International Journal of the Physical Sciences Vol. 6(24), pp. 5682-5687, 16 October, 2011
Available online at http://www.academicjournals.org/IJPS
DOI: 10.5897/IJPS11.1034
ISSN 1992 - 1950 ©2011 Academic Journals

Full Length Research Paper

Differential evolution algorithms for grid scheduling problem

Amid Khatibi Bardsiri[1] and Marjan Kuchaki Rafsanjani[2]*

[1]Bardsir Branch, Islamic Azad University, Kerman, Iran.
[2]Department of Computer Science, Shahid Bahonar University of Kerman, Kerman, Iran.

Accepted 6 September, 2011

Differential evolution (DE) has recently emerged as simple and efficient algorithm for global optimization over continuous spaces. DE shares many features of the classical genetic algorithms (GA). But it is much easier to implement than GA and applies a kind of differential mutation operator on parent chromosomes to generate the offspring. Grid computing aims to allow unified access to data, computing power, sensors and other resources through a single virtual laboratory. Scheduling is a key problem in emergent computational systems, such as grid and P2P, in order to benefit from the large computing capacity of such systems. In this paper, we present differential evolution algorithm based on schedulers for efficiently allocating jobs to resources in a grid computing system. Several variations for DE are examined in order to identify which works best for the grid scheduling problem.

Key words: Grid computing, differential evolution, crossover probability, scale factor, population.

INTRODUCTION

The emerging paradigm of grid computing and the construction of computational grids are making the development of large scale applications possible from optimization and other fields (Foster and Kesselman, 1998). The development or adaptation of applications for grid environments is being challenged by the need of scheduling a large number of jobs to resources efficiently. Moreover, the computing resources may vary in regard to their objectives, scope and structure as well as to their resource management policies, such as access and cost. Grid computing and distributed computing, dealing with large scale and complex computing problems, is a hot topic in the computer science and research (Minhas et al., 2011). Mixed-machine heterogeneous computing (HC) environments utilize a distributed suite of different machines, interconnected with computer network, to perform different computationally intensive applications that have diverse requirements (Tracy et al., 1998). Miscellaneous resources should be orchestrated to perform a number of tasks in parallel or to solve complex tasks atomized to variety of independent subtasks (Baca, 1989). Various sciences can benefit from the use of grids to solve CPU-intensive problems, creating potential benefits to the entire society. Indeed, the grid environment is dynamic and, also the number of resources to manage and the number of jobs to be scheduled are usually very large making thus the problem a complex large scale optimization problem. Task scheduling is an integrated part of parallel and distributed computing. Intensive research has been done in this area and many results have been widely accepted. With the emergence of the computational grid, new scheduling algorithms are in demand for addressing new concerns arising in the grid environment. In the heterogeneous nature of grid the job has to wait in the queue, the waiting time of the job in the queue depends on the following factors, such as load and availability of the resources. Task scheduling is mapping a set of tasks to a set of resources to efficiently exploit the capabilities of such resources. It has been shown, that an optimal mapping of computational tasks to available machines in an HC suite is a NP-complete problem (Fidanova and Durchova, 2006) and as such, it is a subject to various heuristic and meta-heuristic algorithms. The heuristics applied to the task scheduling problem include Min-min heuristic, Max-min heuristic, longest job to fastest resource- shortest job to fastest resource (LJFR-SJFR) heuristic, sufferage heuristic, work queue heuristic and

*Corresponding author. E-mail: kuchaki@mail.uk.ac.ir.

```
1.  Population initialization X(0) ← {x_1(0), ..., x_m(0)}
2.  g ← 0
3.  Compute {f(x_1(g)), ..., f(x_m(g))}
4.  While the stopping condition is false do
5.     For  i = 1 to m do
6.        y_i ← generate Mutant(X(g))
7.        z_i ← Crossover(x_i(g), y_i)
8.        If f(z_i) < f(x_i(g)) then
9.           x_i(g + 1) ← z_i
10.       Else
11.          x_i(g + 1) ← x_i(g)
12.       End if
13.    End for
14.    g ← g + 1
15.    Compute {f(x_1(g)), ..., f(x_m(g))}
16. End while
```

Figure 1. A summary of differential evolution.

2	3	1	6	4	5	1	13

Figure 2. Job-to-Resource representation (chromosome).

others (Izakian et al., 2009; Braun et al., 2001; Munir et al., 2007). In this work, several DEs are proposed and efficiently implemented in Matlab environment using a generic approach based on a skeleton for DEs (Price et al., 2005). The implementation has been extensively tested to identify a set of appropriate values for the parameters that conduct the search. We have used the benchmark of Braun et al. (2001). We have done extensive experimenting and fine tuning of parameters and have thus, identified the configuration of operators and parameters that outperforms existing implementations for static instances of the problem. We aim to minimize the completion time (makespan). Note that the makespan is the most important parameter of the grid scheduling problem.

DIFFERENTIAL EVOLUTION

Differential evolution (DE) is a reliable, versatile and easy to use stochastic evolutionary optimization algorithm. DE is a population-based optimizer that evolves real encoded vectors representing the solutions to given problem. The DE starts with an initial population of N real-valued vectors. The vectors are initialized with real values either randomly or so, that they are evenly spread over the problem domain (Price and Storn, 1995). During the optimization, DE generates new vectors that are pertubations of existing population vectors. The algorithm perturbs vectors with the scaled difference of two randomly selected population vectors and adds the scaled random vector difference to a third selected population vector to produce so called trial vector. The trial vector competes with a member of the current population with the same index. If the trial vector represents a better solution than the population vector, it takes its place in the population. Differential evolution is parameterized by two parameters (Price and Storn, 1997a). Scale factor F∈ (0, 1) controls the rate at which the population evolves and the crossover probability C∈ [0, 1] determines the ratio of bits that are transferred to the trial vector from its opponent. Figure 1 shows the pseudo code of DE algorithm (Zaharie, 2007).

The idea of differential evolution algorithm is to obtain a new individual by adding the weighted difference vector of any two individuals to another individual with certain rules. In the proposed scheduling algorithm, the solution is represented as an array of length equal to the number of jobs. The value corresponding to each position i in the array represent the resource to which job i was allocated. The job-to-resource representation (Chromosome) is illustrated in Figure 2.

Assume schedule *S* from the set of all possible schedules, Sched. Each chromosome has a fitness value, which is the makespan that results from the matching of tasks to machines within that chromosome. For differential evolution, we define a fitness function *fit(S): Sched* → *R* that evaluates each schedule:

$$fit(S) = Makespan(S) \quad (1)$$

We have implemented differential evolution for scheduling of independent tasks on heterogeneous independent environments.

Problem definition

The main goal of the task scheduling in grid computing systems is the efficiently allocating tasks to machines. Tasks that originate from different users/applications are independent. We use the expected time to compute (ETC) matrix model introduced by Ali et al. for formulating the problem (Ali et al., 2000). It is assumed that an accurate estimate of the expected execution time for each task on each machine is known prior to execution, and contained within an ETC matrix. ETC[*i,j*] is the expected execution time of task *i* in machine *j*. Using the ETC matrix model, the scheduling problem can be defined as follows:

1. A number of independent tasks to be allocated to the available resources; because of non-preemptive scheduling, each task has to be processed completely in a single machine.

Table 1. A summary of DE parameters.

Parameter	Value
Population size	30
Probability of crossover	0.3
Scaling factor	0.9
Initial population	Random

2. The number of machines available to participate in the allocation of tasks.
3. Ready[m] represents the ready time of the machine after completing the previously assigned tasks.
4. ETC matrix of size $n \times m$, where n represents the number of tasks and m represents the number of machines.

A meta-task is defined as a collection of independent task (that is, task does not require any communication with other tasks) (Barun et al., 1998). Tasks derive mapping statically. For static mapping, the number of tasks, n and the number of machines, m is known a priori. Assume that $C_{i,j}$ ($i \in \{1,2,...,n\}$, $j \in \{1,2,..., m\}$) is the completion time for performing ith task in jth machine and W_j ($j \in \{1,2,...,m\}$) is the previous workload of M_j, then Equation 2 shows the time required for M_j to complete the tasks included in it. According to the aforementioned definition, makespan can be estimated using Equation 3 (Xhafa and Abraham, 2007).

$$\sum C_j + W_j \quad (2)$$

$$Makespan = \max_{j \in 1,...,m} \{\sum C_j + W_j\} \quad (3)$$

RESULTS AND DISCUSSION

In grid scheduling systems, a major challenge is to manage the consumers' job based on their quality of service (QoS) and provider nodes' satisfaction. Most of the capable job scheduling polices operate on the basis of meta-scheduling systems (Bouyer et al., 2011). Despite the efforts that current grid schedulers with various scheduling algorithms have made to provide comprehensive and sophisticated functionalities, they have difficulty guaranteeing the quality of schedules they produce (Lopez and Raja, 2011). Here, we apply a powerful population based meta-heuristic algorithm from the differential evolution to the task scheduling problem. We have implemented DE in Matlab environment by adapting the algorithmic skeletons defined (Price et al., 2005). Though, dynamic scheduling is our eventual aim, using static instances, we are able to see the quality of the schedule produced by our DE implementation.

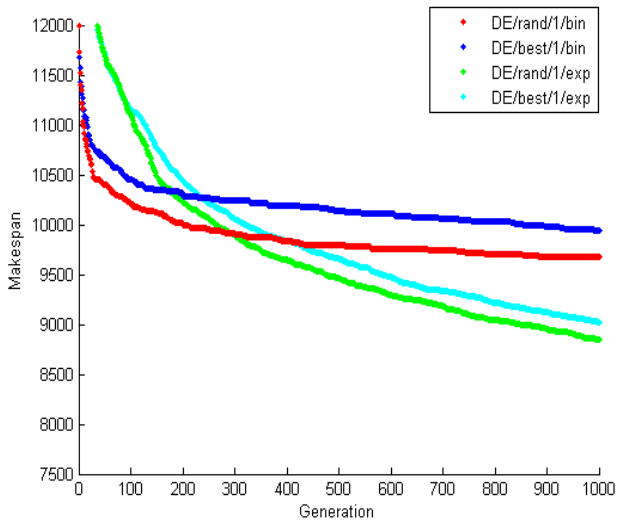

Figure 3. Comparison of different DE strategies.

Moreover, it is very useful in finding an appropriate combination of operators and parameters that work well in terms of robustness. The experimental results discussed subsequently were obtained on a PC with 2 GHz processor and 2 GB of RAM. In this paper, we used the benchmark proposed (Braun et al., 2001). The simulation model is based on expected time to compute (ETC) matrix for 512 tasks and 16 machines, therefore each chromosome is a 512 × 1 vector. We use i-lo-lo matrix for all cases and give the tables of values for the parameters and for the each case graphical representation is given.

Strategy

The differentiation operation can be realized by many search strategies. Actually, it is the process of mutation which demarcates one DE scheme from another (Adeyemo and Otieno, 2009). We can now have an idea of how the different DE schemes are named. The general convention is DE/x/y/z. DE stands for differential evolution, x represents a string denoting the type of the vector to be perturbed (whether it is randomly selected or it is the best vector in the population with respect to fitness) and y is the number of difference vectors considered for perturbation of x. Each mutation strategy was combined with either the 'exponential' type crossover or the 'binomial' type crossover (Price, 1999); z stands for the type of crossover being used. From the tuning of parameters, we obtained the values of parameters given in Table 1. For these values, the resulting DEs behaviors are graphically shown in Figure 3.

The relative performance order of the strategies from best to worst was: (1) rand/1/exp, (2) best/1/exp, (3) rand/1/bin and (4) best/1/bin.

Table 2. Values of parameters for comparing crossover probability performance.

Parameter	Value
Strategy	DE/rand/1/exp
Population size	30
Scaling factor	0.9
Initial population	Random

Table 3. Parameter values used for comparing the performance of scale factor.

Parameter	Value
Strategy	DE/rand/1/exp
Population size	30
Probability of crossover	0.1
Initial population	Random

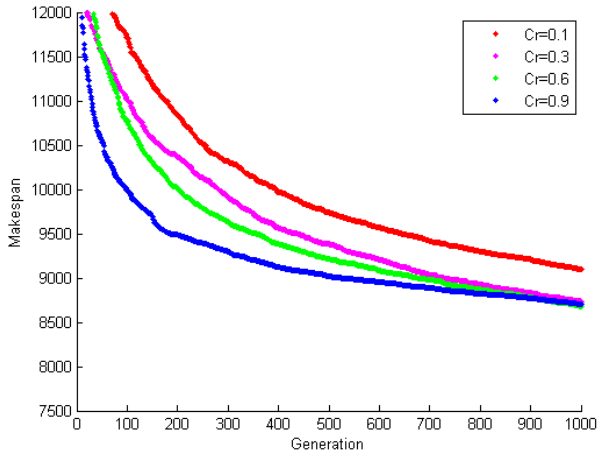

Figure 4. Comparison of the performance of crossover probability.

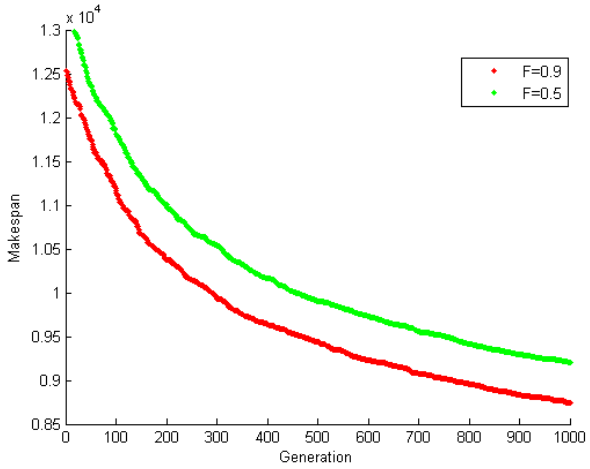

Figure 5. Comparison of the performance of scale factor.

Crossover probability

To increase the potential diversity of the population, a crossover operation comes into play after generating the donor vector through mutation. The DE family of algorithms can use two kinds of crossover schemes, such as exponential and binomial. The principal role of crossover is as a construction. There is no such mutation that can achieve higher levels of construction than crossover (Price and Storn, 1997b). The constant of crossover reflects the probability with which the trial individual inherits the actual individual's genes. Crossover furnishes the high diversity of a population. Moreover, small values of crossover constant (Cr) increase the diversity of population. We obtained the following values for the parameters (Table 2 and Figure 4).

Results show that higher values for Cr improve the solution of scheduling problem efficiently. Upper limit for Cr (0.9) provided fastest and smoothest convergence.

Scale factor

The constant F is a control parameter, which manages the tradeoff between exploitation and exploration of the space. The constant of differentiation F is a scaling factor of the difference vector. Exploration efficiency can be controlled by the differentiation constant F as well. F has considerable influence on exploration: small values of F lead to premature convergence and high values slow down the search. Usually, F is fixed during the search process. However, there are some attempts to relax this parameter. The comparison of the performance of the best scale factor value is presented in Figure 5 and the values for the parameters given in Table 3. The graphical representation clearly indicates that the better value of F parameter is higher value (0.9). But researchers naturally consider some techniques, such as self-adaptation to avoid manual tuning of the scale factor parameter. Usually self-adaptation is applied to tune the control parameter F.

Size of population

DE optimizes a problem by maintaining a population of candidate solutions. The size of population NP is a very important factor. It should not be too small in order to avoid stagnation and to provide sufficient exploration. The increase of NP induces the increase of a number of function evaluations; that is, it retards convergence. If the population converges prematurely, then NP should be

Table 4. Values of parameters for comparing the performance of NP parameter.

Parameter	Value
Strategy	DE/rand/1/exp
Probability of crossover	0.5
Scaling factor	0.9
Initial population	Random

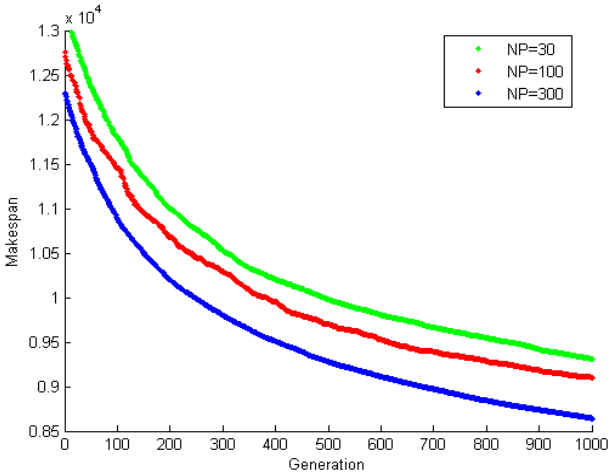

Figure 6. Performance of the population size values.

Table 5. Values of parameters for comparing initial population methods.

Parameter	Value
Strategy	DE/rand/1/exp
Population size	30
Probability of crossover	0.9
Scaling factor	0.1

increased. Furthermore, the correlation between NP and F may be observed. The parameters are given in Table 4 and their resulting comparison is as shown in Figure 6. In Figure 6 you can see that the large population size values achieve the best makespan reduction.

Initial population

The DE starts with an initial population of N real-valued vectors. The vectors are initialized with real values either randomly or they are evenly spread over the problem domain. The latter initialization usually leads to better results of the optimization process.

In the second set of the experiments, the initial population

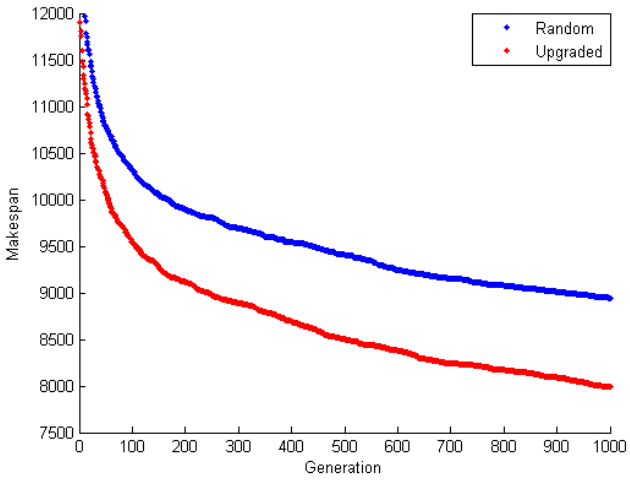

Figure 7. Comparison of the performance of initial population methods.

of the DE was upgraded with vector obtained by Min-min scheduling heuristic (Freund and Gherrity, 1998). Much better results were obtained when we upgraded the initial population with candidate solution obtained by the heuristic algorithm. In such case, the algorithm managed to exploit the different sub-optimal solutions provided at the beginning and converged to better schedules. We obtained the following values for the parameters (Table 5 and Figure 7).

CONCLUSION AND FUTURE WORK

Task scheduling is a critical design issue of distributed computing. A computational grid is a highly distributed environment. Scheduling in grid computing systems is an NP-complete problem. Therefore, using heuristic algorithms is a suitable approach in order to cope with its difficulty in practice. We presented an extensive study on the usefulness of DE algorithm for designing efficient grid schedulers when makespan parameter is minimized. Differential evolution (DE) has a number of parameters that determine its behavior and efficacy in optimizing a given problem. This paper gives a list of good choices of parameters for various optimization scenarios which should help the practitioner achieve better results with little effort. The goal of the scheduler is minimizing makespan. There are three main control parameters of the DE algorithm: the mutation scale factor F, the crossover constant Cr, the population size NP. Finally, we focus on the effect of each of these parameters on the performance of the DE as well as the state-of-the-art methods for tuning these parameters. Selecting an optimal set of control parameter values is a problem specific task for DE. The trial and error method used for tuning the control parameters is time consuming, less reliable and requires multiple runs on the given problem.

Our future work is proposed to develop adaptive DE based schedulers to grid scheduling problem.

REFERENCES

Adeyemo J, Otieno F (2009). Optimizing Planting Areas Using Differential Evolution (DE) and Linear Programming (LP). Int. J. Phy. Sci. (IJPS), 4(4): 212-220.

Ali S, Siegel H, Maheswaran M, Hensgen D (2000). Modeling Task Execution Time Behavior in Heterogeneous Computing Systems. Tamkang J. Sci. Eng., 3: 195-207.

Baca D (1989). Allocating Modules to Processors in a Distributed System. IEEE Trans., Software Eng., pp. 1427-1436.

Bouyer A, Abdullah A, Mokhtari M (2011). Localized Job Scheduling System Using Cooperative and System-centric Scheduling Policy for Market-oriented Grids. Scientific Res. Essays J. (SRE), 6(17): 3729-3750.

Braun T, Siegel H, Beck N, Boloni L, Maheswaran M, Reuther A, Robertson J, Theys M, Yao B (1998). A Taxonomy for Describing Matching and Scheduling Heuristics for Mixed-machine Heterogeneous Computing Systems. Proceedings of the 17th IEEE Symposium on Reliable Distrib. Systems, pp. 330-335.

Braun T, Siegel H, Beck N, Boloni L, Maheswaran M, Reuther A, Robertson J, Theys M, Yao B, Hensgen D, Freund R (2001). A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems. J. Parallel Dist. Comput., 61(6): 810-837.

Fidanova S, Durchova M (2006). Ant Algorithm for Grid Scheduling Problem. Large Scale Compu., LNCS, Springer, 3743: 405-412.

Foster I, Kesselman C (1998). The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann Publishers.

Freund R, Gherrity M (1998). Scheduling Resources in Multi-user Heterogeneous Computing Environment with Smart Net. Proceedings of the 7th IEEE HCW.

Izakian H, Abraham A, Snasel V (2009). Comparison of Heuristics for Scheduling Independent Tasks on Heterogeneous Distributed Environments. Proceedings of the International Joint Conference on Computational Sciences and Optimization, IEEE., 1: 8-12.

Lopez D, Raja S (2011). Dynamic Task Scheduling Using Service Time Error and Virtual Finish Time. J. Eng. Comput. Innovations (JECI), 2(5): 90-97.

Munir E, Jian-Zhong J, Sheng-Fei S, Rasool Q (2007). Performance Analysis of Task Scheduling Heuristics in Grid. Proceedings of the Int. Conf. Machine Learning and Cybernetics (ICMLC), 6: 3093-3098.

Minhas A, Hadi F, Shah S (2011). A Novel Cost-based Framework for Communication in Computational Grid Using Anycast Routing. Int. J. Phy. Sci. (IJPS), 6(10): 2348-2355.

Price K, Storn R (1997a). Differential Evolution: A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces", J. Global Optimization, 11(4): 341-359.

Price K, Storn R (1997b). Differential Evolution: Numerical Optimization Made Easy. Dr. Dobb's J., pp. 18–24.

Price K (1999). An Introduction to Differential Evolution. In: Corne, D., Dorigo, M., Glover, V. (eds.) New Ideas in Optimization, Mc Graw-Hill, pp. 79–108.

Price K, Storn R, Lampinen L (2005). Differential Evolution A Practical Approach to Global Optimization. Natural Computing Series, Springer-Verlag.

Tracy M, Braun T, Siegel H (1998). High-performance Mixed-machine Heterogeneous Computing. 6th Euro-micro Workshop on Parallel and Distrib. Processing, pp. 3-9.

Xhafa F, Abraham A (2008). Meta-heuristics for Grid Scheduling Problems. In Meta-heuristics for Scheduling in Distribution Computation. Environment, Springer, 146: 1-37.

Zaharie D (2007). A Comparative Analysis of Crossover Variants in Differential Evolution. Proceedings of the International Multi-conference on Computer Science and Information Technology, pp. 171–181.