Indonesian Journal of Electrical Engineering and Computer Science
Vol. 25, No. 3, March 2022, pp. 1615~1624
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v25.i3.pp1615-1624

Optimized scheduling of scientific workflows based on iterated local search

Alaa Abdalqahar Jihad[1], Sufyan T. Faraj Al-Janabi[2], Esam Taha Yassen[1]
[1]Computer Center, University of Anbar, Ramadi, Iraq
[2]College of Computer Science and Information Technology, University of Anbar, Ramadi, Iraq

Article Info

Article history:

Received Jul 15, 2021
Revised Jan 13, 2022
Accepted Jan 21, 2022

Keywords:

Cloud computing
Optimization
Quality of service
Scheduling
Scientific workflows

ABSTRACT

Recent years have witnessed a great interest in scientific applications with large data and processing-intensive, so cloud computing is used which provides the resources needed to implement and run these applications. One of the challenges in the management of scientific workflow applications is scheduling them to solve many combinatorial optimization problems, including reducing execution time, cost, resource utilization, and energy consumption. Due to the fact that the iterated local search algorithm (ILS) has been successfully applied to solve many combinatorial optimization problems, this paper investigates the performance of ILS in solving the scientific workflow scheduling problem which is a highly constrained problem. The main components that are different from one problem to others are the ILS parameters, local search, and perturbation, which must be carefully designed. The performance of the standard ILS has been examined and compared with the latest technology. The experimental results show that the proposed algorithm (ILS) obtained good results compared to the best-known results in the literature. This is due to the ILS being an adaptable metaheuristic, which can be simply adapted to different search situations and instances.

Corresponding Author:

Alaa Abdalqahar Jihad
Computer Center, University of Anbar
Ramadi, Iraq
Email: it.alaa.heety@uoanbar.edu.iq

1. INTRODUCTION

In recent times, there has been a lot of use of big data applications that combine thousands of interrelated tasks with precedence constraints. These complex implementations are considered as a directed acyclic graphs (DAGs) model [1]. The scientific workflow model is widely used in many fields to describe various scientific problems such as bioinformatics, astroinformatics, and geoinformatics. The scientific workflows are data-intensive, computation-intensive, and require many processing hours [2].

To handle scientific applications which are increasingly data-intensive, computational resources that aid in parallel execution, such as grids, clusters, and clouds, are used. Cloud computing is the latest trend in scalable distributed computing, which makes available technology resources at an adaptive price for use, on-demand, and over the Internet. This can be an interesting alternative instead of buying and owning physical data centers [3], [4].

When useing the cloud computing environment, various challenges must be dealt with to implement a scientific workflow application. One of these is scheduling, which is the process of allocating resources to tasks to improve one or more goals [5]. Since DAG scheduling is an NP-complete problem, one of the main problems is to find the optimal schedule [6]. Thus, some challenges need to be taken into consideration, such as the performance variation of virtual machines (VMs) and their common and heterogeneous nature [7].

This paper proposes implementation of the iterated local search (ILS) algorithm to solve the scientific workflow scheduling problem to reduce makespan. We analyze its performance, adjust its parameters, and present a comparison of the results obtained by the proposed algorithm with some other known algorithms. The results show that the ILS algorithm is very flexible and offering many implementation options to the developers in this respect.

The remaining of this paper is organized as follows: section 2 reviews some of the related literature. Next, the research method of this work is presented in section 3. Then, section 4 describes the performance evaluation of the proposed algorithm. Section 5 contains the experimental results of the work along with the discussions. Finally, the paper is concluded in section 6.

2. RELATED WORKS

This section provides a brief overview of the related work in which metaheuristics have been used in scheduling scientific workflows. Indeed, it includes a brief presentation of the problems and the proposed improvements to the ILS algorithm. Metaheuristics are algorithms that can be used to solve a variety of optimization issues. There are many metaheuristics based optimization approaches for scheduling of workflow. Hu *et al.* [8] suggested an algorithm of multiobjective scheduling (MOS) based on particle swarm optimization (PSO) that aims to decrease the cost and makespan and satisfying terms of the reliability. Also, Manasrah and Ali [9] designed a GA-PSO hybridized algorithm that targets to decrease the makespan and cost, as well as balancing the load of contingent tasks. This algorithm is allocating tasks to the resources. Furthermore, Song *et al.* [10] introduced a workflow model with composite tasks. They developed a nested particle swarm optimization which uses two types of population (the outer populations and the inner populations). The outer populations are improving the scheduling instruction of tasks, and the inner populations are enhancing the service instances mission. Maio and Kimovski [11] proposed a multi-objective workflow offloading (MOWO) algorithm based on the NSGA-II metaheuristic, with the goal of reducing response time, efficiency, and expense. Their strategy is based on the pareto principle. Also, Ma *et al.* [12] offered a deadline and the cost aware genetic optimization algorithm, aiming to reduce the cost of execution with terms of deadline. First, they divided the tasks into different levels. After that, they generated individuals with minimal time and cost. To accurately represent the cloud's heterogeneous and robust properties, three strings were used to code the genes in the proposed algorithm.

Faragardi *et al.* [13] introduced a heterogeneous earliest finish time (HEFT) modification and a greedy resource provisioning (GRP) algorithm, that organizes the instance types into groups based on their performance. The aim was to minimize the makespan while taking into consideration a budget limitation for the cost-per-hour. They adjusted the HEFT algorithm to consider the budget limit. Finally, in the work by Adhikari *et al.* [14] the workload of cloud servers, makespan, resource usage, and stability were the goals of a workflow scheduling approach based on the firefly algorithm (FA). Concering the ILS algorithm, ILS has been used in many fields including, the travelling salesman problem [15], [16], vehicle routing problem [17], [18], flow-shop problem [19], [20], task scheduling on cloud computing [21], precedence-constraint task list scheduling [22], the parallel machine scheduling problems [23], and the quadratic assignment problem [24]. Several enhancements were proposed over time including, using clusters and a modified multi-restart [15], using multi types of neighborhoods moves [17], ILS memory-based [18], use of a biased-randomized [20], [25], and hybrid with other metaheuristics algorithms [21], [22]. In this paper, the ILS algorithm is implemented for scientific workflow scheduling, and according to the authors' knowledge, this implementation is new in this domain. In addition, the performance of the algorithm is analyzed by modifying important parameters in the algorithm. The goal of this work is to reduce the makespan.

3. RESEARCH METHOD

In this section, system models will be described to illustrate the working environment. The cloud and workflow models will be described, which are necessary to visualize and understand the proposed work model. Then we explain the iterated local search algorithm and its steps, including the local search algorithm.

3.1. System modeling

3.1.1. Cloud model

The cloud model consists of data centers, each data center consists of several physical machines. Each resource has processing capacity, storage, memory, and bandwidth. The cloud provides different types of VMs. The configuration of the VM type varies concerning the performance of the CPU, memory, storage, bandwidth, and operating system. The price of the VM is the cost per unit time interval. It should be noted that VMs can be acquired and terminated at any time. Workflows can be scheduled for any of the available resources.

3.1.2. Workflow model

The workflow (W) consists of a set of computational tasks with dependency constraints between them [26] and is represented as a DAG. It is defined by two sets W(T, E), where T is the set of n-tasks, and E is the set of dependencies between these tasks. The task starts when you receive input data from previous tasks, and ends when you send output data to subsequent tasks. The task that has no previous task is called the input task, and the task that does not have any subsequent task is called the exit task. Each Ti task in the workflow has length, required processing elements, deadline, transfer file size, lists for parent and child tasks. The execution time of a task depends on its length (in MI) and the performance of the VM (in MIPS) that is used to perform the task. For illustration, Figure 1(a) is an example of a workflow with n=5 tasks. If task T5 is the last task, we need to finish processing T3 and T4 and then combine their outputs as input for task T5 to be processed, task T4 does not process until task T2 is completed, and so on. The task is not implemented until all the list of parents are processed and their outputs are received. Every task that is handled sends its outputs to its list of childs. Indeed, each workflow task has the followings: i) Task execution time: For a given task, the time required to execute that task within the VM; ii) Task start time: The start time of the task inside the VM is calculated by:

$$ST(t_i) = \begin{cases} o & if\ t_i\ is\ input\ task \\ \max\{AvaiTime(VMt_i), \max\{FT(t_j)\}\} & otherwise \end{cases} \quad (1)$$

where $t_j \in$ parent(t_i), and AvaiTime(VMt_i) is the earliest time when VMt_i ready to excut any task; and iii) Task finish time: It is the start time of the task with the time the task is executed. The time available for the VM will also be updated at this time, calculated by (2):

$$FT(t_i) = ST(t_i) + ET(t_i) + \sum TT(t_j, t_i) \quad (2)$$

where $ET(t_i)$ is execution time, $TT(t_j, t_i)$ is transfer cost between t_j and t_i, and $t_j \in parent(t_i)$.

3.2. Iterated local search

The ILS algorithm is one of the metaheuristic algorithms that process one solution at a time (single solution based); in this paper the proposed ILS is illustrated in Algorithm 1, has a modular nature that makes it adaptable as a basic template for designing algorithms [27].

Algorithm 1. Iterated local search

```
Input: Tasks of workflow (T) and available virtual machines (VMs)
Output: The best schedule (Schedule) for assigning VMs to T
1:  Parameters initializing: number of non improvement iteration NonItr, number of
neighbors N, and perturbation ratio Pr.
2:  S_current = Generate a random initial solution
3:  S_current = LocalSearch(S_current, N)
4:  S_best = S_current
5:  i=1
6:  While i< NonItr do
7:      S_perturbation = Perturbation(S_current, Pr)
8:      S_perturbation = LocalSearch(S_perturbation, N)
9:      if f(S_perturbation) < f(S_best)
10:             S_best= S_perturbation
11:              i=1
12:     else
13:             i=i+1
14:     S_current  = AcceptanceCriterion(S_current, S_perturbation)
15: end while
```

There are four main components of the ILS algorithm to consider [28]: i) Initial solution: generate an empty schedule (*Schedule*) where the length of schedule equal to number of tasks and each index represent the task id (t_i). Randomly assign *VMs* for each task (t_i) calculate the cost (Makespan) of the generated solution; ii) Local search: in order to invest the search space, the local search algorithm (Hill clamping) is adopted. Hill-climbing is perhaps the simplest and oldest metaheuristic method. It starts with a given initial solution. At each iteration, the heuristic is replaced the current solution with an adjacent solution that is better than the current one that improves the objective function [29].

Algorithm 2 illustrated the main steps of the utilized local search, in this algorithm the initial solution (S_0) comes from the steps of the ILS; iii) Perturbation: it is the process of selected and changed a part of the solution to escape from the optimum solution, the perturbations should be made in a proportion that is not too large and at the same time not too small [27]; iv) Acceptance criterion: it is select the solution that

will continue in the next iteration; and v) The solution is represented as in Figure 1(b), where the location refers to the task number and the cell data refers to the number of the VM assigned to it.

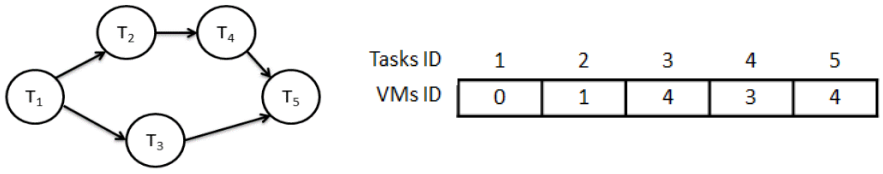


Figure 1. The workflow model and how it is represented in (a) workflow with five tasks and (b) representation of the solution

Algorithm 2. Local search

```
Input: initial solution (S0), number of neighbors N // S0 Solution from ILS Steps
Output: The best solution Sbest (local optima).
Scurrent = S0
NoImprovement = 0
While NoImprovement = 0Do
    NoImprovement = 1
    Neighbors(Scurrent ) ←Generate N candidate solutions (Sc) ∈ Neighbors of Scurrent
    for each Sc in Neighbors (Scurrent )
            if f(Sc) < f(Scurrent)
            Scurrent = Sc
            NoImprovement = 0
    Endfor
Endwhile
```

4. PERFORMANCE EVALUATION

In this section, we explore the details of the relevant performance evaluation parameters and explain how they are calculated and adjusted in the various considered scenarios. The experimental setup and metric that was used in the evaluation will be explained. In addition to explaining the calculation of makespan algorithm.

4.1. Experimental setup

To compare the performance of the ILS algorithm, WorkflowSim simulator [30] has been used for the implementation. WorkflowSim is an improvement for CloudSim to simulate scientific workflow in cloud computing. We have taken measures on personal computer with the following configuration: Intel Core i7 2.10 GHz 16 GB memory, run on Windows 10. The different experiments are the result of the application of several VM allocation. Five VMs were used in the test. Table 1 presents the specifications of VMs used in the simulation. The proposed algorithm is evaluated by using four types of realistic workflows, namely Montage, CyberShake, Epigenomics, Inspiral, and for several instances.

Table 1. The specifications of VMs

VMs	RAM	Bandwidth	MIPS	No. CPU
0	512	800	800	1
1	512	900	900	1
2	512	500	500	1
3	512	600	600	1
4	512	700	700	1

4.2. Evaluation metric

The main metric used to study the performance of the proposed algorithm is makespan. Makespan is defined as the completion time of the output task to end the application [29]. Makespan can also be defined as the schedule length for executing the workflow, also known as the deadline [6]. Makespan is calculated by (3).

$$\text{Makespan} = \text{Max}\{\text{FT}(t_i)\} \text{ where } t_i \in \text{Tasks} \quad (3)$$

The time or cost of transferring data TT_{ij} between any two tasks such as t_i and t_j in a workflow, where t_i is parent to t_j, depends on the amount of data sent from t_i to t_j, and the network bandwidth. TT_{ij} is zero if t_i and t_j are assigned to the same VM [31]. The data transfer time is calculated as follows [32]:

$$\text{TT}_{ij} = \begin{cases} data_{ij}/bw & if\ VMt_i \neq VMt_j \\ 0 & otherwise \end{cases}$$

where $data_{ij}$ the size of the data which needs to be transferred from the t_i to t_j, and bw is the communication bandwidth between the VMt_i and VMt_j. Algorithm 3 illstrates the method which is adopted to calculate the makespan.

Algorithm 3. Calculation of makespan

```
Input: Solution S (Schedule of tasks workflow)
Output: Makespan of solution S
1:  vmNum = number of available VMs, taskNum = number of Tasks
2:  k=0, MaxTime=0, Ready=0, TransferCost=0, Makespan=0
3:  While k<=taskNum do
4:      For each task (ti) in Task List (TL) do
5:              If ti is not handled
6:                      TransferCost=0, MaxTime=0, Ready=1
7:                      For each parent of ti (Pti) do //where Pti ∈ Parent List (PL)ti
8:                              If Pti is not handled
9:                                      Ready=0
10:                             else
11:                                     MaxTime= Max(MaxTime, FT(Pti))
12:                     EndFor
13:             If Ready=1
14:                     For each Pti in (PL)ti do
15:                             If VMti <> VMPti
16:                                  TransferCost=TransferCost+ TT(Pti, ti)
17:                     EndFor
18:                     ST(ti) = Max(MaxTime, AvaiTime(VMti)) + TransferCost
                        // where AvaiTime(VMti) is Available Time of VMti
19:                     FT(ti) = ST(ti) + (Length of ti / Mips of VMti)
20:                     Makespan=Max(Makespan, FT(ti))
21:                     AvaiTime(VMti) = FT(ti)
22:                     k=k+1
23:     EndFor
24: EndWhile
```

5. RESULTS AND DISCUSSION

In this section, we explain the results of analyzing the performance of the algorithm with the results of its implementation. For adjusting the algorithm parameters, the performance of the algorithm has been analyzed using different values. Figures 2(a)-(c) (see Appendix) shows the performance of the algorithm based on the makespan value, using deferent cases of Montage and Epigenomics workflows. The performance has been analyzed firstly by adjusting the number of iterations in the ILS algorithm, which is the number of perturbation and local search work. The algorithm stops when it reaches the number of iterations in which the solution has not improved. Secondly, the number of neighbors extracted from the current solution in the local search algorithm is also considerd. Finally, adjusting the perturbation ratio of the solution.

After trying several different values, we note that as the number of iterations increases, the solution improves to a certain extent, and the improvement stops even when the iterations increase. The number of neighbors generated in local search is approximately better depending on the number of tasks. Finally, the lower the percentage of perturbation, would be the better to a certain extent. After implementing the algorithm on many scientific workflow instances, the results are shown in Table 2, the first column represents the application of the scientific workflow, while the second, third and fourth columns represent the proposed algorithm results obtained. The remaining columns represent the results of the algorithms that were compared. The obtained results of the proposed algorithm are compared with heterogeneous earliest finish time (HEFT), minimum completion time (MCT), round robin (RR) algorithms. According to the proposed algorithm compared to other, we note that the results of implementing the proposed algorithm are better in many cases except in CyberShake.

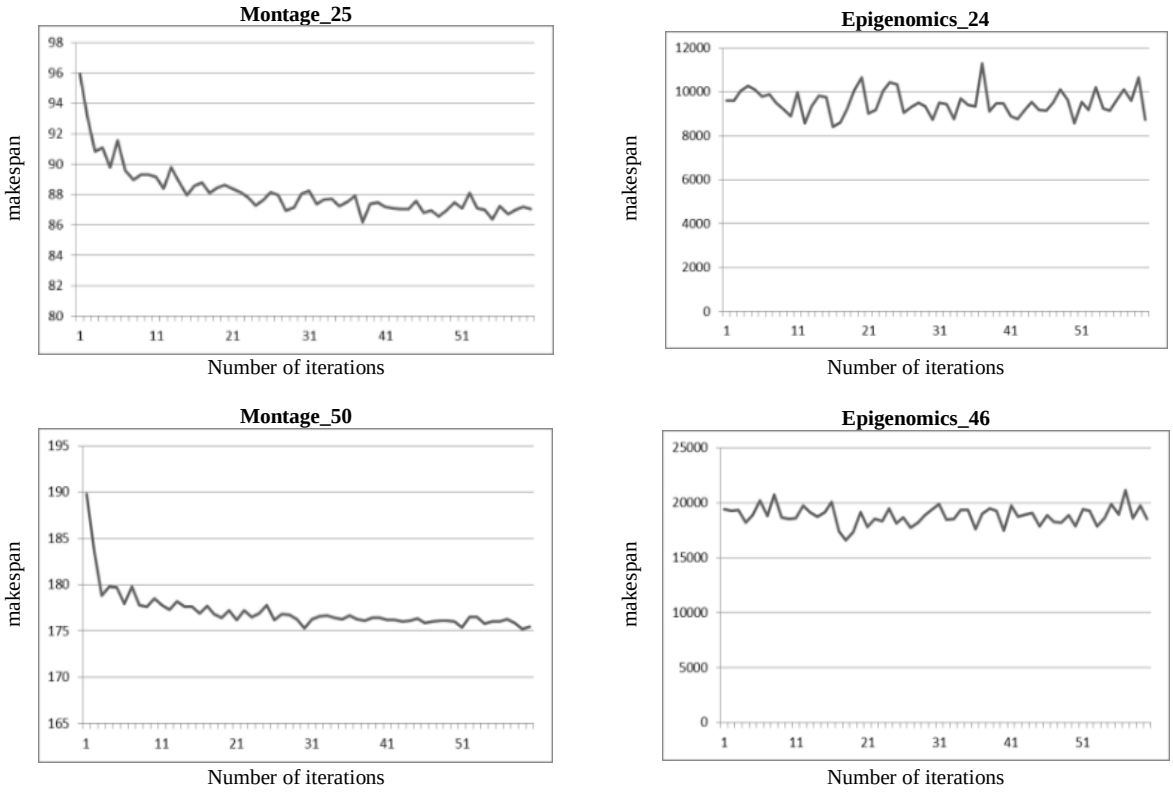
Table 2. Average makespan comparison of ILS with HEFT, MCT, and RR for scientific workflows

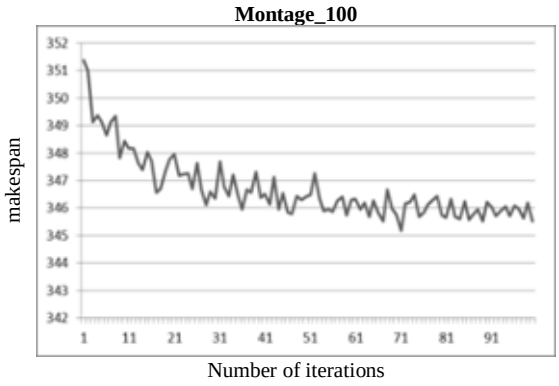
Workflow	ILS			HEFT	MCT	RR
	Min	Av	Max			
Montage_25	**84.76**	86.79	88.67	85.71	91.35	87.63
Montage_50	**172.97**	174.57	175.44	179.1	181.06	183.91
Montage_100	**345.16**	346.54	347.94	347.79	348.79	353.11
Montage_1000	**3536.37**	3539.02	3540.77	3538.97	3542.18	3587.62
CyberShake_30	**379.96**	389.16	396.97	380.46	425.62	393.25
CyberShake_50	544.27	562.27	582.2	**521.53**	539.9	527.74
CyberShake_100	831.22	853.22	891.05	**753.22**	761.27	767.34
CyberShake_1000	6831.79	7037.25	7334.78	6807.57	**6790.68**	6840.51
Epigenomics_24	**6534.76**	9170.13	11277.38	6548.63	9837.18	7941.59
Epigenomics_46	**14046.82**	18131.44	21325.42	15146.16	18250.16	17270.46
Epigenomics_100	**121506.62**	138622.89	159865.11	127531.9	129116.61	131399.43
Epigenomics_997	1108351.75	1109442.35	1110026.87	**1107585.38**	1112766.72	1120257.65
Inspiral_30	**2103.01**	2154.71	2204.35	2306.69	2267.42	2360.8
Inspiral_50	**3488.41**	3551.4	3603.18	3572.75	3978.44	4154.08
Inspiral_100	**6089.93**	6597.02	8167.79	7747	6254.53	6640.41
Inspiral_1000	**65064.44**	65066.06	65069.51	65114.86	65816.79	65669.68

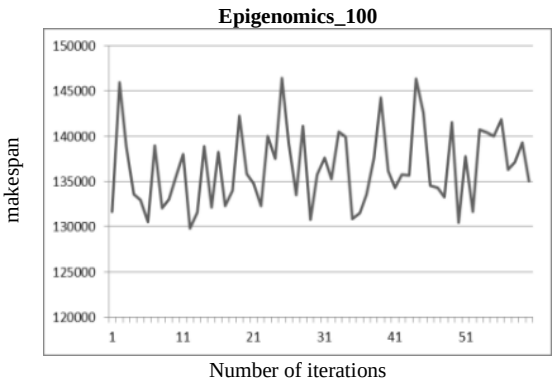
6. CONCLUSION

Recently, a scientific workflow has become a rich area of research that is attracting researchers as well as practitioners in different research domains. Accordingly, reducing the makespan of the scientific workflow represents the main objective of this paper. The standard ILS metaheuristic is successful in tackling various combinatorial optimization problems; therefore, the paper hypothesizes that ILS would be successful in tackling scientific workflow. Based on the fact that the ILS components play a prominent role in improving their behavior during the search, the appropriate selection of these components leads to enhancing the performance of the ILS. The ILS has been implemented on a realistic scientific workflow and the obtained results have been compared with these of the HEFT, MCT, and RR algorithms. The obtained results supported the above-mentioned hypothesis, as the ILS has attained competitive results, if not superior, and generalized well overall tested instances.

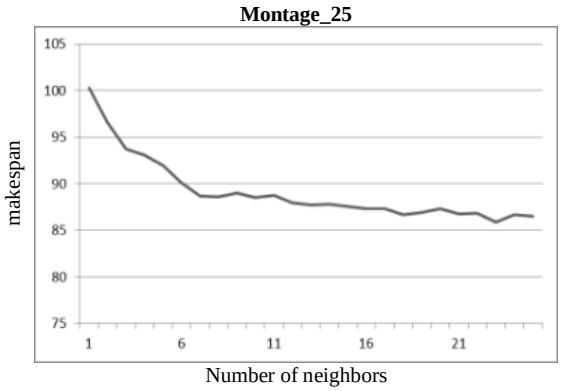
APPENDIX

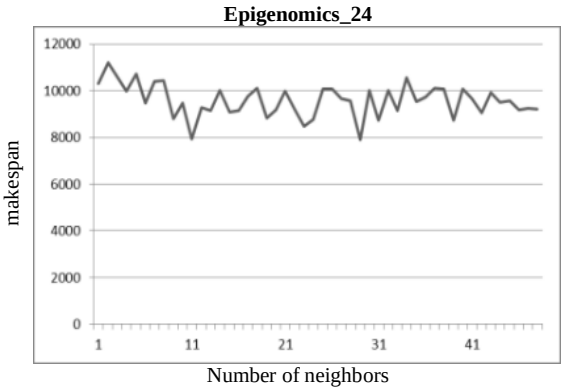


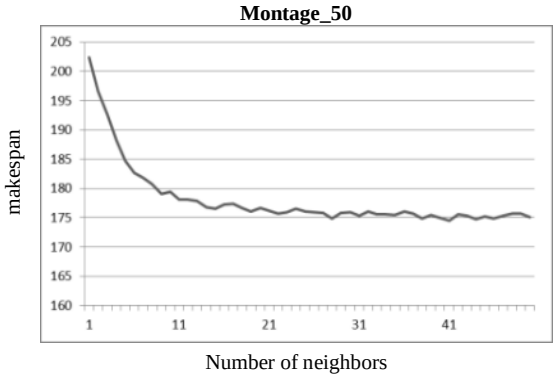

Montage_100
makespan
Number of iterations

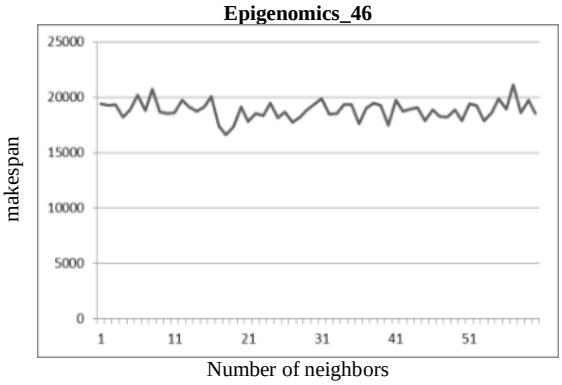

Epigenomics_100
makespan
Number of iterations

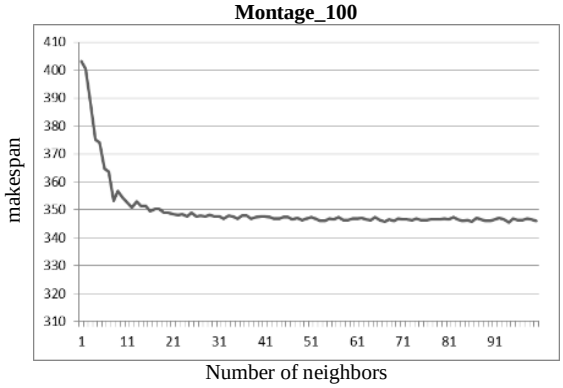
(a)

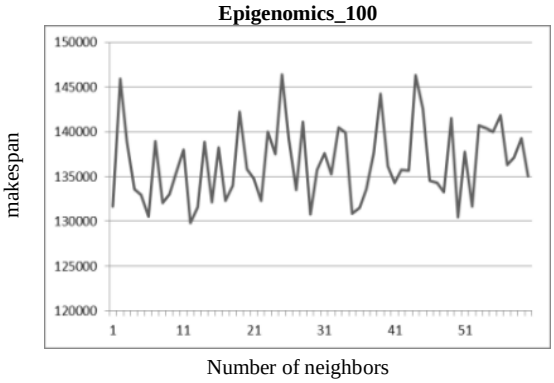

Montage_25
makespan
Number of neighbors


Epigenomics_24
makespan
Number of neighbors

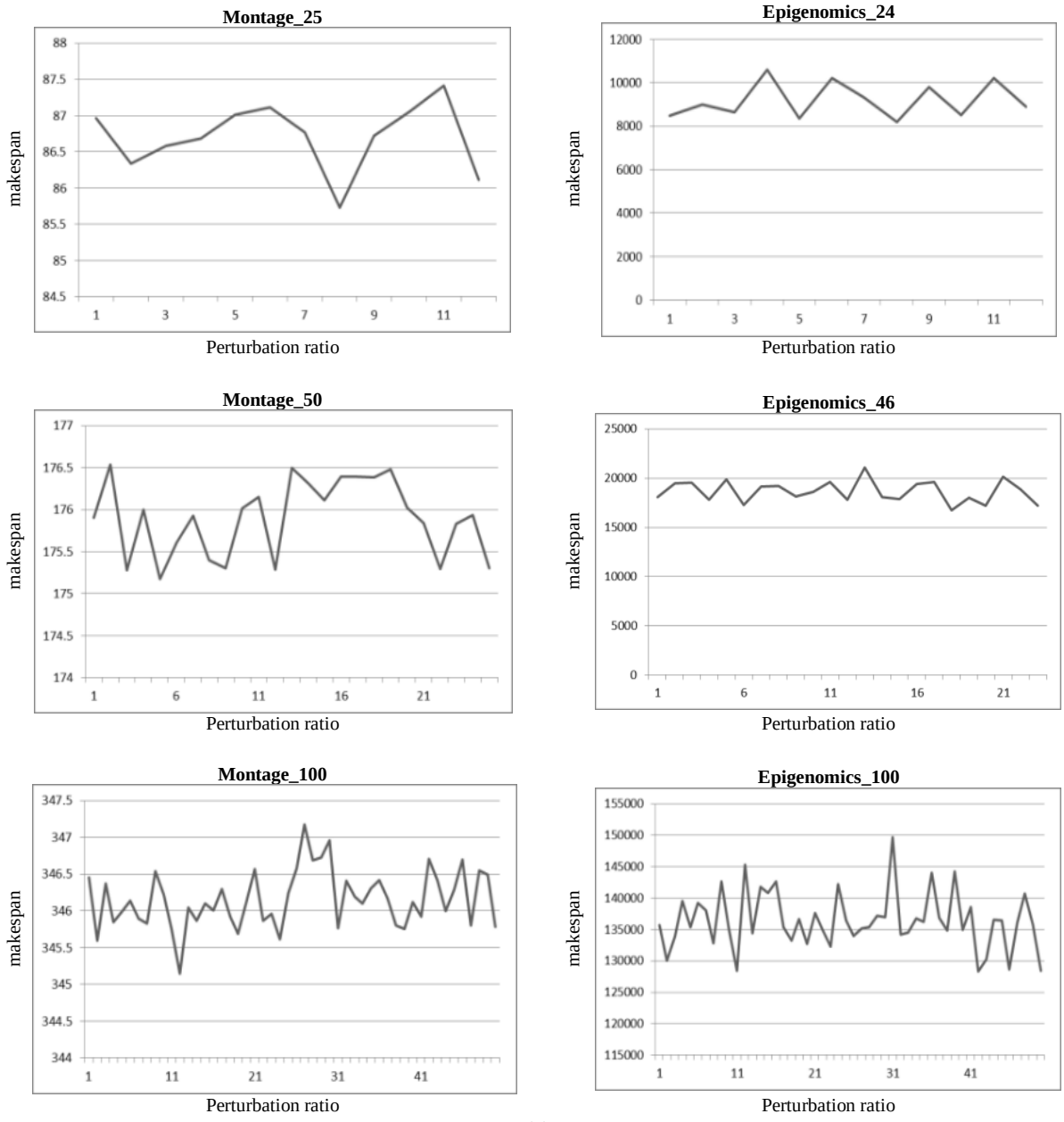

Montage_50
makespan
Number of neighbors


Epigenomics_46
makespan
Number of neighbors


Montage_100
makespan
Number of neighbors


Epigenomics_100
makespan
Number of neighbors

(b)



(c)

Figure 2. The relation between number of iterations, number of neighbors, and perturbation ratio with quality of solutions (makespan) of (a) number of iterations, (b) number of neighbors, and (c) perturbation ratio

ACKNOWLEDGEMENTS

The authors would like to acknowledge the contribution of the University of Anbar (www.uoanbar.edu.iq) via their prestigious academic staff in supporting this research with all required technical and academic support.

REFERENCES

[1] W. Ahmad, B. Alam, S. Ahuja, and S. Malik, "A dynamic VM provisioning and de-provisioning based cost-efficient deadline-aware scheduling algorithm for Big Data workflow applications in a cloud environment," *Cluster Comput.*, vol. 24, no. 1, 2021, doi: 10.1007/s10586-020-03100-7.

[2] M. A. Rodriguez and R. Buyya, "A taxonomy and survey on scheduling algorithms for scientific workflows in IaaS cloud computing environments," *Concurr. Comput.*, vol. 29, no. 8, 2017, doi: 10.1002/cpe.4041.

[3] J. Liu, S. Lu, and D. Che, "A survey of modern scientific workflow scheduling algorithms and systems in the era of big data," in *Proceedings-2020 IEEE 13th International Conference on Services Computing, SCC 2020*, 2020, doi: 10.1109/SCC49832.2020.00026.
[4] X.-F. Liu, Z.-H. Zhan, J. D. Deng, Y. Li, T. Gu, and J. Zhang, "An energy efficient ant colony system for virtual machine placement in cloud computing," *IEEE Trans. Evol. Comput.*, vol. 22, no. 1, pp. 113-128, 2016, doi: 10.1109/TEVC.2016.2623803.
[5] A. Osman, A. Sagahyroon, R. Aburukba, and F. Aloul, "Optimization of energy consumption in cloud computing datacenters," *Int. J. Electr. & Comput. Eng.*, vol. 11, no. 1, 2021, doi: 10.11591/ijece.v11i1.pp686-698.
[6] M. A. Aziz and I. H. Ninggal, "Scalable workflow scheduling algorithm for minimizing makespan and failure probability," *Bull. Electr. Eng. Informatics*, vol. 8, no. 1, pp. 283-290, 2019, doi: 10.11591/eei.v8i1.1436.
[7] K. K. Chakravarthi, L. Shyamala, and V. Vaidehi, "Cost-effective workflow scheduling approach on cloud under deadline constraint using firefly algorithm," *Appl. Intell.*, vol. 51, no. 3, 2021, doi: 10.1007/s10489-020-01875-1.
[8] H. Hu *et al.*, "Multi-objective scheduling for scientific workflow in multicloud environment," *J. Netw. Comput. Appl.*, vol. 114, 2018, doi: 10.1016/j.jnca.2018.03.028.
[9] A. M. Manasrah and H. B. Ali, "Workflow Scheduling Using Hybrid GA-PSO Algorithm in Cloud Computing," *Wirel. Commun. Mob. Comput.*, vol. 2018, 2018, doi: 10.1155/2018/1934784.
[10] A. Song, W.-N. Chen, X.-N. Luo, Z.-H. Zhan, and J. Zhang, "Scheduling Workflows with Composite Tasks: A Nested Particle Swarm Optimization Approach," *IEEE Trans. Serv. Comput.*, 2020, doi: 10.1109/TSC.2020.2975774.
[11] V. D. Maio and D. Kimovski, "Multi-objective scheduling of extreme data scientific workflows in Fog," *Futur. Gener. Comput. Syst.*, vol. 106, pp. 171-184, 2020, doi: 10.1016/j.future.2019.12.054.
[12] X. Ma, H. Gao, H. Xu, and M. Bian, "An IoT-based task scheduling optimization scheme considering the deadline and cost-aware scientific workflow for cloud computing," *Eurasip J. Wirel. Commun. Netw.*, vol. 2019, no. 1, 2019, doi: https: 10.1186/s13638-019-1557-3.
[13] H. R. Faragardi, M. R. S. Sedghpour, S. Fazliahmadi, T. Fahringer, and N. Rasouli, "GRP-HEFT: A budget-constrained resource provisioning scheme for workflow scheduling in IaaS clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 6, pp. 1239-1254, 2019, doi: 10.1109/TPDS.2019.2961098.
[14] M. Adhikari, T. Amgoth, and S. N. Srirama, "Multi-objective scheduling strategy for scientific workflows in cloud environment: A Firefly-based approach," *Appl. Soft Comput. J.*, vol. 93, 2020, doi: 10.1016/j.asoc.2020.106411.
[15] G. E. A. Fuentes, E. S. H. Gress, J. C. S. T. Mora, and J. M. Marin, "Solution to travelling salesman problem by clusters and a modified multi-restart iterated local search metaheuristic," *PLoS One*, vol. 13, no. 8, p. e0201868, 2018, doi: 10.1371/journal.pone.0201868.
[16] C. Archetti, D. Feillet, A. Mor, and M. G. Speranza, "An iterated local search for the Traveling Salesman Problem with release dates and completion time minimization," *Comput. \& Oper. Res.*, vol. 98, pp. 24-37, 2018, doi: 10.1016/j.cor.2018.05.001.
[17] J. Brandão, "Iterated local search algorithm with ejection chains for the open vehicle routing problem with time windows," *Comput. \& Ind. Eng.*, vol. 120, pp. 146-159, 2018, doi: 10.1016/j.cie.2018.04.032.
[18] J. Brandão, "A memory-based iterated local search algorithm for the multi-depot open vehicle routing problem," *Eur. J. Oper. Res.*, vol. 284, no. 2, pp. 559-571, 2020, doi: 10.1016/j.ejor.2020.01.008.
[19] H. Zohali, B. Naderi, M. Mohammadi, and V. Roshanaei, "Reformulation, linearization, and a hybrid iterated local search algorithm for economic lot-sizing and sequencing in hybrid flow shop problems," *Comput. \& Oper. Res.*, vol. 104, pp. 127-138, 2019, doi: 10.1016/j.cor.2018.12.008.
[20] D. Ferone, S. Hatami, E. M. González-Neira, A. A. Juan, and P. Festa, "A biased-randomized iterated local search for the distributed assembly permutation flow-shop problem," *Int. Trans. Oper. Res.*, vol. 27, no. 3, pp. 1368-1391, 2020, doi: 10.1111/itor.12719.
[21] K. Loheswaran, T. Daniya, and K. Karthick, "Hybrid cuckoo search algorithm with iterative local search for optimized task scheduling on cloud computing environment," *J. Comput. Theor. Nanosci.*, vol. 16, no. 5-6, pp. 2065-2071, 2019, doi: 10.1166/jctn.2019.7851.
[22] A. Santiago *et al.*, "GRASP and Iterated Local Search-Based Cellular Processing algorithm for Precedence-Constraint Task List Scheduling on Heterogeneous Systems," *Appl. Sci.*, vol. 10, no. 21, p. 7500, 2020, doi: 10.3390/app10217500.
[23] E. Queiroga, R. G. S. Pinheiro, Q. Christ, A. Subramanian, and A. A. Pessoa, "Iterated local search for single machine total weighted tardiness batch scheduling," *J. Heuristics*, vol. 27, no. 3, pp. 353-438, 2021, doi: 10.1007/s10732-020-09461-x.
[24] S. Shah, "Implementation of iterative local search (ILS) for the quadratic assignment problem," 2020, doi: 10.36227/techrxiv.12814232.
[25] A. Estrada-Moreno, M. Savelsbergh, A. A. Juan, and J. Panadero, "Biased-randomized iterated local search for a multiperiod vehicle routing problem with price discounts for delivery flexibility," *Int. Trans. Oper. Res.*, vol. 26, no. 4, pp. 1293-1314, 2019, doi: 10.1111/itor.12625C.
[26] J. Liu, E. Pacitti, P. Valduriez, and M. Mattoso, "Parallelization of scientific workflows in the cloud," INRIA, 2014.
[27] H. R. Lourenço, O. C. Martin, and T. Stützle, "Iterated local search: Framework and applications," in *Handbook of metaheuristics*, Springer, pp. 129-168, 2019, doi: 10.1007/978-3-319-91086-4_5.
[28] N. Garg, D. Singh, and M. S. Goraya, "Energy and resource efficient workflow scheduling in a virtualized cloud environment," *Cluster Comput.*, vol. 24, no. 2, pp. 767-797, 2021, doi: 10.1007/s10586-020-03149-4.
[29] M. Ala'Anzy and M. Othman, "Load balancing and server consolidation in cloud computing environments: A meta-study," *IEEE Access*, vol. 7, pp. 141868-141887, 2019, doi: 10.1109/ACCESS.2019.2944420.
[30] W. Chen and E. Deelman, "Workflowsim: A toolkit for simulating scientific workflows in distributed environments," in *2012 IEEE 8th international conference on E-science*, 2012, pp. 1-8, doi: 10.1109/eScience.2012.6404430.
[31] A. Tarafdar, K. Karmakar, S. Khatua, and R. K. Das, "Energy-Efficient Scheduling of Deadline-Sensitive and Budget-Constrained Workflows in the Cloud," in *International Conference on Distributed Computing and Internet Technology*, 2021, pp. 65-80, doi: 10.1007/978-3-030-65621-8_4.
[32] P. Han, C. Du, J. Chen, F. Ling, and X. Du, "Cost and makespan scheduling of workflows in clouds using list multiobjective optimization technique," *J. Syst. Archit.*, vol. 112, 2021, doi: 10.1016/j.sysarc.2020.101837.

BIOGRAPHIES OF AUTHORS

Alaa Abdalqahar Jihad was born in Anbar-Iraq in 1985. He received his B.Sc. from Faculty of Computer Science at Anbar University, Iraq in 2009. The MSc. degree Faculty of Computer Science at Anbar University, Iraq 2012. His research interests are, Metaheuristics, Scheduling, Artificial Intelligent, Data Mining, Machine Learning and Natural Language Processing. He can be contacted at email: it.alaa.heety@uoanbar.edu.iq.

Sufyan T. Faraj Al-Janabi was born in Haditha, Iraq (1971). He obtained his B.Sc. (1992), M.Sc. (1995), and Ph.D. (1999) in Electronic and Communications Engineering from the College of Engineering, Nahrain University in Baghdad. He was started as a faculty member in the Computer Engineering Dept., the University of Baghdad in 1999. Prof. (Faraj) Al-Janabi is the winner of the 1st Award for the Best Research Paper in Information Security from the Association of Arab Universities (AARU), Jordan, 2003. He is also the winner of the ISEP fellowship 2009 and the Fulbright fellowship 2010, USA. He is a member of ACM, ASEE, IACR, and IEEE. He can be contacted at email: sufyan.aljanabi@uoanbar.edu.iq.

Esam Taha Yassen is a lecturer in the college of Computer and Information Technology at the University of Anbar, Iraq since 2002. He has obtained his PhD in Computer Science at The University Kebangsaan Malaysia (UKM) in 2015. His main research areas include metaheuristics, hyper-heuristics and combinatorial Optimization problems especially, routing and scheduling. He has been served as a programme committee for four international conferences and reviewers for high impact journals. He is a researcher in Data Mining and Optimization Research Group (DM0), Centre for Artificial Intelligent (CAIT), UKM. Currently, he is the manager of Computers Centre in University of Anbar. He can be contacted at email: co.esamtaha@uoanbar.edu.iq.