Natural Computing (2023) 22:615–630
https://doi.org/10.1007/s11047-023-09950-5

Precise makespan optimization via hybrid genetic algorithm for scientific workflow scheduling problem

Pablo Barredo[1] · Jorge Puente[1]

Accepted: 8 June 2023 / Published online: 8 July 2023

Abstract
Task scheduling in scientific workflows represents an NP-hard problem due to the number of interdependent tasks, data transfers, and the possible execution infrastructure assignments in cloud computing. For this reason, metaheuristics are one of the most widely applied optimisation techniques. Makespan is one of the main objectives in this problem. However, this metric needs to be complemented with a quality measure with respect to the actual execution time in order to avoid incurring more costs than expected by using an over-optimistic approximation. This research applies a new enhanced disk-network-computing evaluation model, that takes into account the communication among the storage devices involved, which plays an important role in actual schedules. The model is implemented in a genetic algorithm and the well-known heuristic HEFT. We propose different hybridisation metaheuristics in conjunction with a new accuracy metric to measure the difference between the makespan approximations and the real one. The new evaluation model is able to improve accuracy with respect to the standard model, and the proposed hybrid methods significantly improve makespan in the case of heterogeneous infrastructures.

Keywords Workflow scheduling · Hybrid genetic algorithm · Cloud computing · Evolutionary algorithms · Makespan accuracy

1 Introduction

The Cloud has enabled a paradigm shift for researchers by allowing access to a fully scalable, on-demand infrastructure. There are several service options offered by providers, but the most notorious are Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). The SaaS model is a fully managed Solution offered by the provider, Office 365 is one of the main examples, in which the client receives a software where the only concern is the usage of the application. In the PaaS model, the client pays for an environment ready for the deployment of applications managed by the client, for example a Web application, where the cloud provider offers a web server, a database server and a backend. The third model is IaaS, where the product is the ability to rent virtual machines (VMs) with different capabilities and the ability to scale massively both horizontally (number of VMs) and vertically (the capabilities of a single server). The payment model is to pay only for the computation resources (CPU and RAM) and the customer has the option to pay per unit of time. Most providers also charge for network and storage usage on a pay-per-use basis. Examples include Microsoft Azure, Google Compute Engine and Amazon EC2. This model gives the user more control by allowing them to choose a specific operating system (OS), CPU, memory, bandwidth and data storage.

This research will focus on the use of the IaaS model as it allows the researcher to create an infrastructure that can be matched to the requirements of the workflow being executed. Furthermore, it allows the researcher to have a different infrastructure for each specific execution of the different workflows.

Scientific workflows represent a complex scenario with up to thousands of interdependent tasks that need to be mapped to a myriad of possible hosts. Workflows are typically represented using a Direct Acyclic Graph (DAG).

✉ Jorge Puente
puente@uniovi.es

[1] Department of Computer Science, University of Oviedo, Campus de Viesques s/n, 33271 Gijón, Spain

Each task takes time to execute depending on the VM selected. The storage unit and network connection between VMs can also have a significant impact on the final execution time. The schedule is conditioned by a number of quality of service (QoS) constraints. Typically, the researcher will want to consider the completion time of the DAG (makespan) or the cost, among other constraints. This mapping of workflow tasks to VMs is known as the *Workflow Scheduling Problem* and is considered an NP-complete problem (Madni et al. 2017).

In the last decade, we can find in the literature numerous methods that try to solve this problem. This is the case of heuristic methods, such as the Heterogeneous Earliest Finish Time (HEFT) heuristic proposed in Topcuoglu et al. (2002), which provide a good balance between execution time and performance, or MOHEFT, proposed by Durillo and Prodan (2014), which is a multi-objective version. However, for more complex workflows, metaheuristic approximations are commonly used. Different bio-inspired algorithms have been proposed, some of them based on physical processes, this is the case of Yuan et al. (2021), where Simulated Annealing is applied in a multi-objective approach, Biswas et al. (2019) propose the use of a Gravitational Search Algorithm (GSA), or Adhikari and Amgoth (2019) apply an Intelligent Water Drop Algorithm that optimises the makespan and the infrastructure involved. There are also hybrid approaches, for example Elaziz et al. (2019), propose a hybridisation of the Moth Search Algorithm (MSA) and the Differential Evolution (DE), or Ye et al. (2019), where a heuristic and a genetic algorithm are combined to minimise the execution time of the workflow.

A significant number of approaches seek to minimise makespan because it is one of the most important goals for scientists, but we have found that their makespan estimates often do not receive the attention they deserve. Inaccurate makespan values can have a negative impact if the scientist take them as a source of truth. An algorithm may generate a good schedule in real execution, but it is essential that it does not deviate from the predicted makespan; a failure to meet time constraints in a cloud computing scenario can generate unexpected costs if the algorithm is over-optimistic. To reduce this gap, the computational model must avoid some simplifications; one commonly found is the omission of disk communications when only network transfers are considered. This inclusion can have a large impact, making the model more accurate in workflows where communication represents a significant part of the execution time.

Traditionally, the network has been the bottleneck in any intensive data transfer. However, as technology has improved (e.g. self-balancing link aggregation of multiple network adapters or fibre connections), this is no longer a technical limitation but a budgetary issue. This is no the case of public cloud providers such as Google, Amazon and Microsoft which offer high-tech data centres that are constantly upgrading their infrastructure in terms of processors/networking/storage devices due to the competitiveness of the market.

For instance, Fig. 1 illustrates the comparison of network bandwidth and maximum disk speeds related to the Google Cloud C2 machine series, a set of virtual machine models designed for compute-intensive workloads, and the available range of virtual storage devices and services in this Cloud provider (Google 2023). As we can see, in almost all configurations the bottleneck is storage speed and not network bandwidth.

The model considered in this paper is the Disk-Network-Communication (DNC) evaluation model previously introduced in Barredo and Puente (2022), which is designed to take into account storage communication times in workflow tasks.

In the current paper we will develop an extended study of this model, looking to evaluate its accuracy improvements in both data-intensive and compute-intensive workflow problems, and its contribution to makespan optimisation. We proposed some hybrid metaheuristics combining elements of the HEFT heuristic and a Genetic Algorithm, and contrast their results with those of the previous standard Network-Computing (NC) evaluation model.

The main contributions of the current paper are to:

- Reformulate of HEFT heuristic and a lamarckian GA algorithm in terms of the new DNC evaluation model.
- Propose different hybrid algorithms combining the GA and both components and heuristic solutions of the well-known HEFT heuristic, to minimize makespan.
- Introduction of an accuracy metric to study the similitude of estimated makespan with respect to real execution times, based on simulations using Wrench framework (Casanova et al. 2020).
- Design of two different cloud computing scenarios by varying infrastructure sizes and specs, and solving instances of seven real scientific workflow problems from WFCommons repository (Coleman et al. 2022).
- Experimental study covering seven different real scientific workflow applications. All workflow execution instances were generated from real Pegasus WMS (workflow management system) executions, and available in WFCommons repository. The study covers estimated makespans accuracy of DNC vs NC model, and real makespan improvements of hybrid proposals with respect to standard HEFT heuristic and the proposed lamarckian genetic algorithm.

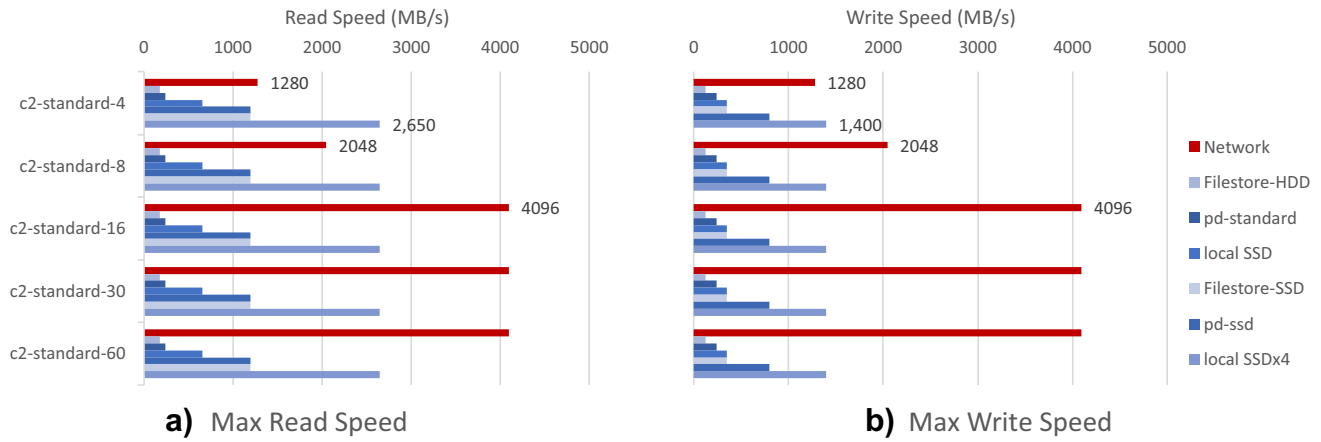

Fig. 1 Read and Write transfer speeds of different options of storage in Google Cloud infrastructure, compared against the network bandwidth in its C2 machine series

The remainder of the paper is organised as follows. In the next section, we give the formulation of the Scientific Workflow Scheduling model. The proposed solving methods are described in Sect. 3. In Sect. 4, we report the results of the experimental study. Finally, in Sect. 5, we summarise the main conclusions and outline some ideas for future work.

2 The scientific workflow scheduling model

Computational applications in distributed systems are workflows modelled as a set of tasks interconnected by precedence constraints. The execution of a task can be initiated as soon as the necessary data are available, i.e. after all predecessors have been completed. After execution, every task generates an output dataset. This dataset is required by its successor tasks in the workflow before their execution. Most workflow applications can be represented in the form of a direct acyclic graph where the nodes are the tasks, and the arcs are the precedence constraints. This is the case for multiple scientific applications in very different research fields such as bioinformatics (1000genome, Epigenomics, SoyKB, SRA Search), agroecosystem (Cycles), Seismology (Seismic Cross Correlation) or astronomy (Montage) (Juve et al. 2013).

2.1 Definition of the workflow scheduling problem

Scientific workflows are represented as a direct acyclic graph $G = (T, A)$, where $T = \{t_1, t_2, ..., t_n\}$ is the set of nodes representing the n tasks of the problem, and $A = \{(t_i, t_j) | 1 \leq i \leq n, 1 \leq j \leq n, i \neq j\}$ represents the set of arcs or dependency constraints among the tasks. The nodes are labelled with their corresponding task sizes in *MFLOPs* (Million Floating Point Operations), and the arcs are labelled as $data(i, j)$, i.e., the dataset size in *MB* to transfer from t_i to t_j. Moreover, t_{entry} and t_{end} are fictitious tasks with null computation and communication, representing the entry and exit points of the workflow, respectively.

The resource model consists in a cloud service provider that offers an IaaS platform as a set of mixed types of hosts, or VMs, to its clients. Let M={vm_1, vm_2,...,vm_m} be the set of VMs, each one modelled as a tuple $< pc, nb, ds >$, where pc, nb and ds are the processing capacity in *GFLOPS* (GFLOPs per second), network bandwidth in *MB/s* and disk reading/writing speed in *MB/s* respectively.

Now, given a workflow $G = (T, A)$ and an IaaS infrastructure as a set of VMs M, the goal of the Workflow Scheduling Problem - defined by the tuple (G, M) - is twofold. Firstly, we need to find a feasible solution $S = (Hosts, Order)$ where *Hosts* is a mapping from tasks to VMs and *Order* is a topological order of G. And then, we want this schedule be optimal in the sense that its *makespan* is minimal.

Specifically, the goal is:

$$minimize\ EFT(t_{exit}) \tag{1}$$

where $EFT(t_{exit})$ is the estimated finish time of the task t_{exit}.

The schedule of the tasks and the corresponding estimated value of *makespan* are defined in accordance with the applied evaluation model: the standard NC model or the new DNC model. In the next subsections both models are formally introduced.

2.2 Schedule evaluation models

In the context of metaheuristic optimization, thousands of schedules are generated and subsequently evaluated. Standard evaluation models tend to simplify the real infrastructure factors involved in computational executions

in Cloud or on-premise infrastructures. The main advantage is that their low computational cost allows them to be used as a decoding model in metaheuristic optimization methods. However, the main drawback is the underestimation of the real computational cost in an expensive pay-per-use scenario.

The most common evaluation model for workflow schedules is the one used in Ghorbannia Delavar and Aryan (2014), Durillo and Prodan (2014), Zhu et al. (2016), Chakravarthi et al. (2022), which only considers CPU processing times and data sets network communications between different hosts. This Network-Computation model (NC) ignores all disk accesses from/to tasks in their data sets acquisition/generation. Other common simplification in NC model only considers the latest network communication time from predecessor tasks to establish the starting time estimation of a successor task. For compute-intensive workflows, the NC model may generate quite accurate makespan approximations.

However, in a cloud computing environment (usually a pay-per-use infrastructure environment) not only computing times or networking communications are relevant, but also disk data input/output transference times. This extra time should be considered at least in data-intensive workflows where data transfer operations are not negligible with respect to computing times. In this paper we will focus in a new extended evaluation definition called Disk-Network-Computation model (DNC), proposed in Barredo and Puente (2022), which considers network communications and all disk operations.

To illustrate, we would present in the Fig. 2 a workflow with 3 tasks of 1 TU (time unit) of computation each, said tasks have to transfer 1 DU (data unit) of information. Figure 3 illustrates the execution of the workflow having two hosts: Host A has a disk with a speed of 1 DU per TU and Host B has 0.5 DU per TU. For model a) we only have to consider the network of 1 DU per TU (block $C_{2,3}$), so the makespan is 3 TU; for model b) there are some new concepts, all tasks have to write data to disk (W_Disk_A and W_Disk_B), but host B takes twice the time. Task T3 on host A cannot start reading until all the files have been written, then it can start reading the first file (R_Disk_A), and because the files are read sequentially, the data from T2 has to wait. Another key difference is in the idea of using the slowest medium, the disk of host B is slower than the bandwidth, so the transfer ($C_{2,3}$) takes 2 TU. The final important an often omitted factor is that the current task may need to write its output data to disk, in this case it takes 1 TU, giving a final makespan of 8 TU.

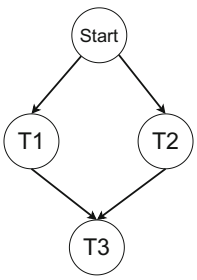

Fig. 2 Workflow describing a 3 task with one fictitious entry task to have one entry point and one endpoint

When we apply an evaluation model to a workflow problem solution, we get the schedule and its corresponding *a priori* or estimated makespan, but it is only after execution of the workflow - which in this work will be done via IaaS simulations - that we get the actual makespan. In Barredo and Puente (2022) a robustness measure was introduced to quantify the difference between a priori makespan estimation and real makespan. In this work we introduce the concept of *accuracy* of a makespan estimation of an schedule s as its similarity degree with respect to the real makespan, which is defined as:

$$accuracy(s) = \frac{makespan_{est}(s)}{makespan_{sim}(s)} \tag{2}$$

where $makespan_{sim}(s)$ is the actual makespan from the simulation and $makespan_{est}(s)$ is the a priori makespan estimated by the scheduler of the solution s. The surrogate model used by the scheduler generates optimistic makespan estimations, consequently, the *accuracy* value must be ≤ 1.

The main contribution of the DNC evaluation model is the improved accuracy of the estimated makespan. An accurate a priori makespan will offer scientists a relevant information to take decisions about the computing infrastructure to rent depending on the budget and the urgency of the results.

2.3 Disk-network-computing vs network-computing processing model

In this section, the DNC model estimations are defined and their differences to the NC model are highlighted. The DNC model differs from standard NC model in communications and processing times, because only DNC model considers disk accesses. The computation of tasks is similar but it is necessary to reformulate the main concepts.

The *makespan* estimation using DNC model is calculated applying the following definitions:

Definition 1 The computation time of task t_i on a machine vm_k, denoted ct_i^k, is defined as:

$$ct_i^k = size(t_i)/pc_k \tag{3}$$

where $size(t_i)$ is the size of the task t_i measured in GFLOPs and pc_k is the processing capacity of the virtual machine vm_k in GFLOPS.

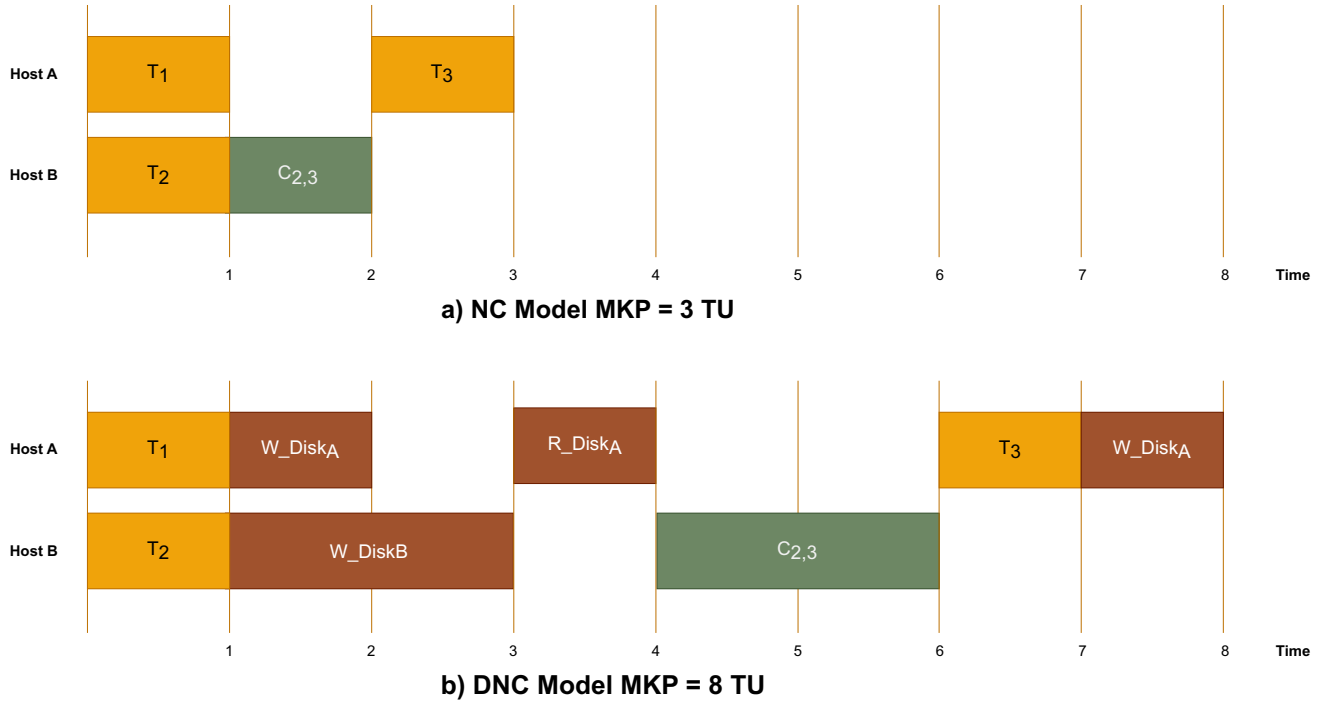

Fig. 3 Processing phases of tasks execution in NC and DNC evaluation models for the workflow in Fig. 2. In **a** NC model, only input data from network transmissions of the last predecessor task to finish are considered; in **b** DNC, on the other hand, we consider all input data transmissions from both network and local disk, and only after the last predecessor task has written its results to local disk. Finally, the current task is not considered finished until data output is written to local disk

Definition 2 The data transfer time between tasks t_i and t_j mapped on vm_k and vm_l respectively is:

$$dt_{i,j}^{k,l} = \begin{cases} \dfrac{data(i,j)}{ds_k} & : k = l \\ \dfrac{data(i,j)}{min(ds_k, nb_k, nb_l)} & : k \neq l \end{cases} \quad (4)$$

where $data(i, j)$ is the output-data size from t_i to t_j, and vm_k and vm_l are the VMs where tasks are scheduled respectively. In this DNC model, when tasks t_i and t_j are scheduled in the same VM, data input files should be read from local disk, ds_k is the disk speed of vm_k, and nb_k and nb_l are the network bandwidths of vm_k and vm_l respectively. In NC model all data transfers from/to disk are ignored, so in $k = l$ case of Eq. 4 the communication time is zero.

Definition 3 The estimated finish time of task t_i executed on machine vm_k involves not only the processing time but also complete input and output data transfer operations. It is defined as:

$$EFT(t_i, vm_k) = EST(t_i, vm_k) + input_{i,k} + ct_i^k + output_{i,k} \quad (5)$$

where $input_{i,k}$ is the communication time for input data of t_i on vm_k from all its predecessors. It is defined as:

$$input_{i,k} = \sum_{t_j \in pred(t_i)} dt_{j,i}^{l,k} \quad (6)$$

where each predecessor task t_j is executed on its corresponding machine vm_l. The corresponding $output_{i,k}$ is the writing time for all output data of t_i in the vm_k local disk, that is:

$$output_{i,k} = \frac{\sum_{t_j \in succ(t_i)} data(i,j)}{ds_k} \quad (7)$$

NC model, in contrast to DNC model, considers only the computation time ct_i^k while all input/output data transference is ignored.

Definition 4 The estimated starting time of task t_i on vm_k is defined as:

$$EST(t_i, vm_k) = avail\left(i, k, \max_{t_j \in pred(t_i)} EFT(t_j, vm_l)\right) \quad (8)$$

where each predecessor task t_j is executed on its corresponding machine vm_l and $avail(i, k, m)$ is the earliest available time slot of vm_k after m to compute t_i.

DNC model use an insertion policy which assigns the earliest idle time slot between two already-scheduled tasks on the assigned VM. The length of the time slot should be

at least capable of cover not only computation but also data transfer times of the considered task. Additionally, scheduling in this idle time slot should preserve precedence constraints.

3 Solving methods

In this work we propose different solving methods. First, a reformulated version of HEFT, one of the most used scheduling heuristics for makespan optimization in scientific workflows. Second, an efficient genetic algorithm based on Barredo and Puente (2022). And finally, we combine both previous methods to generate several hybrid genetic algorithms exploding HEFT's components to improve makespan quality with no loss of accuracy.

3.1 HEFT heuristic with DNC model

The original HEFT was proposed in Topcuoglu et al. (2002), its main idea is to schedule tasks so that the earliest finish time (*EFT*) is minimized for all the tasks. Both phases of HEFT with classic NC model ($HEFT_{NC}$), and HEFT heuristic using new DNC model ($HEFT_{DNC}$) are described as follows:

Phase 1: *Calculating priority of tasks*

In this phase, the priority of each task is calculated using average execution time and average communication time. The priorities are calculated from bottom to up direction. The sequence of tasks will be generated from highest to lowest priority, satisfying all workflow precedence constraints. In $HEFT_{NC}$ the priority of task t_i is given by

$$prioNC(t_i) = \overline{ct_i} + max_{t_j \in succ(t_i)}(\overline{dt_{i,j}} + prioNC(t_j)) \quad (9)$$

where, $\overline{ct_i}$ is the average execution time of task t_i and $\overline{dt_{i,j}}$ is the average communication time between task t_i and tasks t_j. The main difference in $HEFT_{DNC}$ is that the priority of a task includes not only average communication with the highest priority successor task, but with all its successors and predecessors, from Eq. 6 and Eq. 7. The priority is defined as follows:

$$\begin{aligned} prioDNC(t_i) = & \overline{input_i} + \overline{ct_i} + \overline{output_i} \\ & + max_{t_j \in succ(t_i)}(prioDNC(t_j)) \end{aligned} \quad (10)$$

where $\overline{input_i}$ and $\overline{output_i}$ are the average communication times between task t_i and all its predecessors and successors respectively.

Phase 2: *Mapping tasks to VMs*

The actual mapping of tasks to VMs is performed in this phase according to their priority. In $HEFT_{NC}$ the task with the highest priority is scheduled first, by calculating earliest finish time, considering only the computation time, on all available VMs. In contrast $HEFT_{DNC}$, using eq. 5, considers not only computation time but also all input/output data transference for a more realistic *EFT* estimation.

3.2 Genetic algorithm

In this section, we introduce the main components of the genetic algorithm proposed to solve the Workflow Scheduling Problem and study the accuracy of the solutions using both NC and DNC evaluation models. This evolutionary algorithm combines previous algorithms from Barredo and Puente (2022) and Palacios et al. (2015). Algorithm 1 shows a pseudocode of the GA: it is a generational genetic algorithm with random mating selection and replacement by tournament between parents and offspring, which confers the GA an implicit form of elitism. The GA uses one of the two evaluation models: NC or DNC. As a result we will have two different genetic algorithms: NC-GA and DNC-GA respectively. Both algorithms require the following parameters: population size (pop_{size}), number of generations (max_{gens}), crossover and mutation probabilities (p_c and p_m).

Algorithm 1 Genetic Algorithm

Require: A Workflow instance, a VMs infrastructure and parameters (pop_{size}, max_{gens}, p_c and p_m)
Ensure: A schedule for workflow and tasks to VMs assignment
1: Generate a pool P_0 of random solutions. /*initial population*/
2: Evaluate each chromosome of P_0 using DNCevaluator
3: **for** $t \leftarrow 0$ to $\max_{gens}$ **do**
4: **Selection**: organize the chromosomes in P_t pairs at random
5: **Recombination**: make each pair of chromosomes and mutate both offspring in accordance with p_c and p_m
6: **Evaluation**: evaluate offspring chromosomes
7: **Replacement**: make a tournament selection 4:2 among every pair of parents and their offspring to generate P_{t+1}
8: **return** the best generated solution

Coding Schema The coding schema is based on permutations of tasks (Ye et al. 2019; Zhu et al. 2016), each one with a specific VM assignment. So, a gene is a pair (i,k), $1 \leq i \leq |T|$ and $1 \leq k \leq |M|$, and a chromosome includes a gene like this for every task. For example, given an instance with 4 tasks and 2 VMs, a feasible chromosome is the following: chr_1: ((1 2) (4 1) (2 1) (3 2)) which represents the task ordering (t_1, t_4, t_2, t_3) with VMs assignments (vm_2, vm_1, vm_1, vm_2) respectively. We only consider task orders that codify a topological order so every task must be located in a gene after its last predecessor and before its first successor in the chromosome. Therefore, the individuals generated in the initial population and by the genetic operators must be consistent with the task dependencies constraints.

Decoding Schema The schedule represented by a chromosome is calculated following the selected evaluation model as a decoder. The genes are processed from left to right in the chromosome sequence. For each gene (i,k), the task t_i is scheduled at the earliest free gap of vm_k, after the latest finish time of its predecessors in the workflow, where the processing time of task t_i (computation and communications - depending on the evaluation model) fits. The makespan of the built schedule is the latest finish time of all the workflow tasks. In order to accelerate the convergence to optimal solutions, the lamarckian learning (Houck et al. 1996) is considered as the last stage of decodification and evaluation phase. As a result, the gene order of the chromosome is recoded according to the resulting topological order of the generated schedule.

Crossover The mating operator must establish the order and VM assignment of the tasks at the generated offspring. A feasible schedule permutation must follow all the dependencies which exist among tasks. For chromosome mating, we follow Zhu et al. (2016) and the so called CrossoverOrder algorithm. First the operator randomly chooses a crossover position, which splits each parent sequence into two subsequences. After that, the two first substrings are taken to be the initial sequence of the offspring and then filling the remaining positions with the genes representing the remaining tasks taken from the other parent, while keeping their relative order. The resulting task orders will not cause any dependency conflict since the order of any two tasks should have already be present in at least one parent.

Mutation The mutation operator cannot break the task order dependencies. First, we select a random task T_i. Next, we identify all predecessors and successors of T_i. Then the operator locates the longest subsequence of genes holding T_i that doesn't include any predecessor or successor of T_i. Finally, T_i is moved to a randomly chosen location inside this subsequence. Consequently the assigned VM is mutated to a random index in the set of VMs.

Initial Population The pop_{size} initial individuals of the population are generated at random but following a topological order of the tasks, and with valid VM index assignments for all tasks.

We start with an empty chromosome, and then we identify as candidate tasks those that have t_{start} as their only predecessor in the workflow. At every step we extract at random a task T from candidate tasks, this task will be appended at the end of the partial chromosome. Then, we update candidate tasks by adding all the successors of T which have all their predecessors in the chromosome. The process is repeated until the set of candidate tasks gets empty. The resulting chromosome tasks sequence follows a topological order.

The initial assignment of virtual machine k is selected at random for every task T_i in range $1 \leq k \leq |M|$.

3.3 Hybrid genetic algorithms

Although $HEFT_{DNC}$ heuristic and previous genetic algorithm methods bring reasonable quality solutions to the workflow scheduling problem, hybridisation can be applied to improve the quality of their results. The idea is to inject the knowledge of the different two phases of $HEFT_{DNC}$ in the DNC genetic algorithm (GA_{DNC}). As a result, two new different hybrid genetic algorithms are available:

HGA_{Ph1}: in this algorithm the $HEFT_{DNC}$ tasks ranking is used in all schedule generations. Therefore, the task order information is omitted in chromosomes, and in the decodification operation the tasks order is previously fixed and only information about VMs assignment comes from the chromosome.

HGA_{Ph2}: this version gets the task order from the chromosome and assigns the tasks to the VM with the lowest earliest completion time (EFT in eq. 5) of all available VMs. Since the VM assignment information in the chromosome is unnecessary, it is omitted.

A priori one of the advantages of the proposed hybrid algorithms is the reduction of the solution space, so that the same number of evaluated individuals represents a larger explored subspace of solutions. On the other hand, the risk of the injected heuristic information is the convergence bias towards local minima.

Additionally, the heuristic individual generated by $HEFT_{DNC}$ is considered in the initial population of the proposed genetic and hybrid algorithms: GA^H_{DNC}, HGA^H_{Ph1} and HGA^H_{Ph2}. They are all elitist algorithms, so we only need to add one copy to the initial population to guarantee that the quality of the final solution is as high as the $HEFT_{DNC}$ version.

4 Experimental study

This section evaluates the accuracy and quality of solutions from the different solving methods presented in Sect. 3. First, the workflow problems used, the evaluation metrics and the simulation platform are described. Then, the experimental study and its results, using a cloud simulator built on top of the Wrench library (Casanova et al. 2020), are analysed.

4.1 Workflows instances

The scientific workflows instances considered in this study correspond to seven different problems from the WFCommons repository (Coleman et al. 2022). All of them are workflow execution instances generated using Pegasus workflow management system (Deelman et al. 2019). The different problems present diverse characteristics but can be classified in two main types: compute-intensive and data-intensive workflows. In the data-intensive there is another sub-type that can be described as "collision prone" in where the tasks have a huge number of dependencies and can produce exhaustion in the drives and network interfaces. An example of a workflow of the former sub-type can be found in the Fig. 4.

All used workflow problems are now presented accompanied by a brief description:

- *1000Genome* This workflow uses data from the 1000 Genome projects and finds mutational overlaps in order to provide a data for the evaluation of health conditions caused by mutations.
- *Cycles* It is a Agroecosystem model used to simulate the perturbations of biogeochemical process caused by different agricultural practices.
- *Epigenomics* This workflow is related to genome sequencing operations.
- *Montage* It consists in the reprojection and background correction to compose a mosaic using Flexible Image Transport System (FITS) telescope images.
- *Seismology* The workflows represent the process of taking multiple seismic stations and cross-correlating the measurements of acceleration.
- *SoyKB* Is the genomics pipe of re-sequencing the soybean germplasm to change its traits.
- *SRASearch* This workflow is the process of searching the Sequence Read Archive(SRA) and transforming the data to have aligned sequencing reads.

To evaluate the impact of workflow dimensions of each proposed problem we have selected four instances for each problem, having four sizes: extra-small (50–100 task),

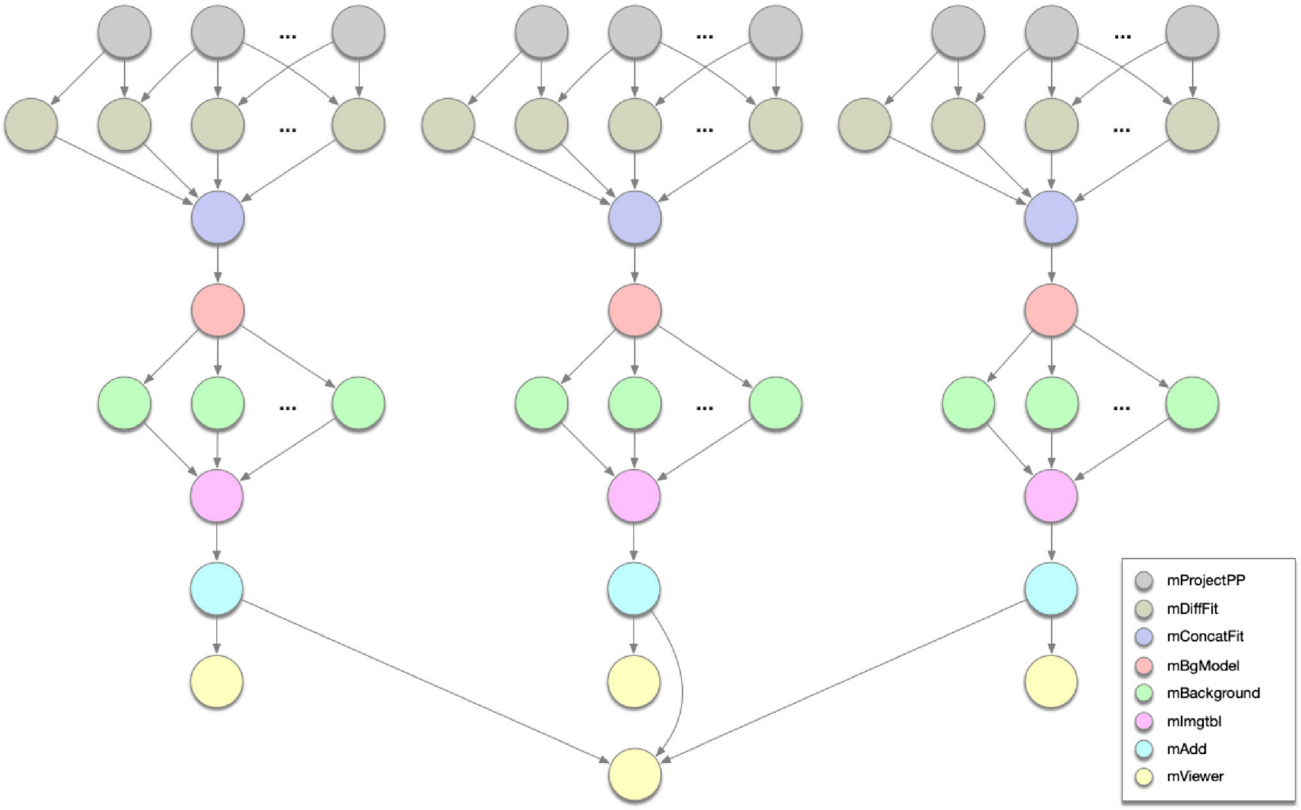

Fig. 4 Diagram of Montage Scientific Workflow (source: wfcommons.org)

Table 1 Analysis of ccr for every instance grouped by problem

Problem	Type	Instance	Tasks	ccr (%)
Seismology	Data	Seismology-chameleon-100p-001	101	0.02
		Seismology-chameleon-500p-001	501	0.02
		Seismology-chameleon-700p-001	701	0.02
		Seismology-chameleon-1000p-001	1001	0.02
Cycles	Compute	Cycles-chameleon-1l-1c-9p-001	67	0.06
		Cycles-chameleon-2l-1c-12p-001	437	0.03
		Cycles-chameleon-2l-1c-9p-001	133	0.04
		Cycles-chameleon-5l-1c-12p-001	1091	0.06
Epigenomics	Data	Epigenomics-chameleon-hep-1seq-100k-001	41	1.55
		Epigenomics-chameleon-ilmn-1seq-100k-001	125	1.13
		Epigenomics-chameleon-hep-6seq-100k-001	507	0.84
		Epigenomics-chameleon-ilmn-6seq-100k-001	863	1.74
SRASearch	Data	Srasearch-chameleon-10a-005	22	2.44
		Srasearch-chameleon-20a-003	42	0.86
		Srasearch-chameleon-40a-003	84	1.41
		Srasearch-chameleon-50a-003	104	1.19
Montage	Compute	Montage-chameleon-2mass-005d-001	58	2.27
		Montage-chameleon-2mass-01d-001	103	3.11
		Montage-chameleon-dss-10d-001	472	2.15
		Montage-chameleon-dss-125d-001	1066	5.09
SoyKB	Data	Soykb-chameleon-10fastq-10ch-001	96	17.24
		Soykb-chameleon-10fastq-20ch-001	156	14.46
		Soykb-chameleon-30fastq-10ch-001	256	15.67
		Soykb-chameleon-40fastq-20ch-001	546	13.31
1000genome	Data	1000genome-chameleon-2ch-250k-001	82	25.53
		1000genome-chameleon-4ch-250k-001	164	19.09
		1000genome-chameleon-12ch-250k-001	492	24.44
		1000genome-chameleon-18ch-250k-001	738	24.14

small (100–200), medium (200–500), and large (5000–1000) if they are available in the repository.

Table 1 contains all instances used in the experimentation followed by the number of tasks and the communication to computation ratio (CCR) as a metric to know the percentage of the total execution spent in communications with respect to computation time (Xu et al. 2014) here adapted to DNC model:

$$CCR = \sum_{i \in T} \frac{(\overline{input_i} + \overline{output_i})}{\overline{ct_i}} \tag{11}$$

where $\overline{input_i}$ and $\overline{output_i}$ are the average communication times between task t_i and all its predecessors and successors respectively, and $\overline{ct_i}$ is the average execution time of task t_i.

4.2 Benchmark platform and experimental configuration

Wrench is a simulator library that can calculate the real makespan of a given workflow on a specific infrastructure. A simulator can provide us a close estimation of metrics like makespan, energy or cost, without the need of buy/hire real hardware. The selected software is a C++ library that can be used to build a custom simulator.

We have simulated a processes/communications system inspired by HTCondor (Thain et al. 2005), the High-Throughput Computing environment under the well-known Pegasus WMS, where each computing host has direct access to a local disk, and remote disks from the rest of hosts are accessible by their network interface connections - such as the NFS service in Linux. Each host has a network interface connected to a virtual switch interconnecting all hosts. Tasks read files from local or remote disk, but always store the output files in the local disk.

The hardware available in a scientific cluster can be quite diverse, whether it is an on-premise cluster or one available in the public cloud, and therefore the proposed experimentation uses different clusters with diverse configurations. For this reason, we have considered both homogeneous and heterogeneous infrastructure configurations.

To study the impact of the new DNC evaluation model we have designed two scenarios: 1) ScFast, where there are hosts with 441 GFLOPS CPU, a disk of 115 MB/s and a network of 125 MB/s. 2) ScMixed, where half of the hosts incorporate a 200 MB/s disk and the other half a 20 MB/s disk, CPUs and network specs are the same of ScFast. For each scenario we have different number of servers going from 1 to 16 in powers of 2. The simulation was run on a Linux computer with the following specs: Intel Core i7-10700k@3.8Ghz, 32GB RAM 3200 MHz, 1TB SSD 2400MB/s.

We made a C++ application based on the GA implemented in Barredo and Puente (2022) that contains several methods for calculating the makespan of a given workflow in a user-defined host infrastructure.

In all the experiments the genetic and hybrid algorithms are configured exactly the same with an initial population of 100 individuals and 1000 generations. The crossover has a probability of 1.0 and the mutation is 0.1.

Each experiment is run 10 times per workflow instance and number of hosts for both scenarios. Each run is validated with the simulator having the real makespan of each individual run.

4.3 Accuracy study

In this section the accuracy of both evaluation models is studied. Table 2 show the behaviour of NC and DNC model using heuristic (HEFT) and metaheuristic (GA) optimization algorithms in both scenarios (ScFast and ScMixed). Problems are presented in ascending ordered of average communication computation rate ($\overline{ccr}$) of its instances. Makespan Accuracy values presented are the worst-case obtained for solutions on all different size instances of the problem over the diverse set of hosts considered.

In NC Model problems identified as compute-intensive in Coleman et al. (2022) (Cycles and Montage) should, a priori, get higher accuracy than the data-intensive ones (1000genome, Seismology, Epigenomics, SRASearch and SoyKB). However, actual results show that when $\overline{ccr}$ ratio increases, meaning a higher communication load on the problem, accuracy decreases notably. In *ScFast* scenery, accuracy drops below 82% (in HEFT) and 81% (in GA) for 1000*genome*, the problem with worst results of the benchmark. This accuracy reduction is even more dramatic in *ScMixed* scenario, where accuracy levels drop below 48% (in HEFT) and 43% (in GA), respectively, again in 1000*genome* problem.

On the other hand, DNC model solves with remarkably high accuracy rates ($\geq 97\%$) in all scenarios and for almost all problems and independently of their $\overline{ccr}$. The only one exception is Montage problem where accuracy drops below of 95%/91% (HEFT) and 96%/93% (GA) in *ScFast*/*ScMixed* scenarios, respectively. The nature of high communications concurrency of Montage could justify these

Table 2 Accuracy of the NC and DNC models. DNC gets superior accuracy in all configurations, as we can observe

Model	Problem ($\overline{ccr}$)	*HEFT*		*GA*	
		ScFast (%)	ScMixed (%)	ScFast (%)	ScMixed (%)
NC	Seismology (0.02%)	99.66	99.16	99.66	99.23
	Cycles (0.05%)	99.59	97.71	99.57	97.71
	Epigenomics (1.31%)	95.47	81.50	95.69	85.49
	SRASearch (1.47%)	96.82	88.34	96.09	85.92
	Montage (3.15%)	86.97	72.07	92.36	78.55
	SoyKB (15.17%)	87.08	54.03	87.08	54.03
	1000genome (23.30%)	81.34	43.29	80.79	42.58
DNC	Seismology	99.84	99.86	99.84	99.87
	Cycles	99.99	99.99	99.99	99.99
	Epigenomics	99.11	99.11	98.87	99.13
	SRASearch	99.40	97.29	99.66	98.19
	Montage	94.90	90.50	95.90	92.98
	SoyKB	100	100	100	100
	1000genome	99.99	99.99	99.99	100

Table 3 Makespan accuracy, worst value found for each problem is shown (in bold best value for each problem), using DNC model, of genetic and hybrid algorithms with random and heuristic initial population in ScFast infrastructure

$\overline{ccr}$ (%)	Problem	HGA_{Ph1} (%)	HGA_{Ph2} (%)	GA^H (%)	HGA^H_{Ph1} (%)	HGA^H_{Ph2} (%)
0.02	Seismology	**99.84**	**99.84**	**99.84**	**99.84**	**99.84**
0.05	Cycles	**99.99**	**99.99**	**99.99**	**99.99**	**99.99**
1.31	Epigenomics	**99.17**	98.81	98.82	98.76	99.04
1.47	SRASearch	**99.50**	99.34	99.39	99.34	99.26
3.15	Montage	**95.08**	94.91	95.04	94.51	94.62
15.17	SoyKB	**100**	**100**	**100**	**100**	**100**
23.30	1000genome	99.99	99.99	**100**	99.99	99.99

Table 4 Makespan accuracy, worst value found for each problem is shown (in bold best value for each problem), using DNC model, of genetic and hybrid algorithms with random and heuristic initial population in ScMixed infrastructure

$\overline{ccr}$ (%)	Problem	HGA_{Ph1} (%)	HGA_{Ph2} (%)	GA^H (%)	HGA^H_{Ph1} (%)	HGA^H_{Ph2} (%)
0.02	Seismology	**99.87**	**99.87**	99.86	**99.87**	99.86
0.05	Cycles	**99.99**	**99.99**	**99.99**	**99.99**	**99.99**
1.31	Epigenomics	**99.28**	99.18	99.11	99.16	99.11
1.47	SRASearch	98.42	**98.47**	97.65	98.44	97.71
3.15	Montage	91.64	**92.64**	91.18	91.40	90.39
15.17	SoyKB	**100**	**100**	**100**	**100**	**100**
23.30	1000genome	99.99	99.99	**100**	**100**	99.99

results because neither of the two models are designed to manage this issue.

These results reinforce the decision of adopting DNC as the evaluation model for scheduling scientific workflows graphs in an IaaS independently of the hardware configuration.

Now, using only DNC Model, Tables 3 and 4 summarise the worst accuracy levels of the different proposed genetic and hybrid algorithms evolving from random and heuristic initial population. The accuracy levels do not vary with respect to the pattern shown in the simple heuristic and genetic algorithms. All values are above 97%, again with the sole exception of Montage, the benchmark's high concurrency problem, where accuracy drops below 91% and 95% in the worst cases of the *ScMixed* and *ScFast* scenarios respectively. In conclusion, the different algorithms presented seem to solve all instances in each scenario studied with high accuracy when using the DNC model, as opposed to the NC model, which gives significantly worse results in heterogeneous infrastructures.

4.4 Makespan optimization study

In this section we study the efficiency of the different proposed algorithms using the new evaluation model to optimise the workflow completion time. Instead of directly comparing makespan, which vary widely with the dimensions of each workflow, we will use *makespan percentage error* (*MPE*) as a performance metric:

$$MPE = \frac{makespan_{HEFT_{NC}} - makespan}{makespan_{HEFT_{NC}}} \quad (12)$$

makespan refers to the solution completion time to be evaluated, and the denominator is the makespan of the solution obtained from the HEFT heuristic using the previous NC model as a quality baseline, which is the computational most inexpensive, but not necessarily least accurate, of the studied methods in this work. Positive values of *MPE* mean better quality solutions, and negative values mean worse results, both with respect to HEFT. Tables 5, 6, 7, 8 and 9 show average *MPE* results for all instances of each problem and infrastructure scenario.

Firstly, we study the behaviour of the plain GA using NC model and corresponding heuristic and genetic methods in DNC model. The average *MPE* results of each problem in *ScFast* scenario are summarized in Table 5. GA_{NC} is more efficient than $HEFT_{NC}$ in Montage, Seismology and Epigenomics, except in the configuration with maximum number of hosts in the last two problems, but it is worse on the remaining four problems, mainly in configurations with high number of hosts. Using the DNC model, globally $HEFT_{DNC}$ has a marginal improvement in its results in all problems, the exception being Montage using the highest number of hosts where it obtains a substantial improvement. GA_{DNC} performs better on 6/7 problems, but fails on the configurations with the maximum number of hosts.

In *ScMixed* scenario, Table 6, GA_{NC} again improves *MPE* on the same 3 problems. However, $HEFT_{DNC}$ and GA_{DNC} obtain substantial improvements in almost all, 7/6 problems respectively, and mainly in the most complex host configurations (16 hosts).

Table 5 Summary of results from HEFT and GA methods using both evaluation models in ScFast scenario

Problem	Hosts	GA_{NC} (%)	$HEFT_{DNC}$ (%)	GA_{DNC} (%)
1000genome	2	−0.04	**0.14**	**0.14**
	4	0.02	0.43	**0.79**
	8	1.41	0.25	**2.17**
	16	−2.84	**0.58**	−1.71
Cycles	2	−0.35	**−0.02**	−0.35
	4	−3.04	**0.00**	−3.01
	8	−7.64	**0.01**	−8.15
	16	−16.75	**0.01**	−17.68
Epigenomics	2	1.78	0.09	**1.82**
	4	**3.11**	0.07	2.69
	8	**2.86**	0.07	2.47
	16	−0.60	**0.18**	−0.53
Montage	2	0.25	0.24	**0.94**
	4	0.76	0.74	**1.31**
	8	1.73	**3.80**	2.10
	16	0.78	**4.79**	0.64
Seismology	2	**0.01**	0.00	**0.01**
	4	**0.01**	0.00	**0.01**
	8	**0.05**	0.00	**0.05**
	16	−0.86	**0.00**	−0.83
SoyKB	2	0.03	0.22	**0.37**
	4	0.01	0.48	**0.56**
	8	−0.19	0.08	**0.16**
	16	−0.65	**0.02**	−0.36
SRASearch	2	0.02	0.04	**0.31**
	4	−0.01	0.09	**1.18**
	8	−2.06	**0.28**	−1.71
	16	−4.12	**0.18**	−4.83

Average makespan relative error (MPE) of all instances of each problem are reported (in bold best value for each problem and number of hosts)

Table 6 Summary of results from HEFT and GA methods using both evaluation models in SCMixed scenario

Problem	Hosts	GA_{NC} (%)	$HEFT_{DNC}$ (%)	GA_{DNC} (%)
1000genome	2	0.27	37.05	**37.98**
	4	−0.12	37.07	**37.81**
	8	−2.78	36.57	**37.02**
	16	−2.52	**38.80**	36.80
Cycles	2	−0.46	**0.91**	0.59
	4	−2.96	**0.83**	−1.97
	8	−7.39	**0.79**	−6.86
	16	−16.67	**0.72**	−15.84
Epigenomics	2	2.93	5.93	**7.29**
	4	5.14	7.33	**9.80**
	8	5.28	9.14	**11.88**
	16	4.07	14.02	**14.43**
Montage	2	−0.53	1.55	**3.90**
	4	0.23	1.89	**4.97**
	8	6.97	11.29	**13.33**
	16	4.01	13.29	**11.97**
Seismology	2	0.02	0.08	**0.09**
	4	0.00	**0.07**	0.06
	8	0.09	0.10	**0.19**
	16	−0.79	**0.14**	−0.36
SoyKB	2	−2.47	22.28	**23.29**
	4	−1.41	18.71	**19.01**
	8	−0.30	**15.79**	15.61
	16	−0.96	11.20	**11.24**
SRASearch	2	1.56	4.94	**7.13**
	4	−0.23	3.82	**5.15**
	8	−0.10	**6.86**	6.43
	16	2.65	**9.86**	6.98

Average makespan relative error (MPE) of all instances of each problem are reported (in bold best value for each problem and number of hosts)

Table 6 shows that, in general, the algorithms using the DNC model not only improve on the previous model, but even in the more complex ScMixed scenario with a higher number of hosts, these differences are maintained (1000Genome, Cycles and Seismology) or even increased (Epigenomics, Montage and SRSSearch). However, in the instances of soyKB problem the differences between the two models decrease as the number of hosts increases. This is due to the combination of a graph topology with a significant number of high in-degree nodes (i.e. tasks with a high number of predecessors' data dependencies) and a huge volume of data to be transferred. This combination generates high concurrency with a fast and long saturation of communication channels, both disk and network. In these cases, the new model has less room for improvement.

Table 7 Summary of the statistical study comparing the makespan quality of the heuristic and genetic proposed methods using both evaluation models

Method	$HEFT_{NC}$	GA_{NC}	$HEFT_{DNC}$	GA_{DNC}
$HEFT_{NC}$	–	–	–	–
GA_{NC}	–	–	–	–
$HEFT_{DNC}$	✔	✔	–	–
GA_{DNC}	✔	✔	–	–

The symbol (✔) means that the row method is significantly better than the corresponding column method

Due to the structural nature of these differences, these conclusions also apply to the results obtained with the

Table 8 Summary of results from Hybridization methods and inclusion of HEFT solution in the initial population using both evaluation models in SCFast scenario

Problem	Hosts	HGA_{Ph1} (%)	HGA_{Ph2} (%)	GA^H (%)	HGA^H_{Ph1} (%)	HGA^H_{Ph2} (%)
1000genome	2	**0.14**	**0.14**	**0.14**	**0.14**	**0.14**
	4	0.14	0.97	0.80	0.58	**0.98**
	8	−1.14	3.78	3.18	0.36	**3.83**
	16	−4.23%	2.32	1.54	0.58	**2.68**
Cycles	2	0.06	−0.09	**0.08**	0.07	**0.08**
	4	−1.73	−2.81	0.16	0.13	**0.27**
	8	−7.30	−6.19	0.56	0.01	**1.23**
	16	−15.57	−12.28	0.02	0.01	**0.69**
Epigenomics	2	−1.90	**1.84**	1.82	0.45	1.83
	4	−12.00	**3.96**	2.59	0.18	**3.96**
	8	−28.82	6.19	2.83	0.36	**6.24**
	16	−37.88	7.43	0.56	0.11	**7.50**
Montage	2	0.52	**0.89**	0.67	0.48	0.61
	4	0.25	**2.32**	1.46	0.72	2.00
	8	1.62	**5.48**	4.71	3.80	5.00
	16	1.70	**5.86**	5.30	4.51	5.80
Seismology	2	**0.01**	**0.01**	**0.01**	**0.01**	**0.01**
	4	−0.07	0.04	0.04	0.01	**0.05**
	8	−1.10	0.22	0.10	0.01	**0.25**
	16	−3.36	0.08	0.01	0.00	**0.34**
SoyKB	2	−0.42	**0.45**	**0.45**	0.29	**0.45**
	4	−0.48	**0.80**	0.69	0.48	**0.80**
	8	−0.76	0.86	0.35	0.08	**0.88**
	16	−0.95	0.93	0.19	0.02	**0.97**
SRASearch	2	0.01	0.21	**0.27**	0.07	0.15
	4	0.43	1.06	0.99	0.73	**1.07**
	8	−1.95	**0.73**	0.39	0.14	0.66
	16	−4.07	**0.59**	−0.22	−0.16	0.57

Average makespan relative error (MPE) of all instances of each problem are reported (in bold best value for each problem and number of hosts)

hybrid versions of the algorithms, as can be seen from Table 9.

A statistical analysis using non-parametric Friedman test for paired samples (using standard 0.05 significance level) signals significant differences among the studied methods (p-value $< 2.2e-16$). A Bonferroni post-hoc analysis reveals both genetic and heuristic DNC versions are statistical significantly better than NC versions, Table 7 resumes statistical results.

Regarding the hybrid proposals, now using only the DNC model, Table 8 and Table 9 show *MPE* results for *SCFast* and *SCMixed* scenarios respectively. Globally, while HGA_{Ph1} is unable to achieve better results than the genetic and heuristic methods, HGA_{Ph2} outperforms both previous methods in all problems. The only exception is the computing intensive problem Cycles, where the hybrid method is still unable to improve the heuristic *HEFT*.

Finally, in order to exploit the synergy between the genetic and heuristic approaches, we have introduced the HEFT solution in the initial population of the proposed genetic (GA^H) and hybrid algorithms (HGA^H_{Ph1} and HGA^H_{Ph2}). As a result, HGA^H_{Ph2} is the method with the best solutions in all problems and scenarios. It is only outperformed by the genetic algorithm with heuristic population (GA^H) on the SRASearch and Montage problem on *SCFast* scenario running on two hosts configuration. A new Friedman test shows that there are significant differences among the different methods using the DNC model (p-value $< 2.2e-16$). Table 10 summarises the results of the Bonferroni post-hoc tests, which show the significant superiority of hybridising the genetic algorithm with the scheduling phase of HEFT and using heuristic initial population (HGA^H_{Ph2}).

Table 9 Summary of results from Hybridization methods and inclusion of HEFT solution in the initial population using both evaluation models in SCMixed scenario

Problem	Hosts	HGA_{Ph1} (%)	HGA_{Ph2} (%)	GA^H (%)	HGA^H_{Ph1} (%)	HGA^H_{Ph2} (%)
1000genome	2	37.10	39.01	38.08	37.23	**39.05**
	4	36.90	39.26	37.70	37.22	**39.31**
	8	34.09	38.79	37.55	36.57	**40.28**
	16	35.09	39.34	38.83	38.80	**41.13**
Cycles	2	0.95	1.13	0.95	0.97	**1.27**
	4	−0.89	−0.60	1.07	0.85	**1.35**
	8	−5.86	−2.45	1.47	0.80	**2.18**
	16	−14.22	−9.55	0.72	0.72	**1.28**
Epigenomics	2	3.67	7.77	7.22	6.25	**7.78**
	4	−3.82	11.10	9.47	7.42	**11.13**
	8	−18.52	15.11	11.59	9.14	**15.28**
	16	−23.09	20.57	15.08	14.02	**20.75**
Montage	2	2.23	**4.86**	2.93	2.26	3.85
	4	2.86	**6.36**	3.34	2.30	4.84
	8	10.33	**16.23**	13.36	11.20	14.86
	16	10.77	**16.75**	15.10	13.19	16.37
Seismology	2	0.08	**0.11**	0.09	0.08	**0.11**
	4	0.01	0.12	0.09	0.09	**0.13**
	8	−1.08	0.37	0.20	0.12	**0.39**
	16	−3.42	0.28	0.14	0.14	**0.53**
SoyKB	2	22.05	**24.83**	23.59	22.56	24.80
	4	17.34	**20.00**	19.26	18.71	19.87
	8	14.01	16.57	15.98	15.79	**16.66**
	16	8.98	**12.36**	11.48	11.20	12.29
SRASearch	2	5.95	**7.78**	7.06	5.97	**7.78**
	4	4.98	6.44	5.31	5.19	**6.55**
	8	7.03	10.62	7.23	9.29	**10.82**
	16	7.49	11.23	10.11	9.87	**11.85**

Average makespan relative error (MPE) of all instances of each problem are reported (in bold best value for each problem and number of hosts)

Table 10 Summary of the statistical study comparing the makespan quality of the DNC versions of heuristic, genetic and hybrid proposed methods using random and heuristic initial population

Method	*HEFT*	*GA*	HGA_{Ph1}	HGA_{Ph2}	GA^H	HGA^H_{Ph1}	HGA^H_{Ph2}
HEFT	–	–	✔	–	–	–	–
GA	–	–	✔	–	–	–	–
HGA_{Ph1}	–	–	–	–	–	–	–
HGA_{Ph2}	✔	✔	✔	–	✔	✔	-
GA^H	–	–	–	–	–	✔	–
HGA^H_{Ph1}	–	–	–	–	–	–	–
HGA^H_{Ph2}	✔	✔	✔	✔	✔	✔	-

The symbol (✔) means that the row method is significantly better than the corresponding column method

5 Conclusions and future work

In this paper, we have presented the challenges of calculating the makespan of a scientific workflow and how simplifications for the sake of computational speed can affect the precision of the work. In the pay-per-use model of cloud computing, an inaccurate estimate of makespan will lead to a misleading decision about the necessary cost and time of the hired infrastructure. We have analysed the benefits of using a more appropriated computational model

(DNC) to improve the accuracy of the makespan estimates generated by the optimisation algorithms. We have introduced several improvements, (1) in this context lamarckian evolution had improved the makespan quality of the hybrid evolutionary algorithms solutions. (2) Different hybridization algorithms of a well-known heuristic as HEFT with the proposed genetic algorithm had been developed to improve the makespan, with HGA^{H}_{Ph2} being the one with the best solutions. (3) An accuracy measure had been introduced and applied to study its correlation with workflow problem typology, with respect to computation and communication ratio (CCR), in different cloud IaaS scenarios and using current real-world scientific workflow problems. We have observed how cloud infrastructure simulators include additional features, such as concurrency and saturation levels of disk and network transfer channels, which depends on the underlying hardware being simulated, but its complexity and computational cost prevent its effective use in scheduler evaluation models. For this reason, we want to use these simulator features to extend the DNC model and improve the performance of our solutions on mixed-feature workflows, such as those in the Montage problem.

In testing different hybridisation methods, we found that all the proposed algorithms have some kind of workflow problems where they work best. For this reason, in future work we want to explore the development of new population-based evolutionary algorithms, applying different heuristic decoding methods to each solution, and letting its best estimate guide the evolution in a multi-decoding approximation. In addition, we aim to experiment in a multi-objective context to study the impact of this new evaluation model, as well as to design new memetic proposals (Tang and Pan 2015; Zuo et al. 2017; Mencía et al. 2022) which combine intensive search with an appropriate exploration-exploitation balance (Guo et al. 2020; Lou et al. 2021; Osuna-Enciso et al. 2022) to minimise other valuable objectives in parallel, such as cost or energy consumption.

Author Contributions All authors reviewed the manuscript. The specific contribution of each author is as follows: PB: Conceptualisation, Methodology, Software, Validation, Formal Analysis, Investigation, Resources, Data curation, Writing-Original Draft, Visualisation. JP: Conceptualisation, Methodology, Investigation, Writing – Review & Editing, Supervision.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature. This research has been supported by the Spanish State Agency for Research (AEI) under research project PID2019-106263RB-I00.

Data availibility The data that supports the findings of this study is available online, in the Repository Section of http://www.di.uniovi.es/iscop/.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Ethical Approval Not applicable.

References

Adhikari M, Amgoth T (2019) An intelligent water drops-based workflow scheduling for IaaS cloud. Appl Soft Comput 77:547–566

Barredo P, Puente J (2022) Robust makespan optimization via genetic algorithms on the scientific workflow scheduling problem. In: bio-inspired systems and applications: from robotics to ambient intelligence, pp 77–87. Springer International Publishing

Biswas T, Kuila P, Ray AK, Sarkar M (2019) Gravitational search algorithm based novel workflow scheduling for heterogeneous computing systems. Simul Model Pract Theory 96:101932

Casanova H, Ferreira da Silva R, Tanaka R, Pandey S, Jethwani G, Koch W, Albrecht S, Oeth J, Suter F (2020) Developing accurate and scalable simulators of production workflow management systems with WRENCH. Futur Gener Comput Syst 112:162–175

Chakravarthi KK, Neelakantan P, Shyamala L, Vaidehi V (2022) Reliable budget aware workflow scheduling strategy on multi-cloud environment. Clust Comput 25(2):1189–205

Coleman T, Casanova H, Pottier L, Kaushik M, Deelman E, Ferreira da Silva R (2022) WfCommons: a framework for enabling scientific workflow research and development. Futur Gener Comput Syst 128:16–27

Deelman E, Vahi K, Rynge M, Mayani R, Da Silva RF, Papadimitriou G, Livny M (2019) The evolution of the pegasus workflow management software. Comput Sci Eng 21(4):22–36

Durillo JJ, Prodan R (2014) Multi-objective workflow scheduling in Amazon EC2. Clust Comput 17(2):169–189

Elaziz MA, Xiong S, Jayasena KP, Li L (2019) Task scheduling in cloud computing based on hybrid moth search algorithm and differential evolution. Knowl-Based Syst 169:39–52

Ghorbannia Delavar A, Aryan Y (2014) HSGA: a hybrid heuristic algorithm for workflow scheduling in cloud systems. Clust Comput 17(1):129–137

Google (2023) Google compute engine docs, compute-optimized machine family. https://cloud.google.com/compute/docs/compute-optimized-machines. Accessed 15 May 2023

Guo W, Xu P, Zhao Z, Wang L, Zhu L, Wu Q (2020) Scheduling for airport baggage transport vehicles based on diversity enhancement genetic algorithm. Nat Comput 19(4):663–672

Houck CR, Joines JA, Kay MG (1996) Utilizing Lamarckian evolution and the Baldwin effect in hybrid genetic algorithms. North Carolina State Univ, Department of Industrial Engineering, Raleigh

Juve G, Chervenak A, Deelman E, Bharathi S, Mehta G, Vahi K (2013) Characterizing and profiling scientific workflows. Futur Gener Comput Syst 29(3):682–692

Lou Y, Yuen SY, Chen G (2021) Non-revisiting stochastic search revisited: results, perspectives, and future directions. Swarm Evol Comput 61:100828

Madni SHH, Abd Latiff MS, Abdullahi M, Abdulhamid SM, Usman MJ (2017) Performance comparison of heuristic algorithms for task scheduling in IaaS cloud computing environment. PLoS One 12(5):1–26

Mencía R, Mencía C, Varela R (2022) A memetic algorithm for restoring feasibility in scheduling with limited makespan. Nat Comput 21(4):577–587

Osuna-Enciso V, Cuevas E, Morales Castañeda B (2022) A diversity metric for population-based metaheuristic algorithms. Inf Sci 586:192–208

Palacios JJ, González MA, Vela CR, González-Rodríguez I, Puente J (2015) Genetic tabu search for the fuzzy flexible job shop problem. Comput Oper Res 54:74–89

Tang M, Pan S (2015) A hybrid genetic algorithm for the energy-efficient virtual machine placement problem in data centers. Neural Process Lett 41(2):211–221

Thain D, Tannenbaum T, Livny M (2005) Distributed computing in practice: the condor experience. Concurr Comput Pract Exp 17(2–4):323–356

Topcuoglu H, Hariri S, Wu MY (2002) Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Trans Parallel Distrib Syst 13(3):260–274

Xu Y, Li KK, Hu J, Li KK (2014) A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues. Inf Sci 270:255–287

Ye X, Li J, Liu S, Liang J, Jin Y (2019) A hybrid instance-intensive workflow scheduling method in private cloud environment. Nat Comput 18(4):735–746

Yuan H, Bi J, Zhou M, Liu Q, Ammari AC (2021) Biobjective task scheduling for distributed green data centers. IEEE Trans Autom Sci Eng 18(2):731–742

Zhu Z, Zhang G, Li M, Liu X (2016) Evolutionary multi-objective workflow scheduling in cloud. IEEE Trans Parallel Distrib Syst 27(5):1344–1357

Zuo Y, Gong M, Jiao L (2017) Adaptive multimeme algorithm for flexible job shop scheduling problem. Nat Comput 16(4):677–698