WCCI 2012 IEEE World Congress on Computational Intelligence
June, 10-15, 2012 - Brisbane, Australia IEEE CEC

Evolving Multi-objective Strategies for Task Allocation of Scientific Workflows on Public Clouds

Claudia Szabo
School of Computer Science
The University of Adelaide
Australia, SA, 5005
Email: claudia.szabo@adelaide.edu.au

Trent Kroeger
School of Computer Science
The University of Adelaide
Australia, SA, 5005
Email: trent.kroeger@adelaide.edu.au

***Abstract*—With the increase in deployment of scientific application on public and private clouds, the allocation of workflow tasks to specific cloud instances to reduce runtime and cost has emerged as an important challenge. The allocation of scientific workflows on public clouds can be described through a variety of perspectives and parameters and has been proved to be NP-complete. This paper presents an optimization framework for task allocation on public clouds. We present a solution that considers important parameters such as workflow runtime, communication overhead, and overall execution cost. Our multi-objective optimization framework builds on a simple and extensible cost model and uses a heuristic to determine the optimal number of cloud instances to be used. Using the Amazon Elastic Compute Cloud (EC2) and Amazon Simple Storage Service (S3) as an example, we show how our optimization heuristics lead to significantly better strategies than other state-of-the-art approaches. Specifically, our single-objective optimization is slightly better than a simple heuristic and a particle swarm optimization approach for small workflows, and achieves significant improvements for larger workflows. In a similar manner, our multi-objective optimization obtains similar results to our single-objective optimization for small-size workflows, and achieves up to 80% improvement for large-size workflows.**

I. Introduction

Cloud computing is increasingly becoming the platform of choice for affordable, easy-to-use off-site computation, due mainly to its appealing advantages such as pay-per-use, elasticity, and availability among others [17]. However, it has yet to become as widely accepted as initially envisaged [9], [17], [25]. In particular in the scientific community, there is a need for a computational platform to replace High Performance Computing (HPC) for on-demand, responsive, and highly customized computational scientific research [9]. Cloud computing offers an appealing promise for the execution of emerging computationally and I/O intensive scientific workflows. The suitability of cloud computing for scientific computation has been the focus of research in the past years [9], [25], with recent work proposing solutions that schedule and execute scientific workflows in the cloud [5], [24], [26].

The task allocation problem is an NP-complete problem [10] and as such various heuristics have been proposed [5], [26]. Traditional task allocation algorithms look at the problem from a provider perspective, in which the focus is to maximize the utilization of specific *homogeneous* resources in a grid or cluster, when multiple workflows from different users are submitted [1], [15]. In contrast, in cloud computing, virtual clusters can be easily set up using *heterogeneous* resources [3], and the perspective is shifted to the user that is concerned with reduced runtime at minimal costs [17]. The inherent variety of cloud providers and solutions leads to a different formulation of the task allocation problem, as different types of resources (hardware, software), and utility functions (single or multi-objective, performance, cost, profit, energy consumption) can be devised [17].

In this paper, we present a problem formulation for the allocation of scientific workflows on public clouds and discuss an optimization framework to solve it. Our framework considers communication overhead and transfer bandwidth between scientific tasks and proposes both single objective and multi-objective evolutionary algorithms to reduce the total workflow runtime, as well as the total cost of executing the workflow on a platform built from various cloud instances. To illustrate these algorithms, we employ a model based on the Amazon Elastic Compute Cloud (EC2) and Scalable Storage Service (S3), but our framework can be easily extended to other cloud providers. We analyze several scientific workflows of different sizes and characteristics. Firstly, our experiments allocate I/O - intensive workflows such as Montage [4], which creates science-grade astronomic image mosaics using data collected from telescopes. We next analyze computationally-intensive workflows such as Epigenomics [4], which maps short DNA segments to previous existing references. Secondly, we analyze scientific workflows of varying size, in the range of tens, hundreds, and thousands of tasks.

This paper is organized as follows. Section II presents the task allocation problem in detail, and related work is discussed in Section III. The formulation of this problem as an optimization problem follows in Section IV. Our experiments follow in Section V. Section VI concludes this paper and presents our future work.

II. Execution of Scientific Workflows on Public Clouds

A scientific workflow is a directed acyclic graph (DAG) representation of a sequence of tasks in a program that solves a scientific problem [16]. Tasks in a workflow are routines that may require one or more input files and that produce

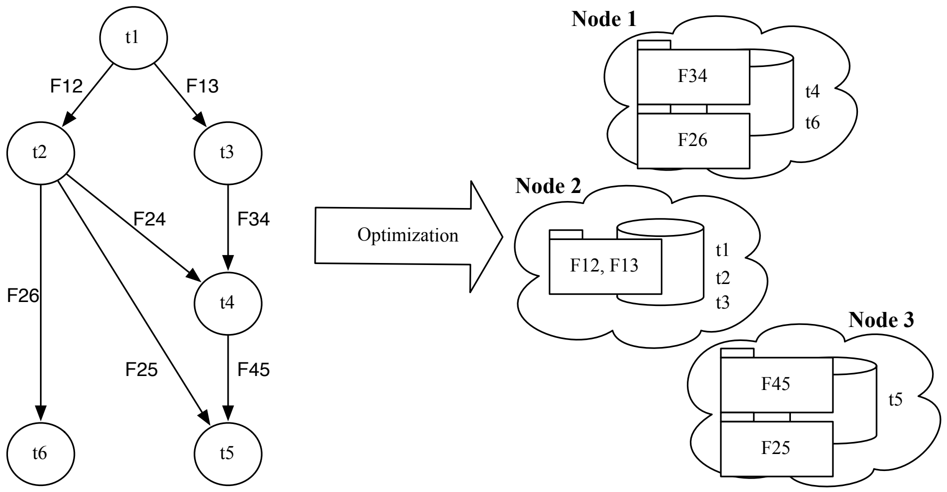


Fig. 1. Allocation of Scientific Workflows on Public Clouds

one or more output files. Task dependence is represented by the DAG, in which tasks are nodes connected by edges representing file dependencies. More formally, we can define a scientific workflow as the set of tasks $T = \{t_i\}$, of size $m = |T|$, that are organized based on their dependencies in the set $W = \{(t_i, t_j, F_{ij}) | t_i, t_j \in T, F_{ij} \in \Phi\}$. In the DAG representation, a weighted edge between task t_i and t_j has the weight F_{ij}, as shown in Fig. 1. The set Φ contains the file sizes, in gigabytes (GB), transferred between the tasks in the workflow, with F_{ij} representing the total size of the files produced by task t_i that are needed by its child t_j. The task allocation problem looks at assigning workflow tasks on various cloud instances, as shown in Fig. 1, where the six workflow tasks are allocated to three cloud instances.

Scientific workflows are both data and computationally intensive, with file sizes up to hundreds of gigabytes exchanged between tasks. A key challenge is how to allocate tasks on specific cloud instances to ensure a minimum communication overhead between tasks that exchange large files, e.g. Montage, which has 99% execution time taken by data operations [12], [13], [21]. The optimization of the allocation of tasks on a number of nodes can be defined from a variety of perspectives and using various parameters. For example, from a cloud user perspective, of interest are inter-task communication, communication topology, and different instance types depending on the task size. Moreover, a trade-off between runtime and cost might occur for large scientific workflows, as more cloud instances (hence an increase in cost) might be employed to reduce the runtime. From a cloud provider perspective, issues of importance are the allocation of heterogeneous resources, a minimized inter-process communication, as well as meeting availability and reliability. For a cloud provider, optimization functions could look at profit, performance, meeting deadlines, ensuring security, and reducing energy consumption. Lastly, when clouds from different providers are aggregated in a federated cloud, network latency, reliability, and availability become important challenges. From this perspective, the objective is to increase profit, to ensure appropriate time-zones for sensitive applications running on geographically distributed clouds, and reduce energy consumption. Moreover, the execution of scientific workflows on a parallel platform can be *static* or *dynamic*. A static allocation considers the initial status of the cloud platform. In the case of dynamic scientific workflows, where the load and output of individual tasks and cloud instances changes over time, a task needs to be moved from one cloud instance to another transparently for the other tasks in the workflow, and at minimal impact on the runtime and cost.

Hereafter, we use the term *node* or *cloud instance* to refer to a virtual machine instance obtained from a cloud provider that offers Infrastructure-as-a-Service [17], in which clusters can be formed using bare-metal virtual machine instances. In this paper, we employ the Amazon Elastic Cloud Compute (EC2) service as an example of such a cloud. Our model assumes that tasks are allocated to nodes to be executed in a specific order. In other words, our optimization framework will not only assign tasks to specific nodes but also specify the order in which these tasks are executed, to minimize runtime and cost. Thus our proposed optimization framework performs both *allocation* and *scheduling* of tasks on a number of nodes.

III. Related Work

Deelman et al. [7] have proposed the Pegasus workflow management system (WMS) as a framework that maps complex scientific workflows into grid resources, using the Condor system. Similar to most workflow management systems [16], [18], Pegasus schedules tasks based on their earliest start or finish time, and/or resource utilization, and does not consider other parameters such as placement of data and resource cost. A recent paper investigates the suitability of cloud computing for the execution of Pegasus workflows, but only a single cloud instance (in which the Condor scheduler is installed) is considered [22]. Other approaches include matrix-based heuristics [26], hypergraph-based heuristics [5], and particle swarm optimizations [19], [24].

In matrix-based heuristics, a similarity matrix is built for files exchanged in the workflow [26]. In this matrix, entry $d(ij)$ shows the number of tasks that require both files $f(i)$ and $f(j)$. In the initial stage, the similarity matrix is used to cluster the files such that highly related files are placed in the same site. When a task arrives during the workflow execution, it is greedily assigned to the execution site that contains most of its files. Similarly, when a data file is generated, it is placed in a site that contains files with a high similarity score with the input file. While a greedy approach offers an increase in speed, it might not lead to an optimal solution, and similarity scores between data files are difficult to calculate.

Hypergraph-based heuristics [5] model the workflow as a hypergraph considering both data placement and task assignment. This permits simultaneous construction of data placement and task assignment schemes that distribute the storage and computational loads among the execution sites with respect to some pre-determined ratios. This static allocation approach assumes that storage and execution capabilities of each cloud instance, as well as the exact number of cloud instances, is provided by the user, together with information about cloud instance desirability. The approach achieves a 35% improvement over the matrix-based heuristic in terms of load balancing. However, this approach assumes a cost model based only on the communication overhead (i.e. the number of files exchanged between sites) in a static workflow.

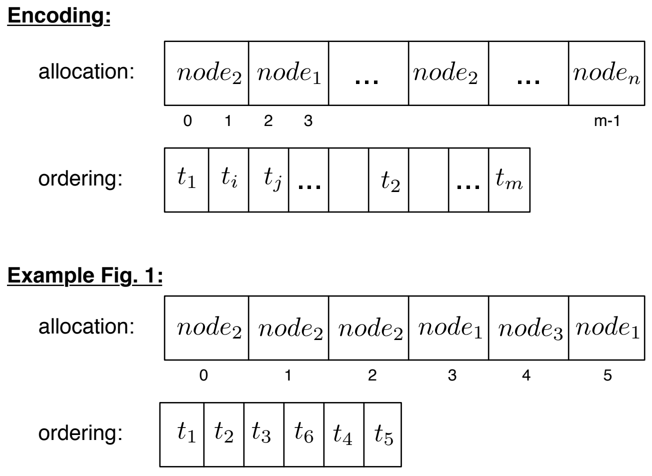
From a provider perspective, evolutionary algorithms have been used to allocate tasks on cluster nodes to reduce compute node utilization [15], but the task order on the compute nodes, as well as execution cost and runtime are not considered. Other approaches [19], [24] model the task-resource mapping as a particle swarm optimization problem (PSO) [14] to minimize the overall cost of execution. However, this limits the optimization to a single objective, such as runtime, as current multi-objective particle swarm optimizers only deal with small problems. As we show in Section V, this is also the case for our simple PSO implementation, which cannot allocate workflows with size greater than 500 tasks. In contrast, we propose both single and multi-objective evolutionary algorithm based optimization frameworks. Our single-objective approach achieves good results for small workflows of up to 100 tasks, while our multi-objective approach obtains significant improvements for large-size workflows of up to 1,000 tasks.

IV. Problem Formulation

We represent solutions to the task allocation problem by two chromosomes. The first chromosome specifies the allocation strategy and maps individual tasks to nodes within the cloud. The second chromosome in each candidate solution represents a total ordering over all tasks, as shown in Fig. 2. Specifically, in the first chromosome the index represents the task id in the workflow, from 0 to $m-1$, and the values represent the node id, from 1 to n. The second chromosome contains a task ordering from t_1 to t_m. For the example in Fig. 1, we have $n = 3, m = 6$, and an encoding as shown in Fig. 2.

With information about both the task allocation and ordering, each solution can be evaluated to estimate the overall runtime and total cost of scientific workflow execution. While there is likely to be a correlation between runtime and total cost for the execution of scientific workflows, the highly parallel nature of these workflows and communication overheads may give rise to significant trade-offs between these two objectives. To handle this, we propose the application of recently developed multi-objective evolutionary algorithms, such as NSGA-II to optimize towards a solution [6].

Rather than evolving solutions towards a single optima, multi-objective evolutionary algorithms maintain a set of optimal solutions - termed collectively a pareto front - that represent different trade-off solutions to be chosen by the scientist. In this case, the NSGA-II algorithm is used to optimize both the allocation and ordering strategies simultaneously by applying appropriate crossover and mutation operators. This problem is interesting from an algorithmic point of view as it involves a number of complexities, including the multi-objective nature of the problem, co-evolution of two chromosomes and the construction of crossover and mutation operators that respect task dependencies. The following subsections describe how solutions to this problem may be evaluated, with respect to the objectives of runtime and cost, and how solutions may be modified using problem-specific operators.



Encoding:
allocation:
$node_2$ $node_1$ ... $node_2$ ... $node_n$
0 1 2 3 m-1
ordering:
t_1 t_i t_j ... t_2 ... t_m
Example Fig. 1:
allocation:
$node_2$ $node_2$ $node_2$ $node_1$ $node_3$ $node_1$
0 1 2 3 4 5
ordering:
t_1 t_2 t_3 t_6 t_4 t_5

Fig. 2. Chromosome Encoding

A. Fitness Function Evaluation

Each solution is evaluated using the following method. For each possible allocation a of workflow W on n cloud instances we use the notation $a = \{a_1 = (\ldots, t_{i1}, \ldots, t_{j1}), a_2 = (\ldots, t_{i2}, \ldots, t_{j2}), \ldots, a_n = (\ldots, t_{in}, \ldots, t_{jn})\}$ to denote that a_i represent all the tasks running on the cloud instance i. For each allocation a, our evaluation method returns a three-dimensional array representing the communication overhead, runtime, and cost respectively. The communication overhead is calculated as the sum of the file sizes exchanged by workflow tasks across cloud instances. Consider the case shown in Fig. 3, when a parent t_i is allocated to a cloud instance and all or some of its child tasks t_j are allocated to another cloud instance. When using the Scalable Storage Service (S3) model, the parent will upload its output files to S3 (the first component in Equation 1) and the children will download the files from S3 when needed (the second component in Equation 1). The additional upload incurred by the parent uploading its files to S3 might seem like an additional overhead and would discourage the use of S3. However, all of the scientific workflows we analyzed are characterized by a parent with a large number of children (over 500 in one case). In this case, the impact of the single upload is minimal. Besides being established as a de-facto approach, the use of S3 is also justified by the need of data persistence, parallelized access to buckets, and increased bandwidth [9], [23]. More formally,

$$overhead(a) = \sum F_i + \sum F_{ij} \tag{1}$$

where F_i represents all of 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 $(t_i, t_j, F_{ij}) \in W$.

The runtime of a scientific workflow given a specific allocation a is calculated as the maximum endtime of the execution of tasks per each cloud instance, assuming all input/output files are present, plus the transfer overhead required to transfer the needed files from one cloud instance to another. Thus

$$runtime(a) = max_k(max_i(end\_time(t_{ik})) + \frac{overhead(a)}{bandwidth} \tag{2}$$

where $t_{ik} \in a_k, a_k \in a$, and the value of $bandwidth$ and other Amazon EC2 and S3 parameters are specified in Table I. The maximum end_time of tasks per node is calculated using a simple algorithm that at each iteration visits each cloud instance and determines the first task that can be executed. The runtime of that task is then established according to the workflow specification file[1] and the value of the end_time variable is increased accordingly. This follows the behavior of a simple first-come first-served scheduler. When all tasks have been executed, the maximum end_time of all nodes is considered. This follows the assumption that all cloud instances will be shut down only when the last task completes.

The cost of running a scientific workflow given a specific allocation a is calculated as the cost of keeping all cloud instances running, $runtime\_cost$, and the cost of communication when using the Amazon S3 model, $S3\_cost$:

$$cost(a) = runtime\_cost(a) + S3\_cost(a) \tag{3}$$

Fig. 3 shows a simple example of the calculation of $S3\_cost$ when using Amazon S3. As it can be seen, we consider the use of S3 whenever a parent task i and all or some if its child tasks j are not allocated to the same cloud instance. In this case, the parent will upload its output files to Amazon S3 using buckets of 5GB each, as per the Amazon S3 instructions [2]. This results in a number p_i of PUT requests, which are priced by Amazon according to Table I, as $cost_p$. All children that are not allocated to the same node as their parent will obtain the required files from S3 using GET requests for each 5GB bucket. This results in a number g_j of GET requests, which are priced by Amazon according to Table I, as $cost_g$. Thus we have

$$S3\_cost(a) = \sum_{i,j} p_i * cost_p + g_j * cost_g$$

where $\exists k, l, k \neq l$ such that $t_i \in a_k$ and $t_j \in a_l$, and $(t_i, t_j, F_{ij}) \in W$.

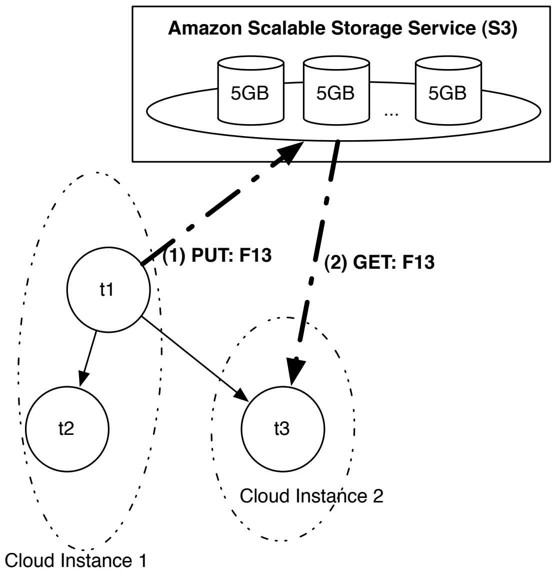


Fig. 3. Using the Amazon Scalable Storage Service (S3)

B. Problem-specific Operators

The proposed optimization framework is based on common metaheuristic algorithms that are used in conjunction with the fitness function evaluation method described above. Both our single-objective and multi-objective algorithms apply the same set of evolutionary operators for selection, crossover and mutation. Due to solution constraints implied by the workflow task-graphs, the crossover and mutation operators have problem-specific aspects that are based on a similar optimization problem for the allocation of processes to computing nodes [15].

For each algorithm, a standard Binary Tournament selection operator is used at the beginning of each algorithm iteration to select two parent solutions from the population. This selection operator chooses two pairs of solutions uniformly at random from the population, performs a comparison of the fitness of the solutions within each pair and selects the best solution from each pair. The two selected solutions are then considered parent solutions for the subsequent crossover operator.

Two separate crossover operators, corresponding to the two chromosomes of the parent solutions, are applied, with specified probability. Firstly, a standard single-point crossover is applied to the allocation chromosomes of the solutions, where a crossover point is chosen uniformly at random to divide the chromosome. The first segment of the chromosome from one parent is then combined with the second segment of the other parent to yield valid allocation chromosomes for two offspring solutions. In a second step, an order crossover operator is applied to the ordering chromosomes of the parent solutions. Such an operator is necessary since the ordering chromosome is constrained by the workflow task-graph and the application of a standard single-point crossover may lead

[1]The runtime of each task varies depending on the workload of the system. We employ benchmark values as described in Section V.

to the generation of infeasible solutions. For this operator, a crossover point is chosen uniformly at random for each parent solution and the first segment of the chromosomes for the offspring solutions are populated as per the standard single point crossover operator. Genes that were not included in the first segment are then added in the same order as they appear in the alternate parent chromosome.

A mutation operator is then applied, with specified probability, to the offspring solutions. Due to task-graph constraints, mutation of these solutions is performed as a two step process. If a solution is chosen for mutation, then each gene within its allocation chromosome is randomly flipped, with probability $1/n$, where n is the number of tasks. The flipped gene is assigned a random integer value, which represents a cloud node. However, a random mutation is not appropriate for the ordering chromosome and will often lead to solutions that invalidate workflow constraints. Instead, the second part of the process applies a swap mutation operator to the ordering chromosome. Our implementation pre-processes the input workflow to identify a list of all possible pairs of tasks that may be swapped, without violating precedence constraints. The swap mutation operator randomly chooses one pair from this list and exchanges this pair of values within the ordering chromosome.

V. Experiments

A. Experimental Parameters

Towards a realistic cloud model, we implemented a number of parameters as shown in Table I. Our cloud computing model considers a Infrastructure-as-a-Service (IaaS) model, in which the cloud provider offers the users the infrastructure required to run any type of application, regardless of its implementation and deployment details. In IaaS, the cloud users employ a virtual machine instance in a similar manner as they would use a physical machine. For our experiments, we use the characteristics of Amazon `c1.xlarge` instances, which are suited for computationally intensive applications as they have high I/O performance and a large memory capacity. This is because this type of instance has been found to be similar to the cluster nodes on which the scientific workflows in Table II were executed and for which the runtime benchmark values were obtained, as published by the Pegasus project [20]. The Pegasus project publishes the specifications of a number of scientific workflows shown in Table II. The published information includes workflow size, workflow DAG, as well as details about the execution of the workflows (runtime, file transferred, etc.) on a cluster composed of instances equivalent to Amazon `c1.xlarge` instances [12]. These workflows have been used as benchmark workflows in a variety of projects and as such are employed in our study [5], [24].

Parameter	Value
Cloud provider	Amazon
Cloud type	Infrastructure-as-a-Service (IaaS)
Cloud instance - type - CPU - memory Cloud instance price	 c1.xlarge 20 EC2 Compute Units 7 GB 0.68$/hr
S3 object size	5 GB
S3 request PUT price	0.01$
S3 request GET price	0.001$
S3 bandwidth	12 MB/s

TABLE I
CLOUD CHARACTERISTICS

Workflow	Type I/O; Memory; CPU	Number of Tasks	Number of Cloud Instances
Montage	High; Low; Low	25; 50; 100; 1,000	5; 6; 10; 77
CyberShake	Low; Low; Medium	30; 50; 100; 1,000	3; 5; 9; 5
Inspiral	Low; Medium; Medium	30; 50; 100; 1,000	3; 3; 9; 50
Sipht	Low; Medium; High	30; 60; 100; 1,000	4; 8; 12; 128
Epigenomics	Low; Medium; High	24; 46; 97; 997	2; 5; 2; 15

TABLE II
SCIENTIFIC WORKFLOW CHARACTERISTICS

Our framework determines an optimal configuration of tasks executing on a number n of cloud instances that form a virtual cloud cluster. It is important to highlight here that determining the optimal value of n for a particular workflow can impact the performance and results of the optimization. An approach would be to set a pre-defined value of n for all scientific workflows. However, as it can be seen in Table II, the size of the workflows varies from 25 to 1,000 and a single value of n might not be optimal for all. In contrast, we propose a simple heuristic to determine a good value of the number of cloud instances the workflow will be executed on. Our heuristic determines, for each workflow DAG, n as the maximum number of task sets that can be executed in parallel, as shown in Table II. Our experiments show that the heuristic values of n lead to better results than a fixed value of n, say $n = 10$, for all workflows. Space constraints prevent us from showing the results here. An analysis of the suitability of this heuristic, as well as introducing n as an objective in our optimization framework, is part of our future work.

B. Heuristic Allocation

For comparison, we implement a simple heuristic allocation that aims to reduce the communication overhead incurred by the allocation. Informally, the heuristic tries to allocate parent-child pairs of tasks that have large files exchanged between them on the same cloud instance. The heuristic proposes the concepts of *task bundle* and task *parallel sets*. Firstly, tasks $t_{i+2}, ti + 3$ in a chain, e.g. $t_i, t_{i+2}, ti + 3, t_{i+4}$ with no children and no execution constraints in the tree, and in which $F_{i+2,i+3}$ has a large size, can be aggregated, i.e. executed on the same cloud instance. The heuristic then attempts to determine tasks that can be executed in parallel and puts them in *parallel sets*. The tasks in the parallel sets have no parents or have the same parent and will be assigned to different cloud instances. Based on the task bundles and the parallel sets, the heuristic then goes through the entire workflow tree and allocates tasks to cloud instances to minimize overhead as discussed above. The allocation is then evaluated using the evaluation function discussed in Section IV-A. As shown in tables IV and III, this heuristic obtains good results for workflows of small size, but does not perform well for large workflows, because of the large number of children (more than 500) for each parent, that are allocated on different cloud instances and thus incur a high communication overhead.

C. Particle Swarm Optimization

In a similar manner, we implement a simple particle swarm optimization (PSO) algorithm, as proposed in [24]. The algorithm was implemented using jSwarm [11] and constructs a particle with size equal to the number of tasks. The PSO uses cost functions described in Equations 1 - 3. We executed 30 independent optimization runs for each workflow-objective combination and recorded the results. The mean and standard deviation for these runs are shown in tables III and IV. Our implementation obtains results similar to the above heuristic for workflows of small size, below 100 tasks. However, our implementation does not scale beyond workflows with more than 500 tasks. A solution for this known problem is suggested in [19] and involves performing a pre-ordering of tasks followed by allocation of workflow tasks on a first-come first-served basis. Since this contradicts our problem formulation, we have not implemented this solution here.

D. Single Objective Evolutionary Algorithm

We implement a single-objective evolutionary algorithm to optimize the task allocation and ordering with respect to cost and runtime. We embed the problem formulation and operators described above within a standard steady-state genetic algorithm, implemented within the jMetal optimization framework [8]. We configure this algorithm with a population size of 10, mutation and crossover probabilities of 0.9 and a maximum number of evaluations set to 100000. We have evaluated other parameter configurations but found that these mutation and crossover probabilities lead to the best results. We executed 30 independent optmization runs for each workflow to minimize overall execution cost and separately for each workflow to minimize execution runtime. The mean and standard deviation for these runs are shown in tables III and IV.

E. Multi-Objective Evolutionary Algorithm

As the principal contribution of this paper, we implement a multi-objective evolutionary algorithm to optimize the task allocation and ordering simultaneously with respect to both cost and runtime. We embed the problem formulation and operators described above within the standard NSGA-II algorithm, implemented within the jMetal optimization framework. To allow comparison with the single-objective case, we also configure this algorithm with a population size of 10, mutation and crossover probabilities of 0.9 and a maximum number of evaluations set to 100000. We again executed 30 independent optimization runs for each workflow to minimize, simultaneously, the overall execution cost and runtime. Formulating this problem as a multi-objective optimization allows, users of scientific workflows to consider tradeoffs between cost and runtime. An illustrative example of this is given in Fig. 4, which shows the pareto front generated for cost and runtime optimization of the Sipht 1000 workflow.

To allow for meaningful comparison with the single-objective variant, the mean and standard deviation for the best cost and runtime achieved for each of these runs are shown in Table III and Table IV respectively.

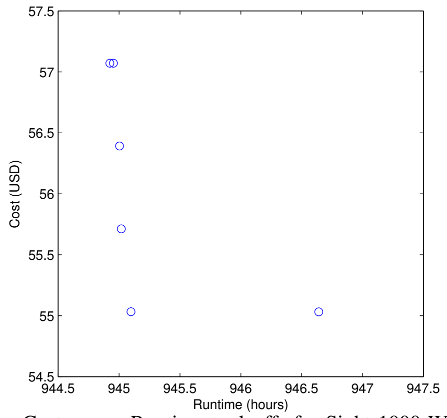


Fig. 4. Cost versus Runtime tradeoffs for Sipht 1000 Workflow

F. Analysis of Results

Tables III and IV summarize the experimental results and list the mean cost and runtime optimization results for each algorithm-workflow combination and the standard deviation across all runs for the stochastic algorithms trialled. An examination of this data shows that the evolutionary algorithms proposed in this paper clearly out-perform the heuristic allocation and particle swarm optimization approaches, particularly for workflows with larger number of tasks. This is confirmed statistically through the application of Mann-Whitney-Wilcoxon tests to compare the performance of the particle swarm optimization algorithm to that of the single objective evolutionary algorithm for workflows of size aprox. 100. The results of these tests, shown in tables V and VI, confirm that the single-objective algorithm achieves better results for all workflows tested at a level of significance less than 0.001.

It is also observed from the data that for large workflows, the multi-objective algorithm seems to out-perform the single-objective algorithm. This is also confirmed statistically through the application of the Mann-Whitney-Wilcoxon tests, shown in tables VII and VIII. These results show that the multi-objective algorithm achieves better results for both cost and runtime optimization for all workflows tested at a level of significance less than 0.001. One possible reason for this result is that the NSGA-II algorithm implicitly includes a diversity mechanism to spread solutions across the pareto front. It is recognised that on some occasions this can lead to increased performance, compared to single objective algorithms within which no such diversity mechanisms have been implemented.

Workflow	**Mean**$_{PSO}$	**Mean**$_{SO}$	**W**$_{Statistic}$	**p-value**
Montage 100	27.2	2.49	900	<0.001
CyberShake 100	369.92	1.36	900	<0.001
Inspiral 100	7.41	2.16	900	<0.001
Sipht 100	30.26	3.02	900	<0.001
Epigenomics 97	111.93	6.53	900	<0.001

TABLE V
MANN-WHITNEY-WILCOXON COMPARISON FOR PSO AND SINGLE OBJECTIVE ALGORITHMS WITH RESPECT TO COST (US $)

Workflow	Heuristic Allocation	PSO		Single Objective		Multi-Ojbective	
		Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
Montage 25	2.26	6.8	0.001	0.88	0.31	0.77	0.23
Montage 50	3.8	13.7	0.001	1.31	0.34	1.17	0.36
Montage 100	6.95	27.2	0.001	2.49	0.56	2.44	0.70
Montage 1,000	64.55	-	-	**54.02**	2.39	**45.32**	2.00
CyberShake 30	31.01	139.4	0.001	1.20	0.32	1.20	0.21
CyberShake 50	43.99	192.44	0.001	1.36	0.001	1.36	0.005
CyberShake 100	82.52	369.92	0.001	1.36	0.001	1.36	0.001
CyberShake 1,000	910.35	-	-	**4.11**	1.36	**1.37**	0.003
Inspiral 30	2.74	2.88	0.27	1.224	0.29	1.08	0.33
Inspiral 50	4.11	3.42	0.12	1.36	0.001	1.31	0.17
Inspiral 100	4.13	7.41	0.48	2.16	0.17	2.05	0.001
Inspiral 1,000	7.35	-	-	**13.11**	1.29	**11.39**	1.08
Sipht 30	4.22	5.44	0.001	1.71	0.43	1.15	0.31
Sipht 60	11.12	18.04	0.73	2.31	0.49	2.14	0.37
Sipht 100	13.32	30.26	2.38	3.02	0.61	2.98	0.58
Sipht 1,000	98.59	-	-	**67.78**	2.51	**57.31**	2.88
Epigenomics 24	15.12	52.76	4.25	1.67	0.38	1.69	0.39
Epigenomics 46	22.45	75.53	5.71	2.13	0.001	2.13	0.001
Epigenomics 97	62.88	111.93	14.65	6.53	0.35	6.51	0.37
Epigenomics 997	372.53	-	-	**106.15**	25.93	**19.81**	3.88

TABLE III
COMPARISON OF OPTIMIZATION METHODS WITH RESPECT TO COST (US $)

Workflow	Heuristic Allocation	PSO		Single Objective		Multi-Ojbective	
		Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
Montage 25	1.46	8.76	0.04	0.07	0.04	0.02	0.003
Montage 50	3.15	18.97	0.09	0.36	0.47	0.24	0.35
Montage 100	7.34	38.42	0.09	1.91	0.97	1.54	0.93
Montage 1,000	79.72	-	-	**62.94**	2.70	**52.94**	2.17
CyberShake 30	37.73	203.36	0.002	0.07	0.009	0.068	0.01
CyberShake 50	52.30	281.65	0.002	0.07	0.005	0.08	0.01
CyberShake 100	100.30	540.50	0.02	0.10	0.01	0.11	0.001
CyberShake 1,000	1252.89	-	-	**2.89**	3.28	**0.14**	0.01
Inspiral 30	0.18	1.22	0.03	0.37	0.002	0.37	0.06
Inspiral 50	0.50	1.58	0.05	0.40	0.003	0.39	0.03
Inspiral 100	0.60	2.12	0.07	0.38	0.01	0.37	0.05
Inspiral 1,000	2.94	-	-	**2.82**	0.17	**2.03**	0.19
Sipht 30	1.67	3.87	0.009	1.01	0.0001	0.99	0.09
Sipht 60	2.78	7.65	0.05	1.25	0.08	1.20	0.01
Sipht 100	4.00	10.93	0.28	1.40	0.11	1.35	0.13
Sipht 1,000	29.06	-	-	**17.76**	1.05	**15.06**	0.88
Epigenomics 24	13.63	73.6	6.20	0.86	0.19	1.72	0.02
Epigenomics 46	18.21	114.18	19.60	1.55	0.0001	1.96	0.001
Epigenomics 97	35.84	156.49	21.73	8.44	0.11	8.42	0.11
Epigenomics 997	443.80	-	-	**67.93**	7.32	**21.05**	3.64

TABLE IV
COMPARISON OF OPTIMIZATION METHODS WITH RESPECT TO RUNTIME (HOURS)

Workflow	Mean$_{PSO}$	Mean$_{SO}$	W$_{Statistic}$	p-value
Montage 100	27.2	2.49	900	<0.001
CyberShake 100	369.92	1.36	900	<0.001
Inspiral 100	7.41	2.16	900	<0.001
Sipht 100	30.26	3.02	900	<0.001
Epigenomics 97	111.93	6.53	900	<0.001

TABLE VI
MANN-WHITNEY-WILCOXON COMPARISON FOR PSO AND SINGLE OBJECTIVE ALGORITHMS WITH RESPECT TO RUNTIME (HOURS)

Workflow	Mean$_{SO}$	Mean$_{MO}$	W$_{Statistic}$	p-value
Montage 1000	54.02	45.32	900	<0.001
CyberShake 1000	4.11	1.37	691	<0.001
Inspiral 1000	13.11	11.39	718	<0.001
Sipht 1000	67.78	57.31	900	<0.001
Epigenomics 997	106.15	19.81	900	<0.001

TABLE VII
MANN-WHITNEY-WILCOXON COMPARISON FOR SINGLE AND MULTIPLE OBJECTIVE ALGORITHMS WITH RESPECT TO COST (US $)

Workflow	Mean$_{SO}$	Mean$_{MO}$	W$_{Statistic}$	p-value
Montage 1000	62.94	52.94	900	<0.001
CyberShake 1000	2.89	0.14	768	<0.001
Inspiral 1000	2.82	2.03	899	<0.001
Sipht 1000	17.76	15.06	866	<0.001
Epigenomics 997	67.93	21.05	900	<0.001

TABLE VIII
MANN-WHITNEY-WILCOXON COMPARISON FOR SINGLE AND MULTIPLE OBJECTIVE ALGORITHMS WITH RESPECT TO RUNTIME (HOURS)

VI. CONCLUSION AND FUTURE WORK

The allocation of scientific workflows on public clouds is a NP-complete problem that can be analyzed from a variety of perspectives and using different parameters. In this paper, we propose an evolutionary approach that looks at optimizing execution cost and execution runtime, when the scientific workflow is executed on an Amazon EC2 cloud that uses

Scalable Storage Service (S3). The contributions of this paper are twofold. Firstly, we propose single and multi-objective evolutionary algorithms with problem-specific operators for mutation and crossover. Secondly, we embed within these algorithms a simple and extensible cost model to evaluate a specific allocation and ordering of tasks executed on a set of cloud instances. This evaluation considers communication overhead, runtime, cost, as well as bandwidth and other cloud parameters. In contrast to existing work [15], our approach considers the *allocation* of tasks on a specific cloud instance, as well as the *scheduling* of tasks execution on the instance.

Our experiments employ computationally-intensive and I/O intensive scientific workflows as specified by the Pegasus project, with a variety of sizes from 25 to 1,000 tasks. In comparison with a simple heuristic and particle swarm optimization implementation, our single-objective approach performs by a factor of 10 better for workflows of small size, of up to 100 tasks, both for execution cost and runtime. Specifically, for workflows of size 100, our single objective optimization consistently out-performs the other approaches, obtaining a cost of under \$10 for all workflows. For example, for the `Sipht` workflow, the cost of allocation obtained by our single-objective optimization is around 2\$, whereas the cost obtained by the heuristic and PSO is around \$12 and \$30 respectively. Our multi-objective approach performs only slightly better for small workflows, but offers an 80% improvement for larger workflows, where a significant trade-off can be observed. For example, for the computationally-intensive `Epigenomics` workflow, our multi-objective approach obtains a cost of around \$20, as opposed to \$100 obtained by our single-objective approach.

The allocation of scientific workflow tasks on public clouds can be defined using a variety of parameters. In this paper, we have only considered execution cost, execution runtime, and communication overhead, but other parameters can be considered in future work. Specifically, of interest are heterogeneous cloud instance types as well as realistic runtime evaluations. Moreover, the number of cloud instances was established using a simple heuristic and would be a good candidate objective in a multi-objective implementation. Lastly, we propose to increase the efficiency of our algorithms by improving problem-specific operators and investigating alternative genome representations.

References

[1] D. Abramson, C. Enticott, and I. Altinas. Nimrod/K: Towards Massively Parallel Dynamic Grid Workflows. In *Proceedings of the ACM/IEEE Conference on Supercomputing*, 2008.

[2] Amazon. Best Practices for Using Amazon S3. http://aws.amazon.com/articles/Amazon-S3/1904, (Jan. 2012).

[3] Arstechnica. \$1,279 per Hour, 30,000-core Cluster Built on Amazon EC2 Cloud. http://arstechnica.com/business/news/2011/09/30000-core-cluster-built-on-amazon-ec2-cloud.ars, last retrieved Jan. 2012.

[4] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M.-H. Su, and K. Vahi. Characterization of Scientic Workows. In *Proceedings of the Third Workshop on Workows in Support of Large-Scale Science*, pages 1–10, 2008.

[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] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197, 2002.

[7] 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 Programming*, 13:219–237, 2005.

[8] J. Durillo, A. Nebro, F. Luna, B. Dorronsoro, and E. Alba. jmetal: a java framework for developing multi-objective optimization metaheuristics. *Departamento de Lenguajes y Ciencias de la Computación, University of Málaga, ETSI Informática, Campus de Teatinos, Tech. Rep. ITI-2006-10*, 2006.

[9] 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.

[10] D. Fernandez-Baca. Allocating Modules to Processors in a Distributed System. *IEEE Transaction on Software Engineering*, 15:1427–1436, 1989.

[11] JSwarm. PSO Optimization Package. http://jswarm-pso.sourceforge.net/, last retrieved Jan. 2012.

[12] G. Juve, E. Deelman, K. Vahi, G. Mehta, B. P. Berman, B. Berriman, and P. Maechling. Scientific Workflow Applications on Amazon EC2. *Grids, Clouds and Virtualization*, pages 71–91, 2010.

[13] D. S. Katz, J. C. Jacob, G. Berriman, J. Good, A. C. Laity, E. Deelman, C. Kesselman, G. Singh, M.-H. Su, and T. A. Prince. A Comparison of Two Methods for Building Astronomical Image Mosaics on a Grid. In *Proceedings of the International Conference on Parallel Processing*, pages 85–94, 2005.

[14] J. Kennedy and R. Eberhart. Particle Swarm Optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 1942–1948, 1995.

[15] Y.-K. Kwok and I. Ahmad. Efficient Scheduling of Arbitrary Task Graphs to Multiprocessors Using a Parallel Genetic Algorithm. *Journal of Parallel and Distributed Computing*, 47:58–77, 1997.

[16] B. Ludascher, 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. *Concurrency and Computation: Practice and Experience*, 18:1039–1065, 2006.

[17] N. I. of Standards and Technology. Cloud Computing Synopsis and Recommendations. http://csrc.nist.gov/publications/drafts/800-146/Draft-NIST-SP800-146.pdf, last retrieved Jan. 2012.

[18] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. R. Pocock, A. Wipat, and P. Li. Taverna: a Tool for the Composition and Enactment of Bioinformatics Workflows. *Bioinformatics*, 20:3045–3054, 2004.

[19] S. Pandey, L. Wu, S. Guru, and R. Buyya. A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In *Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications*, pages 400–407, 2010.

[20] Pegasus Project. Workflow Generator. https://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator, last retrieved Jan. 2012.

[21] T. Shibata, S. Choi, and K. Taura. File-access Characteristics of Data-intensive Workflow Applications. In *Proceedings of the 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pages 522–525, 2010.

[22] J.-S. Vckler, G. Juve, E. Deelman, M. Rynge, and G. B. Berriman. Experiences Using Cloud Computing for A Scientific Workflow Application. In *Proceedings of 2nd Workshop on Scientific Cloud Computing*, 2011.

[23] E. Walker. Benchmarking Amazon EC2 for High-Performance Scientific Computing. *USENIX Login*, 33:18–23, 2008.

[24] Z. Wu, Z. Ni, L. Gu, and X. Liu. A Revised Discrete Particle Swarm Optimization for Cloud Workflow Scheduling. In *International Conference on Computational Intelligence and Security*, pages 184–188, 2010.

[25] N. Yigitbasi, A. Iosup, and D. Epema. C-Meter: A Framework for Performance Analysis of Computing Clouds. In *Cluster Computing and the Grid*, pages 472–477, 2009.

[26] D. Yuan, Y. Yang, X. Liu, and J. Chen. A Data Placement Strategy in Scientific Cloud Workflows. *Future Generation Computing Systems*, 26:1200–1214, 2010.