Heuristics for Efficient Resource Allocation in Cloud Computing

Su Seon Yang and Nong Ye*

School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, USA

*Corresponding Author: Nong Ye, School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, Box: 878809; Tempe, AZ 85287-8809, Arizona, USA, Tel: 480-965-7812; Email: nongye@asu.edu

Received Date: Mar 30, 2019 / Accepted Date: Apr 25, 2019 / Published Date: Apr 27, 2019

Abstract

The resource allocation in cloud computing determines the allocation of computer and network resources of service providers to service requests of users for meeting user service requirements. It is not scalable to solve the resource allocation problem as an optimization problem to obtain the optimal solution in real time. This paper presents the development and testing of heuristics for the efficient resource allocation to obtain near-optimal solutions in a scalable manner. We first define the resource allocation problem as a Mixed Integer Programming (MIP) optimization problem and obtain the optimal solutions for various resource-service problem types. Based on the analysis of the optimal solutions, we design heuristics for the efficient resource allocation. Then we evaluate the performance of the resource allocation heuristics using various resource-service problem types and different numbers of service requests and resources. The results show the comparable performance of the heuristics to the optimal solutions. The resource allocation heuristics also demonstrate the better computational efficiency and thus scalability than solving the MIP problems to obtain the optimal solutions.

Keywords: Resource allocation; Cloud computing; Heuristics; Mixed integer programming

Introduction

A cloud is defined as a large pool of virtualized resources such as hardware, platforms, or services that are built on a distributed infrastructure of physical resources [1,2]. Resource management in cloud computing is a challenging task as it has to satisfy objectives of Cloud Computing Service Providers (CCSP) and cloud users in an unpredictable environment due to the fluctuating workload and large shared resources [3]. Resource allocation aims to ensure that the requested services’ requirements are met by CCSP’s resources. The success of cloud computing depends upon the allocation of resources to requested services in an efficient and effective way [4]. Centralized methods of resource allocation require a central entity to either solve the resource allocation problem directly or coordinate a solution of the resource allocation problem with information of all service requests’ requirements and resource states of all
service providers. However, centralized methods face the challenge of scalability to solve a large-scale resource allocation problem and obtain the optimal solution in real time [5]. Decentralized methods rely on interactions among service providers and between service providers and end users to seek the solution of a resource allocation problem, without any central entity. Using such self-management principles in a decentralized manner does not necessarily guarantee the optimal or near-optimal solutions for resource allocation. This paper focuses on centralized methods, especially heuristics to obtain the computational efficiency as well as optimal or near optimal solutions of resource allocation.

Heuristics for centralized resource allocation with computational scalability have not been well addressed in existing work on cloud computing. In comparison with the reviewed work as summarized in Table 1, this study considers the satisfaction of QoS and resource requirements stated in Service Level Agreements (SLA) when determining the allocation of computer and network resources to service requests, rather than allowing SLA violations. To ensure the satisfaction of all service requests’ requirements, this study introduces the use of system models [6] for the precise resource allocation. System models capture how services impose loads on resources and change the state of resources and the system performance. The heuristics in this study deal with the insufficient resource capacity by generating solutions with some service requests dropped as in the optimal solutions when the overall resource capacity is not sufficient to satisfy all service requests, rather than outsourcing the overloaded requests to external clouds. Moreover, the heuristic solutions in this study are generated with scalability. The performance of the heuristics is compared to the optimal solutions in this study.

| Studies | Service Level Agreement (QoS, Resource Requirements, etc.) | System Models | Insufficient resource capacity | Scalability | Compared with optimal solution |
|---------|-------------------------------------------------------------|---------------|--------------------------------|-------------|---------------------------------|
| This study | YES | YES | YES | YES | YES |
| [7-10] | YES | NO | NO | YES | NO |
| [11-16] | NO | NO | NO | NO | NO |
| [17] | YES | NO | YES | YES | YES |
| [18] | YES | NO | NO | YES | NO |
| [19] | YES | NO | YES | YES | NO |
| [20-26] | YES | NO | NO | NO | NO |
| [27,28] | YES | NO | NO | NO | YES |
| [29] | NO | YES | YES | NO | NO |
| [30] | YES | YES | YES | YES | NO |

Section 2 provides the Mixed Integer Programming (MIP) formulation of the optimization problem for resource allocation. The optimal solutions of the optimization problem for various resource-service problem types are examined to gain insights and develop heuristics for the efficient resource allocation. Section 3 describes various resource-service problem cases and different numbers of service requests and resources used to test and evaluate the performance of the heuristics. Section 3 also introduces performance measures and presents the comparison of the heuristic solutions with the optimal solutions. Section 4 concludes the paper.
Heuristics of centralized resource allocation

In this section, we first present the MIP formulation of a resource allocation problem. We then analyze the optimal solutions of various problem cases to develop heuristics that capture the essence of decisions made in the optimal solutions. Finally, we describe the heuristics for various problem types and the implementation of the heuristics.

The formulation of a resource allocation optimization problem

The resource allocation is often addressed as an optimization problem consisting of objectives, decision variables, and constraints. There are mainly three types of optimization objectives in resource allocation:
1) Resource performance objectives such as resource utilization, load balancing, and energy saving by switching on and off servers depending on their workload and resource status [6,9,11,16,17,23,26,31-34].
2) System performance objectives such as the system throughput measured by the number of jobs executed by a system [6,15,20,22,33,35,36], and
3) Application performance objectives such as response time (e.g., execution time and makespan), quality of service (QoS), etc. [6,11,12,14,15,19,22,24,26,29,30,35,37,38-40].

The optimization problem of resource allocation is subject to various types of constraints. Application requirements of resource and QoS are stated in SLA between cloud users and CCSPs covering CPU and memory requirements for host machines, bandwidth, delay, and QoS requirements (e.g., execution time) of services and applications. Capacity limits of resources are given to indicate the maximum capacity of each system resource. System models are provided to describe how services impose resource workloads and thus change the state of system resources which in turn affect the performance of services [41,42]. The predicted behavior and performance of resources, the system and services based on such system models are essential to the precise resource allocation [43].

In the formulation of the optimization problem, decision variables are used to assign requested services to CCSPs and determine the specific level of each service parameter for each requested service that is satisfied by the assigned CCSP. Each type of service is associated with certain service parameters (e.g., a web service is associated with service parameters such as bandwidth and response time (Chen, Farley, & Ye, 2004) that are satisfied at certain levels by system resources on the assigned CCSPs. System models for each CCSP are included in the constraints to predict the resource workloads and service performance for the precise resource allocation. The optimization problem addresses resource, system and application objectives of resource allocation and is solved for each epoch of service requests in the stream of dynamically arriving service requests based on the time-varying state of system resources. The optimization problem of resource allocation is formulated as a MIP problem. The following variables are used in the problem formulation.

- Decision variables
  \[ X_{k,i} = 1 \text{ if service request } k \text{ is assigned to server } i \]
  \[ 0 \text{ if service request } k \text{ is not assigned to server } i \]
  where \( k \) denotes a given service request, \( k = 1, ..., K \); \( i \) denotes a given server, \( i = 1, ..., I \).

- \( A_{k,s,d,i} \): denotes the level of service parameter \( ds \) for service request \( k \) satisfied on server \( i \)
  where \( A_{k,s,d,i} \) is a positive integer indicating the level satisfied, where \( s \) denotes service type \( s, s = 1, ..., S \),
  and \( d \) denotes service parameter \( d \) of service \( s, d = 1, ..., D_s \).

- Given inputs
Heuristics for Efficient Resource Allocation in Cloud Computing

DOI: https://doi.org/10.36811/rjcse.2019.110001

RJCSE: April; 2019: Page No: 01-21

\[ U_{ks} = 1 \text{ if service request } k \text{ uses service } s \]
\[ 0 \text{ if service request } k \text{ does not use service } s. \]

This input is from the user generating service request \( k \), \( k = 1, \ldots, K \), and the input ensures
\[ \sum_k U_{ks} = \text{for a given } k. \]

\[ V_{is} = 1 \text{ if service } s \text{ is provided by server } i \]
\[ 0 \text{ if service } s \text{ is not provided by server } i. \]

This input is from server \( i \), \( i = 1, \ldots, I \).

\( A^l_{sd} \): the limit (i.e. the maximum level) of service parameter \( d \) of service request \( k \) on server \( i \).

This input is from the user generating service request \( k \), \( k = 1, \ldots, K \).

\[ R^l_{iw} : \text{limit of resource variable } w \text{ of server } i, \text{ where } R^l_{iw} \text{ takes a positive real value, and } w \text{ denotes resource variable } w \text{ of server } i, \]
\[ w_i = 1, \ldots, W^l_i. \]

This input is from server \( i \) to indicate the resource capacity, \( i = 1, \ldots, I \).

\[ Q^l_{sp} : \text{limit of QoS variable } p \text{ of service request } k, \text{ where } Q^l_{sp} \text{ takes a positive real value, and } p \text{ denotes QoS variable } p \text{ of service } s, \]
\[ p_s = 1, \ldots, P^s. \]

QoS variable \( p \) of service \( s \), \( p_s = 1, \ldots, P^s \).

This input is from the user generating service request \( k \) to specify QoS requirements, \( k = 1, \ldots, K \).

- Server-client coordination constraint

\[ \sum_i X_{ki} \leq 1 \quad \forall k \quad (1) \]

This constraint requires that service request \( k \) can be assigned to one server \( i \) at most.

- Service constraints

\[ X_{ki} U_{ks} \leq V_{is} \quad \forall k, i, s \quad (2) \]

This constraint requires if \( X_{ki} = 1 \) and \( U_{ks} = 1 \), then \( V_{is} = 1 \) that is, if service request \( k \) is assigned to server \( i \), the service type \( s \) of service request \( k \) must be provided by server \( i \).

\[ A_{dis} \leq A^l_{sd} \quad \forall i, k, d \quad (3) \]

This constraint enforces that the level of service parameter \( d \) of service request \( k \) on server \( i \) should not exceed the limit (i.e. the maximum level).

- Service-resource-QoS relation constraints

\[ R^l_{iw} = X_{ki} F_{mi} (A_{ki}, \ldots, A_{ki}, \ldots) \quad \forall i, k, w_i \quad (4) \]

where \( R^l_{iw} \) denotes the amount of resource variable \( w \) of server \( i \) taken by service request \( k \) in function \( F_{mi} \) of the assigned level of service parameter \( d \) of service request \( k \) on the server only if \( X_{ki} = 1 (\text{service request } k \text{ is assigned to the server } i) \), and \( R^l_{iw} \) takes a positive real value.

This constraint gives relations of service parameters with resource usages.

\[ Q^l_{sp} = X_{ki} G_{pi} (R^l_{iw}, \ldots, R^l_{im}) \quad \forall i, k, p \quad (5) \]

Where \( Q^l_{sp} \) denotes the value of QoS variable \( p \) of service request \( k \) on server \( i \) in function \( G_{pi} \) of the service’s resource usages on the server only if \( X_{ki} = 1 (\text{service request } k \text{ is assigned to the server } i) \), and \( Q^l_{sp} \) takes a positive real value.

This constraint gives the relation of resource usages with QoS performance.

- Resource capacity constraints

\[ \sum R^l_{iw} \leq Q^l_{iw} \quad \forall i, w_i \quad (6) \]

This constraint enforces that the total resource amount on the resource variable \( w \) of server \( i \) used by all service requests should not exceed the maximum resource capacity for this resource variable.

- QoS requirement constraints

\[ Q^l_{sp} \leq Q^l_{sp} \text{ or } Q^l_{sp} \geq X_{ki} Q^l_{sp} \quad \forall i, k, p \quad (7) \]

This constraint enforces that the QoS level of \( p \) for the client \( k \)’s service request at server \( i \) is equal to or less than the maximum QoS requirement or equal to or greater than the
minimum QoS requirement, only if \( X_{ki} = 1 \) (service request \( k \) is assigned to server \( i \)).

- Objective function

\[
\sum_k \sum_{ps} \left| \frac{\sum_{Q_{kps}} - Q'_{kps}}{Q'_{kps} P_s} \right| \quad (8)
\]

The objective function is to make the actual levels of the QoS variables closest to the QoS requirements. The difference between the actual QoS level (\( Q_{kps} \)) and the required QoS level (\( Q'_{kps} \)) for each QoS variable (\( p_s \)) is first normalized by the required QoS level and the total number of QoS variables (\( P_s \)), and then is summed over QoS variables and all service requests. If a requested service is not served, the term inside the two summations in Formula 8 produces the value of 1. If a requested service is not served, the term inside the two summations in Formula 8 produce the value in (0,1). Hence, minimizing the difference between the actual QoS level provided and the required QoS level of each service request in the objective function also makes more service requests to be served.

**Problem cases for obtaining optimal solutions**

Sets of different problem cases are designed to generate the optimal solutions which are then analyzed to identify heuristics for generating optimal or near-optimal resource allocation solutions in a computationally efficient manner. In the problem cases, there are requests for two types of services: the communication intensive service (e.g., video streaming) and the computation intensive service (e.g., data encryption). We consider one service parameter for the communication intensive service (e.g., the size of video data) and one service parameter for the computation intensive service (e.g., the key length for data encryption). We also consider one QoS variable for the communication intensive service (e.g., delay) with three levels of Low (L), Medium (M) and High (H) and one QoS variable for the computation intensive service (e.g., completion time) with three levels of L, M and H. If the actual QoS level provided to a service request is equal to or greater than a required QoS level, then the service request is considered satisfied. Table 2 shows three levels of QoS variables and the maximum level of service parameters for the communication intensive service and the computation intensive service used in the three problem sets.

| Table 2: QoS levels and maximum level of service parameters for two service types. |
|---|---|---|---|
| Set | Communication Intensive Service (s=1) | Computation Intensive Service (s=2) |
| QoS Levels | Limit of service parameter (\( A'_{sid,i} \)) | QoS Levels | Limit of service parameter (\( A'_{sid,i} \)) |
| \( Q'_{kps} \) | \( A'_{sid,i} \) | \( Q'_{kps} \) | \( A'_{sid,i} \) |
| 1 | L (5) | 5 | L (6) | 5 |
| | M (15) | | M (17) | |
| | H (25) | | H (30) | |
| 2 | L (8) | 5 | L (7) | 5 |
| | M (15) | | M (14) | |
| | H (21) | | H (25) | |
| 3 | L (4-5) | 4 | L (3-4) | 4 |
| | M (15-17) | | M (12-14) | |
| | H (21-23) | | H (18-19) | |

In the problem cases, there are two types of servers: communication-centered server and computation-centered server. Both types of servers have two types of resources: the CPU resource and the network bandwidth resource. On the communication-centered server, the bandwidth resource has more capacity than the CPU resource. On the computation-centered
server, the CPU resource has more capacity than the bandwidth resource. Table 3 shows the capacity limits of two resource variables for the CPU resource and the bandwidth resource on the communication-centered server and the computation-centered server, which appear in Formula 6 in the formulation of the MIP optimization problem. Three resource levels of Small (S), Medium (M) and Large (L) are used for the communication-centered server, and three resource levels of S, M and L are used for the computation-centered server.

| Set | Communication Centered | Computation Centered | Number of Servers |
|-----|------------------------|----------------------|-------------------|
| 1   | S (30,100)             | S (116,27)           | 2                 |
|     | M (36,200)             | M (232,32)           |                   |
|     | L (41,350)             | L (406,35)           |                   |
| 2   | S (45,100)             | S (105,50)           | 2                 |
|     | M (65,135)             | M (155,70)           |                   |
|     | L (90,245)             | L (205,80)           |                   |
| 3   | S (24–26, 81–84)       | S (78–79, 25–26)     | 3                 |
|     | M (30–33, 93–95)       | M (87–89, 29–30)     |                   |
|     | L (38–40, 104–107)     | L (95–97, 32–33)     |                   |

In Set 1, we have each communication-centered server set its resource capacity limits to one of three levels: 30, 36 and 41 as the capacity limits of the CPU resource for the S, M and L levels, respectively, and 100, 200 and 350 as the capacity limits of the bandwidth resource for the S, M and L levels, respectively. We also have each computation-centered server set its resource capacity limits to one of three levels: 116, 232 and 406 as the capacity limits of the CPU resource for the S, M and L levels, respectively, and 27, 32 and 35 as the capacity limits of the bandwidth resource for the S, M and L levels, respectively. Similarly in Set 2, we have each communication-centered server set its resource capacity limits to one of three levels: 45, 65 and 90 as the capacity limits of the CPU resource for the S, M and L levels, respectively, and 100, 135 and 245 as the capacity limits of the bandwidth resource for the S, M and L levels, respectively. In Set 3 we randomly select a value in a range as the capacity limits of two resources (the CPU resource and the bandwidth resource) at each of the three capacity levels (S, M and L). For each communication-centered server, a specific value is randomly selected from 25 to 26 as the capacity limit of the CPU resource and from 81 to 84 as the capacity limit of the bandwidth resource for the S level, from 30 to 33 as the capacity limit of CPU resource and from 93 to 95 as the capacity limit of the bandwidth resource for the M level, and from 38 to 40 as the capacity limit of the CPU resource and from 104 to 107 as the capacity limit of the bandwidth resource for the L level. For each computation-centered server, a specific value is randomly selected from 25 to 26 as the capacity limit of the CPU resource and from 81 to 84 as the capacity limit of the bandwidth resource for the S level, from 87 to 89 as the capacity limit of the CPU resource and from 29 to 30 as the capacity limit of the bandwidth resource for the M level, and from 95 to 97 as the capacity limit of the CPU resource and from 32 to 33 as the capacity limit of the bandwidth resource for the L level. Note that each server can serve both communication intensive services and computation intensive services.
As in our previous work (Yau et al., 2009) using resource and QoS impact models of various services, this study also uses two variables of resource usages: the CPU resource usage ($R_{ki}$) and the bandwidth resource usage ($R_{ki2}$) to determine the QoS performance of two services. A communication intensive service requires the more usage of the bandwidth resource than the usage of the CPU resource, while the computation intensive service requires the more usage of the CPU resource than the usage of the bandwidth resource. The resource and QoS impact models of services in Formula 4 and Formula 5 are needed in the MIP optimization formulation. Many forms of models (e.g., linear or nonlinear) can be used in Formula 4 and Formula 5. Without loss of generality, linear models are used in this study (see Table 4). As shown in Table 4, various parameter values in these linear models are used in the three different problem sets such that heuristics identified from analyzing the optimal solutions of the resource allocation problems are not influenced by parameter values.

### Table 4: Resource and QoS impact models of two service types.

| Set | Communication Intensive Service ($s = 1$) | Computation Intensive Service ($s = 2$) |
|-----|----------------------------------------|----------------------------------------|
| 1   | $R_{ki1} = 0.1 * A_{kd,i}$<br>$R_{ki2} = 5.0 * A_{kd,i}$<br>$Q_{kp,i} = 2R_{ki1} + R_{ki2}$ | $R_{ki1} = 5.8 * A_{kd,i}$<br>$R_{ki2} = 0.3 * A_{kd,i}$<br>$Q_{kp,2i} = 2R_{ki1} + R_{ki2}$ |
| 2   | $R_{ki1} = 0.3 * A_{kd,i}$<br>$R_{ki2} = 4.0 * A_{kd,i}$<br>$Q_{kp,1i} = 2R_{ki1} + R_{ki2}$ | $R_{ki1} = 3.8 * A_{kd,i}$<br>$R_{ki2} = 0.5 * A_{kd,i}$<br>$Q_{kp,2i} = 2R_{ki1} + R_{ki2}$ |
| 3   | $R_{ki1} = 0.3 * A_{kd,i}$<br>$R_{ki1} = 5.0 * A_{kd,i}$<br>$Q_{kp,i} = 3R_{ki1} + R_{ki2}$ | $R_{ki1} = 4.7 * A_{kd,2i}$<br>$R_{ki2} = 0.1 * A_{kd,2i}$<br>$Q_{kp,2i} = R_{ki1} + 2R_{ki2}$ |

The specific number of service requests for each capacity-request condition of each problem case is determined by looking into resource usages based on $F$ functions of service-resource relations and the corresponding QoS values based on $G$ functions of resource-QoS relations in Table 4 to meet the QoS requirements of all the service requests in Table 2 with the given resource capacity in Table 3. Table 5 gives two sets of the parameters used in the heuristics to allow us examining possible effects of different parameter values on the resource allocation solutions.

### Table 5: Two sets of parameters used in obtaining heuristic solutions.

| Parameter | $\alpha$ | $\beta$ | $\gamma$ | $\delta_1$ | $\delta_2$ | $\delta_3$ |
|-----------|---------|--------|---------|--------|--------|--------|
| Set 1     | 0.90   | 0.80   | 0.85   | 0.75   | 0.80   | 0.85   |
| Set 2     | 0.98   | 0.90   | 0.80   | 0.70   | 0.80   | 0.90   |

The MIP problems are implemented in ILOG OPL Development Studio IDE Version 6.1. ILOG CPLEX 11.2.0 is used as a solver to the MIP problem. A laptop computer used to run the software is a Samsung Q320 with Intel Core 2 Duo T6500 2.1 GHz processor, 4 GB RAM, and Windows 7. The ILOG OPL and CPLEX are integrated into C# code in Microsoft Visual Studio 2010. The C# code first loads all the
necessary input files of the problem including
the given inputs as well as service-resource
relation functions and resource-QoS relation
functions for each service type. With the loaded
input files, the C# code then calls ILOG OPL
and CPLEX to run the MIP optimization and
solve the problem to generate an optimal
solution. The computation time of obtaining the
optimal solution is recorded by the C# code. Note
that times required for loading input files
and generating output files are also included in
the computation time of obtaining the optimal
solution.

**Analysis of optimal solutions to develop
heuristics**

The optimal solution to each problem is
analyzed to gain insights into the resource
allocation decision made in the optimal
solution. Based on the analysis and insights
gained from the optimal solutions, we develop
heuristics for Case A, B and C of resource-
service conditions. For all problems of Case A
where each server has the sufficient resource
capacity to satisfy all service requests of all
clients, the optimal solutions assign most
service requests to one server (i.e., the first
server with \(i=1\) in the problem formulation).
Hence, the following heuristic is identified for
Case A as shown in Table 6. The optimal
solutions to the problems of Case B, where each
server does not have the sufficient resource
capacity to satisfy all service requests but the
total resource capacity of all servers is
sufficient to satisfy all service requests, reveal
six different heuristics as shown in Table 7. For
Case C problems, not only each server but all
servers together do not have the sufficient
resource capacity to satisfy all service requests.
Table 8 gives the heuristics for Case C
problems based on the analysis of their optimal
solutions to these problems.

| **Table 6**: The heuristics for Case A problems. |
|---------------------------------------------|
| **Heuristic** | **Description** |
| A-1 | Designate one server as the dominant server, and select the dominant server to serve a service request with the probability of \(\alpha\) and another server to serve the service request with the probability of \((1-\alpha)\). The parameter, \(\alpha\), takes a value in \((0, 1]\) and is closer to 1 than 0 (e.g., 0.9). |

| **Table 7**: The heuristics for Case B problems. |
|---------------------------------------------|
| **Heuristic** | **Description** |
| B-1(a) | Select a server randomly to serve a service request. |
| B-1(b) | Server to serve service request with the probability of \(\beta\) and another server to serve the service request with the probability of \((1-\beta)\). The parameter, \(\beta\), takes a value in \((0, 1]\) and is closer to 1. This heuristic is same as A-1 applying to B cases. |
| B-2(a) | Select a server of one server type (e.g., communication centered server) randomly to serve a service request of the same type (e.g., communication intensive service) with the probability of \(\gamma\) and a server of a different server type (e.g., computation centered server) to serve the service request with the probability of \((1-\gamma)\). The parameter, \(\gamma\), takes a value in \((0, 1]\) and is closer to 1. |
| B-2(b) | Designate a server of each server type as the dominant server of the server type, select the dominant server of one server type to serve a service request of the same type with the probability of \(\gamma\), and the dominant server of a different server type to serve the service request with the probability of \((1-\gamma)\). The parameter, \(\gamma\), takes a value in \((0, 1]\) and is closer to 1. |
Select a server of one server type (e.g., communication centered server) randomly to serve a service request of the same type (e.g., communication intensive service) with a given QoS level of L, M or H with the corresponding probability of $\delta_1, \delta_2$ or $\delta_3$ respectively, and a server of a different server type (e.g., computation centered server) to serve the service request with the given QoS level of L, M or H with the probability of $(1-\delta_1), (1-\delta_2)$ or $(1-\delta_3)$, respectively. Each parameter, $\delta_1, \delta_2, \delta_3$ takes a value in $(0, 1]$, and $\delta_1 \leq \delta_2 \leq \delta_3$. 

**Table 8:** The heuristics for Case C problems.

| Heuristic | Description |
|-----------|-------------|
| C-1 | Select a server randomly to serve a service request. If the selected server is full, then a service request is randomly assigned to another server. Drop a service request if its QoS requirement cannot be satisfied by the available resource capacity. This heuristic is same as B-1(a) with the addition of dropping a service request due to the insufficient capacity. |
| C-2 | Select a server of one server type (e.g., communication centered server) randomly to serve a service request of the same type (e.g., communication intensive service) with the probability of $\gamma$ and a server of a different server type (e.g., computation centered server) to serve the service request with the probability of $(1-\gamma)$. Drop a service request if its QoS requirement cannot be satisfied by the available resource capacity. The parameter, $\gamma$, takes a value in $(0, 1]$ and is closer to 1. This heuristic is same as B-2(a) with the addition of dropping a service request due to the insufficient capacity. |
| C-3 | Select a server of one server type (e.g., communication centered server) randomly to serve a service request of the same service type (e.g., communication intensive service) with a given QoS level of L, M or H at the corresponding probability of $\delta_1, \delta_2$ or $\delta_3$ respectively, and a server of a different server type (e.g., computation centered server) to serve the service request with the given QoS level of L, M or H with the probability of $(1-\delta_1), (1-\delta_2)$ or $(1-\delta_3)$, respectively. Drop a service request if its QoS requirement cannot be satisfied by the available resource capacity. Each parameter, $\delta_1, \delta_2, \delta_3$ takes a value in $(0, 1]$, and $\delta_1 \leq \delta_2 \leq \delta_3$. This heuristic is same as B-3 with the addition of dropping a service request due to the insufficient capacity. |

The performance evaluation of heuristic solutions in comparison with optimal solutions

**Problem cases**

We design various problem cases, including the sets (Sets 1 and 3) of problem cases described in Section 2 and two additional sets (Sets 4 and 5) of larger problem cases with large numbers of servers (ten and twenty servers) and service requests, to evaluate the solution obtained from the heuristics in comparison with the optimal solution for each problem case. Two different sets of parameter values in Table 5 are used to examine possible effects of different parameter values on the resource allocation solutions obtained from applying the heuristics. Tables 9-11 show the set-up of the additional larger problem cases.
The heuristics are implemented in an algorithm that is implemented in the C# code in Microsoft Visual Studio 2010. The same laptop computer running the ILOG CPLEX software to obtain the optimal solutions, a Samsung Q320 with Intel Core 2 Duo T6500 2.1 GHz processor, 4 GB RAM, and Windows 7, is used to run the heuristic algorithm. The C# code first loads all the necessary input files including the given inputs as well as service-resource relation functions and resource-QoS relation functions for each service type. With the loaded input files, the C# code then use an appropriate heuristic to solve each problem case and generate the heuristic solution. The computation time of obtaining a heuristic solution is recorded by the C# code. Note that times required for generating output files are also included in the computation time of obtaining the heuristic solution.

### Table 9: QoS variables and maximum level of service parameters for two service types.

| Set | Communication Intensive Service ($s = 1$) | Computation Intensive Service ($s = 2$) |
|-----|-------------------------------------------|-----------------------------------------|
| QoS Levels | Limit of service parameter | QoS Levels | Limit of service parameter |
| $Q_{kp1}$ | $A_{kd1}$ | $Q_{kp1}$ | $A_{kd1}$ |
| 4 | L (6–7) | 3 | L (6–7) | 3 |
| M (13–14) | | M (14–15) | |
| H (20–21) | | H (22–23) | |
| 5 | L (5–6) | 3 | L (5–6) | 3 |
| M (12–13) | | M (11–12) | |
| H (18–19) | | H (17–18) | |

### Table 10: Capacity limits of resource variables ($R_{q1}$,$R_{q2}$) and number of servers.

| Set | Communication Centered | Computation Centered | Number of Servers |
|-----|------------------------|----------------------|-------------------|
| 4 | S (23–24, 104–105) M (31–32, 166–167) L (39–40, 208–209) | S (115–116, 21–22) M (183–184, 28–29) L (229–230, 35–36) | 10 |
| 5 | S (19–20, 95–96) M (25–26, 139–140) L (31–32, 177–178) | S (91–92, 19–20) M (133–134, 26–27) L (169–170, 32–33) | 20 |

### Table 11: Resource and QoS impact models of two service types.

| Set | Communication Intensive Service ($s = 1$) | Computation Intensive Service ($s = 2$) |
|-----|-------------------------------------------|-----------------------------------------|
| $R_{q1}$ | $R_{q2}$ |
| $Q_{kp1}$ = 3$R_{k1}$ + 2$R_{k2}$ | $Q_{kp1}$ = $R_{k1}$ + 3$R_{k2}$ |
| 4 | $R_{q1}$ = 0.2 $A_{kd1}$ | $R_{q1}$ = 7.6 $A_{kd1}$ |
| $R_{q2}$ = 6.9 $A_{kd1}$ | $R_{q2}$ = 0.1 $A_{kd1}$ |
| $Q_{kp1}$ = $R_{k1}$ + 3$R_{k2}$ | $Q_{kp1}$ = $R_{k1}$ + 2$R_{k2}$ |
| 5 | $R_{q1}$ = 0.1 $A_{kd1}$ | $R_{q1}$ = 6.0 $A_{kd1}$ |
| $R_{q2}$ = 6.3 $A_{kd1}$ | $R_{q2}$ = 0.1 $A_{kd1}$ |
| $Q_{kp1}$ = $R_{k1}$ + 2$R_{k2}$ | $Q_{kp1}$ = $R_{k1}$ + 2$R_{k2}$ |
Performance measures for the solution optimality

Regarding the solution optimality, two measures are introduced: the number of dropped service requests and the average ratio of actual QoS values that all served service requests receive based on the resource-service assignment. We examine Formula 8 below which is also described in Section 2 and presents the objective function of the optimization problem.

Minimize \[ \sum_k \sum_{p} \left| \frac{\sum_i Q_{kpsi} - Q_{kpsi}^j}{Q_{kpsi}^j * P_i} \right| \]  \hspace{1cm} (8)

If service request \( k \) is served by a server, the minimum limit of QoS requirement of this service must be satisfied, thus \( \sum_i Q_{kpsi} \geq Q_{kpsi}^j \), and the formula in Eq. 8 can be written as Eq. 9.

\[ \sum_k \sum_{p} \left( \frac{\sum_i Q_{kpsi}}{Q_{kpsi}^j * P_i} - \frac{1}{P_i} \right) \]  \hspace{1cm} (9)

Since in this study we have one QoS variable \( (ps = 1) \) for each service type, Eq. 9 becomes Eq. 10 for all served service requests.

\[ \sum_k \left( \frac{\sum_i Q_{kpsi}}{Q_{kpsi}^j} - 1 \right) \]  \hspace{1cm} (10)

If service request \( k \) is dropped (i.e., not served by any server), \( \sum_i Q_{kpsi} = 0 \) thus \( \sum_i Q_{kpsi} < Q_{kpsi}^j \), the formula in Eq. 8 can be written as Eq. 11.

\[ \sum_k \sum_{p} \left( \frac{1}{P_i} - \frac{\sum_i Q_{kpsi}}{Q_{kpsi}^j * P_i} \right) \]  \hspace{1cm} (11)

Since in this study we have one QoS variable \( (ps = 1) \) for each service type, Eq. 11 becomes Eq. 12 for all dropped service requests. Eq. 12 indicates the total number of dropped service requests.

\[ \sum_k 1 \]  \hspace{1cm} (12)

Hence, Formula 8 for the objective function is the sum of two parts: Formula 10 for all served service requests and Formula 12 for all dropped service requests. Formula 12, measuring the number of dropped service requests, is the first performance measure used to evaluate the solution optimality. From Formula 10, the second performance measure of Service AVERAGE ratio (SAV ratio) is derived and shown below in Formula 13.

\[ \sum_k \left( \frac{\sum_i Q_{kpsi}}{Q_{kpsi}^j} \right) / \sum_k \]  \hspace{1cm} (13)

The main part of Formula 10 considers the ratio of the provided QoS to the required QoS for each served service request, which is then summed over all served service requests. The smaller the SAV ratio, the better the minimization of the objective function is achieved. Formula 13 is the second performance measure of the solution optimality and gives the average ratio of the provided QoS to the required QoS for all served service requests.

Therefore, two performance measures, the total number of dropped service requests and the SAV ratio, are used to compare and evaluate the optimal solutions and the heuristic solutions with regard to the solution optimality.

Solution optimality

Table 12 shows the comparisons between the optimal solutions and the heuristic solutions using the first set of parameters in Table 5 for the problem cases in Sets 1, 3, 4 and 5. The heuristic solutions are compared with the optimal solutions in the following four aspects: the percentage of problem cases whose heuristic solutions are same as the optimal solutions, the percentage of problem cases whose heuristic solutions are different from the optimal solutions, the difference in the number of dropped service requests of the heuristic solution from that of the optimal solution for each problem case shown by the average, range
and standard deviation (denoted by SD in Table 12) over all the problem cases in each type (Case A, Case B and Case C), and the comparison in the SAV ratio of the heuristic solution from that of the optimal solution for each problem case shown by the average, range and standard deviation (denoted by SD in Table 12) over all the problem cases in each type (Case A, Case B and Case C).

| Set | Case | Same Decision | Different Decision | Difference in Dropped Requests by (heuristic solution - optimal solution) in number (%) | SAV ratio |
|-----|------|----------------|-------------------|------------------------------------------------------------------------------------------------|----------|
| 1   | A    | 44.44% (112/252) | 55.56% (140/252)  | 0                                                                                               | Heuristic Solutions Avg. Range St.dev 1.04 1.02~1.08 0.02 |
|     |      | Same above      | Heuristic Solutions Avg. Range St.dev     |                                                                                                 | Optimal Solutions Same above          |
|     | B    | 58.33% (147/252) | 41.67% (105/252)  | 0                                                                                               | Heuristic Solutions Avg. Range St.dev 1.04 1.02~1.08 0.02 |
|     |      | Same above      | Heuristic Solutions Avg. Range St.dev     |                                                                                                 | Optimal Solutions Same above          |
|     | C    | 52.78% (133/252) | 47.22% (119/252)  | Avg. Range St.dev 2.25 (4.39%) 0.00 ~25.58 4.61                                             | Heuristic Solutions Avg. Range St.dev 1.04 1.02~1.08 0.02 |
|     |      | Same above      | Heuristic Solutions Avg. Range St.dev     |                                                                                                 | Optimal Solutions Same above |
| 2   | A    | 37.70% (95/252)  | 62.30% (157/252)  | 0                                                                                               | Heuristic Solutions Avg. Range St.dev 1.20 1.03~1.63 0.14 |
|     |      | Same as above   | Heuristic Solutions Avg. Range St.dev     |                                                                                                 | Optimal Solutions Same as above       |
|     | B    | 53.57% (135/252) | 46.43% (117/252)  | 0                                                                                               | Heuristic Solutions Avg. Range St.dev 1.21 1.05~1.51 0.13 |
|     |      | Same as above   | Heuristic Solutions Avg. Range St.dev     |                                                                                                 | Optimal Solutions Same as above       |
|     | C    | 61.90% (156/252) | 38.10% (96/252)   | Avg. Range SD 0.62 (1.45%) -0.92 ~10.87 1.62                                                | Heuristic Solutions Avg. Range SD 1.19 1.04~1.49 0.12 |
|     |      | Same as above   | Heuristic Solutions Avg. Range SD         |                                                                                                 | Optimal Solutions Same as above       |
| 3   | A    | 43.25% (109/252) | 56.75% (143/252)  | 0                                                                                               | Heuristic Solutions Avg. Range SD     | 1.11 1.03~1.32 0.06                     |
|     |      | Same as above   | Heuristic Solutions Avg. Range SD         |                                                                                                 | Optimal Solutions Same as above       |
For the 252 Case A problem cases in Set 1, the heuristic solutions for 44.40% of these problem cases are same as the optimal solutions, and the heuristic solutions for 55.56% of these problem cases are different from the optimal solutions. The heuristic solutions serve all the service requests with the same QoS values of the optimal solutions and thus have the same SAV ratio of 1.04 in the average of all the problem cases with the range of 1.02 to 1.08 and the standard deviation of 0.02. As shown in Formula 13, the SAV ratio is an average ratio of the provided QoS to the required QoS for all the served service requests. If a service request is served, its provided QoS is equal to or greater than its required QoS, and thus its SAV ratio is equal to or greater than 1. Like the optimal solutions, the heuristic solutions do not drop any service request due to the sufficient resource capacity at each server to satisfy all the service requests in all Case A problems.

For the 252 Case B problem cases in Set 1, the heuristic solutions for 58.33% of these problem cases are same as the optimal solutions, and the heuristic solutions for 41.67% of these problem cases are different from the optimal solutions. The heuristic solutions serve the all service requests with the same average, range and standard deviations of the QoS value of the optimal solutions. The heuristic solutions do not drop any service request as same as the optimal solutions due to sufficient resource capacity over all servers to satisfy all service requests of all clients in Case B.

For the 252 Case C problem cases in Set 1, the heuristic solutions for 52.78% of the problem cases are same as the optimal solutions, and the heuristic solutions for 47.22% of the problem cases are different from the optimal solutions. The heuristic solutions have the same SAV ratio of the optimal solutions in the average, range...
and standard deviation. Since neither any server nor all servers together have the sufficient resource capacity to satisfy all the service requests in Case C, both the optimal solutions and the heuristic solutions drop some service requests. The heuristic solutions drop only about 2 more service requests (or about 4.39% of the service requests) in average than the optimal solutions, and the range of differences is 0.00 to 25.58 with the standard deviation of 4.61. The heuristic solutions drop a few more service requests than the optimal solutions because the heuristic algorithm processes service requests sequentially and thus make temporally local decisions rather than making temporally global decisions by considering all service requests at one time in the optimal solutions. Hence, the heuristic solutions serve two fewer service requests than the optimal solutions in average but provide similar QoS values to those of the optimal solutions for all the served service requests.

For Set 3 of problem cases, the heuristic solutions for 37.70% of the 252 Case A problem cases and 53.57% of the 252 Case B problem cases are same as the optimal solutions, and the heuristic solutions for 62.30% of the 252 Case A problem cases and 46.43% of the 252 Case B problem cases are different from the optimal solutions. The heuristic solutions serve all the service requests with the same average, range and standard deviation of the SAV ratios of the optimal solutions in Case A and Case B. Like the optimal solutions, the heuristic solutions do not drop any service request due to the sufficient resource capacity of each server in Case A or all the servers together in Case B to satisfy all the service requests.

For Case C problem cases in Set 3, the heuristic solutions for 61.90% of the 252 problem cases are same as the optimal solutions, and the heuristic solutions for 38.10% of the problem cases are different from the optimal solutions. The heuristic solutions have similar average, range and standard deviation of the SAV ratios of the optimal solutions. Since neither any server nor all the servers together have the sufficient resource capacity to satisfy all the service requests, both the optimal solutions and the heuristic solutions drop some service requests. The heuristic solutions drop about one more service request in average (1.45% of all the service requests) than the optimal solutions, and the range of differences is -0.92 to 10.87 with the standard deviation of 1.62. Hence, the heuristic solutions serve one fewer service request than the optimal solutions in average but provide similar QoS values of the optimal solutions for all the served service requests. The negative value of -0.92 in the difference of the dropped service requests indicates that the heuristic solution serves about one more service request than the optimal solution, which is unusual. Heuristic algorithm processes service requests sequentially and thus make temporally local decisions to select a service request by considering only resource consumption rather than making temporally global decisions by considering both resource consumption and objective value of service requests at one time in the optimal solutions.

For Case A and Case B problem cases in Set 4, the heuristic solutions for 43.25% of the 252 Case A problem cases and 17.06% of the 252 Case B problem cases are same as the optimal solutions, and 56.75% of the Case A problem cases and 82.94% of the Case B problem cases are different from the optimal solutions. The heuristic solutions serve all the service requests with the same average, range and standard deviation of the SAV ratios of the optimal solutions in both Case A and Case B. Like the optimal solutions, the heuristic solutions do not drop any service request in both Case A and Case B due to the sufficient resource capacity at each server in Case A or all the servers together in Case B to satisfy all the service requests.

For Case C problem cases in Set 4, the heuristic solutions for 57.94% of the 252 problem cases are same as the optimal solutions, and the heuristic solutions for 42.06% of the problem cases are different from the optimal solutions. The heuristic solutions have similar average, range and standard deviation of the SAV ratios of the optimal solutions.
of the optimal solutions. Since neither any server nor all the servers together have the sufficient resource capacity to satisfy all service requests in Case C, both the optimal solutions and the heuristic solutions drop some service requests. The heuristic solutions drop about five more service request in average (2.82% of all the service requests) than the optimal solutions, and the range of differences is 0.00 to 79.92 with the standard deviation of 12.71. Hence, the heuristic solutions serve five fewer service requests than the optimal solutions in average but provide similar QoS values of the optimal solutions for all the served service requests.

For Case A and Case B problem cases in Set 5, the heuristic solutions for 38.89% of the 252 Case A problem cases and 20.24% of the 252 Case B problem cases are same as the optimal solutions, and the heuristic solutions for 61.11% of the Case A problem cases and 79.76% of the Case B problem cases are different from the optimal solutions. The heuristic solutions have the same average, range and standard deviation of the SAV ratios of the optimal solutions. The heuristic solutions do not drop any service request due to the sufficient resource capacity at each server in Case A or all the servers together in Case B to satisfy all the service requests.

For Case C problem cases in Set 5, the heuristic solutions for 58.33% of the 252 problem are same as the optimal solutions, and the heuristic solutions for 41.67% of the problem cases are different from the optimal solutions. The heuristic solutions have similar average, range and standard deviation of the SAV ratios of the optimal solutions. Since neither any server nor all the servers together have the sufficient resource capacity to satisfy all the service requests, both the optimal solutions and the heuristic solutions drop some service requests. The heuristic solutions drop about ten more service request in average (2.69% of all the service requests) than the optimal solutions, and the range of differences is 0.00 to 168.85 with the standard deviation of 24.74. Hence, the heuristic solutions serve ten fewer service requests than the optimal solutions in average but provide similar QoS values of the optimal solutions for all the served service requests.

The four sets (1,3,4, and 5) of problem cases are run using the second set of parameters in Table 5 to obtain the heuristic solutions. Table 13 gives the comparisons between the optimal solutions and the heuristic solutions. The performance results of the heuristic solutions with the second set of parameters are similar to those with the first set of parameters.

| Table 13: Comparisons between the optimal and the heuristic solutions with the second set of parameters. |
|---|---|---|---|---|---|
| Set | Case | Same Decision | Different Decision | Difference in Dropped Requests by (heuristic solution - optimal solution) in number (%) | SAV ratio |
| A | 50.00% (126/252) | 50.00% (126/252) | 0 | Heuristic Solutions | Avg. Range SD | 1.04 1.04–1.08 0.02 |
| 1 | 63.10% (159/252) | 36.90% (93/252) | 0 | Heuristic Solutions | Avg. Range SD | 1.04 1.04–1.08 0.02 |
| Optimal Solutions | Same above | Optimal Solutions | Same above |
|   | Heuristic Solutions | Optimal Solutions |
|---|---------------------|-------------------|
| C | 55.16% (139/252) 44.84% (113/252) Avg. Range SD 2.23 (4.35%) 0.00 ~26.46 4.62 | Heuristic Solutions Avg. Range SD 1.04 1.02~1.08 0.02 |
|   | Optimal Solutions Same above |                      |
| A | 42.46% (107/252) 57.54% (145/252) 0 | Heuristic Solutions Avg. Range SD 1.20 1.03~1.59 0.14 |
|   | Optimal Solutions Same as above |                      |
| B | 51.98% (131/252) 48.02% (121/252) 0 | Heuristic Solutions Avg. Range SD 1.20 1.03~1.47 0.13 |
|   | Optimal Solutions Same as above |                      |
| C | 61.51% (155/252) 38.49% (97/252) Avg. Range St.dev 0.77 (1.73%) -0.84 ~14.19 2.04 | Heuristic Solutions Avg. Range SD 1.18 1.03~1.47 0.11 |
|   | Optimal Solutions Same as above |                      |
| A | 57.54% (145/252) 42.46% (107/252) 0 | Heuristic Solutions Avg. Range SD 1.11 1.03~1.32 0.06 |
|   | Optimal Solutions Same as above |                      |
| B | 19.84% (50/252) 80.16% (202/252) 0 | Heuristic Solutions Avg. Range SD 1.11 1.04~1.24 0.05 |
|   | Optimal Solutions Same as above |                      |
| C | 57.94% (146/252) 42.06% (106/252) Avg. Range SD 6.02 (3.09%) 0.00 ~116.38 15.66 | Heuristic Solutions Avg. Range SD 1.10 1.04~1.22 0.04 |
|   | Optimal Solutions Same as above |                      |
| A | 37.70% (95/252) 62.30% (157/252) 0 | Heuristic Solutions Avg. Range SD 1.10 1.02~1.32 0.06 |
|   | Optimal Solutions Same as above |                      |
| B | 22.22% (56/252) 77.78% (196/252) 0 | Heuristic Solutions Avg. Range SD 1.11 1.05~1.23 0.05 |
|   | Optimal Solutions Same as above |                      |
Scalability

The computation times of obtaining the optimal solutions and the heuristic solutions are compared. Table 14 shows the computation times (in seconds) of obtaining the optimal and heuristic solutions for the four sets with the first set of parameters in the heuristics. As Table 14 illustrates, with the increasing number of service requests, the increasing number of servers (from two servers in Set 1 to twenty servers in Set 5), and the increasingly limited resource capacity from Case A to Case C, it takes the longer computation time in average and the larger range of computation times to get the optimal and heuristic solutions. However, the rate of computation time increase with the increasing problem size and complexity is much larger for getting the optimal solutions than getting the heuristic solutions. Hence, using the heuristics is much more scalable than solving the MIP optimization problem. Table 15 shows computation times (in seconds) of obtaining the optimal and heuristic solutions for the four sets of problem cases with the second set of parameters. The computation time results using the second set of parameters are similar to the ones with the first set of parameters. Hence, using the heuristics is much more scalable than solving the MIP optimization problem.

|                | Set 1     | Set 3     | Set 4     | Set 5     |
|----------------|-----------|-----------|-----------|-----------|
| **Optimal**    | Avg.      | Range     | SD        | Avg.      | Range     | SD        | Avg.      | Range     | SD        | Avg.      | Range     | SD        |
| Case A         | 0.12      | 0.02 - 0.6| 0.11      | 0.06      | 0.02 - 0.3| 0.05      | 0.07      | 0.03 - 0.39| 0.05      | 0.1       | 0.03 - 0.48| 0.08      |
| Case B         | 0.2       | 0.03 - 0.82| 0.13      | 0.09      | 0.03 - 0.37| 0.06      | 0.87      | 0.05 - 4.19| 0.85      | 3.43      | 0.00 - 17.68| 3.37      |
| Case C         | 0.41      | 0.03 - 3.31| 0.45      | 1.5       | 0.00 - 64.88| 6.14      | 32.83     | 0.07 - 138.54| 50.33     | 59.2      | 0.23 - 170.6 | 65.05     |
| **Heuristic**  | Avg.      | Range     | SD        | Avg.      | Range     | SD        | Avg.      | Range     | SD        | Avg.      | Range     | SD        |
| Case A         | 0         | 0.00 - 0  | 0         | 0         | 0.00 - 0  | 0         | 0         | 0.00 - 0  | 0         | 0         | 0.00 - 0  | 0         |
| Case B         | 0         | 0.00 - 0.08| 0.01      | 0         | 0.00 - 0.11| 0.01      | 0.01      | 0.00 - 0.56| 0.05      | 0.03      | 0.00 - 1.77| 0.16      |
| Case C         | 0         | 0.00 - 0.01| 0         | 0         | 0.00 - 0  | 0         | 0         | 0.00 - 0.02| 0         | 0.01      | 0.00 - 0.07| 0.01      |

Table 14: Computation times (in sec) of obtaining the optimal and heuristic solutions with the first set of parameters.
Table 15: Computation times (in sec) of obtaining the optimal and heuristic solutions with the second set of parameters.

|               | Set 1          | Set 3          | Set 4          | Set 5          |
|---------------|---------------|---------------|---------------|---------------|
|               | Avg.          | Range         | SD            | Avg.          | Range         | SD            | Avg.          | Range         | SD            |
| Optimal       |               |               |               |               |               |               |               |               |               |
| Solutions     |               |               |               |               |               |               |               |               |               |
| Case A        | 0.09          | 0.02 - 0.42   | 0.08          | 0.04          | 0.02 - 0.2    | 0.03          | 0.06          | 0.02 - 0.48   | 0.04          |
| Case B        | 0.16          | 0.03 - 0.69   | 0.12          | 0.08          | 0.02 - 0.36   | 0.05          | 0.74          | 0.05 - 4.56   | 0.76          |
| Case C        | 0.85          | 0.03 - 85.33  | 6.49          | 2.58          | 0.03 - 118.72 | 13.05         | 27.23         | 0.06 - 127.22 | 42.86         |
| Heuristic     |               |               |               |               |               |               |               |               |               |
| Solutions     |               |               |               |               |               |               |               |               |               |
| Case A        | 0             | 0.00 - 0      | 0             | 0             | 0.00 - 0      | 0             | 0             | 0.00 - 0      | 0             |
| Case B        | 0             | 0.00 - 0.17   | 0.01          | 0             | 0.00 - 0.14   | 0.01          | 0             | 0.00 - 0.22   | 0.02          |
| Case C        | 0             | 0.00 - 0      | 0             | 0             | 0.00 - 0      | 0             | 0             | 0.00 - 0.02   | 0             |

Conclusions

In this paper, we present the heuristics developed to capture the resource-service allocation decisions made in the optimal solutions for various problem cases. We then compare the heuristic solutions with optimal solutions on extensive sets of problem cases in regard to their solution optimality and scalability. The heuristic solutions perform comparable to the optimal solutions in terms of the solution optimality. In Case A problem cases with the sufficient resource capacity at each server and Case B problem cases with the sufficient resource capacity from all the servers together for all service requests, there are no dropped service requests in both the optimal solutions and the heuristic solutions, and all the heuristic solutions have the same average, range and standard deviations of the SAV ratios of the optimal solutions and thus the same QoS performance. For Case C problem cases with the insufficient resource capacity of all the servers together for all the service requests, the heuristic solutions drop a few more service requests (i.e., 1% ~ 4% of all the service requests) than the optimal solutions because the heuristics process service requests sequentially and thus make temporally local decisions rather than making global decisions by considering all service requests at one time in the optimal solutions. Nevertheless, the heuristic solutions provide the comparable QoS performance for all the served service requests to that of the optimal solutions with similar average, range and standard deviation of SAV ratios.

The computation time of obtaining both an optimal solution and a heuristic solution increases as the problem size (the number of service requests and the number of servers) and the problem complexity (from Case A to Case C) increase. However, the rate of computation time increase with the increasing problem size and complexity is much larger for obtaining the optimal solutions than obtaining the heuristic solutions, especially for Case B and Case C with a larger solution space due to the limited and insufficient resource capacity on each server or all the servers together for all the service requests to search for the optimal solutions. Hence, using the heuristics is much more scalable in real time than solving the MIP optimization problem.
Acknowledgement

This work is sponsored by the National Science Foundation (NSF) under grant number CCF0725340. The U.S. government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of NSF or the U.S. Government.

References

1. Endo PT, Palhares AV, Pereira NN, et al. 2011. Resource Allocation for Distributed Cloud: Concepts and Research Challenges. IEEE Network. 25: 42-46. Ref.: https://bit.ly/2vr4t4N
2. Foster I, Zhao Y, Raiciu I, et al. 2008. Cloud Computing and Grid Computing 360-Degree Compared. Grid Computing Environments Workshop. 1-10. Ref.: https://bit.ly/2PsCU4x
3. Marinescu DC. 2013. Cloud Computing: Theory and Practice. 1 edition: Morgan Kaufman.
4. Shyamala K, Rani TS. 2015. An Analysis on Efficient Resource Allocation Mechanisms in Cloud Computing. Indian Journal of Science and Technology. 8: 814-821.
5. Selvi ST, Valliyammal C, Dhatchayani VN. 2014. Resource Allocation Issues and Challenges in Cloud Computing. 2014 International Conference on Recent Trends in Information Technology. 1-6. Ref.: https://bit.ly/2V0LEFa
6. Ye N, Yang SS, Aranda BM. 2013. The Analysis of Service Provider-User Coordination for Resource Allocation in Cloud Computing. Information KnowledgeSystems Management. 12: 1-24. Ref.: https://bit.ly/2IWLh7k
7. Messina F, Pappalardo G, Santoro C. 2012. Decentralised Resource Finding in Cloud/Grid Computing Environments: A Performance Evaluation. 2012 IEEE 21st International WETICE. 143-148. Ref.: https://bit.ly/2DwrKXu
8. Messina F, Pappalardo G, Santoro C. 2014. Decentralised Resource Finding and Allocation in Cloud Federations. 2014 International Conference on Intelligent Networking and Collaborative Systems. 26-33. Ref.: https://bit.ly/2DBzfwC
9. Papagiannis C, Leivadeas A, Papavassiliou S, et al. 2013. On the Optimal Allocation of Virtual Resources in Cloud Computing Networks. IEEE TRANSACTIONS ON COMPUTERS. 62: 1060-1071. Ref.: https://bit.ly/2GCxvnd
10. Son S, Jung G, Jun SC. 2013. An SLA-based cloud computing that facilitates resource allocation in the distributed data centers of a cloud provider. J Supercomput. 606-637. Ref.: https://bit.ly/2IUy66I
11. Kadda BB, Benhammadi F, Sebbak, F, et al. 2015. New Tasks Scheduling Strategy for Resources Allocation in Cloud Computing Environment. 6th International Conference on Modeling, Simulation, and Applied Optimization (ICMSAO). 1-5. Ref.: https://bit.ly/2Dwt4cU
12. Li K, Wang Y, Liu M. 2014. A Task Allocation Scheme Based on Response Time Optimization in Cloud Computing. Distributed, Parallel, and Cluster Computing. 1-19. Ref.: https://bit.ly/2DvZw2
13. Srinivasa KG, Kumar KS, Kaushik US, et al. 2014. Game Theoretic Resource Allocation in Cloud Computing. 2014 Fifth International Conference on the Applications of Digital Information and WEb Technologies (ICADIWT). 36-42. Ref.: https://bit.ly/2vin8zS
14. Wang Z, Fang T. 2014. Task Scheduling Model Based on Multi-Agent and Multi-Objective Dynamical Scheduling Algorithm. Journal of Networks. 9: 1588-1595. Ref.: https://bit.ly/2ZqjIT4
15. Yang Z, Qin X, Li W, et al. 2013. Optimized Task Scheduling and Resource Allocation in Cloud Computing Using PSO based Fitness Function. Information Technology Journal. 12: 7090-7095.
16. Zhou J, Dutkiewicz E, Liu RP, et al. 2014. Modified Elite Chaotic Immune Clonal Selection Algorithm for Sever Resource Allocation in Cloud Computing Systems. 17th International Symposium on Wireless Personal Multimedia Communications (WPMC2014). 226-231. Ref.: https://bit.ly/2IFufLo

17. Zuo X, Zhang G, Tan W. 2014. Self-Adaptive Learning PSO-Based Deadline Constrained Task Scheduling for Hybrid IaaS Cloud. IEEE Transactions on Automation Science and Engineering. 11: 564-573. Ref.: https://bit.ly/2IEB2F9

18. Goudarzi H, Pedram M. 2011b. Multi-dimensional SLA-based Resource Allocation for Multi-tier Cloud Computing Systems. 2011 IEEE 4th International Conference on Cloud Computing. 324-331. Ref.: https://bit.ly/2GAT1ZC

19. Nesmachnow S, Iturriaga S, Dorronsoro B. 2015. Efficient Heuristics for Profit Optimization of Virtual Cloud Brokers. IEEE Computational Intelligence Magazine. 33-43. Ref.: https://bit.ly/2UBdVp9

20. Atiewi S, Yussof S, Ezanee M. 2015. A Comparative Analysis of Task Scheduling Algorithms of Virtual Machines in Cloud Environment. Journal of Computer Science. 804-812. Ref.: https://bit.ly/2V0QGcY

21. Goudarzi H, Pedram M. 2011a. Maximizing Profit in Cloud Computing System via Resource Allocation. 2011 31st International Conference on Distributed Computing Systems Workshops. 1-6. Ref.: https://bit.ly/2DzgKbV

22. Hsu CH, Chen TL, Park JH. 2008. On improving resource utilization and system throughput of master slave job scheduling in heterogeneous systems. The Journal of Supercomputing. 45: 129-150. Ref.: https://bit.ly/2DzgeCH

23. Liu Z, Zhou H, Fu S, et al. 2014. Algorithm Optimization of Resources Scheduling Based on Cloud Computing. Journal of Multimedia. 9: 977-984. Ref.: https://bit.ly/2Zvkbtz

24. Suresh A, Vijayakarthick P. 2011. Improving Scheduling of Backfill Algorithms using Balanced Spiral Method for Cloud Metascheduler. IEEE-International Conference on Recent Trends in Information Technology. ICRTIT. 624-627. Ref.: https://bit.ly/2XEbRW1

25. Varalakshmi P, Judgi T, Hafsa MF. 2013. Local Trust Based Resource Allocation in Cloud. 2013 Fifth International Conference on Advanced Computing (ICoAC). 591-596. Ref.: https://bit.ly/2W7sij

26. Wu X, Deng M, Zhang R, et al. 2013. A Task Scheduling Algorithm based on QoS-driven in Cloud Computing. 1st International Conference on Information Technology and Quantitative Management. 17: 1162-1169. Ref.: https://bit.ly/2L2mPUF

27. Sharma S, Tantawi A, Speitner M, et al. 2010. Decentralized Allocation of CPU Computation Power for Web Applications. Performance Evaluation. 67: 1187-1202. Ref.: https://bit.ly/2ZvkvbO

28. Wei Y, Blake BM. 2013. Decentralized Resource Coordination across Service Workflows in a Cloud Environment. 2013 Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises. 15-20. Ref.: https://bit.ly/2USELWe

29. Dhingra A, Paul S. 2014. Green Cloud: Heuristic based BFO Technique to Optimize Resource Allocation. Indian Journal of Science and Technology. 7: 685-691. Ref.: https://bit.ly/2Zy6fIP

30. Kumar K, Feng J, Nimmagadda Y, et al. 2011. Resource Allocation for Real-Time Tasks using Cloud Computing. 2011 Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN). 1-7. Ref.: https://bit.ly/2GFV67U

31. Kuribayashi Si. 2011. Optimal Joint Multiple Resource Allocation Method for Cloud Computing Environments. International Journal of Research and Reviews in Computer Science (IJRRCS). 2: 1-8. Ref.: https://bit.ly/2ZzBTfC
32. Rezvani M, Akbari MK, Javadi B. 2015. Resource Allocation in Cloud Computing Environments Based on Integer Linear Programming. Section B: Computer and Communications Networks and Systems, The Computer Journal. 58: 300-314. Ref.: https://bit.ly/2W9857q
33. Urgaonkar R, Kozat UC, Igarashi K, et al. 2010. Dynamic Resource Allocation and Power Management in Virtualized Data Centers. 2010 IEEE/IFIP Network Operations and Management Symposium-NOMS. 479-486. Ref.: https://bit.ly/2XBb2xQ
34. Yin B, Wang Y, Meng L, et al. 2012. A Multi-dimensional Resource Allocation Algorithm in Cloud Computing. Journal of Information & Computational Science. 9: 3021-3028. Ref.: https://bit.ly/2IHKHuL
35. Mehdi NA, Mamat A, Ibrahim H, et al. 2011. Impatient Task Mapping in Elastic Cloud using Genetic Algorithm. Journal of Computer Science. 7: 877-883. Ref.: https://bit.ly/2VgJuOg
36. Shi W, Hong B. 2010. Resource Allocation with a Budget Constraint for Computing Independent Tasks in the Cloud. 2nd IEEE International Conference on Cloud Computing Technology and Science. 327-334. Ref.: https://bit.ly/2vjsP0j
37. Laili Y, Tao F, Zhang L, et al. 2013. A Ranking Chaos Algorithm for Dual Scheduling of Cloud Service and Computing Resource in Private Cloud. Computers in Industry. 64: 448-463. Ref.: https://bit.ly/2UF0IDc
38. Mao Z, Shang Y, Liu C, et al. 2013. Utility-based Price Proportion in Cloud Resource Allocation. Information Technology Journal. 12: 6882-6886. Ref.: https://bit.ly/2DvZapu
39. Sindhu S, Mukherjee S. 2013. A Genetic Algorithm based Scheduler for Cloud Environment. 4th International Conference on Computer and Communication Technology (ICCCCT). 23-27. Ref.: https://bit.ly/2GxISFx
40. Wang Z, Su X. 2015. Dynamically hierarchical resource-allocation algorithm in cloud computing environment. The Journal of Supercomputing. 2748-2766. Ref.: https://bit.ly/2ZCfwX5
41. Yau SS, Ye N, Sarjoughian HS, et al. 2009. Toward Development of Adaptive Service-Based Software Systems. IEEE Transactions on Services Computing. 2: 247-260. Ref.: https://bit.ly/2IGncS9
42. Ye N, Yau S, Huang D, et al. 2010. Models of dynamic relations among service activities, system state and service quality on computer and network systems. Information, Knowledge, Systems Management. 9: 99-116. Ref.: https://bit.ly/2GFWkQI
43. Berman F. 1999. High-performance schedulers. In the Grid Blueprint for a New Computing Infrastructure, edited by Ian Foster and Carl Kesselman. San Francisco, CA: Morgan Kaufman Publishers.
44. Wei Y, Blake BM, Saleh I. 2013. Adaptive Resource Management for Service Workflows in Cloud Environments. 2013 IEEE 27th International Symposium on Parallel & Distributed Processing Workshops and PhD Forum. 2147-2156. Ref.: https://bit.ly/2IUDwP6