A Novel Algorithm for Optimizing Multiple Services Resource Allocation

Amjad Gawanmeh
Department of Electrical and Computer Engineering,
Khalifa University, UAE and
Department of Electrical and Computer Engineering,
Concordia University, Montreal, Canada

Alain April
Department of Software Engineering, Universite´ du Qu
´ebec,´Ecole de Technologie Sup´erieure, Montreal, Canada

***Abstract*—Resource provisioning becomes more and more challenging problem in cloud computing environment since cloud-based services become more numerous and dynamic. The problem of scheduling multiple tasks for multiple users on a given number of resources is considered NP-Complete problem, and therefore, several heuristic based research methods were proposed, yet, there are still many improvements can be done, since the problem has several optimization parameters. In addition, most proposed solutions are built on top of several assumptions and simplifications by applying computational methods such as game theory, fuzzy logic, or evolutionary computing. This paper presents an algorithm to address the problem of resource allocation across a cloud-based network, where several resources are available, and the cost of computational service depends on the amount of computation. The algorithm is applicable without restrictions on cost vector or compaction time matrix as opposed to methods in the literature. In addition, the execution of the algorithm shows better utility compared to methods applied on similar problems.**

***Keywords*—Cloud computing; Cloud Services; Scheduling; Parallel and Distributed systems**

I. Introduction

Cloud computing [1] has emerged recently as a new paradigm that moved enterprise computing from the classical host-based architecture pattern into the elastic computing pattern. This new service-oriented vision delivers resources and applications on-demand based on the pay-per-use concept. The cloud system should provide the application users with robustness, fault tolerance, execution automation, and powerful computing facilities. This implies various cloud service requirements and QoS to be maintained. Therefore, several challenges arise throughout the design development and deployment of these systems. In addition, unlike conventional software and hardware systems, a wide range of different issues should be considered in the design and operation of cloud based systems. On the other hand, the size of data centers have been continuously increased in order to accommodate the increasing demand while at the same time reducing the management costs. Finally, virtualization has been heavily used in order to increase the utilization of server resources by consolidating many virtual machines into a single physical server. Therefore, as cloud-based services become more numerous and dynamic, resource provisioning becomes more and more challenging.

While QoS problems in service and cloud based systems have been addressed in the literature, the problem of constrained resource allocation problem attracts a lot of attention, as it is considered difficult problem since it is affected by several aspects, such as assumptions about the services, tasks, subtasks, and communication between servers. The problem of scheduling multiple tasks for multiple users on given number of resources is considered difficult, and therefore, several research methods were proposed, yet, there are still many improvements can be done, since the problem has several optimization parameters and in addition, most proposed solutions are built on top of several assumptions and simplifications.

In this paper, we address the problem of resource allocation in which service demanders intend to solve sophisticated parallel computing problem by requesting the usage of resources across a cloud-based network, where several resources are available, and the cost of computational service depends on the amount of computation. The contributions in this paper include providing a simple algorithm for resource allocation for single user with multiple subtasks. Then, propose another algorithm for the multiple users problem using a selection function. The proposed algorithm is illustrated on example that was proposed in the literature, and the proposed method in this paper outperforms previous ones by two aspects: first, it can be applied on the general case without any restriction on the type of the computation time matrix for resource, nor on the cost function, as opposed to previous work. Second, the algorithm provides better schedule in terms of utility.

The rest of this paper is organized as follows: Section II provides a brief review on the state of the art on the subject. Section III describes the formalization of the single and multiple users problem. Section IV presents the proposed algorithm to solve the single user scheduling problem. Section V presents a selection function based algorithm for scheduling the multiple users problem. The proposed method is illustrated on an example in Section VI. Finally Section VII concludes the paper with future work hints.

II. Related Work

In this section we present a brief state of the art review on related work that addressed the problem of resource allocation in cloud computing. In fact, more detailed reviews can be found in several surveys, for instance, [2], [3], [4], [5], [6], and [7].

Walker *et al.* [8] presented a model in which the resultant profit is calculated by obtaining the storage space from cloud. Buyya et al. [9] have proposed an infrastructure for resource allocation in multiple clouds based on commodity pricing method. The work in [10] used virtualization in order to allocate data center resources dynamically based on application demands and support. The authors used skewness to measure the unevenness in the multidimensional resource utilization of a server and combined different types of workloads hoping to improve the overall utilization of server resources by reducing skewness. The work in [11] proposed a resource allocation technique by ranking the tasks of users based on error criteria to maintain the consistency in the priorities of the tasks. The authors in [12] presented a web semantic based resource allocation method using multi-agent technologies to model interoperability between the users and different resource.

Ergu *et al.* [13] proposed a model for task-oriented resource allocation in a cloud computing environment based on available resources and user preferences. The method considers response time as the measure in their allocation procedure. Genetic algorithms based tasks scheduling in cloud systems were addressed in [14], [15], and [16]. The work in [17] addressed distributed convergence to fair allocations of CPU resources for time-sensitive applications for applications with service-level adaptation, that are executed by each application independently. The work in [18] addressed resource allocation in computational mobile grid.

Karthik *et al.* [19] suggested to allocate the resources based on the speed and cost of different VMs in IaaS. It differs from other related works, by allowing the user to select VMs and reduces cost for the user. Zhen Kong *et al.* [20] proposed a mechanism to allocate virtualized resources among selfish VMs in a non-cooperative cloud environment. Hence, VMs care essentially about their own benefits without any consideration for others. Paul *et al.* [21] allocated cloud resources by obtaining resources from remote nodes when there is a change in user demand¿ In this method over-provisioning and under-provisioning of resources can cause several issues. The work in [22] dynamically allocated CPU resources to meet QoS objectives by first allocating requests to high priority applications. This method depends on prioritizing tasks which has several limitations in cloud environment. The work in [23] proposed to use live migration as a resource provisioning mechanism but all of them use policy based heuristic algorithm to live migrate VM which is difficult in the presence of conflicting goals.

Several other computing techniques were adopted for resource allocation in cloud computing. For instance, game theory has also been used in several locations for resource allocation in cloud computing, for instance in [24], and [25]. Several other methods used fuzzy pattern recognition to propose dynamic resource-allocation algorithms for multiple cloud nodes such as [26], [27], and [28]. On the other hand, cloud resource allocation using auction mechanism is addressed in [29], [30], [31], [32], and, [33]. These mechanisms does not ensure profit maximization due to its truth telling property under constraints.

Despite the large amount of work on the subject, there is still lack of practicable solution for cloud computing systems because most cloud-based computational services are multiple QoS-constrained, in order to improve the utility of scheduling algorithms [25]. The problem addressed in this paper considers multi-users with dependent tasks, each is composed of several subtasks to be scheduled on multiple resources. This paper presents a generic solution that can be applied the problem without the constrains on the execution time and cost imposed in previous work. The proposed mechanism is based on utility function trying to find a local optimum solution for the NP-hard problem.

III. Problem Description and Formalization

We first present problem description for resource allocation for a single user, then we describe the general case, for multiple users as presented in previous work. A user u has a service S that is composed of k sub tasks that are parallel and dependent subtasks with equal amount of computation. In fact, this assumption is practical, since in case of unequal amounts of computing, the cost values of resources can be adjusted to normalize the amount of commutations for tasks. There are m computational resources that are available to the user, $\{R_1, R_2, ..., R_m\}$. Each resource, R_j, has a fixed price p_j according to its capacity 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 $t = \{t_1, t_2, ..., t_m\}$. The objective is to assign subtasks for each user to a number of resources such that the the total cost is minimized, where cost represents the expense and execution time for completing tasks for all users. The scheduling problem solution is a non-negative vector v of m elements, each represent the number of subtasks assigned for each resource. The entry v_i is the number of subtasks of the task S allocated to resource R_i. The allocation vector v must satisfy $\sum_{i=1}^{m} v_i = k$.

We derive two vectors from vectors v, t, and p as follows: the first is the completion time vector, $\hat{t}$ and the second is the expense vector $\hat{e}$. The entry $\hat{t}_i$ of $\hat{t}$ is the turnaround time it takes for resource R_i to complete v_j subtasks of the task S. The entry $\hat{e}_i$ of vector $\hat{e}$ is the expense S pays for resource R_j to complete v_j subtasks. These two vectors are defined as follows: $\hat{t} = v \cdot t$, $\hat{e} = v \cdot t \cdot p$. Based on these we calculate two values for schedule v, the total execution time t_{max}, and the total expense e_v. The execution time for task S is the maximum execution time of tasks assigned to resources, $t_{max} = max\{\hat{t}_i | \hat{t}_i \in \hat{t}\}$, where $\hat{t}_i$ denotes the i^{th} element of the vector $\hat{t}$. 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 merit value of the expense using the following utility function: $u(v) = \frac{1}{w_t \times t_v + w_e \times e_v} = \frac{1}{w_t \times max\{\hat{t}_i | \hat{t}_i \in \hat{t}\} + w_e \times \sum_{i=1}^{m} e_i}$ The objective is to find v that maximizes u. while we will present our proposed methodology for a generalizers case, as opposed to the presented solution in there. Next we give an example to illustrate the problem described above. Given five computational resources $(R_1 - R5)$, $m = 5$, and given the price vector $p = (1, 1.2, 1.5, 1.8, 2)$, there is a task (S), that has three subtasks, $k = 3$. The execution time vector for each subtask using above resources is given as $t = (5, 4.2, 3.6, 3, 2.8)$. Assume that a

schedule $v = (1, 1, 1, 0, 0)$ is used, then we can calculate $\hat{t}$ and $\hat{e}$ as follows: $\hat{t} = (5, 4.2, 3.6, 0, 0)$ and $\hat{e} = (5, 5.04, 5.4, 0, 0)$, then we can calculate, $t_v = 5$ and $e_v = 15.44$. Assuming $w_e = w_t = 1$, then $u = \frac{1}{t_v + e_v} = 0.0489$. On the other hand, the schedule $v = (0, 0, 1, 1, 1)$ will yield to $u = 0.05$.

Next, we present the model for the problem with multiple users sharing the same number of resources as described above. Given n users $\{u_1, u_2, ..., u_n\}$, who are interested in executing n services, $\{S_1, S_2, ..., S_n\}$, where every S_i is one single task that is composed of k_i subtasks, that are parallel and dependent subtasks with equal amount of computation. $k = \{k_1, k_2, ..., k_n\}$ represents the number of subtasks for all users. There are m computational resources that are shared by all users, $\{R_1, R_2, ..., R_m\}$ with price vector p. All users who allocate their services to resource R_j proportionally share the capacity and expense of resource R_j. In addition, each resource R_i require specific time, t_{ij}, to execute subtask τ for user u_j using resource R_i, forming the execution time vector T_i for every resource. The collection of all vectors $T_1, T_2, \ldots, T_m$ forms the execution time matrix T. Since all subtasks of S_i are parallel and dependent, the completion time of task S_i is defined similar to above as $max\{T_{ij} | T_{ij} \in T_i\}$, where T_{ij} denotes the element of matrix T in row i and column j.

The objective is to assign subtasks for each user to a number of resources such that the the total cost is minimized, where cost represents the expense and execution time for completing tasks for all users. The scheduling problem solution is a non-negative matrix A of m rows and n columns, each element represents the number of subtasks assigned for each resource. The entry A_{ij} is the number of subtasks of the task S_i allocated to resource R_j. The allocation matrix A must satisfy the following constraint $\sum_{j=1}^{n} A_{ij} = k_j$, where $j = \{1, 2, \ldots, m\}$.

We derive two matrices from matrix T, schedule A, and vector p as follows: the first, is the completion time matrix, $\hat{T}$ and the second is the expense matrix $\hat{E}$. The entry $\hat{T}_{ij}$ of $\hat{T}$ is the turnaround time it takes for resource R_j to complete A_{ij} subtasks of the task S_i. The entry $\hat{E}_{ij}$ of matrix $\hat{E}$ is the expense user S_i pays for resource R_j to complete A_{ij} subtasks. These two matrices are defined as follows:

$$\hat{T}_{ij} = \sum_{l=1}^{n} A_{lj} \cdot T_{lj},$$

$$\hat{E}_{ij} = \frac{\hat{T}_{ij} \cdot p_j}{\sum_{l=1}^{n} A_{lj}}.$$

Based on these two matrices, we calculate two vectors for schedule A, the first vector, T_{max}, is composed of m elements, each element T_{max_i} represents the execution time for task S_i using schedule A, and is defined as the maximum execution time of tasks assigned by user i to all resources, $T_{max_i} = max\{\hat{T}_{ij} | j = \{1, 2, ..., n\}\}$. The second vector, E_{sum}, is composed of m elements, each element, E_{sum_i}, represents the total expense for user S_i, and is defined as the summation of all expenses paid to all resources by user i, $E_i^A = \sum_{j=1}^{m} \hat{E}_{ij}$. We assign weights for schedule costs as follows, w_t for execution time weight, and w_e for expense weight. Then we can use definition for a merit value of the expense use the utility function defined above: $u_i^A = \frac{1}{w_t \times T_i^A + w_e \times E_i^A} = \frac{1}{w_t \times max\{\hat{T}_{ij} | j = \{1,2,...,n\}\} + w_e \times \sum_{j=1}^{m} \hat{E}_{ij}}$. Then the utility of schedule A can be defined as the summation of utilities of all users: $u^A = \sum_{i=1}^{n} u_i^A$.

Next we give an example to illustrate the problem described above for multiple users. We will use a similar problem to the one described in [25], however, problem description in [25] is based on the following two assumptions: (1) the price vector of all resources $p = (p_1, p_2, .., p_m)$ satisfies $p_1 < p_2 < \cdots < p_m$, and (2) the corresponding execution time of any subtask of an arbitrary task S_i satisfies $T_{i1} > T_{i2} > \cdots > T_{im}$. We believe that the above assumption is valid for only limited applications, and the method used based on game theory can have a Nash equilibrium only under the given assumption. The algorithm proposed in this paper can be applied on any price vector, p, and execution time matrix, T, regardless of the order. We will explain the problem statement using the same example, since it is going to be used as illustrative example on the proposed algorithm later. Given the price vector p described above, $k = \{2, 3, 4\}$, the execution time matrix T and schedule A given as below, then we calculate the two matrices $\hat{T}$, $\hat{E}$ and two vectors T_{max} and E_{sum} as follows:

$$T = \begin{pmatrix} 6 & 5 & 4 & 3.5 & 3 \\ 5 & 4.2 & 3.6 & 3 & 2.8 \\ 4 & 3.5 & 3.2 & 2.8 & 2.4 \end{pmatrix}, \quad A = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \end{pmatrix}$$

$$\hat{T} = \begin{pmatrix} 0 & 0 & 0 & 7 & 6 \\ 0 & 8.4 & 7.2 & 6 & 0 \\ 4 & 7 & 6.4 & 0 & 4.8 \end{pmatrix}, \quad \hat{E} = \begin{pmatrix} 0 & 0 & 0 & 6.3 & 6 \\ 0 & 5.04 & 5.4 & 5.4 & 0 \\ 4 & 4.2 & 4.8 & 0 & 4.8 \end{pmatrix}$$

$T^A = (7,\ 8.4,\ 7)$, $E^A = (12.3,\ 15.84,\ 17.8)$

$$u_i^A = \begin{pmatrix} \frac{1}{7+12.3} = 0.0518135 \\ \frac{1}{8.4+15.84} = 0.0412541 \\ \frac{1}{7+17.8} = 0.0403226 \end{pmatrix}, \text{ and finally } u^A = \sum_{i=1}^{n} u_i^A = 0.13339.$$

This example shows the optimization needed in order to enhance utility of the scheduling problem, while there is tradeoff between the execution time and the price for every assignment. First, a proposed solution for the single user case is provided, where the optimum solution can be obtain. In fact, while similar problems have been addressed in the literature, we believe that the solution we provide is the most efficient one. The algorithm for the multiuser problem, which is considered NP compete problem, will be based on a selection function that is designed using the solution for the single user problem. We define a function that calculates the effect of scheduling tasks into resources, and use it for the choice of proper allocation of every service.

Algorithm 1 Singel User Resource Allocation

procedure SINGLE USER ALLOCATION
  Input: k, m, t, p, n.
  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 δ
  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 l such that
      (1) $1 \leq l \leq n$,
      (2) $\forall i \cdot 1 \leq i \leq n \Rightarrow \delta_l \leq \delta_i$,
    $v_l = v_l + 1$, allocate current task to R_l
    if $v_l \times t_l > t_{max}$ ***then***
      $t_{max} = v_l \times t_l$
    endif
    $\delta_l = max(t_{max}, t_l \times (1 + v_l)) + p_l \times t_l \times (1 + v_l)$,
    $j = j + 1$
  UNTIL $j = k$; all tasks are allocated
  END
end procedure

IV. Optimizing Single User Scheduling

In this section we present the solution for the single user optimization problem based on the description in the previous section. The objective is to find the allocation vector v the maximizes the utility $u(v)$. For k subtasks and m resources, it is obvious that the search space for the problem is non-polynomial, hence exploring all possible scenarios is nonlinear.

In order to show how the proposed algorithm finds the optimum solution, we first create a variable t_{max} and initiate it to 0. Then, we define a selection function, δ, as a vector of m elements, $\delta = (\delta_1, \delta_2, \ldots, \delta_m)$, which is defined initially as follows:

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

Then, we start the allocation of k subtasks by following an iterative process, where in every iteration, we assign one subtask into one resource. We choose l, such that satisfies the following conditions: (1) $1 \leq l \leq n$, and (1) $\forall i \cdot 1 \leq i \leq n \Rightarrow \delta_l \leq \delta_i$. This means that δ_l is minimum in δ. Then, we assign the current subtask into resource R_m. After the assignment, we modify the schedule v by incrementing v_l, calculate the new $\hat{t}_l$, δ_l, and $\hat{t}_{max}$. Then we repeat the process again, until all items are processed.

This algorithm has a complexity of worst case $O(nm)$, i.e. it should perform in linear time vs $n \times m$. Applying the above algorithm on the single user example explained above will result in $v = (0, 0, 1, 1, 1)$, with utility $u = 0.05$. In addition considering the scheduling problem for multiusers presented in the previous section, if the problem is solved individually, i.e. every user schedule his subtasks using the algorithm 1 above, then, the resulting schedule will be $v_1 = (0, 0, 0, 1, 1)$, $v_2 = (0, 0, 1, 1, 1)$, $v_3 = (1, 1, 1, 0, 1)$, all combined together will give schedule A provided below. Hence, under this assumption, the schedule for multiuser will have the utility of $u^A = 0.1273$. Obviously, this is not the optimum solution for the multiuser case. For instance, the schedule given in the example in the previous section yields better utility. Therefore, in the next section, we parent a selection function based algorithm for utility optimization for the multiuser scheduling problem described above.

$$A = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \end{pmatrix}$$

V. Optimizing Multi-user Scheduling

The algorithm is developed based on the fact that the single user algorithm alone does not lead into the optimum solution when generalized into the multiple users case. Obviously the reason, is that any subtask scheduled at a given resource, affects the execution time of that resource for other users. Therefore, we will develop the scheduling algorithm based on two issues: scheduling subtasks as pairs, and using a selection function that takes into account the current optimum choice for the single users scheduling problem and combine it with other facts, such as the number of unscheduled subtasks. In fact, The objective is to find the allocation matrix A the maximizes the utility u^A form a given execution time matrix T and price vector p. The problem is described in the literature as NP-Complete, and hence heuristic based solutions can be used to provide the best possible solution. For n users, each has k_i subtasks, where $1 \leq i \leq n$, and m resources, we first create two empty matrices $\hat{T}$ and $\hat{E}$, each of $n \times m$ elements and initialize them to zeros. We also create the two vectors T_{max}, and E_{sum}, each of n elements, and a variable T_E, all initialized to zeros as well. Let Δ be a matrix of $n \times m$ elements, initialized as follows: $\Delta_{ij} = p_j \times T_{ij} + T_{ij}$.

Next, we define a vectors, T_A of m elements initialized to zero, represents the total number of subtasks assigned to every resource. At any time during the scheduling process, $T_{A_j} = \sum_{i=1}^{m} A_{ij}$ and, $T_E = \sum_{j=1}^{m} E_{sum_j}$. Finally, we define the selection function, $\Theta_i = 0.01 \times k_i + \frac{1}{\Delta_{il}}$, that takes into account the number of unscheduled subtasks, where Δ_{il} is minimum in row i in the matrix Δ.

Subtasks are then allocated for users using the selection function, Θ, to choose two users, α and β, such that $1 \leq \alpha, \beta \leq n$, and the following two conditions are satisfied, (1) $\forall i \cdot i > 0, i \leq n \Rightarrow \Theta_\alpha \geq \Theta_i$, and (2) $\forall i \cdot i > 0, i \leq n, i \neq \alpha, \Rightarrow \Theta_\beta \geq \Theta_i$. Then, for each user, α and β , we choose the entry that leads to best single user utility using the single user optimization algorithm, γ_α for task α, and γ_β for task β, such that $1 \leq \gamma_\alpha \leq k_\alpha$, and $1 \leq \gamma_\beta \leq k_\beta$, and the following two conditions are satisfied: (1) $\forall j \cdot j > 0, j \leq m \Rightarrow \Delta_{\alpha\gamma_\alpha} \leq \Delta_{\alpha j}$ and (2) $\forall j \cdot m > 0, j \leq n, j \neq \gamma_\alpha \Rightarrow \Delta_{\beta\gamma_\beta} \leq \Delta_{\beta j}$. Hence, two subtasks with minimum values in Δ are to be scheduled for two user .

In fact, the assignment of the two subtasks are chosen simultaneously for a valid reason, when a single subtask is assigned the allocation matrix will be modified, and hence the local utility matrix Δ will be modified. After several considerations, it was found that sleeting a single subtask and assigning it to the best available entry Δ_{ij} leads to bad

Algorithm 2 Multiple User Resource Allocation Algorithm

procedure ALLOCATION
  Input: k, n, m, T, p.
  Output: A.
  Initialize:
  • Initialize vectors and variables
    $A = \{0\}$, $T_{max} = \{0\}$ $E_{sum} = \{0\}$, $T_E = 0$, $\hat{T} = \{0\}$; initialize all elements to 0

  • Initialize the allocation function vector Δ
    $i = 0$
  REPEAT
    $j = 0$, $i = i + 1$
    REPEAT
      $\Delta_{ij} = p_j \times T_{ij} + T_{ij}$
      $j = j + 1$
    UNTIL $j = m$;
  UNTIL $i = n$;
  • choose two elements using selection function
  REPEAT
    • choose α and β using selection function Θ.
    • choose γ_α such that $\Delta_{\alpha\gamma_\alpha}$ is minimum
    • choose γ_β such that $\Delta_{\beta\gamma_\beta}$ is minimum and $\gamma_\beta \neq \gamma_\alpha$
    • Schedule subtask for user α at Resource R_{γ_α}
    • Schedule subtask for user β at Resource R_{γ_β}
    • updates all variables for by assigning subtask for user α assigned to service γ_α
      $A_{\alpha\gamma_\alpha} += 1$, $T_{A_{\gamma_\alpha}} += 1$,
      $\hat{T}_{\alpha j} = \hat{T}_{\alpha j} + p_i \times T_{ij}$,
      $T_{max_\alpha} = max(T_{max_\alpha}, \hat{T}_{\alpha j})$
      $E_{sum_\alpha} = E_{sum_\alpha} + p_\alpha \times T_{\alpha j}$
      $T_E = T_E + p_\alpha \times T_{\alpha j}$
      $\Delta_{ij} = p_i \times T_{ij} + \sum_{l=1}^{n} max((T_{A_l} + 1) \times T_{il}), T_{max_l}) + T_E.$
      $k_\alpha = k_\alpha - 1$.
    • updates all variables for by assigning subtask for user β to resource γ
    • Repeat above steps using β and γ_β instead of α and γ_α, respectively.
  UNTIL at most one task has unscheduled subtasks
    • choose α such that $k_\alpha > 0$
  REPEAT
    • choose γ_α such that $\Delta_{\alpha\gamma}$ is minimum.
    • updates all variables for α and γ
      $k_\alpha = k_\alpha - 1$.
  UNTIL $k_\alpha = 0$
  END
end procedure

schedule, due to the effect of every single subtask scheduled on the final commutation time (maximum time) for every user. We then update the list of parameters in two steps, each for one assignment. All variables $\hat{T}$, Δ, T_{max} , E_{sum},T_A, and T_E are updated as follows for each task assignment of subtask for user α by scheduling at at service γ_α as follows:

$$A_{\alpha\gamma_\alpha} += 1,\ T_{A_{\gamma_\alpha}} += 1\ ,\ \hat{T}_{\alpha\gamma_\alpha} = \hat{T}_{\alpha\gamma_\alpha} + p_{\gamma_\alpha} \times T_{\alpha\gamma_\alpha},$$

$$T_{max_\alpha} = max(T_{max_\alpha}, \hat{T}_{\alpha\gamma_\alpha}),$$

$$E_{sum_\alpha} = E_{sum_\alpha} + p_\alpha \times T_{\alpha\gamma_\alpha},$$

$$T_E = T_E + p_\alpha \times T_{\alpha\gamma_\alpha},$$

$$\Delta_{ij} = p_i \times T_{ij} + \sum_{l=1}^{n} max((T_{A_l} + 1) \times T_{il}), T_{max_l}) + T_E.$$

Values in the matrix Δ are updated based on the best utility for the single user case described in the previous section. In the next step of the same iteration, the same process is repeated for the assignment of the subtask of user β for service γ_β. In the next iteration, the function Θ is used to select two tasks, and the function Δ is used to select two schedules for each tasks, and then, the best combination is selected. The process is repeated until all tasks are scheduled. In case, there is only one task left with one or more subtasks, then direct assignment using the function Δ is conducted until all subtasks are processed. The outcome depends on the selection function, i.e., the order on which the users are selected for scheduling and the resource their subtasks are allocated to. Hence, we intend to test the algorithm for more than one selection function. Algorithm 2 shows the step by step description of the process and is illustrated on a detailed example in the next Section. In addition, the algorithm is tested with tow different selection functions, and in both it outperforms evolutionary and game theory based algorithms.

VI. ILLUSTRATIVE EXAMPLE

Let us consider the multi-user example given in Section 3, and demonstrate the algorithm above in order to find an appropriate allocation matrix. All initial values for inputs are as given in the example above. We start calculating initial values using the algorithm above, then we choose $\alpha = 3, \beta = 2$ using the selection function. Then we choose two entries in Δ with minimum values: $\gamma_\alpha = 5$ and $\gamma_\beta = 4$. The combination $(\gamma_\alpha, \gamma_\beta) = (5, 4)$ is selected, hence, a subtask for user u_3 will be scheduled to service R_5, followed by a subtask of user u_2 to service R_4. Based on this, the allocation matrix, A, will be updated with these schedules in the next step. This is illustrated in the initial step indicated as λ_0 in Table I below, where the numbers in bold represent the active ones.

In the next step, λ_1, k_3 and k_2 are decremented, leading to $k = \{2, 2, 3\}$. All variables are updated and new selection function values are calculated, which will lead to $\alpha = 3, \beta = 2$, $\gamma_\alpha = 2$, $\gamma_\beta = 3$. Hence, subtask for user u_3 is assigned to R_2, and for user u_2 is assigned to R_3, which will appear in A in the next step λ_2, where k becomes $\{2, 1, 2\}$, and the selection functions gives $\alpha = 3, \beta = 1$, $\gamma_\alpha = 1$, and $\gamma_\beta = 5$. Hence, subtask of u_3 is assigned to R_1, and subtask of u_1 is assigned to R_5. The step λ_3 leads to $k = \{1, 1, 1\}$, $\alpha = 3, \beta = 2$, $\gamma_\alpha = 2$, and $\gamma_\beta = 4$. Hence, subtask of u_3 is assigned to R_2, and subtask of u_2 is assigned to R_4. In step λ_4, $k = \{1, 0, 0\}$, therefore only user u_1 has subtasks, selection function for $\alpha = 1$ will give $\gamma_\alpha = 3$, hence, the last subtask for user u_1 is assigned to R_3. The final allocation matrix obtained as below, which give $u^A = 0.134$. Note that this utility value is better than the result obtained using game theory with Nash equilibrium in [25].

$$A = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 2 & 0 \\ 1 & 2 & 0 & 0 & 1 \end{pmatrix}$$

We observed that the selection function has direct effect on the outcome for the algorithm, therefore, we did execute the algorithm by choosing the selection function as follows $\Theta_i = \frac{1}{1+k_i} + \frac{1}{\Delta_{il}}$, then applying the above steps will lead into the schedule A given below, which has a utility of $u^A = 0.1416$, obviously, this shows an enhancement of around 6.2% over the utility in achieved in [25].

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 1 & 2 & 0 \\ 2 & 2 & 0 & 0 & 0 \end{pmatrix}$$

TABLE I: Execution of the algorithm on the above example

Step	A					Δ					k	Θ	α, γ_α	β, γ_β
λ_0	0	0	0	0	0	12.0	11.0	10.0	9.8	9.0	2	0.131		
	0	0	0	0	0	10.0	9.2	9.0	**8.4**	8.4	3	**0.149**	3, **5**	2, **4**
	0	0	0	0	0	8.0	7.7	8.0	7.8	**7.2**	4	**0.179**		
λ_1	0	0	0	0	0	27.6	26.6	25.6	31.9	30.0	2	0.059		
	0	0	0	1	0	22.6	21.8	**21.6**	24.0	26.2	2	**0.066**	3, **2**	2, **3**
	0	0	0	0	1	21.2	**20.9**	21.2	26.8	22.8	3	**0.078**		
λ_2	0	0	0	0	0	38.9	46.4	44.5	42.6	**40.2**	2	**0.046**		
	0	0	1	1	0	33.3	40.2	35.9	34.7	35.8	1	0.040	3, **1**	1, **5**
	0	1	0	0	1	**31.4**	34.6	38.2	36.4	33.0	2	**0.052**		
λ_3	0	0	0	0	1	59.4	56.4	55.8	53.9	55.6	1	0.028		
	0	0	1	1	0	58.8	56.2	**53.2**	52.0	60.0	1	**0.029**	3, **2**	2, **4**
	1	1	0	0	1	51.4	**50.6**	54.2	52.4	54.4	1	**0.030**		
λ_5	0	0	0	0	1	71.4	76.9	**67.6**	72.2	67.6	1	0.025	1	
	0	0	1	2	0	68.4	73.5	65.0	66.8	69.6	0	0.015	3	
	1	2	0	0	1	63.4	66.1	64.4	67.8	66.4	0	0.016		
λ_6	0	0	1	0	1						0			
	0	0	1	2	0						0			
	1	2	0	0	1						0			

VII. Conclusion and Future Work

While cloud computing technology is increasingly being used, effective resource allocation methods are required in order to maximize profit for cloud service providers, provide energy efficient methods, and at the same time achieving user satisfaction. This paper proposes a model for task-oriented resource allocation in cloud computing, where the problem of scheduling single user with multiple dependant subtask on multiple available resources with different execution time and cost is addressed first. Then, the general problem of scheduling multiple users, each with multiple subtask on multiple available resources is addressed. The proposed solution is provided by introducing a selection pairwise function based on subtasks completion time and task costs. The problem addressed in this paper is described as NP-Complete, hence, the presented solution present a local optimum schedule. The proposed method, compared to existing techniques, is applicable on scheduling problems without any restrictions on the execution time and price, in particular as compared to game theoretic approaches [25], where the execution time must be given in ascending order, and the price in descending in order to find Nash equilibrium. In addition, the proposed method outputs schedule with better utility than game theoretic one.

We intend to provide implementation for the method, and test for problems with large number of users and resources. In addition, we intend to try to modify the selection function to further improve the utility of the schedule by considering the weight of the subtasks to be scheduled in the future including price and completion time. In addition, since formal methods have been proposed thouroughly for the analysis of differnt cloud applications [34], we intend to use formal analysis to model and validate the proposaed algorithm.

References

[1] Borko Furht and Armando Escalante, *Handbook of Cloud Computing*, Springer, 2010.

[2] Shabnam Khan, "A survey on scheduling based resource allocation in cloud computing," *International Journal For Technological Research In Engineering*, vol. 1, no. 1, 2013.

[3] K.C. Okafor, F.N. Ugwoke, A.A. Obayi, V.C. Chijindu, and O.U. Oparaku, "Analysis of cloud network management using resource allocation and task scheduling services," *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 1, pp. 375–386, 2016.

[4] VP Anuradha and D Sumathi, "A survey on resource allocation strategies in cloud computing," in *International Conference on Information Communication and Embedded Systems*. IEEE, 2014, pp. 1–7.

[5] V Vinothina, R Sridaran, and Padmavathi Ganapathi, "A survey on resource allocation strategies in cloud computing," *International Journal of Advanced Computer Science and Applications*, vol. 3, no. 6, pp. 97–104, 2012.

[6] Artan Mazrekaj, Isak Shabani, and Besmir Sejdiu, "Pricing schemes in cloud computing: An overview," *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 2, pp. 80–86, 2016.

[7] Sunilkumar S Manvi and Gopal Krishna Shyam, "Resource management for infrastructure as a service (iaas) in cloud computing: A survey," *Journal of Network and Computer Applications*, vol. 41, pp. 424–440, 2014.

[8] Edward Walker, Walter Brisken, and Jonathan Romney, "To lease or not to lease from storage clouds," *Computer*, vol. 43, no. 4, pp. 44–50, 2010.

[9] Rajkumar Buyya, Rajiv Ranjan, and Rodrigo N Calheiros, "Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services," in *Algorithms and architectures for parallel processing*, pp. 13–31. Springer, 2010.

[10] Zhen Xiao, Weijia Song, and Qi Chen, "Dynamic resource allocation using virtual machines for cloud computing environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1107–1117, 2013.

[11] Anil Singh, Kamlesh Dutta, and Avtar Singh, "Resource allocation in cloud computing environment using ahp technique," *International Journal of Cloud Applications and Computing*, vol. 4, no. 1, pp. 33–44, 2014.

[12] Jorge Ejarque, Javier Álvarez, Raül Sirvent, and Rosa M Badia, "Resource allocation for cloud computing: A semantic approach," *Open Source Cloud Computing Systems: Practices and Paradigms: Practices and Paradigms*, p. 90, 2012.

[13] Daji Ergu, Gang Kou, Yi Peng, Yong Shi, and Yu Shi, "The analytic hierarchy process: task scheduling and resource allocation in cloud computing environment," *The Journal of Supercomputing*, vol. 64, no. 3, pp. 835–848, 2013.

[14] Yujia Ge and Guiyi Wei, "Ga-based task scheduler for the cloud computing systems," in *International Conference on Web Information Systems and Mining*. IEEE, 2010, vol. 2, pp. 181–186.

[15] Joanna Kołodziej, Samee Ullah Khan, Lizhe Wang, and Albert Y Zomaya, "Energy efficient genetic-based schedulers in computational grids," *Concurrency and Computation: Practice and Experience*, vol. 27, no. 4, pp. 809–829, 2015.

[16] Sagnika Saha, Souvik Pal, and Prasant Kumar Pattnaik, "A novel scheduling algorithm for cloud computing environment," in *Computational Intelligence in Data Mining, Volume 1*, pp. 387–398. Springer, 2016.

[17] Georgios C Chasparis, Martina Maggio, Enrico Bini, and Karl-Erik Årzén, "Design and implementation of distributed resource management for time-sensitive applications," *Automatica*, vol. 64, pp. 44–53, 2016.

[18] Dinesh Prasad Sahu, Karan Singh, and Shiv Prakash, "Resource allocation and provisioning in computational mobile grid," *International Journal of Applied Evolutionary Computation*, vol. 6, no. 2, pp. 1–24, 2015.

[19] Karthik Kumar, Jing Feng, Yamini Nimmagadda, and Yung-Hsiang Lu, "Resource allocation for real-time tasks using cloud computing," in *International Conference on Computer Communications and Networks*. IEEE, 2011, pp. 1–7.

[20] Zhen Kong, Cheng-Zhong Xu, and Minyi Guo, "Mechanism design for stochastic virtual resource allocation in non-cooperative cloud systems," in *IEEE International Conference on Cloud Computing*. IEEE, 2011, pp. 614–621.

[21] Paul Marshall, Kate Keahey, and Tim Freeman, "Elastic site: Using clouds to elastically extend site resources," in *IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*. IEEE Computer Society, 2010, pp. 43–52.

[22] Daniel Gmach, Jerry Rolia, and Lucy Cherkasova, "Satisfying service level objectices in a self-managing resource pool," in *IEEE International Conference on Self-Adaptive and Self-Organizing Systems*. IEEE, 2009, pp. 243–253.

[23] Atsuo Inomata, Taiki Morikawa, Minoru Ikebe, Yoshihiro Okamoto, Satoru Noguchi, Kazutoshi Fujikawa, Hideki Sunahara, and Sk Md Mizanur Rahman, "Proposal and evaluation of a dynamic resource allocation method based on the load of vms on iaas," in *IFIP International Conference on New Technologies, Mobility and Security*. IEEE, 2011, pp. 1–6.

[24] Fei Teng and Frédéric Magoulès, "A new game theoretical resource allocation algorithm for cloud computing," in *Advances in Grid and Pervasive Computing*, pp. 321–330. Springer, 2010.

[25] Guiyi Wei, Athanasios V Vasilakos, Yao Zheng, and Naixue Xiong, "A game-theoretic method of fair resource allocation for cloud computing services," *The journal of supercomputing*, vol. 54, no. 2, pp. 252–269, 2010.

[26] Zhanjie Wang and Xianxian Su, "Dynamically hierarchical resource-allocation algorithm in cloud computing environment," *The Journal of Supercomputing*, pp. 1–19, 2015.

[27] Fengyu Guo, Long Yu, Shengwei Tian, and Jiong Yu, "A workflow task scheduling algorithm based on the resources' fuzzy clustering in cloud computing environment," *International Journal of Communication Systems*, vol. 28, no. 6, pp. 1053–1067, 2015.

[28] Dorian Minarolli and Bernd Freisleben, "Virtual machine resource allocation in cloud computing via multi-agent fuzzy control," in *International Conference on Cloud and Green Computing*. IEEE, 2013, pp. 188–194.

[29] Wei-Yu Lin, Guan-Yu Lin, and Hung-Yu Wei, "Dynamic auction mechanism for cloud resource allocation," in *IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*. IEEE, 2010, pp. 591–592.

[30] Hong-Yi Chang, Hsin-Che Lu, Yu-Huei Huang, Yuan-Wei Lin, and Yih-Jou Tzang, "Novel auction mechanism with factor distribution rule for cloud resource allocation," *The Computer Journal*, p. bxt008, 2013.

[31] Maryam Fayazi, Mohammad Reza Noorimehr, and Sayed Enayatollah Alavi, "Resource allocation in cloud computing using imperialist competitive algorithm with reliability approach," *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 3, pp. 323–331, 2016.

[32] Parnia Samimi, Youness Teimouri, and Muriati Mukhtar, "A combinatorial double auction resource allocation model in cloud computing," *Information Sciences*, 2014.

[33] Chonho Lee, Ping Wang, and Dusit Niyato, "A real-time group auction system for efficient allocation of cloud internet applications," *IEEE Transactions on Services Computing*, vol. 8, no. 2, pp. 251–268, 2015.

[34] Amjad Gawanmeh and Ahmad Alomari, "Challenges in formal methods for testing and verification of cloud computing systems," *Scalable Computing: Practice and Experience*, vol. 16, no. 3, pp. 321–332, 2015.