Cluster Computing (2021) 24:115–139
https://doi.org/10.1007/s10586-020-03208-w

Towards decomposition based multi-objective workflow scheduling for big data processing in clouds

Emmanuel Bugingo[1] · Defu Zhang[1] · Zhaobin Chen[1] · Wei Zheng[1]

Received: 28 February 2020 / Revised: 2 November 2020 / Accepted: 5 November 2020 / Published online: 24 November 2020

Abstract
A workflow is a group of tasks that are processed in a particular order to complete an application. Also, it is a popular paradigm used to model complex big-data applications. Executing complex applications in a distributed system such as cloud or cluster implicates optimization of several conflicting objectives such as monetary cost, energy consumption, total execution time of the application (makespan). Regardless of this trend, most of the workflow scheduling approaches focused on single or bi-objective optimization problem. In this paper, we considered the problem of scheduling workflows in a cloud environment as a multi-objective optimization problem, and hence proposed a multi-objective workflow-scheduling algorithm based on decomposition. The proposed algorithm is capable of finding optimal solutions with a single run. Our evaluation results show that, by a single run, the proposed approach manages to obtain the Pareto Front solutions which are at least as good as schedules produced by running a single-objective scheduling algorithm with constraints for multiple times.

Keywords Multi-objective optimization · Workflow scheduling · Cloud computing · Decomposition approach

1 Introduction

Recently, workflow has become a popular paradigm to model the execution process of big data application in distributed environments such as clouds and clusters [1]. Since then, research on workflow scheduling has got much attention with an objective of producing optimal execution time. It is well-known that workflow scheduling in a heterogeneous environment is NP-Complete [2]. Traditionally, optimizing the overall execution time (makespan) of the workflow has been an important and common objective of workflow scheduling. Many works in literature have designed different heuristic algorithms to get the schedule with minimized makespan. However, as using cloud computing gains popularity, makespan optimization is no longer the only objective to be considered for optimization during workflow scheduling. Many other significant objectives that can be recognized as important as makespan have arisen such as cost, energy, reliability, utilization, etc. Those objectives need to be taken into consideration together with makespan during workflow scheduling. Therefore, modern cloud workflow scheduling algorithms must be able to optimize more than one objectives at the same time.

Generally, the main concern for cloud computing customers, when selecting virtual machine (VM) instances to execute their workflows, is the monetary cost. Cloud VM instances renting price is charged based on the computation capacity which can be reflected directly to the CPU frequency settings; for example, the pricing models adopted by CloudSigma [3] and Elastichosts [4] charge customers based on the selected CPU frequencies assigned to the VM instances during the execution of each workflow task. These VM instances' CPU frequency parameters are set in between maximum and minimum with a variation step frequency. Customers are likely to choose VM instances with lower price, but optimal makespan schedule will remain a critical requirement that cannot be neglected. In case of single objective optimization, VM instances running at high CPU frequency are a better choice for makespan optimization, while VM instances running at low CPU

✉ Wei Zheng
zhengw@xmu.edu.cn

[1] Department of Computer Science, School of Informatics, Xiamen University, Xiamen, China

frequency may be a better choice for cost optimization. However, it is not easy to make an appropriate CPU frequency selection if both objectives need to be taken into consideration. In addition, it is not a wise idea to execute all the tasks on the VM instances running only on high or low CPU frequencies, because such strategy can not handle the execution time difference of the workflow tasks and/or the complexity caused by the data dependencies among tasks. This strategy may also affect execution of each individual task which may directly reflect to the total makespan and the total monetary cost. Therefore, with such pricing scheme, an interesting challenge that arise to the customers will be how to properly select VM and tune their CPU frequencies for each tasks so that makespan and the cost of executing his/her application are minimized. The objective of this paper is to provide the solution toward such problem.

Makespan which is defined as the overall completion time of the whole workflow can be divided into two parts: summation of the execution time of the workflow tasks in critical path and the data transfer time that comes from tasks dependencies, especially when the two adjacent tasks are scheduled on different VM instances. The latter depends on the bandwidth of the transmission line, while the former depends on the CPU frequency allocated to the VM instance during the execution process of each task. In this paper, we only focus on the variability of the CPU frequency under fixed bandwidth. The renting cost of VM instances depends on the execution time of each task and the time unit charge rate which is normally decided by the selected CPU frequency according to a certain pricing models such as linear, superlinear and sublinear(more information about those pricing models can be found in [5]). Selecting high CPU frequency will result in smaller execution time of each task, however, the cost reduction achieved by this small execution time will not afford the cost increased by selecting this CPU frequency. This is to say, using VM instances with high processing capacity(CPU frequency) may ensure the minimal completion time of the workflow(makespan) under a high cost while using VM instances with low processing capacity(CPU frequency) may ensure minimal monetary cost under high completion time. As a result, selecting VM's CPU frequency for the improvement of the total cost values cannot be achieved without deteriorating the makespan values. So in this context, cost and makespan are conflicting objectives. Note that there is no single solution that can optimize both objectives at the same time. In this case, decision-makers may have to select the final preferred solution from the Pareto optimal objective vectors. Therefore, approximating the set of all the Pareto optimal objective vector is the appropriate way to deal with this multi-objective optimization problem.

It is well-known that, solutions to a multi-objective optimization problem under trivial situations could be an optimal solution of a scalar optimization problem in which the objective is an aggregation of all the objectives [6]. Therefore, the approximation of the Pareto Front(PF) can be decomposed into a number of scalar objective optimization sub-problems. To the best of our knowledge, none of the majority of the current state of the art multi-objective workflow scheduling algorithms [7–11] considered decomposition.

Therefore, with cost and makespan minimization in mind, in our previous work [12], we proposed a Workflow Scheduling Algorithm Based on Decomposition(WSABD). Given a scientific workflow with deterministic model of execution time and communication time, and a set of resources with variable CPU frequencies and cost, WSABD starts with initialization step using CFMax [14], then updates the functional values, finally, checks for the satisfaction of stopping criteria. The update step will be executed cyclically until the stopping criteria is satisfied. Then the algorithm will return the set of Pareto Front solutions. Different from the Evolution algorithm, WSABD speeds up its runtime to generate final schedule by using a search operation rather than overlapping mutations. The main contribution of that paper [12] is to propose a novel workflow scheduling algorithm with three variants, which incorporates decomposition approaches in workflow scheduling and uses search operation rather than mutations. In this paper, we significantly extend our previous work [12] by:

- Studying the performance of the algorithm when different decomposition approaches are considered.
- Studying the performance of the algorithm when different DAG structures are used.
- Studying the performance and the runtime of the proposed algorithm when different pricing models are used.
- Using the Hyper-volume indicator to investigate the quality of the set of PF solutions generated by the proposed algorithm.
- Studying the performance of the proposed algorithm when different settings such as population, iteration number, and the number of VMs are used.
- Studying the time complexity and the time overhead of the proposed algorithm.

The rest of this paper is organized as follows: Sect. 2 presents the related works, Sect. 3 describes the system and application model. Section 4 identifies the problem to be solved. The proposed algorithm is described in Sect. 5. Evaluation settings and findings are presented in Sect. 6. Finally, Sect. 7 concludes this paper and summarizes the future works.

2 Related works

Workflow scheduling and resource provisioning have become the fundamental research topic in the cloud computing platform. A remarkable number of works have been done to deal with optimization problems, such as single objective, bi-objective or multi-objective optimization problem. Among those focused on optimization of makespan as a single objective, HEFT [15] is the lightweight workflow scheduling heuristic for a heterogeneous environment like a cloud. Given a set of VM instances, HEFT ranks tasks according to their priority values, and then schedules them one after another to these VMs, aiming at minimizing the overall execution time of whole workflow with the consideration of the data transfer time among tasks. Because of its low complexity, HEFT has been employed by other researchers to provide new workflow scheduling algorithms [5, 14, 17–19]. With the objective of mapping all the workflow tasks to the available VMs so that makespan and cost are minimized, [20] proposed a bi-objective algorithm which is a hybrid of HEFT and GSA (Gravitational Search Algorithm).

As cloud computing emerges, modern workflow scheduling algorithms have to be able to optimize more than one objective. Different researches have been carried out to respond to this trend [8–11, 21, 22]. Mostly, multi-objective workflow scheduling algorithms rely on finding the Pareto set and then finding non-dominated solutions from the Pareto set. Different from other usual objectives, the work presented in [9] designed a new systematic method that considers both task security demands and interactions in securing task placement in the cloud. This work proposed a heuristic algorithm that is based on task completion time and security requirements. Most of the multi-objective workflow-scheduling algorithms considered two to three objectives at once, [8] focused on the optimization of Task Scheduling using a novel approach: Dynamic dispatch Queues (TSDQ) and hybrid meta-heuristic algorithms; And proposed two hybrid meta-heuristic algorithms: One based on Fuzzy Logic with Particle Swarm Optimization algorithm (TSDQ-FLPSO), and other based on Simulated Annealing with Particle Swarm Optimization algorithm (TSDQ-SAPSO).

The proposed algorithm approximates the optimal solution by considering user-specified constraints on objectives in a dual strategy: maximize the distance to the user's constraints for dominant solutions and minimize it. Evolutionary algorithms are an excellent way to solve the multi-objective optimization problem. However, they are designed for non-constrained problems. With the aim of investigating the proper task-VM mapping plan to minimize the total financial cost and the degree of imbalance under deadline constraints, the algorithm proposed in [23] modifies NSGA-II (Non-dominated Sorting Genetic Algorithm-II), and then makes it accept constraints. The modified version is used to solve the considered optimization problem. Inspired by the hybrid chemical reaction optimization algorithm, [24] proposed an energy-efficient workflow scheduling algorithm Even-though the proposed algorithm is for energy reduction, it also minimize the makespan of the schedule. This study, come up with a novel measure of determining the amount of energy to be saved when considering a DVS-enabled environment. Decomposition is a traditional multi-objective optimization strategy that decomposes a multi-objective optimization problem into a number of scalar optimization problems and optimizes them simultaneously. [6] presented a multi-objective evolutionary algorithm that is based on the decomposition techniques. However, this work is not designed for workflow scheduling purposes.

Differentiating from the work presented above, this paper proposes a workflow-scheduling algorithm based on decomposition. Our algorithm uses a search operation to get a new solution rather than overlapping mutations. To generate the initial population our algorithm employs CFMax [14].

3 System and application model

3.1 Application model

We assume the presence of cloud computing VM instances that are charged based on the pay-as-you-go basis of the CPU frequency used to execute each task in the workflow. Each allocated VM instance is provisioned from the beginning of the execution time of the task until its completion time. Information about data transfer between tasks and execution time of tasks when the VM instances run at their maximum CPU frequency is known in advance as illustrated in Fig. 1 and Table 1 respectively. We also consider a workflow application modeled as Directed Acyclic Graph(DAG) $G = (T, D)$, where T represents a set of interdependent tasks $T = \{t_1, t_2, .., t_n\}$ and D represents a set of intermediate data to be transferred between two adjacent tasks $D = \{d_{ij}\}$ (Fig. 1 for illustration). We use $pred(t_i)$ to determine a set of predecessors of task t_i, and $Succ(t_i)$ to determine a set of successors of task t_i. If task t_i is adjacent to task t_j, task t_i is a parent of task t_j, t_j is a child of task t_i. Task t_j can not start its execution before all its parents are completed and transmitted all required data d_{ij} to it. If a task is executed on the VM instance using a CPU frequency lower than the maximum, its execution time can be calculated by:

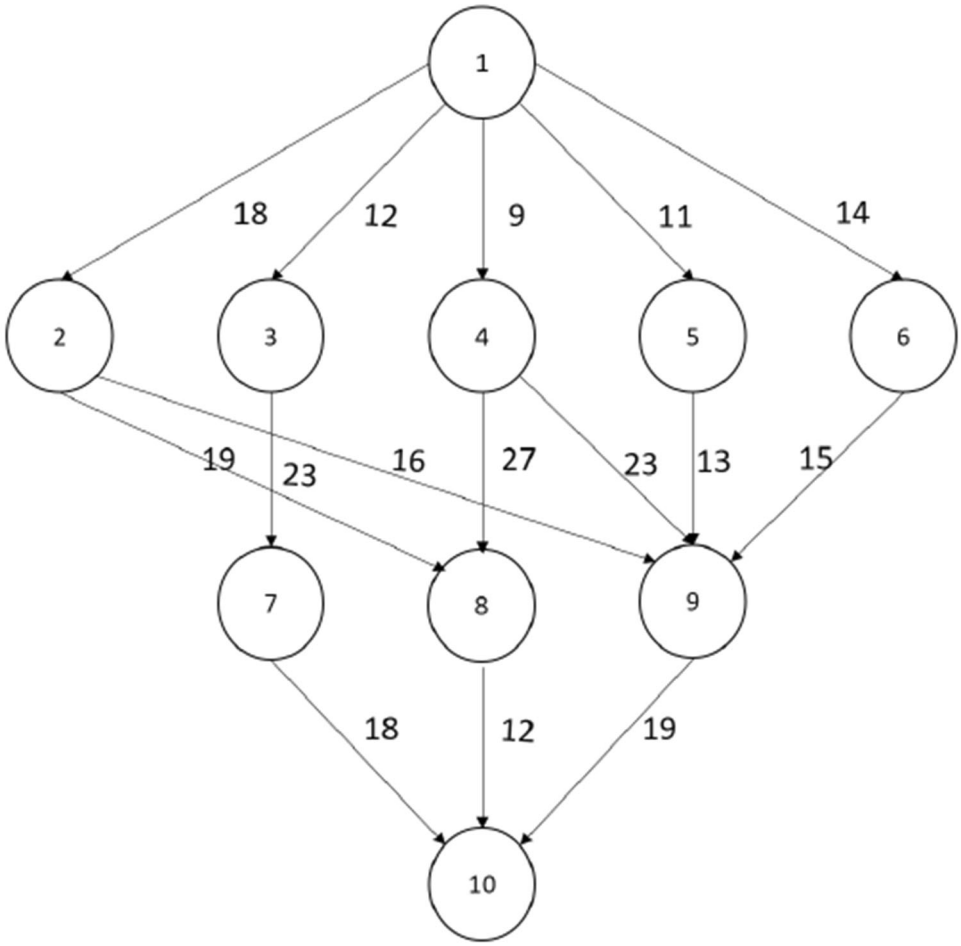

Fig. 1 DAG example with communication time

Table 1 An example of execution time

Tasks Id	*VM*1	*VM*2	*VM*3
1	30	12	16
2	27	21	72
3	4	36	6
4	21	6	20
5	20	16	81
6	28	6	48
7	35	14	7
8	5	48	30
9	21	3	36
10	48	2	64

$$ET_{(t,f)} = \left(\beta \cdot (\frac{f_{max}}{f} - 1) + 1\right) \cdot ET_{(t,f_{max})} \quad (1)$$

where $ET_{(t,f_{max})}$ is the execution time when task t_i runs at the maximum CPU frequency and the parameter β indicates the impact of the CPU frequency on task execution time in the range of 0 to 1. In this paper, we set β=0.4 by default.

3.2 System model

The heterogeneous VM instances operate on variant CPU frequencies in between maximum and minimum value (f_{min},f_{max}) with a step f_{step} that determines the variability level as illustrated in Table 2. Each VM instance is charged according to the CPU frequency allocated to each task. We adopted three pricing models presented in [19]. Let $C_{(m,f)}$ represent the price charged per time unit of a VM instance m with CPU frequency f, $C_{(m,f_{min})}$ represent the price charged per time unit of a VM instance m running at minimum CPU frequency f_{min}, and δ represent the coefficient to tune the charging rate of the price according to f.

Table 2 An example of CPU frequency settings for 3 VMs

Machines	$MaxCPU_{fr}$	$MinCPU_{fr}$	$StepCPU_{fr}$
1	4200	2100	300
2	3600	2400	300
3	3000	2000	200

Then, with the linear pricing model, $C_{(m,f)}$ can be calculated as bellow:

$$C_{(m,f)} = C_{(m,f_{min})} + \delta_m . \frac{f_i - f_{min}}{f_{min}} \tag{2}$$

With Super-linear pricing model $C_{(m,f)}$ can be calculated as bellow:

$$C_{(m,f)} = C_{(m,f_{min})} + \delta_m . \left(\left(1 + \frac{f_i - f_{min}}{f_{min}} \right) \cdot \log \left(\frac{f_r - f_{min}}{f_{min}} \right) \right) \tag{3}$$

With Sub-linear pricing model $C_{(m,f)}$ can be calculated as bellow:

$$C_{(m,f)} = C_{(m,f_{min})} + \delta_m . \log \left(1 + \frac{f_i - f_{min}}{f_{min}} \right) \tag{4}$$

Let also $EC_{(t,m,f)}$ denote the task execution cost on the VM instance running at a frequency f. $EC_{(t,m,f)}$ is calculated as:

$$EC_{(t,m,f)} = ET_{(t,f)} \cdot C_{(m,f)} \tag{5}$$

The total cost to execute the whole workflow tasks is calculated as:

$$TC = \sum_{\forall (t,m) \in S} EC_{(t,m,f)} \tag{6}$$

where S is the schedule which describes the tasks-VM mapping and the operating CPU frequency of each VM instance.

4 Problem formulation and basics of decomposition algorithm

In this paper we consider the problem of minimizing cost and makespan as a multi-objective optimization problem (MOP) which can be written as follows:

$$\begin{aligned} &minimize F(x) = (f_1(x), ..., f_m(x))^T \\ &Subject to x \in X \end{aligned} \tag{7}$$

where X is the decision space (variable space), F:$X \rightarrow R^m$ consists of m values, m is the number of objective functions and R^m is the the objective space.$\{F(x) x \in X\}$ is a set called attainable objective set. Mostly, the objectives in Eq. 7 contradict each other. The only possible way to balance them is to find the trade-off among them which can be achieved by using Pareto optimality.

The aim of multi-objective optimization algorithms is to find the trade-off between contradicting objectives. During multi-objective workflow scheduling, there may be a big or even infinite number of solutions. However, only non-dominated solutions can be taken by decision-makers for the selection of the final preferred solution. A solution S_a is said to dominate a solution S_b if and only if S_a is better than S_b in both objectives. $F(x')$ is said to be Pareto Optimal if there is no solution x such that $F(x)$ dominates $F(x')$. This means that any change in Pareto optimal values for the satisfaction of one objective must lead to the change in at least other objective. A set of all Pareto optimal solutions is called Pareto Set(PS), and a set of all Pareto optimal objective vectors is called Pareto Front(PF). Some mathematical models have been developed to approximate PF. However, it is well-known that Pareto Optimal solutions for a multi-objective problem under slight conditions can be the optimal solutions of a scalar optimization problem in which objective is a combination of both the weight vectors [6]. Hence, the approximation of the PF can be decomposed into a number of scalar objective optimization subproblems. In this paper, we adopted five decomposition approaches: Weighted Sum (WS), Tchebycheff (TE), Penalty Boundary Intersection (PBI), Modified Tchebycheff (MTE) and NIMBUS, to decompose the problem of approximation of the PF into a number of scalar optimization problems.

4.1 Weighted sum approach

This approach considers a concave combination of different objectives. Let $\lambda = (\lambda_1, ..., \lambda_m)_T$ represent the weight vector, i.e., $\lambda_i \geq 0$ for all $i = 1, .., m$ and $\sum_{i=1}^{m} \lambda_i = 1$. The Pareto optimal point of the Eq. 7 can be calculated as follows:

$$\begin{aligned} &minimize\, g^{ws}(x|\lambda) = \sum_{i=1}^{m} \lambda_i f_i(x) \\ &Subject\, to\, x \in X \end{aligned} \tag{8}$$

where $g^{ws}(x|\lambda)$ is used to express that λ is a coefficient vector in the objective function, x represents the values of the variables to be optimized (cost and makespan in our case), λ is used as a weight vector that facilitates the approach to generate a set of different Pareto optimal vectors. This approach will mostly work well as long as the PF is convex (concave in the case of maximization). However, in real-world not all Pareto Front vectors are concave or convex. In case of non-concave this approach will hardly work. To overcome these deficiencies, some efforts have been made to incorporate other techniques such as $\epsilon - constraint$ into this approach.

4.2 Tchebycheff approach

The scalar optimization problem can be represented as follows:

$$\begin{aligned} &minimize\, g^{te}(x|\lambda, z) = \max_{1 \le i \le m} \{\lambda_i |f_i(x) - z_i|\} \\ &\quad Subject\, to\, x \in X \end{aligned} \tag{9}$$

where $z^* = (z_1^*, ..., z_m^*)^T$ is a reference point ,i.e., $z_i^* = min(\{f_i(x)(x \in X)\})$ for each $i = 1, ..., m$. For each PF point z^* generated by Eq. 9 there is a weight vector λ that emphasizes that z^* is an optimal solution. Therefore, by alternating the weight vector, this approach will generate multiple different Pareto optimal solutions.

4.3 Modified Tchebycheff approach

The scalar optimization problem can be represented as follows:

$$\begin{aligned} &minimize\, g^{mte}(x|\lambda, z) = \max_{1 \le i \le m} \{\frac{1}{\lambda_i} |f_i(x) - z_i|\} \\ &\quad Subject\, to\, x \in X \end{aligned} \tag{10}$$

where $z^* = (z_1^*, ..., z_m^*)^T$ is a reference point ,i.e., $z_i^* = min(\{f_i(x)(x \in X)\})$ for each $i = 1, ..., m$. For each PF point z^* generated by Eq. 10 there is a weight vector λ that emphasizes that z^* is an optimal solution. Like *TE* approach, alternating the weight vector will cause the *MTE* approach to generate multiple different Pareto optimal solutions (more information about this approach can be found in [25]).

4.4 Penalty-based boundary intersection

The scalar optimization problem can be represented as follows:

$$\begin{aligned} &minimize\, g^{pbi}(x|\lambda, z) = d_1 + \theta d_2 \\ &\quad Subject\, to\, x \in X \\ &\text{where } d_1 = \frac{\|(F(x) - z)^T \lambda\|}{\|\lambda\|} \\ &\text{and } d_2 = \|F(x) - (z + d_1 \lambda)\| \end{aligned} \tag{11}$$

where x is a vector containing the variables of both objectives to be optimized (in this paper we consider cost and makespan), λ is a weight vector, z represents the reference point which corresponding to the minimal values for both objectives considered(cost and makespan), θ is a penalty parameter that has to be greater than 0, d_1 is the distance between z^* and y, d_2 is the distance between $F(x)$ and line L. It worths mentioning that *PBI* and *TE* use the same set of evenly distributed weight vectors when the number of considered objectives is two. However, *PBI* has more advantages over *TE*.

- The resultant optimal solutions generated by *PBI* are much more uniformly distributed than those generated by *TE*, especially when the number of the weight vectors is not large.
- For *TE*, when x dominates y it is still possible that $g^{te} = (x|\lambda, z^* = g^{te} = (y|\lambda, z^*)$. However, this is a rare case for *PBI*.

To achieve these advantages, penalty factor values have to be set. It is well known that a too large or too small number of penalty factors will decline the performance of this approach.

4.5 NIMBUS approach

Miettinen et al. [26] describes NIMBUS as an interactive classification-based multi-objective optimization approach. It uses the same principle as other decomposition algorithms. The scalar optimization problem can be represented as follows:

$$\begin{aligned} &minimize\, g^{nbs}(x|\lambda, z) = \max_{1 \le i \le m} \{(\lambda_i (f_i(x) - z_i)), \\ &\quad (\lambda_j (f_i(x) - z_j^*))\} + p \sum_{i=1}^{m} \lambda_i f_i(x) \\ &\quad Subject\, to\, x \in X \end{aligned} \tag{12}$$

where z is the ideal objective vector, z^* is the aspiration levels for the objective function, $p > 0$ is a relatively small scalar bounding trade-off, and λ is a weight vector used to scale up or down the values of the considered objectives. More details about *WS*, *TE* and *PBI* approaches can be found in [6] and more details about *NIMBUS* can be found in [27].

Based on the models and assumptions above, we present the multi-objective workflow scheduling algorithm, which generates schedules and properly tunes the CPU frequency for each task so that makespan and total cost of the submitted workflow are minimized.

5 The proposed algorithm

This section describes the Workflow Scheduling Algorithm Based on Decomposition (WSABD), a multi-objective algorithm proposed to solve the problem described in Sect. 4.

5.1 Algorithm description

As presented in algorithm 2, WSABD takes seven elements (*W*, *VMs*, *SC*, *N*, *WV*, *T*, *A*) as input. Those input elements are described as follows: *W*(Workflow) is a set of tasks with known execution time and communication time, *VMs* is a set of resources with CPU frequencies and associated prices, *SC* is a fixed number of iterations used as the stopping criteria, *N* is the number of sub-problems considered, *WV* is a uniform distribution of *N* Weight Vectors: $\lambda^1 ... \lambda^N$ (N=2), *T* is the number of weight vectors in the neighborhood of each weight vector, *A* is a decomposition approaches(selected from the set of decomposition approaches) that is used to compute and compare new solutions. In our case, *A* can be one of those five approaches: WS, TE, MTE, PBI and NIMBUS. WSABD returns *EP* as the output, where *EP* is a set of non-dominated solutions. The proposed algorithm consists of three main steps: Initialization, Update and checking the stopping criteria.

Algorithm 1 Using a Specific Decomposition Approach

switch (A)
case (WSABD-WS)**:**
  Calculate $g^{ws}(x^j \| \lambda^j)$, $g^{ws}(y^{'} \| \lambda^j)$ using Eq.(8), **break**
case (WSABD-TE)**:**
  Calculate $g^{te}(x^j \| \lambda^j, z)$, $g^{te}(y^{'} \| \lambda^j, z)$ using Eq.(9), **break**
case (WSABD-MTE)**:**
  Calculate $g^{mte}(x^j \| \lambda^j, z)$, $g^{mte}(y^{'} \| \lambda^j, z)$ using Eq.(10), **break**
case (WSABD-PBI)**:**
  Calculate $g^{pbi}(x^j \| \lambda^j, z)$, $g^{pbi}(y^{'} \| \lambda^j, z)$ using Eq.(11), **break**
case (WSABD-NIMBUS)**:**
  Calculate $g^{nbs}(x^j \| \lambda^j, z)$, $g^{nbs}(y^{'} \| \lambda^j, z)$ using Eq.(12), **break**
end switch
if $(g^*(y^{'} \| \lambda^j, z) \leq g^*(x^j \| \lambda^j, z))$ **then**
  Set $x^j = y^{'}$ and $FV^j = F(y^{'})$
end if

$$RW_{(t,m,f)} = TC_{(t,m',f')} - TC_{(t,m,f)} \quad (13)$$

Algorithm 2 WSABD (Workflow Scheduling Algorithm Based On Decomposition)

Input: W, VMs, SC, N, WV, T, A
Output: EP: External Population(Non Dominated Solutions)
Step1: Initialization :
1: Set iteration = 0, $EP = \emptyset$
2: Calculate Euclidean distance between any two weight vectors, then work out the T closest weight vectors to each weight vector and store these vectors in B.
3: Initialize W and VMs
4: Initialize the population using CFMax[14]
5: Calculate the individual cost and makespan of the population as the objective function value (FV)
6: Initialize the reference point z, z=($minimized_{cost}$, $minimized_{makespan}$)
Step 2: Update
7: **for** (individual: population) **do**
8: **if** ($index[individual]$ is even) **then**
9: Get new individual $y^{'}$ ← through minimized-cost
10: **else**
11: Get new individual $y^{'}$ ← through minimized-makespan
12: **end if**
13: Calculate the cost and makespan of $y^{'}$, rename them as $FV(y^{'})$
14: Update reference point z using FV($y^{'}$)
Update the solutions of this individual's neighbors
15: **for** (j ∈ B (individual)) **do**
16: Call **Algorithm 1** with A as one of the designated decomposition approach (SW, TE, MTE, PBI, $NIMBUS$).
17: **end for**
Update EP
18: Remove from EP all the vectors dominated by $FV(y^{'})$
19: add $FV(y^{'})$ to EP if no Vectors in EP dominate $FV(y^{'})$
20: **end for**
21: iteration +=1
Step 3: Checking the stopping criteria
22: **if** (iteration==SC) **then**
23: **return** EP
24: **else**
25: Go to step 2.
26: **end if**

In our previous study [28], we proposed a workflow scheduling algorithm with two variants (CFMax, CFMin). Like all other studies that focused on optimizing workflow scheduling objectives under user's deadline [13, 16, 31], the purpose of our former study [28] was to minimize the users' monetary expenditure for the submitted workflow application under a given deadline. The experimental results of our proposal show that CFMax performs better than CFMin. To satisfy the user's deadline regardless of the total cost, CFMax starts with a makespan-aware scheduling algorithm (such as HEFT, MIN-MIN, MCT, or MAXMIN like in [14]) and schedules each task to the appropriate VM instance using the maximum CPU frequency. To guarantee the cost reduction, a reduction-weight(RW) table is created to measure the cost reduction impacted by the task reassignment and CPU frequency re-allocation. The values are inserted into *RW* table according to Eq. (13), where $TC_{(t,m',f')}$ represents new task's cost after changing the CPU frequency, and $TC_{(t,m,f)}$ represents the cost of executing the tasks on the VM instance under current CPU frequency. To take a re-assignment decision, the combination of VM instance and CPU frequency that produces the maximum value in *RW* table is selected as the winner.

In the first step of *WSABD*, we initialize the inputs, CFMax is used to generate the initial population (line 4 of Algorithm 2). In the second step, we update the initialized variables by iteratively changing vector variables and iteration settings. We compute new solutions according to the updated settings and update new solutions according to the decision from a designated decomposition approach. In the third step, we check if stopping criteria is satisfied then return the Non-Dominated solutions (namely, *EP*), otherwise go to step two.

In detail, step one (from lines 1 to 6) consists of initialization of the input variables such as the number of weight vectors in the neighborhood (T), weight vector

indexes B, VM instance information, DAG information. After the population is initialized using CFMax, the individual cost and makespan are calculated as objective function values. Line 4 in algorithm 2 says that FV stands for objective function values. Among them the minimum one is selected as the initial reference point z. Note that T plays an important role in limiting the search operation to a certain extent. The second step (from line 7 to 26) consists of two sub-steps. In the first sub-step (from line 7 to 15), we update the individual(y') by searching either the minimized cost or the minimized makespan based on the position index of the individual in the population. If the index of the individual in the population is even, we get new individual by minimizing cost otherwise we get new individual by minimizing makespan. Then the new cost and the new makespan are calculated according to the new individual values and renamed as $(FV(y'))$. Finally, the reference point is updated according to $(FV(y'))$. In the second sub-step (from line 16 to 26), the solutions of the individual's neighbors are updated. As shown in the algorithm 1, in this stage we use one of the approaches defined in Eqs. (8), (9) and (11). For each selected approach, we calculate the $g^*(y'\|\lambda^j, z)$ and $g^*(x^j\|\lambda^j, z)$. Before resetting the current cost-makespan(x^j) value to be equal to the new individual y', the two values are compared first. If the new values $(g^*(y'\|\lambda^j, z))$ are less or equal to the current values $(g^*(x^j\|\lambda^j, z))$, update is allowed otherwise the current values are kept and the algorithm continues. EP is also updated according to the new values of $FV(y')$. The second step(the update step) is repeated until the stopping criteria is satisfied then the final EP is returned.

5.2 WSABD complexity

WSABD algorithm has three main parameters: population size ($PopSize$), the number of neighbors (T) and the number of iterations ($iterNum$). $PopSize$ corresponds to the number of single objective sub-problems that the algorithm decomposes such multi-objective problem into, and indicates search breadth of the algorithm. T can be expressed as a sub-problem by using the number of adjacent subproblems. $iterNum$ represents search depth of the algorithm. Suppose that there exists a workflow with N tasks and VMs available resources with C selections of CPU frequency on average. According to Algorithm 2, the update step needs $iterNum$ times. Each time $Popsize$ individuals in population have to be updated in turn. In the process of updating each individual, the solutions of its T neighbors will be recalculated by Algorithm 1. Therefore, the time complexity of $WSABD$ is: $iterNum \times PopSize \times Max(T, N \times VMs \times C)$, where $N \times R \times C$ is the number of computations required to construct the RW table.

6 Evaluation

In order to evaluate the performance of the proposed algorithm, we perform two types of experiments by using different parameter settings (as presented in Sect. 6.1) in each experiment. The first experiment learns the variability of cost and makespan values under fixed parameters settings (the number of VMs, the number of iterations, population size and the number of neighbors), and the second one learns the variability of the cost and makespan values under changeable parameter settings. We mainly focus on three types of evaluations including optimization results, runtime results and HyperVolume(HV) results. For the first experiment, we presented all the three types of results, while for the second experiment we just focus on the HV results. The results and their discussions are presented in the subsections below Fig. 2.

6.1 Evaluation settings

The algorithm presented in Sect. 2 and its simulation tools are implemented in Java. A PC with 4-core Intel i5-7300HQ @2.5GHz CPU and 8GB-RAM is used as an experimental environment. Each simulated resource runs at a CPU frequency in the range between maximum and minimum, with a variation step, selected randomly as shown in Table 2. We considered three pricing models described in section [5], with c_{min} and δ presented in Table 3 as in our previous work [14]. We also considered three real workflows (Montage, Inspiral, and Epigenomics) which are different in structure. [30] described **Montage** as a type of workflow that is created by the NASA/IPAC Infrared Science Archive as an open toolkit that can be used to generate custom mosaics of the sky from input images in the Flexible Image Transport System (FITS) format. [30] described **LIGO Inspiral Analysis** (also known as LIGO or Inspiral in short) as a type of workflow that is created to analyze data from the coalescing of compact binary systems such as binary neutron stars and black holes. The same work described **Epigenomics** as a type of workflow that is created by the USC Epigenome Center and Pegasus Team to automate various operations in genome sequence processing. The considered workflows are defined by DAX files which follow DAX XML specifications. We download the DAX files describing a 1000-node Montage workflow, a 1000-node Inspiral workflow and 997-node Epigenomics workflow from the website of Pegasus workflow Generator [29] and use them as the inputs of the scheduling algorithm in our simulator. For the optimization experiments, we set the input parameters such as population size, the number of neighbors, the number of the iterations and the number of VMs

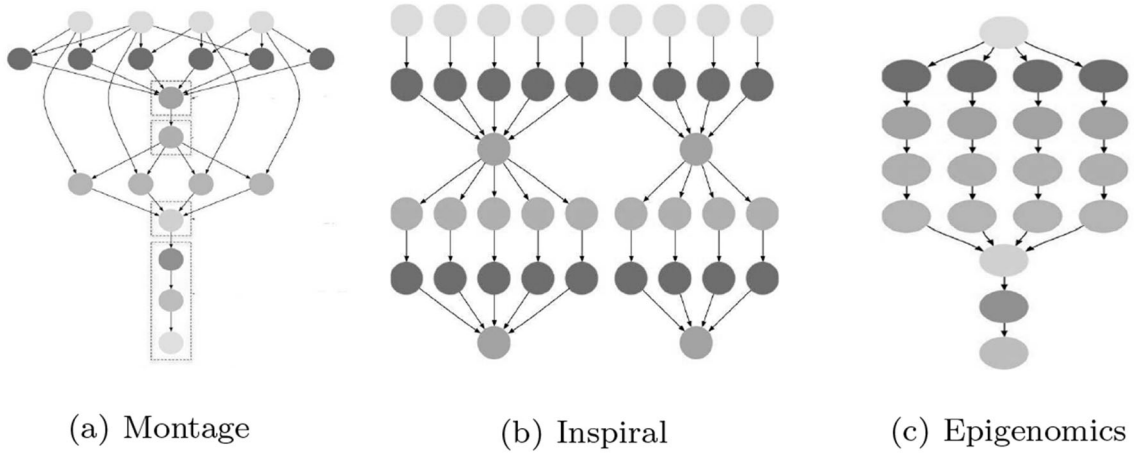

Fig. 2 Structure of the considered workflows (*Source*: [29])

Table 3 c_{min} and δ values

Pricing model	c_{min}	δ
Linear	9.24	3.33
Superlinear	9.29	4.44
Sublinear	2.78	12

as 20, 5, 2000 and 15 respectively. WSABD is applied to each workflow with the parameters defined above.

6.2 Performance results

To evaluate the performance of the proposed algorithm, we considered **Hyper-Volume indicator**, one of the performance indicators used to measure the single score to indicate the quality of a set *PF* solutions acquired by the proposed algorithm using different settings. For a 3D data, this score is equal to the volume covered by the *PF* and the selected reference point *R*. However, in case of 2D (shown in Fig. 3), it refers to the area covered by the *PF* set and the selected reference point *R*. The *HV* enclosed by the PF and reference point *R* is calculated as follows:

$$HV(PF, R) = \cup_{v \in PF} volume(v) \tag{14}$$

where $volume(v)$ is the area bounded by the solution v in *PF* and *R*. For the minimization problem, the larger the HV is,the closer the solution set obtained by the algorithm is to the lower left corner of the coordinate axis, the better the convergence of the algorithm and the distribution of the solution set are. When the *HV* values are stable, the solution set obtained by the algorithm is not changed, and the algorithm is convergent.

6.3 Fixed parameter settings

6.3.1 Optimization results

In this section we present and discuss the optimization results of the proposed algorithm. Given the parameters and the settings described above, CFMax is firstly

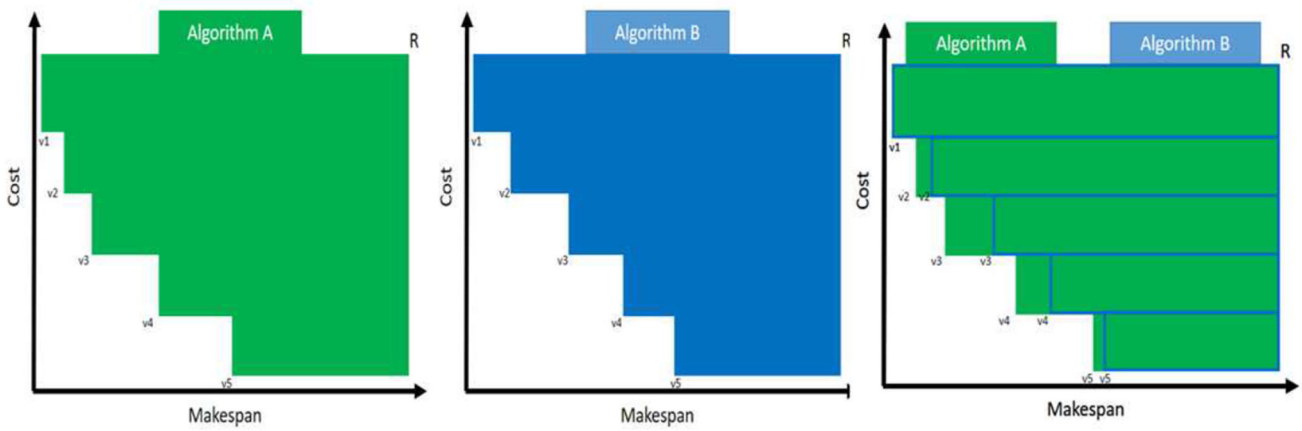

Fig. 3 Calculation of HV results

executed, then cost and makespan results are collected. The results generated by CFMax are also the initial population for WSABD. WSABD can approximate optimal solutions (set of PF solutions) at the end of a single run. Note that WSABD is made of five decomposition approaches. As long as the iteration number is not yet satisfied and all possibilities are not tried yet, the algorithm will continue to generate new solutions. The experimental results are shown in Figs. 4, 5, 6 7, 8, 9, 10, 11 and12.

Note that users can determine the solutions that suit their needs based on some constraints such as budget and/or deadline. Based on the optimization results presented in these figures, the key findings of this subsection can be summarized as follows:

- The overall optimization results show that Montage produces small value results of cost and makespan while Epigenomics produces high-value results.
- When the considered parameter settings are applied to Montage and Inspiral for linear and superlinear pricing models, WSABD-TE achieves high makespan than other algorithms.
- When the pricing model is sublinear, no algorithm achieves the same results as the others.
- In case of sublinear pricing model, cost and makespan results generated by our algorithm start higher than the ones generated by CFMAX.

6.3.2 Runtime results

To evaluate the runtime of WSABD, the algorithm will run 100 times with each decomposition approach under aforementioned environments and settings, then the average time consuming will be calculated as the runtime result. The runtime results are shown in Figs. 13, 14 and 15.

Based on the runtime results presented in those figures, the key findings of this subsection can be summarized as follows:

- The overall runtime results demonstrate that when DAGs are Montage and Inspiral, WSABD-TE's runtime is higher than the runtime of other decomposition approach options.
- For Epigenomics, the runtime of WSABD-PBI becomes higher when the pricing models are linear and superlinear, but it becomes second lower when the pricing model is sublinear.
- For Montage, lower runtime can be achieved by WSABD-WS for linear pricing model, WSADB-MTE for superlinear pricing model and WSABD-PBI for sublinear pricing model.
- For Inspiral, lower runtime can be achieved by WSABD-NIMBUS for both linear and superlinear pricing models, and by WSABD-NIMBUS for sublinear pricing model.

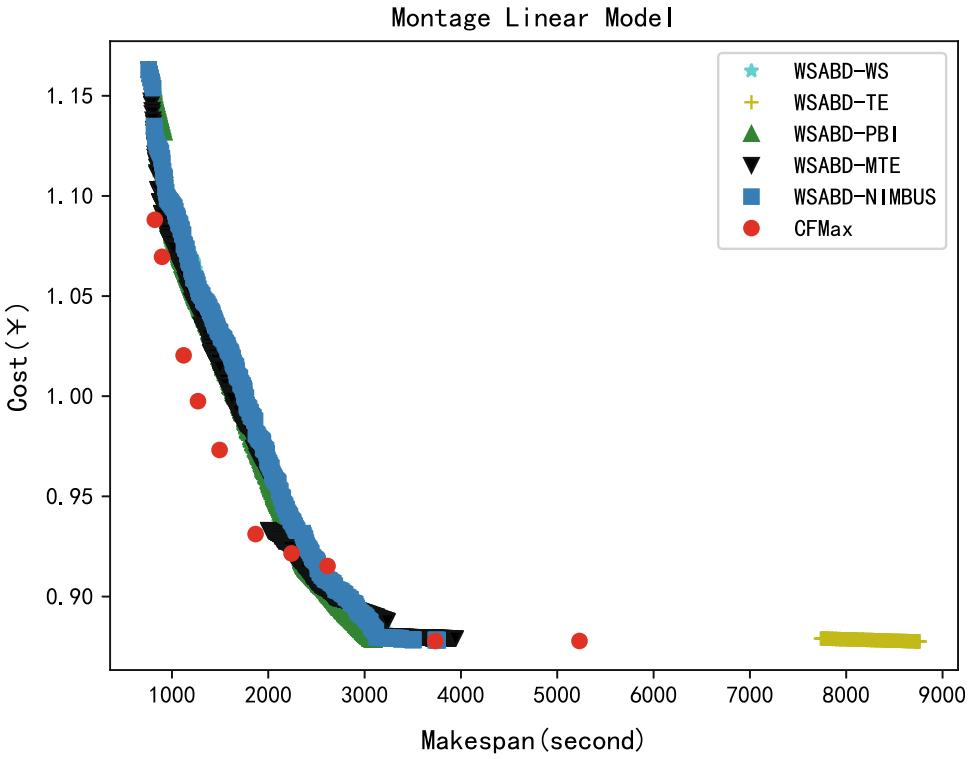

Fig. 4 Montage optimization results (Linear model)

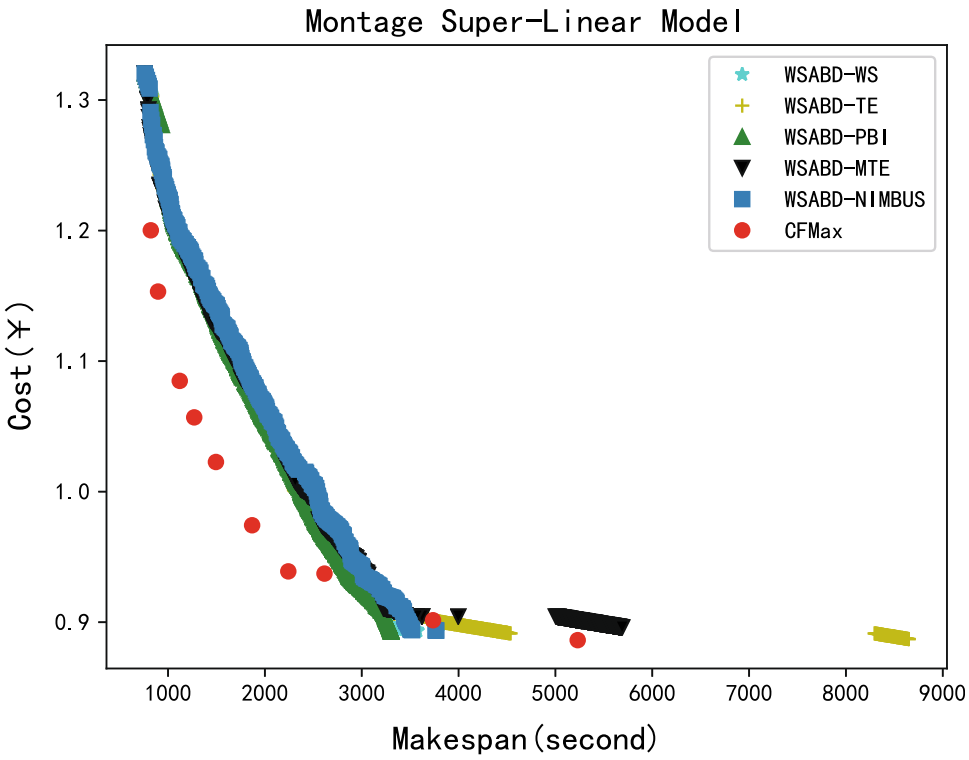

Fig. 5 Montage optimization results (Superlinear model)

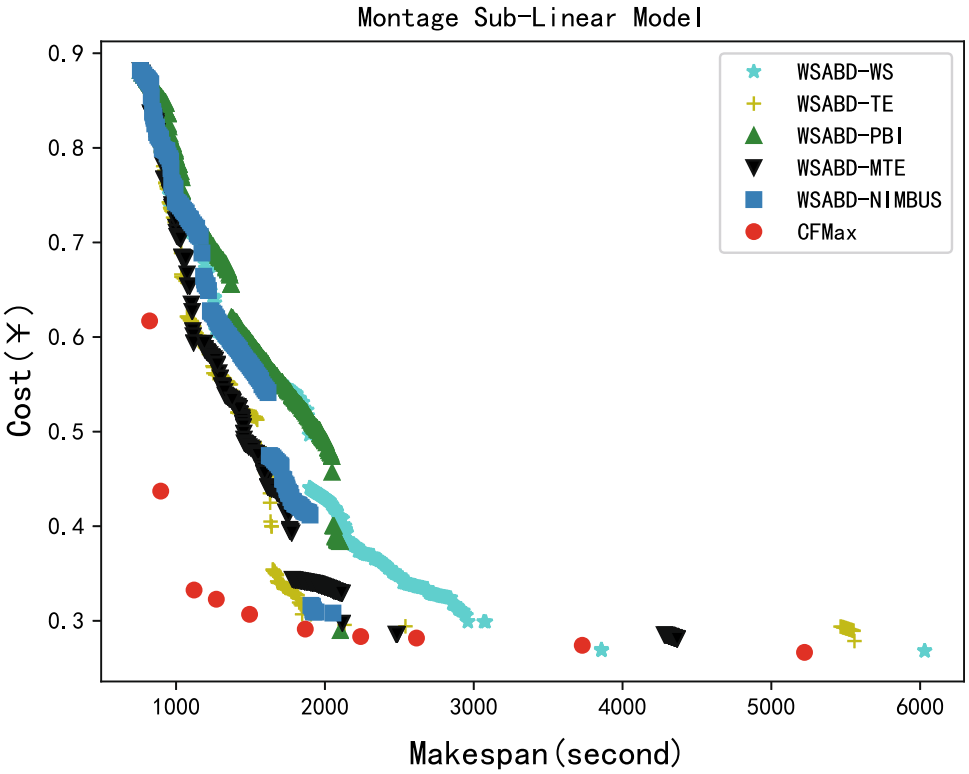

Fig. 6 Montage optimization results (Sublinear model)

– For Epigenomics, lower runtime can be achieved by WSABD-MTE for both linear and sublinear pricing models, and by WSABD-WS for superlinear pricing model.

6.3.3 HV results

With the same parameters settings as used for optimization and runtime experiments, we also evaluated performance

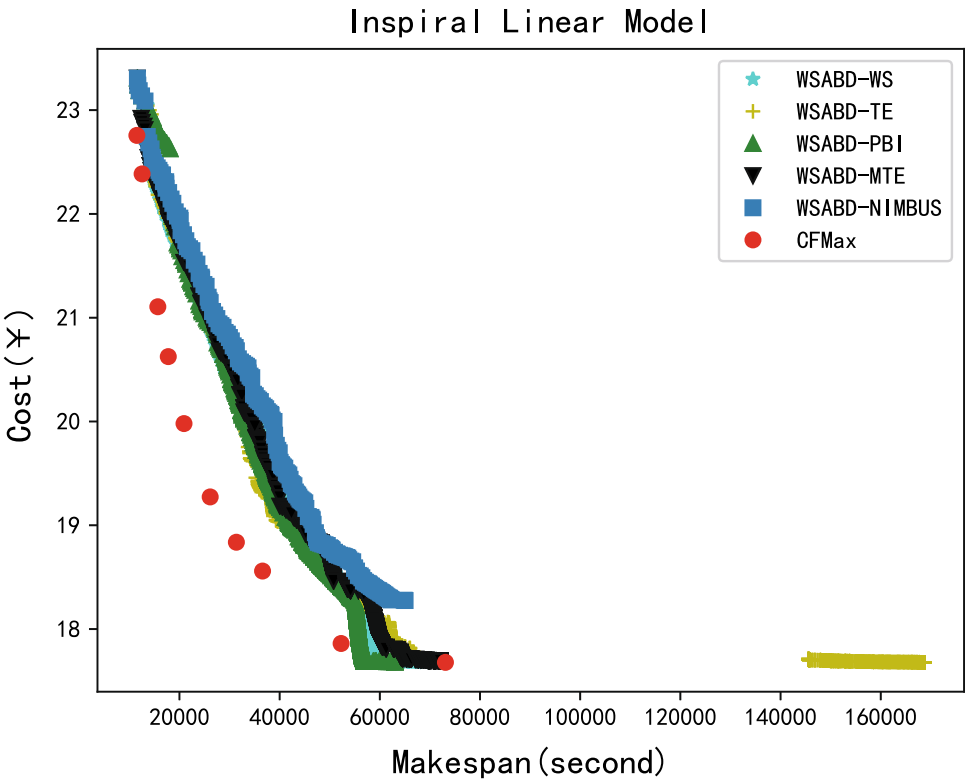

Fig. 7 Inspiral optimization results (Linear model)

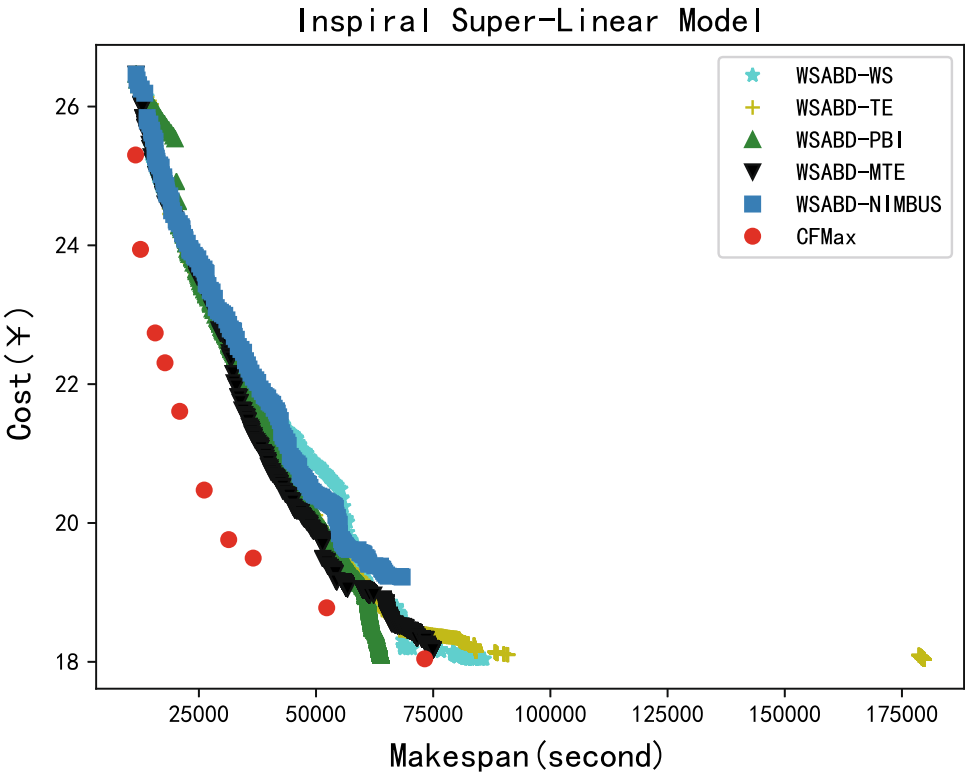

Fig. 8 Inspiral optimization results (Superlinear model)

of the proposed algorithm based on the HV results of each algorithm. It worths mentioning that the CFMAX results will mostly be in the lower left corner of the coordinate axis compared to the results of the proposed algorithm, which gives CFMAX the ability to dominate in many cases. If the results of CFMAX are the best, we also search

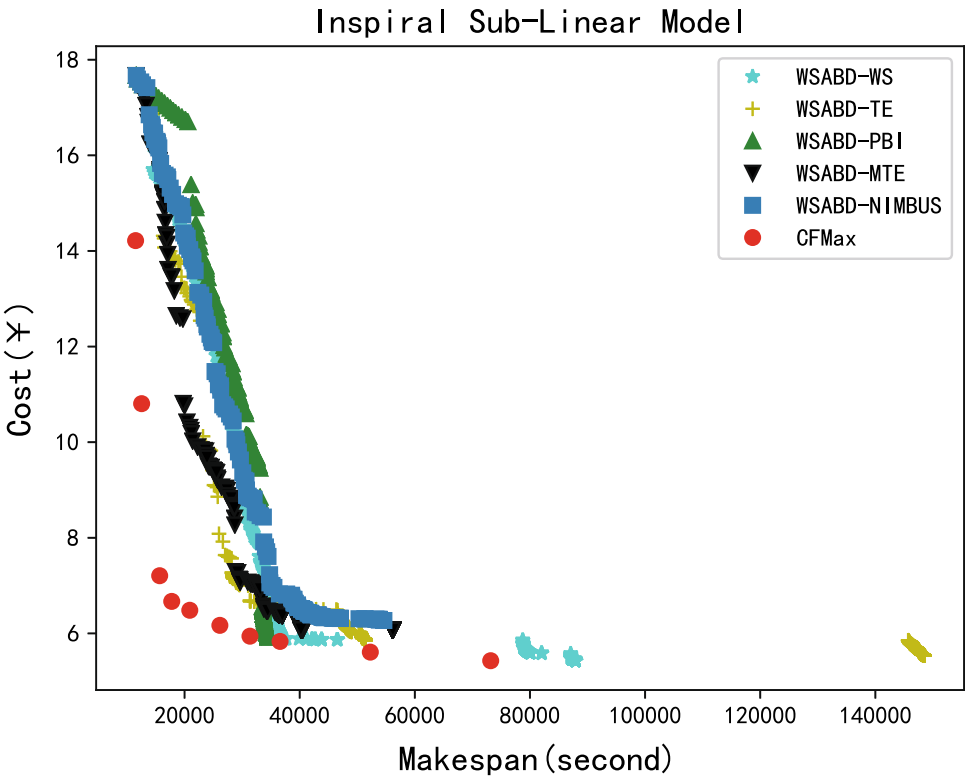

Fig. 9 Inspiral optimization results (Sublinear model)

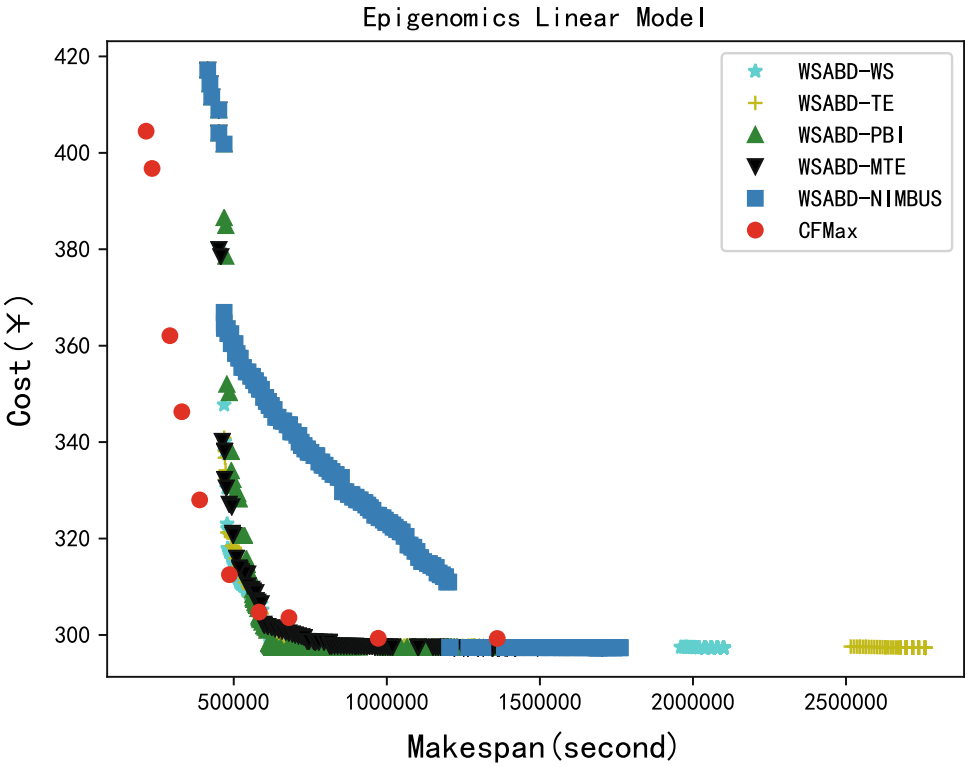

Fig. 10 Epigenomics optimization results (Linear model)

for other best results excluding CFMAX. We tabulated all the results obtained in Table 4 and the best results are in bold.

– The results generated by CFMAX dominate the results generated by our algorithm in 7/9 cases.

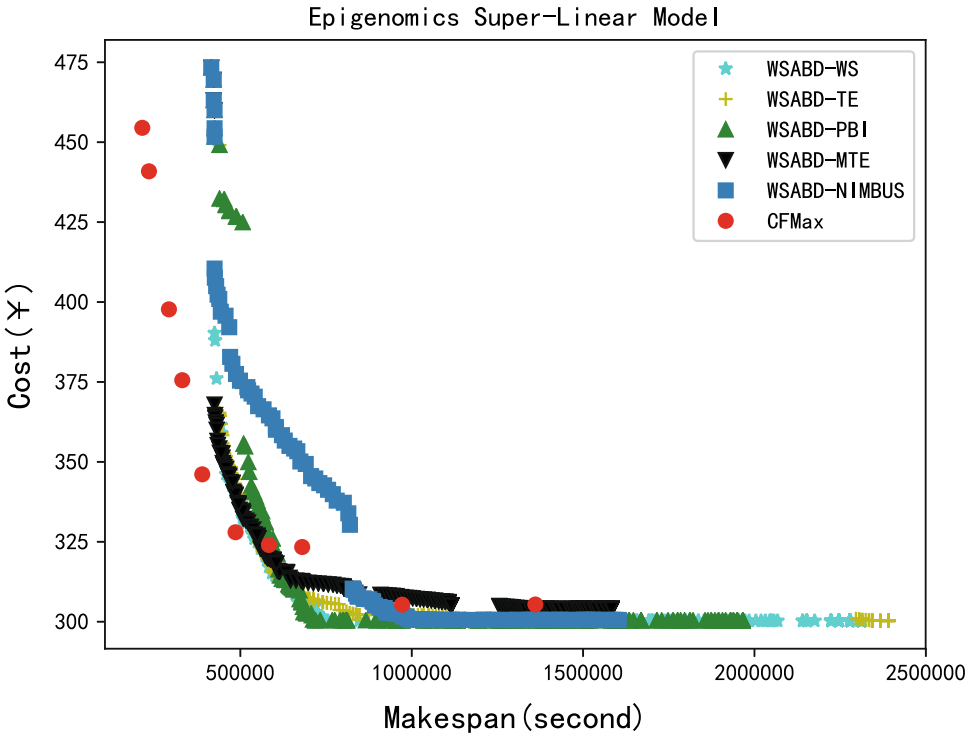

Fig. 11 Epigenomics optimization results (Superlinear model)

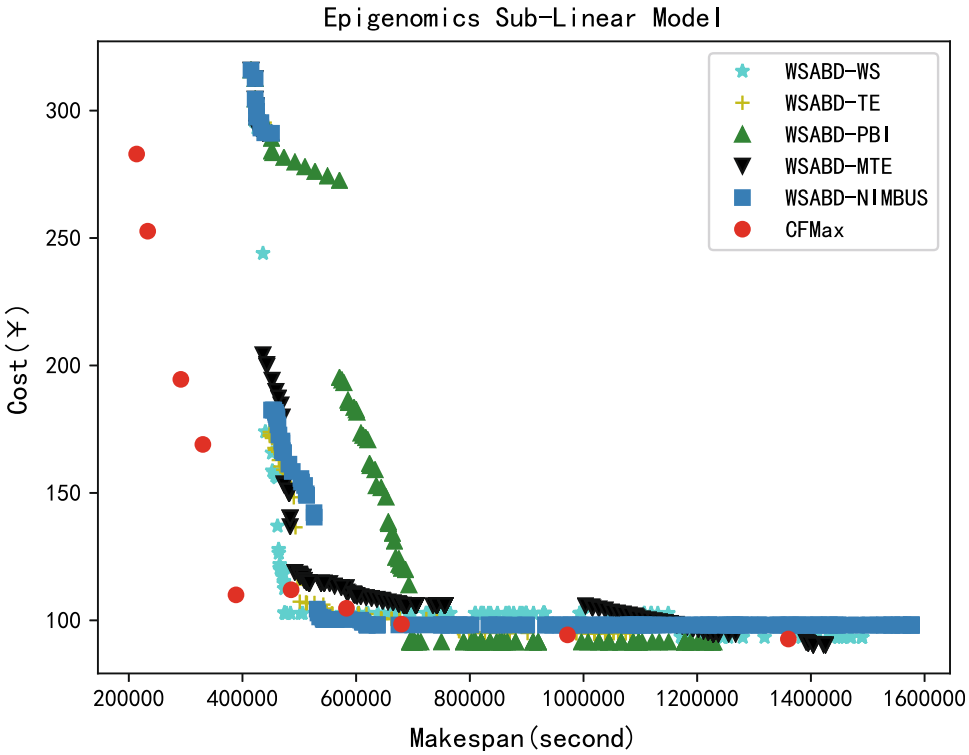

Fig. 12 Epigenomics optimization results (Sublinear model)

– When the results generated by CFMAX are excluded, the results generated by WSABD-TE dominate the results generated by other variants in 4 cases (two cases when DAG is Montage and two more cases when DAG is Inspiral), followed by WSABD-WS (3 cases when DAG is Epigenomics). Both WSABD-TE and WSABD-PBI have equal number of cases (WSABD-TE has one case when DAG is Montage and WSABD-PBI has one case when DAG is Inspiral). WSABD-NIMBUS didn't achieve best solution in any case.

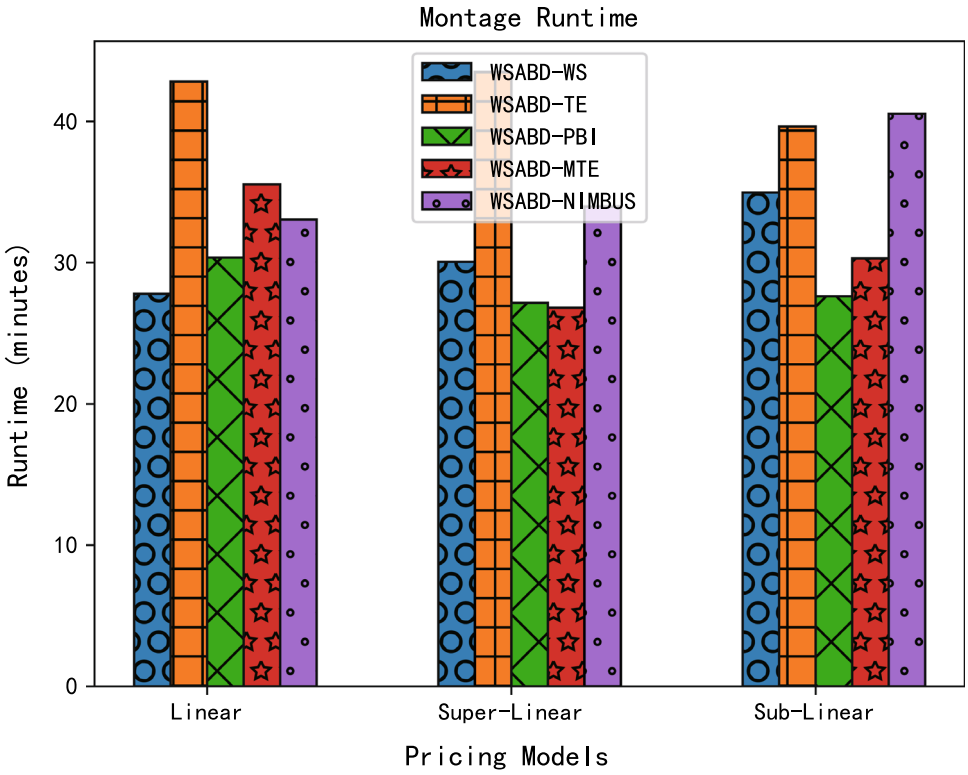

Fig. 13 Runtime results (Montage)

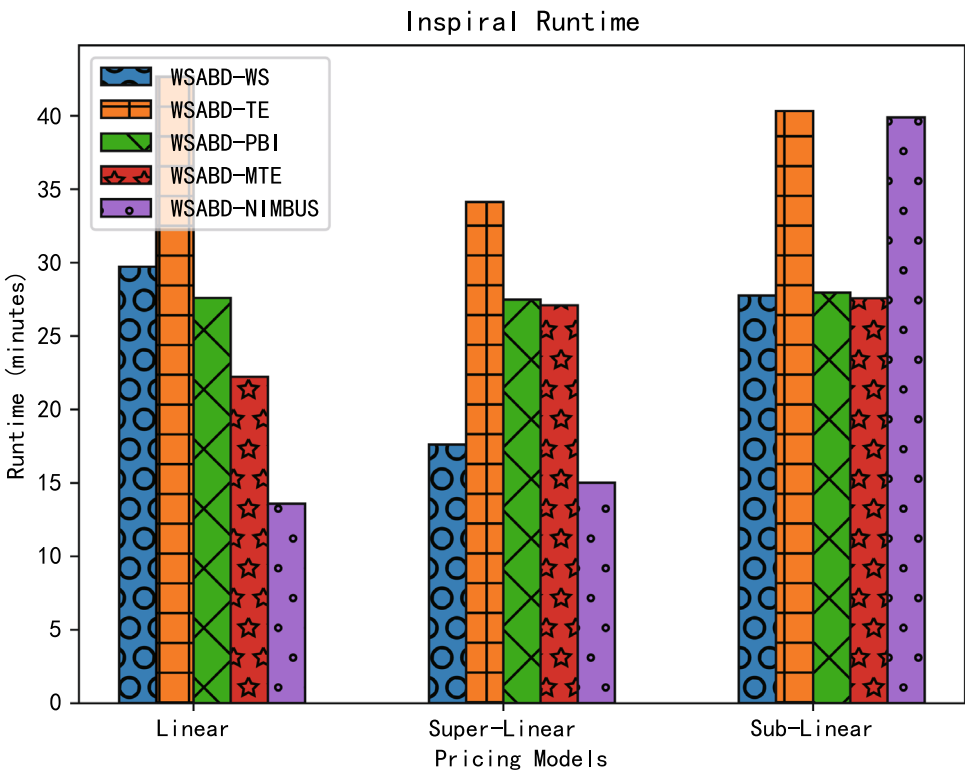

Fig. 14 Runtime results (Inspiral)

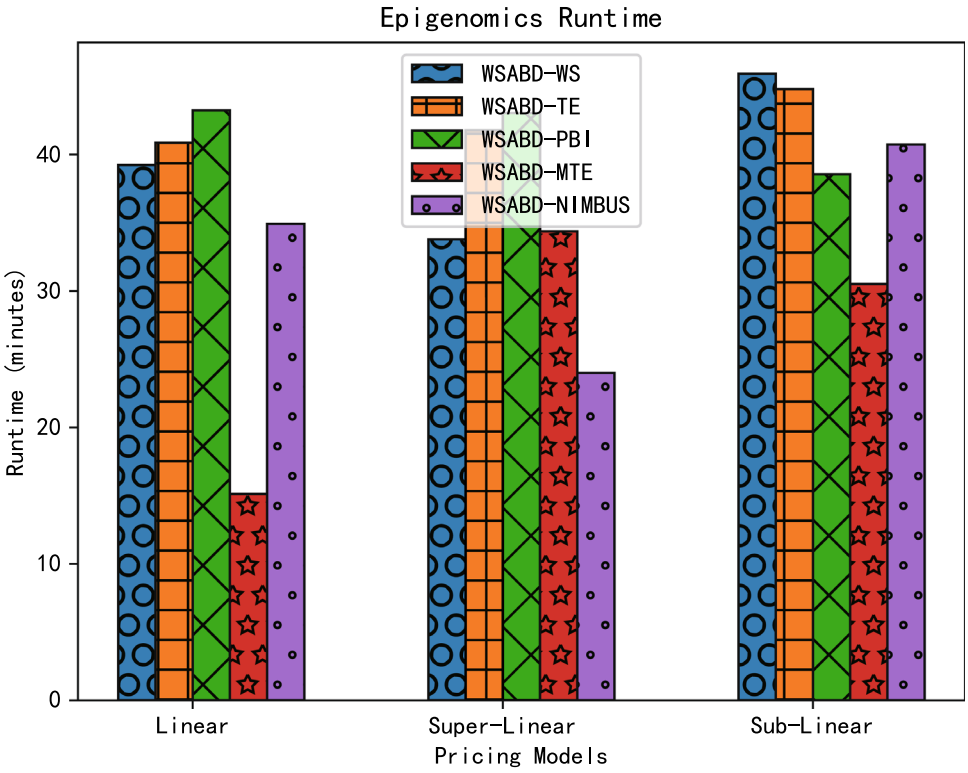

Fig. 15 Runtime results (Epigenomics)

Table 4 HV results

VMs number	DAG	Pricing model	CFMax	WSABD-WS	WSABD-TE	WSABD-PBI	WSABD-MTE	WSABD-NIMBUS
20	Montage	Linear	883.6580748	945.1748272	952.2814573	962.8265572	**971.9816618**	941.9516815
		Superlinear	**3123.68134**	2991.146019	**3002.836243**	2998.052127	2958.494405	2963.747069
		Sublinear	**2424.833044**	2114.245207	2267.72918	2145.37271	**2287.652909**	2209.138739
	Inspiral	Linear	**428303.1905**	395642.8858	394394.1149	**397720.2619**	394918.0159	373603.8355
		Superlinear	**1194755.019**	1100360.469	1105248.901	1108718.658	**1113435.646**	961463.6799
		Sublinear	**1099927.187**	976499.5293	979471.3129	954825.5295	**1014434.873**	961894.0812
	Epigenomics	Linear	**3.15E+08**	**2.98E+08**	2.97E+08	2.96E+08	2.98E+08	2.73E+08
		Superlinear	3.94E+08	**4.06E+08**	4.03E+08	3.98E+08	3.95E+08	3.88E+08
		Sublinear	**4.56E+08**	**4.05E+08**	4.05E+08	3.83E+08	4.01E+08	4.01E+08

Bold is used to highlight the leading HV results

6.4 Variation of other parameter settings

In the second experiment, we used the same parameter settings. However, only one of them will be changed each time while all others remain at their default values as in the previous experiment. We let the number of VMs change from 5 to 35 with a variation step of 5, population size change from 5 to 30 with a variation step of 5, the number of neighbors change from 2 to 7 with a variation step of 1, deadline ratio change from 1.5 to 4 with a variation step of 0.5 and iteration number change from 500 to 3000 with a variation step of 500. The HV results for variation of the number of VMs, variation of population size, variation of the number of neighbors and variation of the number of iterations are presented in Tables 5, 6, 7 and 8 respectively. For each table, we omit the HV results generated by the parameter settings corresponding to those we used in the previous experiment, which can be found in Table 4. Global ranking of the performance of HV results cannot be used to make the final decision about the best variant, because the performance of the algorithm depends on the parameter settings of the environment and the structure of the DAG processed. Therefore, we also analyze HV results of each individual variant, re-rank based on DAG structure.

Table 5 Variation of the number of VMs

VM number	DAG	Pricing model	CFMax	WSABD-WS	WSABD-TE	WSABD-PBI	WSABD-MTE	WSABD-NIMBUS
5	Montage	Linear	1247.084295	1331.774162	1331.594375	1332.769794	**1334.936818**	1212.870389
		Superlinear	1579.335783	**1598.19786**	1574.420988	1544.757726	1568.625954	1534.394471
		Sublinear	2343.289093	2349.954767	2345.160123	2319.480024	**2380.470293**	2220.571114
	Inspiral	Linear	544575.2546	565386.1234	564267.9693	563371.6132	**567490.5052**	527709.9506
		Superlinear	**622235.2558**	**615815.0179**	604150.6882	609200.3701	608607.1543	593304.5555
		Sublinear	1146498.215	**1126936.618**	1108407.286	1103098.19	1077409.642	1006983.177
	Epigenomics	Linear	4.55E+08	4.68E+08	4.69E+08	4.62E+08	**4.69E+08**	4.68E+08
		Superlinear	3.44E+08	3.68E+08	**3.73E+08**	3.60E+08	3.71E+08	3.68E+08
		Sublinear	8.49E+08	8.45E+08	**8.50E+08**	7.82E+08	8.49E+08	8.46E+08
10	Montage	Linear	817.3749734	870.0314652	**880.7707218**	879.2516886	877.7201896	862.8290109
		Superlinear	2401.197162	2370.917929	**2419.20799**	2412.914858	2364.955866	2363.055243
		Sublinear	**1150.10072**	1004.04987	**1093.475487**	1074.932167	1048.907581	1083.424953
	Inspiral	Linear	417343.374	401843.6834	435455.0984	428311.2236	**442193.3487**	413116.6943
		Superlinear	927579.9332	907432.9248	921190.096	911315.146	**934132.2835**	899642.8796
		Sublinear	**640463.7563**	542602.8079	528278.4526	551145.5922	554588.9421	**567511.7769**
	Epigenomics	Linear	**2.62E+08**	2.49E+08	**2.57E+08**	2.49E+08	2.55E+08	2.30E+08
		Superlinear	**3.25E+08**	3.15E+08	3.23E+08	3.10E+08	**3.25E+08**	2.79E+08
		Sublinear	**8.52E+08**	7.49E+08	**7.58E+08**	7.25E+08	7.58E+08	7.18E+08
15	Montage	Linear	2961.954857	2860.979754	2949.46585	**2975.396053**	2970.230368	2960.888966
		Superlinear	**5827.342632**	5279.122252	**5738.927605**	5671.819275	5717.463902	5693.015831
		Sublinear	**2889.434558**	2553.959551	2570.490599	**2624.907426**	2520.411522	2585.641249
	Inspiral	Linear	**1046581.971**	944171.3817	1006686.569	1000297.435	**1007994.738**	998591.1995
		Superlinear	**2020266.922**	1837528.629	1936523.739	1899646.448	1933480.354	**1955569.489**
		Sublinear	**1259329.84**	1061019.924	1062464.059	**1088559.34**	1049582.763	1074929.971
	Epigenomics	Linear	**6.06E+08**	5.36E+08	5.53E+08	5.29E+08	**5.56E+08**	5.49E+08
		Superlinear	**5.42E+08**	4.75E+08	4.90E+08	4.47E+08	**4.96E+08**	4.24E+08
		Sublinear	**1.19E+09**	1.05E+09	**1.05E+09**	1.03E+09	1.05E+09	1.02E+09
25	Montage	Linear	4414.477666	4144.004822	**4433.941367**	4145.395336	4402.309872	3610.920274
		Superlinear	**4237.857051**	3731.262105	**4183.670513**	3597.404424	4163.056544	2690.204646
		Sublinear	**9560.173569**	9206.284594	**9239.23521**	9219.547329	9186.324666	8970.889719
	Inspiral	Linear	1692277.785	1631752.23	**1698485.246**	1572322.941	1679245.759	1311889.926
		Superlinear	**1593085.544**	1540460.673	1570713.261	1320910.099	**1573269.118**	1111451.554
		Sublinear	**3762062.36**	3563251.04	**3639362.112**	3581036.748	3630198.274	3568024.582
	Epigenomics	Linear	4.74E+08	5.50E+08	**5.51E+08**	5.42E+08	5.50E+08	5.31E+08
		Superlinear	3.09E+08	4.74E+08	**4.81E+08**	4.60E+08	4.77E+08	3.83E+08
		Sublinear	9.17E+08	1.04E+09	**1.06E+09**	1.03E+09	1.05E+09	1.01E+09
30	Montage	Linear	**5099.614878**	4947.479908	**5020.113031**	4833.768677	5018.678465	4483.113871
		Superlinear	**5527.008735**	**5362.682444**	5313.070334	5166.430444	5285.642257	5044.792911
		Sublinear	**5147.575707**	4731.989333	4748.439223	4803.845634	**4841.885255**	4712.334711
	Inspiral	Linear	**1168629.64**	1020846.074	1088272.174	1022862.923	**1097265.552**	965468.1694
		Superlinear	**2167845.683**	**2096505.83**	2067228.915	1968980.149	2077915.646	1865157.6
		Sublinear	**2127034.814**	1908153.801	1901642.177	1895052.666	**1938057.106**	1912167.417
	Epigenomics	Linear	**3.52E+08**	3.42E+08	3.42E+08	3.43E+08	**3.43E+08**	3.40E+08
		Superlinear	5.22E+08	**5.40E+08**	5.35E+08	5.37E+08	5.38E+08	5.31E+08
		Sublinear	**9.10E+08**	8.71E+08	8.68E+08	**8.74E+08**	8.61E+08	8.68E+08

Table 5 (continued)

VM number	DAG	Pricing model	CFMax	WSABD-WS	WSABD-TE	WSABD-PBI	WSABD-MTE	WSABD-NIMBUS
35	Montage	Linear	**1337.971372**	1206.179901	1254.350379	**1263.218441**	1246.454869	1205.529621
		Superlinear	**4554.544231**	4287.503439	4253.011201	**4299.82844**	4296.877946	3964.328632
		Sublinear	**3085.152368**	2650.593517	2714.196532	2764.402922	2887.64218	**2892.714978**
	Inspiral	Linear	**493805.0204**	388199.319	428930.6055	**428931.2975**	428923.9006	390933.8616
		Superlinear	**1590869.126**	1373787.974	**1549350.757**	1543306.776	1540391.023	1292495.994
		Sublinear	**1424939.914**	1291413.199	1274342.18	1231318.114	1300694.294	**1305097.688**
	Epigenomics	Linear	**5.58E+08**	5.17E+08	**5.20E+08**	5.15E+08	5.19E+08	4.94E+08
		Superlinear	**5.02E+08**	4.14E+08	**4.31E+08**	4.18E+08	4.24E+08	3.70E+08
		Sublinear	**1.08E+09**	8.45E+08	9.80E+08	8.63E+08	**1.01E+09**	7.63E+08

Bold is used to highlight the leading HV results

6.4.1 Variation of the number of VMs

- The overall HV results for VMs number variation show that each variant can at least achieve the best results in some case.
- The general results are collected for each decomposition approach and ranked as follows: WSABD-TE the first with 21/54 cases, WSABD-MTE the second with 16/54 cases, WSABD-PBI the third with 7/54 cases, WSABD-WS the fourth with 6/54 cases and lastly WSABD-NIMBUS with 4/54 cases. Among 54 cases, CFMAX achieves best results in 35 cases (where 11/18 cases are from Montage, 13/18 cases are from Inspiral and 11/18 cases are from Epigenomics).
- For Montage, the variants are ranked as follows: WSABD-TE the first with 8/18 cases, WSABD-PBI the second with 4/18 cases, WSABD-MTE the third with 3/18 cases, WSABD-WS the fourth with 2/18 cases and finally WSABD-NIMBUS the last with 1/18 cases.
- For Inspiral, the variants are ranked as follows: WSABD-TE the first with 7/18 cases, WSABD-WS, WSABD-TE and WSABD-NIMBUS the second with 3/18 cases each and finally WSABD-PBI the last with 2/18 cases.
- For Epigenomics, the variants are ranked as follows: WSABD-TE the first with 10/18 cases,WSABD-MTE the second with :6/18 cases, WSABD-WS, WSABD-PBI the third with 1/18 case each and finally WSABD-NIMBUS the last with 0/18 case.

6.4.2 Variation of population size

- Based on the general results, variants are ranked as follows: WSABD-WS the first with 15/36 cases, WSABD-TE the second with 9/36 cases, WSABD-MTE the third with 7/36 cases, WSABD-PBI the fourth with 5/36 cases and lastly WSABD-NIMBUS with 0/36 case. Among 36 cases, CFMAX achieves best results in 28 cases (where 8/12 cases are from Montage, 12/12 cases are from Inspiral and 8/12 cases are from Epigenomics).
- For Montage, the variants are ranked as follows: WSABD-WS,WSABD-TE, WSABD-PBI, WSABD-MTE the first with 3/12 cases for each of them and finally WSABD-NIMBUS the last with 0/12 case.
- For Inspiral, the variants are ranked as follows: WSABD-MTE the first with 4/12 cases, WSABD-WS, WSABD-TE the second with 3/12 cases for each, WSABD-PBI the third with 2/12 cases and finally WSABD-NIMBUS the last with 0/12 case.
- For Epigenomics, the variants are ranked as follows: WSABD-WS the first with 9/12 cases, WSABD-TE the second with 3/12 cases, and finally WSABD-PBI, WSABD-MTE and WSABD-NIMBUS the last with 0/12 case for each of them.

6.4.3 Variation of the number of neighbors

- Based on the general results, the variants are ranked as follows: WSABD-MTE the first with 16/45 cases, WSABD-WS the second with 15/45 cases, WSABD-TE the third with 9/45 cases, WSABD-PBI the fourth with 6/45 cases and lastly WSABD-NIMBUS with 0/45 case. Note that both WSABD-WS and WSABD-PBI achieve the best result in one case(when the number of neighbours is 3 for Epigenomics with linear pricing model). Among 45 cases, CFMAX achieves best result in 36 cases (where 9/15 cases are from Montage, 15/15 cases are from Inspiral and 12/15 cases are from Epigenomics).

Table 6 Variation of population size

Population size	DAG	Pricing model	CFMax	WSABD-WS	WSABD-TE	WSABD-PBI	WSABD-MTE	WSABD-NIMBUS
15	Montage	Linear	883.6581	946.0799	940.4845	**962.595**	947.1697	940.7039
		Superlinear	**3123.681**	2970.281	2976.019	**3014.735**	2849.032	2975.399
		Sublinear	**2424.833**	2024.071	2164.851	2050.897	**2257.574**	2255.457
	Inspiral	Linear	**428303.2**	388126.7	387917.1	**398900.7**	390977.9	370856.3
		Superlinear	**1194755**	1097187	1104004	**1112183**	1066077	978949.7
		Sublinear	**1099927**	933906.5	950408.5	907990.1	**984129.4**	937200.3
	Epigenomics	Linear	**3.15E+08**	2.98E+08	**2.98E+08**	2.90E+08	2.97E+08	2.70E+08
		Superlinear	3.94E+08	**4.06E+08**	4.04E+08	3.81E+08	4.01E+08	3.84E+08
		Sublinear	**4.56E+08**	**4.04E+08**	4.04E+08	3.56E+08	3.99E+08	4.00E+08
25	Montage	Linear	2261.237	2337.112	**2337.527**	2332.029	2281.359	2307.218
		Superlinear	**3069.102**	**3018.627**	2952.456	2898.38	2872.828	2883.886
		Sublinear	**2500.781**	2379.387	2392.144	2273.449	**2418.745**	2310.326
	Inspiral	Linear	**425992.3**	**396660**	396378.9	393939.6	395238.4	387715
		Superlinear	**1171308**	1100271	**1105440**	1075313	1059324	1022338
		Sublinear	**1166485**	1021267	1023967	1025877	**1106652**	1028845
	Epigenomics	Linear	**2.21E+08**	2.03E+08	**2.03E+08**	1.91E+08	2.03E+08	1.85E+08
		Superlinear	3.51E+08	**3.61E+08**	3.60E+08	3.32E+08	3.58E+08	3.55E+08
		Sublinear	**7.43E+08**	**6.97E+08**	6.83E+08	6.34E+08	6.91E+08	6.67E+08
30	Montage	Linear	883.6581	958.8003	959.1056	**959.7215**	958.0858	934.4328
		Superlinear	**3123.681**	**3072.5**	3006.892	2951.017	2924.345	2936.537
		Sublinear	**2424.833**	**2304.005**	2317.63	2200.089	2344.32	2238.63
	Inspiral	Linear	**428303.2**	**398970**	398690	396247.3	397506.4	390017.5
		Superlinear	**1194755**	1123590	**1128915**	1098372	1080989	1044219
		Sublinear	**1099927**	958191	957398.4	961597.7	**1042006**	965892
	Epigenomics	Linear	**3.15E+08**	2.97E+08	**2.98E+08**	2.86E+08	2.97E+08	2.79E+08
		Superlinear	3.94E+08	**4.06E+08**	4.05E+08	3.76E+08	4.03E+08	4.00E+08
		Sublinear	**4.56E+08**	**4.10E+08**	4.00E+08	3.49E+08	4.02E+08	3.90E+08
35	Montage	Linear	883.6581	962.2531	**974.5004**	958.4175	967.8415	944.0039
		Superlinear	**3123.681**	3056.053	3032.034	2916.34	**3077.496**	2871.608
		Sublinear	**2424.833**	2278.059	**2361.796**	2232.877	2208.966	2251.576
	Inspiral	Linear	**428303.2**	399660.9	**404355**	394493.6	399492.2	389252.4
		Superlinear	**1194755**	**1158030**	1141958	1089811	1142178	1077164
		Sublinear	**1099927**	1005478	943538.8	962780.1	**1026846**	980392.3
	Epigenomics	Linear	**3.15E+08**	**2.98E+08**	2.97E+08	2.89E+08	2.98E+08	2.77E+08
		Superlinear	3.94E+08	**4.05E+08**	4.03E+08	3.82E+08	4.03E+08	4.01E+08
		Sublinear	**4.56E+08**	**4.09E+08**	3.97E+08	3.65E+08	4.02E+08	3.90E+08

Bold is used to highlight the leading HV results

- For Montage, the variants are ranked as follows: WSABD-MTE the first with 8/15 cases, WSABD-TE the second with 3/15 cases, WSABD-WS and WSABD-PBI the third with 2/15 cases for each of them, and finally WSABD-NIMBUS the last with 0/15 case.
- For Inspiral, the variants are ranked as follows: WSABD-WS, WSABD-TE and WSABD-MTE the first with 4/15 cases for each of them, WSABD-PBI the second with 3/15 cases each and finally WSABD-NIMBUS the last with 0/15 case.
- For Epigenomics, the variants are ranked as follows: WSABD-WS the first with 9/15 cases, WSABD-MTE the second with 4/15 cases, WSABD-TE the third with 2/15 cases, WSABD-PBI the fourth with 1/15 case and finally WSABD-NIMBUS the last with 0/18 case. Note that both WSABD-WS and WSABD-PBI achieve the

Table 7 Variation of the number of neighbors

Neighbors	DAG	Pricing model	CFMax	WSABD-WS	WSABD-TE	WSABD-PBI	WSABD-MTE	WSABD-NIMBUS
2	Montage	Linear	**3040.53132**	2697.280552	**2948.654647**	2732.246694	2853.8945	2394.681853
		Superlinear	**2759.08683**	1908.699626	**2412.42815**	1870.23911	2009.671301	1424.692537
		Sublinear	**8218.84184**	6221.312891	**7031.707494**	6266.861459	6526.449425	5574.150734
	Inspiral	Linear	**1336699.57**	1181193.908	**1244320.503**	1170358.829	1216864.051	1115077.959
		Superlinear	**1223801.33**	788417.1595	**1031857.147**	797840.6922	831862.6803	803636.7236
		Sublinear	**3498478.51**	2577095.922	**3048801.547**	2564464.287	2732370.41	2852250.295
	Epigenomics	Linear	**4.39E+08**	3.87E+08	3.96E+08	3.80E+08	**3.97E+08**	3.91E+08
		Superlinear	**2.72E+08**	2.18E+08	**2.22E+08**	1.80E+08	1.92E+08	2.22E+08
		Sublinear	**1.03E+09**	8.66E+08	**8.68E+08**	7.78E+08	8.05E+08	8.61E+08
3	Montage	Linear	946.434864	**1029.490087**	1014.823159	1024.710356	1011.308631	1010.008642
		Superlinear	**3130.8442**	2984.690854	2989.248121	2854.238075	**3010.185078**	2970.497435
		Sublinear	**2529.84178**	2139.921424	2348.483098	**2392.217321**	2388.17169	2370.863019
	Inspiral	Linear	**426927.354**	**398961.7415**	390587.0772	398031.8257	391146.4367	382131.1995
		Superlinear	**1203271.76**	1107946.499	**1115842.89**	1075316.586	1114661.583	1053146.718
		Sublinear	**1253555.32**	**1168460.331**	1102384.507	1144635.045	1129196.652	1120640.019
	Epigenomics	Linear	**2.13E+08**	**1.95E+08**	1.94E+08	1.95E+08	1.89E+08	1.89E+08
		Superlinear	2.44E+08	**2.48E+08**	2.45E+08	2.44E+08	2.48E+08	2.43E+08
		Sublinear	**2.85E+08**	**2.34E+08**	2.29E+08	2.15E+08	2.29E+08	2.31E+08
4	Montage	Linear	1110.55893	1180.48101	1175.728567	**1190.111283**	1173.49076	1184.895124
		Superlinear	3145.01442	3001.871145	3010.981361	3019.507982	**3030.00758**	2939.576142
		Sublinear	2466.16037	2264.993307	2272.054294	2189.084681	**2303.107632**	2250.299429
	Inspiral	Linear	**459051.484**	421666.5766	421114.7429	**428240.9922**	425797.5249	418423.3825
		Superlinear	**1222987.4**	1137931.868	1128564.267	1135471.927	**1138138.822**	1112831.535
		Sublinear	**1206443.25**	1078749.485	1084737.85	1050926.635	**1109165.156**	1048416.973
	Epigenomics	Linear	**2.29E+08**	2.10E+08	2.10E+08	2.04E+08	**2.12E+08**	2.08E+08
		Superlinear	3.47E+08	**3.56E+08**	3.55E+08	3.48E+08	3.51E+08	3.52E+08
		Sublinear	**5.34E+08**	**4.81E+08**	4.79E+08	4.57E+08	4.76E+08	4.75E+08
6	Montage	Linear	3134.35665	3215.795809	3206.770446	3208.951946	**3215.996905**	3198.308583
		Superlinear	**3162.83676**	**3068.140801**	3046.08706	3031.895568	3022.267191	3017.771168
		Sublinear	**2445.40401**	2170.238152	2280.271262	2158.42557	**2317.430956**	2228.858601
	Inspiral	Linear	**473155.01**	**445152.8263**	441403.0503	442621.3362	438093.9456	422766.313
		Superlinear	**1222983.26**	1134335.786	1126451.555	**1135651.166**	1125929.036	1089597.349
		Sublinear	**1114788.47**	1002299.947	1003071.393	964367.8298	**1015476.533**	985671.2818
	Epigenomics	Linear	**2.04E+08**	1.86E+08	1.86E+08	1.75E+08	**1.87E+08**	1.58E+08
		Superlinear	**3.01E+08**	**3.10E+08**	3.08E+08	2.84E+08	3.04E+08	2.89E+08
		Sublinear	**3.63E+08**	**3.15E+08**	3.08E+08	2.66E+08	3.06E+08	3.08E+08
7	Montage	Linear	954.410295	1022.020154	1021.26312	1033.006167	**1040.571356**	1022.875328
		Superlinear	**3073.43762**	2957.893611	2959.794842	2947.326943	**2960.190193**	2917.250429
		Sublinear	**2504.37396**	2128.098181	2357.25422	2212.654175	**2370.485052**	2308.201677
	Inspiral	Linear	**748322.35**	713761.7613	715261.0099	**718044.848**	715488.0597	696852.6661
		Superlinear	**1222987.4**	**1150691.52**	1128291.34	1136938.403	1136352.44	1009529.753
		Sublinear	**1119034.64**	968026.3072	977332.5364	974127.8329	**1020832.533**	968845.3049
	Epigenomics	Linear	**2.65E+08**	2.47E+08	2.47E+08	2.38E+08	**2.48E+08**	2.19E+08
		Superlinear	4.07E+08	**4.18E+08**	4.16E+08	3.92E+08	4.14E+08	4.08E+08
		Sublinear	**3.93E+08**	**3.38E+08**	3.34E+08	2.98E+08	3.37E+08	3.37E+08

Bold is used to highlight the leading HV results

Table 8 Variation of the number of iterations

Iterations	DAG	Pricing model	CFMax	WSABD-WS	WSABD-TE	WSABD-PBI	WSABD-MTE	WSABD-NIMBUS
500	Montage	Linear	599.3740905	645.6761404	608.9243021	**679.0471441**	629.2245782	641.551717
		Superlinear	**1540.363275**	1132.438952	1112.003655	**1441.25312**	1079.011736	1114.039896
		Sublinear	**2448.101375**	1876.854374	1919.497952	**2168.111257**	1909.319637	1913.945616
	Inspiral	Linear	**384537.6762**	351692.6357	332026.0286	**353976.2256**	338943.9989	340882.8981
		Superlinear	**985692.799**	885689.8978	836745.3736	**901044.7966**	823353.7722	802569.1562
		Sublinear	**1050135.31**	**921088.2103**	895515.7364	906129.5746	906539.6029	898875.473
	Epigenomics	Linear	**3.08E+08**	**2.90E+08**	2.90E+08	2.89E+08	2.90E+08	2.65E+08
		Superlinear	3.84E+08	**3.96E+08**	3.92E+08	3.87E+08	3.85E+08	3.76E+08
		Sublinear	**3.83E+08**	**3.32E+08**	3.32E+08	3.09E+08	3.20E+08	3.29E+08
1000	Montage	Linear	1110.611705	1173.509429	1169.356136	**1189.391551**	1175.979261	1166.488039
		Superlinear	**3123.68134**	2980.536835	2970.725872	**2998.052127**	2885.824027	2948.00095
		Sublinear	**2424.833044**	2114.245207	2232.737602	2145.37271	**2237.635544**	2209.138739
	Inspiral	Linear	**428303.1905**	395642.8858	392376.1649	**397720.2619**	393405.3703	384382.162
		Superlinear	**1194755.019**	1100304.562	1105248.901	**1108718.658**	1104875.358	949301.2155
		Sublinear	**1099927.187**	976499.5293	977469.6956	954825.5295	**1009623.671**	961771.8474
	Epigenomics	Linear	**3.15E+08**	**2.98E+08**	2.97E+08	2.96E+08	2.98E+08	2.72E+08
		Superlinear	3.94E+08	**4.06E+08**	4.03E+08	3.98E+08	3.95E+08	3.88E+08
		Sublinear	4.56E+08	**4.05E+08**	4.05E+08	3.83E+08	4.01E+08	4.01E+08
1500	Montage	Linear	932.5621587	994.7965555	1001.215471	**1011.646898**	1007.917859	990.7716783
		Superlinear	**3123.68134**	2983.702381	**3002.836243**	2998.052127	2976.58609	2947.853385
		Sublinear	**2424.833044**	2114.245207	2267.72918	2145.37271	**2327.061524**	2209.138739
	Inspiral	Linear	**428303.1905**	395642.8858	394394.1149	**397720.2619**	394919.6828	375874.31
		Superlinear	**1194755.019**	1100309.06	1105248.901	**1108718.658**	1105380.218	981304.7862
		Sublinear	**1099927.187**	976499.5293	979471.3129	954825.5295	**1010063.677**	961808.7982
	Epigenomics	Linear	**3.15E+08**	**2.98E+08**	2.97E+08	2.96E+08	2.98E+08	2.73E+08
		Superlinear	3.94E+08	**4.06E+08**	4.03E+08	3.98E+08	3.95E+08	3.88E+08
		Sublinear	**4.56E+08**	**4.05E+08**	4.05E+08	3.83E+08	3.99E+08	4.01E+08
2500	Montage	Linear	907.8168617	969.7367758	976.4550295	**986.9439745**	986.118392	966.180143
		Superlinear	**3123.68134**	2991.807938	3002.836243	2998.052127	**3006.657693**	2962.363467
		Sublinear	**2424.833044**	2114.245207	2267.72918	2145.37271	**2310.521636**	2209.138739
	Inspiral	Linear	**428303.1905**	395642.8858	394394.1149	**397720.2619**	394979.5101	383424.4574
		Superlinear	**1194755.019**	1100465.336	1105248.901	1108718.658	**1122584.305**	961982.7268
		Sublinear	**1099927.187**	976499.5293	979471.3129	954825.5295	**1014096.155**	961648.1593
	Epigenomics	Linear	**3.15E+08**	**2.98E+08**	2.97E+08	2.96E+08	2.98E+08	2.73E+08
		Superlinear	3.94E+08	**4.06E+08**	4.03E+08	3.98E+08	3.95E+08	3.88E+08
		Sublinear	**4.56E+08**	**4.05E+08**	4.05E+08	3.83E+08	4.03E+08	4.01E+08
3000	Montage	Linear	950.7167304	1012.75331	1019.381153	**1029.770381**	1029.286097	1009.969241
		Superlinear	**3123.68134**	2992.767981	**3002.836243**	2998.052127	2993.490941	2966.301806
		Sublinear	**2424.833044**	2114.245207	2267.72918	2145.37271	**2270.05881**	2209.138739
	Inspiral	Linear	**428303.1905**	395662.5404	394394.1149	**397720.2619**	394804.1418	373939.2554
		Superlinear	**1194755.019**	1100465.336	1105248.901	1108718.658	**1131151.106**	965240.6034
		Sublinear	**1099927.187**	976499.5293	979471.3129	954825.5295	**1015524.828**	962223.8243
	Epigenomics	Linear	**3.15E+08**	**2.98E+08**	2.97E+08	2.96E+08	2.98E+08	2.73E+08
		Superlinear	3.94E+08	**4.06E+08**	4.03E+08	3.98E+08	3.95E+08	3.88E+08
		Sublinear	**4.56E+08**	**4.05E+08**	4.05E+08	3.83E+08	4.03E+08	4.01E+08

Bold is used to highlight the leading HV results

best result when the number of neighbours is 3 under linear pricing model.

6.4.4 Variation of the number of iterations

- Based on the general results the variants are ranked as follows: WSABD-WS and WSABD-PBI the first with 16/45 cases for each of them, WSABD-MTE the third with 12/45 cases, WSABD-TE the fourth with 2/45 cases and lastly WSABD-NIMBUS with 0/45 case.-Note that both WSABD-WS and WSABD-MTE achieve the best result in one case(when the number of iterations is 2500 for Epigenomics with linear pricing model). Among 36 cases, CFMAX achieves best result in 28 cases (where 8/12 cases are from Montage, 12/12 cases are from Inspiral and 8/12 cases are from Epigenomics).
- For Montage, the variants are ranked as follows: WSABD-PBI the first with 8/15 cases, WSABD-MTE the second with 5/15 cases, WSABD-TE the third with 2/15 cases, finally WSABD-WS, WSABD-NIMBUS the last with 0/15 case.
- For Inspiral, the variants are ranked as follows: WSABD-PBI the first with 8/15 cases, WSABD-MTE the second with 6/15 cases, WSABD-WS the third with 1/15 case, finally WSABD-TE, WSABD-NIMBUS the last with 0/15 case.
- For Epigenomics, the variants are ranked as follows: WSABD-WS the first with 15/15 cases WSABD-MTE the second with 1/15 case and finally WSABD-TE, WSABD-PBI and WSABD-NIMBUS the last with 0/15 case.Note that both WSABD-WS and WSABD-MTE achieve the best result when the number of iterations is 2500 under linear pricing model.

7 Conclusion

In this paper, the problem of minimizing the makespan and monetary cost of a submitted workflow is considered and modeled as a multi-objective optimization problem. A novel workflow scheduling algorithm based on decomposition is proposed to assist in the tuning of the CPU frequency for each task so that both makespan and cost can be minimized. The evaluation results on optimization show that in different conditions, all the variants of the proposed algorithm can at least perform well in some cases. And the runtime evaluation results show that different parameter settings cause runtime variability for all the tested cases. However, the proposed algorithm still has room for further improvements. The use of cloud and/or cluster computing requires the optimization of more than two objectives at the same time. On one hand, multiple objectives have to be considered to further test the capability of the proposed algorithm. On the other hand, the algorithm complexity shall be lowered to provide better scalability. Future works could consider different algorithms to initialize the population of the proposed algorithm besides CFMAX. Additionally, the efficiency of the proposed algorithm could be tested under a real cloud platform.

Funding This work was supported by the National Science Foundation of Fujian Province of China (No. 2018J01107), and was also jointly supported by the National Natural Science Foundation of China (NSFC, Grant No. 61672439).

Compliance with ethical standards

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

References

1. Hu, Z., Li, D., Guo, D.: Balance resource allocation for spark jobs based on prediction of the optimal resource. Tsinghua Sci. Technol. **25**(04), 487–497 (2020)
2. Garey, M.R., Johnson, D.S.: Computers and Intractability. A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York (1990)
3. Cloudsigma.: Cloudsigma. (2009). https://www.cloudsigma.com/ , Accessed 27 Jan 2020
4. Elasticshosts.: Elastichosts. (2008). https://www.elastichosts.com/. Accessed 27 Jan 2020
5. Pietri, I., Sakellariou, R.: Cost-efficient cpu provisioning for scientific workflows on clouds. In: Altmann, J., Silaghi, G.C., Rana, O.F. (eds.) Economics of Grids, Clouds, Systems, and Services. Springer International Publishing, Cham (2016)
6. Zhang, Q., Li, H.: Moea/d: a multiobjective evolutionary algorithm based on decomposition. IEEE Trans. Evolut. Comput. **11**(6), 712–731 (2007)
7. Alla, H.B., Alla, S.B., Touhafi, A., Ezzati, A.: A novel task scheduling approach based on dynamic queues and hybrid metaheuristic algorithms for cloud computing environment. Clust. Comput. **21**(3), 1797–1820 (2018)
8. Hosseinzadeh, M., Ghafour, M.Y., Hama, H.K., Vo, B., Khoshnevis, A.: Multi-objective task and workflow scheduling approaches in cloud computing: a comprehensive review. J. Grid Comput. **18**, 327–356 (2020)
9. Abazari, F., Analoui, M., Takabi, H., Fu, S.: Mows: multi-objective workflow scheduling in cloud computing based on heuristic algorithm. Simul. Modell. Pract. Theory **93**, 119–132 (2019)
10. Hu, H., Li, Z., Hu, H., Chen, J., Ge, J., Li, C., Chang, V.: Multi-objective scheduling for scientific workflow in multicloud environment. J. Netw. Comput. Appl. **114**, 108–122 (2018)
11. Zhou, X., Zhang, G., Sun, J., Zhou, J., Wei, T., Hu, S.: Minimizing cost and makespan for workflow scheduling in cloud using fuzzy dominance sort based heft. Future Gener. Comput. Syst. **93**, 278–289 (2019)
12. Bugingo, E., Zheng, W., Zhang, D., Qin, Y., Zhang, D.: (2019) Decomposition based multi-objective workflow scheduling for

cloud environments. In: 2019 Seventh International Conference on Advanced Cloud and Big Data (CBD), pp. 37–42
13. Iranmanesh, A., Naji, H.R.: DCHG-TS: a deadline-constrained and cost-effective hybrid genetic algorithm for scientific workflow scheduling in cloud computing. Clust. Comput. (2019). https://doi.org/10.1007/s10586-020-03145-8
14. Emmanuel, B., Qin, Y., Wang, J., Zhang, D., Zheng, W.: Cost optimization heuristics for deadline constrained workflow scheduling on clouds and their comparative evaluation. Concurr. Comput. **30**(20), e4762 (2018)
15. Topcuoglu, H., Hariri, S., Min-You, Wu: Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Trans. Parall. Distribut. Syst. **13**(3), 260–274 (2002)
16. Ahmad, W., Alam, B., Ahuja, S., Malik, S.: A dynamic VM provisioning and de-provisioning based cost-efficient deadline-aware scheduling algorithm for Big Data workflow applications in a cloud environment. Clust. Comput. (2020). https://doi.org/10.1007/s10586-020-03100-7
17. Ijaz, S., Munir, E.U.: MOPT: list-based heuristic for scheduling workflows in cloud environment. J. Supercomput. **75**(7), 3740–3768 (2020)
18. Zhou, N., Lin, W., Feng, W., Shi, F., Pang, X.: Budget-deadline constrained approach for scientific workflows scheduling in a cloud environment. Clust. Comput. (2020). https://doi.org/10.1007/s10586-020-03176-1
19. Zheng, W., Qin, Y., Bugingo, E., Zhang, D., Chen, J.: Cost optimization for deadline-aware scheduling of big-data processing jobs on clouds. Future Gener. Comput. Syst. **82**, 244–255 (2018)
20. Choudhary, A., Gupta, I., Singh, V., Jana, P.K.: A GSA based hybrid algorithm for bi-objective workflow scheduling in cloud computing. Future Gener. Comput. Syst. **83**, 14–26 (2018)
21. Xue, C., Lin, C., Hu, J.: Scalability analysis of request scheduling in cloud computing. Tsinghua Sci. Technol. **24**(03), 249–261 (2019)
22. Zhang, H., Xie, J., Ge, J., Shi, J., Zhang, Z.: Hybrid particle swarm optimization algorithm based on entropy theory for solving DAR scheduling problem. Tsinghua Sci. Technol. **24**(03), 281–290 (2019)
23. Zhang, M., Li, H., Liu, L., Buyya, R.: An adaptive multi-objective evolutionary algorithm for constrained workflow scheduling in clouds. Distribut. Parall. Databases **36**(2), 339–368 (2018)
24. Singh, V., Gupta, I., Jana, P.K.: An energy efficient algorithm for workflow scheduling in IAAS cloud. J. Grid Comput. **18**, 357–376 (2020)
25. Li, F., Liu, J., Huang, P., Shi, H.: (2018) An indicator and decomposition based steady-state evolutionary algorithm for many-objective optimization. Math. Probl. Eng. (2018)
26. Miettinen, K., Mustajoki, J., Stewart, T.J.: Interactive multiobjective optimization with nimbus for decision making under uncertainty. OR Spectrum **36**(1), 39–56 (2014)
27. Miettinen, K., Mäkelä, M.M.: Synchronous approach in interactive multiobjective optimization. Eur. J. Operat. Res. **170**(3), 909–922 (2006)
28. Zheng, W., Emmanuel, B., Wang, C., Qin ,Y., Zhang, D.: Cost optimization for scheduling scientific workflows on clouds under deadline constraints. In: 2017 Fifth International Conference on Advanced Cloud and Big Data (CBD), pp. 51–56 (2017)
29. Juve, G.: Workflowgenerator. (2014). https://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator. Accessed 27 Jan 2020
30. Juve, G., Chervenak, A., Deelman, E., Bharathi, S., Mehta, G., Vahi, K.: Characterizing and profiling scientific workflows. Future Gener. Comput. Syst. **29**(3), 682–692 (2013b)
31. Sun, T., Xiao, C., Xu, X.: A scheduling algorithm using sub-deadline for workflow applications under budget and deadline constrained. Clust. Comput. **22**(3), 5987–5996 (2019)

Emmanuel Bugingo received his Masters degree in Computer Science from XiamenUniversity in 2016. He is now a Ph.D. student in the Department of ComputerScience, Xiamen University. His research interests include cloud computing,big data processing and workflow scheduling.

Defu Zhang is a Professor of Computer Science Department in Xiamen University. His current research interests include Cloud Computing, big data, computational intelligence, and data mining. He has published over 40 journal articles.

Zhaobin Chen received his B.Sc degree in Computer Science from Xiamen University in 2018. He is now a Masters student in the Department of Computer Science, Xiamen University. His research interests include cloud computing, system modelling and scheduling algorithms.

Wei Zheng received his Ph.D. degree in Computer Science from the University of Manchester in 2010. He received a B.Sc. and a Masters degree in Computer Science from Tsinghua University in 2002 and 2005, respectively. He is now an Associate Professor at Xiamen University. His research interests include scheduling, performance modelling and distributed computing.