TR 04-006

A GA-based Approach for scheduling Decomposable Data Grid Applications

Seonho Kim and Jon Weissman

February 18, 2004

Technical Report

A GA-based Approach for Scheduling Decomposable Data Grid Applications

Seonho Kim and Jon B. Weissman

February 14, 2004

Department of Computer Science & Engineering
University of Minnesota
4-192 EE/CS Building
200 Union Street SE
Minneapolis, MN 55455-0159 USA

A GA-based Approach for Scheduling Decomposable Data Grid Applications

Seonho Kim and Jon B. Weissman[†]
Dept. of Computer Science and Engineering,
University of Minnesota, Twin Cities
{shkim,jon}@cs.umn.edu

Abstract

Data Grid technology promises geographically distributed scientists to access and share physically distributed resources such as compute resource, networks, storage, and most importantly data collections for large-scale data intensive problems. The massive size and distributed nature of these datasets poses challenges to data intensive applications. Scheduling Data Grid applications must consider communication and computation simultaneously to achieve high performance. In many Data Grid applications, Data can be decomposed into multiple independent sub datasets and distributed for parallel execution and analysis. In this paper, we exploit this property and propose a novel Genetic Algorithm based approach that automatically decomposes data onto communication and computation resources. The proposed GA-based scheduler takes advantage of the parallelism of decomposable Data Grid applications to achieve the desired performance level. We evaluate the proposed approach comparing with other algorithms. Simulation results show that the proposed GA-based approach can be a competitive choice for scheduling large Data Grid applications in terms of both scheduling overhead and the quality of solutions as compared to other algorithms.

1. Introduction

Grid computing has become a promising technology for providing seamless access to heterogeneous resources and achieving a high performance benefit in wide-area environments [1]. In its early stages, Grid computing research focused mainly on the coordination of geographically distributed compute resources for high performance. However, in many areas of science and engineering such as high-energy physics and aerospace, requirements have emerged for collaborating and sharing huge amounts of widely distributed data as well as sharing high-performance compute resources. For example, in LHC experiments at CERN, huge amounts of data (tera byte or peta byte in scale) collected by observing collisions of particles, are analyzed through different levels of data processing operations [14][15]. Data Grids have been proposed to address this data requirement [20][21][22]. Data Grid technology enables geographically distributed scientists to access and share physically distributed heterogeneous resources such as compute resources, network, storage, and widely distributed datasets for large-scale data intensive

[†] The authors would like to acknowledge the support of the National Science Foundation under grants NGS-0305641 and ITR-0325949, the Department of Energy's Office of Science under grant DE-FG02-03ER25554, the Minnesota Supercomputing Institute and the Digital Technology Center at the University of Minnesota

problems. Research activities in the Data Grid community have mainly focused on defining core functionalities and basic infrastructure necessary for handling widely distributed large amounts of data [19].

In a Data Grid, as in a Computational Grid, resources are shared among applications. To achieve high performance, applications and resources should be scheduled efficiently. While a Computational Grid achieves high performance mainly by scheduling applications to powerful computing resources, communication and computation should be considered jointly when scheduling Data Grid jobs because of the potential communication bottleneck originating from the massive size and the widely distributed nature of datasets. There have been studies on scheduling computation and communication jointly for Data Grid applications. However, most of them do not reflect a characteristic typical in many data intensive applications, that data can be decomposed into multiple independent sub datasets and distributed for parallel execution and analysis. High Energy Physics (HEP) experiments fall into this category [14]. HEP data are characterized by independent events, and therefore this characteristic can be exploited when parallelizing the analysis of data across multiple sites. We exploit the parallelism to achieve desired performance levels when scheduling large Data Grid applications.

When parallel applications require multiple data from multiple data sources, the scheduling problem is challenging along several dimensions – how should data be decomposed, should data be moved to computation or vice-versa, and which computing resources should be used. We can solve the problem optimally by adding some constraints (e.g., decomposing data into sub datasets of the same size). Another approach is to use heuristics such as those based on optimization techniques e.g. genetic algorithm, simulated annealing, and tabu search.

This paper proposes a novel Genetic Algorithm (GA) based approach to address scheduling of decomposable Data Grid applications, where communication and computation are considered at the same time. The proposed algorithm is novel in two ways. First, it automatically balances load, that is, data in our case, onto communication/computation resources and generates a near optimal schedule. Second, it does not require a job to be pre-decomposed - most GA based approaches for scheduling problems have addressed the problem of scheduling ***m*** pre-decomposed tasks onto ***n*** computing nodes. We examined the relative quality of the solution generated with the GA-based approach as compared to other algorithms (constraint based approaches). We developed a suite of algorithms for comparison: two Divisible Load Theory (DLT) based algorithms (baseline algorithm and Iterative DLT (IDLT)) and Constrained DLT (CDLT) [9], and Tasks on Data Present (TDP) [2][5].

We found that the GA based approach with a good initial population, in general, outperform other algorithms. However, some approaches (IDLT and TDP) perform competitively under certain Grid configurations (short jobs: 10 ~ 1000 seconds) compared to the GA based algorithms because their scheduling time (50 ~ 100 milli-seconds) is much less compared to that of GA based approaches (10 ~ 180 seconds). Since the running time of the GA-based approaches is negligible for large jobs (0.002% ~ 0.5% of total execution time), the proposed GA-based approach can achieve the desired performance level for large jobs (requiring at least a couple of hours of execution time) - common in Data Grid applications such as CMS experiments [14].

This paper is organized as follows. Some related works are reviewed in Section 2. In Section 3, problem space and models are described. Section 4 presents the proposed GA-based approach. Experimental results are discussed in Section 5. Finally, we conclude and discuss conclusion and future work in Section 6.

2. Related Work

In data intensive environments, the location of input data objects should be taken into consideration when selecting resources for a job. Intuitively, the scheduler should map a job to the site where the total latency of data transfer can be minimized. Job scheduling in data intensive environments, has recently been studied [2][3][4][5]. [2][5] examines several scheduling heuristics and several data replication strategies. In ChicSim [2], data movement and job scheduling are performed separately as decoupled processes. They found that benefits such as good system utilization and low response time can be achieved when jobs are always scheduled where data is located. In Bricks [5], they examined combinations of scheduling and file replication algorithms. They investigated several scheduling mechanisms considering computing resources and location of datasets as well as the impact of data replication. In [4], Park et al. propose a new scheduling model of executing a job on one site while considering both computation and communication in a Data Grid environment. They found that replication of data has considerable impact on enhancing the performance of the scheduler. Orlando et al [3] examined the on-line MCT (Minimum Completion Time) heuristic strategy for scheduling high performance data mining tasks on top of the Knowledge Grid. The most important difference between these works and our work is that none of these works considered parallel applications. Parallelism significantly complicates the scheduling problem because problem partitioning now becomes an integral part of the scheduling process.

Divisible Load Theory (DLT) has recently emerged as a powerful tool for modeling data-intensive computational problems and allowing tractable performance analysis of systems incorporating communication and computations issues [6][7][8][9]. DLT exploits the parallelism of a divisible application, a model of computation which is continuously divisible into parts of arbitrary size (chunks), by distributing loads in a single source onto multiple computing resources. However, in the real world, DLT is not applicable directly to the problem of finding a globally optimal distribution of load for applications requiring multiple datasets from different distributed data sources because the system of equations is under-constrained. In [8], Ko et al. suggested a two-step approach to address this problem: (1) finding the optimal number of computing nodes (a set) to distribute a particular load, and (2) balancing load over sets. Wong [9] addresses this problem by adding the additional constraint that each worker node receives the same load fraction from each data source. However, any solution provided by these approaches is not globally optimal.

The Genetic Algorithm (GA) has widely been used as a practical and robust optimization and search method in various areas [10][11][13]. GA has recently been applied to many scheduling problems [10][11][18] which are, in general, NP-complete [17]. Most GA based approaches for scheduling problems have addressed the problem of scheduling ***m*** pre-decomposed tasks onto ***n*** computing nodes. However our approach is different in that we design and apply a novel GA-based approach for the decomposition of a job into multiple sub-tasks while simultaneously considering communication and computation.

3. Problem Description and Models

We consider a virtual organization (VO) Grid model, as shown in figure 1, that is applicable to numerous distributed science applications such as high energy and nuclear physics [20][21][22], climate analysis [23], and bio-informatics [24], to name a few. A VO consists of ***n*** virtual sites. We assume virtual sites are connected to the Grid with a relatively low bandwidth network relative to the network within a virtual site. A virtual site may consist of multiple physical sites if they are interconnected by a high bandwidth network. Each site consists of computing resources, storage systems, large data collections. A virtual site, e.g., in the MONARC [16] context, can be thought of as a "regional center", a composite object containing a number of data servers and processing nodes where all are connected to a LAN. In this paper, we consider scheduling of a single parallel Data Grid application onto a network such as in figure 1. Scheduling multiple Data Grid applications competing for shared resources is a subject of future work.

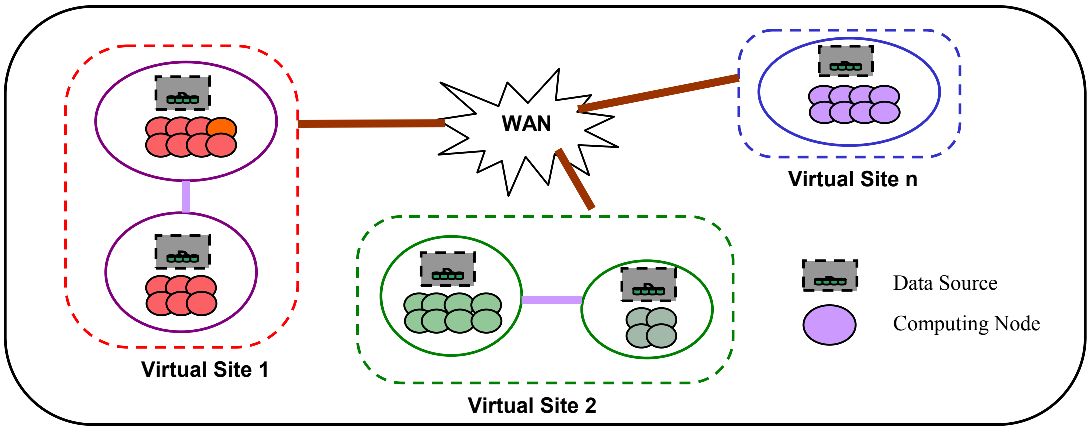


Figure1. Virtual Organization Grid model

3.1 Job and Data Model

We target data intensive applications that can be decomposed into multiple independent subtasks and executed in parallel across multiple sites without any interaction among sub tasks. We consider job decomposition by decomposing input data objects into multiple smaller data objects of arbitrary size and processing them on multiple virtual sites. For example, in theory, HEP jobs are arbitrarily divisible at event granularity and intermediate data product processing granularity [14]. The physical datasets may represent raw data (collected with detectors) or an intermediate data product stored at different locations. We assume that a job requires a very large logical input data set (***D***) that consists of ***m*** physical datasets and each physical dataset (of size L_k) resides at a data source (DS_k, *k=1..m)* of a particular site. The scheduling problem is to decompose ***D*** into datasets (D_i, *i=1..n*) across ***n*** virtual sites in a VO given its initial physical decomposition. We assume that the decomposed data can be analyzed on any site by applying the same data parallel operation ***p***. Hence a decomposable job ***J*** can be modeled as follows:

p(D_i) *: an operator* ***p*** *is applied to* D_i *at site* ***i***

p_{aggr}*: an operator aggregating output data sets*

$$J = p(D) = p_{aggr}(p(D_1), p(D_2), \dots p(D_n))$$

$$size(D) = \sum_{k=1}^{m} L_k \quad D = \sum_{i=1}^{n} D_i$$

We assume that the cost of ***p*** in processing $\boldsymbol{D_i}$ is linear in the size of data object $\boldsymbol{D_i}$ and that the size of output data generated by ***p*** is linear with respect to the size of input data $\boldsymbol{D_i}$. However, both of these assumptions can be easily relaxed. Figure 2 shows how the logical input data (***D***) is decomposed onto networks and computing resources. A dataset of size $\boldsymbol{L_k}$ in a data source $\boldsymbol{DS_k}$ is decomposed into $\{ l_{ki} : L_k = \sum l_{ki}$, where ***i=1..n*** $\}$ and transferred to each worker site. After processing the datasets transferred to site ***i*** ($\boldsymbol{D_i}$, where $\boldsymbol{size(D_i) = d_i}$, $d_i = \sum l_{ki}$, ***k=1.. m***), the output is transferred to the destination site and aggregated. The size of output data is a function of the size of the data ($\boldsymbol{d_i}$).

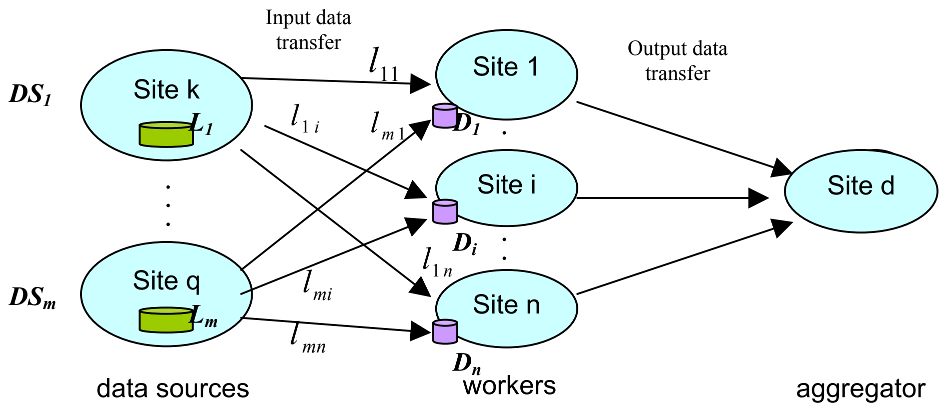


Figure 2. Data Decomposition and processing

In this paper, we do not consider the replication of data and the overlapping of communication and computation (we will investigate these in future work). The execution time of an application is the maximum among all the execution times of the sub tasks.

3.2 Cost Model

The execution time of a subtask allocated to the site ***i*** ($\boldsymbol{T_i}$) and the turn around time of a job ***J*** ($\boldsymbol{T_{turn\_around\_time}}$) can be expressed as follows:

$$T_i = T_{input\_cm}(i) + T_{cp}(i) + T_{output\_cm}(i,d) \text{ and } T_{turnaround\_time} = \max_{i=1,\dots,n} T_i$$

The cost ($\boldsymbol{T_i}$) includes input data transfer ($\boldsymbol{T_{input\_cm}(i)}$), computation ($\boldsymbol{T_{cp}(i)}$), and output data transfer to the client at the destination site ***d*** ($\boldsymbol{T_{output\_cm}(i,d)}$).

$$T_{input\_cm}(i) = \max_{k=1..m} \{l_{ki} \cdot \frac{1}{z_{ki}}\},\ T_{cp}(i) = d_i \cdot \omega_i \text{ and } T_{output\_cm}(i,d) = f(d_i) \cdot z_{id}$$

where z_{ij} : the network bandwidth between site ***i*** and ***j*** and

ω_i : the computing time to process a unit dataset of size 1MB at site ***i***

$f(d_i)$: output data size : a function of the size of input data

We assume that data from multiple data sources can be transferred to a site ***i*** concurrently in the wide area environment and computation starts only after the assigned data set is totally transferred to the site. Hence, the problem of scheduling a divisible job onto ***n*** sites can be stated as deciding the portion of original workload (***D***) to be allocated to each site, that is, finding a distribution of $\{l_{ki}\}$ which minimizes the turn-around time of a job. The proposed GA-based approach uses this cost model when evaluating solutions at each generation.

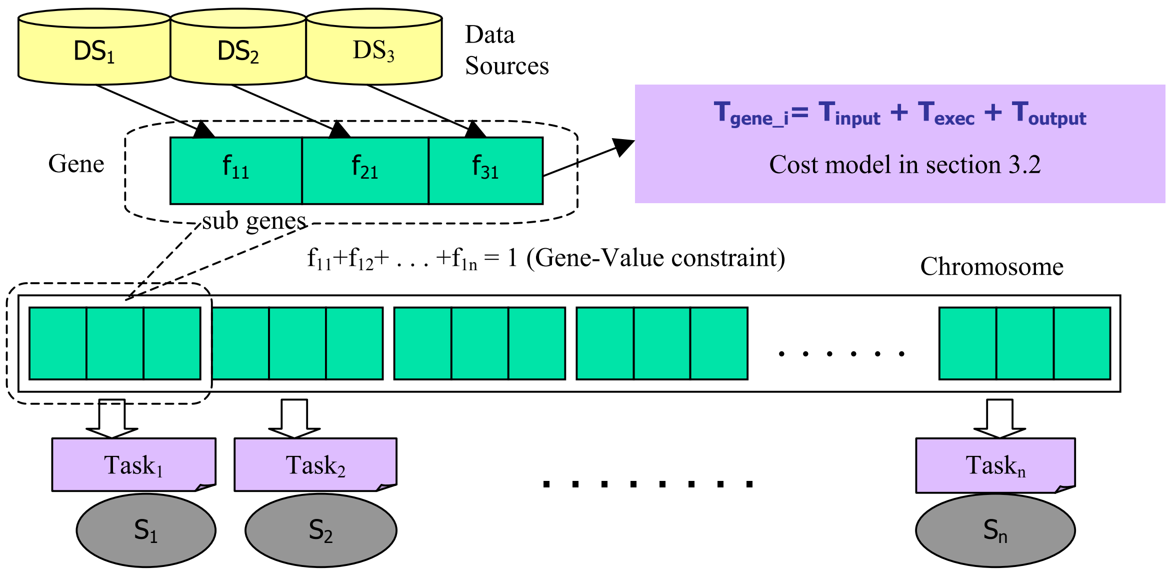
4. GA-based Job Decomposition and Scheduling

The GA based search methods are rooted from the mechanisms of evolution and natural genetics [13]. They search large solution spaces to find near-optimal solutions. A general genetic algorithm consists of several steps. The first step of the GA is to generate an initial population. The second step is to evaluate chromosomes with a fitness function and rank them by fitness value reflecting how good a chromosome is. After the evaluation of the initial population, evolution starts. The first step of evolution is the selection process. In this step, competitive chromosomes (the fittest ones) survive and are duplicated with high probability. After a new generation is produced by the selection process crossover operation follows. In this step, pairs of chromosomes are randomly selected and some of their corresponding genes are exchanged to generate two new chromosomes. In the next step, some of chromosomes are transformed into other chromosomes by the mutation process. Finally, chromosomes are evaluated and ranked for next round of evolution. These four steps (selection/ crossover/ mutation/ evaluation) are applied iteratively until stopping condition is met.

4.1 Problem Representation for GA

4.1.1 Chromosome Representation. To apply the GA approach to an optimization problem, a solution needs to be represented as a chromosome encoded as a set of strings. We designed a representation for our problem as follows. Given ***n*** sites, a job is decomposed into ***n*** sub tasks and each task is allocated to one of ***n*** sites. The job may require multiple input files distributed among ***m*** data sources. A chromosome consists of ***n*** genes and each gene is composed of ***m*** sub-genes as shown in Fig. 3. Each gene in a chromosome matches a task allocated to a site. That is, a gene g_i corresponds to a task t_i assigned to site S_i for $1 \le i \le n$. Each sub-gene of a gene is associated with a real value, f_{ki}, in the range 0 to 1, where $1 \le k \le m$ and $1 \le i \le n$. This value represents a portion of workload assigned to task t_i from data source DS_k, the S_k containing the required input data. Since f_{ki} is a portion of workload L_i in DS_i, $\sum_{i=1}^{n} f_{ki} = 1$, for each ***k*** from ***1*** to ***m*** [Gene-Value Constraint]. In order to associate a sub gene with a float value, a binary string representation of a float is used [12]. We discuss this in more detail when we discuss the mutation operation**.**

4.1.2 Fitness Function and Evaluation. Each chromosome is associated with a fitness value evaluated by a fitness function (or objective function) and all chromosomes in the population are ranked by these values. The associated fitness value is the time when the last subtask finishes its execution. The cost model discussed in section 3.2 is used as a fitness function.



DS1
DS2
DS3
Data Sources
Gene
f11
f21
f31
sub genes
Tgene_i= Tinput + Texec + Toutput
Cost model in section 3.2
f11+f12+ . . . +f1n = 1 (Gene-Value constraint)
Chromosome
Task1
Task2
Taskn
S1
S2
Sn

Figure 3. Chromosome Representation

4.1.3 Population Initialization. In this step, a collection of chromosomes is generated as an initial population. We developed two methods: Random Initialization (GA_Random) and Initialization based on application hint (GA_Hint). GA_Random spawns a pool of chromosomes randomly and selects a predetermined number of chromosomes after evaluating chromosomes according to their fitness values.

The basic idea behind the method GA_Hint is using application specific information (e.g. the ratio of computation to communication) to generate a seed chromosome of good quality (high fitness value). Intuitively, computation intensive jobs would be better scheduled on sites with powerful computing resources, while communication intensive jobs on sites with required input data sets to reduce data transfer time. We calculate $cap_i = comp_i \cdot ccRatio + data_i / ccRatio$ for each site**,** where *comp*$_i$ is the normalized computing power of site *i*, *data*$_i$ represents the normalized portion of required input data sets residing in site *i* and *ccRatio* is the non-zero ratio of computation and communication. The value of each gene is, then, set with $cap_i \Big/ \sum_{i=1}^{n} cap_i$. Now each sub gene of a gene has the same value. Since data transfers from other data source sites to a data source site are unnecessary, we adjust values of sub genes representing the transfer of input data sets from other data source sites. We use the sigmoid function to smoothly and continuously threshold those values: $value_{adjusted} = \dfrac{value_{current}}{1 + e^{-a \cdot \log(ccRatio)}}$. Finally, a predefined number of slightly different twin chromosomes are generated mutating the seed chromosome. After extensive experiments, we found that a=2 gives the best result. In order to prevent premature convergence, identical chromosomes are not allowed as members of the initial population.

4.1.4 Selection Methods. After each generation, a new generation is generated from the previous population. The chromosomes in the population are sorted by fitness values. Better solutions have more chance to survive through the selection step. Two selection schemes are implemented and tested: rank-based roulette wheel selection scheme [13] and proportionate selection scheme. The rank-based roulette wheel selection scheme allocates a sector on a roulette wheel to each chromosome. The ratio of angles of two adjacent sectors is a constant. The basic idea of rank-based roulette wheel selection is to allocate larger sector angle to better chromosomes so that better solutions will be included in the next generation with higher probability. A new chromosome for the next generation is cloned after a chromosome as an offspring if a randomly generated number falls in the sector corresponding to the chromosome. Alternatively, the proportionate selection scheme generates chromosomes for the next generation only from a predefined percentage of the previous population.

4.1.5 Crossover Operation. In this step, pairs of chromosomes are picked at random from the current population and a uniform crossover operator is applied only if a randomly generated number is less than a predefined crossover rate ($\boldsymbol{r_x}$) (chromosome-level crossover). As shown in figure 4, for each gene in a chromosome, a random number is generated and two genes are exchanged only if the number is less than a predefined gene crossover rate ($\boldsymbol{r_{gx}}$) (gene-level crossover). After crossover, each chromosome is normalized to meet the Gene-Value Constraint. From extensive experiments, we found the proposed GA-based approaches works with $r_x=r_{gx}=0.6$.

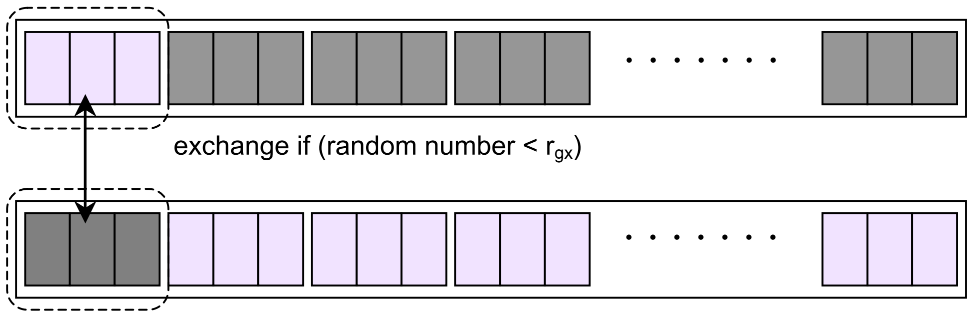
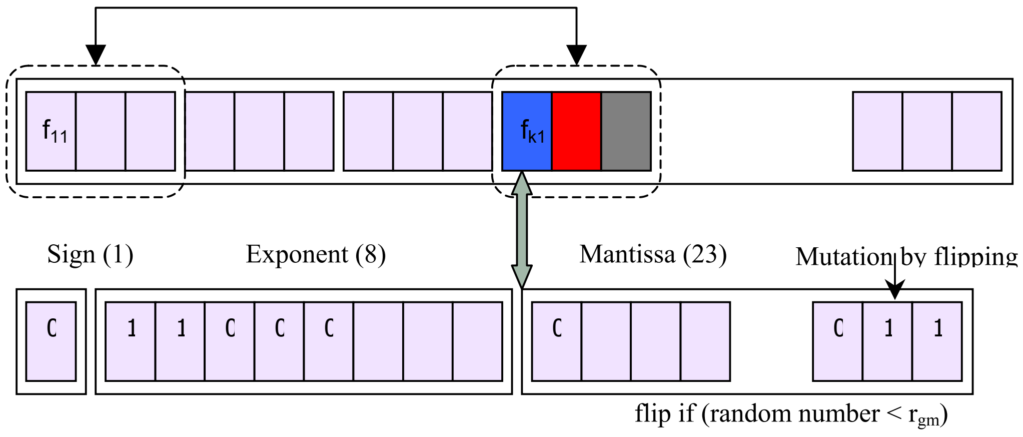


Figure 4. Uniform Crossover

4.1.6 Mutation Operation. After the crossover operation, chromosomes are subjected to a mutation operation. Each sub gene contains a binary array string representation of a 32 bit floating point number as shown in Fig. 5 and randomly selected bits are flipped from 0 to 1 or vice versa. Since real numbers associated with each sub gene are in the range 0 to 1, the mutation operator flips only bits from the mantissa part. We tested two different mutation schemes: uniform mutation and two points mutation. For both schemes, two parameters are given: mutation rate ($\boldsymbol{r_m}$) and gene mutation rate ($\boldsymbol{r_{gm}}$). For each chromosome in current population, a random number is generated and the mutation operator is applied only if the random number is less than $\boldsymbol{r_m}$. If a chromosome is subjected to the mutation operation (chromosome-level mutation), then the two points mutation scheme selects two genes randomly and sub genes of one of the two genes are mutated (gene-level mutation). Each bit from the mantissa is flipped only if a random number is less than $\boldsymbol{r_m}$. After mutating one of two genes, the sub-gene values of the other gene are revised as shown in figure 5. On the other hand, in the uniform mutation scheme, all genes in a chromosome that is subjected to chromosome-level mutation are subjected to gene-level mutation in a manner similar to the two-points mutation scheme.

4.2.7 Stopping Condition. We used three criteria as stopping conditions. (1) The evolution stops if the total number of iterations reaches a predefined number of iterations, (2) if the fittest chromosome of each generation has not changed much, that is, the difference is less than 10^{-3} over a predefined number, or (3) if all chromosomes have the same fitness values, i.e., when the algorithm has converged.



f'_{k1} : mutated value

$$f'_{11} = (f_{11} + f_{k1}) - f'_{k1}$$

Figure 5. Two Points Mutation

5. Experiment Results

5.1 Algorithms

To investigate the effectiveness of the proposed GA approach as compared to other constraint based approaches, we developed two DLT based algorithms (a baseline algorithm and IDLT) and for comparison, we examined another DLT based algorithm (CDLT) discussed in [9] and a simple mapping strategy named TDP (Tasks on Data Present). We compared them with two different GA approaches: GA_Random and GA_Hint discussed in section 4.1.3. Our intuition is that different approaches may be better in different situations.

5.1.1 Baseline Algorithm. The basic idea behind the baseline algorithm is to collect all required input datasets to one site and then apply DLT to get a distribution of load. In this algorithm, one data source is selected from a set of data sources by evaluating and ranking them based on two values: the estimated transfer time of all input data sets from other data sources to the data source site (Data Gathering Time: DGT_i) and the estimated execution time (EET_i). Since we assume concurrent data transfer, DGT_i is the maximum among all the data transfer times from other data sources to the data source i. Since DLT gives the optimal solution for the single data source case, we apply DLT to get EET_i. Finally, we choose a data source with $\min_i \{DGT_i + EET_i\}$ and collect all input data sets into this site and apply DLT to decide the optimal distribution of loads.

5.1.2 Iterative DLT (IDLT). For the case of a single data source, DLT finds the optimal distribution of the load by solving equations systematically. However, for the case of multiple sources, DLT cannot get the optimal distribution of load because the system of equations is under-constrained. We developed a near optimal algorithm based on the DLT algorithm: Iterative DLT. In this algorithm, DLT is applied to each data source to find the optimal distribution of workloads for the workload in the data source. Then it allows one-directional data transfer between two data source sites in order to eliminate unnecessary data transfer between data sources.

5.1.3 Constrained DLT (CDLT). For the case of $\boldsymbol{m}$ data sources and $\boldsymbol{n}$ sites, we have $\boldsymbol{n}$ equations with $\boldsymbol{mn}$ unknowns ($\boldsymbol{l_{ki}}$'s). In order to solve this under-constrained system, additional constraints need to be added to this system. One possible constraint we tested, suggested in [9], is that each worker node receives the same load fraction from each data source. With this constraint, the system of equations is solvable for a unique solution.

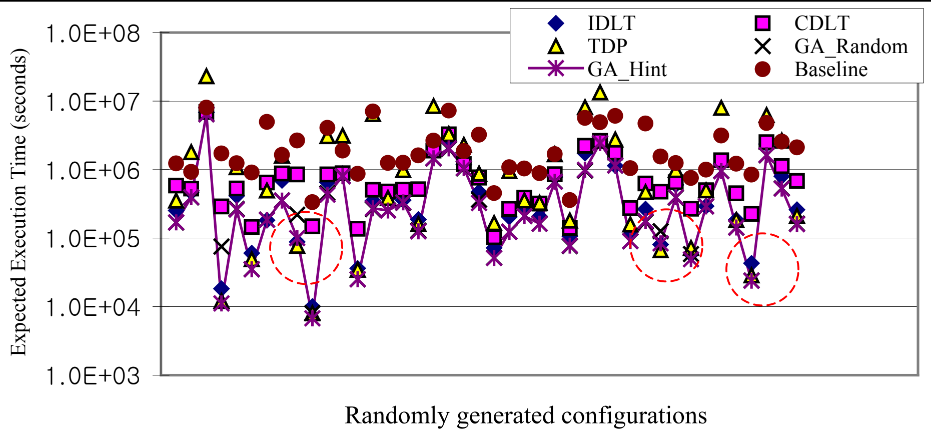
5.1.4 Task Data Present (TDP). Since applications in a Data Grid usually deal with huge data sets of tera- or peta-byte scale, running jobs on sites having required data sets might be a better choice than moving data to sites with powerful computing resources. In previous work [2][5], this heuristic was examined with other strategies for non-divisible applications. We modified this heuristic for our job model. This strategy maps tasks only to the sites where required data is present. Each task processes the data sets residing at that site. There is no input data transfer in this case.

5.2 Experimental Results

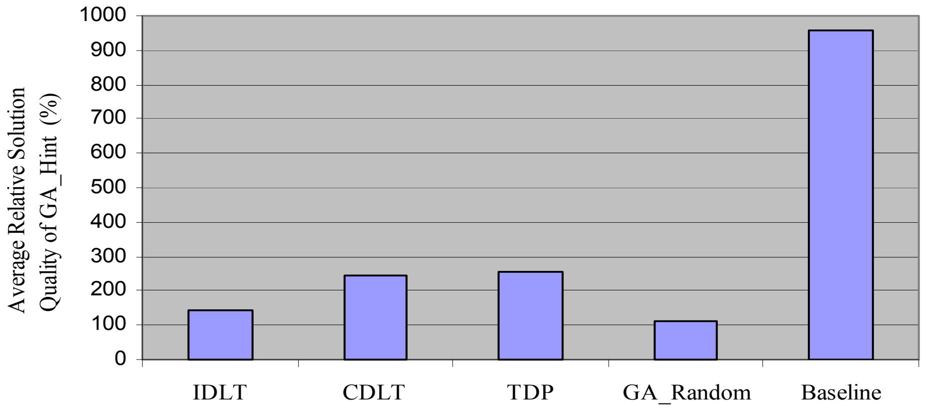
To measure the performance of the proposed GA-based approach against other approaches, randomly generated experimental configurations were used. The estimated expected execution time for processing a unit dataset on each site, the network bandwidth between sites, input data size, and the ratio of output data size to input data size were randomly generated with uniform probability over some predefined ranges. The network bandwidth between sites is uniformly distributed between 1Mbyte/sec and 10Mbyte/sec. The location of $\boldsymbol{m}$ data sources ($\boldsymbol{DS_k}$) is randomly selected and each physical dataset size ($\boldsymbol{L_k}$) is randomly selected with a uniform distribution in the range of 1GB to 1TB. We assume that the computing time spent in a site $\boldsymbol{i}$ to process a unit dataset of size 1MB is uniformly distributed in the range $1/r_{cb}$ to $10/r_{cb}$ seconds, where r_{cb} is the ratio of computation speed to communication speed.

We examined the overall performance of each algorithm by running them under 200 randomly generated Grid configurations. We set the GA related parameters ($r_x = r_{gx} = 0.6$, $r_m = 0.5$, and $r_{gm} = 0.6$) with values that we found after some preliminary experiments and we varied other parameters: ccRatio (0.001 ~ 1000), oiRatio: the ratio of output data size to input data size (0 ~ 1), r_{cb} (10 ~ 500), n (3 ~ 20), m (2 ~ 20). Overall, GA_Hint outperforms other algorithms including GA_Random as shown in figure 6 (a) and (b). GA_Hint finds the best solution in general (averagely 45% better solutions compared to IDLT algorithm, 140% for CDLT, 160% for TDP, 13% for

GA_Random, and 800% for Baseline algorithm). However, for some configurations (short jobs, ccRatio=0.001), TDP and IDLT occasionally offer slightly better solutions, as marked in a circle in figure 6 (a).



(a)



(b)

Figure 6. Experiments on 200 randomly generated configurations (a) the expected execution time (b) the average relative quality of solutions of GA_Hint compared to other algorithms

We further examined the impact of various parameters by conducting two different experiments: (1) varying GA-related parameters such as mutation rate, crossover rate, generation number, selection schemes, and population initialization schemes and (2) varying application specific parameters.

5.2.1 Impact of application related parameters. Figure 7 and 8 show the impact of application specific parameters on the performance of the algorithms. As shown in figure 7 (a), for communication intensive applications (small ccRatio), TDP, GA-based approaches, and IDLT generate better solutions than other algorithms in general. On the other hand, TDP does not perform well for compute intensive applications (large ccRatio) because applications that fall into this type should be scheduled on sites with powerful computing resources to reduce data transfer overhead. As ccRatio increases, GA based approaches, IDLT, CDLT, and the baseline algorithm show similar performance. However, GA_Hint offers more performance gain as ccRatio increases even though the relative solution quality decreases as shown in figure 7 (b). It is because the total execution time also increases. In terms of the ratio of performance gain, GA-based approaches give the best results around ccRatio=10 by at least 40 % as shown in figure 7 (c).

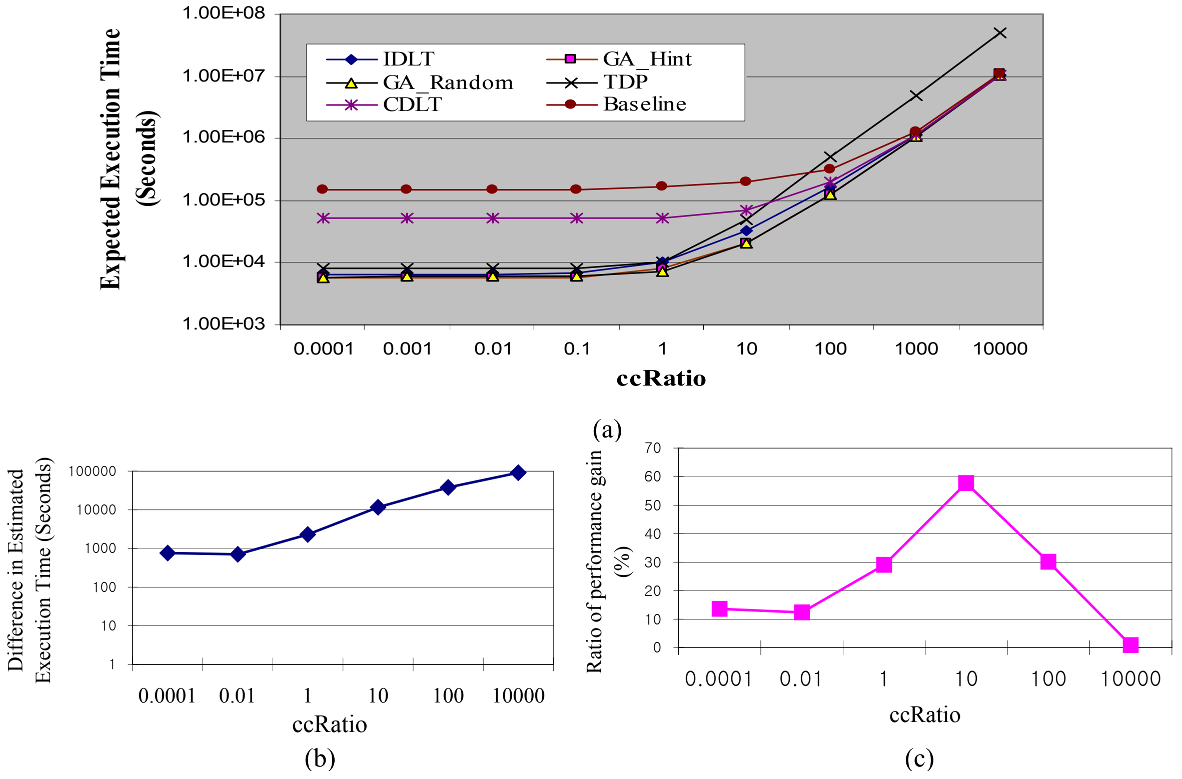
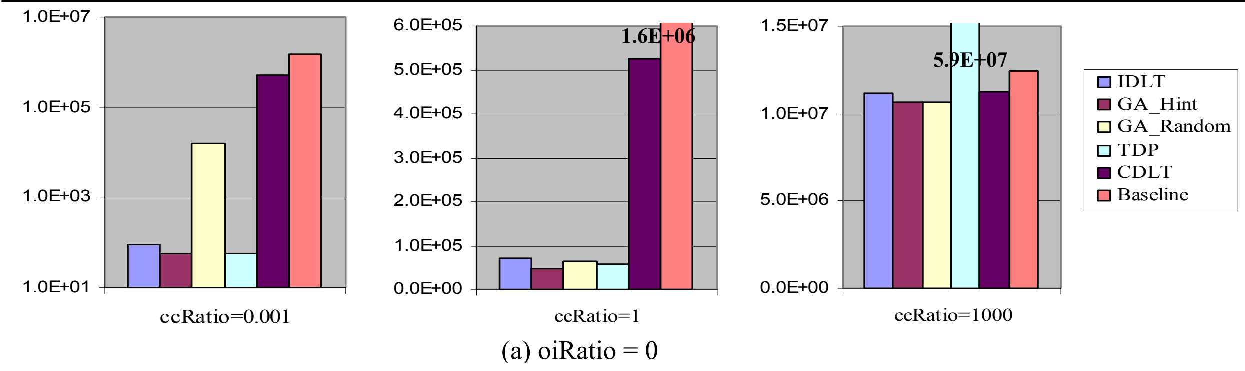


Figure 7. Impact of the ratio of computation to communication (a) The expected execution time as ccRatio increases (b) The difference in estimated execution time of solutions between GA_Hint and IDLT (c) The ratio of performance gain to total execution time

Figure 8 shows the impact of the ratio of output data size to input data size. GA_Hint and TDP perform well for communication intensive applications that generate small output data compared to input data size (low oiRatio) while GA_Random does not perform well because this algorithm starts with a population of poor quality as shown in figure 8 (a). For computation intensive applications, the ratio of output data size to input data size does not affect the performance of the algorithms much.

5.2.2 Impact of GA related parameters. Figure 9 shows the performance comparison of two GA related schemes: selection scheme and population initialization scheme. In terms of selection scheme, proportionate scheme outperforms rank-based roulette wheel selection scheme (RW) for non compute intensive jobs while RW works better for compute intensive jobs. From the observation that GA_Hint outperformed GA_Random, we found that heuristic information about system or applications can be used to enhance the performance of the GA-based approach when generating the initial population.



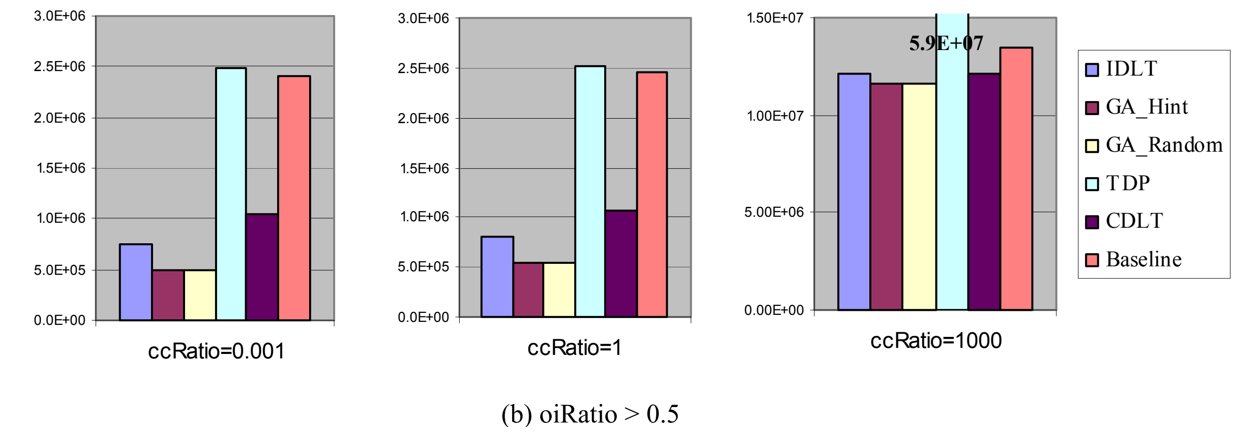


Figure 8. The impact of output data size to input data size

(a) oiRatio = 0 : No output or small size of output (b) oiRatio > 0.5

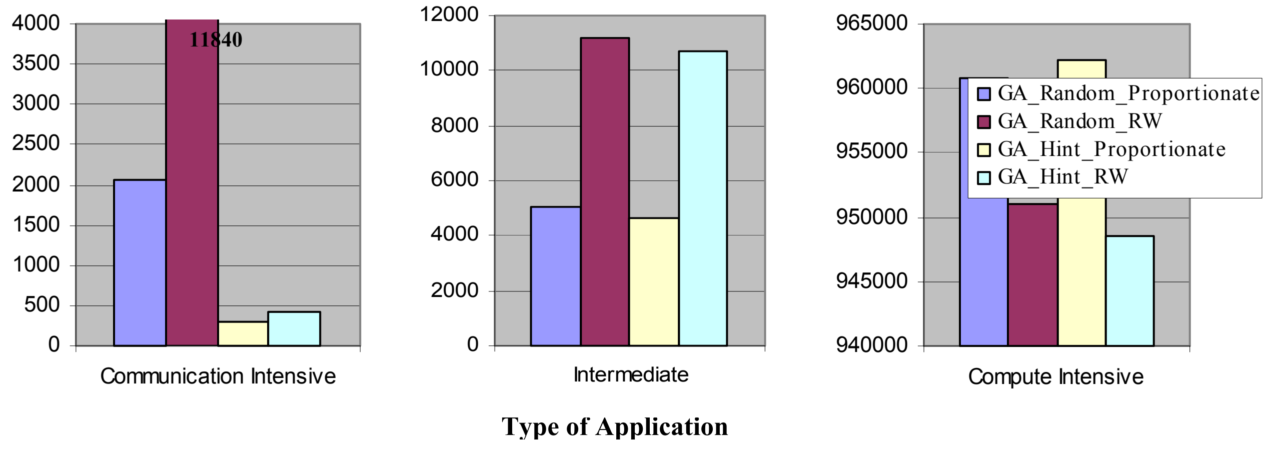
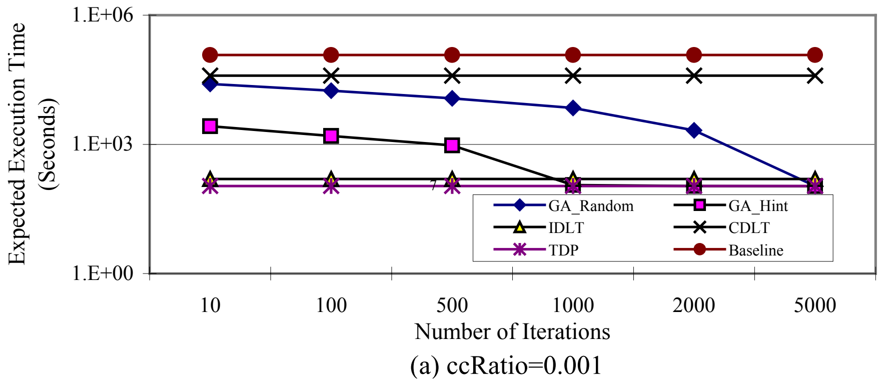
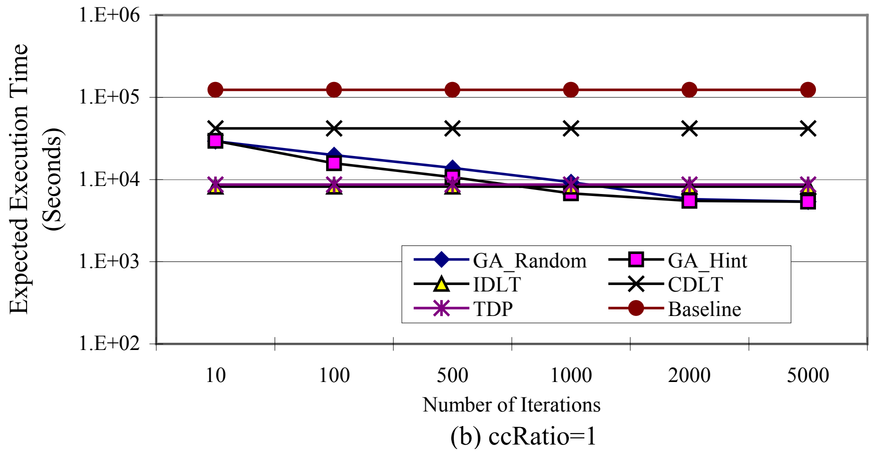


Figure 9. Comparison of GA related schemes: selection scheme and Initialization scheme

We examined the impact of the number of iterations on the quality of solutions generated. Figure 10 (a) shows GA_Hint starts converging after about 1500 iterations for communication intensive applications. However, GA_Random converges relatively slowly. Shown in figure 10 (b) and (c), both GA_Hint and GA_Random converge quickly as ccRatio increases: after about 700 iterations for intermediate type of applications and 300 for compute intensive type of applications.





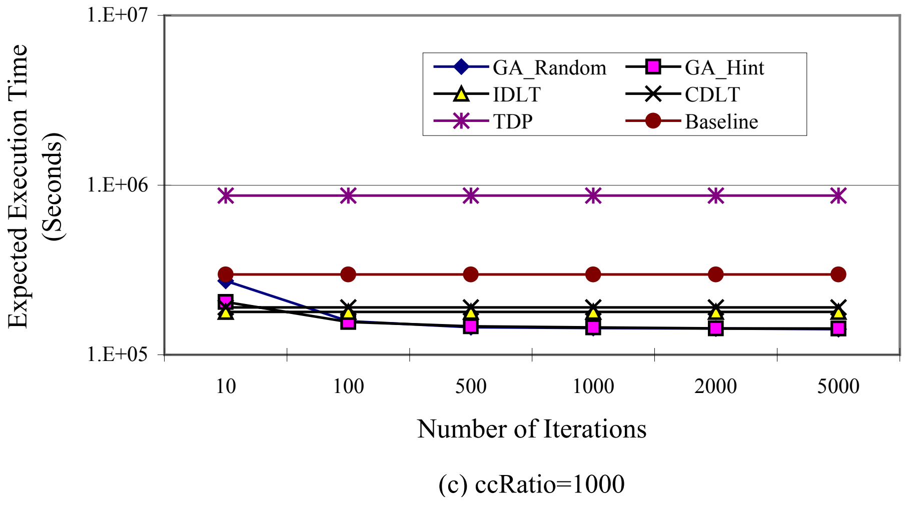


Figure 10. Impact of number of iteration (a) communication intensive application (ccRatio=0.001) (b) intermediate application (ccRatio=1) (c) compute intensive application (ccRatio=1000)

Finally, we measured the scheduling overhead of the GA-based approach with 1000 iterations to investigate the applicability of the approach. We tried different population sizes from 100 to 10000. The population size represents the total number of genes in population: (***n*** x ***m*** x # of chromosomes in population). Figure 11 shows that the GA based approach takes about 3 minutes for 1000 iterations with the configuration of population size 10000. Compared to other algorithms (scheduling time: 50 ~ 100 ms), the running time of the GA-based approach will be much longer. However the running time of the GA-based approach is negligible (0.002% ~ 0.5% of total execution time) for large jobs (10^4~10^8 seconds, as shown in Figure 6 (a)) - the most common case in Data Grid applications. On the other hand, the GA-based approach might not be a good choice for short jobs (10~1000 seconds). This observation suggests that the proposed GA-based approach can be a competitive choice for scheduling large jobs (requiring at least a couple of hours of execution time) common in Data Grid applications such as CMS experiments [14].

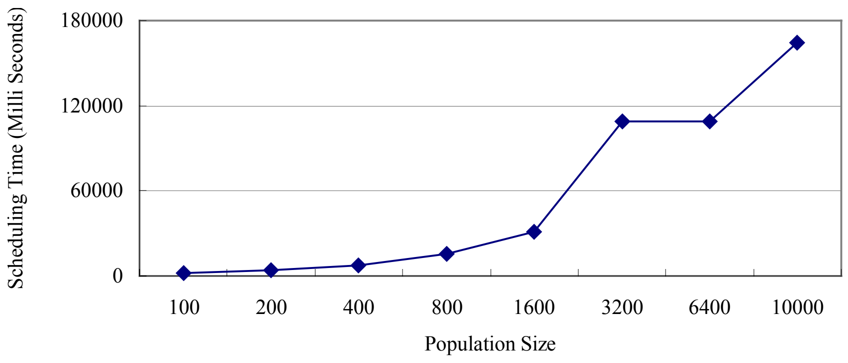


Figure 11. Running time of GA-based algorithm
(measured on Pentium 4 2.x GHz dual processors, RAM: 512 MB, OS: Linux)

In summary, we found the following: (1) In general, GA_Hint outperforms other algorithms over a wide set of parameters. (2) The proposed GA-based approach will be a competitive choice for scheduling large Data Grid applications in terms of both the scheduling overhead and the quality of solutions as compared to other algorithms. (3) Since a good initial population can improve the solution quality and convergence rate (reduced scheduling time), we may use one of the other less expensive algorithms to generate an initial population.

6. Conclusion and Future work

In this paper, a novel GA-based approach is proposed to address the problem of scheduling a divisible Data Grid application while considering communication and computation at the same time in wide area data intensive environment. We examined the overall performance of the proposed approach and investigated the impact of various parameters such as ccRatio, oiRatio, selection schemes, population initialization schemes, and the number of iterations. Results show that the proposed GA-based approach outperformed other algorithms in general. The proposed GA-based approach will be a competitive choice for scheduling large Data Grid applications in terms of both scheduling overhead and the quality of solutions as compared to other algorithms. The results from experiments on GA-related parameters suggest that the initialization of population with chromosomes of good quality is critical to GA-based approach in terms of the quality of solution and the convergence rate.

Even though we targeted Data Grid applications as an application model, this approach can be applied to other applications where computation and communication should be considered simultaneously. As a next step, we will extend this work for the case of multiple jobs competing for shared resources.

References

[1] I. Foster and C. Kesselman, The GRID: Blueprint for a New Computing Infrastructure, Morgan Kaufmann, 1999

[2] K. Ranganathan, I. Foster, "Decoupling Computation and Data Scheduling in Distributed Data-Intensive Applications", 11th IEEE International Symposium on High Performance Distributed Computing, 2002

[3] S. Orlando, et al, "Scheduling High Performance Data Mining Tasks on a Data Grid Environment", Proceedings of Int. Conf. Euro-Par 2002

[4] S.M. Park, and J.H. Kim, "Chameleon: A Resource Scheduler in A Data Grid Environment", In Proceedings of the 3rd IEEE International Symposium on Cluster Computing and the Grid (CC-GRID 2003), 2003

[5] A. Takefusa, O. Tatebe, S. Matsuoka, and Y. Morita, "Performance Analysis of Scheduling and Replication Algorithms on Grid Datafarm Architecture for High Energy Physics Applications," Proceedings on the 12 IEEE international symposium on HPDC, 2003

[6] T.G. Robertazzi, "Ten Reasons to Use Divisible Load Theory", IEEE Computer, 63-68, May 2003

[7] V. Bharadwaj, D. Ghose, V. Mani, and T.G. Robertazzi. Scheduling Divisible Loads in Parallel and Distributed Systems. IEEE Computer Society Press, 1996

[8] K. Ko and T.G. Robertazzi., Scheduling in an Environment of Multiple Job Submissions," Proceedings of the 2002 Conference on Information Sciences and Systems, Princeton University, Princeton NJ, March 2002

[9] H.M. Wong, D. Yu, V. Bharadwaj, T.G. Robertazzi., "Data Intensive Grid Scheduling: Multiple Sources with Capacity Constraints", 2003, Submitted for publication

[10] L. Wang, H.J. Siegel, V.P. Roychowdhury, and A.A. Maciejewski, "Task Matching and Scheduling in Heterogeneous Computing Environments Using a Genetic Algorithm-Based Approach", Journal of Parallel and Distributed Computing47, 8-22, 1997

[11] A. Abraham, R. Buyya, and B. Nath., "Nature's Heuristics for Scheduling Jobs on Computational Grids", Proceedings of 8th IEEE International Conference on Advanced Computing and Communications, 2000

[12] L. Budin, M. Golub, and A. Budin., "Traditional Techniques of Genetic Algorithms Applied to Floating-Point Chromosome Representations," Proceddings of the 41st Annual Conference KoREMA, pp.93-96, Opatija, 1996,

[13] M. Srinivas and L.M. Patnaik, "Genetic Algorithms: A Survey," IEEE Computer 27, 617-26, June 1994

[14] K. Holtman, "CMS Requirements for the Grid", in Proceedings of the International Conference on Computing in High Energy and Nuclear Physics (CHEP2001), 2001

[15] K. Holtman, "HEPGRID2001: A Model of a Virtual Data Grid Application", in Proceedings of HPCN Europe2001, 2001

[16] "Models of Networked Analysis at Regional Centers for LHC Experiments (MONARC) Phase 2 Report", 2000

[17] M.R. Garey and D.S. Johnson, "Computers and Intractability, a Guide to the Theory of NP-Completeness", W.H. Freeman and Company, New York, 1979

[18] A.Y. Zomaya, F. Ercal, and S. Olariu., "Solutions to Parallel and Distributed Computing Problems", A Wiely-Interscience Publication, 2001

[19] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, S. Tuecke,., "The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets", Journal of Network and Computer Applications, 2001

[20] Particle Physics Data Grid, http://www.ppdg.net/

[21] Grid Physics Network, http://www.griphyn.org/

[22] EU DataGrid Project, http://www.eu-datagrid.org/

[23] Earth Science System, http://www.npaci.edu/Thrusts/ESS/projects/geography.html

[24] Bioinformatics Infrastructure for Large-Scale Analyses, http://www.npaci.edu/Alpha/bioinfo.html