Optimizing Job/Task Granularity for Metagenomic Workflows in Heterogeneous Cluster Infrastructures

Somayeh Mohammadi[1], Latif PourKarimi[2], Manuel Zschäbitz[1], Tristan Aretz[1], Ninon De Mecquenem[3], Ulf Leser[3] and Knut Reinert[1]

[1]Department of Mathematics and Computer Science, Freie Universität, Berlin, Germany
[2]Department of Mathematics and Computer Science, Razi University, Kermanshah, Iran
[3]Department of Mathematics and Computer Science, Humboldt-Universität zu Berlin, Berlin, Germany

Abstract

Data analysis workflows are popular for sequencing activities in large-scale and complex scientific processes. Scheduling approaches attempt to find an appropriate assignment of workflow tasks to the computing nodes for minimizing the makespan in heterogeneous cluster infrastructures. A common feature of these approaches is that they already know the structure of the workflow. However, for many workflows, a high degree of parallelization can be achieved by splitting the large input data of a single task into chunks and processing them independently. We call this problem task granularity, which involves finding an assignment of tasks to computing nodes and simultaneously optimizing the structure of a bag of tasks. Accordingly, this paper addresses the problem of task granularity for metagenomic workflows. To this end, we first formulated the problem as a mathematical model. We then solved the proposed model using the genetic algorithm. To overcome the challenge of not knowing the number of tasks, we adjusted the number of tasks as a factor of the number of computing nodes. The procedure of increasing the number of tasks is performed interactively and evolutionarily. Experimental results showed that a desirable makespan value can be achieved after a few steps of the increase.

Keywords

Data Analysis Workflow, Mathematical Programming, Makespan Minimization, Run Time Prediction, Genetic Algorithm

1. Introduction

Scientists in many domains, such as bioinformatics, remote sensing, and physics, use Data Analysis Workflows (DAWs) to sequence activities involved in large-scale and complex scientific processes [1]; [2]. These DAWs are typically represented as a Directed Acyclic Graph (DAG), which consists of a set of tasks and some directed edges between the tasks; edges show data dependencies between tasks and the priority order of task execution.

Scientists often use heterogeneous cluster infrastructures to run their DAWs because of privacy and financial concerns. Heterogeneous clusters provide high-performance computing environments that enable efficient data analysis and the execution of large-scale DAWs in a reasonable amount of time [3]. DAWs are often executed on large amounts of data, resulting in long runtimes that can exceed days or weeks [4]; [5]; [6]. Thus, in such environments, the key objective is to schedule DAW tasks across computing resources in such a way that the total execution time, also known as the makespan, is minimized.

It is well known that a high degree of parallelization can be achieved in many DAWs by splitting the input data of individual tasks into chunks and processing them independently [7]. For example, in metagenomic DAWs, the size of a reference genome as frequent input data can vary from several KB to hundreds of GB, and the reference genome typically contains thousands of genome files. The reference genome can be divided into different bins of genome files and they are processed by several independent tasks in parallel. The main challenge here is how to partition the input data; what should be the appropriate size of each chunk of input data, and how should each task be assigned to a computing node so that the makespan is minimized. We call this problem task granularity. In heterogeneous environments, this challenge is aggravated because the computing power of the existing computing nodes is different, so choosing the input size of each task to be executed on each of these computing nodes is a very effective means in optimizing the makespan. Since each task of the DAW is equivalent to a job for the cluster when a workflow is submitted for execution, the terms task and job are used interchangeably in this study.

In this paper, we propose a novel approach to task granularity for metagenomic DAWs in cluster infrastructures with makespan minimization. We first formulate the problem as a mathematical model. We solve then the proposed model using the Genetic Algorithm method. Since the calculation of makespan requires a proper estimation of tasks runtime, we apply three different methods for this estimation and also compare their accuracy.

The paper is organized as follows: Section 2 presents a review of the related work, and Section 3 illustrates the problem statement. The proposed mathematical model is introduced in Section 4. Section 5 discusses solving the proposed model using genetic algorithm. Job runtime estimation is presented in Section 6, and experimental results are presented in Section 7. Finally, Section 8 provides concluding remarks and plans for further studies.

2. Related works

In this section, we first discuss data access patterns used in scientific workflows. We then cover the scheduling of scientific workflows on heterogeneous clusters. Finally, we focus on methods for predicting the runtime of tasks, as these estimates are often used as input for scheduling approaches.

Published in the Proceedings of the Workshops of the EDBT/ICDT 2024 Joint Conference (March 25-28, 2024), Paestum, Italy.
somayeh.mohammadi@fu-berlin.de (S. Mohammadi); l.pourkarimi@razi.ac.ir (L. PourKarimi); manuez42@zedat.fu-berlin.de (M. Zschäbitz); aret01@zedat.fu-berlin.de (T. Aretz); mecquenn@informatik.hu-berlin.de (N. D. Mecquenem); leser@informatik.hu-berlin.de (U. Leser); Reinert@fu-berlin.de (K. Reinert)

2.1. Data access patterns used in scientific workflows

The data access patterns of workflow applications have been addressed by several studies [8]; [9]; [10]; [11]. Accordingly, the most commonly used patterns in scientific workflows are as follows (Fig. 1):

- **Pipeline:**This is the most basic and familiar pattern. A set of computational tasks is chained in a sequence such that the output of a parent task is the input of its child task in the chain. Because of the line dependencies in a pipeline pattern, the execution of a task cannot begin until the execution of its parent has completed and it has received the data generated by its parent.
- **Scatter:** An input data is divided into several chunks. These chunks are distributed into multiple tasks (a bag of tasks), which can be executed simultaneously because there are no dependencies between them.
- **Gather:** Multiple chunks of data are produced by multiple tasks. All of them are used as input data by a subsequent task. The later task may need to receive all the chunks and integrate them to start execution.

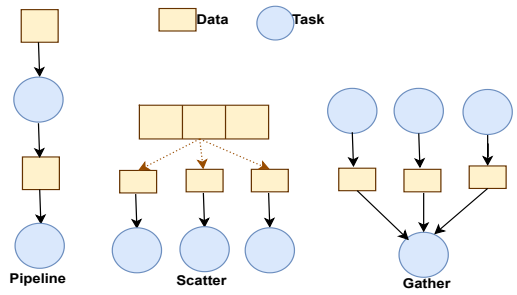


Figure 1: Common data access patters in scientific workflows.

Obviously, the scatter pattern can be so effective in reducing makespan in a distributed environment such as a cluster and cloud because it provides parallel execution of tasks on computing nodes. The implementation of scatter is an NP-hard problem (See Section 3.1), so the user is not able to do it manually to achieve an acceptable makespan. In this study, we propose an approach to address this problem.

2.2. Scheduling of scientific Workflows on heterogeneous clusters

Generally, workflow scheduling on heterogeneous infrastructures can be done in two ways, statically or dynamically [12]; [2]. Static scheduling assigns tasks to compute resources in advance, assuming that accurate information about workflow and infrastructure resources is available. Dynamic scheduling doesn't require such assumptions. On the one hand, many heuristic and meta-heuristic approaches have been provided for this problem. HEFT [13] is considered to be the most famous of these. On the other hand, mathematical optimization approaches such as MILP [14] have proposed optimal solutions to this problem and have also analyzed the problem more extensively. However, the presented state-of-the-art scheduling approaches have in common that they already know the structure of the DAG and find an optimal or sub-optimal assignment of tasks to the computing nodes.

By addressing the scattering problem of a bag, our proposed approach not only provides a suitable structure for the bag and consequently for the DAG, but also finds the best scheduling of tasks to computing nodes with the objective of minimizing the makespan.

2.3. Task runtime prediction

Most of state-of-the-art techniques for workflow scheduling rely on accurate predictions of tasks runtime [15]. Therefore, the problem of predicting the runtime of scientific workflow tasks based on historical data has been studied extensively. Recent research has used machine learning techniques to address this issue [16]; [17]; [18]; [19]. They build their initial models on historical data before the actual workflow execution. [20]; [18] use neural network methods, which are known to require large training data sets to perform well, while [3] employs a Bayesian linear regression model, which can work with few training points and provides uncertainty estimates for its predictions. Most existing approaches use the size of the file on the hard drive as input to their prediction models.

In this research, the objective is to minimize the makespan. To compute the makespan value, an acceptable estimate of the runtime of jobs is required. Assuming that we have some historical data of execution traces of jobs on computing nodes, we use three different methods to predict the runtime of jobs (See Section 6).

3. Problem statement

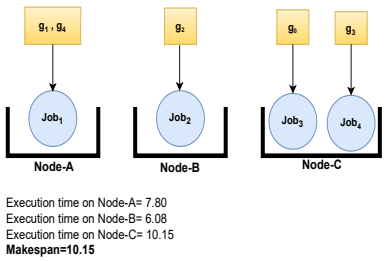
This study addresses the problem of job/task granularity of scientific workflows in heterogeneous cluster environments with the aim of minimizing makespan. The case study is a metagenomic workflow where a reference genome containing a set of genome files is the main input data of the workflow. Building FM index (Full-text index in Minute space) over a reference genome (reference genome indexing) is a common and time-consuming task in metagenomic DAWs [21] and such a task is often used by the bioinformatics workflow community, so it is a good case study for optimizing job/task granularity.

In general, the system model includes the following steps: the first step is to collect a historical dataset of job execution traces on different computing nodes of the cluster. If this dataset is not already available, it can be collected by data sampling [3]. In the second step, a proper estimation of the job runtime and its memory consumption is performed using a prediction method. Then, by solving the proposed mathematical model, the optimal size of each chunk of input data for each job and also the assignment of jobs to computing nodes is obtained. Finally, the optimal job granularity and assignment is used to execute the DAW in the cluster.

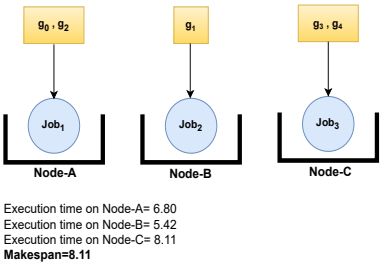
3.1. A motivating example

Suppose there is a reference genome that contains five genome files of the following sizes: $g_0 = 10$, $g_1 = 15$, $g_2 = 20$, $g_3 = 25$, $g_4 = 35$. Moreover, assume that the cluster infrastructure has three computing nodes A, B and C. The runtime of a job with an input size of S on the computing nodes is calculated by the following functions:

- $f_A(S) = \ln S^2 + 1$
- $f_B(S) = \ln S^2 + 2S$
- $f_C(S) = \ln S^2 - 4S - 10$



(a) A possible assignment.



(b) A better assignment.

Figure 2: Two possible job granularities and assignments with different makespan values.

In Fig. 2, two different possible job building and assignments of the genome files is depicted. However, there are $3^5 = 243$ different states as a solution among which the solution with the minimum makespan is the best.

In a real example, Archaea[1] has 488 genome files. Assuming that the number of cluster computing nodes is 10. Any approach based on complete enumeration and trial and error for assigning these genomes to computing nodes requires comparing a maximum number of 10^{488} to different assignments. This approach is obviously impractical. It is noticeable that Archaea is a very small reference genome among the available reference genomes. Therefore, applying the above mentioned approach for solving the related assignment problem is not efficient or even applicable. The best approach to deal with this problem is to create a mathematical model for the problem and then apply available efficient algorithms for solving mathematical optimization models.

4. The proposed mathematical model

In a mathematical model, the objective function, decision variables and problem constraints are expressed in mathematical expressions. These models provide a deep insight into the structure of the problem [12]; [22]. They are therefore suitable not only for solving the problem using classical methods, but also for solving the problem using heuristic or meta-heuristic methods.

The input data for the mathematical model are described in Table 1. Also, this table explains the decision variables existing in the proposed model. In this section, the formulation of the problem constraints and the objective function of the proposed model are presented in detail.

[1] https://ftp.ncbi.nlm.nih.gov/genomes/refseq/archaea/

4.1. Required memory for running jobs

This constraint states that the memory limitation of kth node must be met. This constraint must be done for all jobs and computing nodes.

$$mem_k \geq f_{mem}(\sum_{i=1}^{n} S_i \cdot y_{ij} \cdot x_{jk}) \quad \forall j \in \{1,2,...,J\}, \forall k \in \{1,2,...,v\} \tag{1}$$

For each J_j and CN_k, $\sum_{i=1}^{n} S_i \cdot y_{ij} \cdot x_{jk}$ denotes the input size of J_j on CN_k and f_{mem} estimates the memory required for J_j.

4.2. Assigning jobs to cluster nodes

Constraints (2) and (3) imply that non-empty jobs must be assigned to exactly one node. Constrain (4) enforces that empty jobs cannot be assigned to any node.

$$\frac{1}{n}\sum_{i=1}^{n} y_{ij} \leq \sum_{k=1}^{v} x_{jk} \qquad \forall j \in \{1,2,...,J\} \tag{2}$$

$$\sum_{k=1}^{v} x_{jk} \leq 1 \qquad \forall j \in \{1,2,...,J\} \tag{3}$$

$$\sum_{k=1}^{v} x_{jk} \leq \sum_{i=1}^{n} y_{ij} \qquad \forall j \in \{1,2,...,J\} \tag{4}$$

4.3. Objective function

For constructing the objective function the following points should be highlighted:

- The objective function deals with the runtime of some bags of jobs on some computing nodes.
- If t_k (Eq. (5))denotes for the above mentioned time for CN_k then the total runtime (makespan) equals to $\max(t_k) \quad 1 \leq k \leq v$. The objective function aims to minimize this time (Eq. (6)).
- When more than one task is assigned to a node, the node wastes a certain amount of time between executing two jobs. $\sum_{j=1}^{J}(x_{jk} - 1)$ denotes that time.
- For each CN_k, $\sum_{j=1}^{J} f_k(\sum_{i=1}^{n} y_{ij} \cdot s_i \cdot x_{jk})$ calculates the summation of the runtime of jobs assigned to CN_k. $\sum_{i=1}^{n} y_{ij} \cdot s_i \cdot x_{jk}$ is the input size of J_j , in which, $f_k()$ denotes an implicit function of that.

Runtime of jobs on each CN_k is calculated by Eq.5.

$$t_k = (\sum_{j=1}^{J} x_{jk} - 1) \cdot st_k + (\sum_{j=1}^{J} f_k(\sum_{i=1}^{n} y_{ij} \cdot s_i \cdot x_{jk})) \tag{5}$$

Therefore, the objective function of the model is as follows:

$$\min(\max(t_k)) \quad 1 \leq k \leq v \tag{6}$$

This can be expressed as follows:

$$\begin{aligned} &\min\ \alpha \\ &\text{subject to:} \\ &\alpha \geq t_k \quad \forall k \in \{1,2,...,v\} \\ &\alpha \geq 0 \end{aligned}$$

Table 1
Models parameters/variables and their description.

Notation of parameters	Description
$CN = \{node1, node2, ..., node_v\}$	Cluster node set
v	The number of nodes of the cluster
CN_k	kth node of the cluster
mem_k	Accessible memory size of CN_k for running tasks
st_k	Switch time between two jobs for CN_k
$Gen = \{g1, g2, ..., g_n\}$	The set of genomes of the reference genome
n	The number of genomes of the reference genome
g_i	i_{th} genome of the reference genome
S_i	Size of g_i
$Job = \{J1, J2, ..., J_J\}$	The set of jobs
J_j	j_{th} job
J	The number of jobs
Notation of decision variables	**Description**
x_{jk}	1 iff J_j is assigned to CN_k, otherwise 0
y_{ij}	1 iff g_i is binned to J_j, otherwise 0

5. Genetic optimization for solving the proposed model

Suppose there is a reference genome of a certain size with genome files $g_1, g_2, \ldots, g_n$. We want to group the genomes into a number of jobs and then assign the jobs to computing nodes $CN_1, \ldots, CN_v$ of a cluster infrastructure.

It can be seen that the proposed model is a non-linear binary mathematical model. Due to the special structure of the constraints and the objective function of this model, linearizing it leads to a binary linear model with a significantly large number of constraints. Solving this model is very time consuming in terms of computation (it may even be impossible). On the other hand, in contrast to classic approaches, using genetic algorithms is a very powerful approach for treating discret models even if the model is nonlinear [23]. Based on this fact, using genetic approach can be an efficient approach for solving the proposed model without any linearization. In the following, we explain how to implement the presented model using a Genetic Algorithm (GA).

5.1. Chromosome structure

GAs mimic optimization during optimization by modelling genetic recombination and a fitness function. Hence, when using GA to solve a particular problem, the first concern to be addressed revolves around the determination of a suitable chromosome coding. Each chromosome represents the different parameters that characterize a solution to the problem. [24]. In the solution, we consider a population of individuals, each individual being a potential solution to the problem described by the individual chromosome. The initial population is generated at random.

In this study, job granularity involves determining the assignment of genome files to jobs and the assignment of jobs to cluster nodes. Thus, a solution (chromosome) is a two-dimensional array in which the indices indicate the genome file number. The elements of the first row contain the job numbers and the elements of the second row contain the computing node numbers. The representation of a chromosome in the GA implementation is shown in Fig. 3. As Fig. 3 shows g_1 and g_3 are assigned to J_1 while g_2 is assigned to J_4. Moreover, J_1 and J_4 are scheduled to CN_3 and CN_2, respectively.

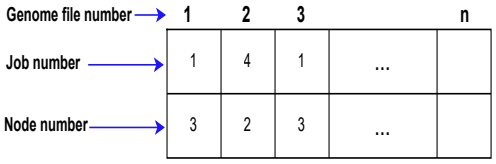


Figure 3: A chromosome encoding.

5.2. GA operators

The crossover operator can help to inherit some chromosome fragments of excellent individuals to subsequent generations. In this study, the single-point crossover technique [23] is adapted during the performing of the crossover operator to produce new individuals. These new individuals are then assessed for their potential to contribute to the next generation of the population. An example is shown in Fig. 4. After crossover, the first new individual is not feasible to add to the next generation because J_5 has been assigned to two different computing nodes.

The mutation operator is a technique that replaces some gene values with others to increase population diversity. We use swap mutation [25] to explore new regions of the solution space, where two positions on a chromosome are randomly selected and swapped. After mutation, the potential of new individuals to contribute to the next generation of the population is evaluated.

6. JOB RUNTIME ESTIMATION

In the GA algorithm, each individual should be assigned a value of the fitness score, which is shown in the objective function defined in Eq.5. Suppose the runtime of job_j on node CN_k is equal to t_j. The challenge here is that there is no explicit function for calculating t_j; in other words, how

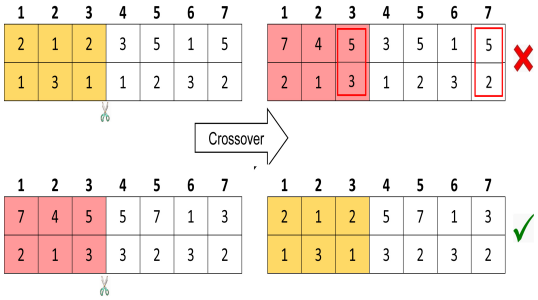


Figure 4: An example of crossover.

long does it take to execute a job_j of the size S_j on a computing node CN_k? Therefore, we use the following different methods to predict the runtime of jobs on computing nodes, assuming that we have historical data from task execution traces:

- **Linear Regression**: Linear regression is a popular and simple machine learning algorithm that models the relationship between dependent and independent variables by analyzing and learning from current training results using a linear expression of independent variables [26].
- **Polynomial approximation**: According to mathematical theorems, any continuous function can be optimally approximated by a suitable polynomial [27]. Based on this theory, a polynomial of appropriate degree can be used to approximate the unknown function using the given historical data. In practice, lower degree polynomials are considered to avoid the socialisation of higher degree polynomials. Here we only consider degrees one, two and three.
- **Logarithmic of a polynomial approximation (log-pol)**: As shown in [28], the curves of runtime according to data input size are logarithmic like and on the other hand, as mentioned above, the polynomials are a suitable method for estimation, so we use a combination of logarithmic and polynomial as a new approximation method.

These methods assume that there is a relationship between a task's input size and its runtime, using the input size as the independent variable. Thus, they can be used to predict task runtime for any task input sizes. Similar to related work, these methods use the size of the file on the hard disk as an input to their prediction models. In a heterogeneous cluster, a same task may have different run times on different computing nodes. Therefore, the methods create their prediction models for each computing node.

7. Experimental results

We developed a read-mapping workflow [2] for metagenomic data in the popular workflow engine Snakemake [29]. The workflow was run on Allegro [3], a cluster infrastructure. We created a historical dataset from traces of the workflow execution on the cluster as a historical execution trace.

We selected and used three reference genomes with different sizes (small, medium and large) as input data of the workflow to perform the experiments. The specifications of the input data are described in Table 2. We also looked into different cluster sizes of 4, 8, 16, and 24 computing nodes in the experiments.

As previously stated, our approach consider the job granularity and scheduling problems simultaneously for a bag of task in a workflow while related works have addressed only the scheduling problem; Indeed, they have assumed that there are a number of jobs, each with a certain size, and they schedule these jobs to the cluster nodes so that the total runtime is minimized.

Most existing approaches use the file size on disk as the input for their predictions or approximate models. In [30] is a detailed discussion of why uncompressed input data size for compressed files should be used. Accordingly, we use the uncompressed file size to predict the runtime and the used memory of jobs.

Table 2
Reference genomes specifications.

Reference Genome	Data Size	# Genome files
Archaea	1.4 GiB	488
Bacteria_1 [4]	30 GiB	7167
Bacteria_2 [5]	88 GiB	22185

7.1. Accuracy comparison of job runtime estimation methods

We compare the accuracy of the methods used to estimate job runtime (See Section 6). The linear regression was implemented using the sklearn.linear_model[6] library. The polynomial and log-pol have been implemented using polyfit() from the Numpy[7] library and Curve_fit() from the Scipy.optimize[8] library, respectively. For comparison we use the Mean Absolute Percentage Error (MAPE). This metric is calculated using the Eq. 7 where A_i is the actual value, F_i is the predicted value and n is the number of fitted points.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{A_i - F_i}{A_i}\right| \quad (7)$$

Figs. 5-7 show results for each input data separately with different cluster sizes. As shown in these figures, the polynomial method gives a more accurate estimate of the values of job run times than log-pol. Log-pol also outperforms linear regression.

7.2. Changing the number of jobs to improve makespan

One of the main challenges in this problem is that the number of jobs is not known in advance. The most obvious idea is to consider this number as equal to the number of genome files, i.e. n. This seems logical at first sight, since the possibility to consider empty jobs allows to potentially find any possible clustering for genome files in the form of jobs. However, from a practical point of view, this is inappropriate and impossible in most cases, because due to the large number of genome files, the number of decision variables and constraints increases significantly, making it impossible to solve

[2] https://github.com/CRC-FONDA/A2-job-granularity/tree/main/MG-HIBF
[3] https://www.mi.fu-berlin.de/w/Cluster/WebHome
[6] https://scikit-learn.org/stable/modules/linear_model.html
[7] https://numpy.org/doc/stable/reference/generated/numpy.polyfit.html
[8] https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.curve_fit.html

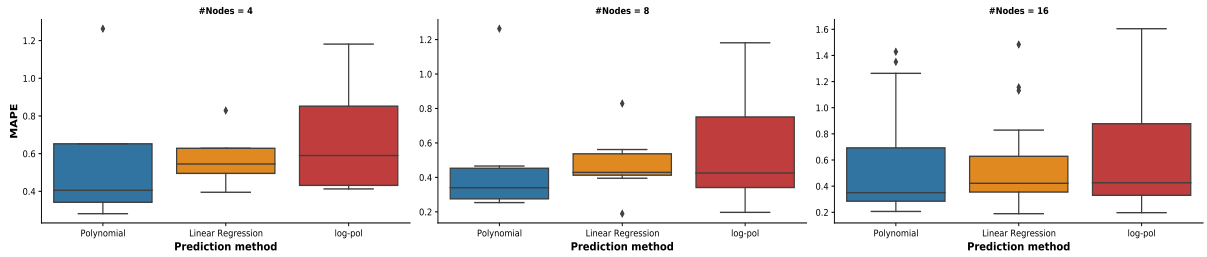


Figure 5: Accuracy of different prediction methods for Archaea.

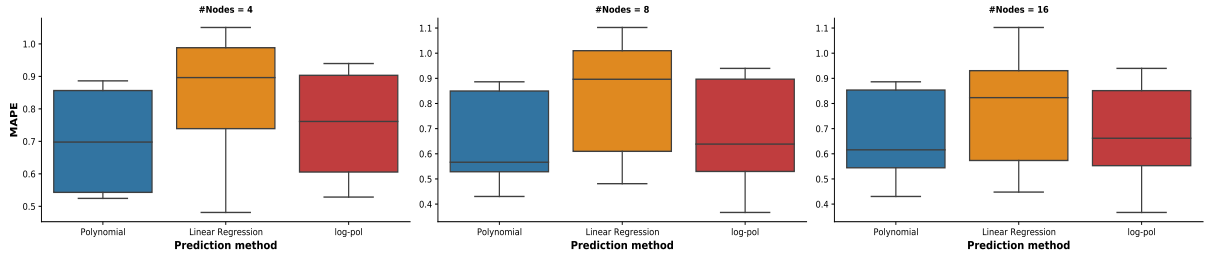


Figure 6: Accuracy of different prediction methods for Bacteria30G.

the problem in a reasonable time, and even if the problem is solved, the propagation of computational errors leads to incorrect and unreasonable results.

Considering that jobs are supposed to be assigned to nodes, another idea is to consider the number of jobs as a factor of the number of nodes, i.e. v; more precisely:

$$J = kv \quad for\ some\ \ k \in \mathbb{N} \tag{8}$$

The coefficient k in the Eq. 8 changes interactively and evolutionarily from one to higher; that is, after solving the problem for k=1 and calculating the makespan, we consider the resulting solution as an initial solution (an individual of the initial population) for k=2. This procedure continues and evolves until the distance between two consecutive makespan values is negligible or insignificant from the decision maker's point of view. On the one hand, this process is compatible with the evolutionary nature of the GA used to solve the above sequential problems, because in each step, with the solution of the previous step as the initial solution, the value of the fitness function in the current step starts to improve from the value of the previous step. Therefore, the makespan value in each step is better than or equal to the previous step (i.e., it evolves). On the other hand, given the stopping condition of the procedure, there is no need to solve problems with large number of jobs.

From the experimental results presented in Tables 3 to 5, the following points can be highlighted:

- As the number of jobs increases, the makespan and the number of unused nodes decrease.
- As the number of nodes increases, the makespan decreases.
- The procedure of increasing the number of jobs may be stopped for two reasons: Firstly, increasing the number of jobs does not improve the makespan (significantly). Secondly, increasing the number of jobs may be stopped by decision maker (especially if no significant improvement in makespan is expected).
- As can be seen in the fourth row of Table 3, the makespan does not improve as the number of jobs increases from v to $2v$. Obviously, a higher number of jobs does not improve the makespan (such cases are marked in bold in the tables). Thus, we have

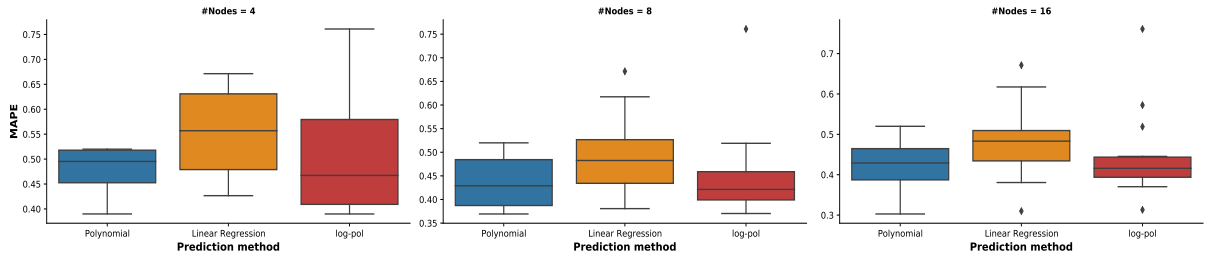


Figure 7: Accuracy of different prediction methods for Bacteria88G.

Table 3
Obtained results for Archaea.

	$J = v$		$J = 2v$		$J = 3v$		$J = 4v$	
#Nodes	Makespan	#Unused nodes	Makespan	#Unused nodes	Makespan	#Unused nodes	Makespan	#Unused nodes
4	539	2	**404**	0	**404**	0	-	-
8	339	1	334	0	**312**	0	**312**	0
16	285	5	252	0	**251**	0	**251**	0
24	**206**	5	**206**	3	-	-	-	-

Table 4
Obtained results for Bacteria-30GiB.

	$J = v$		$J = 2v$		$J = 3v$		$J = 4v$	
#Nodes	Makespan	#Unused nodes	Makespan	#Unused nodes	Makespan	#Unused nodes	Makespan	#Unused nodes
4	5804	1	5124	0	**5077**	0	**5077**	0
8	3050	1	2957	0	**2952***	0	-	-
16	**1937**	3	**1937**	3	-	-	-	-
24	2016	7	**1859**	1	**1859**	0	-	-

not calculated them. Moreover, in the last row of Table 5, increasing the number of jobs from $3v$ to $4v$ only leads to 1% decrease in the makespan, which is not significant (such cases are marked in the tables in bold and with an asterisk). So we stopped the procedure. It should be noted that the DM may stop the procedure when the number of jobs is $3v$ due to the insignificant improvement in makespan and the abandonment of an unused node.

8. Conclusion and Future works

In this paper, an approach to the task/job granularity problem for metagenomic DAWs in cluster infrastructures with makespan minimization was proposed. The problem was first formulated as a mathematical model and then the proposed model was solved using the GA method. One of the main challenges in this problem is that the number of jobs is not known in advance. We overcame this challenge by adjusting the number of jobs as a factor of the number of computing nodes. For each increase in the number of jobs, the makespan is calculated. This procedure continues and evolves until the distance between two successive makespan values is negligible or insignificant from the decision maker's point of view. Experimental results showed that a desirable makespan value can be obtained after a few steps of increasing the number of jobs. Furthermore, the calculation of makespan requires a proper estimation of the task runtime, so we applied three different methods for this estimation. Experimental results showed that the polynomial approximation outperforms.

In the future, we aim to generalize our proposed model so that it can be applied to other scientific domains. Since the proposed approach does not schedule the workflow, but optimizes a single step of the workflow, we intend to integrate it into a scheduling approach in the future work.

Acknowledgments

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) as FONDA (Project 414984028, SFB 1404).

References

S. Mohammadi, L. PourKarimi, H. Pedram, Integer linear programming-based multi-objective scheduling for scientific workflows in multi-cloud environments, The Journal of Supercomputing 75 (2019) 6683–6709.

S. Abdi, L. PourKarimi, M. Ahmadi, F. Zargari, Cost minimization for deadline-constrained bag-of-tasks applications in federated hybrid clouds, Future Generation Computer Systems 71 (2017) 113–128.

J. Bader, F. Lehmann, L. Thamsen, U. Leser, O. Kao, Lotaru : Locally predicting workflow task runtimes for resource management on heterogeneous infrastructures, Future Generation Computer Systems 150 (2024) 171–185. URL: https://doi.org/10.1016/j.future.2023.08.022. doi:10.1016/j.future.2023.08.022.

R. da Silva, A.-C. Orgerie, H. Casanova, R. Tanaka, E. Deelman, F. Suter, Accurately simulating energy consumption of I/O-intensive scientific workflows, in: Computational Science–ICCS 2019: 19th International Conference, Faro, Portugal, June 12–14, 2019, Proceedings, Part I 19, Springer, 2019, pp. 138–152.

R. F. da Silva, H. Casanova, A.-C. Orgerie, R. Tanaka, E. Deelman, F. Suter, Characterizing, modeling, and accurately simulating power and energy consumption of i/o-intensive scientific workflows, Journal of computational science 44 (2020) 101157.

P. Maechling, E. Deelman, L. Zhao, R. Graves, G. Mehta, N. Gupta, J. Mehringer, C. Kesselman, S. Callaghan, D. Okaya, SCEC CyberShake workflows—automating probabilistic seismic hazard analysis calculations, in: Workflows for e-Science, Springer, 2007, pp. 143–163.

A. M. Kintsakis, F. E. Psomopoulos, P. A. Mitkas, Data-aware optimization of bioinformatics workflows in hybrid clouds, Journal of Big Data 3 (2016) 1–26.

S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M.-H. Su, K. Vahi, Characterization of scientific workflows, in: Workflows in Support of Large-Scale Science, 2008. WORKS 2008. Third Workshop on, IEEE, 2008, pp. 1–10.

L. B. Costa, H. Yang, E. Vairavanathan, A. Barros, K. Maheshwari, G. Fedak, D. Katz, M. Wilde, M. Ripeanu,

Table 5
Obtained results for Bacteria-88GiB.

	$J = v$		$J = 2v$		$J = 3v$		$J = 4v$	
#Nodes	Makespan	#Unused nodes	Makespan	#Unused nodes	Makespan	#Unused nodes	Makespan	#Unused nodes
4	**18965**	0	**18965**	0	-	-		
8	14673	1	11432	0	11119	0	**10341***	0
16	7821	4	7385	1	**6127**	0	**6127**	0
24	6078	6	5487	2	4911	1	**4849***	0

S. Al-Kiswany, The case for workflow-aware storage: An opportunity study, Journal of Grid Computing 13 (2015) 95–113.

E. Vairavanathan, S. Al-Kiswany, L. B. Costa, Z. Zhang, D. S. Katz, M. Wilde, M. Ripeanu, A workflow-aware storage system: An opportunity study, in: 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012), IEEE, 2012, pp. 326–334.

J. M. Wozniak, M. Wilde, Case studies in storage access by loosely coupled petascale applications, in: Proceedings of the 4th Annual Workshop on Petascale Data Storage, 2009, pp. 16–20.

S. Mohammadi, H. Pedram, L. PourKarimi, Integer linear programming-based cost optimization for scheduling scientific workflows in multi-cloud environments, Journal of Supercomputing 74 (2018) 4717–4745. doi:10.1007/s11227-018-2465-8.

H. Topcuoglu, S. Hariri, M.-Y. Wu, Performance-effective and low-complexity task scheduling for heterogeneous computing, IEEE transactions on parallel and distributed systems 13 (2002) 260–274.

S. Mohammadi, L. PourKarimi, F. Droop, N. De Mecquenem, U. Leser, K. Reinert, A mathematical programming approach for resource allocation of data analysis workflows on heterogeneous clusters, The Journal of Supercomputing (2023) 1–30.

T. Dziok, K. Figiela, M. Malawski, Adaptive multi-level workflow scheduling with uncertain task estimates, in: Parallel Processing and Applied Mathematics: 11th International Conference, PPAM 2015, Krakow, Poland, September 6-9, 2015. Revised Selected Papers, Part II, Springer, 2016, pp. 90–100.

R. F. Da Silva, G. Juve, E. Deelman, T. Glatard, F. Desprez, D. Thain, B. Tovar, M. Livny, Toward fine-grained online task characteristics estimation in scientific workflows, in: Proceedings of the 8th workshop on workflows in support of large-scale science, 2013, pp. 58–67.

R. F. Da Silva, G. Juve, M. Rynge, E. Deelman, M. Livny, Online task resource consumption prediction for scientific workflows, Parallel Processing Letters 25 (2015) 1541003.

F. Nadeem, D. Alghazzawi, A. Mashat, K. Fakeeh, A. Almalaise, H. Hagras, Modeling and predicting execution time of scientific workflows in the grid using radial basis function neural network, Cluster Computing 20 (2017) 2805–2819.

S. M. Sadjadi, S. Shimizu, J. Figueroa, R. Rangaswami, J. Delgado, H. Duran, X. J. Collazo-Mojica, A modeling approach for estimating execution time of long-running scientific applications, in: 2008 IEEE international symposium on parallel and distributed processing, IEEE, 2008, pp. 1–8.

M. H. Hilman, M. A. Rodriguez, R. Buyya, Task runtime prediction in scientific workflows using an online incremental learning approach, in: 2018 IEEE/ACM 11th International Conference on Utility and Cloud Computing (UCC), IEEE, 2018, pp. 93–102.

K. Kianfar, C. Pockrandt, B. Torkamandi, H. Luo, K. Reinert, Optimum Search Schemes for approximate string matching using bidirectional FM-index, arXiv preprint arXiv:1711.02035 (2017).

S. Abdi, M. Ashjaei, S. Mubeen, Cognitive and Time Predictable Task Scheduling in Edge-cloud Federation, in: 2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA), IEEE, 2022, pp. 1–4.

S. Katoch, S. S. Chauhan, V. Kumar, A review on genetic algorithm: past, present, and future, Multimedia tools and applications 80 (2021) 8091–8126.

G. Tremblay, R. Sabourin, P. Maupin, Optimizing nearest neighbour in random subspaces using a multi-objective genetic algorithm, in: Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004., volume 1, IEEE, 2004, pp. 208–211.

I. De Falco, A. Della Cioppa, E. Tarantino, Mutation-based genetic algorithm: performance evaluation, Applied Soft Computing 1 (2002) 285–299.

T. M. H. Hope, Linear regression, in: Machine Learning, Elsevier, 2020, pp. 67–81.

W. Rudin, Principles of Real Analysis. Mathematics Series, 1976.

C. Valdes, V. Stebliankin, G. Narasimhan, Large scale microbiome profiling in the cloud, Bioinformatics 35 (2019) i13—-i22.

J. Köster, S. Rahmann, Snakemake—a scalable bioinformatics workflow engine, Bioinformatics 28 (2012) 2520–2522.

J. Bader, F. Lehmann, L. Thamsen, J. Will, U. Leser, O. Kao, Lotaru: Locally Estimating Runtimes of Scientific Workflow Tasks in Heterogeneous Clusters, arXiv preprint arXiv:2205.11181 (2022).