Optimized scheduling of scientific workflows based on iterated local search

Alaa Abdalqahar Jihad¹, Sufyan T. Faraj Al-Janabí², Esam Taha Yassen¹
¹Computer Center, University of Anbar, Ramadi, Iraq
²College of Computer Science and Information Technology, University of Anbar, Ramadi, Iraq

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.

Keywords: Cloud computing, Optimization, Quality of service, Scheduling, Scientific workflows

This is an open access article under the CC BY-SA license.

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 using 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). Concerning 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 children. 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:

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

where \( t_i \in \text{parent}(t_i) \), and \( \text{AvaiTime}(VM_{t_i}) \) is the earliest time when \( VM_{t_i} \) ready to execute 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 \( t_j \in \text{parent}(t_i) \).

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: \( S_{\text{current}} = \text{Generate a random initial solution} \)
3: \( S_{\text{current}} = \text{LocalSearch}(S_{\text{current}}, N) \)
4: \( S_{\text{best}} = S_{\text{current}} \)
5: \( i = 1 \)
6: While \( i < \text{NonItr} \) do
7: \( S_{\text{perturbation}} = \text{Perturbation}(S_{\text{current}}, Pr) \)
8: \( S_{\text{perturbation}} = \text{LocalSearch}(S_{\text{perturbation}}, N) \)
9: if \( f(S_{\text{perturbation}}) < f(S_{\text{best}}) \) then
10: \( S_{\text{best}} = S_{\text{perturbation}} \)
11: \( i = 1 \)
12: else
13: \( i = i + 1 \)
14: \( S_{\text{current}} = \text{AcceptanceCriterion}(S_{\text{current}}, S_{\text{perturbation}}) \)
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 climbing) 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](image)

Algorithm 2. Local search

Input: initial solution \( S_0 \), number of neighbors \( N \) // \( S_0 \) Solution from ILS Steps

Output: The best solution \( S_{best} \) (local optima).

1: \( S_{current} = S_0 \)
2: NoImprovement = 0
3: While NoImprovement = 0 Do
4: NoImprovement = 1
5: Neighbors\( (S_{current}) \) Generate \( N \) candidate solutions \( \{S_i\} \) \( \in \) Neighbors of \( S_{current} \)
6: for each \( S_i \) in Neighbors \( (S_{current}) \)
7: if \( f(S_i) < f(S_{current}) \)
8: \( S_{current} = S_i \)
9: NoImprovement = 0
10: Endfor
11: 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 sever al instances.

<table>
<thead>
<tr>
<th>VMs</th>
<th>RAM</th>
<th>Bandwidth</th>
<th>MIPS</th>
<th>No. CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>512</td>
<td>800</td>
<td>800</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>512</td>
<td>900</td>
<td>900</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>512</td>
<td>500</td>
<td>500</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>512</td>
<td>600</td>
<td>600</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>512</td>
<td>700</td>
<td>700</td>
<td>1</td>
</tr>
</tbody>
</table>

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).
Optimized scheduling of scientific workflows based on iterated local search (Alaa Abdalqahar Jihad)

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]:

$$TT_{ij} = \begin{cases} \frac{data_{ij}}{bw} & \text{if } VM_{t_i} \neq VM_{t_j} \\ 0 & \text{otherwise} \end{cases}$$

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 $VM_{t_i}$ and $VM_{t_j}$. Algorithm 3 illustrates 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 ($t_i$) in Task List (TL) do
5:     If $t_i$ is not handled
6:       TransferCost=0, MaxTime=0, Ready=1
7:     For each parent of $t_i$ ($Pt_i$) do //where $Pt_i \in$ Parent List (PL) $t_i$
8:       If $Pt_i$ is not handled
9:       Ready=0
10:      else
11:       MaxTime= Max(MaxTime, FT($Pt_i$))
12:   EndFor
13:   If Ready=1
14:     For each $Pt_i$ in (PL) $t_i$ do
15:       If VM$_{t_i}$ <> VM$_{Pt_i}$
16:         TransferCost=TransferCost+ TT($Pt_i$, $t_i$)
17:     EndFor
18:     ST($t_i$) = Max(MaxTime, AvailTime(VM$_{t_i}$)) + TransferCost
19:     // where AvailTime(VM$_{t_i}$) is Available Time of VM$_{t_i}$
20:     FT($t_i$) = ST($t_i$) + (Length of $t_i$ / Mips of VM$_{t_i}$)
21:     Makespan=Max(Makespan, FT($t_i$))
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 considered. 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

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

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 those 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
Optimized scheduling of scientific workflows based on iterated local search (Alaa Abdalqahar Jihad)
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


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, Nahra 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.