Cloud Computing offers utility-oriented IT services to users worldwide. Based on pay-as-you-go model, it enables hosting of pervasive applications from consumer, scientific and business domains. This project focuses on improving scheduling solution for cloud computing systems. The scheduling method includes diversity detection and improvement detection operators. The proposed algorithm is based heuristic in which combines several heuristic techniques. But the proposed algorithm takes the advantage of all heuristic techniques. The solution dynamically determines and heuristic to find better candidate solution. To evaluate performance of the proposed, this project easily compares method with several implementing CloudSim (Simulator). The results indicate that Hyper heuristic significantly reduce makespan of task scheduling compared with other scheduling algorithms.

**Keywords:** Electrochemical, Escherichia Coli bacteria, Contamination event

**INTRODUCTION**

Cloud computing is typically defined as a type of computing that relies on sharing computing resources rather than having local servers or personal devices to handle applications. The main goal of cloud computing is used to traditional supercomputing, or high-performance computing power that is normally used by military and research facilities to perform tens and trillions of computations per second in consumer-oriented applications such as financial portfolios or even to deliver personalized information. Cloud computing is an approach of delivering Information Technology (IT) services over Internet and emerging as a promising technology to offer services pay-as-you-go basis. The elastic property of cloud enables customer to scale up or down the resources based upon the requirements dynamically and reduces up-front IT investment significantly. Cloud computing promises to cut the operational cost and capital costs and more importantly IT departments focus on strategic projects instead of keeping the datacenter running. Cloud computing has received large scale as a rising access for distributed Information and Communication Technologies (ICT) services as a utility.
The mechanism of providing the services is necessary to improve the utilization of datacenter resources which are operating in most dynamic workload environments. Datacenters are the needed components of cloud computing, in a single datacenter generally hundreds and thousands of virtual servers run at any reason of time, hosting many tasks and at the same time the cloud system keeps receiving the batches of task requests. During this context, one has to notice few target servers out many powered on servers, which can perform a lot of incoming tasks. So, task scheduling is a valuable issue which is greatly influences the performance of cloud service provider.

**Objective**

Scheduling is the one of the most prominent activities that executes in the cloud computing environment. The main objective of the scheduling algorithms in cloud environment is to utilize the resources properly in order to get the minimum execution time. Seemingly, it has given the unlimited computing resources of the new type of computing system unfortunately there exist no polynomial time-scheduling algorithms to optimize the allocation of these computing resources because most scheduling problems are either NP-hard or NP-complete. At present, a simple example explained that there are being faced that is less than 0.02% of the candidate solutions between the makespan of the optimum solution and 1.01 times the makespan of the optimum solution.

The scheduling problem can usually be considered as a problem for the allocation of a set of given tasks, $T = \{T_1, T_2, \ldots, T_p\}$ to a set of given machines $M = \{M_1, M_2, \ldots, M_m\}$, subject to the constraints of optimizing one or more predefined measures or objective functions. When there is one and only one machine, i.e., $m = 1$, the scheduling problem is referred to as a single processor (single machine) scheduling problem. When there is more than one machine, i.e., $m \geq 2$, the scheduling problem is regarded as a multiprocessor parallel machine scheduling problem.

**Scope of the Project**

Scheduling is a complex task in the cloud computing. Due to uncertainty of cloud computing, several heuristic algorithms have been widely used. The project uses various heuristic techniques to find suitable scheduling in cloud. The Existing rule-based scheduling algorithm is not suitable for finding the suitable scheduling solution due to dynamic and uncertainty issues of the cloud. The solution in existing algorithm is far from optimal solutions. To get optimal solution, suitable scheduling algorithm is needed.

Heuristic Scheduling Optimization problems are in Class NP-hard. This issue can be solved by list method, heuristic method or approximation method. The list of method, an optimal solution can be selected if all the possible solutions are enumerated and related one by one. When number of reason is large, exhaustive enumeration is not suitable for scheduling trouble. In that case heuristic is a suboptimal algorithm to find reasonably good solutions logically fast. Approximation algorithms are used to find approximate solutions to optimized solution. These algorithms are used for trouble when exact polynomial time algorithms are known. Enlarge task data conditions in large scale data processing systems is crucial for the job achievement time. Most of the access to improve data locality are either greedy and ignore global
optimization, or suffer from high counting complexity.

**Related Research Work**

Lizheng et al. [1], proposed various problems that leads to complex applications which can be divided into two classes. The one was computing intensive, the other was data intensive. As far as the data intensive application, scheduling method should decrease the data movement which means decreases the transferring time but the computing intensive tasks of scheduling strategy should schedule the data to the high performance computer. In order to take cloud computing, scientific workflow would gain a more utilizations. It deal with a lot of new objection, of which data and task scheduling were one to efficient schedule all the tasks of an application was the most important problem. It focused on minimizing the total executing cost and transferring time. In order to scale down the executing time, applications schedule the computing intensive tasks to the high performance computer.

Shaobin et al. [2], explained in-depth based on principle of Particle Swarm Optimization (PSO) and improves the algorithm in resources scheduling strategy of the cloud computing. Through analysis, the outcome shows that this method could reduce the task average running time, and raise the rate availability of resources. Particle Swarm Optimization (PSO) is an adaptive searching algorithm placed on group. Because of its benefit of parallel distribution, scalability, easy to understand, secure robustness, with high resilience and robust in effective environments, Particle swarm optimization (PSO) solves frequent connective optimization problems successfully. Task scheduling problem could select a better one from various combinations distributed to task by resources. Particle Swarm Optimization is very sufficient to solve resource scheduling problem in cloud environment.

Rubing et al. [3], explained the scheduling problem of large-scale applications inspired from real-world, characterized by a huge number of homogeneous and concurrent tasks that were the main sources of bottlenecks but it had great potential for optimization. The scheduling problem is formulated as a new sequential cooperative and proposes a communication and storage-aware multi-objective algorithm that optimizes two user objectives (i.e.) execution time and economic cost, while fulfilling two constraints network bandwidth and storage requirements. During the time, comprehensive experiments using both simulation and real-world applications that demonstrate the efficiency and effectiveness in terms of algorithm complexity, makespan, cost, system-level efficiency, fairness, and other aspects compared with other related algorithms.

Hamid et al. [4], introduced pricing model and truthful mechanism for scheduling single tasks considering two objectives such as monetary cost and completion time. With respect to the social cost of the mechanism, i.e., minimizing the completion time and monetary cost extended the mechanism for dynamic scheduling of scientific workflows. The theoretically analyzed the truthfulness and the efficiency of the mechanism and extensive experimental results showing significant impact of the selfish behavior of the cloud providers on the efficiency of the whole system. The experiment conducted using real-world and synthetic workflow applications demonstrate that solutions could be dominated in most cases the Pareto-optimal solutions estimated by two classical multi-objective evolutionary algorithms.
SCHEDULING BASED HYPER-HEURISTIC ALGORITHM

Hyper-Heuristic Scheduling Algorithm (HHSA) uses to operators namely the improvement detection and diversity detection operators to balance the intensification and diversification in the search of solutions during the convergence process. One of the heuristic algorithms in the basin of successor will be selected by the Low-Level Heuristic (LLH) selection operator as the heuristic algorithm to be performed. The selected hyper-heuristic algorithm and Low-Level Heuristic (LLH) will then be performed repeatedly until completion precedent is face. The selected LLH will evolve the solution for iterations by using the determine function to balance the intensification and diversification of the search directions, which in turn rely on the information provided by the improvement detection operator and by the diversity detection operator to decide whether to select a new LLH or not.

A. HYPER-HEURISTIC SCHEDULING ALGORITHM

1. Determine up the parameters.
2. Input the scheduling problem.
3. Compute the population of solutions W={W_1, W_2, ..., W_N}.
4. Randomly select a heuristic algorithms H_i from the successor basin H_i.
5. While the completion precedent is not face.
6. Update the population of solution W by using the selected algorithm H_i.
7. D1 = Improvement_Detection(W).
9. If \( \psi(H_i, D_1, D_2) \)
10. Randomly select a new H_i.
11. W=Perturb(W).
12. End.
13. End.
14. Output the best so distant solution as the final solution.

Algorithm Description
The Hyper-Heuristic Scheduling algorithm is step 1 determine up the parameters \( \Omega_{\text{max}} \) and \( \Omega_{\text{ni}} \), where \( \Omega_{\text{max}} \) denotes the maximum number of iterations the selected low-level heuristic algorithm is to be run; \( \Omega_{\text{ni}} \) the number of iterations the solutions of the selected low-level heuristic algorithm are not improved. A Step 2 reads in the tasks and jobs to be scheduled, i.e., the problem. Step 3 compute the population of solutions W={w_1, w_2, ..., w_N}, where N is the population size. Step 4, a heuristic algorithm H_i is anyway selected from the successor basin H={H_1, H_2, ..., H_n}. As distant as the proposed described herein is concerned, the low-level heuristic successor basin consists of ant colony optimization, genetic algorithm, particle swarm optimization. The selected hyper-heuristic algorithm (LLH) H_i will then be performed repeatedly until the completion precedent is met, as shown in step 5-13. The selected LLH H_i will evolve the solution W for \( \Omega_{\text{max}} \) iterations by using the determine function \( \Omega(H_i, W_1, W_2) \), as defined in equation to balance the intensification and diversification of the search directions, which in turn rely on the information provided by the improvement detection operator denoted D_1 in step 7 and by the diversity detection operator denoted D_2 in step 8 to decide whether to select a new LLH or not. For single-solution-based heuristic algorithms (SSBHA), only D_1 is
used whereas for population-based heuristic algorithms (PBHA), both $D_1$ and $D_2$ are used. in equation 1.

$$\psi(H_i, D_1, D_2) = \left\{ \begin{array}{ll}
\text{false, if } H \not\sim S \text{ and } D_2 = \text{true,} \\
\text{false, if } H \not\sim P \text{ and } D_1 = \text{true and } D_2 = \text{true,} \\
\text{true, otherwise,} \end{array} \right. \quad \ldots(1)$$

Where $S$ denotes the set of SSBHAs; $P$ the set of PBHAs. When the determine function function $\Phi(H_i, F_1, F_2)$ returns a true, the proposed algorithm will anyway select a new heuristic algorithm $H_i$ and then return the solution $W$ to the disruption operator to fine-tune the result. Note that the single solution-based heuristic algorithms employ one and only one search direction at each iteration on the convergence process while the population-based heuristic algorithms employ multiple search directions at each iterations.

### The Improvement Detection Operator

A simple random method is used to select the low-level heuristic $H_i$ from the successor basin $H$. According to consideration, the perfect so distant makespan (BSFMK) for both Simulate Annealing (SA) and Genetic Algorithm (GA) could continue to improve the results at early iterations (e.g., less than 200 iterations), but it is crucial to improve the results at later iterations (e.g., after 800 iterations), especially when the search directions converge to a small number of directions. From these considerations, a simple approach to identify when to select a new Low Level Heuristic is as given in equation 2.

$$F_1 = \left\{ \begin{array}{ll}
\text{false, BSFMK is not improved after } \Phi_{ri} \text{ iterations,} \\
\text{true, otherwise,} \end{array} \right. \quad \ldots(2)$$

This approach only checks to see whether the solutions found by $H_i$ BSFMK, are being improved or not.

### THE DIVERSITY DETECTION OPERATOR

If the randomly selected Low Level Heuristic (LLH) is a Population Based Heuristic Algorithm (PBHA), the diversity detection operator is used by HHSA to decide “when” to change the low-level heuristic algorithm. The diversity of the compute solution will be used as a threshold. The diversity of the current solution is computed as the average of the task distances between individual solutions which are defined as follows: If a task in two different individuals is assigned to the same VM, the task distance is 0; otherwise, the task distance is 1, as given in equation 3.

$$D_2 = \left\{ \begin{array}{ll}
\text{true, } D(Z) > \Box \\
\text{false, otherwise,} \end{array} \right. \quad \ldots(3)$$

Where $\Box$ is defined as $\mu^1 - 3 \times \sigma^1$ where $\mu^1$ and $\sigma^1$ denotes appropriately, the average distance and the standard deviation of the distances of the initial solution.

### The Perturbation Operator

In addition to the perturbation operators of selected Low Level Heuristic (LLH) itself, the proposed algorithm will disrupt the solutions obtained by LLH before they are passed on to the newly selected LLH. The successor solutions created by the low-level heuristic can be perturbed by assigning a different temperature to each individual to balance the intensification and diversification of the search. The makespan of the tasks executed are calculated and the performance is evaluated.
WORKFLOW SCHEDULING ON CLOUDSIM

CloudSim is an extensible simulation toolkit or framework that enables modeling, simulation and experimentation of Cloud computing system and application providing environments. Cloudsim can be used to construct a datacenter with a set of virtual machines as the resource.

Table 1: Parameter Settings of the Five Cloud Scheduling Algorithms of Workflow Scheduling on Cloudsim

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA</td>
<td>Temperature t=10, Temperature reduce t_r=0.8</td>
</tr>
<tr>
<td>ACO</td>
<td>Pheromone updating fact p=0.1, Choosing probability q_0=0.97, Related influence weights =1</td>
</tr>
<tr>
<td>PSO</td>
<td>Inertia weight w=0.7</td>
</tr>
<tr>
<td>HHSA</td>
<td>Max iteration of low-level algorithm max=50, Non-improved iteration threshold m=6</td>
</tr>
</tbody>
</table>

SIMULATION RESULTS OF WORKFLOW SCHEDULING

To evaluate the performance of HHSA for the workflow scheduling problem, compare it with two traditional rule-based algorithms and heuristic algorithms, namely, max-min, simulated annealing, particle swarm optimization, and ant colony optimization results of heuristics (SA, PSO, and ACO) are better than max-min. Moreover, the results also show that HHSA outperforms all the other scheduling algorithms, namely, max-min, SA, PSO, and ACO.

Heuristic usually can find better results than traditional scheduling algorithms (i.e. max-min) in terms of makespan. In addition, the results of ACO are similar to those of the proposed algorithm for the first for the results of heuristics (SA, PSO, and ACO) are better than max-min. Moreover, the results also show that HHSA outperforms all the other scheduling algorithms, namely, max-min, SA, PSO, and ACO. The results of best-so-far makespan, min-min and max-min get almost exactly the same makespan in all iterations.
For the SA, the higher the temperature which will get a larger fine-tune rate for the current solution, the poorer the best-so-far makespan compared to the other non-traditional algorithms at the early iterations. For the SA, the higher the temperature which will get a larger fine-tune rate for the current solution, the poorer the best-so-far makespan compared to the other non-traditional results, the end results of HHSA are usually better than the other scheduling algorithms, especially for complex and large-scale problems. A closer look at these simulation results shows that the max-min algorithm does not scale well as the number of tasks increases. It can provide a result that is close to the other heuristic scheduling algorithms in terms of the makespan (more precisely, the differences are no more than 3). However, as the number of tasks increases, the results the differences between max-min other heuristic scheduling algorithms are also increased.

For instance, that the differences between max-min and other heuristic scheduling algorithms are more than five. The results search process of these scheduling algorithms will converge to a stable state very quickly when the data set is small. A good example is the results of ACO and HHSA, which show that the result at iteration 50 is very close to the result at iteration 100. But the results search process of ACO and HHSA can still find a better result after iteration 100, meaning that the heuristic scheduling algorithms still have a chance to find a better result at later iterations when the data set becomes large.

In addition, based on the observations of the convergence of these scheduling algorithms, it can easily know their performance. For small data sets, the convergence speeds of ACO and HHSA described are quite close to each other. However,

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Max-min</th>
<th>SA</th>
<th>PSO</th>
<th>ACO</th>
<th>HHSA</th>
</tr>
</thead>
<tbody>
<tr>
<td>e-Economic</td>
<td>1.51.71.7</td>
<td>2.12.65.16</td>
<td>2.02.62.7</td>
<td>2.52.62.7</td>
<td>2.62.62.6</td>
</tr>
<tr>
<td>e-Protein</td>
<td>3.63.63.5</td>
<td>2.83.82.2</td>
<td>2.73.82.8</td>
<td>3.83.82.9</td>
<td>3.83.83.8</td>
</tr>
</tbody>
</table>
as the number of tasks increases, the convergence speeds of ACO and HHSA. They show that HHSA has a higher chance to find a better result than ACO does because it keeps finding a better result after the ACO has reached a stable state.

ACKNOWLEDGMENT

First offer our deepest gratitude to the almighty who has given us the strength and good health during complete this paper. I express profound gratitude to our respected dean Dr. K.M MEHATA Ph.D., for his encouragement and guidance. I articulate our gratitude to our head of the department Dr. SHARMILA SANKAR M.E.,Ph.D., who had taught me the way to do successful paper. I gratitude to my respected coordinator Dr. ANGELINA GEETHA M.C.A.,M.E.,Ph.D., who had the novel ideas for this paper. And I wish to express and sincere thanks to our internal guide respected Dr. S. ARANGANATHAN M.E.,Ph.D., who had helped us lot and given valuable suggestions, which made us to finish this paper in very successful and neat way. Without whose constant encouragement and whole idea, the paper would not have been possible.

CONCLUSION

This survey presents a large enforce hyper-heuristic algorithm to find better scheduling solutions for cloud computing systems. The proposed algorithm need two detection operators to automatically resolve when to change the low-level heuristic algorithm and a perturbation operator to calibrate the solutions obtained by each low-level algorithm to further improve the scheduling results in terms of makespan. As the simulation results show, the proposed algorithm can not only provide better results than the traditional rule-based scheduling algorithms. In addition, the simulation results show further that the proposed algorithm concentrate faster than the other heuristic algorithms check out in this study for most of the data sets. In brief, the main idea of the proposed “hyper-heuristic” algorithm is to advantage the strengths of all the low-level algorithms while not increasing the computation time, by executing one and only one low-level algorithm at each iteration. This is basically different from the so-called hybrid heuristic algorithm, which runs more than one low-level algorithm each iteration, thus involving a much longer computation time.

In the future, target will be on finding more efficient detection operators and perturbation method to enhance the performance of the proposed algorithm. It can also attempt to implement HHSA to different analysis domains.

REFERENCES


