Volume 3, Issue 1, January 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Distributed Job Scheduling on The Grid Using Particle Swarm Optimization (Pso) Algorithm

Thanapal P
Research Scholar
CMJ University
Shillong, Meghalaya, India

Dr. Gunasekaran G
Principal
Meenakshi College of Engineering
Chennai, India

Abstract— This paper represents a swarm intelligence simulated load balancing algorithms for job scheduling in grid. ParticleX fly in problem search space to find optimal or near-optimal solutions. One is based on ant colony optimization which is called AntX, the ParticleX shares the workload among node. The other one is based on scheduling algorithm with GridSim tool kit. In the Greedy Randomized Adaptive Search Procedure (GRASP), a number of iterations are conducted to search a possible optimal solution for scheduling tasks on resources. A solution is generated at each iterative step and the best solution is kept as the final schedule. In this paper we used a Grid simulation toolkit (GridSim) dedicated to Grid simulations for grid job scheduling Experimental studies show that the proposed novel approach is more efficient than the Cluster based and Duplication based scheduling and GRASP.

Keywords— GRASP, GridSim, AntX, ParticleX, Grid Information Servers, Scheduling Algorithm

I. Introduction

The computers that we use today are loaded with the high speed internet, powerful processors and high speed network with low cost commodity components. These facilities led the possibility of utilizing geographically distributed data and multiple resources for solving large scale problems in engineering and technologies. Recent work on these topics leads to the emergence of a new model called Grid Computing [12].

Computational Grid is a collection of a various set of virtual organizations (VOs). Based on the policies, VO provides access to various resources and services that are heterogeneous and are distributed in various geographical areas. Grid environment are dynamic in nature because at any moment, any resources or services can be added or removed. Providing quality of service is an important aspect in many distributed computation and communication. Service is used to illustrate the minutiae of the resources within the grid. Grid resources and services are registered within one or more Grid Information Servers (GIS). If the user wants to utilize the resources, the request has to be submitted to the Grid Resource Broker (GRB). Since it is a distributed environment, different user may request different resources and the resources are in different capabilities. GRB collects the requests from the end user, query it with GIS and allocate the resources accordingly. Cao [2] used agents to schedule grid. In this method different resources and services are regarded as different agents and grid resource discovery and advertisement are performed by these agents. Buyya [3] used economic based concepts including commodity market, posted price modeling, contract net models, bargaining modeling etc for grid scheduling. As mentioned in [5] scheduling is NP-complete. Meta-heuristic methods have been used to solve well known NP-complete problems. In [6] Yarkhanan and Dongarra used simulated annealing for grid job scheduling. GAs for grid job scheduling is addressed in several works [7], [8], [9] and [11]. Abraham et al. [10] used fuzzy PSO for grid job scheduling. Different criteria can be used for evaluating the efficacy of scheduling algorithms and the most important of which are make span and flow time. Make span is the time when grid finishes the latest job and flow time is the sum of finalization times of all the jobs. An optimal schedule will be the one that optimizes the flow time and make span [10].

The method proposed in [10] aims at simultaneously minimizing make span and flow time. A version of discrete particle swarm optimization (DPSO) is proposed for grid job scheduling and the goal of scheduler is to minimize the two parameters mentioned above simultaneously. This method is compared to the method presented in [10] in order to evaluate its efficacy. The results show the presented method is more efficient and this method can be effectively used for grid scheduling. The remainder of this paper is organized in the following manner.

The state of current research on scheduling algorithms for the new generation of computational environments will be surveyed and open problems will be discussed. The remainder of this paper is organized as follows. An overview of the Grid scheduling problem is presented with a generalized scheduling architecture. The progress made to date in the design and analysis of scheduling algorithms for Grid computing is reviewed.

II. Overview of the Grid Scheduling Problem

A computational Grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities [16]. It is a shared environment implemented via the deployment of a persistent, standards-based service infrastructure that supports the creation of, and resource sharing within, distributed communities. Resources can be computers, storage space, instruments, software applications, and data, all connected through the Internet and a middleware software layer that provides basic services for security, monitoring, resource management, and so forth. Resources owned by various administrative organizations are shared under locally defined policies that specify what is shared, who is allowed to access what, and under what conditions [17]. The real and specific problem that underlies the Grid concept is coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations [15].

From the point of view of scheduling systems, a higher level abstraction for the Grid can be applied by ignoring some infrastructure components such as authentication, authorization, and resource discovery and access control. Thus, in this paper, the following definition for the term *Grid* adopted: "*A type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed autonomous and heterogeneous resources dynamically at runtime depending on their availability, capability, performance, cost, and users' quality-of-service requirements*" [14].

To facilitate the discussion, the following frequently used terms are defined:

- A ***task*** is an atomic unit to be scheduled by the scheduler and assigned to a resource.
- The ***properties*** of a task are parameters like CPU/memory requirement, deadline, priority, etc.
- A ***job*** (or ***metatask***, or ***application***) is a set of atomic tasks that will be carried out on a set of resources. Jobs can have a recursive structure, meaning that jobs are composed of sub-jobs and/or tasks, and sub-jobs can themselves be decomposed further into atomic tasks. In this paper, the term *job*, *application* and *metatask* are interchangeable.
- A ***resource*** is something that is required to carry out an operation, for example: a processor for data processing, a data storage device, or a network link for data transporting.
- A ***site*** (or ***node***) is an autonomous entity composed of one or multiple resources.
- A ***task scheduling*** is the mapping of tasks to a selected group of resources which may be distributed in multiple administrative domains.

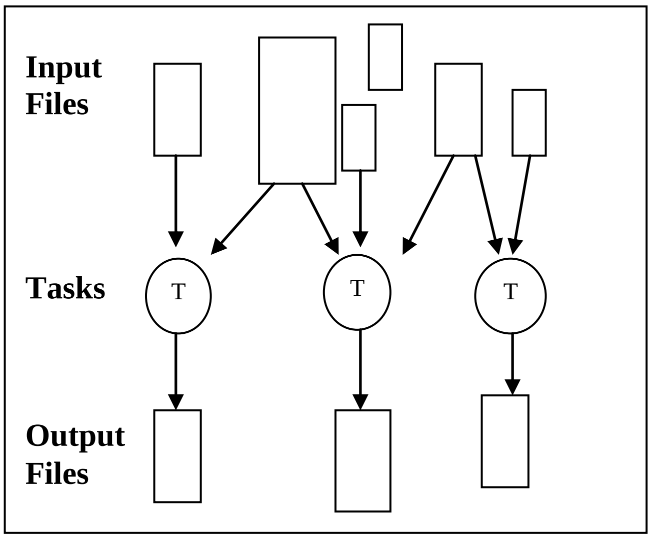

Figure 1 Application Model

A. Application Model

We define a parameter sweep application as a set of n independent sequential tasks $\{T_i\}_{i=1\ldots n}$. By independent we mean that there are no inter-task communications or data dependencies (i.e. task precedences). We assume that the input to each task is a set of files and that a single file might be input to more than one task. In our model, without loss of generality, each task produces exactly one output file. Figure 1 shows an example with input file sharing among tasks. We assume that the size of each input and output file is known a-priori. This model is motivated by our primary target application for PST: MCell [29], a micro physiology application that uses 3-D Monte-Carlo simulation techniques to study molecular bio-chemical interactions within living cells. A MCell run is composed of multiple Monte-Carlo simulations for cell regions whose geometries are described in (potentially very large) files. For instance, MCell can be used to study the trajectories of neurotransmitters in the 3-D space between two cell membranes for different deformations of the membranes, where each deformation is described in a geometry file. Additional files of variable sizes are also needed for describing the initial locations of different molecules. The model described above is adequate for our purpose and should be general enough to accommodate other applications (e.g. general Monte-Carlo simulations). MCell users and developers anticipate large-scale runs that contain tens of thousands of tasks with each task processing hundreds of MBytes of input and output data, with various task-file usage patterns. We propose an extended novel approach is more efficient than the Periodic and Event driven online rescheduling using GridSim.

III. GRIDSIM: GRID MODELING AND SIMULATION TOOLKIT

The GridSim toolkit provides a comprehensive facility for simulation of different classes of heterogeneous resources, users, applications, resource brokers, and schedulers. It can be used to simulate application schedulers for single or multiple administrative domains distributed computing systems such as clusters and grids. Application schedulers in grid environment, called resource brokers, perform resource discovery, selection, and aggregation of a diverse set of distributed resources for an individual user. That means, each user has his own private resource broker and hence, it can be targeted to optimize for the requirements and objectives of its owner. Whereas schedulers, managing resources such as clusters in a single administrative domain, have complete control over the policy used for allocation of resources. That means, all users need to submit their jobs to the *central* scheduler, which can be targeted to perform global optimization such as higher system utilization and overall user satisfaction depending on resource allocation policy or optimize for high priority users.

A. Features

Salient features of the GridSim toolkit include the following:

- It allows modeling of heterogeneous types of resources.
- Resources can be modeled operating under space- or time -shared mode.
- Resource capability can be defined (in the form of MIPS as per SPEC benchmark).
- Resources can be located in any time zone.
- Weekends and holidays can be mapped depending on resource's local time to model non-Grid (local) workload.
- Resources can be booked for advance reservation.
- Applications with different parallel application models can be simulated.
- Application tasks can be heterogeneous and they can be CPU or I/O intensive.
- There is no limit on the number of application jobs that can be submitted to a resource.

B. System Architecture

We employed a layered and modular architecture for grid simulation to leverage existing technologies and manage them as separate components.

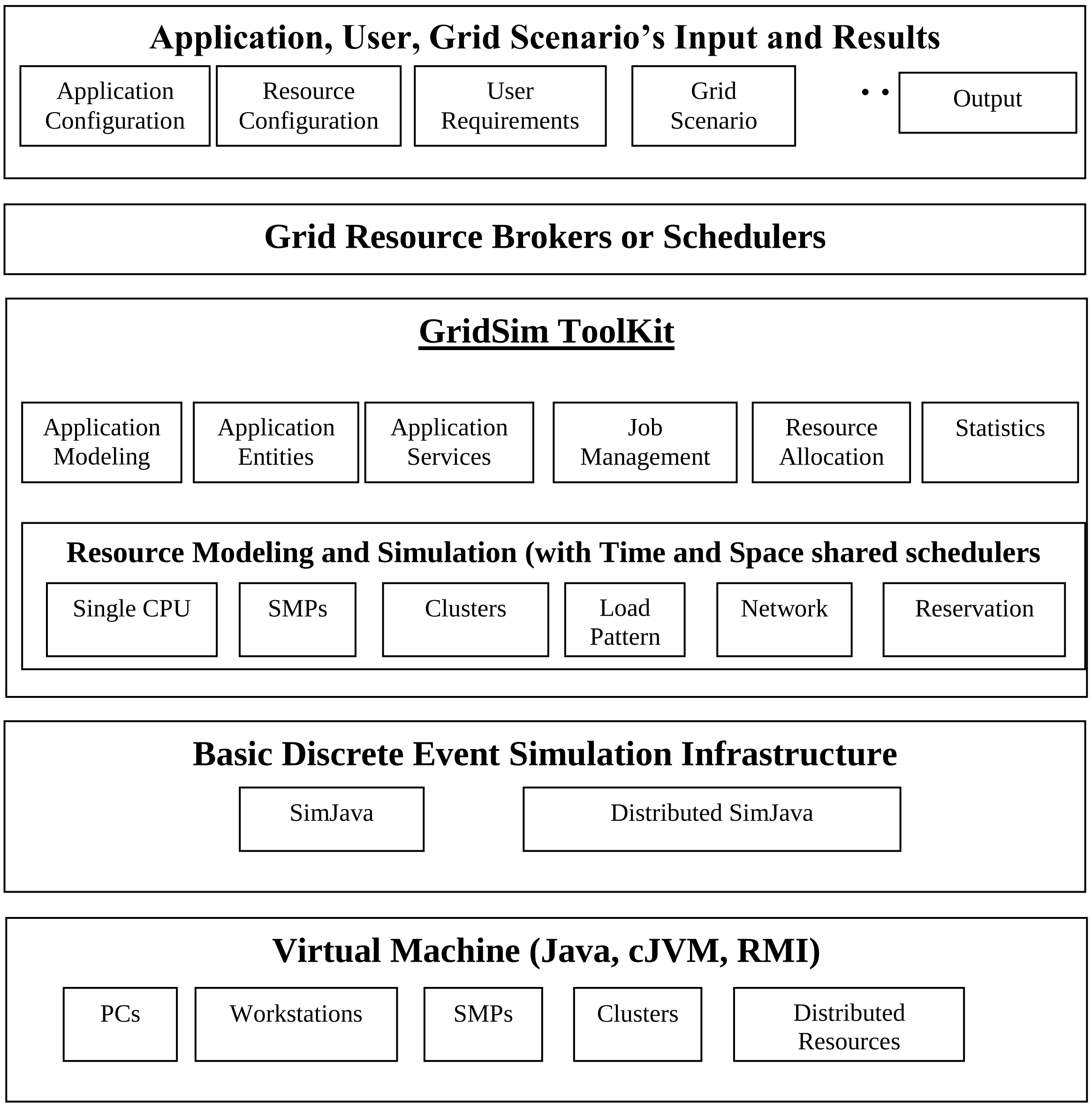

Figure 2: A modular architecture for GridSim platform and components.

A multi-layer architecture and abstraction for the development of GridSim platform and its applications is shown in Figure 2. The first layer is concerned with the scalable Java's interface and the runtime machinery, called JVM (Java Virtual Machine), whose implementation is available for single and multiprocessor systems including clusters [21].

The second layer is concerned with a basic discrete-event infrastructure built using the interfaces provided by the first layer. One of the popular discrete-event infrastructure implementations available in Java is SimJava [20]. Recently a distributed implementation of SimJava is also made available. The third layer is concerned with modeling and simulation of core Grid entities such as resources, information services, and so on; application model, uniform access interface, and primitives application modeling and framework for creating higher level entities. The GridSim toolkit focuses on this layer that simulates system entities using the discrete-event services offered by the lower-level infrastructure.

The fourth layer is concerned with the simulation of resource aggregators called grid resource brokers or schedulers. The final layer focuses on application and resource modeling with different scenarios using the services provided by the two lower-level layers for evaluating scheduling and resource management policies, heuristics, and algorithms. In this section, we briefly discuss SimJava model for discrete events (a second-layer component) and focus mainly on the GridSim (the third-layer) design and implementation. The resource broker simulation and performance evaluation is highlighted in the following sections.

IV. ALLOCATION ALGORITHM

This section presents a heuristic algorithm derived from the well known AntX technique. It first defines the basic concepts of the AntX and then explains how it may be employed for solving the allocation problem in terms of reliability.

A. Basic concepts

AntX is a global optimization technique which attempts to find the lowest point in an energy landscape [19,18]. The technique was derived from the observations of how slowly cooled molten metal can result in a regular crystalline structure. The distinctive feature of the algorithm is that it incorporates random jumps to potential new solutions. This ability is controlled and reduced as the algorithm progresses.

The objective function of the optimization problem is treated as the energy of a dynamic system while the temperature is introduced to randomize the search for a solution. The state of the dynamic system being simulated is related to the state of the system being optimized. The procedure is the following: the system is submitted to a high temperature and is then slowly cooled through a series of temperature levels.

B. Applying the algorithm

To implement the ParticleX algorithm, a number of decisions must be made. These decisions are concerned with the choice of an energy function, a cooling schedule, a neighborhood structure and the choice of annealing parameters such as an initial temperature, a cooling factor, an increasing factor of the inner loop repetitions and a stopping condition. Each decision needs to be made with care as they affect the speed of the algorithm and the quality of solutions.

The structure of the algorithm may be sketched as follows:

```
Randomly select an initial solution y;
Compute the cost at this solution $E_y$ ;
Select an initial temperature T;
Select a cooling factor $\alpha < 1$;
Select an initial chain $n_{rep}$;
Select a chain increasing factor $\beta > 1$;
Repeat
    Repeat
        Select a neighbor solution n to y;
        Compute the cost at n, $E_n$;
        $\delta = E_n - E_s$ ;
        If $\delta < 0$,
            d = n; $E_y = E_n$;
        Else
            Generate a random value f in the range (0,1);
            If f < exp(-δ /T ),
                y = n; $E_y = E_n$;
            End
        End
    Until iteration = $n_{rep}$ (equilibrium state at T)
    Set $T = \alpha * T$ ;
    Set $n_{rep} = \beta * n_{rep}$;
Until stopping condition = true
$E_y$ is the cost and y is the solution.
```

C. Individual task scdeduling

The individual task scheduling is the simplest scheduling method for scheduling workflow applications and it makes schedule decision based only on one individual task.

The Myopic algorithm [23] has been implemented in some Grid systems such as Condor DAGMan [22].

The algorithm schedules an unmapped ready task to the resource that is expected to complete the task earliest, until all tasks have been scheduled.

Algorithm 1 Myopic Scheduling Algorithm

```
Input: A workflow graph Ω(Γ, ^)
Output: A schedule
    1. while Et ε Γ is not completed do
    2. task ← get a ready task whose parent tasks have been scheduled
    3. r ← get a resource which can complete t at earliest time
    4. schedule t or r
    5. end while
```

D. Cluster based and Duplication based scheduling

Both cluster based scheduling and duplication based scheduling are designed to avoid the transmission time of results between data interdependent tasks, such that it is able to reduce the overall execution time. The cluster based scheduling clusters tasks and assign tasks in the same cluster into the same resource, while the duplication based scheduling use the idling time of a resource to duplicate some parent tasks, which are also being scheduled on other resources. Bajai and Agrawal [24] proposed a task duplication based scheduling algorithm for network of heterogeneous systems (TANH). The algorithm combine cluster based scheduling and duplication based scheduling and the overview of the algorithm is shown in Algorithm 1. It first traverses the task graph to compute parameters of each node including earliest start and completion time, latest start and completion time, critical immediate parent task, best resource and the level of the task. After that it clusters tasks based on these parameters. The tasks in a same cluster are supposed to be scheduled on a same resource. If the number of the cluster is greater than the number of resources, it scales down the number of clusters to the number of resources by merging some clusters. Otherwise, it utilizes the idle times of resources to duplicate tasks and rearrange tasks in order to decrease the overall execution time.

Algorithm 2 TANH Algorithm

```
Input: A workflow graph Ω(Γ, ^)
Output: A schedule
1. compute parameters for each task node
2. cluster workflow tasks
3. if the number of clusters greater than the number of available resources then
4.   Reducing the number of clusters to the number of available resources
5. else
6.   perform duplication of tasks
7. end if
```

E. Greedy Randomized Adaptive Search Procedure (GRASP)

A Greedy Randomized Adaptive Search Procedure (GRASP) is an iterative randomized search technique. Feo and Resende [28] proposed guidelines for developing heuristics to solve combinatorial optimization problems based on the GRASP concept. Binato et al [26] have shown that the GRASP can solve job-shop scheduling problems effectively. Recently, the GRASP has been investigated by Blythe et al [27] for workflow scheduling on Grids by comparing with the Min-Min heuristic on both computational- and data intensive applications. Algorithm 3 describes a GRASP. In a GRASP, a number of iterations are conducted to search a possible optimal solution for scheduling tasks on resources. A solution is generated at each iterative step and the best solution is kept as the final schedule (Algorithm: line 1-5). A GRASP is terminated when the specified termination criterion is satisfied, for example, after completing a certain number of iterations. In general, there are two phases in each iteration: construction phase and local search phase.

Algorithm 3 GRASP algorithm

```
Input: A workflow
Output: bestSchedule

1. while stopping criterion not satisfied do
2.     schedule ← createSchedule(workflow)
3.     if schedule is better than bestSchedule then
4.         bestSchedule ← schedule
5. end while

6. procedure createSchedule(workflow)
7.     solution ← constructSolution(workflow)
8.     nSolution ← localSearch(solution)
9.     if nSolution is better than solution then
```

```
10.             return nSolution
11.         return solution
12.  end createSchedule

13.  procedure constructSolution(workflow)
14.         while schedule is not completed do
15.             T ← get all unmapped ready tasks
16.             make a RCL for each t ε T
17.             subSolution ← select a resource randomly for each t ε T from its RCL.
18.             solution ← solution U  subSolution
19.             update information for further RCL making.
20.         end while
21.         return solution
22.  end constructSolution

23.  procedure localSearch(solution)
24.         nSolution ← find a optimal local solution.
25.         return nSolution
26.  end localSearch
```

F. Comparison of best-effort scheduling algorithms

The overview of the best effort scheduling is listed in Table 1. In general, the heuristic based algorithms can produce a reasonable good solution in a polynomial time. Among the heuristic algorithms, individual task scheduling is simplest and only suitable for simple workflow structures such as a pipeline in which several tasks are required to be executed in sequential.

Unlike individual task scheduling, list scheduling algorithms set the priorities of tasks in order to make an efficient schedule in the situation of many tasks compete for limited number of resources. The priority of the tasks determines their execution order. The batch mode approach orders the tasks required to be executed in parallel based on their execution time whereas the dependency mode approach orders the tasks based on the length of their critical path. The advantage of the dependency mode approach is that it intent to complete tasks earlier whose interdependent tasks required longer time in order to reduce the overall execution time.

However, its complexity is higher since it is required to compute the critical path of all tasks. Another drawback of the dependency mode approach is that it cannot efficiently solve resource competition problem for a workflow consisting of many parallel tasks having the same length of their critical path. The dependency-batch mode approach can take advantage of both approaches, and Sakellariou and Zhao [25] shows that it outperforms the dependency mode approach in most cases. However, computing task priorities based on both batch mode and dependency mode approach results in higher scheduling time.

Table 1 Comparison of Best-effort Scheduling Algorithms

Scheduling Method	Algorithm	Complexity	Features
Individual task scheduling	Myopic	O(vm)	Decision is based on one task
Cluster based scheduling / Duplication based scheduling	TANH	O(v^2)	Replicating tasks to more than one resources in order to reduce transmission time
Greedy Randomized Adaptive Search Procedure (GRASP)	GRASP	Guided random search	Global solution obtained by comparing differences between randomized schedules over a number of iteration

V. CONCLUSIONS

The Load balancing scheduling algorithms in grid are compared. Particles fly in problem search space to find optimal or near-optimal solutions. One is based on ant colony optimization which is called AntX, the other one is based on particle swarm optimization and is called ParticleX
The workflow scheduling algorithms for Grid computing have categorized current existing Grid workflow scheduling algorithms as either best-effort based scheduling.

Best-effort scheduling algorithms target on community Grids in which resource providers provide free access. The comparison of these algorithms in terms of computing time, applications and resources scenarios has also been examined in detail. Since the service provisioning model of the community Grids is based on best effort, quality of service and service availability cannot be guaranteed. Therefore, we have also discussed several techniques on how to employ the scheduling algorithms in dynamic Grid environments. QoS constraint based scheduling algorithms target utility Grids in which service level agreements are established between service providers and consumers. In general, users are charged for service access based on their usage and QoS levels. The objective functions of QoS constraint based scheduling algorithms are determined by QoS requirements of workflow applications., we have focused on examining scheduling algorithms which intend to solve performance optimization problems based on two typical QoS constraints using GridSim..

REFERENCES

[1] Hesam Izakian, Behrouz Tork Ladani, Kamran Zamanifar, Ajith Abraham. A Novel Particle Swarm Optimization Approach for Grid Job Scheduling. Information Systems, Technology and Management Communications in Computer and Information Science Vol 31, 2009, pp 100-109.

[2] J. Cao, agent-based resource management for grid computing, Ph.D. Thesis, Department of Computer Science University of Warwick, London, 2001.

[3] R. Buyya, Economic-based Distributed Resource Management and Scheduling for Grid Computing, Ph.D. Thesis, School of Computer Science and Software Engineering Monash University, Melbourne, 2002.

[4] J. Cao, D.J. Kerbyson, G.R. Nudd, Performance Evaluation of an Agent-Based Resource Management Infrastructure for Grid Computing, in: Proceedings of 1st IEEE/ACM International Symposium on Cluster Computing and the Grid (2001) 311-318.

[5] E.G. Coffman Jr. (Ed.), Computer and Job-Shop Scheduling Theory, Wiley, New York, NY, 1976.

[6] A. Yarkhan, J. Dongarra, Experiments with scheduling using simulated annealing in a grid environment, in: 3rd International Workshop on Grid Computing (2002) 232–242.

[7] V. Di Martino, M. Mililotti, Sub optimal scheduling in a grid using genetic algorithms, Parallel Computing 30 (2004) 553–565.

[8] D. Liu, Y. Cao, CGA: Chaotic Genetic Algorithm for Fuzzy Job Scheduling in Grid Environment, Springer-Verlag Berlin Heidelberg (2007) 133–143.

[9] Y. Gao, H. Rong, J.Z. Huangc, Adaptive grid job scheduling with genetic algorithms, Future Generation Computer Systems 21 (2005) 151–161.

[10] A. Abraham, H. Liu, W. Zhang, T.G. Chang, Scheduling Jobs on Computational Grids Using Fuzzy Particle Swarm Algorithm, Springer-Verlag Berlin Heidelberg (2006) 500-507.

[11] A. Abraham, R. Buyya, B. Nath, Nature's heuristics for scheduling jobs on computational grids, In: 8th IEEE International Conference on Advanced Computing and Communications (ADCOM 2000), India, ISBN 0070435480, Tata McGraw-Hill Publishing Co. Ltd, New Delhi, pp. 45-52, 2000.

[12] B. Lowekamp, N. Miller, D. Sutherland, T. Gross, P. Steenkiste, and J. Subhlok. A Resource Query Interface for Network-Aware Applications. In Proceedings of the 7th IEEE Smposium on High-Performance Distributed Computing, July 1998.

[13] J. Stiles, T. Bartol, E. Salpeter, , and M. Salpeter. Monte Carlo simulation of neuromuscular transmitter release using MCell, a general simulator of cellular physiological processes. Computational Neuroscience, pages 279{284, 1998.

[14] M. Baker, R. Buyya and D. Laforenza, *Grids and Grid Technologies for Wide-area Distributed Computing*, in J. of Software-Practice & Experience, Vol. 32, No.15, pp: 1437-1466, December 2002.

[15] I. Foster, C. Kesselman and S. Tuecke, *The Anatomy of the Grid*: *Enabling Scalable Virtual Organizations*, in the International J. Supercomputer Applications, 15(3), pp. 200-220, fall 2001.

[16] I. Foster and C. Kesselman (editors), *The Grid: Blueprint for a Future Computing Infrastructure*, Morgan Kaufmann Publishers, USA, 1999.

[17] I. Foster and A. Iamnitchi, *On Death, Taxes, and the Convergence of Peer-to-Peerand Grid Computing*, in Proc. of 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03), Berkeley, CA, USA, February 2003.

[18] E. Aarts, J. Korst, Simulated Annealing and Boltzmann Machines, Wiley, New York, 1989.

[19] S. Kirkpatrick, C.D. Gelatt, J.M.P. Vecchi, Optimization by simulated annealing, Science 220 (1983) 671–680.

[20] F. Howell and R. McNab, *SimJava: A Discrete Event Simulation Package For Java With Applications In Computer Systems Modelling*, First International Conference on Web-based Modelling and Simulation, San Diego, CA, Society for Computer Simulation, January 1998.

[21] Y. Aridor, M. Factor, and A. Teperman, *cJVM: a Single System Image of a JVM on a Cluster*, Proceedings of the 29th International Conference on Parallel Processing (ICPP 99), September 1999, Fukushima, Japan, IEEE Computer Society Press, USA.

[22] T. Tannenbaum, D. Wright, K. Miller, and M. Livny, Condor – A Distributed Job Scheduler, Computing with Linux, The MIT Press, MA, USA, 2002.

[23] E. Tsiakkouri et al., “Scheduling Workflows with Budget Constraints”, In the CoreGRID Workshop on Integrated research in Grid Computing, S. Gorlatch and M. Danelutto (Eds.), Technical Report TR-05-22, University of Pisa, Dipartimento Di Informatica, Pisa, Italy, Nov. 28-30, 2005, pages 347-357.

[24] R. Bajaj and D. P. Agrawal, "Improving Scheduling of Tasks in a Heterogeneous Environment," IEEE Transactions on Parallel and Distributed Systems, vol. 15, pp. 107-118, 2004.

[25] F. Berman, H. Casanova, A. Chien, K. Cooper, H. Dail, A. Dasgupta, W. Deng, J. Dongarra, L. Johnsson, K. Kennedy, C. Koelbel, B. Liu, X. Liu, A. Mandal, G. Marin, M. Mazina, J. Mellor-Crummey, C. Mendes, A. Olugbile, M. Patel, D. Reed, Z. Shi, O. Sievert, H. Xia, and A. YarKhan, New Grid Scheduling and Rescheduling Methods in the GrADS Project, International Journal of Parallel Programming (IJPP), 33(2-3):209-229, 2005.

[26] S. Binato et al., A GRASP for job shop scheduling. In Ribeiro and Hansen, eds, Essays and surveys on metaheuristics, pp59-79, Kluwer Academic Publishers, 2001.

[27] J. Blythe et al., Task Scheduling Strategies for Workflow-based Applications in Grids, IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2005). IEEE Press.

[28] T. A. Feo and M. G. C. Resende, Greedy Randomized Adaptive Search Procedures, Journal of Global Optimization, 6:109-133, 1995.