Dual Objective Security Driven Scheduling Model for Computational Grid using GA

R. Kashyap, D.P. Vidyarthi

Abstract— **The conflict between achieving good performance, in terms of time etc., and achieving high quality of security protection introduces new challenges in security critical grid scheduling. Extensive study indicates that the scheduling performance is affected by the heterogeneities of security and computational power of resources. Different jobs may have varied security requirement and even the same security requirement may exhibit different security overhead on different nodes. This paper proposes a GA based dual objective scheduling algorithm, Dual Objective Security Driven Scheduling using Genetic Algorithm (DO-SDSG). Maximization of security offered to tasks with minimization of security overhead are the two objectives of DO-SDSG. Being a dual objective scheduling problem, it alternatively optimizes the objectives. When one objective is optimized the other one is taken as a constraint and vice-versa. The simulation study demonstrates that the proposed algorithm delivers better makespan, better security with less security overhead in comparison to other such algorithms viz. MinMin, MaxMin, SPMinMin and SPMaxMin.**

Index Terms- **Grid computing, Job scheduling, Genetic algorithm, Security, Security overhead.**

I. Introduction

A computational grid is a collection of geographically dispersed heterogeneous computing resources, giving the image of a single large virtual computing system to users [1] [2] [3]. Scheduling on such platform is an important and complex task more so being grid a heterogeneous system. The main challenge for job scheduling in grid is its highly dynamic environment, in which computing resources have their own access policies, security, availability etc. At the same time, resources are of greater heterogeneity in terms of their architectural design that includes desktop PCs to supercomputers. Thus, grid which is privately owned, heterogeneous, non dedicated network of computers utilizes the idle time of thousands of computing devices to harness the high performance computing power. From the users' point of view, the objective of the grid computing is to simplify distributed heterogeneous computing in the manner the World Wide Web has simplified information sharing over the Internet [4]. The key factors in making the grid computing feasible are: the evolution of standards such as TCP/IP and Ethernet; the ever-increasing bandwidth of networks; the increasing availability of idle CPU cycles on networked machines, and the emergence of Web services as a logical and open choice for software computing tasks[5][6]. Quite often, grid computing extensively supports collaborative projects on the internet. Most of these projects have stringent security requirements. Sometimes, the application itself imbibes security up to some extent, but more often it is to be supported and ensured by the grid environment. The dynamic and multi-institutional nature of the grid introduces challenging security threats that warrant the development of the new technical approaches towards this problem. In a security aware grid environment, responsibility is delegated to the scheduler for allocating the computational jobs on those resources that gives best possible security, while keeping in mind the computational and security heterogeneity of the resources.

The motivation of the grid computing is to aggregate the power of widely distributed resources to provide non-trivial Quality of Service (QoS) to users in which security is an important QoS parameter. To offer security as QoS, security choices must be offered to the user/application in form of Security Level (SL) and in turn user/application may request a certain level of security in form of Security Demand (SD). The underlying mechanism must be equipped to enter into an agreement for the services delivery at the requested security level. The notion of variant security requirement and security ranges at first seem strange, as the feeling was either to have security or ignore it at all. This is true at a gross scale since without some minimum level of security, a system will be considered inadequate for user's requirements. But if the user's minimum requirements are met, there may be some choices as to what is adequate. The second argument for the variant security would be; why a user would require anything less than the highest level of security. The answer lies in the associated cost which may be in form of monetary charges or even system performance degradation. There are many applications that specify their security demands and the service provider support them. Thus security services are quantified. Irvine and Levin, in their work, have emphasized the importance of quantifying the security services [7]. Sypropoulou and Agar [8] have worked on the quantification of the IPSec Protocol.

Task scheduling in grid is an NP-hard optimization problem, so many heuristic and meta-heuristic algorithms are in use aiming for suboptimal solutions. Meta-heuristics like Simulated Annealing (SA) [9], Genetic Algorithm (GA) [10][11], Ant Colony Optimization (ACO) [12], Particle Swarm Optimization (PSO) [13], etc. are also used for grid scheduling as they generally produce higher quality results than simple heuristics. Though these techniques may take a bit longer as they have to generate and evaluate many solutions not just one. These nature based meta-heuristics

R. Kashyap is with Lal Bahadur Shastri Institute of Management, Delhi, India (phone: 91-11-25307700; fax: 91-11-24522474; e-mail: rekhakashyap@lbsim.ac.in).

D. P. Vidyarthi is with School of Computer and Systems Sciences, Jawaharlal Nehru University, Delhi, India (dpv@jnu.ac.in).

follow the Darwin's natural selection principle i.e. only the fittest can survive. GA, a population-based meta-heuristic, was proposed by John Holland [11]. It is inspired by evolutionary biology and follows most of their characteristics such as inheritance, mutation, crossover, and selection. GA considers a solution as an organism, thus better the quality of the solution higher its survival probability. Solution evolves by applying genetic operators such as crossover (also called recombination) and mutation. GA can escape from the local optimal in search for the global optimal. In this paper, we use genetic algorithm for job scheduling to address the heterogeneity of security mechanism in a computational grid. The proposed Dual objective Security Driven Scheduling using Genetic algorithm (DO-SDSG) schedules the tasks in a way which improves the security of the heterogeneous grid with minimal security overhead.

This paper is organized in the following manner. Next section discusses some related works in this field. Scheduling strategy is described in section 3. Section 4 briefs the proposed security model of this work. Proposed model DO-SDSG is analyzed in section 5. Experimental results and observations for DO-SDSG and the compared heuristics are presented in section 6 epitomizing the work in the last section.

II. Related Work

In order to achieve the promising potential of underlying distributed resources in the grid, effective scheduling algorithms are fundamentally important. Scheduling has three phases: resource discovery, system selection and job execution [14].Effective grid computing is possible only if the resources are scheduled well. Scheduling tasks for the grid is an NP-hard problem as grid is a geographically dispersed heterogeneous multiprocessing environment. Consequent to this is the emergence of many heuristic and evolutionary approaches towards the scheduling problem [15] [16] [17] [18] [19] [20]. Some well known heuristic based grid scheduling algorithms, proposed in the literature, are as follows.

Casanova et al. [21] proposed an adaptive grid scheduling algorithm for parameter sweep applications, where tasks can share input files. It was extended to Sufferage heuristics as XSufferage. DFPLTF (Dynamic Fastest Processor to Largest Task First) is a scheduling heuristic which gives highest priority to the largest task [22]. Fujimoto and Hagihara [23] have proposed Round Robin (RR) grid scheduling algorithm for parameter sweep applications. MinMin and MaxMin are well known algorithms used in real world distributed resource management systems such as SmartNet [24]. MinMin gives highest priority to the task that can be completed first. In MinMin, the grid site offering the earliest completion time is tagged and the task that has the minimum earliest completion time is allocated to the respective site. MaxMin also tags the grid site that offers the earliest completion time but highest priority is given to the task that has maximum earliest completion time. All the above mentioned algorithms are not security aware and hence unsuitable for security aware applications.

The goal of a security aware scheduler is to meet the desired security requirements and at the same time offer a high level of performance with respect to one or more parameters e.g. makespan, average response time, site utilization etc. [25][26][27]. Further, security heterogeneity and the grid dynamism makes security aware grid scheduling more challenging as the security overhead is node dependent. Some of the security-aware schedulers discussed in the literature are as follows. Song, Kwok and Hwang [28] envision a secure scheduling framework with the risk involved while dispatching the jobs to the remote nodes. They proposed three scheduling strategies based on different risk levels and modified the MinMin and Sufferage heuristics in three modes; a) Secure mode (jobs were only scheduled to those nodes which can ensure security) b) Risky mode (jobs were scheduled to any available nodes without considering the risks between jobs and nodes), and c) F-risky mode (jobs were scheduled to available nodes to take at most F risks). SPMinMin and SPMaxMin [29] [30] are an improvement over the secure mode suggested by Song. In SPMinMin and SPMaxMin security requirement is the guiding parameter for scheduling decision and they guarantee the security of the job while minimizing the makespan. All above algorithms are security aware but makes no effort to optimize the security beyond the minimum security requirement.

SATS, suggested by Xie and Qin [31], takes into account heterogeneities in security and computation. SATS also provides a means of measuring overhead incurred by security services. It tries to improve security and minimize makespan. Xiaoyong et al. [32] incorporated security into inter-task dependency and proposed a security driven scheduling algorithm (SDS) to improve security of HDS (Heterogeneous distributed systems) while minimizing the makespan, risk probability and speedup. Proposed DO-SDSG works towards optimizing security similar to SATS, but being a GA based scheduler searches for the global optimum escaping local optimum.

Chao-Chin and Ren-Yi [33] proposed a GA based scheduling, addressing the heterogeneities of fault tolerant mechanism in the computational grid. They improved upon the job failure rate optimizing the makespan, whereas the proposed algorithm improves the total security value minimizing the security overhead.

The proposed work, DO-SDSG is an extension of earlier work, SDSG (Security Driven Scheduling using Genetic Algorithm) [39]. SDSG aims at maximizing security, restricting security overhead under a certain limit. DO-SDSG is a GA based dual objective scheduling algorithm, where the two considered objectives are; maximize the security offered to the tasks and minimize the security overhead. Being a dual objective scheduling problem, it alternatively optimizes the objectives. When one objective is optimized the other one is taken as a constraint and vice-versa.

III. SCHEDULING STRATEGY

The proposed model considers a grid consisting of number of non dedicated processing nodes which, in turn, may have a single processor or a group of heterogeneous or homogeneous processors. A job is comprised of *"n"* independent tasks with different computational size and security level. The tasks have soft deadlines and are independent i.e. without any precedence constraint. The list of terminologies, used in this paper, is as follows.

A task T_i is characterized as $T_i = (Sz_i, SL_i)$ where, Sz_i is the computational size in terms of millions of instructions and SL_i is the security level assigned to the i^{th} task.

Processing node N_j of the grid is characterized as N_j= (SP_j, BT_j) where, SP_j is the speed of node in MIPS and BT_j is the begin time of the node (time to execute already assigned tasks to the node).

A schedule of the job is depicted as the set of 5 tuples $<T_i, P_j, BT_j, SO_{ij}, CT_{ij}>$ in which, T_i is i^{th} task, P_j is j^{th} processing node, BT_j is Begin Time at j^{th} processing node, SO_{ij} is the Security Overhead of i^{th} task on j^{th} node and CT_{ij} is the completion time of the i^{th} task on the j^{th} processing node. CT_{ij} is calculated as in equation 1.

$$CT_{ij} = BT_j + ET_{ij} + SO_{ij} \qquad (1)$$

Here, ET_{ij} is Execution Time of i^{th} task on j^{th} processing node and SO_{ij} is the Security Overhead of i^{th} task on j^{th} node. Begin time of every node at the start of schedule is assumed to be zero but once the execution start, it will be affected.

IV. SECURITY STRATEGY

A. Security Model

One of the key factors behind the growing interest in grid computing is the evolution of standards such as TCP/IP and Ethernet in networking. For the TCP networking model, IPsec, TLS/SSL and SSH are the popularly used security protocols operating on its network, transport and application layer respectively [34] [35] [36] [37]. These protocols offer security to any grid application by the common security services of key exchange, authentication, confidentiality and integrity. Each protocol is further configured to match differing security requirements through cipher suite negotiations where cipher suite, is a named combination of key exchange, authentication, encryption, and integrity algorithms used to negotiate the security settings for a network connection. In the present work, SSL V3 protocol is considered and security levels are assigned for the cipher suites supported by it. *SL* for each cipher-suite is based on the weighted sum of security services involved in the cipher-suite. Cipher-suite offering more security (algorithms with longer keys) has more computational cost and therefore is assigned a higher security level. The security level also provides a mechanism for calculating the security overhead expenses. Subset of cipher suites supported by SSL V3 protocol is given in Table 1. The third row SSLCipherSpec SSL_RSA_WITH_DES_CBC_SHA indicates use of DES with 56-bit encryption. The fourth row SSL CipherSpec SSL_RSA_WITH_3DES_ EDE_ CBC_SHA indicates use of 3DES with 168-bit encryption. Among the six cipher suites, mentioned in the Table 1, the first one provides the weakest security and the last one provides the strongest security.

B. Security Overhead Computation

Security overhead is calculated as suggested by Tao Xie and Xiao Qin [38] (equation 2), where, SL_i is in the range [1, 2, 3…..R] and 1 and R are the lowest and highest security level. For the experiments, shown in this paper, R is set to be 15. The rationale behind this security overhead model is based on the observation that the security overhead of a particular application tends to be proportional to the execution time of the application. In other words, security overhead depends upon the amount of data to be secured and thus is the product of the execution time (which depends upon data size) and relative security required as shown in equation 2. Xie Tao [30] and Xiaoyong Tang [31] have proposed more precise model for calculating security overhead in which they calculate security overhead for each security service namely authentication, integrity and confidentiality. Simpler security overhead calculation has been effectuated in all the algorithms since for the comparison purposes; the result will not be affected. Total computation time considering security overhead is shown in equation 3.

$$SO_{ij} = ET_{ij}\left(SL_i / R\right) \qquad (2)$$

$$CT_{ij} = BT_j + ET_{ij}(1 + SL_i /R) \qquad (3)$$

TABLE 1
THE SUBSET OF CIPHER SUITES SUPPORTED BY SSL V3 PROTOCOL

SSLCipherSpec	SSL_RSA_WITH_ RC4_128_MD5	Security Level 1
SSLCipherSpec	SSL_RSA_WITH_ RC4_128_SHA	Security Level 2
SSLCipherSpec	SSL_RSA_WITH_ DES_CBC_SHA	Security Level 3
SSLCipherSpec	SSL_RSA_WITH_3 DES_EDE_CBC_SH A	Security Level 4
SSLCipherSpec	SSL_RSA_EXPORT _WITH_RC4_40_M D5	Security Level 5
SSLCipherSpec	SSL_RSA_EXPORT _WITH_RC2_CBC_ 40_MD5	Security Level 6

V. THE PROPOSED WORK

This work proposes a dual objective genetic algorithm, DO-SDSG, which aims at maximizing the security value of the job with minimal total security overhead. An important decision in such dual objective optimization is how to evaluate the quality of solutions since the conflicting and incommensurable nature of some of the criteria makes this process more complicated. The possible alternatives are as follows.

- *Combine the objectives:* This is one of the classical methods to evaluate the solution fitness in multi-objective optimization. It refers to converting the multi-objective problem into a single-objective by combining the various criteria into a single scalar value. The most common way of doing this is by setting weights to each criterion and add them all together using an aggregating function.
- *Pareto-based evaluation***:** In this approach, a vector containing all the objective values represents the solution fitness and the concept of dominance is used to establish preference between solutions. A solution *x* is said to be non inferior or non-dominated if there is no other solution that is better that *x* in all the criteria.
- *Alternating the objective:* This is also an approach that has been used for quite some time. It refers to optimizing one criterion at a time while imposing constraints on the others. The most challenging task in such optimization problem is to decide upon the value of constraints.

The proposed GA based scheduler, DO-SDSG, alternates the two objectives: total security offered and security overhead. It works in two phases. Each phase performs the crossover, mutation and selection operations to produce new solutions. Each solution is associated with a fitness value, used to evaluate the quality of that particular solution. The three operations are repeated number of times, called generations, until the termination condition is reached. In the first phase we optimize (maximize) the security offered considering security overhead as a constraint. The constrained value of security overhead for the first phase is the average security overhead over randomly created 500 schedules. After a fixed number of generations the algorithm terminates yielding the best possible schedule. In the second phase, security overhead is optimized (minimized) while total security offered is the constraint. The total security achieved by the best solution of the first phase is used as the constrained value of the security in this phase, i.e. the solutions having total security less than the constrained value are allowed to survive. The steps followed in the two phases are elaborated below.

DO-SDSG─Phase 1

This is the first phase where we optimize (maximize) the security offered considering security overhead as a constraint. The various steps involved in it are discussed below.

A. Coding of Solutions

The encoding of individuals (also known as chromosome, solution string etc.) of the population is a key issue in genetic algorithm. In DO-SDSG, a fixed length integer number encoding is used where feasible solution is encoded in a vector called schedule. Chromosome is of the size equal to the number of jobs to be scheduled.

Each element of the vector is an ordered pair (*NodeID, SecurityLevel*) as shown in Fig. 1. The ith entry, in the schedule, indicates that the i^{th} *task* is scheduled on the node with identity equal to *NodeID* and will be executed with the security having value equal to S*ecurityLevel*. For example, 3rd entry (4, 6) in the schedule (Figure 1) indicates that the 3rd task is scheduled on the node with *NodeID* 4 and offered *SecurityLevel* value equal to 6. For each schedule *Node ID* is randomly generated within the permissible range of nodes. Security level is randomly generated between the ranges as shown in equation 4.

$$SDMin_i \le SL_i \le SLMax_j \qquad (4)$$

Here, SL_i is the security offered to the i^{th} task, $SDMin_i$ is the minimum security demanded by the i^{th} task and $SLMax_j$ is the max security which the j^{th} node can offer.

Task ID	1	2	3	4	5	6
	2,3	3,5	4,6	3,1	4,3	2,8

Fig.1 The fixed length integer number encoding pattern of chromosomes

B. Population

For initial population, we keep on generating random schedules till we fetch 100 schedules having total security overhead within constrained value denoted as SO_{ctr}. It is obtained by averaging the security overhead from randomly generated 500 schedules and is calculated as shown in equation 5 and 6.

$$SO_{ctr} = \sum_{K=1}^{m} SO_k \Big/ m \qquad (5)$$

$$SO_k = \sum_{i=1}^{n} SO_{ij}, \forall j \qquad (6)$$

Here, SO_k is the total security overhead of the k^{th} schedule, m is total number of schedules and n is the number of tasks comprising the schedule.

C. Fitness function and Selection

Fitness function is one of the important components of GA to measure the quality of solution and is problem dependent. In the first phase of DO-SDSG, we maximize the security of the solution with security overhead as a constraint. If the constraint is violated the solution is infeasible and has no fitness. Thus fitness of the schedule is measured as the total security realized by the tasks. Equation 7 shows the fitness function.

$$Maximize\ Fit(f(x)) = \sum_{i=1}^{n} SL_i \qquad (7)$$

$$\text{subject to } SO(x) \le SO_{ctr}, i = 1, 2, \ldots, n$$

$$\text{where } SO(x) = \sum_{i=1}^{n} SO_{ij}, \forall_j$$

Here, SL_i is the security level of i^{th} task, $SO(x)$ is the total security overhead of the entire schedule x, SO_{ij} is the security overhead of the i^{th} task on the corresponding j^{th} node and n is the total number of tasks to be scheduled.

Parents for the next generation are selected after computing the fitness, $Fit(f(x))$ of each chromosome in the current population. For selection, roulette wheel selection is used which is similar to the roulette in the gambling games. The selection operator allows the algorithm to take biased decision favoring good individuals through generations. To accomplish this, good individuals are replicated while bad individuals are removed. As a consequence, post selection population is likely to be dominated by good schedules. Each individual is assigned an interval proportional to its fitness and is selected if the randomly drawn number belongs to its interval. Better the fitness, better the possibility of its being selected. Selection procedure involves following steps.

- Calculate the selection probability of schedule x denoted as $P(x)$ using equation 8.

$$P(i) = \frac{Fit(f(i))}{\sum_{i=1}^{N} f(x)} \quad ,i=1,2,..,n \qquad (8)$$

- Calculate partial sum of fitness values (equation 9).

$$partSum(i) = partSum(i-1) + P(i),\ i = 1,2,\ldots,n \qquad (9)$$

- Finally, generate a random number R between 0 and 1. If $partSum(i-1) < R < partSum(i)$, then i is a

probable parent. The operation is repeated till we obtain *N* probable parents for the next generation.

D. Crossover and Mutation

Genetic algorithms are based on principles that crossing two individual can result in offspring's normally better than both the parents. Crossover is a recombination operator that combines subparts of the two parent chromosomes to produce offspring that contain some parts of genetic material from both the parents. We have adopted single-point crossover method with probability $p_c=0.7$. Firstly, parents are selected based on the above mentioned selection scheme, then a crossover point is randomly selected, and we exchange chromosome beyond this point. Since the genes of our chromosome are ordered pair of *Node ID* and *Security Level*, crossover not only changes the task node relationship but also the security value associated with it and may result in a new schedule whose security overhead exceeds the SO_{ctr}. Thus only schedules whose security overhead does not exceed SO_{ctr} are allowed to participate in crossover.

After selection and crossover, mutation is performed to lead the search to get out of local optimum. The mutation operation randomly selects a gene in a chromosome, and then mutates its value. In our case the mutation operator changes the node and security value of a randomly selected task in an arbitrary chromosome with probability $p_m=0.06$. The schedules whose security overhead exceeds SO_{ctr} post mutation is brought back to its pre mutation state.

E. Termination

The entire process of selection, crossover and mutation is repeated till the algorithm converges or 1000 generations are reached (whichever happens earlier), and candidates having best fitness in the final generation is considered as the best solution.

DO-SDSG– Phase II

In the second phase security overhead is optimized (minimized) while total security offered is kept as the constraint. The various steps involved in it are discussed below. The total security achieved by the best solution of the first phase is used as the constrained value of security for this phase.

A. Coding of Solutions

The coding of solutions is same as used in the first phase.

B. Initial Population

The best schedule obtained from the first phase is taken as one of the candidates for the initial population in this phase. The remaining population is randomly generated till we get 99 more schedules whose total security is within the constrained value (TS_{ctr}). The constrained value for the security is obtained from the first phase. It is the total security value attached with the best solution obtained from first phase and is calculated as shown in equation 10.

$$TS_{ctr} = \sum_{i=1}^{n} SL_i \text{ , n=total number of tasks.} \qquad (10)$$

C. Fitness function and Selection

In the second phase of SDSG security overhead is minimized keeping total security as a constraint. The fitness of the schedule is the total security overhead incurred by all the tasks and is measured using equation 11.

$$minimize\ Fit(f(x)) = \sum_{i=1}^{n} SO_{ij}, \forall_j \qquad (11)$$

$$subject\ to\ SL(x) \le TS_{ctr}$$

$$where\ SL(x) = \sum_{i=1}^{n} SL_i$$

Here, SO_{ij} is the security overhead of i^{th} task on the corresponding j^{th} node, $SL(x)$ is the total security of schedule x and SL_i is Security level of i^{th} task. After we compute the fitness $Fit(f(x))$ of each chromosome in the current population, parents for the next generation are selected using roulette wheel as was done in the first phase.

D. Crossover and Mutation

The crossover and mutation operator for this phase are applied in the same way, as in the first phase.

E. Termination

The entire process of selection, crossover and mutation is repeated till the algorithm converges or 1000 generations are reached (whichever happens earlier), and candidates having best fitness in the final generation is considered as the best final solution.

VI. Experimental Results and Observations

In this section, we describe the experimental settings, performance criterion and simulation results for the proposed and studied algorithms. The comparative performance study of DO-SDSG has been conducted with five more scheduling algorithms, i.e. MinMin, MaxMin, SPMinMin, SPMaxMin and SDSG. A brief description of these algorithms is given below.

- **MinMin–** gives highest priority to the task that can be completed first. In this, for each task the grid site that offers the earliest completion time is tagged and the task that has the minimum earliest completion time is allocated to the respective node. MinMin executes shorter task in parallel whereas longer task follows the shorter one[24].
- **MaxMin–** here the grid site that offers earliest completion time is tagged. Highest priority is given to the task with maximum earliest completion time. The idea behind Max-Min is overlapping long running task with short running ones. MaxMin executes many shorter tasks in parallel with the longer one [24].
- **SPMinMin–** Security Prioritized MinMin(SPMinMin) allocates highest security demanding tasks first on the faster resources. Tasks having same security requirement are then scheduled according to MinMin.

- **SPMaxMin–** Security Prioritized MaxMin (SPMaxMin) allocates highest security demanding tasks first on the faster resources. Tasks having same security requirement are then scheduled according to MaxMin(SA).
- **SDSG–** is a security driven scheduling using genetic algorithm (SDSG) which aims at maximizing security while restricting security overhead under a certain limit.

The grid simulator, used for the study, is implemented in java. It consists of Grid-Generator and Task-Generator. Grid generator, based on some defined range, generates the grid. Task generator produces the task. Properties of resources in the simulations are random and uniformly distributed among a predefined range of resource parameters as depicted in Table 2. For random value generation, the random function of java API is used with current time as the seed value. Experimental study considers the complete heterogeneous environment in terms of security offered by the nodes, speed of the nodes, size of the task, and security demand of the task. Since MinMin and MaxMin are not security-offering scheduling algorithms; we imposed security overhead cost in their implementations for a fair comparison with DO-SDSG, SDSG, SPMinMin and SPMaxMin. The aim of DO-SDSG is to maximize the security offered to the tasks with minimal security overhead. Being a dual criterion scheduling problem, we alternatively optimized the criteria in two phases. The schedule offering the best solution in the final generation of the second phase is selected as the optimal schedule. DO-SDSG improves the security value over generations and finally settles for the best possible in the termination phase. For the comparisons to be fair, the other algorithms are made to perform for the same security values as used in DO-SDSG over the following performance metrics:

- *Security Overhead (extra computational expenses for securing the data)* = $SO_{ij} = ET_{ij}\left(SL_i / R\right)$
- *Makespan* (completion time of the entire job) = $Max\ [CT_{ij}]\ i=1, 2,\ldots n.$
- *Average response time* (time period between the task arrival and its completion time) = $\sum_{i=1}^{n} (CT_{ij} - AT_i) / n$.

Here, ET_{ij} is Execution Time of i^{th} task on j^{th} processing node and SO_{ij} is the Security Overhead of i^{th} task on j^{th} node.SL_i is the security level offered to the i^{th} task. CT_{ij} is the completion time of the i^{th} task on the assigned j^{th} processor= $CT_{ij} = BT_j + ET_{ij} + SO_{ij}$, AT_i is the arrival time of the i^{th} task and n is number of tasks in a schedule.

Makespan reflects the entire job efficiency whereas average response time indicates the performance of majority of the tasks within the schedule. A better makespan is an indication that the schedule does not suffer and a better average response time suggests that majority of tasks does not suffer.

TABLE 2
INPUT PARAMETERS FOR THE SIMULATION EXPERIMENTS

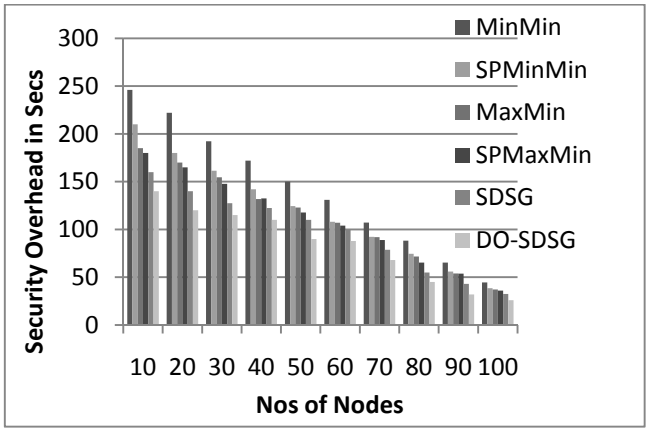
Parameter	Value Range
No of nodes	30
Speed of the processing node (SP)	1, 2, 5, 10 (MIPS)
Security level of the processing node (SL)	1 to 15
No. of tasks	100 to 1500 (fixed 150)
Size of tasks	10 to 1000 (MB)
Population size	100
Max generations	1000
Crossover probability(p_c)	0.7
Mutation probability(p_m)	0.06

A. Performance Impact by varying Number of Tasks

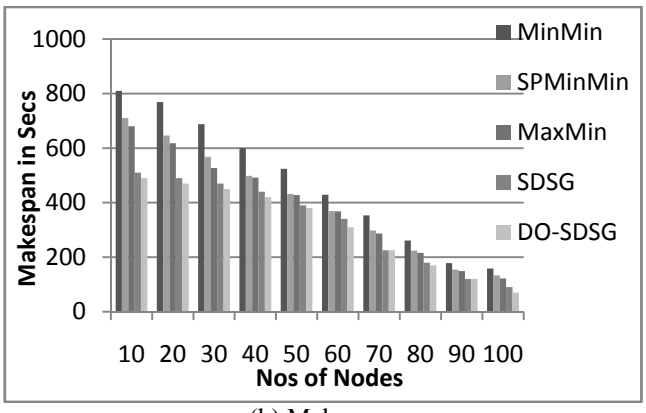
Security overhead, makespan and average response time of DO-SDSG is compared with other heuristics by varying scheduled number of tasks from 100 to 1500 on a grid with 50 heterogeneous nodes. In fig 2, the experimental results show that when the number of tasks increases, the time for finishing the tasks also increases. It is also observed that DO-SDSG performs better than all other algorithms, achieving less security overhead and makespan for similar security values. Since MinMin and MaxMin are not security aware algorithm, making them run with higher security values resulted in much higher security overhead and makespan. Although SPMinMin and SPMaxMin are security aware algorithms and give priorities to higher security demanding tasks but they do not optimize security overhead. DO-SDSG considers the best solution of SDSG as one of the inputs in its second phase, thus the results of DO-SDSG is atleast better than SDSG as shown in the result. The improvement of DO-SDSG over other algorithms, for makespan and security overhead is better with increase in the number of tasks as shown in fig. 2(a) and 2(b). Average response time measures the overall wait time for the entire task set. Since MinMin and SPMinMin gives priority to smaller tasks, they show better response time and DO-SDSG is the third best among all the algorithms compared for average response time metric (fig. 2(c)).

B. Performance Impact by varying Number of Nodes

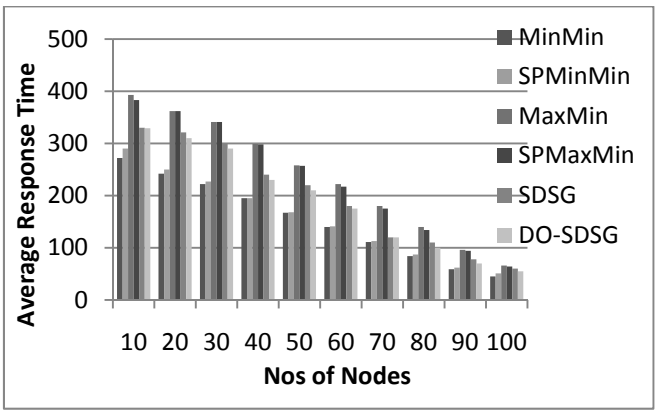
DO-SDSG is compared with other algorithms by varying number of nodes from 10 to 100, for 1000 heterogeneous tasks. The parameters defining the nodes and tasks are randomly generated between the ranges defined in Table 2. It is observed that for all the models there is an improvement in makespan and security overhead when more nodes are available to schedule same number of tasks. It seems to be intuitive. It is further observed that even if the number of nodes is scaled, DO-SDSG gives better performance of makespan and security overhead as it has a better scheduling approach to schedule tasks to the appropriate computing nodes (fig. 3(a) and 3(b)). Average response time of MinMin is better than all other algorithms irrespective of number of nodes present in the grid. DO-SDSG and SDSG showed better response time than MaxMin, and SPMaxMin.



(a) Security Overhead

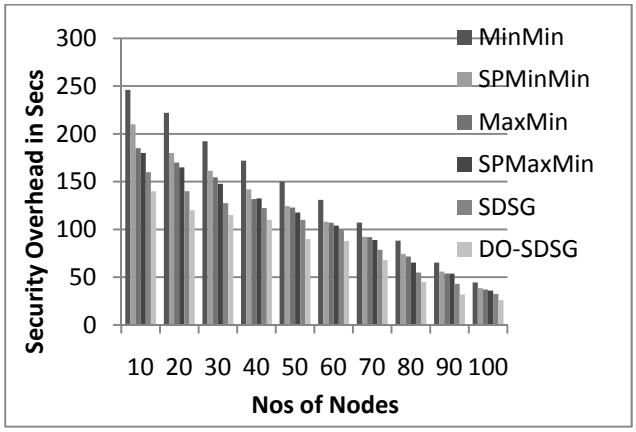


(b) Makespan

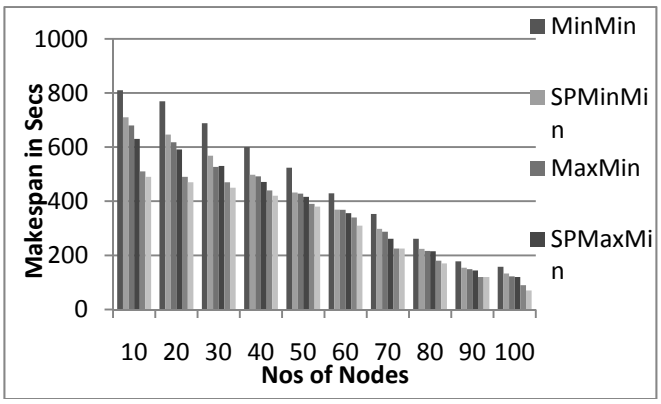


(c) Average response Time

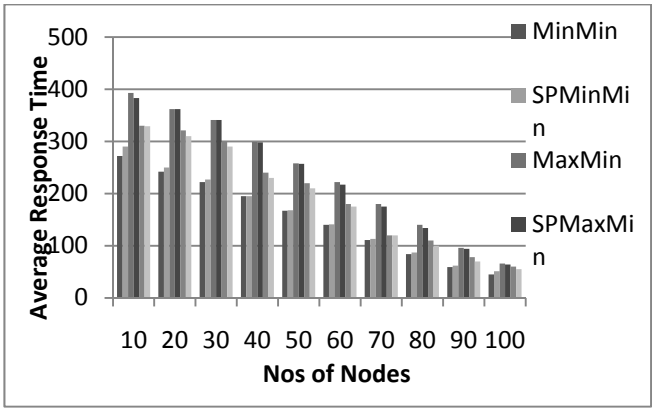
Fig. 2. Performance comparisons varying the number of tasks



(a) Security Overhead



(b) Makespan



(c) Average response Time

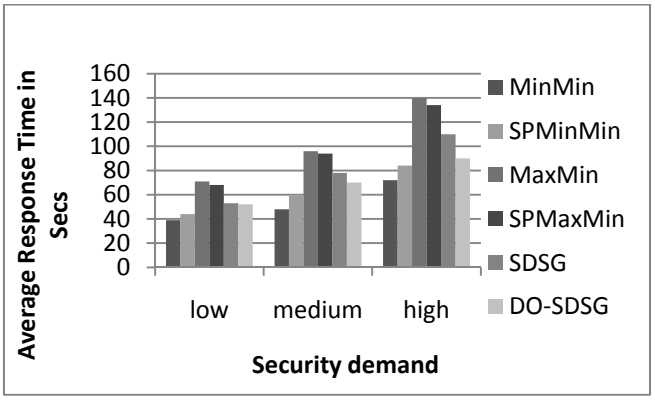
Fig. 3. Performance comparisons varying the number of Nodes

C. Performance Impact by varying Security Demand

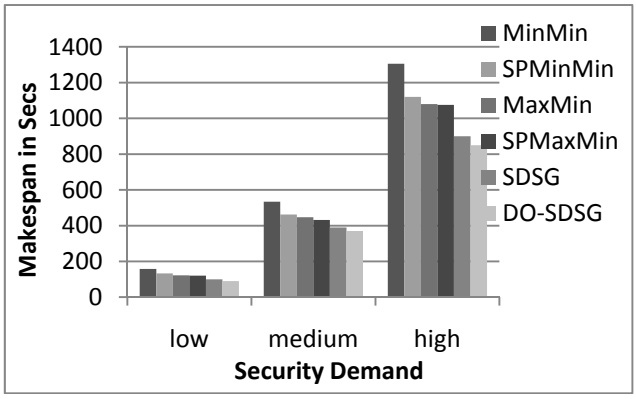
Next, we compared the security overhead, makespan and average response time of DO-SDSG with other heuristics when 500 tasks were scheduled on 30 heterogeneous nodes, while varying security from low to high. The configuration of low to high is done as shown in Table 3. Performance of DO-SDSG was found to be better than other algorithms for security overhead and makespan performance metric for all levels of security demand. It is further observed that with the increase in security demand DO-SDSG showed much improvement over other algorithms for security overhead and makespan (fig. 4(a) and 4(b)). The reason being: DO-SDSG optimizes on security overhead by allowing only those solutions to survive which are within a constrained value of security overhead and tries to explore further for solutions having lesser security overhead.

Table 3. Security Value assignment

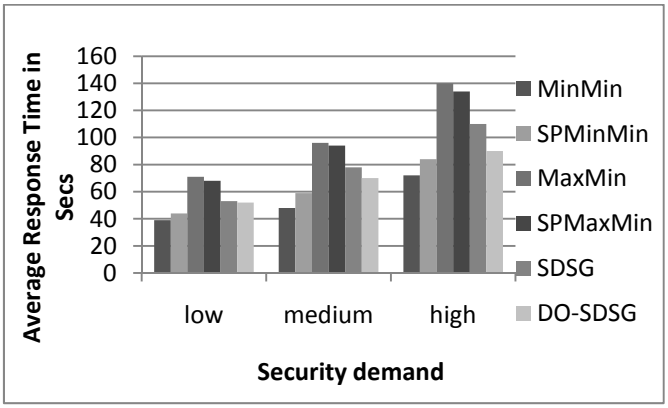
Security Level	Value
1 to 5	low
6 to 10	medium
11 to 15	high



(a) Security Overhead



(b) Makespan



(c) Average response Time

Fig. 4. Performance comparisons varying the Security Demand

VII. Conclusion

The paper proposes a GA based scheduling algorithm for large computational grid, which makes efforts to incorporate security into task scheduling. Non security aware algorithm do not consider security overhead and security constraints of a task and therefore possibly assign the task to a node that only result in small computation time but with a large total execution time (which is sum of computation time and security overhead). In most of the results obtained in the graph, it is obvious that DO-SDSG performs better than other similar algorithms. Moreover, it optimizes the security incurring lesser security overhead. GA, due to its very nature, is capable of exploiting and exploring in the whole range of solution search space globally and picking near optimal scheduling solution. The proposed DO-SDSG being security aware genetic algorithm makes effort to optimize quality of security and at the same time satisfy high level of performance metric i.e. security overhead. Experimental results confirm that DO-SDSG performs better than other compared heuristics giving better makespan and security overhead for same level of security.

References

[1] Foster I. Kesselman C. Tsudik G. Tuecke S. Security Architecture for Computational Grids. ACM Conference on Computers and Security 1998; 83-91.

[2] Foster I. Kesselman C. Tuecke S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International Journal of High Performance Computing Applications 2001; pp. 200-222.

[3] Foster I. "What is Grid? A three point checklist, Paper (Grid Today,1(6).)" http://dlib.cs.odu.edu/WhatIsTheGrid.pdf [2002].

[4] Berman, F., Chien, A., Cooper, K., Dongarra, J., Foster, I.,Dennis Gannon, L. J., Kennedy, K., Kesselman, C., Reed,D., Torczon, L., and Wolski, R.(2001). 'The GrADSproject: Software support for high-level grid application development'. International Journal of High PerformanceComputing Applications 15(4): pp 327–344.

[5] Naedele, M. (2003) 'Standards for XML and Web Services Security, Computer' vol.36, No.4, pp 96-98.

[6] Prabhakar, S., Ribbens, C. & Bora, P.(2002) 'Multifaceted web services: An approach to secure and scalable grid scheduling Scheduling'. Proceedings of UROWEB..

[7] Irvine, C. E. and Levin, T. E.(1999) 'Toward a Taxonomy and Costing Method for Security Services', Proceedings of the 15th Computer Security Application Conference, Phoenix, AZ. (1999).

[8] Sypropoulou, E., Agar, C., Levin, T. E., and Irvine, C. E.(2002) 'IPsec Modulation for Quality of Security Service', Proceedings of the International System Security Engineering Association Conference, Orlando Florida.

[9] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by Simulated Annealing," *Science,* vol. 220, no. 4598, pp. 671-680,May 1983.

[10] J. H. Holland, *Adaptation in Natural and Artificial Systems.* AnnArbor, MI: Univ. of Michigan Press, 1975.

[11] Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison Wesley, 1989.

[12] E. Bonabeau, M. Dorigo and G. Theraulaz, "Inspiration for Optimizationfrom Social Insect Behavior," *Nature,* vol. 406, pp. 39-42,Jul. 2000.

[13] J. Kennedy and R. C. Eberhart, "Particle Swarm Optimization,"*Proc. IEEE Int'l Conf. Neural Networks (ICNN 95),* Perth, Australia,pp. 1942-1948, Nov. 1995.

[14] Schopf, J. M.(2004) 'Ten Actions When Grid Scheduling — The User as a Grid Scheduler'. In Grid Resource Management — State of the Art and Future Trends, J. Nabrzyski, J. Schopf, and J. Weglarz (eds.), pp. 15–23. Kluwer Academic Publishers

[15] Doulamis, N. Doulamis A. Varvarigos E. Varvarigou T.(2007) 'Fair scheduling algorithms in grids', IEEE Transactions on Parallel and Distributed Systems 18 (11), pp 1630-1648.

[16] Hai, Z., Yuping, W.(2008) 'Security-Driven Task Scheduling Based on Evolutionary Algorithm. International Conference on Computational Intelligence and Security'.

[17] Braun, T., Hensgen, D,. Freund, R., Siegel, H., Beck, N., Boloni, L., Maheswaran, M., Reuther, A., Robertson, J., Theys, M., Yao, B.(2001). 'A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems', Journal of Parallel and Distributed Computing, pp 810-837.

[18] Abawajy, J. An efficient adaptive scheduling policy for high performance Computing, Future Generation Computer Systems 25 (3), 364-370., (2009).

[19] Kalantari, M., Akbari, M. K.(2009) A parallel solution for scheduling of real time applications on grid environments, Future Generation Computer Systems 25 (7), pp704-716.

[20] Kun-Ming, Y., Cheng-Kwan, C.(2008) 'An Adaptive Scheduling Algorithm for Scheduling Tasks in Computational Grid', Seventh International Conference on Grid and Cooperative Computing .

[21] Casanova, H. and Dongara, J.(1996) ' NetSolve: A Network Server for Solving Computational Science Problems'. In Proceedings of the 1996 ACM/IEEE Supercomputing Conference'.

[22] Paranhos, D., Cirne, W., & Brasileiro, F.(2003) 'Trading cycles for information: Using replication to schedule bag-of-tasks applications on computational grids'. In International Conference on Parallel and

Distributed Computing (Euro-Par), Lecture Notes in Computer Science, volume 2790, pp169–180.

[23] Fujimoto, N., & Hagihara, K.(2003) 'Near-optimal dynamic task scheduling of independent coarse-grained tasks onto a computational grid'. In 32nd Annual International Conference on Parallel Processing (ICPP-03), pp. 391–398.

[24] Freund, R. F., Gherrity, R. M., Ambrosius, S., Campbell, M. Halderman, D. Hensgen, E. Keith, T. Kidd, M. Kussow, , Lima, J. D., Mirabile, F. L., Moore, L., Rust, B., & Siegel, H. J.(1998) 'Scheduling resources in multi-user, heterogeneous, computing environments with smartnet', In the 7th IEEE Heterogeneous Computing Workshop (HCW'98),pp. 184–199.

[25] Welch, V., Siebenlist, F., Foster, I., Bresnahan, J., Czajkowski, K., Gawor, J., Kesselman, C., Meder, S., Pearlman, L., Tuecke, S.(2003) 'Security for Grid Services, Proc. Int'l Symp. High Performance Distributed Computing (HPDC-12).

[26] Xie, T., Qin, X. (2005) "Enahancing Security of Real-Time Applications on Grids through Dynamic Scheduling", Proc. 11th Workshop Job Scheduling Strategies for Parallel Processing JSSPP; pp146-158.

[27] Xie, T., Qin, X.(2008) 'Security-Aware Resource Allocation for RealTime Parallel jobs on Homogeneous and Heterogeneous Clusters'. In IEEE Transactions on Parallel and Distributed systems, Vol. 19, No. 5.

[28] Song, S., Kwok, Y., K. & Hwang, K.(2005) "Trusted Job Scheduling in Open Computational Grids: Security-Driven Heuristics and A Fast Genetic Algorithms," Proc. Int'l Symp. Parallel and Distributed Processing.

[29] Kashyap, R., Vidyarthi, D. P. (2009) 'Security Prioritized Computational Grid Scheduling Model: An Analysis. International Journal of Grid and High Performance Computing', 1(3), pp. 73-84 (2009).

[30] Kashyap, R., Vidyarthi, D. P.(2009) 'A Security Prioritized Scheduling Model for Computational Grid'. In International conference at HPC Asia. pp 416-424.

[31] Xie, T., Qin, X.(2007) 'Performance Evaluation of a New Scheduling Algorithm for Distributed Systems with Security Heterogeneity'. J. Parallel Distributed. Computing ; pp1067– 1081.

[32] Xiaoyong, T., Kenli, L., Zeng, Z., Bharadwaj, V.(2010) 'A Novel Security-Driven Scheduling Algorithm for Precedence Constrained Tasks in Heterogeneous Distributed Systems', IEEE Transaction on computers Vol. 6, No. 1.

[33] Chao-Chin W. Ren-Yi S.(2010) 'An integrated security-aware scheduling strategy for large-scale computational grids'. Future Generation Computer Systems, pp198-206.

[34] Luo Q. Lin Y. Analysis and Comparison of Several Algorithms in SSL/TLS Handshake Protocol **.** Proceedings of the International Conference on Information Technology and Computer Science 2009.

[35] Stallings W. Cryptography and Network Security: Principles and Practice, 4/E. Prentice Hall: 2008.

[36] Salter M. Rescorla E. Housley R. RFC 5430 Suit B Profile for Transport layer Security and TLS version 1.2. http://tools.ietf.org/html/rfc5430 [March 2009].

[37] Dierks T. Rescorla E RFC 4346 The Transport layer Security (TLS) Protocol Version 1.1. http://tools.ietf.org/pdf/rfc4346.pdf [April 2006].

[38] Xie, T., Sung, A., Qin, X.(2005) 'Dynamic Task Scheduling with Security Awareness in Real-Time Systems', Proceedings of the 19th International Parallel and Distributed Processing Symposium (IPDPS'05), the 4th Int'l Workshop on Performance Modeling, Evaluation, and Optimization of Parallel and Distributed Systems, IEEE/ACM, Denver, CO.

[39] Kashyap R., Vidyarthi, D.P. , Security-Driven Scheduling Model for Computational Grid using Genetic Algorithm , Lecture Notes in Engineering and Computer Science: Proceedings of The World Congress on Engineering and Computer Science 2011, WCECS 2011, 19-21 October, 2011, San Francisco, USA, pp 382-387.