Grid Scheduling using PSO with Naive Crossover

Vikas Singh
ABV- Indian Institute of Information Technology and Management,
GwaliorMorena Link Road,
Gwalior, India

Deepak Singh
Raipur Institute of Technology Raipur
Raipur, India

Shyam Swarup
ABV- Indian Institute of Information Technology and Management, Gwalior
Morena Link Road,
Gwalior, India

ABSTRACT
Grid computing can be defined as applying the resources of many computers in a network to a problem which requires a great number of computer processing cycles or access to large amounts of data. Thetask scheduling problem is the problem of assigning the tasks in the system in a manner that will optimize the overall performance of the application, while assuring the correctness of the result. In this paper we use the technique of PSO with Naive crossover to solve the taskscheduling problem in grid computing. The aim of using thistechnique is use the given resources optimally and assign the task to the resources efficiently. The simulated results show that PSO with Naive Crossover proves to be a better algorithm when applied to resource allocation anddisk scheduling in grid computing.

Keywords
Grid scheduling, PSO with Naive Crossover Operator, Tasks, Resources

1. INTRODUCTION
Grid computing is a network that is not in the same place but distributed resources such as computers, peripherals, switches, instruments, and data. Grid computing appears to be a promising trend for three reasons:

1. Its ability to make more cost-effective use of a given amount of computer resources.
2. It is a way to solve problems that can't be approached without an enormous amount of computing power.
3. Because it suggests that the resources of many computers can be cooperatively and perhaps synergistically harnessed and managed as collaboration toward a common objective.

Grid computing can be used to compute large number of tasks on the resources which are geographically remotely located. Task scheduling is a challenging problem in grid computing environment [6]. Many parallel applications consist of multiple computational components. While the execution of some of these components or tasks depends on the completion of other tasks, others can be executed at the same time, which increases parallelism of the problem.

Abraham et al and Braun et al [1] address the dynamic scheduling of jobs to the geographically distributed computing resources. They provide an introduction of computational grids followed by a brief description of the three nature's heuristics namely Genetic Algorithm (GA), Simulated Annealing (SA) and Tabu Search (TS). Yang gao et al [7] proposed two algorithms that use the predictive models to schedule jobs at both system level and application level. M Aggrawal et al [2] presented a Genetic Algorithm based scheduler. The proposed scheduler used in both the intra-grid of a large organization and in a research grid consisting of large clusters, connected through a high bandwidth dedicated network. Shanshan Song et al [11] proposed a new Space-Time Genetic Algorithm (STGA) for trusted job scheduling in which they consider three security-driven heuristic modes: secure, risky and f -risky. Lee Wang et al [12] developed a heuristic based approach to matching and scheduling in heterogeneous computing environment. In Lin JianNing et al [8] scheduling-algorithm based on genetic algorithm (GA) is addressed.

In this paper simulated results prove that PSO with Linear Crossover proves to be better when it is applied for resource allocation in the field of grid computing. This paper is organized as follows. In section 2 the issues related to task scheduling is discussed. In section 3 particle swarm algorithm is introduced. In section 4 we discuss a need of a new algorithm i.e. PSO with Linear Crossover. In section 5 we implement PSO with Linear Crossover in our problem. In section 6 we see the simulation result of work done in section 5.

2. TASK SCHEDULING ISSUES IN GRID COMPUTING
Computational grid can be combination of hardware and software that can be used to solve complex computational problems. The resource in a computational grid can be anything which can be used to solve the given problem. For example a set of printers which are used for printing a set a documents. The overall objective of task scheduling is to minimize the completion time and to utilize the resources effectively and usually it is easy to get theinformation about the ability to process data of the available resource.[1][13].

The problem of task scheduling arises in a situation where there are more tasks than the available resources. Consider a scenario wherein there arex={1,2,3,4….X} tasks to be done and there are y={ 1,2,3,4…..Y} resources available. With the condition that the task is not allowed to migrate between resources. In such a situation if we have y> x then there is no reason for developing new algorithms for task scheduling because then resources can be allocated to the tasks on first come first serve basis, but if y< x then we need to develop new algorithms for task scheduling because now inefficient resource allocation can greatly hamper the efficiency and throughput of the scheduler. To formulate the problem, define T_a ={1,2,3,….X} as x independent tasks permutation and R_b={1,2,3,y} as y computing resources.

Suppose that the processing time $P_{a,b}$ for task a computing on b resource is known. The completion time F(x) represents the total cost time of completion[13]. The objective is to find an permutation matrix m=(M_{ab}), such that:

Mab= 1if resource a performs task b
elseMab =0
which minimizes total cost:

$$F(x) = \sum\sum Pab * Mab \quad (1)$$

subject to $\sum Mab = 1; \forall b \in T$ (2)

$$Mab \in \{0,1\}, \forall a \in R; \forall b \in T \quad (3)$$

The minimal F(x) represents the length of schedule whole tasks working on available resources. The scheduling constraints (2) guarantee that each task is assigned to exactly one resource.

3. PARTICLE SWARM OPTIMISATION

The particle swarm optimization algorithm, originally introduced in terms of social and cognitive behavior by Kennedy and Eberhart (1995) [9], solves problems in many fields, especially engineering and computer science. The individuals, called particles henceforth, are flown through the multi-dimensional search space with each particle representing a possible solution to the multi-dimensional optimization problem. Each solution's fitness is based on a performance function related to the optimization problem being solved. The movement of the particles is influenced by two factors using information from iteration-to-iteration as well as particle-to-particle. As a result of iteration-to- iteration information, the particle stores in its memory the best solution visited so far, called p_{best}, and experiences an attraction towards this solution as it traverses through the solution search space. As a result of the particle-to-particle interaction, the particle stores in its memory the best solution visited by any particle, and experiences an attraction towards this solution, called g_{best} , as well. The first and second factors are called cognitive and social components, respectively. Afteriteration, the$p_{best}$and $g_{best}$are updated for each particle if a better or more dominating solution (in terms of fitness) is found. This process continues, iteratively, until either the desired result is converged upon, or it is determined that an acceptable solution cannot be found within computational limits. For an n-dimensional search space, the i^{th} particle of the swarm is represented by an n-dimensional vector, $X_i = (x_{i1}, x_{i2} \ldots x_{in})^T$. The velocity of this particle is represented by another n-dimensional vector $V_i = (v_{i1}; v_{i2} \ldots v_{in})^T$. The previously best visited position of the ith particle is denoted as $P_i = (p_{i1}, p_{i2}, \ldots p_{in})^T$. `g'is the index of the best particle in the swarm. The velocity of the i^{th} particle is updated using the velocity update equation given by

$$v_{id} = v_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id}) \quad (4)$$

and the position is updated using

$$x_{id} = x_{id} + v_{id} \quad (5)$$

where $d = 1, 2 \ldots n$; $i = 1; 2 \ldots S$, where S is the size of the swarm; c_1 and c_2 are constants, called cognitive and social scaling parameters respectively (usually, $c_1 = c_2$; r_1 , r_2 are random numbers, uniformly distributed in [0, 1]). Equations (4) and (5) are the initial version of PSO algorithm. A constant, V_{max}, is used to arbitrarily limit the velocities of the particles and improve the resolution of the search. Further, the concept of an inertia weight was developed to better control exploration and exploitation. The motivation was to be able to eliminate the need for V_{max} . The inclusion of an inertia weight (*w)* in the particle swarm optimization algorithm was first reported in the literature in 1998 (Shi and Eberhart, 1998) [13]. The resulting velocity update equation becomes:

$$v_{id} = w * v_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id}) \quad (6)$$

Eberhart and Shi (2000) [14] indicate that the optimal strategy is to initially set *w* to 0.9 and reduce it linearly to 0.4, allowing initial exploration followed by acceleration toward an improved global optimum.

4. PSO WITH NAIVE CROSSOVER OPERATOR

Particle swarm optimization is a population based heuristic search technique. PSO uses iterative process to search the global optima in solution space. Crossover operator with PSO has a property of better exploration in initial generations soby using crossover search area is explored in a relatively better manner even in later generations. PSO has a higher convergence rate, by using crossover with PSO premature convergence is also reduced so that PSO does not get trapped in local optima. Crossover can help the particles to jump out of the local optima by sharing the others' information. Two Particles generated by PSO are randomly selected for crossover operation and two new offspring are formed. The best offspring (in terms of fitness) is selected from the new offspring. This new best offspring replaces the worst parent particle which is selected for crossover. The replacement is done if the new best offspring has the good fitness value than the parent particle.

4.1 Naive Crossover

Naive crossover operator [3] produces two off springs from a pair of parents by randomly selecting a cross site between 1 and n, parents solution dimension and replacing the former and latter half of each parent from the cross site. Let $(x_1^{(1,t)} x_2^{(1,t)} x_3^{(1,t)} x_4^{(1,t)} \ldots x_n^{(1,t)})$ and $(x_1^{(2,t)} x_2^{2,t)} x_3^{(2,t)} x_4^{(2,t)} \ldots x_n^{(2,t)})$ are two parent solutions of dimension *n* at generation *t.* A cross site of value 3 will produce the offsprings shown in equation (7) and (8).

Offspring 1: $(x_1^{(1,t)} x_2^{(1,t)} x_3^{(1,t)} x_4^{(2,t)} \ldots x_n^{(2,t)})$ (7)

Offspring 2: $(x_1^{(2,t)} x_2^{(2,t)} x_3^{(2,t)} x_4^{(1,t)} \ldots x_n^{(1,t)})$ (8)

5. PROPOSED METHODOLOGY

In this paper we have proposed a solution for grid scheduling using PSO with Naive Crossover operator. For solving any optimization problem we have to first formulate the problem according to optimization problem.

5.1 Individual Representation

To solve the problem, representation of the individual and fitness value is required, so we have to first represent the grid scheduling problem in terms of PSO with Naive Crossover

operator. In grid scheduling we have a set of tasks and a set of resources as input and a sequence, which informs that which task is to be operated on which resource and in which order as output. PSO with Naive Crossover is based on population concept and each individual in population represents a solution, in case of grid scheduling problem, solution is asequence of tasks which are to be performed. So we have to first formulate each individual of PSO with Naive Crossover. We took dimension as a number oftask and value as an initial sequence from the possible set of sequences to find optimal sequence, we represent task set as $X_{id}=\{t_1,t_2,t_3....t_x\}$ where i is the particular individual and d represents the dimension index and set of resources $R_{lm}=\{r_1,r_2,r_3....r_y\}$, where *l* represents a particle/ sequence and m represents the tasks which are assigned to a resource. Task id and resource id is given to each task and each resource so that they can be easily differentiated from one other. Such as t_1 is the task id of first task and r_1 is the resource id of firstresource. Here x represents the total number of tasks. For eg.if we have 10 taskswhich are to be performed on 5 available resources, then we have dimension value as 10.
Assuming that we have a dimension value={9,4,7,0,3,6,1,2,8,5}. Here
9 represent value for first dimension of an individual which indicates 10^{th} task.
4 represent value for second dimension of an individual which indicates 5^{th} task.
7 represent value for third dimension of an individual which indicates8^{th} task.
0 represents value for fourth dimension of an individual which indicates 1^{st} task.
3 represent value for fifth dimension of an individual which indicates4^{th} task.
6 represent value for sixth dimension of an individual which indicates 5^{th} task.
1 represents value for seventh dimension of an individual which indicates 2^{nd}task.
2 represent value for eighth dimension of an individual which indicates 3^{rd} task.
8 represent value for ninth dimension of an individual which indicates 9^{th} task.
5 represent value for tenth dimension of an individual which indicates 6^{th} task.
Then using the formula

$R_{lm} = X_{id} \bmod M$ i.e. value of Task set mod Total resources (9)

This fomula is used to determine the associated resources for the calculated tasks in the sequence. we can calculate the resource set as {4,0,3,2,1,2,3,1}. From this set, we can interpret that

Task 9 is operated by resource 4,
Task 4 is operated by resource 0,
Task 7 is operated by resource 3,
Task 0 is operated by resource 0,
Task 3 is operated by resource 3,
Task 6 is operated by resource 2,
Task 1 is operated by resource 1,
Task 2 is operated by resource 2,
Task 8 is operated by resource 3,
Task 5 is operated by resource 1

Fitness Function After representation of each individual we have to calculate fitness value of each individual. In case of grid scheduling problem optimal solution is the minimization the value of equation(2). Our main objective is to minimize the fitness value, an individual who have the minimum fitness value is considered as the optimal solution.

5.2 Algorithm: Grid Scheduling Using PSO with Naive Crossover Operator

PHASE 1[Initialization Phase]
for s = 0 to Swarmsize do
for d = 0 to DimensionSize do
Randomly initialize particle with sequence of tasks
Compute resource for that particle/sequence
end for d
Compute fitness of that particle/sequence
Compute global best
end for s

PHASE 2[Update Phase]
repeat
fors = 0 to Swarmsize do
for d = 0 to ProblemDimension do
A new sequence is generated
computeresources for that particle/sequence
end for d
compute fitness of updated particle
if needed update historical information for global best(Pg)
end for s

PHASE 3[Crossover Operator Phase]
ifcrossover criteria is met then then
Select two random particles from current swarm for crossover operation
Apply crossover operation to generate new particle
New offspring generated from parents as a result of crossover.
Compute the resources for that offspring using (6)
compute the fitness of updated particle
Replace the worst parent particle with new best offspring if it is better
ifneeded update the historical information of global best(Pg)
end if
untilStopping Criteria is not met

To solve the grid scheduling problem we have used the Particle Swarm Optimization (PSO) with Naive crossover operator. We set an initial population by selecting a random starting sequence from the set of x! sequences; where x is the total number of tasks. After getting the initial particle we calculate fitness value of each particle, according to equation(2). After that we calculate best among the entire particle and set it as an initial global best.PSO update equation is used to update old population and generate new sequences and then their resources are calculated. These sequences, along with their resources are then used to findthe fitness value of each individual of each particle of the population. After this if crossover criteria is satisfied, then crossover operation performed over two randomly selected particle and as a result a new sequence is generated. Then the resource of this offspring is calculated. Using the sequence and its resources the fitness value of the offspring is calculated. Based on the fitness value, if theoffspring is better than its worst parent then this particle replaces that parent.

6. EXPERIMENTAL RESULTS AND DISCUSSION

This section focuses on the efficiency of the proposed algorithm PSO with linear crossover to solve grid task scheduling. This section shows the experimental results and the parameter settings of the proposed algorithm.

6.1 Experimental Setup

For every algorithm there are some control parameters which are used for its efficient working. Hence, there are some control parameters for PSO with Naive Crossover operator also. We did an extensive literature survey and carried out our own experiments for determining the values of these control parameters. From this we found that the values which we have taken in this experiment are standard values and they are also suitable for this experiment.The first control Parameter is Maximum function evaluation and the value of this parameter we have taken in our experiment as 20,000. The next parameter in our experiment is maximum number of population and we have taken its value to be 40. Another control parameter is number of runs and we have taken its value in our experiment as 30. It must be noted that each run contains maximum function evaluation, which is 20,000 in our experiment. The fourth control parameter is Dimension and it depends upon the number of tasks. The next control parameter is the value of c1 & c2 which we have taken as 1.14. And w (Inertia weight) is also a control parameter and we have taken its value as 0.7.

In this experiment we are using the feature of naive crossover operator in the PSO algorithm. The control parameter for Crossover operator is Probability. Therefore we need to find the value of this parameter also. Its value can range from 0.1 to 0.9. In our experiment we get the best result when the value of probability is 0.8.

6.2 Experimental Results

In this section we analyze the result obtained by our algorithm. To test the efficiency of our algorithm results of PSO with Naive crossover is compared with PSO algorithm results. In a grid scheduling task we already have the information about the number of resources, number of tasks, and the amount of time that will be taken by a resource to complete a task. We just need to find the sequence which will provide us the optimal results. We conducted the experiment by varying the number ofresources as well as varying the number of tasks and then we compared our results with that of PSO. In particular, we have taken three cases in which we have taken different number of resources and tasks.

Here, we are provided with 3 resources and 10 tasks. Given below are the execution time of PSO and PSO with Naive crossover operator with differing probability.

From the table, it can be concluded that by using Naive Crossover operator we get better results when the probability is 0.3, 0.4 & 0.8. The execution timetaken by PSO is 1158 units. The minimum execution time taken by the PSO with Naive crossover operator is 993 units. The sequence generated when the probability used in Naive Crossover operator was 0.8 is 9,0,5,1,2,3,4, 6, 8, 7.

Table 1. Execution time calculated by PSO and PSO with Naive crossover operator for 10 tasks by 3 resources

	Probability used in PSO with Naive crossover operator								
PSO	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
1158	1292	1208	1101	1108	1241	1205	1205	993	1270

Here, we are provided with 5 resources and 17 tasks. Given below are the execution time of PSO and PSO with Naive crossover operator with differing probability.

Table 2. Execution time calculated by PSO and PSO with Naive crossover operator for 17 tasks by 5 resources

	Probability used in PSO with Naive crossover operator								
PSO	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
2258	2521	2461	2452	2376	2380	2097	2334	2293	2293

From the table, it can be concluded that by using Naive Crossover operator we get better results when the probability is 0.6. The time taken by the PSO is 2258 units and the minimum time taken by PSO with naive operator is 2097 units. The sequence generated when the 0.6 probability was used in Naive Crossover operator is 13,6,0,8,7,10,12,13,4,6,14,16,0,1,2,3,5.

Here, we are provided with 10 resources and 27 tasks. Given below are the results of PSO and PSO with Naive crossover operator with differingprobability.

Table 3. Execution time calculated by PSO and PSO with Naive crossover operator for 27 tasks by 10 resources

	Probability used in PSO with Naive crossover operator								
PSO	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
3670	3410	3774	3674	3674	3606	3726	3726	3726	3606

From the table, it can be concluded that by using Naive Crossover operator we get better results when the probability is 0.1, 0.5, and 0.9. The time taken by PSO is 3670 units while the minimum execution time taken by PSO with Naive crossover operator is 3410 units. The sequence generated when the 0.1 probability was used in Naive Crossover operator is 11,24,16,18,6,8,13,20,7,12,19,5,25,23,26,0,1,9,14,21, 2,10,22,15,16,4,17,3.

7. CONCLUSION

It can be concluded from the results that proposed PSO with Naive crossover operator performs better than the existing PSO algorithm. There is no specific value for crossover probability for which we can obtain best results for grid scheduling. It depends upon number of tasks and number of resources. As future work we have the intention to apply other types of nature inspired algorithms to the grid scheduling problem, comparing their results with the ones accomplished by the PSO with Naïve.

8. REFERENCES

[1] Abraham, A., Buyya, R., Nath, B.: Natures heuristics for scheduling jobs on computational grids. In: The 8th IEEE International Conference on Advanced Computing and Communications (ADCOM 2000). pp. 45{52. Citeseer (2000)

[2] Aggarwal, M., Kent, R., Ngom, A.: Genetic algorithm based scheduler for computational grids (2005)

[3] Deb, K.: Multi-objective optimization using evolutionary algorithms. Wiley (2001)

[4] Eberhart, R., Shi, Y.: Comparing inertia weights and constriction factors in particle swarm optimization. In: Evolutionary Computation, 2000. Proceedings of the 2000Congress on. vol. 1, pp. 84{88. IEEE (2000)

[5] Eberhart, R., Shi, Y., Kennedy, J., Corporation, E.: Swarm intelligence. Elsevier (2001)

[6] Foster, I., Kesselman, C.: The grid: blueprint for a new computing infrastructure. Morgan Kaufmann (2004)

[7] Gao, Y., Rong, H., Huang, J.: Adaptive grid job scheduling with genetic algorithmsFuture Generation Computer Systems 21(1), 151{161 (2005)

[8] Jian-Ning, L., Hui-Zhong, W.: Scheduling in Grid Computing Environment Basedon Genetic Algorithm [J]. Journal of computer research and development 12 (2004)

[9] Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Neural Networks,1995. Proceedings., IEEE International Conference on. vol. 4, pp. 1942{1948. IEEE(1995)

[10] Shi, Y., Eberhart, R.: A modi_ed particle swarm optimizer. In: Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence.The 1998 IEEE International Conference on. pp. 69{73. IEEE (1998)

[11] Song, S., Kwok, Y., Hwang, K.: Security-driven heuristics and a fast genetic algorithm for trusted grid job scheduling (2005)

[12] Wang, L., Siegel, H., Roychowdhury, V., Maciejewski, A.: Task matching and scheduling in heterogeneous computing environments using a genetic-algorithm based approach. Journal of Parallel and Distributed Computing 47(1), 8{22 (1997)

[13] Zhang, L., Chen, Y., Sun, R., Jing, S., Yang, B.: A task scheduling algorithm based on pso for grid computing. International Journal of Computational IntelligenceResearch 4(1), 37{43 (2008)