Journal of Theoretical and Applied Information Technology
15th January 2017. Vol.95. No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

OPTIMIZING RESOURCE ALLOCATION SCHEDULING IN CLOUD COMPUTING SERVICES

[1]AMJAD GAWANMEH, [2]AHMAD ALOMARI, [3]ALAIN APRIL
[1]Department of Electrical and Computer Engineering, Khalifa University, Abu Dhabi, UAE
[1]Department of Electrical and Computer Engineering, Concordia University, Montreal, Canada
[23]Department of Software Engineering, Universite du Quebec, Ecole de Technologie Superieure, Montreal, Canada
E-mail: [1]amjad.gawanmeh@kustar.ac.ae, [2]ahmad.alomari.l@ens.etsmtl.ca, [3]alain.april@etsmtl.ca

ABSTRACT

Resource allocation in cloud computing systems is getting more complex and demanding due to the increasing requirements for cloud-based services. Scheduling services using a limited number of resources is problem that has been under study since the evolution of cloud computing. However, there are several open areas for improvements due to the large number of optimization variables. In this work, we intend to presents an algorithm to solve the fundamental problem of multiple tasks resource allocation that are to be scheduled on available services. Several resources will be considered where the cost of available services will depend the computational complexity needed for every service. The proposed algorithm can be applied without constraints on cost or execution time vectors as opposed to most practical and recent existing algorithms. The proposed algorithm is illustrated on two different examples. In addition, the algorithm as implemented and simulated in order to validate its correctness. The experimental results conducting using the proposed method proofs that the algorithm runs in linear time vs. different design parameters. The main limitation of the proposed algorithm is that it is only applicable to the scheduling problem of multiple tasks that has one price vector and one execution time vector. However, providing optimum solution for this particular case, can be helpful in designing heuristic based methods for cloud services are actually run with multiple users with multiple tasks, which requires initial solutions that are usually obtained based on guess or generated randomly.

Keywords: *Could services, Cloud Scheduling, Distributed systems, Cloud Computing, Scheduling*

1. INTRODUCTION

The area of cloud computing [1] is indented to provide computing services for customers over a network using a priced leased method. This paradigm may provide users with the ability of scaling their services requirements. In addition, this emerging concept allowed service providers to share and optimize the usage of distributed resources and services that may belong to the same or different providers. This also allow providers to balance the availability and demand on resources and hence, reduce the cost associated with the increase on available resources. However, since this issue imposed that cloud computing use distributed resources in open networks, several challenges were raised related to optimizing available distributed resources for various cloud computing applications.

While several service scheduling problems have been studied in the literature for cloud systems, the problem of resource allocation includes several optimization factors, and therefore it is considered a difficult one. Most proposed approaches must state a set of assumptions about the available services, number of tasks and their subtasks, and finally communication between these services.

In this work, we intend to propose a method to find optimum solution for resource allocation in available distributed cloud service providers. The problem of resource allocation with multiple users with tasks that has variable execution times on given multiple available resources is considered problem NP-Complete scheduling problem. Therefore, only heuristic based solutions that did not guarantee an optimum solution were provided in the literature. In previous work in [2], an algorithm based on utility function is provided to handle this problem. Even through experimental results have shown improvement over existing methods, it is believed that the design and choice of the selection function can be further improved. This however, requires looking at the basic problem of allocating subtasks using a given number of resources. Finding optimal solution for this basic problem will help building better selection function for the general problem of multiple users with multiple subtasks of variable costs. Therefore, the algorithm proposed in this paper directed towards answering the question about the finding the optimum schedule that cloud system may use in order allocate multiple subtasks with different execution time and weight on a given number of available resources with different capabilities.

The rest of this paper is organized as follows: Section 2 presents related work. Section 3 describes the problem description. Section 4 presents the proposed method with an example to illustrate its execution. The experimental results are illustrated in Section 5. Finally Section 6 concludes the paper with future work hints.

2. RELOATED WORK

There are several detailed reviews about this subject can be found in surveys such as [3], [4], [5], [6], and [7]. However, it is important to present some related work and discuss improvements over existing methods.

Methods that are based on evolutionary genetic algorithms to solve scheduling problem in cloud computing were presented in several works. The work in [8] presented a genetic algorithm method for task level scheduling in Hadoop MapReduce, while the work can help finding local optimum solution, the execution of the load-balancing algorithm can take long time to make a decision for task assignments. The authors in [9] addressed the independent batch scheduling in computational grid by presenting a genetic based algorithm in order to solve the global minimization problem in grid based energy consumption. The main disadvantage of this work, is that it is based only on two criteria, while fixing several other parameters. A genetic based scheduling algorithm was presented by [10] in order to reduce the waiting time of tasks to be scheduled in a cloud environment. There certain applications where the waiting time can be critical, however, in general, the main issue in this work is that optimizing scheduling for waiting time will have big impact on system utilization, which might not be favorable for several cloud applications. While the use of evolutionary based algorithms can help solving very large scale problems, it cannot guarantee obtaining the absolute optimum solutions, therefore it can be useful in certain cloud applications, and however, it cannot be used to provide baseline solutions for scheduling problems. In fact, evolutionary algorithms may make use of methods that provide absolute optimum solution for small scale problems in order to build on these.

Game theory based approaches has also been proposed in [11]. This work considers several criteria in the optimization, such as dynamic allocation, variable resources distribution, different requirement of cloud users and their common information. The main issue with this approach is the requirements of strong assumptions about certain system parameters in order to have Nash equilibrium for the game. A similar approach was proposed in [12] to solve the problem of resource allocation in cognitive networks in order to increase resource utilization efficiency. The main drawback of this approach is that it requires dividing the payoff equally between all users of the cognitive network in order to work. The work in [13] used game theory in order to optimize the usage of resources across a cloud-based network based on the cost of computational services and the amount of computation. The proposed method can be applied on multiple users problem with subtasks, however, in order to apply game theoretic method optimally and find a Nash equilibrium, the execution time must be given in ascending order, and the price in descending.

The use of fuzzy pattern recognition to solve similar problems were also proposed by [14], who presented a dynamically hierarchical resource-allocation method that can be used within multiple cloud nodes. The algorithm requires intercommunication between nodes and prior knowledge about several task parameters. The work in [15] used fuzzy clustering for a workflow task scheduling. The major objective of scheduling is to minimize *makespan* of the precedence constrained applications. The method can only be applied on resource allocation problems that can be modeled as directed acyclic graphs. The work in [16] addressed the problem of finding an adequate tradeoff between two conflicting goals in dynamic resource (re)-allocation for virtual machines in cloud computing to guarantee application performance and to reduce operating costs. This work can be useful in certain applications that have restrictions on specific parameters, such as execution time, or cost, in order to find solutions under these constraints. However, it can be applied in order to optimize the schedule under both parameters simultaneously. Overall, the main problem with fuzzy logic approaches is with the requirement and restrictions imposed on the input space so that the algorithm can provide optimum solution.

Auction mechanisms were also used by [17] for cloud resource allocation. The method works well under certain requirement including a specific value for the bandwidth between the use and the cloud server. Clients are allocated to cloud servers by an auction mechanism when the remaining bandwidth of a cloud server is greater than specific value. The major issue with this approach is that it did not consider the cost of executing specific tasks on the available servers, but only considered bandwidth which is only one factor of the cost. Another auction method were proposed by [18] in order to help cloud providers to decide when and how they will allocate their resources and to which users. While the method can be useful in real time resource allocation, and it can give an effective solution where a certain task is to be scheduled instantly. It is not practical when the resources and tasks are known ahead and are required to be scheduled to optimize utilization of the system.

Other approaches that can be considered related, but addressed problem with different context include profit and pricing based methods, such as the work by Walker *et al.* [19], Karthik *et al.* [20], Buyya *et al.* [21], and Xiao *et al.* [22]. User preferences based scheduling for enhancing QOs was also proposed by Ergu *et al.* [23], Mohan and Satyanarayana [24] and finally by Senthilnathan and Kalaiarasan [25].

These type of methods can be useful in the search for cost-effective solutions for certain QoS requirement. However, solving QoS problem given the number of parameters is multitask scheduling is NP complete problem, therefore to find the absolute optimal schedule in terms of QoS measures is not practical. In order to help in this regard, it is required to solve scheduling problem at small scale first, and find optimum strategy, and then use this strategy to design heuristic based algorithms that can enhance existing scheduling methods.

Even though there are several proposed methods to address resource allocation in cloud computing, on the subject, there is still room for more improved solutions, because the available cloud-based services are dependent on several QoS factors [13]. The main contributions of this work include providing a novel method to find optimum solution to allocate resources on given multiple subtasks to be scheduled a number of resources with given costs, in addition, providing experimental results to show that given solution runs in linear time vs. different design parameters. The method is illustrated on a detailed step by step example that shows how the algorithm evolves in order to reach required solution. Experimental results are then conducted with practical number of subtasks and resources. Finally, the proposed method is applicable without any restriction on the cost vector or execution time for resources.

3. SCHEDULING MULTIPLE SUBTASKS USING MULTIPLE RESOURCES

We provide here the problem description based on previous description in [2] and [8]. It is still necessary to introduce preliminary definitions that are needed to understand the work presented here. Hence, we will first present a formalization for the necessary parts of the problem. Given a number of *k* subtasks for a cloud services user that are to be scheduled using existing *m* computational resources: $\{R_1, R_2, ..., R_m\}$. Each resource, R_j, costs a fixed price p_j, hence forming the price vector $p = \{p_1, p_2, ..., p_m\}$. In addition, each resource R_i require specific time, t_i, to execute any subtask forming the execution time vector $t = \{t_1, t_2, ..., t_m\}$.

It is required to assign the given set of subtasks each into a single selected resource R_j in order to minimize the total cost. Cost is defined using the expense and total execution time for completing all given subtasks. The solution for the given scheduling problem is a resulting vector v that has m elements, each is a non-negative number represent the subtasks assigned to that particular resource. For instance, v_i is element number *i* in vector *v* represents the number of subtasks allocated to m resource *Ri*. Therefore, the solution vector *v* must satisfy:

$$\sum_{i=1}^{m} v_i = k.$$

We define the following two vectors as follows: the execution time vector, denoted as and the expense or cost vector, denoted as ê. The entry $\dot{t}_i$ of $\dot{t}$ is the turnaround time it takes for resource R_j to complete v_j subtasks of the task S. The entry $\hat{e}_i$ of vector ê is the expense S pays for resource R_j to complete v_j subtasks. These two vectors are defined as follows: $\dot{t} = v \cdot t$, and $\hat{e}_i = v \cdot t \cdot p$. Based on these, two values for schedule v are calculated, the first represents the total execution time t_{max}, and the second represents the total expense e_v. The execution time for task S is the maximum execution time of tasks assigned to resources, $t_{max} = max\{ \dot{t}_i \mid \dot{t}_i \in \dot{t}\}$ where $\dot{t}_i$ denotes the ith element of the vector $\dot{t}$. The total expense e_v is the summation of all expenses paid to all resources, $e_v = \sum_{i=1}^{m} e_i$. We assign weights for schedule costs as follows, w_t for execution time weight, and w_e for expense weight. Then we can define a benefit value of the expense using the following utility function:

$$u(v) = \frac{1}{w_t \times t_v + w_e \times e_v}$$

$$= \frac{1}{w_t \times max\{\hat{t}_i | \hat{t}_i \in t\hat{t}\} + w_e \times \sum_{i=1}^{m} e_i}$$

The optimum solution is achieved, when u is maximized. The following example is shown in order to demonstrate the allocation problem. Using five available resources (R_1 - R_5), $m=5$, and using the price vector of $p = (1.2, 1.5, 2, 1.0, 1.8)$, there is a task S with $k = 3$ subtasks. We first obtain the execution time vector for each subtask as follows $t = (4, 3.5, 3.2, 2.8, 2.4)$. Assume that a schedule $v = (1, 0, 1, 0, 1)$ is used, then we can calculate $\dot{t}$ and $\hat{e}$ as follows: $\dot{t} = (4, 0, 3.2, 0, 2.4)$, and $\hat{e} = (4.8, 0, 6.4, 0, 4.32)$, then we can calculate, $t_v = 4$ and $e_v = 15.52$. Assuming $w_e = w_t = 1$, then $u = 1/(t_v + e_v) = 0.0512$. On the other hand, the schedule $v=(1,1,0,1,0)$ will yield to $u = 0.0593$.

There is tradeoff between the execution time and the price for every solution. Hence, to further improve the utility, a scheduling algorithm must be used. The proposed method in this work is intended to provide the optimum solution in linear time vs. number of subtasks or number of resources.

4. OPTIMIZING SINGLE USER SCHEDULING

For a problem with k subtasks and m resources, the optimum solution for the optimization problem described above is achieved by finding the allocation vector v the maximizes the utility $u(v)$. The problem is solved by first defining a variable t_{max} uninitialized to 0. A selection function called δ

```
Algorithm 1 Susbtasks Resource Allocation Algorithm
procedure SUSBTASKS RESOURCE ALLOCATION ALGORITHM
    Input: $k, n, t, p$.
    Output: $v$.
    Initialize:
    • Initialize vectors and variables
      $v = 0$ , $\hat{t}_{max} = 0$, $j = 0$; initialize all elements to 0

    • Initialize the allocation function vector $\delta$
    REPEAT
        $\delta_j = p_j \times t_j + t_j$
        $j = j + 1$
    UNTIL $j = k$ ; simulation time ends
    • Process all elements elements using allocation function
      $j = 0$; Reset counter
    REPEAT
        find $m$ such that
          (1) $1 \leq m \leq n$,
          (2) $\forall m \cdot 1 \leq m \leq n \Rightarrow \delta_m \leq \delta_i$,
        $v_m = v_m + 1$, allocate current task to $R_m$
        if $v_m \times t_m > t_{max}$ then
            $t_{max} = v_m \times t_m$
        endif
        $\delta_m = max(t_{max}, t_m \times (1 + v_m)) + p_m \times t_m \times (1 + v_m)$.
        $j = j + 1$
    UNTIL $j = k$ ; all tasks are allocated
    END
end procedure
```

with m elements is defined as follows: $\delta = (\delta_1, \delta_2, ..., \delta_m)$, and then initialized as indicated below:

$$\delta_i = max(t_{max}, t_i \times (1 + v_i)) + p_i \times t_i \times (1 + v_i).$$

The allocation of the given k subtasks is archived through procedure, where in every step, one subtask is assigned into one resource. In every step, we chose allocation resource m that satisfies the following set of conditions: (1) $1 \leq m \leq n$, and (1) $\forall m \cdot 1 \leq m \leq n \Rightarrow \delta_m \leq \delta_i$. This means that we chose δ_m which is minimum in δ. The current subtask is hence assigned into resource R_m. The schedule v by incrementing v_m is updated accordingly, a new $\dot{t}_m$, δ_m, and t_{max} are then obtained based on the new vector. This step is repeated until all subtasks are scheduled.

This algorithm has a complexity of worst case *O(nk)*. It can be done by sorting δ, and then whenever, δ_i is calculated, it will be inserted into δ while sorted. Applying the above algorithm on the example explained above will result in $v = (0, 0, 0, 2, 1)$, with utility $u = 0.0644$, which is the optimum solution.

4.1 Illustrative Examples

Let us consider two examples in order to demonstrate the algorithm above to find an optimum allocation vector. In the first, we assume the price vector is given as $p = (1, 1.2, 1.5, 1.8, 2)$, the number of resources $M = 5$, number of subtasks is $k = 4$, the execution time vector for each subtask using the given resources is given as $t = (4, 3.5, 3.2, 2.8, 2.4)$. We start calculating initial values of the selection function using the algorithm above, that leads to $\delta = (8, 7.7, 8, 7.84, 7.2)$. In addition, we initiate the allocation vector and t_{max} as follows: $v = (0, 0, 0, 0, 0)$, $t_{max} = 0$. Then we chose m, such that δ_m is minimum, in this case, $\delta_m = 5$, hence, the first subtask will be scheduled to service R_5. This is illustrated in the initial step indicated as λ_0 in Table 1, where the numbers in bold represent the active ones. Based on this, we update the allocation vector, v, the selection function δ, k, and t_{max} in the next step λ_1 as follows: $v = (0, 0, 0, 0, 1)$, $\delta = (8, 7.7, 8, 7.84, 14.4)$, $k = 3$, and $t_{max} = 2.4$. The algorithm proceeds until all subtasks are processed. Table 1 below shows the steps for executing the algorithm on this example. The final allocation vectors becomes $v = (1, 1, 1, 0, 1)$ with $u_v = 0.459$, for $wt = 1.0$ and $w_e = 1.0$, which is the optimum solution.

In the second example, we will use different parameters as follows: the number of resources $M = 10$, number of subtasks is $k = 15$, $p = (4, 8, 11, 10, 9, 7, 13, 16, 6, 12)$, and $t = (12, 11, 5, 7, 5, 8, 6, 8, 11, 5)$. Executing the algorithm, as shown in the table 2, leads into the schedule $v = (2, 1, 2, 1, 3, 1, 1, 1, 2, 1)$, with $u = 0.00104$ for $w_t = 1.0$ and $w_e = 1.0$, which is the optimum solution. In the next Section we present performance evaluation for the algorithm.

The examples above show that the algorithm is simple to implement and at the same time efficient. In fact, when a more complex problem is considered, such as multiple users with tasks with different price vectors, the problem of finding the optimal solution becomes NP complete. However, using a simple and efficient method to find the optimum schedule for the single user case will solve the fundamental step in several heuristic based methods, such as evolutionary ones, that require initial solutions to the problem as a starting point.

5. PERFORMANCE EVALUATION

In this section we study the performance of the given algorithm for different parameters in the problem. We first show the effect of number of subtasks on the execution time of the algorithm. We did an implementation for the algorithm and executed the scheduling process for variable number of subtasks. In order to conduct experimental results, we first provided an implementation for the algorithm in C++. The implementation model of the algorithm is achieved by setting up the selection function with initial values based on input vectors. Then tasks are processed in an iterative process, where every task is allocated for the best available resource provided by the selection function. The selection function is then modified based on this. Next task is scheduled similarly, until all tasks are processed.

In order to study the efficiency of the algorithms, we conducted simulations by generating resources with different costs and commotional power, as well as tasks to be scheduled on these resources. In the first experiment, we set the number of resources to a fixed number of 104. Then, we tested the execution time for algorithm for different the following values of the number of sub-tasks $k = i \times 10^4$, where $i = \{1, 2,..., 100\}$. For every experiment, we initiated p and t with random values. The choice of the price value were done randomly as follows:

```
t[j] = LO+static_cast <float> (rand())
/(static_cast <float>(RAND_MAX/(HI-LO)));

p[j] = LO+static cast <float> (rand())
/(static cast <float>(RAND MAX/(HI-LO)));
```

where the variable *LO* and *HI* represents the ranges for the execution time and the price, and were set to $LO = 1.0$, $HI = 15.0$ for the time, and $LO = 1.0$, $HI = 20.0$ for the price. These values were selected based constipation of variance of computational power between different practical cloud resources, and these values and ranges will have no effect on the actual complexity of the algorithm as described above. All simulations are run on `Windows 7 64 - bit` system with `8.00 GB of RAM`, and `Intel(R) Core(TM)i7-4770 CPU @3.40 GHz`. Figure 1 illustrates the execution time in seconds vs. the number of subtasks, and it shows linear increment.

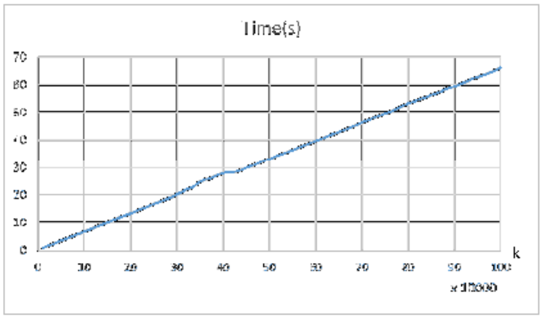

Figure 1: Execution time vs. number of subtasks k

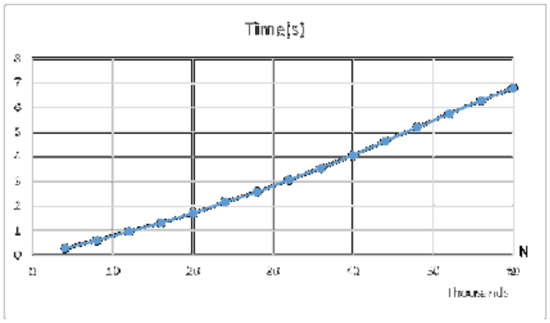

Figure 2: Execution time vs. number of resources N

Next, we executed the scheduling process for variable number of resources, while fixing the number of subtasks to k = 104 subtasks. We initiated p and t with random values similar to above, and then tested the execution time for algorithm for different the following values of subtasks *N = {4000, 8000, ..., 60000}*. Figure 2 illustrates the execution time in seconds vs. the number of resources, and it shows linear increment. Finally, Figure 3 shows the execution time in seconds vs. number of subtasks *k*, and the number of resources simultaneously, which shows linear increment vs. each for different values, i.e., the execution time is linear vs. $k \times n$, which is consistent with the complexity of *O(kn)*. Finally, in order to test the performance of the algorithm under both number of resources and number of subtasks simultaneously, we repeated the experiment while changing number of subtasks from *1k* up to *20k*, with an increment of *1k*, and the number of resources from 1k to *15k*, and calculated the execution time of the algorithm. Figure 2 shows the archived results illustrated as linear behavior.

The results achieved in this work solve the intended problem and provide an algorithm with linear execution time. In addition, this work is fundamental in order to provide an efficient method to define selection function which can be used in finding heuristic based solutions for the NP complete problem of scheduling multiple tasks for multiple users. In fact, the problem of allocating multiple users, each with multiple tasks, on given number of resources is an open one. Several methods have been proposed to provide solutions, however, the presented methods can still be further enhanced if proper optimizing algorithms can be used. In particular, ones that can present optimum solutions for special cases, such as the one peened here. Hence, this algorithm is intended to define a selection function that can serve as heuristic for scheduling multiple users' problem.

6. CONCLUSION AND FUTURE WORK

Cloud computing systems are getting more complex and demanding due to the increasing demand and requirements for cloud based services. On the other hand, providing optimized solutions for scheduling services using a limited number of resources is problem that has gained attention due its impact on cloud computing services. In this work an algorithm is proposed without constraints on cost or execution time vectors as opposed to most practical and recent existing algorithms. The methods is illustrated on practical example that shows its simplicity of execution. On the other hand, the algorithm was tested for variable number of subtasks and available resources. Experimental results show that the algorithm runs in linear time vs. these design parameters.

The method presented in this paper can be used for scheduling problems without imposing constraints on the vectors that represent the execution time and price, as opposed to game theoretic methods that can reach Nash equilibrium only if this condition is satisfied. In addition, the proposed method outputs schedule with best utility as opposed to evolutionary algorithms that does not guarantee the best strategy. As future work, we intend to use the proposed method here and extend it to handle the scheduling problem of multiple users with variable computation time vector for subtasks of different users using a selection function that is designed using the algorithm proposed in this paper.

The main limitation of the proposed algorithm is that it is only applicable to the scheduling problem of multiple tasks that has one price vector and one execution time vector. However, when scheduling multiple users, each with subtasks that have their own price and execution time vector, then the problem becomes NP complete and this method cannot be used. Hence, while the algorithm presented in this work leads into optimum solution, it might not have prominent applications as is, since practical cloud services are actually run with multiple users with multiple tasks, which is considered NP complete problem. However, providing optimum solution for this particular case, can be helpful in designing heuristic based methods that requires initial solutions that are usually obtained based on guess or generated randomly, and then the heuristic method is designed to find better solutions for the multiple users' problem.

REFRENCES:

[1] B. Furht and A. Escalante. *Handbook of Cloud Computing*. Springer, 2010.

[2] A. Gawanmeh and A. April. A novel algorithm for optimizing multiple services resource allocation. *International Journal of Advanced Computer Science and Applications*, 7(6): 428–434, 2016.

[3] M.S. Sagar, B. Singh, and W. Ahmad. Study on cloud computing resource allocation strategies. *International Journal of Advance Research and Innovation*, 1(3):107–114, 2013.

[4] S. Khan. A survey on scheduling based resource allocation in cloud computing. *International Journal for Technological Research in Engineering*, 1(1), 2013.

[5] V. Anuradha and D Sumathi. A survey on resource allocation strategies in cloud computing. *In Information Communication and Embedded Systems (ICICES)*, International Conference on, pages 1–7. IEEE, 2014.

[6] V. Vinothina, R. Sridaran, and P. Ganapathi. A survey on resource allocation strategies in cloud computing. *International Journal of Advanced Computer Science and Applications*, 3(6): 97–104, 2012.

[7] S.S. Manvi and G.K. Shyam. Resource management for infrastructure as a service (IaaS) in cloud computing: A survey. *Journal of Network and Computer Applications*, 41:424–440, 2014.

[8] Y. Ge and G.Wei. GA-based task scheduler for the cloud computing systems. *In Web Information Systems and Mining (WISM)*, International Conference on, volume 2, pages 181–186. IEEE, 2010.

[9] J. Kolodziej, S.U. Khan, L. Wang, and A.Y. Zomaya. Energy efficient genetic-based schedulers in computational grids. *Concurrency and Computation: Practice and Experience*, 27(4):809–829, 2015.

[10]. S. Saha, S. Pal, and P. K. Pattnaik. A novel scheduling algorithm for cloud computing environment. *In Computational Intelligence in Data Mining*, Volume 1, pages 387–398. Springer, 2016.

[11] F. Teng and F. Magoules. A new game theoretical resource allocation algorithm for cloud computing. In Advances in Grid and Pervasive Computing, pages 321–330. Springer, 2010.

[12] A. Velayudham, G. Gohila, B. Hariharan, and M.R. Selvi. A novel coalition game theory based resource allocation and selfish attack avoidance in cognitive radio ad-hoc networks. *Journal of Theoretical of Applied Information Technology*, 64(1), 2014.

[13] G. Wei, A.V. Vasilakos, Y. Zheng, and N. Xiong. A game-theoretic method of fair resource allocation for cloud computing services. *The journal of supercomputing*, 54(2):252–269, 2010.

[14] Z. Wang and X. Su. Dynamically hierarchical resource-allocation algorithm in cloud computing environment. *The Journal of Supercomputing*, pages 1–19, 2015.

[15] F. Guo, L. Yu, S. Tian, and J. Yu. A workfow task scheduling algorithm based on the resources' fuzzy clustering in cloud computing environment. *International Journal of Communication Systems*, 28(6):1053–1067, 2015.

[16] D. Minarolli and B. Freisleben. Virtual machine resource allocation in cloud computing via multi-agent fuzzy control. *In International Conference on Cloud and Green Computing (CGC)*, pages 188–194. IEEE, 2013.

[17] H.Y. Chang, H.C. Lu, Y.H. Huang, Y.W. Lin, and Y.J. Tzang. Novel auction mechanism with factor distribution rule for cloud resource allocation. *The Computer Journal*, bxt008, 2013.

[18] C. Lee, P. Wang, and D. Niyato. A real-time group auction system for efficient allocation of cloud internet applications. *IEEE Transactions on Services Computing*, 8(2):251–268, 2015.

[19] E. Walker, W. Brisken, and J. Romney. To lease or not to lease from storage clouds. *IEEE Computer Journal*, 43(4):44–50, 2010.

[20] K. Kumar, J. Feng, Y. Nimmagadda, and Y.H. Lu. Resource allocation for real-time tasks using cloud computing. *In International Conference on Computer Communications and Networks (ICCCN)*, pages 1–7. IEEE, 2011.

[21] R. Buyya, R. Ranjan, and R.N. Calheiros. Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services. *In Algorithms and architectures for parallel processing*, pages 13–31. Springer, 2010.

[22] Zhen Xiao, Weijia Song, and Qi Chen. Dynamic resource allocation using virtual machines for cloud computing environment. *IEEE Transactions on Parallel and Distributed Systems*, 24(6):1107–1117, 2013.

[23] D. Ergu, G. Kou, Y. Peng, Yong Shi, and Y. Shi. The analytic hierarchy process: task scheduling and resource allocation in cloud computing environment. *The Journal of Supercomputing*, 64(3):835–848, 2013.

[24] V.M. Mohan and K. Satyanarayana. Efficient task scheduling strategy towards QoS aware optimal resource utilization in cloud computing. *Journal of Theoretical of Applied Information Technology*, 80(1), 2015.

[25] P. Senthilnathan and C. Kalaiarasan. A joint design of routing and resource allocation using QoS monitoring agent in mobile ad-hoc networks. *Journal of Theoretical of Applied Information Technology*, 55(2), 2013.

Table 1: Execution of the algorithm on the first example above

λ	v	δ	t_{max}	k	m
λ_0	0, 0, 0, 0, 0	(8.0, 7.7, 8.0, 7.84, **7.2**)	0.0	4	5
λ_1	0, 0, 0, 0, 1	(8.0, **7.7**, 8.0, 7.84, 14.4)	2.4	3	2
λ_2	0, 1, 0, 0, 1	(**8.0**, 15.4, 8.3, 8.54, 14.4)	2.5	2	1
λ_3	1, 1, 0, 0, 1	(16.0, 15.4, **8.8**, 9.04, 14.4)	4.0	1	3
λ_4	1, 1, 1, 0, 1	(16.0, 15.4, 16.0, 9.04, 14.4)	4.0	0	4

Table 2: Execution of the algorithm on the second example above

λ	v	δ	t_{max}	k	m
λ_0	0, 0, 0, 0, 0, 0, 0, 0, 0, 0	(60, 90, 52, 88, **45**, 88, 84, 112, 70, 91)	0	15	5
λ_1	0, 0, 0, 0, 1, 0, 0, 0, 0, 0	(60, 90, **53**, 88, 90, 88, 84, 112, 70, 91)	5	14	3
λ_2	0, 0, 1, 0, 1, 0, 0, 0, 0, 0	(**60**, 90, 104, 88, 90, 88, 84, 112, 70, 91)	5	13	1
λ_3	1, 0, 1, 0, 1, 0, 0, 0, 0, 0	(120, 93, 108, 92, 92, 89, 90, 117, **72**, 96)	12	12	9
λ_4	1, 0, 1, 0, 1, 0, 0, 0, 1, 0	(120, 93, 108, 92, 92, **89**, 90, 117, 140, 96)	12	11	6
λ_5	1, 0, 1, 0, 1, 1, 0, 0, 1, 0	(120, 93, 108, 92, 92, 176, **90**, 117, 140, 96)	12	10	7
λ_6	1, 0, 1, 0, 1, 1, 1, 0, 1, 0	(120, 93, 108, **92**, **92**, 176, 168, 117, 140, 96)	12	9	4,5
λ_7	1, 0, 1, 1, 1, 1, 1, 0, 1, 0	(120, 93, 108, 176, **92**, 176, 168, 117, 140, 96)	12	8	5
λ_8	1, 0, 1, 1, 2, 1, 1, 0, 1, 0	(120, **93**, 108, 176, 135, 176, 168, 117, 140, 96)	12	7	2
λ_9	1, 1, 1, 1, 2, 1, 1, 0, 1, 0	(120, 180, 108, 176, 135, 176, 168, 117, 140, **96**)	12	6	10
λ_{10}	1, 1, 1, 1, 2, 1, 1, 0, 1, 1	(120, 180, **108**, 176, 135, 176, 168, 117, 140, 182)	12	5	3
λ_{11}	1, 1, 2, 1, 2, 1, 1, 0, 1, 1	(120, 180, 156, 176, 135, 176, 168, **117**, 140, 182)	12	4	8
λ_{12}	1, 1, 2, 1, 2, 1, 1, 1, 1, 1	(**120**, 180, 156, 176, 135, 176, 168, 224, 140, 182)	12	3	1
λ_{13}	2, 1, 2, 1, 2, 1, 1, 1, 1, 1	(180, 186, 168, 184, **144**, 178, 180, 234, **144**, 192)	24	2	5,9
λ_{14}	2, 1, 2, 1, 2, 1, 1, 1, 2, 1	180, 186, 168, 184, **144**, 178, 180, 234, 210, 192)	24	1	5
λ_{15}	2, 1, 2, 1, 3, 1, 1, 1, 2, 1	(180, 186, **168**, 184, 184, 178, 180, 234, 210, 192)	24	0	3

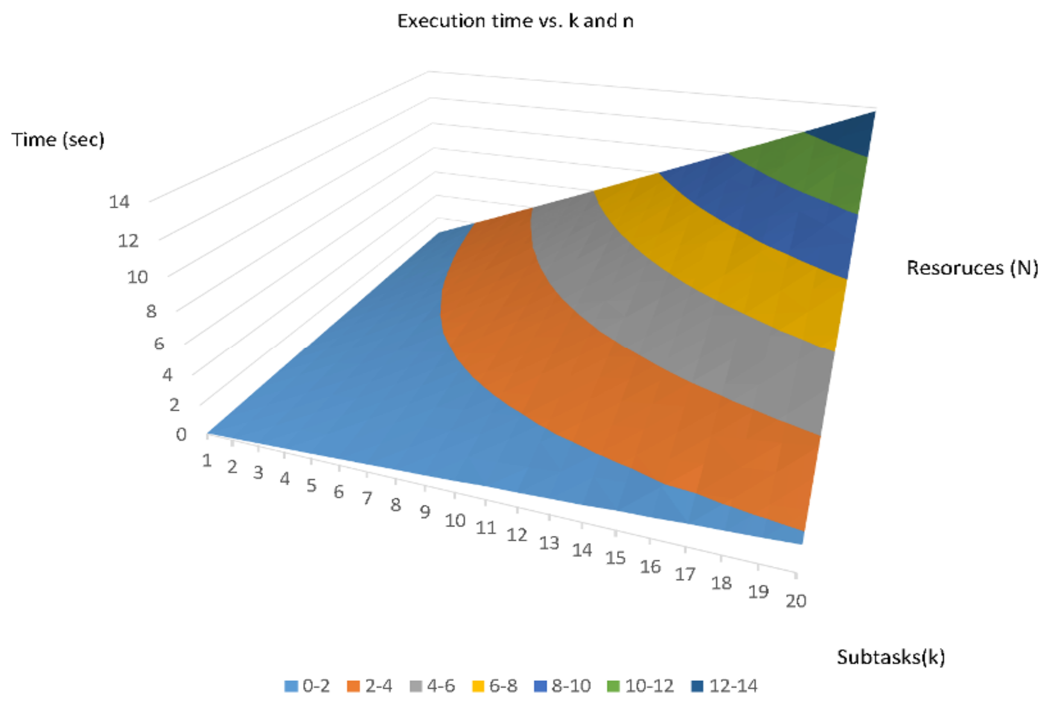

Figure 3: Execution time vs. number of subtasks k and number of resources N