

Problem definition: We aim to schedule a job called JobA from a member called N1. The scheduler will be able to decide the best members for job execution by selecting nodes belonging to the same VO called VOA or to any SCF node. The main idea is to request profile data from all members. The condition is that the schedule of the journey is the order of visiting the best members in such a way that the cost is the minimum distance. Figure 2 illustrates such an interaction.

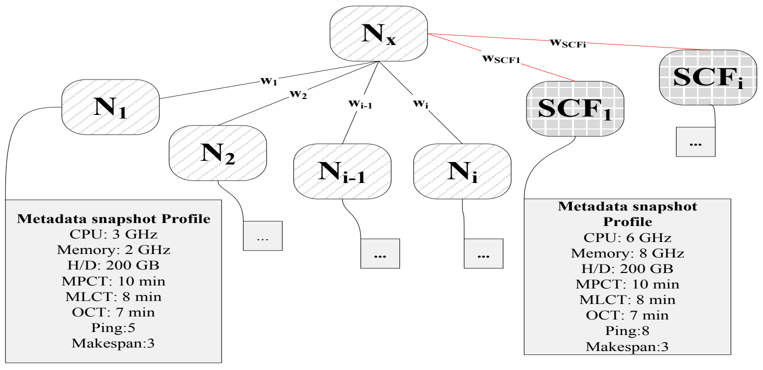

Figure 2: Interaction of members

Data extracted from member’s public profile is stored in the metadata snapshot profile and a weight value for each path is calculated. We assume that Ping and Makespan times [9, 2] are stored from previous jobs.

In order to initialize the algorithm of the previous section we need to identify the requirements of each interaction. In the following section, we present in part A the objectives of the job, in part B the fitness function and in part C the new population generation.

3.6.1 The objectives

In a decentralized grid a member should be capable of sending jobs to the open grid by requesting information of each participant’s public profile. The objectives (requirements and criteria) for the job execution are the following:

1. Number of jobs to schedule: 1 Job
2. Number of parallel jobs to be run: No parallel jobs are selected for this scenario
3. Number of candidate nodes: i
4. Contract pairing: Successful paired
5. Job description pairing: Successful paired
6. Number of required physical resources of each job: Unknown
7. Number of available physical resources of each participant: Retrieved from the metadata snapshot profile and transformed into a binary bit (the chromosome)
8. Communication times: Retrieved from the metadata snapshot profile data (Contained within the chromosome decided in 8)
9. Execution times if are available: Unknown

3.6.2 The fitness utility

The objective function will be the fitness that quantifies the optimality of each solution. In that way, each chromosome may be ranked against others. The best optimal chromosomes will be able to mix their datasets producing a better generation. In this scenario we assume that a well-disposed tactic is agreed between members and contract pairing and job matching is already agreed. So we consider that the fitness function will be to minimize the weight of each node in order to find the best solution.

Fitness utility: Min {Wi: i ∈ NodeNumber}

3.6.3 The new population

The algorithm starts by deciding the selection, crossover and mutation operations. The following section briefly introduces the possible operations at each stage. Each operation is selected according to the problem description.

- The selection operations are:
  1. Greedy initial population that gives preference to linking with nodes that are close to the requester either are within the same VO or SCFs. In this case the ping time is selected to form the chromosome.
  2. Select the best individuals based only to those with the best fitness score which is the minimum weight.
  3. Ranking selection of each participant according to probability linearly proportional to its rank. [2]
  4. Running several Tournaments selections among a few individuals with the best fitness score.

Selection Algorithm

```
Precondition: Best Linking, Best Fitness, Best Ranking, Best Tournament
Precondition: Constant x, i ∈ Node Number, w ∈ Weight_i CPU, Memory, HD, CPUCoefficient, MemoryCoefficient, HDCoefficient, Ping, Makespan
01: Request CPU, Memory, HD, Ping
02: case 1: Best Linking
03:     x = Ping * 2 * Makespan
04: case 2: Best Fitness
05:     x = Min(W_i)
06: case 3: Best Ranking
07:     x = 1/ (CPU*CPUCoefficient + Memory*MemoryCoefficient + HD * HDCoefficient)
```

```
08: case 4: Best Tournament
09:     while (Best Fitness is reached)
10:         Compare Best Linking $_x$ AND Best Ranking $_x$
11:     end while
```

While the selection is completed the crossover operation combines two parents to produce a new offspring with the intention that these children paths selected by the above methods will be better than the parents.

- The crossover operations are:
    1. One Point: Randomly selects a crossover point and interchanges the two parents.
    2. Two Points: Randomly selects two crossover points within a parent.
    3. Uniform: A crossover operator that decides which parent will contribute each of the gene values in the offspring chromosomes.
    4. Heuristic: A crossover operator that uses the fitness values of the two parent chromosomes to determine the direction of the search. The offspring are created according to the following equations: Offspring1 = BestParent + r * (BestParent – WorstParent) and Offspring2 = BestParent where r is a random number between 0 and 1.

Crossover Algorithm

```
Precondition: Parents, Best Parent, Worst Parent, Offspring
Precondition: Constat r ∈ {0,1}
01: case 1: One Point
02:     Randomly select two parents according to Best Linking or Best Fitness or Best Ranking
03:     apply One Point crossover
04: case 2: Two Points
05:     Randomly select two parents according to Best Fitness or Best Ranking
06:     apply Two Point crossover
07: case 3: Uniform
08:     Select Two Parents according to Best Fitness score and Best Ranking score
09:     apply One Point crossover
10: case 4: Heuristic
11:     Select Two Parents according to fitness score
12:     Offspring1 = BestParent + r * (BestParent – WorstParent)
13:     Offspring2 = BestParent
14:     apply One Point crossover between Offspring1 and Offspring2
```

In a small percentage of cases, the child tours are mutated, this is done to prevent all paths in the population from looking identical.

- The mutation operations are:
    1. Flip bit: Simply by inverting the value of the choosing gene (1 to 0 etc.)
    2. Boundary: A mutation operation for integers or floats in which the value of the chosen gene is mutated with either upper or lower bound.
    3. Non-Uniform: An operator that increases the probability of mutation closer to 0 as the generation increases. This mutation operator keeps the population from stagnating in the early stages of the evolution then allows the genetic algorithm to fine tune the solution in the later stages of evolution.
    4. Gaussian: A mutation operator which adds a unit Gaussian random value for preventing new genes of falling outside the user specified bounds.

Mutation Algorithm

```
Precondition: Chromosome $c_i$, Constant x,y
01: case 1: Flip Bit
02:     Invert value of $c_i$
03: case 2: Boundary
04:     While ($c_i \in \{x,y\}$ )
05:     Invert value of $c_i$
06: case 3: Non-Uniform
08:     While $lim(Mutation\ c_i) = 0$
09:     Invert value of $c_i$
```

```
10: case 4: Gaussian
11:     add a random value from a Gaussian distribution to each chromosome element
```

At last, the new child paths are inserted into the population replacing two of the longest paths. The size of the population remains the same and the new paths are repeatedly created until the desired path is reached. Considering the Job_A example the selection algorithms in case:1 initializes communication with preference to nodes closer to the requester. The ping time is used in order to form the chromosome and the crossover operation starting by selecting two parents and applying one point crossover. Finally, the flip bit mutation algorithm is initiative and a new offspring is created. The aforementioned procedure is the simplest GA, however it is possible to reach a local optimum [10, 11]. On the other hand, in the best fitness algorithm the minimum weight is selected based on the physical resources and time constraints, the crossover operator can select case: 1, 2, 4 and the generated offspring can be mutated by case: 2, 3, 4. So the selected offspring will be the minimum spanning tree of a weighted graph. The best ranking selection utility based only in the physical resource information and crossover case: 1, 2 and mutation case: 2, 3, 4 can be selected in order to create the offspring. Finally the best tournament selection utilizes best linking and best ranking and by crossover and mutation the algorithm selects the new offspring.

4. CONCLUSION

The idea of inter-connected grids refers to a huge search space of unlimited members with no boundaries, in which any member of any VO could be able to communicate with everyone. So, if we consider SCF as the resource discovery method; the search space is continually increasing as time passes. This paper intends to deliver a new solution in which grids are seen from a decentralized scope, by utilizing genetic algorithms we may improve the resource discovery process. Future work of this project includes a more sophisticated genetic evolution model in which the procedure will assist to the creation of a new offspring by improving VO members. In this way, internal roles and actions such as policies and knowledge could be updated at any time of a job delegation and new understanding of the public domain can be stored in the snapshot profile.

5. REFERENCES

[1] Abdulal, W., Jadaan, O.A., Jabas, A., Ramchandraram, S., “Rank-based Genetic Algorithm with Limited Iteration for Grid Scheduling”, *2009 First International Conference on Computational Intelligence*, Communication Systems and Networks, 2009.

[2] Carretero, J., Xhafa, F., “Use of genetic algorithms for scheduling jobs in large scale grid applications”, *Technological and economic development of econom,*, vol xii, no 1, 11–17, 2006

[3] De Roure, D., Jennings, N.R., Shadbolt, N.R., “The Semantic Grid: Past, Present and Future”, *Proceedings of the IEEE*, 93 (3) PP. 669-681. 2005

[4] Huang, Y., Bessis, N., Kuonen, P., Brocco, A., Courant, M., Hirsbrunner, B., “Using Metadata Snapshots for Extending Ant-based Resource Discovery Functionality in Inter-cooperative grid Communities”, *International Conference on Evolving Internet (INTERNET 2009)*, IEEE, Cannes/La Bocca, France, August 2009.

[5] Huang, Y., Bessis, N., Brocco, A., Sotiriadis, S., Courant, M., Kuonen, P., Hisbrunner, B., "Towards an integrated vision across inter-cooperative grid virtual organizations", *Future Generation Information Technology (FGIT 2009)*, pp.120-128, Springer LNCS, Jeju island, Korea, 2009.

[6] Huraj, L., Siládi, V., “Authorization through Trust Chains in Ad hoc Grids”, *Euro American Conference on Telematics and Information Systems (EATIS’09)*, Prague, June 2009.

[7] Liu, J., Chen, L., Dun, Y., Liu, L., Dong, G., “The Research of Ant Colony and Genetic Algorithm in grid Task Scheduling”, *International Conference on MultiMedia and Information Technology*, 2008.

[8] Russel, S., and Norvig, P., (2003), Artificial Intelligence: A modern Approach, New Jersey: Pearson Education

[9] Sotiriadis, S., Bessis, N., Huang, Y., Sant, P.And Maple, C. (2010).”Defining Minimum Requirements Of Inter-Collaborated Nodesby MeasuringThe Weight Of Node Interactions”, *4th International conference On Complex,Intelligent And Software Intensive Systems (Cisis-2010)*, 15th-18th February, Krakow

[10] Xhafa, F., Barolli, L., Durresi, A., “Batch Mode Schedulers for grid Systems”, *International Journal of Web and grid Services*, Vol. 3, No. 1, 19-37, 2007.

[11] Xhafa, F., Carretero, J., Barolli, L., Durresi, A., “Immediate Mode Scheduling in grid Systems”, *International Journal of Web and grid Services*, Vol.3 No.2, 219-236, 2007.