Particle Swarm Optimization Approach with Parameter-wise Hill-climbing Heuristic for Task Allocation of Workflow Applications on the Cloud

Simone A. Ludwig
Department of Computer Science
North Dakota State University
Fargo, ND, USA
simone.ludwig@ndsu.edu

***Abstract*—Cloud computing provides the computing infrastructure, platform, and software application services that are offered at low cost from remote data centers accessed over the internet. This so called "utility computing" is changing the future of organizations in which their internal servers are discarded in favor of applications accessible in the cloud. One of the challenges workflow applications face is the appropriate allocation of tasks due to the heterogeneous nature of the cloud resources. There are different approaches, which have been proposed in the past to address the NP-complete problem of task allocation. One such approach that successfully addressed the task allocation problem made use of Particle Swarm Optimization (PSO). This paper further improves the performance of PSO by combining PSO with a local search heuristic. In particular, PSO with a parameter-wise hill-climbing heuristic (PSO-HC) for the execution of computationally-intensive as well as I/O-intensive workflows is introduced. Experiments are conducted using Amazon's Elastic Compute Cloud as the experimental simulation platform looking at the scalability of CPU-intensive and I/O-intensive workflows in terms of cost and execution time.**

***Index Terms*—Workflow execution, task allocation, Amazon Elastic Cloud.**

I. Introduction

Cloud computing has received large amounts of attention in particular in the IT industry. Cloud computing provides the computing infrastructure, platforms, and software application services that are offered at low cost from remote data centers accessed over the internet. This so called "utility computing" is changing the future of organizations in which their internal servers are discarded in favor of applications accessible in the cloud. The benefits that the users envision from the cloud are scalability, reduction of costs of application management, as well as an reduction in overall hardware costs. Providers on the other hand offer the opportunity to leverage existing cluster infrastructure by taking advantage of economic prices when purchasing large volumes of hardware and network capacity. Given the anticipated cost savings from cloud adoption, more and more businesses are expected to migrate some of their applications from dedicated servers to private or public clouds.

Over the years, cloud computing has been attracting more and more attention for providing a flexible, on demand computing infrastructure for many applications [1]. Cloud computing can be utilized by facilitating workflow execution. A workflow enables to structure applications in a directed acyclic graph, where each node represents the task and edges represent inter-task dependencies of the applications [2]. A single workflow consists of a set of tasks, and each task might communicate with another task in the workflow, i.e., the results of one task might be the input for another task.

Cloud computing is very suitable as the execution environment for workflows [3][4]. Solutions to schedule and execute workflows in the cloud have been proposed recently. In [5], an integrated data placement and task assignment algorithm for scientific workflows in clouds is presented. Another data placement strategy for scientific cloud workflows is described in [6]. Both research works address the problem of task allocation that is known to be NP-complete [7]. In traditional task assignment, the algorithms addressed the issue from the provider perspective, whereby the focus was to maximize the utilization of homogeneous resources in a grid or cluster environment for multiple workflow requests. However, in cloud computing, we have virtual clusters that consist mostly of heterogeneous resources and therefore, the focus lies on the user who wants to minimize costs and reduce the runtime of workflow applications. This change in focus directs to solutions of different task allocation formulation since different resources, both software and hardware, and different utility functions can be derived [8]. Different utility functions are performance, cost, profit, energy consumption, etc.

This paper introduces a Particle Swarm Optimization (PSO) approach applied to the task allocation problem for workflow applications running on the cloud. Past research has found that PSO performs very well on the task allocation problem, however, we are further improving the algorithm by combining PSO with a local search heuristic. It is a well-known fact in the evolutionary computation and swarm intelligence community, that hybrid approaches, whereby a metaheuristic algorithm is combined with a local heuristic method, achieve improved performance and show quicker convergence speed. Therefore, we introduce a discrete PSO approach that makes use of

a parameter-wise hill-climbing heuristic (PSO-HC). The improvement of the parameter-wise hill-climbing heuristic works by locally scanning and further updating the quality of the potential solutions of each particle within the PSO iteration.

The paper is structured as follows. Section II describes related work in the area of task allocation. In Section III, the problem is formulated. Section IV describes the proposed algorithm with all its features. In Section V, the experimental setup as well as the results obtained are outlined, and Section VI contains the conclusions of this study.

II. Related Work

Research in the area of workflow task assignment or scheduling algorithms can be summarized as follows. A compromised-time cost scheduling algorithm is presented in [9]. The algorithm considers the characteristics of cloud computing in order to accommodate instance-intensive cost-constrained workflows. This is achieved by balancing execution time and cost with user input.

In [10], an improved cost-based algorithm to improve the computation and communication rate in cloud computing is proposed. This algorithm achieves an efficient mapping of tasks through measurement of resource cost and computation performance.

A Scalable-Heterogeneous-Earliest-Finish-Time Algorithm (SHEFT) [11] for elastically workflow scheduling in Cloud computing is proposed. The experimental results show that SHEFT outperforms several representative workflow scheduling algorithms focusing on optimizing workflow execution time.

The authors in [12] developed a multiple QoS constrained scheduling strategy. The strategy is implemented for multiple workflow management system with multiple QoS. Their approach minimizes the makespan and cost of the execution of workflows on a cloud computing platform.

An ant colony optimization algorithm to schedule large-scale workflows with various QoS parameters is presented in [13]. The authors' algorithm enables the user to specify QoS preferences by applying minimum QoS thresholds for a certain application. The objective of the algorithm is to find a solution that meets all QoS constraints as well as to optimize the user-preferred QoS parameter.

In [14], an optimized scheduling algorithm is introduced. The algorithm is an improved Genetic Algorithm (GA) that uses an automated scheduling policy by increasing the utilization rate of resources as well as speed.

In [15], a PSO based heuristic to schedule applications to cloud resources is introduced. The heuristic takes into account both computation cost and data transmission cost. Experiments are conducted varying the computation and communication costs of workflow applications. The experimental results show that PSO can achieve cost savings by choosing a good distribution of workload onto resources.

A discrete PSO algorithm is proposed in [16] for scheduling applications among cloud services taking both data transmission cost and computation cost into account. Experiments are conducted with a set of workflow applications by varying their data communication costs and computation costs according to a cloud price model. Measures compared are makespan, cost optimization ratio, and the cost savings comparing the algorithm with standard PSO and the Best Resource Selection algorithm.

Our proposed discrete PSO approach further improves the performance of the task allocation in a cloud environment. It makes use of the combination of PSO with a local search heuristic. In particular, PSO is combined with a parameter-wise hill-climbing heuristic to improve the cost and execution time of CPU-intensive and I/O-intensive workflow applications at the same time achieving faster convergence speeds.

III. Problem Formulation

A cloud workflow application can be modeled as a Directed Acyclic Graph (DAG) and can be formally described by $G = (V, E)$, where $V = \{T_1, ..., T_n\}$ is the set of tasks of the workflow, and E represents the data dependencies between these tasks. An arc is described by $d_{i,j} = (T_i, T_j, W_{ij}) | W_{ij} \in \Omega$, and $d_{i,j}$ is the data produced by T_i and consumed by T_j. Set Ω contains the file size in gigabytes, that are transferred between the tasks in the workflow. W_{ij} represents the total size of the files produced by task T_i and consumed by its child T_j. The task at hand is to assign n workflow tasks to m cloud instances. We assume that a child task cannot be executed until all of its parent tasks have been completed.

Figure 1 shows a workflow application with 9 tasks. These 9 tasks can be implemented by four cloud instances (computation and storage instances) that are fully connected and symmetric, as shown in Figure 2.

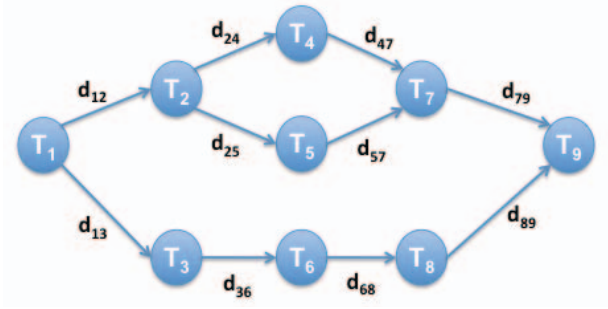

Fig. 1. Example workflow application

The two measures that we investigate are execution time and cost. The execution time of a workflow application for a specific allocation a is given as the maximum completion time of the execution of all tasks running on each cloud instance under the assumption that all input and output files are present, plus the transfer overhead that is required to transfer the necessary files from one cloud instance to another. Therefore, we formally denote:

$$time_{exec}(a) = max_k(time_{end}(T_{ik})) + \frac{overhead(a)}{bandwidth} \quad (1)$$

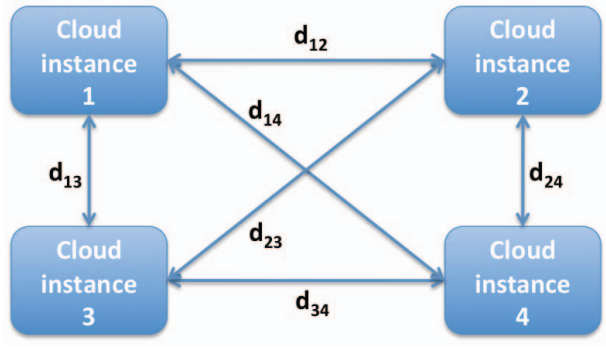

Fig. 2. Allocated cloud instances

where $bandwidth$ is given as per Amazon EC2 and S3 specifications outlined in Tables I and II. The maximum completion time, $time_{end}$, of the tasks that are executed per cloud instance is calculated, whereby each task of the workflow application varies depending on the workload of the system. When all tasks are executed, the maximum completion time, $time_{end}$, of all cloud instances are considered. The overhead portion is given as:

$$overhead(a) = \sum O_i + \sum O_{ij} \quad (2)$$

where O_i represents all the output files of task T_i, and $\exists k, l, k \neq l$ such that $T_i \in a_k$ and $T_j \in a_l$, and $(T_i, T_j, O_{ij}) \in W$.

The cost of executing a workflow application for a specific allocation a is calculated as the cost of running all cloud instances, $cost_{run}$, plus the cost of communication when using the Amazon S3 storage model, $cost_{S3}$.

$$cost(a) = cost_{run}(a) + cost_{S3}(a) \quad (3)$$

and the $cost_{S3}$ portion is calculated as:

$$cost_{S3}(a) = \sum_{ij} n_i * cost_{put} + m_j * cost_{get} \quad (4)$$

where $\exists k, l, k \neq l$ such that $T_i \in a_k$ and $T_j \in a_l$, and $(T_i, T_j, O_{ij}) \in W$. Whenever a parent task i and its children tasks j are not allocated on the same cloud instance, the parent task will upload its output files to the Amazon's S3 storage service using buckets of 5GB each [17] using n_i PUT requests that are priced at $cost_{put}$. Similarly, all children tasks that are not allocated on the same cloud instance need to download the required files from Amazon S3 using m_j GET requests that are priced at $cost_{get}$. The prices for $cost_{put}$ and $cost_{get}$ are given in Table II.

TABLE I
SPECIFICATION OF AN AMAZON CLOUD INSTANCE

Parameter	Value
Type	Infrastructure-as-a-Service (IaaS)
CPU	c1.xlarge
Memory	7GB
Price	0.66$/hr

TABLE II
SPECIFICATION OF AN AMAZON S3 INSTANCE

Parameter	Value
Object size	5GB
Price of PUT request	0.01$
Price of GET request	0.001$
Bandwidth	12MB/s

IV. PROPOSED APPROACH

Since our approach is based on PSO, we first describe standard PSO and then explain our improved PSO version that makes use of a local search heuristic.

A. Standard PSO

The original PSO algorithm was inspired by the social behavior of biological systems [18]. The algorithm is used to search through an n-dimensional search space. First, n particles are randomly initialized within the search space. Each particle is represented by an n-dimensional vector $X_i(i = 1, ..., d)$ representing its location $(x_{i1}, x_{i2}, ..., x_{in})$ that also represent potential solutions to the problem. The personal best position of each particle is denotes as p_i, and the position of the best individual of the entire swarm is denoted as the global best position p_g. The following two equations update the velocity and its new position for each particle in the swarm:

$$v_{id}^{n+1} = w * v_{id}^n + c_1 * r_1(p_{id}^n - x_{id}^n) + c_2 * r_2(p_{gd}^n - x_{id}^n)) \quad (5)$$

$$x_{id}^{n+1} = x_{id}^n + v_{id}^n \quad (6)$$

where $d = 1, 2, ..., D$; $i = 1, 2, ..., N$; $n = 1, 2, ..., iter_{max}$; w is the inertial weight; c_1 and c_2 are two positive constants called acceleration coefficients; r_1 and r_2 are two random values uniformly distributed within the interval $[0, 1]$.

B. PSO with Parameter-wise Hill-climbing Search Heuristic (PSO-HC)

As can be seen from Equations 5 and 6, PSO was initially introduced for continuous optimization problems. However, the task allocation problem is discrete, and therefore, the "update equations" have to be defined. Our task allocation problem can be defined by a set of pairs $M = [< T_i, I_j >](i \in [1, m], j \in [1, n])$, where m is the number of tasks to be executed, and n is the number of cloud instances available in the cloud environment. We define a position to be a feasible solution to the task allocation problem consisting of a set of $< task, instance >$ pairs. Each pair describes a mapping of a task to a cloud instance, which indicates that the position of each particle satisfies the precedence constraint between the tasks. The *velocity* is defined as a set of possible allocations $V = \{e/p(e)|e \in E\}$, E is the set of $< task, instance >$ pairs. $p(e) \in [0, 1]$ shows the possible mapping of a task to an instance. *Subtraction* between two particle positions x_1 and x_2 is defined as a set of pairs that exist in x_1 but not in x_2. This operator is formally known as the relative complement or set theoretic difference between two sets. *Multiplication*

of the random real numbers with the velocity is defined as $cV = \{e/p'(e)|e \in E\}$, whereby

$$p(e) = \begin{cases} 1 & \text{if } c * p(e) > 1 \\ c * p'(e) & \text{otherwise.} \end{cases}$$

Addition of two velocities is defined as choosing the larger of the two: $V_1 + V_2 = \{e/max(p_1(e), p_2(e))|e \in E\}$. The new position of a particle is defined as the steps show above. The constraints between tasks must be taken into account. The $< task, instance >$ pairs of the new X_i^j can be chosen from the sets of the global best, personal best, previous position, or other combinations of feasible pairs.

Since it has been shown in past research endeavors that a hybrid strategy that combines a metaheuristic algorithm with a local heuristic can obtain better results and converges more quickly [19], we have devised a parameter-wise hill-climbing heuristic in order to incorporate this heuristic into our proposed PSO algorithm. The proposed heuristic works as follows: Given a particle vector, its solution quality can be improved locally by scanning the particle elements for updating. Since in our particle representation each particle consists of m elements and the value of each element ranges from 1 to n, we sequentially scan each element and determine a possible replacement with another value to improve a particle's fitness. When an element is examined, the values of the remaining $m-1$ elements remain unaffected. The heuristic is terminated once all the elements of the particle have been examined. The computation for the fitness calculation due to the element updating can be minimized since a value change in one element affects the allocation of another, and therefore, the scanning of the other element does not need to be performed anymore.

Algorithm 1 describes the PSO-HC algorithm step-by-step. The algorithm starts by randomly initializing the particles in the swarm making sure that each particle represents a feasible solution. Then, the personal best position and the global best position are computed. The algorithm then iterates within a while loop until a stopping criterion is met, and calculates the particle's new position with the following three steps. First, elements from the promising set of pairs with larger probability are chosen according to the global best position and personal best position. Second, due to the discrete nature of the task allocation problem, there are usually not enough feasible pairs in global best position vector to generate a new position, and therefore a particle will learn from its previous position. And third, all unmapped tasks will select resources from other feasible pairs. Finally, the global best position is returned as the best solution.

Algorithm 1 PSO algorithm description

```
initialize the swarm of particles
calculate personal best position and global best position
while stopping criterion is not met do
    calculate V_p and V_g
    construct new position x_{i+1}^j
    select pairs from P(V_i^j) = {e|e/p(e) ∈ V_i^j}
    if x_{i+1}^j is completed then select from x_i^j
    else if x_{i+1}^j not completed then select from feasible pairs
    end if
    calculate fitness value
    update personal best position and global best position
    improve solution using hill-climbing heuristic
end while
return global best position
```

V. Experiments and Results

To evaluate the efficiency and effectiveness of the proposed algorithm on the task allocation on the cloud, experiments have been conducted. A large simulation dataset was created that features different problem sizes. The proposed algorithm (PSO-HC) is compared to the basic PSO. All of the experiments were conducted on an Intel Core 2 Duo (2.4GHz, 3MB L2 cache) running the Java version 1.6.2 JDK runtime environment. Average results are shown that are taken from 30 independent runs given the stochastic nature of PSO. Analysis of mean and standard deviation with respect to cost and execution time for CPU-intensive and I/O-intensive workflow applications are presented. In addition, in order to find out whether our results are statistically significant, a paired T-test was run on all results. A significance level of 5% was chosen, and the p-values are displayed in the corresponding tables.

Table III lists the values measured for increasing workflow sizes of an I/O-intensive workflow application. Figure 3 shows the results graphically. As can be seen, PSO-HC outperforms PSO for all 8 workflow sizes. All of the improvements are significant resulting in a p-value of less than 0.05.

TABLE III
Mean and standard deviation of different I/O-intensive workflow sizes with respect to cost (US $)

Workflow size	Cloud instances	PSO-HC	PSO	p-value
25	4	0.843±0.183	0.924±0.194	<0.05
50	6	1.346±0.207	1.406±0.264	<0.05
75	8	2.152±0.192	2.268±0.199	<0.05
100	10	2.963±0.225	3.207±0.237	<0.05
125	12	3.894±0.216	4.461±0.219	<0.05
150	14	5.746±0.249	6.147±0.251	<0.05
175	16	7.591±0.204	7.846±0.216	<0.05
200	18	8.546±0.217	9.763±0.267	<0.05

Table IV and Figure 4 show the results of increasing workflow sizes on the optimization of the execution time. As can be seen, the execution time of the I/O-intensive workflow application when PSO-HC is applied is smaller (and therefore better) than basic PSO for all task sizes measured. The improvements between PSO-HC and PSO become larger with larger workflow sizes.

Figure 5 and Table V display the results of the optimization of the CPU-intensive workflow application. PSO-HC returns optimization values with regard to cost of $0.836 and $6.946 for 25 and 200 tasks, respectively. PSO on the other hand

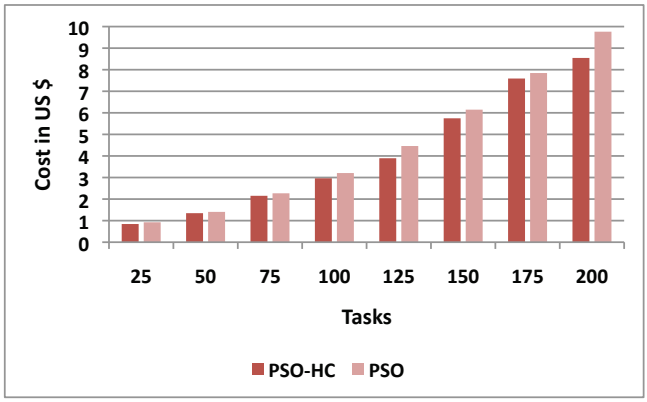

Fig. 3. I/O-intensive workflow sizes with respect to cost (US $) for increasing task sizes

TABLE IV
MEAN AND STANDARD DEVIATION OF DIFFERENT I/O-INTENSIVE WORKFLOW SIZES WITH RESPECT TO EXECUTION TIME (HOURS)

Workflow size	Cloud instances	PSO-HC	PSO	p-value
25	4	0.591±0.034	0.649±0.039	<0.05
50	6	1.104±0.316	1.168±0.324	<0.05
75	8	1.649±0.647	1.784±0.637	<0.05
100	10	1.964±0.783	2.178±0.811	<0.05
125	12	3.076±0.979	3.569±0.994	<0.05
150	14	4.706±0.894	5.209±0.996	<0.05
175	16	6.057±0.863	6.776±0.695	<0.05
200	18	7.549±1.167	7.994±0.778	<0.05

measures larger costs with $0.881 and $7.89 for 25 and 200 tasks, respectively. This again demonstrates that PSO-HC achieves better cost values for any number of tasks the workflow application consists of.

Figure 6 and Table VI show the results for increasing task sizes measuring the execution time in hours. The results are shown for the CPU-intensive workflow application. PSO-HC measures an execution time of 0.558 hours, and PSO takes 0.674 hours to run for a workflow consisting of 25 tasks. For

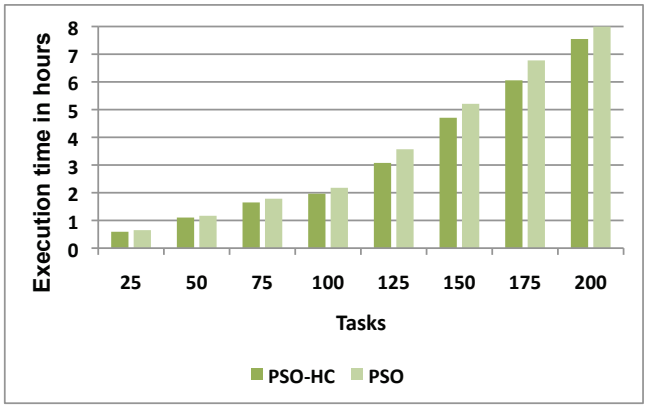

Fig. 4. I/O-intensive workflow sizes with respect to execution time (hours) for increasing task sizes

TABLE V
MEAN AND STANDARD DEVIATION OF DIFFERENT CPU-INTENSIVE WORKFLOW SIZES WITH RESPECT TO COST (US $)

Workflow size	Cloud instances	PSO-HC	PSO	p-value
25	4	0.836±0.012	0.881±0.009	<0.05
50	6	1.169±0.014	1.234±0.005	<0.05
75	8	1.387±0.009	1.543±0.014	<0.05
100	10	2.094±0.016	2.408±0.018	<0.05
125	12	3.417±0.021	3.517±0.019	<0.05
150	14	4.267±0.019	5.046±0.031	<0.05
175	16	5.974±0.024	6.513±0.041	<0.05
200	18	6.946±0.043	7.890±0.037	<0.05

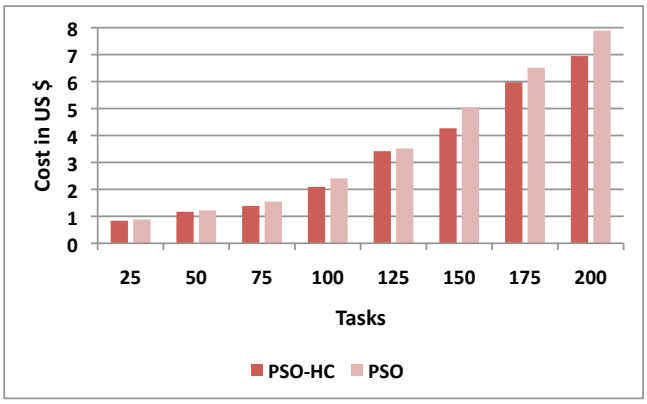

Fig. 5. CPU-intensive workflow sizes with respect to cost (US $) for increasing task sizes

200 tasks, PSO-HC optimizes the execution time achieving a value of 10.578 hours, whereas PSO obtains a value of 11.849 hours. This demonstrates that the improvement for larger task sizes are greater than for smaller task sizes.

TABLE VI
MEAN AND STANDARD DEVIATION OF DIFFERENT CPU-INTENSIVE WORKFLOW SIZES WITH RESPECT TO EXECUTION TIME (HOURS)

Workflow size	Cloud instances	PSO-HC	PSO	p-value
25	4	0.558±0.033	0.674±0.031	<0.05
50	6	0.946±0.027	1.067±0.037	<0.05
75	8	1.899±0.041	2.194±0.062	<0.05
100	10	3.067±0.067	3.749±0.073	<0.05
125	12	5.749±0.055	5.946±0.079	<0.05
150	14	6.543±0.073	7.149±0.086	<0.05
175	16	8.214±0.085	9.287±0.094	<0.05
200	18	10.578±0.091	11.849±0.087	<0.05

VI. CONCLUSION

This paper introduced a PSO approach with a parameter-wise hill-climbing heuristic (PSO-HC) for the investigation of CPU-intensive as well as I/O-intensive workflow applications optimizing cost and execution time. Experiments were conducted using Amazon's Elastic Compute Cloud as the input data for the simulation experiments. Given that PSO has been successfully applied to the task allocation problem before, a further improvement to the performance of the optimization is

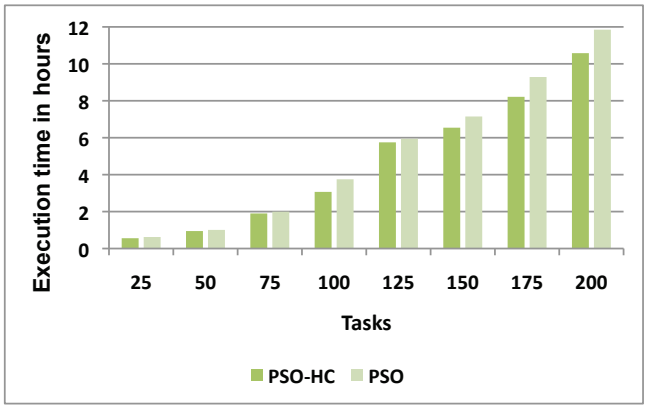

Fig. 6. CPU-intensive workflow sizes with respect to execution time (hours) for increasing tasks sizes

done using a local search technique referred to as a parameter-wise hill-climbing heuristic. Furthermore, in past literature the concern was mostly the execution time of workflows. However, we have considered two optimization measures, execution time and cost. A real-world simulation study was conducted using Amazon's Elastic Compute Cloud resource data.

The experiments were run to measure the scalability of CPU- and I/O-intensive workflow applications in terms of cost and execution time. PSO-HC was compared to basic PSO, since basic PSO has been used by several researchers in the past.

The experimental results show that PSO-HC outperformed PSO for both CPU-intensive as well as I/O-intensive workflow applications. Increasing number of tasks were simulated and all results measuring cost and execution time resulted in improved values. The improvements are in particular noticeable for larger numbers of tasks within a workflow application. All improvements achieved by PSO-HC were significant resulting in a p-value of less than 0.05.

Future work will include the optimization of other parameters such as reliability and trust which are also very important in the area of cloud computing. It would also be interesting to implement a multi-objective PSO algorithm in order to see how well it would perform compared to the single-objective PSO-HC and PSO algorithms. Furthermore, given that the additional hill-climbing heuristic of PSO-HC increases the complexity of the algorithm, and therefore has a longer running time as compared to the basic PSO, this also needs to be taken into consideration. A possible solution to speed up the optimization task would be to employ a distributed computing concept such as MapReduce [20].

REFERENCES

[1] E. Deelman, "Mapping abstract complex workflows onto Grid environments, Journal of Grid Computing 2003; 1:2539.

[2] J. Yu and R. Buyya, "Workflow Scheduling Algorithms for Grid Computing, Technical Report, GRIDS-TR-2007-10, Grid Computing and Distributed Systems Laboratory, The University of Melbourne, Australia, May 2007.

[3] C. Evangelinos and C. N. Hill. "Cloud Computing for Parallel Scientific HPC Applications: Feasibility of running Coupled Atmosphere-Ocean Climate Models on Amazons EC2". In Cloud Computing and its Applications, 2008.

[4] N. Yigitbasi, A. Iosup, and D. Epema. "C-Meter: A Framework for Performance Analysis of Computing Clouds". In Cluster Computing and the Grid, pages 472477, 2009.

[5] U. V. Catalyuek, K. Kaya, and B. Ucar. "Integrated Data Placement and Task Assignment for Scientific Workflows in Clouds". In Proceedings of the Fourth International Workshop on Data Intensive Distributed Computing, 2009.

[6] D. Yuan, Y. Yang, X. Liu, and J. Chen. "A Data Placement Strategy in Scientific Cloud Workflows". Future Generation Computing Systems, 26:12001214, 2010.

[7] R. Cohen, L. Katzir, and D. Raz, "An Efficient Approximation for the Generalized Assignment Problem", Information Processing Letters, Vol. 100, Issue 4, pp. 162-166, November 2006.

[8] Cloud Computing Synopsis and Recommendations. Available from http://csrc.nist.gov/publications/drafts/800-146/Draft-NIST-SP800-146.pdf, last retrieved September 2012.

[9] K. Liu, Y. Yang, J. Chen, X. Liu, D. Yuan and H. Jin, "A Compromised-Time- Cost Scheduling Algorithm in SwinDeW-C for Instance-intensive Cost-Constrained Workflows on Cloud Computing Platform, International Journal of High Performance Computing Applications, vol. 24 no. 4, pp. 445-456, 2010.

[10] Y. Yang, K. Liu, J. Chen, X. Liu, D. Yuan and H. Jin, "An Algorithm in SwinDeW-C for Scheduling Transaction-Intensive Cost-Constrained Cloud Workflows", 4th IEEE International Conference on e-Science, 374-375, Indianapolis, USA, December 2008.

[11] C. Lin, S. Lu, "Scheduling Scientific Workflows Elastically for Cloud Computing, Proceedings of IEEE 4th International Conference on Cloud Computing, 2011.

[12] M. Xu, Li. Cui, H. Wang, Y. Bi, "A Multiple QoS Constrained Scheduling Strategy of Multiple Workflows for Cloud Computing, Proceedings of 2009 IEEE International Symposium on Parallel and Distributed Processing, 2009.

[13] W. Chen, J. Zhang, "An Ant Colony Optimization Approach to a Grid Workflow Scheduling Problem With Various QoS Requirements, IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews, Vol. 39, No. 1, January 2009.

[14] H. Zhong, K. Tao, X. Zhang, "An Approach to Optimized Resource Scheduling Algorithm for Open-source Cloud Systems, Proceedings of Fifth Annual China Grid Conference, 2010.

[15] S. Pandey, L. Wu, S.M. Guru, R. Buyya, "A Particle Swarm Optimization-based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments, Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications, IEEE, Perth, Australia, pp. 400-407, 2010.

[16] Z. Wu, Z. Ni, L. Gu, X. Liu, "A Revised Discrete Particle Swarm Optimization for Cloud Workflow Scheduling", Proceedings of the 2010 International Conference on Computational Intelligence and Security, 2010.

[17] Amazon Simple Storage Service (Amazon S3), access from http://aws.amazon.com/s3/, last retrieved September 2012.

[18] J. Kennedy and R. Eberhart, "Particle swarm optimization". Proceedings of IEEE International Conference on Neural Networks, 4:1942–1948, 1995.

[19] Z. Michalewicz, D.B. Fogel, "How to Solve it: Modern Heuristics", Springer-Verlag, 2002.

[20] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters", Proceedings of the 6th conference on Symposium on Operating Systems Design & Implementation (OSDI'04), Berkeley, CA, USA.