A Cost-Aware Strategy for Deadline Constrained Scientific Workflows

S Manam, K Moessner, S Vural

1,2,3Institute for Communication Systems, 5GIC, University of Surrey, Guildford, UK
E-mail: s.manam@surrey.ac.uk, k.moessner@surrey.ac.uk, s.vural@surrey.ac.uk

Abstract. As the reliability on clouds as an effective platform for scientific workflow applications increase, cloud computing structures are gradually becoming heterogeneous. In service-oriented systems that are heterogeneous, the management of the reliability of resources is a critical issue. Due to this heterogeneity, different models and bandwidths, existing workflows mainly focus on traditional computing environments such as grids. This paper proposes a deadline constrained scheduling strategy for scientific workflows across multiple clouds. To reduce the execution cost and meet deadline at the same time, an adaptive particle swarm optimisation (PSO) technique is proposed. This strategy uses a random single point mutation operator and a two-point crossover operator based on genetic algorithm (GA) technique to optimise computation and data transfer cost. The proposed scheme is evaluated using well-studied scientific workflows. The results obtained from simulations show that the adaptive PSO strategy performs better than existing state-of-the-art scheduling workflow strategies.

1. Introduction

In distributed and parallel systems, task scheduling has been described as a major challenge [1]. With hundreds of interrelated tasks, cloud computing applications are usually computational and data intensive [2]. These interrelated tasks can be effectively represented using workflow models which enable data to be analysed in a structured and distributed manner. In traditional environments, the study of workflow scheduling has been quite intensive with much attention in clustering and grid [3], [4].

Scientific Workflows normally vary in size and sometimes with limited resources to run complex task that require thousands of processing hours and bandwidth resources [5]. These type of workflows require high performance computing environments. More often, developers of these scientific workflows use local workstations such as super computers, grid platforms and clusters to run these workflows [6]. Several works mention in extensive surveys [7], [8], [9] have investigated the use of cloud computing for scientific workflows and these authors agree on the advantages offered by cloud computing in
terms of performance and cost. However, due to data dependencies, on-demand services, computational and communication cost etc., there is need to develop robust scheduling algorithms to meet the need of users.

With the emergence of numerous cloud service providers such as Microsoft Azure, Amazon E2, RackSpace, GoGrid [3] etc., cloud computing has been widely adopted for large scale computing. As pointed out in extensive surveys [10], [11], [12], [13], several authors have addressed workflow scheduling in cloud environments. These works focus on scheduling with uniform weights with an arbitrary number of processors. These works fail to address the fundamental principles of cloud computing such as heterogeneity and elasticity thereby making the scheduling of large workflows difficult.

One major contribution to workflow scheduling in a single cloud proposed by the authors in [14] uses a Particle Swarm Optimisation PSO-based strategy to reduce the overall cost of execution of a workflow and also meet the deadline constraints. However, the authors did not consider cost scheduling for deadline constrained workflow. Motivated by this strategy which is based on PSO algorithm, we propose an adaptive PSO for scheduling scientific workflows across multiple clouds. We take into consideration the characteristics of multiple clouds such as inter-bandwidth between different providers, shutdown time of virtual machines (VMs) as well as the cost of data transfer among providers. The proposed algorithm minimises the execution cost of workflow and at the same time meets deadline constrains across multiple clouds.

The rest of the paper is organised as follows; Some related works are presented in Section II. Section III describes our proposed model while section IV compares our proposed scheme with other state of the art approaches. We conclude the paper in Section V and make some remarks for future work.

2. Related Work

Several strategies [9], [15], [16], [17], [18] have been proposed for scheduling workflows in cloud computing applications. In the section, we discuss the strategies that have been proposed for workflow scheduling.

One of the popular approaches described in literature is the static and dynamic strategies for both task scheduling and resource provisioning [19]. The proposed algorithms focus on the QoS constraints of deadline and budget by maximising the number of executed workflows. One main assumption in this strategy is the variation of task execution time which may vary based on uniform distribution. The paper also considers only single VMs ignoring the heterogeneity in the IaaS clouds.

In [20], the authors propose a dynamic approach for scheduling workflows with minimum cost. In the proposed scheme, the different types of VMs which are available at different prices are considered and leased dynamically based on demand. To minimise the execution time of the workflow ensemble, they propose two approaches; task bundling and instance consolidation which are based on a set of heuristics task. The proposed strategy does not consider the time data is transferred between task which is
very important as it affects both cost and performance in scientific workflows.

In [21], the authors presented a cloud partial critical path algorithm to reduce the execution cost in deadline constrained resource scheduling. Path critical path (PCP) which is associated with the workflow’s exit node must be found by the algorithm. The same VM is used for scheduling of each task which may have been assigned to an already leased instance capable of meeting the requirements of the tasks. These tasks maybe assigned to the cheapest VM that is newly leased in event where these tasks cannot be completed. This process is repeated until all workflow have been scheduled. The authors extend the functionality (IC-PCP) with the same scheduling objectives with different placement of individual task while ICPCP assigns all tasks in a path to the same VM; IC-PCPD2 places individual task on the least expensive VM that can complete the task on time. The authors claim that IC-PCP2 performs better than IC-PCP performs better than IC-PCP2 in most cases based on data transfer time because (IC-PCP) reduces VM to VM communication by scheduling parent and child tasks on the same VM. However, there is no accountability for VM delays or the variation in resource performance. Again, the proposed algorithm may be highly sensitive to performance degradation of the CPU.

In [22], the authors propose a revised discrete PSO (RDPSO) to schedule applications among cloud services. RDPSO focuses on resource scheduling for single workflows and assumes the number of available VMs without taking into consideration the elasticity of resource provisioning. Experimental results show that the proposed scheme reduces the makespan and satisfies deadline constraints at the same time. Similar algorithm is proposed in [23] which is also based on PSO. The proposed algorithm in this approach is based on heuristics, interestingly the authors addressed the computational and data transmission cost and claim that their algorithm enables good distribution of workload. Closely related to our proposed scheme, the authors in [14] an algorithm which is also based on PSO to minimise the execution cost of workflows. This approach takes into consideration the fundamental cloud characteristics regarding the heterogeneity of VMs and their price model. They use a global optimisation technique to reduce the execution cost of workflows. However, the proposed algorithm does not address the difference in bandwidth and the cost of data transfer during workflow scheduling.

3. Proposed Scheme

In the proposed scheme, we consider three main components are considered; deadline constrained workflow, price driven and multi cloud environment. Similar to the approach in [16], the workflow \( W = (T, A) \) is modelled as a directed acyclic graph (DAG) where \( T = t_1, t_2, ..., t_n \) represents a finite set of \( n \) tasks and \( A = a_{ij} = (t_i, t_j) | t_i, t_j \in T \) represents the directed arcs. For each directed arc, there is a data dependency between task \( t_i \) and \( t_j \) which implies that \( t_j \) is executed upon completion of \( t_i \). For \( a_{ij} = (t_i, t_j) \), \( t_i \) is the parent task of \( t_j \) as shown in their workflow in Figure 1.

We represent the cloud providers \( C = c_1, c_2, ..., c_n \) with each service provider offering different types of VMs. For each task, the execution time is computed as \( T_{exe}(t_i, vm_{ijk}) \)
where \( vm_{ijk} \) is the \( k^{th} \) launched VM with \( j^{th} \) type offered by cloud provider across multiple clouds. Based on the VM memory consumption obtained from Amazon EC2 [24], it is assumed that each VM has enough memory to process the workflow. This means that only the processing capacity of the VMs are considered. Similar to the approach in [25], the bandwidth of a single VM is assumed to be infinite. The intra-data transfer \( DT_{\text{intra}}(a_{ij}, c_1) \) and inter-data transfer rate \( DT_{\text{inter}}(a_{ij}, c_1, c_2) \) are computed as shown below.

\[
DT_{\text{intra}}(a_{ij}, c_1) = \frac{Data(a_{ij})}{B_{\text{intra}}(c_1)} \tag{1}
\]

\[
DT_{\text{inter}}(a_{ij}, c_1, c_2) = \frac{Data(a_{ij})}{B_{\text{intra}}(c_1, c_2)} \tag{2}
\]

The intra-data transfer time \( DT_{\text{intra}}(a_{ij}, c_1) \) between task \( i \) and \( j \) in cloud provider \( c_1 \) is computed as shown in Eqn. 1. The inter-data transfer time \( DT_{\text{inter}}(a_{ij}, c_1, c_2) \) between cloud providers \( c_1 \) and \( c_2 \) is described in Eqn. 2. \( Data(a_{ij}) \) is the amount of data transferred between task \( i \) and \( j \). \( B_{\text{intra}}(c_1) \) is the intra bandwidth in provider \( c_1 \) and \( B_{\text{inter}}(c_1, c_2) \) is the bandwidth in provider \( c_1 \) and \( c_2 \) respectively.

3.1. Particle Swarm Optimisation PSO

In this section, we first introduce the basic PSO which was proposed by the authors in [26]. PSO is an evolutionary computation algorithm. The most important component of the PSO is the particle which represents the candidate solution that can move through...
the optimisation problem space. Each particle has its own velocity and position which is used to determine each particle as described in Eqn. 3.

\[ V_{i}^{t+1} = \omega \times V_{i}^{t} + b_{1}n_{1}(pBest_{i}^{t} - X_{i}^{t}) + b_{2}n_{2}(gBest_{i}^{t} - X_{i}^{t}) \]  

(3)

\[ X_{i}^{t+1} = X_{i}^{t} + V_{i}^{t+1} \]  

(4)

where \( i \) is the current iteration of evolutionary population, \( V_{i}^{t} \) and \( X_{i}^{t} \) represent the velocity and the position of the \( i \)th particle at the \( t \) iteration. Generally, \( V^{max} \) which is the maximum velocity is defined to ensure that the particles moving space is also in the range of the solution space. The weight \( \omega \) is used to determine how the previous velocity affects the current velocity and the acceleration coefficients \( b_{1} \) and \( b_{2} \) represent the particles cognitive ability. Random numbers \( n_{1} \) and \( n_{2} \) are used to enhance the randomness of the searching space.

3.2. Adaptive PSO Strategy

In the proposed scheme, we consider a constrained processing strategy similar to [18] with two possible solutions for three scenarios. In the first scenario, we consider the fitness function which will be used for the estimation of the candidate solution. When one solution is feasible and the other is not, the system selects the one that is achievable. The fitness function is defined as:

\[ X_{i}fit = \begin{cases} 0 & \text{if } T_{total} \text{is greater than } D(W)1 \\ else \end{cases} \]

where \( T_{total} \) is the total workflow time of the \( i \)th particle. The second scenario looks at both solutions being achievable, this implies that the solution that has the shorter time for the workflow execution will be selected. To meet the deadline constrain, the scheduling plan with a lower makespan has the possibility of meeting the constraint as shown in the equation below.

\[ X_{ifit} = T_{total}X_{i} \]  

(5)

In a scenario with both solutions being feasible, the cheaper solution is selected and the fitness function is defined as:

\[ X_{ifit} = C_{total}X_{i} \]  

(6)

where \( C_{total}X_{i} \) is the workflow execution time of the \( i \)th particle. This paper aims to enhance the search capability of PSO for a wider solution space. The proposed scheme used a crossover operator and genetic algorithm for mutation operator to overcome premature convergence as seen in traditional PSO. We update the proposed strategy as :

\[ X_{i}^{t} = b_{2} \oplus B_{K}(b_{1} \oplus B_{L})(\omega \oplus \gamma(X^{t-1}_{i}), pBest_{i}^{t-1}) \times gBest_{i}^{t-1} \]  

(7)

where \( \gamma \) represents the mutation operator \( B_{K} \) and \( B_{L} \) are both crossover operators. The flow chart in Figure 2 is used to summarise the proposed scheme. A brief description
of figure 2 is presented below. In Step 1, the relevant particles are initialised, these parameters include: the inertia weight, iterations and the maximum coefficient. During the initialisation, the population of particles are generated. In step 2, the workflow is scheduled based on mapping, then the fitness value is computed based on Eqn. 1, 2 and 3. In Step 3, each particle is updated based on Eqn 1, 2 and 3. In the fourth step, the fitness value is recomputed and if the current value is better than pBEst, the current value is replaced for the pBest particle. In the fifth step, if the current value is better than gBest, the proposed scheme replaces the current value for the gBest then Step three is repeated till the maximum number of iterations are reached.

4. Performance Evaluation

The simulations are carried out on a 64-bit Windows 10 configured with 32GB of memory and 2.89GHz frequency. We set the particle number and the maximum iterations to 100 and 1000 respectively. We define $b_{1\text{start}} = 0.9$, $b_{1\text{end}} = 0.2$, $b_{2\text{start}} = 0.4$ and $b_{2\text{end}} = 0.9$ as described in [27]. The proposed algorithm and its performance evaluation results are discussed in the following subsections.

4.1. Experimental Setup

The experiments are conducted with three scientific workflows which have been widely used and investigated as discussed in [28]. The performance of the VM was measured using the corresponding capacity which are the CPUs. In the experiment, it is assumed that each cloud provided eight VMs with different capacity and cost per hour.

4.2. Experimental Workflows

The proposed algorithm was evaluated on the following three real application workflows used in diverse scientific domains.

- Montage is an astronomical application which is used to generate custom mosaics of the sky based on a set of images. Most of its tasks are characterised as I/O intensive which do not require much processing capacity.
- CyberShake is used in earthquake science to characterize earthquake hazards in a region by generating synthetic seismograms. It may be classified as a data intensive workflow with large memory and CPU requirements.
- LIGO workflow is used in gravitational physics for detecting gravitational waves produced by various events in the universe. This workflow is characterized as having CPU intensive tasks that consume large memory.

4.3. Comparison with State-of-the-art

In this subsection, the proposed scheme is compared to related algorithms proposed in literature. These algorithms are modified to fit into the multi cloud environment.
The two algorithms used as for the performance comparison which are closely related to the proposed scheme. Multi-Cloud Path Critical Path (MCPCPP) [29] and the contribution from the authors in [29] based on PSO strategy. MCPCPP tries to minimise the execution cost of workflows and meeting deadline constraints at the same time while the PSO strategy tries to achieve the same purpose using a single cloud. We modify the PSO to adapt to multi-cloud and refer to it in this paper as Enhanced-PSO (EPSO) for evaluation purposes. To complete the workflow within the scheduled deadline, the experiments are conducted using four deadlines.

\[ D_i(W) = n_1 Min(W), i = (1, ..., 4) \]  

where \( Min(W) \) represents the makespan of the workflow \( W \) as described in HEFT [30] and \( n_1 \) is the value selected from the deadline. In order to evaluate the performance of the three different algorithms (the proposed scheme APSO, MCPCPP and EPSO), each test case is executed 100 times since the synthetic workflows have different characteristics. The workflows were normalised for comparison using normalised workflow cost in Eqn 9.

\[ WC_N(W) = \frac{EC_T(W)}{SC_M(W)} \]  

where \( WC_N(W) \) represents the normalised workflow cost, \( EC_T(W) \) is the total execution cost of workflow and \( SC_M(W) \) is the modified scheduling cost of workflow which is the cheapest scheduling strategy as described in [31]. The completion rate of Montage workflow is presented in Fig. 2. APSO and EPSO performed better than MCPCPP. MCPCPP is less than 30 percent in \( D_1(W) \), close to 80 percent in \( D_3(W) \) and reaches 100 percent in \( D_4(W) \). In Fig 3, the completion rate of Cybershake workflow is presented. MCPCPP performs better than the proposed scheme and EPSO. With the increase in deadline, our proposed scheme outperforms EPSO and MCPCPP. In \( D_4(W) \), all algorithms are close to completion. However, APSO performs better than EPSO and MCPCPP. In Fig. 4, the completion rate of Ligo is presented. Our proposed scheme APSO beforms better than two PSO based algorithms. Although all the algorithms have similar completion rate in \( D_3(W) \) and \( D_4(W) \). APSO performs better than other schemes in \( D_1(W) \) and \( D_2(W) \).

![Figure 2. Completion Rate of Montage workflow across three Scheduling algorithms](image-url)
In this paper, an adaptive PSO strategy (APSO) is proposed for scheduling scientific workflows. The data transfer cost was considered in this paper in as well as the delays in
the VMs. We introduced random two point crossover and one point mutation operator from GA to avoid premature convergence. These operators improve the diversity of the population during evolution. Experimental results show that APSO adapts better than the compared approaches based on PSO. In our future work, we will pay attention to the selection of resource pool based on the structure of the workflow to reduce resource acquisition delays and VM performance variations.

References

[1] N. Xiong, X. Jia, L. T. Yang, A. V. Vasilakos, Y. Li, and Y. Pan, “A distributed efficient flow control scheme for multirate multicast networks,” IEEE Transactions on Parallel and Distributed Systems, vol. 21, pp. 1254–1266, Sep. 2010.

[2] F. Marozzo, D. Talia, and P. Trunfio, “A workflow management system for scalable data mining on clouds,” IEEE Transactions on Services Computing, vol. 11, pp. 480–492, May 2018.

[3] S. Ghazouani and Y. Slimani, “A survey on cloud service description,” J. Netw. Comput. Appl., vol. 91, pp. 61–74, Aug. 2017.

[4] L. Liu, M. Zhang, R. Buyya, and Q. Fan, “Deadline-constrained coevolutionary genetic algorithm for scientific workflow scheduling in cloud computing,” Concurrency and Computation: Practice and Experience, vol. 29, no. 5, p. e3942, 2017. e3942 CPE-16-0064.R2.

[5] S. Singh and I. Chana, “Cloud resource provisioning: survey, status and future research directions,” Knowledge and Information Systems, vol. 49, pp. 1005–1069, Dec 2016.

[6] W. Chen and 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, pp. 29–43, Jan 2009.

[7] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, “Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility,” Future Generation Computer Systems, vol. 25, no. 6, pp. 599 – 616, 2009.

[8] M. Masdari, S. S. Nabavi, and V. Ahmadi, “An overview of virtual machine placement schemes in cloud computing,” Journal of Network and Computer Applications, vol. 66, pp. 106 – 127, 2016.

[9] P. Kumar and R. Kumar, “Issues and challenges of load balancing techniques in cloud computing: A survey,” ACM Comput. Surv., vol. 51, pp. 120:1–120:35, Feb. 2019.

[10] Z. Sanaei, S. Abolfazli, A. Gani, and R. Buyya, “Heterogeneity in mobile cloud computing: Taxonomy and open challenges,” IEEE Communications Surveys Tutorials, vol. 16, pp. 369–392, First 2014.

[11] B. Lin, W. Guo, and X. Lin, “Online optimization scheduling for scientific workflows with deadline constraint on hybrid clouds,” Concurr. Comput. : Pract. Exper., vol. 28, pp. 3079–3095, Aug. 2016.

[12] F. Wu, Q. Wu, and Y. Tan, “Workflow scheduling in cloud: a survey,” The Journal of Supercomputing, vol. 71, pp. 3373–3418, Sep 2015.

[13] F. Fakhfakh, H. H. Kacem, and A. H. Kacem, “Workflow scheduling in cloud computing: A survey,” in 2014 IEEE 18th International Enterprise Distributed Object Computing Conference Workshops and Demonstrations, pp. 372–378, Sep. 2014.

[14] M. A. Rodriguez and R. Buyya, “Deadline based resource provisionignand scheduling algorithm for scientific workflows on clouds,” IEEE Transactions on Cloud Computing, vol. 2, pp. 222–235, April 2014.

[15] Q. Wu, F. Ishikawa, Q. Zhu, Y. Xia, and J. Wen, “Deadline-constrained cost optimization approaches for workflow scheduling in clouds,” IEEE Transactions on Parallel and Distributed Systems, vol. 28, pp. 3401–3412, Dec. 2017.

[16] J. Sahni and D. P. Vidyarthi, “A cost-effective deadline-constrained dynamic scheduling algorithm
for scientific workflows in a cloud environment,” IEEE Transactions on Cloud Computing, vol. 6, pp. 2–18, Jan 2018.
[17] Z. Li, J. Ge, H. Hu, W. Song, H. Hu, and B. Luo, “Cost and energy aware scheduling algorithm for scientific workflows with deadline constraint in clouds,” IEEE Transactions on Services Computing, vol. 11, pp. 713–726, July 2018.
[18] Z. Yu, J. Zhou, W. Guo, L. Guo, and Z. Yu, “Participant selection for t-sweep k-coverage crowd sensing tasks,” World Wide Web, vol. 21, pp. 741–758, May 2018.
[19] M. Malawski, G. Juve, E. Deelman, and J. Nabrzyski, “Cost- and deadline-constrained provisioning for scientific workflow ensembles in iaas clouds,” in SC ’12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, pp. 1–11, Nov 2012.
[20] M. Mao and M. Humphrey, “Auto-scaling to minimize cost and meet application deadlines in cloud workflows,” in SC ’11: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–12, Nov 2011.
[21] S. Abrishami and M. Naghibzadeh, “Deadline-constrained workflow scheduling in software as a service cloud,” Scientia Iranica, vol. 19, no. 3, pp. 680–689, 2012.
[22] Z. Wu, Z. Ni, L. Gu, and X. Liu, “A revised discrete particle swarm optimization for cloud workflow scheduling,” in Proceedings of the 2010 International Conference on Computational Intelligence and Security, CIS ’10, (Washington, DC, USA), pp. 184–188, IEEE Computer Society, 2010.
[23] S. Pandey, L. Wu, S. M. Guru, and R. Buyya, “A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments,” in 2010 24th IEEE International Conference on Advanced Information Networking and Applications, pp. 400–407, April 2010.
[24] A. Iosup, S. Ostermann, M. N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, “Performance analysis of cloud computing services for many-tasks scientific computing,” IEEE Transactions on Parallel and Distributed Systems, vol. 22, no. 6, 2011.
[25] H. M. Fard, R. Prodan, and T. Fahringer, “A truthful dynamic workflow scheduling mechanism for commercial multicloud environments,” IEEE Transactions on Parallel and Distributed Systems, vol. 24, pp. 1203–1212, June 2013.
[26] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in Proceedings of ICNN’95 - International Conference on Neural Networks, vol. 4, pp. 1942–1948 vol.4, Nov 1995.
[27] Y. Shi and R. Eberhart, “A modified particle swarm optimizer,” in 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360), pp. 69–73, May 1998.
[28] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M. Su, and K. Vahi, “Characterization of scientific workflows,” in 2008 Third Workshop on Workflows in Support of Large-Scale Science, pp. 1–10, Nov 2008.
[29] B. Lin, W. Guo, N. Xiong, G. Chen, A. V. Vasilakos, and H. Zhang, “A pretreatment workflow scheduling approach for big data applications in multicloud environments,” IEEE Transactions on Network and Service Management, vol. 13, pp. 581–594, Sep. 2016.
[30] H. Topcuoglu, S. Hariri, and Min-You Wu, “Performance-effective and low-complexity task scheduling for heterogeneous computing,” IEEE Transactions on Parallel and Distributed Systems, vol. 13, pp. 260–274, March 2002.
[31] S. Abrishami, M. Naghibzadeh, and D. H. Epema, “Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds,” Future Gener. Comput. Syst., vol. 29, pp. 158–169, Jan. 2013.