A Hybrid Genetic Algorithm for Optimization of Scheduling Workflow Applications in Heterogeneous Computing Systems

Saima Gulzar Ahmad[a], Chee Sun Liew[a], Ehsan Ullah Munir[b], Tan Fong Ang[a], Samee U. Khan[c]

[a]*Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur, Malaysia.*
[b]*Department of Computer Science, COMSATS Institute of Information Technology, Quaid Avenue, Wah Cantt, Pakistan.*
[c]*Department of Electrical and Computer Engineering, North Dakota State University, Fargo, ND 58108-6050, USA.*

Abstract

Workflow scheduling is a key component behind the process for an optimal workflow enactment. It is a well-known NP-hard problem and is more challenging in the heterogeneous computing environment. The increasing complexity of the workflow applications is forcing researchers to explore hybrid approaches to solve the workflow scheduling problem. The performance of genetic algorithms can be enhanced by the modification in genetic operators and involving an efficient heuristic. These features are incorporated in the proposed Hybrid Genetic Algorithm (HGA). A solution obtained from a heuristic is seeded in the initial population that provides a direction to reach an optimal (makespan)solution. The modified two fold genetic operators search rigorously and converge the algorithm at the best solution in less amount of time. This is proved to be the strength of the HGA in the optimization of fundamental objective (makespan) of scheduling. The proposed algorithm also optimizes the load balancing during the execution side to utilize resources at maximum. The performance of the proposed algorithm is analyzed by using synthesized datasets, and real-world application workflows. The HGA is evaluated by comparing the results with renowned and state of the art algorithms. The experimental results validate that the HGA outperforms these approaches and provides quality schedules with less makespans.

Keywords:
Workflow, Genetic Algorithm, Heuristic, Directed Acyclic Graphs.

Email addresses: saimagulzarahmad@siswa.edu.um.my (Saima Gulzar Ahmad), csliew@um.edu.my (Chee Sun Liew), ehsanmunir@comsats.edu.pk (Ehsan Ullah Munir), angtf@um.edu.my (Tan Fong Ang), samee.khan@ndsu.edu (Samee U. Khan)

1. Introduction

Scientific communities are dealing with the experimentation and simulations that involve huge amount of data. Many scientific fields, such as Astronomy, Bio-informatics, Meteorology, Environmental Science, and Geological Sciences deal with large-scale data [1]. Handling such an amount of data has become a great challenge. The increased complexity, the heterogeneity of the scientific applications and the execution platforms cause additional challenges in the big data management [2]. Scientists use workflow management systems (WMS) to handle such large-scale experiments. The WMS organize and manage large-scale distributed data. Moreover, the WMS simplify the complexity to run data-intensive applications [3].

Many WMS have been developed over the last decade and are widely used by the scientific world, such as Pegasus [4], Swift [5], Kepler [6], Taverna [7], Trident [8], and Konstanz Information Miner (KNIME) [9]. The workflow optimization is a vital part of all WMS that takes place during resource mapping, and scheduling in workflow life-cycle [10]. An efficient resource allocation strategy directly effects the WMS overall performance. Therefore, scientific community has been putting their efforts to propose novel techniques for the workflow optimization and overcome the challenges that emerge with the development in scientific applications.

The major contribution of this research is a new workflow scheduling algorithm, namely the Hybrid Genetic Algorithm (HGA). The HGA is a genetic algorithm (GA) guided by a heuristic to pursue an optimal (makespan) schedule in an efficient manner. Genetic algorithms (GAs) have the capability to search large problem spaces. The heuristic helps GA to enhance the performance. The genetic operators (crossover and mutation) are modified that help the HGA to explore most of the problem space in less amount of time and ultimately the HGA converges. The workflows are usually modeled as Directed Acyclic Graphs (DAGs) [11], in which the nodes or vertices represent tasks, and the edges represent the precedence or data dependencies among the tasks. The same workflow model has been adopted in our work. The behavior of the proposed algorithm is analyzed with the diverse datasets, including synthesized, and real-world application workflows e.g Montage, and CyberShake [1]. Montage and CyberShake workflows are common choice of the researchers as benchmarks to validate the performance of workflow scheduling algorithms. These workflows have different characteristics and include most of the workflow patterns that are required to be analyzed during the performance evaluation of any novel scheduling algorithm. Random workflows provide grounds to check the behavior of proposed algorithm for different sizes and shapes of workflows having different values of out-degree and communication to computation ratio.

The rest of the paper is organized as follows. Section 2 provides the background. The related work is summarized in Section 3. Section 4 describes the proposed

[1] confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator

algorithm. The performance analysis and discussion are presented in Section 5, and Section 6 concludes the paper.

2. Background

Workflows have simplified the management of data-intensive applications by decomposing the applications into smaller tasks that need to be completed in a sequence to reach the results. The concept of workflows has reduced the complexity of large-scale experiments [10]. The WMS have been designed to organize and manage large data using a step-wise approach [12]. The major stages of a WMS are shown in Fig. 1. Scientific experiments and simulations generate huge amounts of data. Different WMS mandate inputs in specific formats. The workflow composition stage constructs a high level workflow (abstract workflow), that identifies the software component and data required for the particular execution without any details of physical execution resources. In the next stage, the abstract workflow is mapped to the physical resources and provides an executable plan in the form of a concrete workflow. Resource allocation is optimized in the mapping stage. The mapped workflow is then scheduled on the physical resources and executed. The performance of the workflow execution is monitored and results are gathered. The resource allocation is optimized in the mapping stage and scheduling during the execution [2]. Efficient scheduling can enhance the system performance [13]. We focus on the latter stage of the optimization in this paper. The HGA is a scheduling algorithm that optimizes the makespan (schedule length) of the workflow and balances the load across all of the available resources.

The paper is based on our previously proposed algorithm Performance Effective Genetic Algorithm (PEGA) [14]. In this paper we presents a hybrid approach by using a schedule of well-known heuristic Heterogeneous Earliest Finish Time (HEFT) in the initial population. In-addition load balancing procedure enhances the equal load distribution among resources. These features provide strength to the proposed algorithm to provide better schedules with lower makespan. HEFT is a well-known list-based scheduling heuristic that outperforms a number of algorithms [15]. HEFT schedule is used as a seed in the initial population of the proposed hybrid approach. List-based scheduling is a two-phase process. In the first phase, the task-priority list is generated, and in the second phase, the processors are allocated. In the HEFT, a priority list is generated in the descending order of up-ward rank or the b-level. The processors allocation for tasks is based on the earliest finish time. Following are the attributes used in HEFT that are b-level, earliest start time, and earliest finish time. These attributes are defined and briefly discussed below.

- Upward rank (b-level):

$$rank_b(v_n) = \overline{w_n} + max_{v_{m \in succ(v_n)}}\{\overline{d_{nm}} + rank_b(v_m)\}, \tag{1}$$

  where $\overline{w_n}$ is the average execution cost (it is the amount of time required to execute a task on execution node), $\overline{d_{nm}}$ is the average communication cost

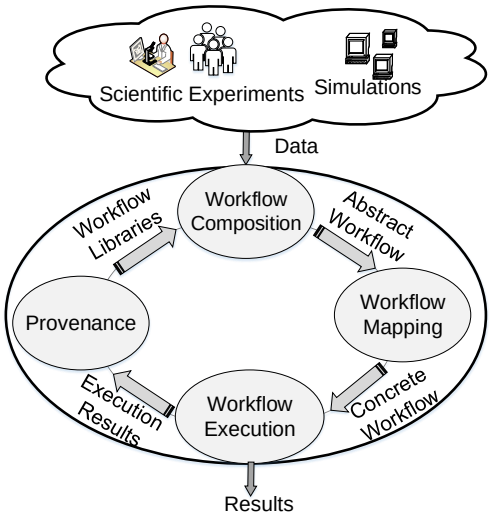


Figure 1: Major stages of a generic workflow management system.

(it is the cost of data transfer from a parent to child node in a workflow) between nodes n and m, and v_m is the successor of v_n. The up-ward rank of each of the task is calculated by the above recursive function and a task priority list is generated by a descending order of the corresponding upward rank values.

- Earliest start time (EST):
  The earliest time of a processor p_k for a task v_n is defined as:

$$EST(v_n, p_k) = max\{avail[k], max_{v_{m \in pred(v_n)}}(AFT(v_m) + d_{nm})\}, \quad (2)$$

  where $avail[k]$ is the time when processor k is ready to execute a new task, $AFT(v_m)$ is the actual finish time of task v_m on processor p_k, d_{nm} is the communication cost of tasks v_n and v_m, and the task v_m is predecessor of v_n. Task v_n can have more than one predecessors. Therefore, the maximum sum of the AFT and d_{nm} among the predecessors will be considered in Eq. 2, which will cause the maximum delay in the execution of the task v_n.

- Earliest finish time (EFT):
  The EFT is the attribute that determines the processor allocation. The EFT of a processor p_k for task v_n can be defined as:

$$EFT(v_n, p_k) = \overline{w_n} + EST(v_n, p_k), \quad (3)$$

  where $\overline{w_n}$ is the average execution cost of task v_n across all of the processors and EST is the earliest start time of a processor p_k for task v_n, which is already defined in Eq. 2.

3. Related Work

The workflow scheduling problem has been widely studied during the last couple of decades. Number of heuristics and evolutionary algorithms has been proposed to solve the workflow scheduling problem. The Modified Critical Path (MCP) [15] is a scheduling algorithm that uses the As Late As Possible (ALAP) parameter to create the priority list of the workflow tasks. The ALAP parameter is the maximum time a task n can be delayed at the latest without violating the precedence constraints. Thereafter, the tasks are arranged in a descending order in terms of the ALAP parameter. The resource allocation is carried out by allocating the tasks on the machine executing them in the earliest finish time.

In meta-heuristics, there are different categories of techniques, such as GAs [14], ant colony optimization [16], and particle swarm optimization [17] that are available in the literature to solve the task scheduling problem. However, GAs gained more popularity among researchers due to the ease of hybridization with other paradigms.

Various hybrid approaches can be found in literature however, our work focuses on a hybrid approach that uses both a heuristic and a genetic algorithm to try to reach an optimal (makespan)solution for workflow scheduling. Various hybrid algorithms have been proposed by combining heuristics with genetic algorithms. A Hybrid Successor Concerned Heuristic-Genetic Scheduling (HSCGS) algorithm was presented in [18], in which the authors, combined a heuristic and a GA. In the first phase, the heuristic named Successor Concerned List Heuristic Scheduling (SCLS) was employed to generate a schedule. The SCLS is a list-based heuristic in which the priority list of tasks was formed based on the up-ward rank and successor of tasks. In the second phase the schedule generated by SCLS heuristic is incorporated in a GA. After number of generations the algorithm converges and provides a schedule with reasonable schedule length. Another recently proposed algorithm, Performance Effective Genetic Algorithm (PEGA) [14] optimizes makespan using a hybrid approach. However, the time complexity of PEGA is high. The PEGA only optimizes makespan while the HGA reduces makespan as well as provides a load balanced schedule. In addition, a heuristic is also incorporated in the HGA that enhances the performance of the HGA by directed search.

A recently proposed GA, Multiple Priority Queues Genetic Algorithm (MPQGA) [19] exploits multiple queues of the tasks (priority lists) in a GA. The chromosomes are represented by the priority lists produced for the DAG by b-level, t-level, and sum of both parameters. The b-level ($rank_b(v_n)$), and t-level ($rank_t(v_n)$) values of each task are calculated by Eq. 1, and Eq. 4 respectively.

$$rank_t(v_n) = max_{v_{m \in pred(v_n)}}\{\overline{w_n} + \overline{d_{nm}} + rank_t(v_m)\}, \tag{4}$$

where $\overline{w_n}$ is the average execution cost, $\overline{d_{nm}}$ is the average communication cost between nodes n and m, and v_m is the predecessor of v_n. The t-level (downward rank) of each of the task is calculated by the above recursive function given in Eq. 1 and a task priority list is generated by an ascending order of the corre-

sponding values of t-level.

The initial population consists of these priority queues based chromosomes and the mapping of tasks on the processors is performed based on earliest finish time parameter. Fitness for each chromosome is then computed using roulette wheel method and fit chromosomes are selected for genetic operations. Single point crossover and swap mutation operation comprises the genetic operations of the MPQGA. Our proposed algorithm differs from the MPQGA by the chromosome representation because in the HGA chromosome length is twice the number of tasks in a DAG and half of the chromosome consists of the random processor allocation to each of the task and the other half represents the priority list of tasks within the DAG. The HGA dominates the MPQGA from its two-fold crossover and mutation operations that are twice as efficient as compared to the traditional genetic operations. Load balancing is an additional strength of our proposed algorithm.

Daoud and Kharma have proposed a two phase algorithm, named the Hybrid Heuristic Genetic Algorithm (H2GS) [20]. In the first phase, schedules in the form of chromosomes are generated by employing the Longest Dynamic Critical Path (LDCP) heuristic. The LDCP generates schedules and thereafter such schedules are used in the initial population of a customized GA, called the Genetic Algorithm for Scheduling (GAS). The produced schedules work as catalyst and support GAS during the second phase to reach the resulting schedule. A two-dimensional chromosomal representation used in the GAS and the customized operators are used to search the problem space. In our proposed technique, we have used a comparatively less complex one-dimensional direct chromosome representation. The genetic operators (crossover and mutation) are modified to a two-fold operators that enhance the process of search to pursue an optimal (makespan) solution. Based on the aforementioned features, the HGA has a capability to arrive efficiently at the best solution.

In [21] an improvement of well-known multi-objective GA, NSGA-II is presented. In the market oriented grid environment the price for the workflow execution is an important constraint. Therefore, in addition to makespan, price is also considered as the objective function in the proposed algorithm. A fuzzy system is implemented to optimize the third objective that is load balancing. The proposed algorithm improved these parameters but the complexity is increased considerably. However, the improvement in the makespan, and load balancing can be achieved by the HGA with much less time.

4. The Proposed Algorithm HGA

4.1. Problem Statement

Consider an appliocation that can be represented as workflow and modeled as DAG. A DAG can be given by a tuple (V, D, P), where V is the set of n number of tasks, D is the set of edges representing the precedence between the

pair of tasks, and P is the set of the available processors. The elements of the DAG tuple are:

$$V = \{v_1, v_2, v_3, \ldots, v_n\},$$
$$D = \{d_1, d_2, \ldots, d_n\}$$

where d represents edge between two nodes,

$$P = \{p_1, p_2, p_3, \ldots, p_m\}.$$

The element D in the DAG representation illustrates the precedence of tasks, which means that for any pair of tasks $(v_i, v_j) \in V$, the execution of v_j cannot be started until v_i is completed. That is to say that, v_j is a successor of v_i. Any task v_i without a predecessor is called as the entry task, and a task v_j with no successor task is called the exit task. There can be more than one entry task and one exit task. During scheduling, the precedence must not be violated, otherwise the schedule will be invalid. The cost to move data from one processor to another is termed as the communication cost represented by element D in the DAG tuple. For any pair of tasks the edge between them is given by $d_i \in D$, the communication cost is represented by d_i that will be a real number.

The execution cost of any task v_i on a processor j from the set of available processors P is represented by w_{ij}. The execution costs of all of the tasks on each of the processor are the numerical values that are represented by an n x m matrix, with n being the number of tasks in the application and m being the number of processors. The workflow scheduling problem is defined as the assignment of a set of tasks V to the set of resources P, such that:

- the precedence constraint is preserved;
- the schedule length is reduced; and
- the load is balanced among processors.

The objectives of the proposed workflow scheduling algorithm are to optimize the makespan and to balance the load. In this paper, it is assumed that information, such as workflow size, task precedence, and resources are known a prior. Such a category of algorithms, in which all of the information is known prior to scheduling, is called deterministic or static algorithms [22]. The workflow scheduling is proven to be NP-hard [23] problem.

4.2. The Proposed Algorithm

The proposed algorithm is a hybrid approach that is a combination of heuristic and genetic algorithm. The outline of the HGA is given in Algorithm 1. The HGA starts with a set of schedules in the form of initial population. The population size N_p, number of generations N_g, crossover, mutation and elitism probabilities are assumed to be user input. Algorithm 1 (line 1) starts with set of chromosomes called initial population. Then the HEFT schedule is seeded into

the initial population. The HEFT heuristic provides guidance to the algorithm that improves the performance of the HGA. The directed search helps the HGA to converge after fewer numbers of generations. The variable N_g represents the number of generations that is used as a termination criterion. The outer for loop at line 3 will run for N_g times.
The population consists of the individuals called chromosomes. The chromosome representation is direct and each of the chromosomes consists of two parts. The left half represents the resource allocation and its length is equal to the number of nodes in the DAG. The genes represent the processor numbers from 1 to P, where P is the maximum number of available processors. The size of the second half is also the number of nodes within a DAG, which represents the sequence or order of the tasks to be scheduled. The precedence constraints must not be violated; otherwise the chromosome will be an invalid chromosome and will not represent a correct schedule. Each chromosome represents a valid schedule. An example chromosome is shown in Fig. 2, and the corresponding schedule on three processors in the form of Gantt Chart is shown in Fig. 3.

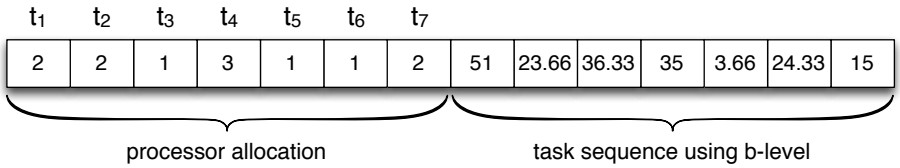


Figure 2: Chromosome representation.

The first half is randomly generated, while the second half is determined by the up-ward rank of tasks using Eq. 1. The sequence of tasks in the example chromosome shown in Fig. 2 will be in the descending order of the b-level. The task priority list will be {1, 3, 4, 6, 2, 7, 5}. If the tasks are mapped on the processors according to the processor allocation shown in the first half of chromosome, then the resulting schedule length will be 51 time units. The corresponding schedule is shown in Fig. 3.

In evaluation process (Algorithm 1, line 4, 5, 6), each schedule in each generation is evaluated based on the fitness function. The fitness of a chromosome x can be defined as:

$$f(x) = c/makespan(x), \tag{5}$$

where c is a constant and the makespan is defined as:

$$makespan(x) = F.T(t_{exit}), \tag{6}$$

with $F.T(t_{exit})$ being the finish time of exit node. In case there are more than one exit task, the makespan is defined by Eq. 7:

$$makespan(x) = max(F.T(t_i)), i = \{1, 2, 3, \ldots, n\}. \tag{7}$$

Algorithm 1: Hybrid Genetic Algorithm

Input: N_p Population Size,
α Elitism Rate,
β Mutation Rate,
N_g Number of Generations.
Output: S Near Optimal Solution.

Initial population generation of size $N_p - 1$;
Seed HEFT schedule as a chromosome in N_p;
for $i = 1$ *to* N_g **do**
    /* Evaluation */
    for $j = 1$ *to* N_p **do**
        Compute fitness value of each chromosome;
    end for
    /* Elitism */
    Number of Elite Chromosomes $E = \alpha \times N_p$;
    Select E Chromosomes having best fitness values as N_E;
    Selection Routine as shown in Algorithm 2;
    Crossover Routine as shown in Algorithm 3;
    Mutation Routine as shown in Algorithm 4;
    /* Next generation */
    /* N_s are the selected chromosomes */
    $N_p = N_E + N_s$;
    Load Balancing Routine as shown in Algorithm 5;
end for
Return Near optimal (makespan) solution S;

Algorithm 2: Selection Routine

Pick two chromosomes at random from initial population;
for $k = 1$ *to* N_g **do**
    if $f(x) < f(y)$ **then**
        Select chromosome x as N_s;
    end if
end for

Algorithm 3: Crossover Routine

Data: Number of crossover $C = N_s/2$
for $k = 1$ *to* C **do**
    Randomly select two chromosomes x_a and x_b for mating;
    Off springs by single point crossover is x_c and x_d;
    Off springs by double point crossover is x_e and x_f;
    Compute fitness values of x_c, x_d, x_e and x_f;
    Select two best off springs as N_s;
end for

Algorithm 4: Mutation Routine

Data: Number of mutations $M = \beta x N_s$
for $l = 1$ *to* M **do**
    Randomly select a chromosome x_i from N_s;
    Perform single point mutation off spring = x_j;
    Perform double point mutation off spring = x_k;
    Compute fitness of x_j and x_k;
    Select offspring with better fitness value as N_s;
end for

Algorithm 5: Load Balancing Routine

Calculate LB parameter for all chromosomes using Eq. 8;
Select 50% of chromosome with higher LB values;
forall the *selected chromosomes* **do**
    Identify the overloaded processor p_l from chromosome y_{ol};
    Randomly select a processor p_i such that $p_i \neq p_l$;
    Replace the p_{ol} by p_n at two places;
    A new chromosome x_{lb} is formed;
    if $f(x_{lb}) > f(y_{ol})$ **then**
        Replace x_{lb} by y_{ol};
    end if
end forall

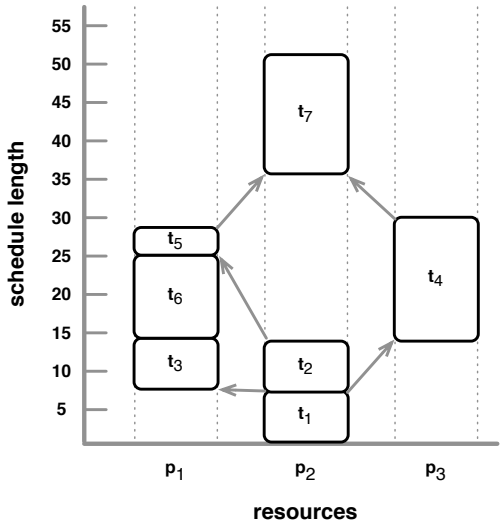


Figure 3: Corresponding schedule of example chromosome shown in Fig. 2.

The evaluation is carried out based on the fitness function defined in Eq. 5. Elitism at line 7 and 8 is also introduced in HGA that copy best chromosomes from each generation to the next generation. It prevents HGA to lose quality solutions. The details of selection at line 9 is given in Algorithm 2 in which fit chromosomes are selected by employing a binary tournament selection for genetic operations. The selected chromosomes then undergo the crossover and mutation. Both genetic operations are modified in the HGA that differentiate the proposed work from existing GAs. The strength of both operators is doubled by combining single and double point crossover and mutation. Pseudocode of crossover and mutation routines are presented in Algorithms 3, and 4, respectively. At the end of each generation, a neighborhood search is performed using a load balancing function. The pseudo-code of load balancing process is given in Algorithm 5. Each generation is completed at line 14 of Algorithm 1 and for next generation algorithm jumps to line 3 until termination criteria is met.

The resources must be utilized in such a fashion so that the resources should neither be overloaded nor stay idle for a long time. The load balancing Algorithm 5 distributes the tasks among the processors in such a way that all of the processors complete the job with the minimum difference in the finish time. At the end of each iteration, the Algorithm 5 is executed that enhances the quality of the schedules in terms of schedule length by balancing the load across all of the processors.

The load balancing parameter (LB) is defined in Eq.8, that is used in Algorithm 5 to determine the quality of the load balance in each of the schedule:

$$LB = S_{Length} - min\{FT_1, FT_2, FT_3, \ldots, FT_n\}, \tag{8}$$

where S_{Length} being the schedule length of the corresponding chromosome and FT_n is the finish time of processor n. The greater the LB value the worse is the load balancing. The complexity of the proposed algorithm is $nm + n^2$, where n is the number of generations and m is the population size. Increase in any of these two factors the runtime of HGA will also increase.

5. Performance Analysis and Discussion

In this section the performance of the proposed algorithm is analyzed. The HGA is evaluated by using datasets with diverse characteristics and the achieved results are compared with the following selected algorithms. We selected heuristics (MCP and HEFT), a generic evolutionary algorithm (PEGA), and recently proposed hybrid genetic algorithms (MPQGA and HSCGS) for comparative analysis with the proposed algorithm. These selected algorithms based on different approaches provide sound grounds to study and compare the behavior of the HGA. We have selected Average Schedule Length (ASL) of 1000 runs as a performance metric. The range bars for ASL of all algorithms shows a 95% of the confidence interval for corresponding ASL. This shows that for any other workflow of similar type the schedule length of that workflow would be in the given interval with 95% of certainty. In some of the bar charts the confidence interval is not distinguishable from the mean value for the scale used in those graphs.

The proposed algorithm is evaluated by simulations in a target system that is heterogeneous. The resources as well as the network links both are heterogeneous in the execution environment. Since the tasks are different based on the type of the workloads, therefore the heterogeneity of execution nodes and tasks both are considered in the heterogeneous execution times of each task on execution nodes. Similarly, the heterogeneity of network links and the edges are implicitly considered as the varying communication costs mentioned on the edges. After number of simulations the most suitable parameters for the proposed algorithm that provides best results with the crossover and mutation probabilities of 0.8 and 0.02 respectively. The population size and the number of generations both attributes are taken as 100 to simplify the simulations.

HEFT is a well-known heuristic that provides good schedules. We seed HEFT schedule in the initial population of HGA that accelerates the process to reach an optimal (makespan). We have made simulations with and without HEFT solution in the initial population of HGA. The results are given in Table 1. The results show that the HEFT solution in the initial population accelerates the runtime of the algorithm.

The datasets selected for the simulations include synthesized, and workflows of real-world applications. The characteristics of the datasets used are tabulated in Table 2. Researchers have been working on the workflow patterns [2] and many

[2] www.workflowpatterns.com

Table 1: Comparision of results with and without HEFT Seed in Initial Population

Workflow Nodes	**HGA with HEFT (sec)**	**HGA without HEFT (sec)**
100	6.55	7.94
500	30.66	39.99
1K	72.11	80.07
2K	143.52	159.45

benchmarks are available for the performance evaluation of new algorithms. As an example, the NAS Grid Benchmarks[3] (NGB) are designed for the performance evaluation of parallel and distributed systems. The benchmarks include four classes of problem obtained from computational fluid dynamics (CFD) applications: Embarrassingly Distributed (ED), Helical Chain (HC), Visualization Pipeline (VP), and Mixed Bag (MB) [24]. These benchmarks represent the typical applications that run on the heterogeneous systems like grid. Each of these consists of computational tasks achieved from NAS parallel Benchmarks (NGB). They symbolize the typical applications that run on the heterogeneous systems like grid. ED represents an important class of applications called parameter studies, in which same program is run several times independently but with different set of input parameters. HC represents long chains of sequential computations. It is the execution of repeating a process one after another. VP is compound of different structured processes. It contains some degree of parallel as well as sequential flow of tasks. The structure of MB is similar to VP but the degree of parallelism, and heterogeneity increases. The selected datasets, i.e. Montage, CyberShake, and Gaussian Elimination are complex and large workflows that consist of most of the NGB classes of problem. These are real-world workflows that demonstrate realistic execution behavior on distributed environment. Together with the synthesized workflows that are generated randomly, we have a comprehensive set of test loads to analyze the performance of the proposed algorithm.

Montage, Cybershake [25], and Gaussian Elimination [22] are used as workflow benchmarks to evaluate the proposed algorithm. These workflows represent real-world problems.

5.1. Montage and CyberShake Benchmarks

Montage is an astronomical image mosaic engine created by NASA that is used to generate a mosaic of the sky. The input astronomical images are combined to form the final mosaic. The geometry of the final mosaic is determined

[3] www.nas.nasa.gov/publications/npb.html

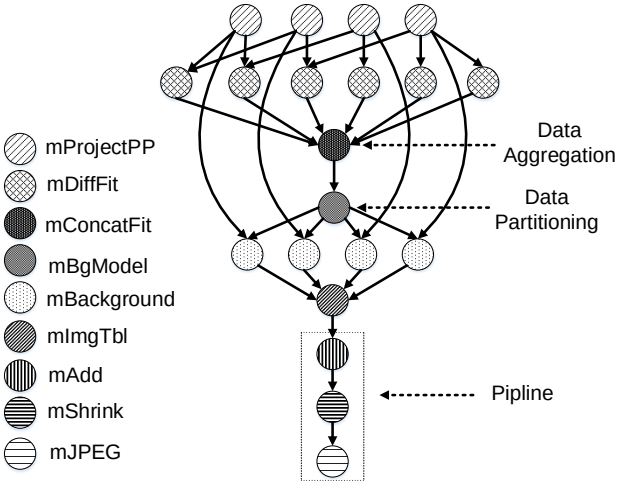
Table 2: Characteristics of datasets

Workflows	**Type**	**Source**	**Nature**	**Nodes**	**Shape**
Montage	Real	[25]	Regular	25,50,100	Fixed 4(a)
CyberShake	Real	[25]	Regular	30,50,100	Fixed 4(b)
Gaussian Elimination	Simulated	Generated	Regular	14,20, ...104,119	Fixed 8
Random	Synthesized	Generated	Random	20,40,60, 80,100	Varying

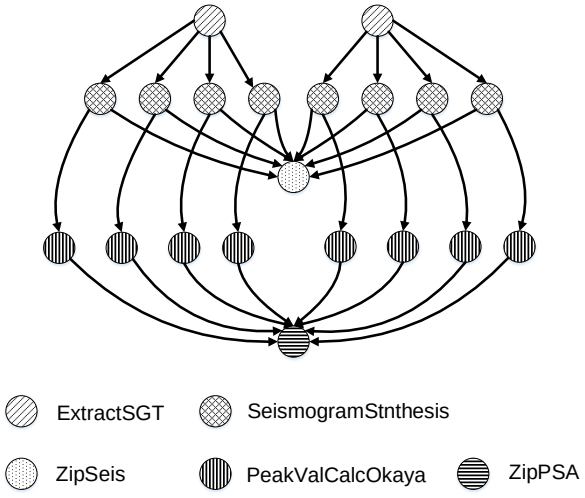
by the input images that can be represented as a workflow. Fig. 4(a) illustrates the structure of a small size montage having 20 nodes. In Montage, there are a lot of jobs with short execution time, such as mProjectPP, mDiffFit, mBackground, and mJpeg that are required to be executed on a number of different data items. On the other hand, some jobs such as mConcatFit, mBgModel, and mAdd take much longer time to execute. The CyberShake workflow is used by the Southern California Earthquake Center (SCEC). The CyberShake workflow is used to identify the earthquake hazards within a region. CyberShake is a relatively simple workflow but can handle large datasets. As an example, a small 20 node CyberShake workflow is shown in Fig. 4(b). CyberShake is compute-intensive as well as data-intensive workflow. The details regarding the characteristics of both workflows can be obtained from [25], in which the authors provided the details of the execution of six diverse workflows including Montage and CyberShake. Therefore, these workflows with different characteristics make them highly suitable to be used for the validation of our proposed work.

We have performed extensive simulations using Montage and CyberShake workflows to evaluate the behavior of the HGA. We obtained the data of these benchmark workflows from [26]. The obtained data was the complete details of previous executions of Montage and CyberShake. All of the algorithms were tested under the same conditions to observe the comparative results. The performance metric used for the performance results is ASL for 1000 runs. In the plots the horizontal axis represents number of processors (P). The plots in Fig. 5 and Fig. 6 show the behavior of algorithms in terms of ASL when the number of processors increase. The HGA outperformed the state of the art algorithms, with its performance becoming better as the processors increase. The average percentage improvement of the HGA over PEGA, MCP, HEFT, HSCGS, and MPQGA was 73.97%, 59.55%, 29.85%, 12.58%, and 6.32%, respectively for 30 nodes CyberShake (Fig. 5(a)).

Similar results were obtained for CyberShake of 50, and 100 nodes with improved performance as the size of workflow increased. The proposed algorithm showed 19.53% and 25.05% improvement over the HSCGS, and 29.85% and 45.99% from the HEFT for 50 (Fig. 5(b)), and 100 (Fig. 5(c)) nodes CyberShake respectively. However, the HGA significantly outperforms the PEGA and MCP in case of the CyberShake workflows. The HGA outperformed the MPQGA by



(a)



(b)

Figure 4: Workflow benchmarks: (a) Montage (b) CyberShake [26]

11.3% for 50 nodes, and 13.86% for 100 nodes Cyberhake workflows. The better performance of the HGA with the increasing size in CyberShake workflows proves the scalability of the HGA.
Similar experiments were carried out with the Montage workflows of 25, 50, and 100 nodes. The results in Fig. 6 show a better performance of proposed algorithm, as compared to the other five algorithms. In case of the 25 nodes (Fig 6(a)) Montage workflow the HGA is better than the PEGA by 32.07%, MCP by 16%, HEFT by 6.7%, and HSCGS by 4.97%. However, minimum average percentage improvement was 5.4% and 8.56% over MPQGA for 50(Fig 6(b)) , and 100 (Fig 6(c)) nodes Montage workflows respectively. For the rest of the algorithms the average percentage improvement is comparatively high. It is noticeable that the performance of proposed algorithm is better for CyberShake workflows as compared to Montage workflows. Both types of workflows have different characteristics, CyberShake is data-intensive workflow as compared to Montage workflow.

The bar charts of Fig. 7 show the overall performance of the CyberShake and Montage workflows of different sizes. The average schedule lengths obtained from proposed algorithm are less considerably than the other five algorithms. The HGA performance improved from MPQGA by 6.54%, HSCGS by 7.99%, HEFT by 10.73%, MCP by 17.7%, and PEGA 29.33% for Cybershake workflows (Fig. 7(a)). In case of the Montage workflows (Fig. 7(b)) average percentage improvement of HGA over MPQGA is 5.46%, and rest of the results are also similar to CyberShake workflow. The proposed algorithm outperforms aforementioned algorithms, however the performance is significantly better than PEGA. In the proposed algorithm the heuristic accelerates the process to search an optimal (makespan) schedule and modified genetic operators help to search the problem space efficiently. These features dominate the HGA among other algorithms and it outperforms completely.

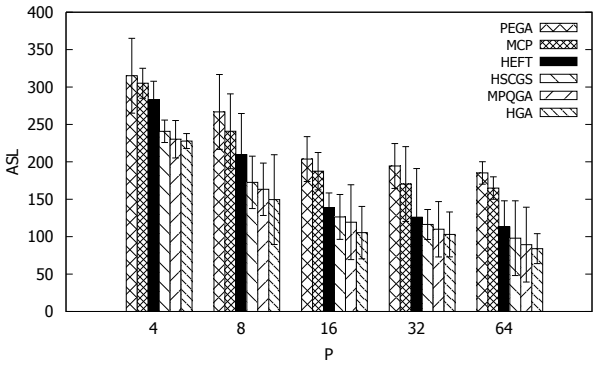
5.2. Gaussian Elimination

The Gaussian elimination algorithm generates a repeating pattern as shown in Fig. 8. The parameter m determines the size of the workflow. As an example, a small 20 nodes Gaussian elimination data flow for $m = 6$ is shown in Fig. 8 to illustrate the structure. Number of nodes for any matrix size can be calculated by using Eq. 9.

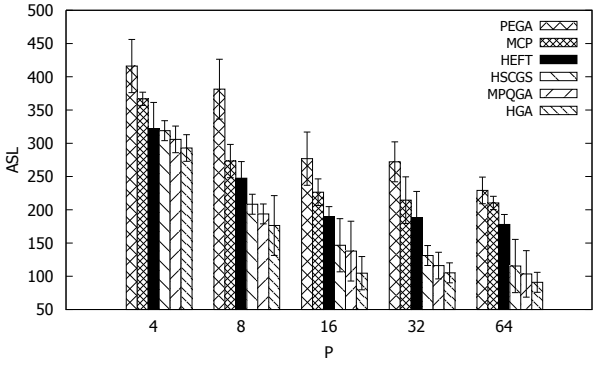
$$n = (m^2 + m - 2)/2, \tag{9}$$

where n being the number of nodes within the graph.

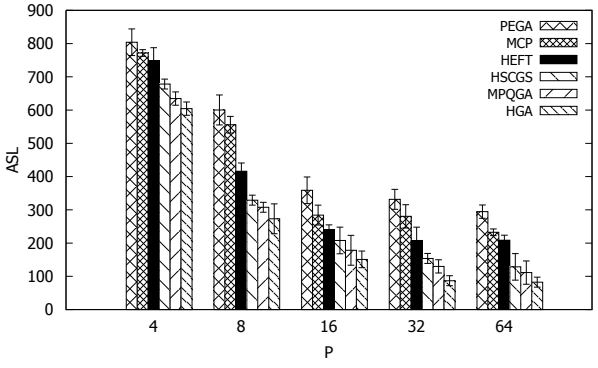
The parameter m is also called as matrix size which determines the number of nodes in Gaussian Elimination workflow by using Eq. 9. Greater the value of parameter m higher the number of nodes in the workflow. We used different values of parameter m and generated workflows of various sizes. The values of parameter m used in our experimentation are from 5 to 15, which generated workflows of suitable sizes for our simulations. HGA and aforementioned algorithms were experimented with these workflows. Fig. 9(a) shows the plot of



(a)

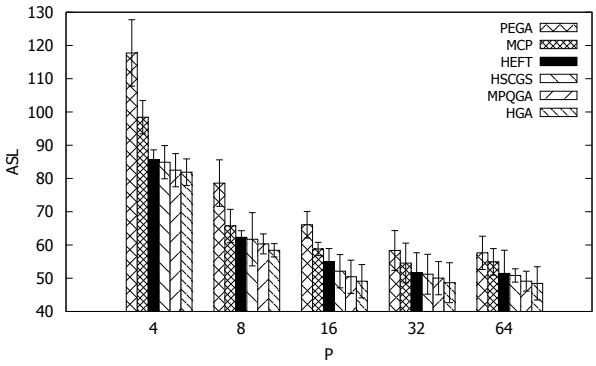


(b)

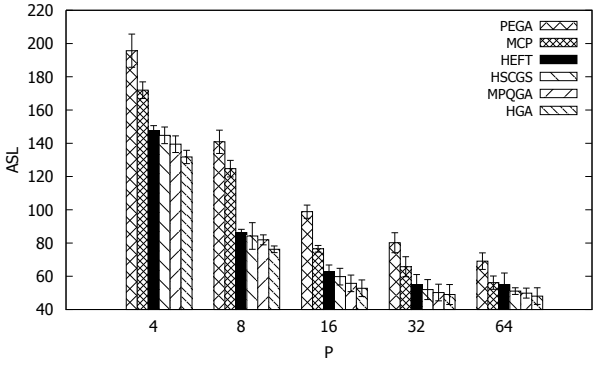


(c)

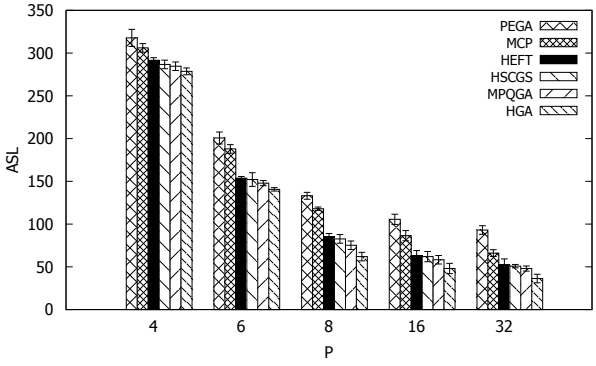
Figure 5: Performance of (a) 30 (b) 50 (c) 100 nodes CyberShake workflows.



(a)

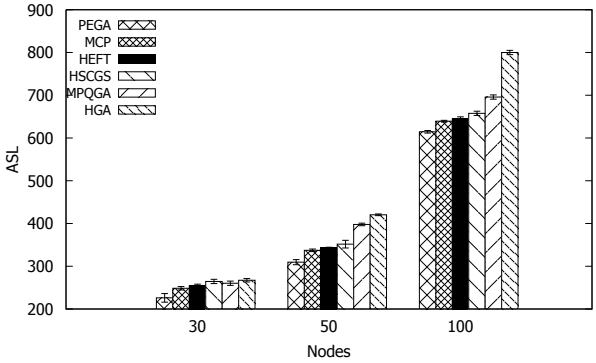


(b)

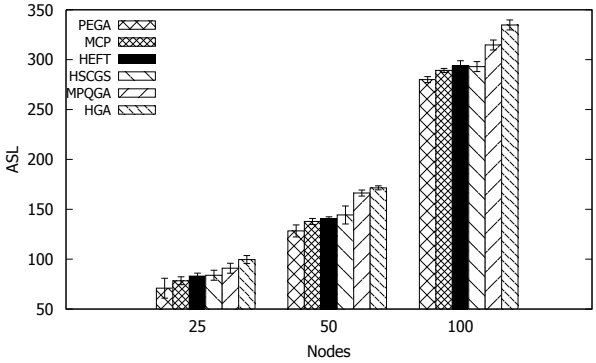


(c)

Figure 6: Performance of (a) 25 (b) 50 (c) 100 nodes Montage workflows.



(a)



(b)

Figure 7: Performance of (a) CyberShake and (b) Montage workflows of different sizes.

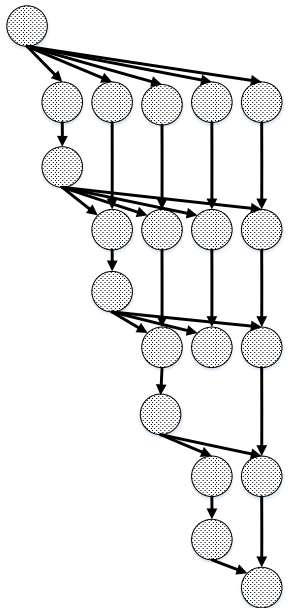


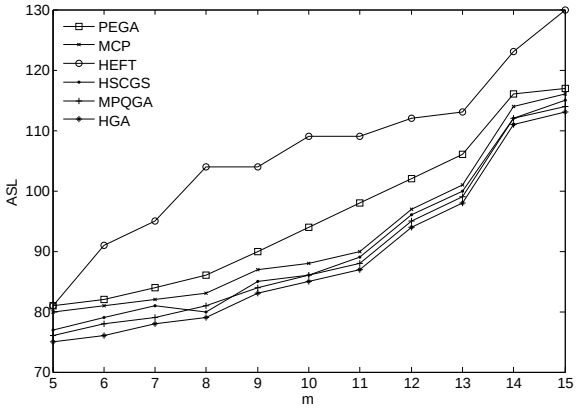
Figure 8: Gaussian elimination workflow for matrix size 6 (20 nodes).

average schedule length with the increase in the workflow size determined by the matrix size (m) taken along x-axis. We can see that as the size of workflows become bigger the average schedule lengths increase because the execution time of bigger workflows will be higher. The plot in Fig. 9(a) shows a significant average improvement of 19.6% of HGA over PEGA, which is considerable improvement as compared to other algorithms. The HGA is better than MCP by 7.86%, HEFT by 4.07%, HSCGS by 2.14%, and MPQGA by 1.32%. The performance of the HGA with increase in the processors was also investigated and the results obtained are presented in bar chart 9(b). The bar chart shows that as compared to MPQGA, and HSCGS, the proposed algorithm did not performed well for less number of processors but as processors increase the results of HGA became better. The HGA results are approximately 28% better than PEGA while about 7% better as compared to HEFT on the average. Proposed algorithm showed considerable improvement for Gaussian Elimination workflow.

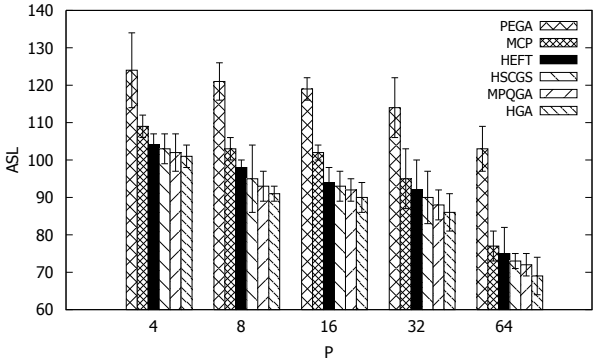
5.3. Synthesized Workflows

The synthesized workflows can be generated randomly [15] based on the following parameters and these are used for performance evaluation of the HGA.

- Workflow Size (n): The parameter n is the number of nodes in a workflow that represents the size of a workflow. Various sizes of random workflows using different values of n given in the following set were used in the performance evaluation of the HGA. $n = \{20, 40, 60, 80, 100\}$.



(a)



(b)

Figure 9: Performance results with Gaussian Elimination workflow with (a) increasing matrix size (m) and (b) increasing processors.

- Shape Parameter (α): The shape of workflow can be handled by the parameter α. If $\alpha < 1$, then longer workflows with less parallelism are generated. If $\alpha > 1$, the small size workflows with higher parallelism are generated. If $\alpha = 1$, the workflows of balanced size are generated that are neither long nor short. The HGA is evaluated with three different sizes of workflows, with α= {0.1, 1, 2}.

- Communication to Computation Ratio (CCR): The parameter CCR determines whether the workflow is computation-intensive or communication-intensive. If CCR > 1, then the workflow is communication-intensive. If CCR < 1, then the workflow is compute-intensive. If CCR = 1 then the workflow is neither communication nor computation-intensive. The values of CCR considered in experimental evaluation were 0.1, 1, and 10.

- The number of resources is represented by the parameter P, we use the number of resources as 2^x, where x = {2, 3, 4, 5, 6}.

- Out degree: This parameter determines the number of edges going out from a node. Because the workflows were random, the out degree was chosen at random.

- Range percentage of computation costs on processors (β): It determines the heterogeneity factor for processor speed. Its higher value causes significant difference in the computation costs of tasks and lower value indicates similar or equal computation costs of tasks. The values of used in simulations are 0.1, 0.5, and 1.

Variety of random workflows with different characteristics were generated to analyze the performance of proposed algorithm. Fig. 10 presents the behavior of algorithms for random 100 nodes workflow when processors increase. We must note that as the resources increase the average schedule lengths decrease up to certain extent, which indicates that execution times can be reduced by exploiting parallelism in workflows but it is limited due to the serial content in workflows. The HGA showed 50.6% over PEGA, 44.57% over MCP, 22.76% over HEFT, 13% over HSCGS, and 6.78% over MPQGA average percentage improvement. Sets of workflows 5k, 10k, and 15k with different characteristics as mentioned above were generated and simulation results are achieved with CCR values of 0.1, 1, and 10. The results are shown in Fig. 11. The HGA showed significant improvement over MPQGA, HSCGS, HEFT, PEGA, and MCP for the results achieved for CCR values of 1 and 10. However, the performance of the proposed algorithm is not convincing for CCR value 0.1. This indicates that HGA performs well for communication-intensive workflows.
The load balancing feature in the proposed algorithm is also evaluated by comparing the results achieved from algorithm with and without load balancing. Results are presented in Fig. 12. The plot shows the strength of load balancing feature of HGA as the average schedule lengths are better with load balancing. The HGA outperformed and considerable better results proved the supremacy of proposed algorithm over the other five algorithms. The time complexity of

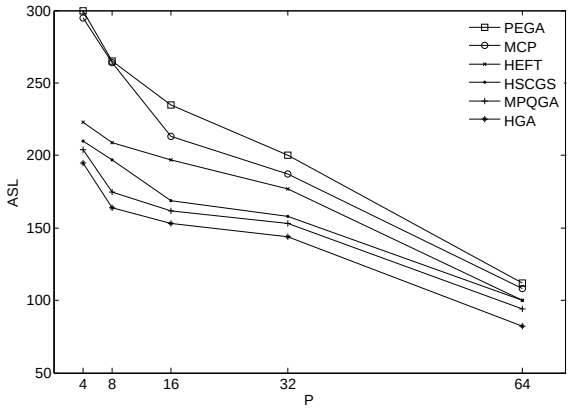


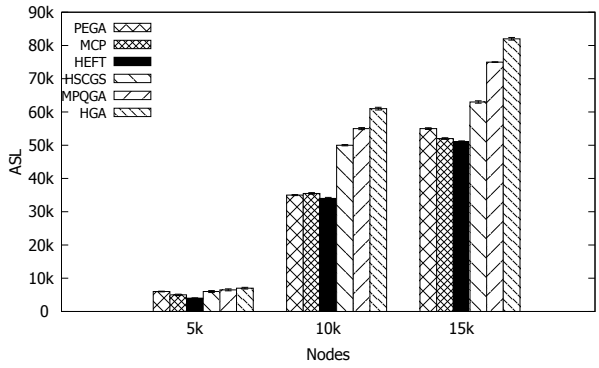
Figure 10: Performance results with random workflows with increasing number of processors.

HGA is $nm + n^2$,where n is number of generations and m is the population size. The proposed algorithm has high complexity as compared to heuristics. Since the proposed algorithm is designed for the static scheduling, therefore the limitation of high complexity of HGA might not affect the system performance.

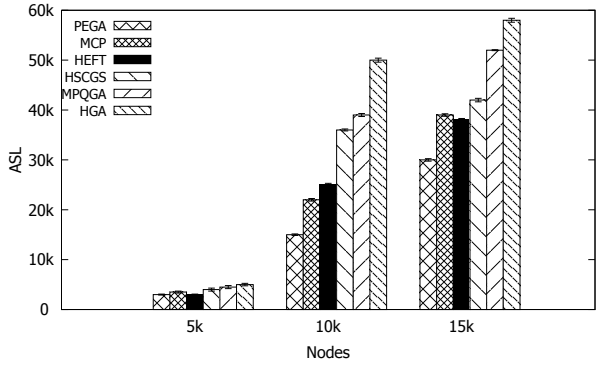
6. Conclusion

In this paper, a genetic evaluation based approach is modified and a new hybrid genetic algorithm (HGA) for workflow scheduling is presented. The proposed algorithm seeds the Heterogeneous Earliest Finish Time (HEFT) based schedule in the initial population that guides the algorithm to reach an optimal (makespan) schedule in fewer generations. Rigorous search with two fold crossover and mutation operators cover the large problem space and enhances the HGA performance. The proposed algorithm optimizes workflow schedule length with an additional feature of load balancing that ensures the optimized resource utilization. The scheduling algorithms with different approaches are compared with the HGA. The simulations with variety and different sizes of datasets show the diversity and scalability of proposed algorithm. The results prove that the HGA outperforms and the quality of schedules is better by reduced schedule lengths. The simulation with different communication to computation ratio (CCR) shows that proposed algorithm performs well for workflows with CCR values greater than 1, that is communication-intensive workflows.

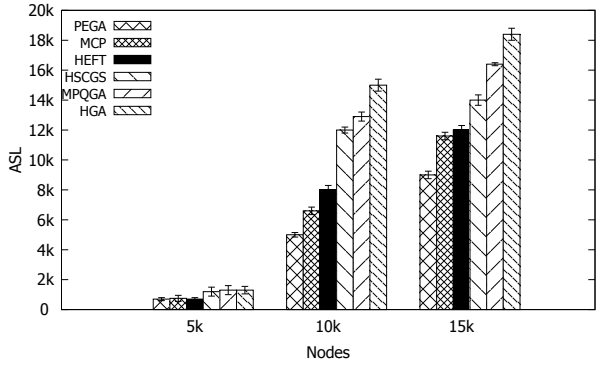
In future work we plan to investigate the performance of the proposed algorithm with more complex workloads with increased processors, workflow size, and heterogeneity. Furthermore, we plan to incorporate the optimization of data-intensive workflows because the scientific applications are becoming more data-intensive and scientific community is trying to solve the challenges caused by the big data.



(a)



(b)



(c)

Figure 11: Performance of synthesized workflows for CCR (a) 0.1 (b) 1 (c) 10.

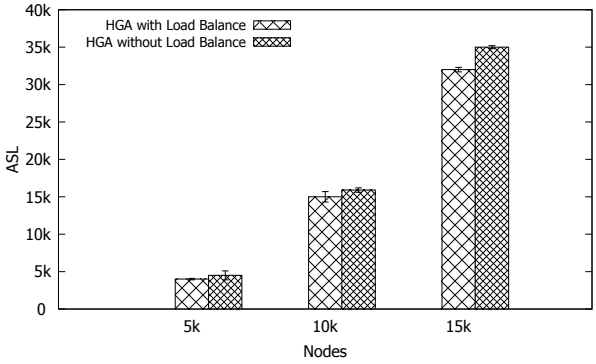


Figure 12: Performance results of HGA with and without load balancing with increasing number of nodes.

Acknowledgment

The work presented in this paper is supported by the Ministry of Education Malaysia (FRGS FP051-2013A and UMRG RP001F-13ICT).

[1] M. P. Atkinson, R. Baxter, P. Besana, M. Galea, M. Parsons, P. Brezany, O. Corcho, J. van Hemert, and D. Snelling, *The DATA Bonanza – Improving Knowledge Discovery for Science, Engineering and Business*. John Wiley & Sons, Inc., 2013.

[2] M. P. Atkinson, C. S. Liew, M. Galea, P. Martin, A. Krause, A. Mouat, O. Corcho, and D. Snelling, "Data-intensive architecture for scientific knowledge discovery," *Distrib. and Parallel Databases*, vol. 30, no. 5-6, pp. 307–324, 2012.

[3] J. Yu and R. Buyya, "A taxonomy of scientific workflow systems for Grid Computing," *SIGMOD Rec.*, vol. 34, pp. 44–49, Sept. 2005.

[4] E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good, A. Laity, J. C. Jacob, and D. S. Katz., "Pegasus: A framework for mapping complex scientific workflows onto distributed systems," *Scientific Program. J.*, vol. 13, no. 3, pp. 219–237, 2005.

[5] M. Wilde, M. Hategan, J. M. Wozniak, B. Clifford, D. S. Katz, and I. Foster., "Swift: A language for distributed parallel scripting," *Parallel Comput.*, vol. 37, no. 9, pp. 633–652, 2011.

[6] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. A. Lee, J. Tao, and Y. Zhao, "Scientific workflow management and the Kepler system: Research articles," *Concurr. Comput. : Pract. Exper.*, vol. 18, pp. 1039–1065, Aug. 2006.

[7] D. Hull, K. Wolstencroft, R. Stevens, C. Goble, M. R. Pocock, P. Li, and T. Oinn, "Taverna: a tool for building and running workflows of services," *Nucleic Acids Res.*, vol. 34, no. 2, pp. 729–732, 2006.

[8] R. Barga, J. Jackson, N. Araujo, D. Guo, N. Gautam, and Y. Simmhan, "The Trident scientific workflow workbench," in *Proceedings of the 2008 Fourth IEEE International Conference on eScience*, ESCIENCE '08, (Washington, DC, USA), pp. 317–318, IEEE Computer Society, 2008.

[9] M. R. Berthold, N. Cebron, F. Dill, T. R. Gabriel, T. Kötter, T. Meinl, P. Ohl, K. Thiel, and B. Wiswedel, "KNIME-the Konstanz information miner version 2.0 and beyond," *SIGKDD Explor. Newsl.*, vol. 11, pp. 26–31, nov 2009.

[10] E. Deelman, D. Gannon, M. Shields, and I. Taylor, "Workflows and e-Science: An overview of workflow system features and capabilities," *Future Gener. Comput. Syst.*, vol. 25, no. 5, pp. 528 – 540, 2009.

[11] Y. Xu, K. Li, L. He, and T. K. Truong., "A DAG scheduling scheme on heterogeneous computing systems using double molecular structure-based chemical reaction optimization," *J. Parallel Distrib. Comput.*, vol. 73, no. 9, pp. 1306–1322, 2013.

[12] A. L. Chervenak, D. E. Smith, W. Chen, and E. Deelman, "Integrating policy with scientific workflow management for data-intensive applications," in *7th Workshop on Workflows in Support of Large-Scale Science*, WORKS'12, pp. 140–149, 2012.

[13] E. U. Munir, S. Mohsin, A. Hussain, M. W. Nisar, and S. Ali, "SDBATS: A novel algorithm for task scheduling in heterogeneous computing systems," in *Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW), 2013 IEEE 27th International*, pp. 43–53, 2013.

[14] S. G. Ahmad, E. U. Munir, and M. W. Nisar, "PEGA: A performance effective genetic algorithm for task scheduling in heterogeneous systems," in *IEEE 14th International Conference on High Performance Computing and Communications*, pp. 1082–1087, 2012.

[15] H. Topcuoglu, S. Hariri, and Min-You, "Performance-effective and low complexity task scheduling for heterogeneous computing," *IEEE Trans. on Parallel and Distrib. Syst.*, vol. 13, no. 3, pp. 260–274, 2002.

[16] R. T. Neto and M. G. Filho, "Literature review regarding Ant Colony Optimization applied to scheduling problems: Guidelines for implementation and directions for future research," *Eng. Appl. of Artif. Intell.*, vol. 26, no. 1, pp. 150–161, 2013.

[17] J. Taheria, Y. C. Lee, A. Y. Zomaya, and H. J. Siegel., "A Bee Colony based optimization approach for simultaneous job scheduling and data replication

in grid environments," *Comput. & Oper. Res.*, vol. 40, no. 6, pp. 1564–1578, 2013.

[18] C. Wang, J. Gu, Y. Wang, and T. Zhao, "A hybrid heuristic-genetic algorithm for task scheduling in heterogeneous multi-core system," in *Algorithms and Architectures for Parallel Processing* (Y. Xiang, I. Stojmenovic, B. Apduhan, G. Wang, K. Nakano, and A. Zomaya, eds.), vol. 7439 of *Lecture Notes in Computer Science*, pp. 153–170, Springer Berlin Heidelberg, 2012.

[19] Y. Xu, K. Li, J. Hu, and K. Li, "A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues," *Inf. Sci.*, vol. 270, pp. 255 – 287, 2014.

[20] M. I. Daoud and N. N. Kharma, "A hybrid heuristic-genetic algorithm for task scheduling in heterogeneous processor networks," *J. Parallel Distrib. Comput.*, vol. 71, no. 11, pp. 1518–1531, 2011.

[21] R. Salimi, H. Motameni, and H. Omranpour, "Task scheduling using NSGA II with fuzzy adaptive operators for computational grids," *J. Parallel Distrib. Comput.*, vol. 74, no. 5, pp. 2333–2350, 2014.

[22] H. Arabnejad and J. G. Barbosa, "List scheduling algorithm for heterogeneous systems by an optimistic cost table," *IEEE Trans. on Parallel and Distrib. Syst.*, vol. 25, no. 3, pp. 682–694, 2014.

[23] M. Caramia and S. Giordani., "A fast metaheuristic for scheduling independent tasks with multiple modes," *Comput. & Ind. Eng.*, vol. 58, no. 1, pp. 64–69, 2010.

[24] R. F. V. der Wijngaart and M. Frumkin, "NAS Grid Benchmarks Version 1.0," tech. rep., NASA Advanced Supercomputing (NAS) Division, NASA Ames Reserach Center, Moffett Field, CA 94035-1000, 2002.

[25] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, and K. Vahi, "Characterizing and profiling scientific workflows," *Future Gener. Comput. Syst.*, vol. 29, no. 3, pp. 682 – 692, 2013.

[26] Pegasus Team, "Workflow Generator." https://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator, 2013.