Fast Algorithms for Optimal Coalition Formation in Federated Clouds

by

Yash Khandelwal, Suresh Purini, Viswanath Puduru Puduru

in

International conference on Utility and cloud computing

Report No: IIIT/TR/2016/-1

Centre for Software Engineering Research Lab
International Institute of Information Technology
Hyderabad - 500 032, INDIA
December 2016

Fast Algorithms for Optimal Coalition Formation in Federated Clouds

Yash Khandelwal
International Institute of Information Technology
Hyderabad, India
yash.khandelwal@research.iiit.ac.in

Suresh Purini
International Institute of Information Technology
Hyderabad, India
suresh.purini@iiit.ac.in

Puduru V. Reddy
GERAD, HEC Montréal
Montréal, Canada
puduru.reddy@gerad.ca

ABSTRACT

In this paper, we formulate the optimal coalition formation in federated clouds as an integer linear programming problem under the cloud service brokerage model proposed by Mashayekhy et al [10]. Then we propose a fast polynomial time greedy algorithm to find a near optimal coalition. The profit generated by the federation obtained using the greedy algorithm is within a negligible 0.06 percent of the optimal on an average. The greedy algorithm finds a federation 200 times faster on an average when compared with the Merge-Split algorithm. The payoff distribution within a federation is determined using exact Banzhaf index computation whereas the Merge-Split algorithm arrives at a payoff using an estimate of Banzhaf values. By computing the payoff distribution after the federation formation, we are able to achieve 66x speedup when compared with the Merge-Split algorithm.

1. INTRODUCTION

Cloud computing through its various service models such as IaaS, PaaS and SaaS is able to substantially reduce the IT infrastructure costs. It enabled new modes of operation which are otherwise impossible. For example, in one of our research projects in large scale distributed model checking [18], we experimented with various Hadoop cluster configurations to understand scalability properties with respect to horizontal and vertical scaling of resources. We established different Hadoop cluster configurations using Amazon Web Service (AWS) by renting resources just for the duration of the experiment. This would have not been financially feasible if not for the ability to acquire and release resources on a demand basis. However, our resource requirements are meagre and do not require a big cloud service provider such as Amazon.

In future, we envisage a distributed collection of small cloud service providers [19], who can serve consumer resource requests either individually or form a *federation* to serve requests which are beyond their individual capacity. Such a market with several service providers enhances competition and avoids monopoly. There are other benefits such as reducing the stress on resources such as power and the associated complex cooling requirements mandated by large data centers [7]. Further, organizations and individuals can contribute part of their idle resources and act as micro cloud service providers.

Among the multiple challenges involved in the evolution of such a market place with many small cloud service providers, we identify two main issues relevant to the work in this paper: 1) How do individual service providers coordinate with each other to serve requests beyond their capacity? 2) What is the interface between the service providers and an end consumer? In this paper, we use the model proposed by Mashayekhy et al. [10], wherein a *broker* acts as a point of contact or interface between all the service providers and consumers (refer Figure 1). We restrict ourselves to this brokerage model which is presented in Section 2, although alternate models in which the cloud service providers can negotiate between themselves to serve a consumer request are possible. The following are the main contributions of this paper:

1. We formulate the optimal federation formation as an integer linear programming problem and then propose a fast polynomial time greedy algorithm for federation formation. The profit generated by a federation obtained using the greedy algorithm is within a negligible 0.06 percent of the optimal on an average. The greedy algorithm finds a federation 200 times faster on an average when compared with the Merge-Split algorithm [10].

2. We perform the payoff distribution computation using exact Banzhaf indices of indiviudal cloud service providers after the federation is determined. This reduces the number of subcoalitions to be considered from $2^{|\mathcal{I}|}$ to $2^{|F|}$ where $\mathcal{I}$ is the set of all cloud service providers and $F \subseteq \mathcal{I}$ is the newly formed federation. When compared with the Merge-Split algorithm which uses approximate Banzhaf indices for determining the payoff, our approach runs 66 times faster.

The layout of the paper is as follows. Section 2 describes the cloud brokerage model we use in this paper and provides the necessary game theory background; Section 3 presents the integer linear programming formulation for the federation formation problem and a polynomial time greedy algo-

UCC '16, December 06-09, 2016, Shanghai, China

DOI: http://dx.doi.org/10.1145/2996890.2996900

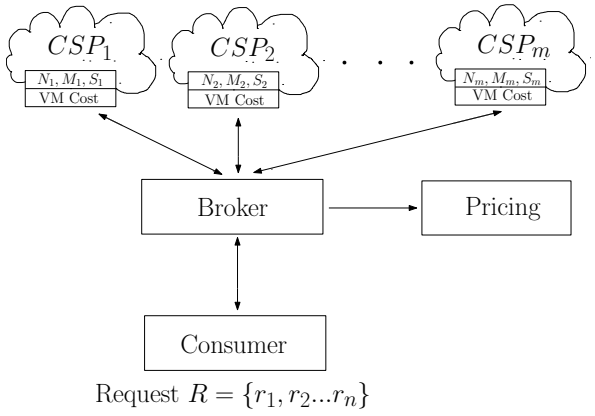


Figure 1: Cloud federation model.

rithm for the same; Section 4 discusses issues pertinent to the optimality and stability of the Merge-Split algorithm; Section 5 presents the payoff distribution scheme; Section 6 presents experimental results; Section 7 contains the related work and finally we conclude in Section 8.

2. BACKGROUND

Our work builds on the cloud federation framework model proposed by Mashayekhy et al. [10] which we provide in this section. Let $\mathcal{I} = \{C_1, \cdots, C_m\}$ be a collection of cloud service providers (CSPs). The capacity of a CSP C_i is given by a 3-tuple (N_i, M_i, S_i) where N_i is the total number of available compute cores, M_i and S_i indicate the total available RAM and secondary storage respectively. Each cloud service provider offers n types of virtual machines. A virtual machine VM_j, $1 \leq j \leq n$, is characterized by a resource vector (w_j^c, w_j^m, w_j^s) where w_j^c, w_j^m and w_j^s denote the cores, memory and storage respectively.

All the CSPs provide their services to consumers through a *broker* (refer Figure 1). The broker offers a type VM_j virtual machine to the consumer at price p_j for $1 \leq j \leq n$. A CSP C_i offers a type VM_j virtual machine to the broker at cost c_{ij}. A consumer specifies his request as an n-tuple $(r_1, \cdots, r_n)$ where r_i indicates the required number of virtual machine instances of type VM_i. When a federation of cloud service providers $F \subseteq \mathcal{I}$ satisfy a consumer request, the value they generate is equal to

$$\sum_{C_i \in F} \sum_{j=1}^{n} x_{ij}(p_j - c_{ij})$$

where x_{ij} is the number of virtual machines of type VM_j served by the service provider C_i. Under our brokerage model, the cloud service providers who are not part of the federation neither generate any value nor influence the value of the federation.

For a given cloud federation $F \subseteq \mathcal{I}$, its value can be maximized by solving the following integer linear program denoted as IP-CFPM (Cloud Federation Profit Maximization) [10].

$$\text{Maximize} \sum_{C_i \in F} \sum_{j=1}^{n} x_{ij}(p_j - c_{ij}) \quad (1)$$

$$\sum_{j=1}^{n} w_j^c x_{ij} \leq N_i, \ (\forall C_i \in F) \quad (2)$$

$$\sum_{j=1}^{n} w_j^m x_{ij} \leq M_i, \ (\forall C_i \in F) \quad (3)$$

$$\sum_{j=1}^{n} w_j^s x_{ij} \leq S_i, \ (\forall C_i \in F) \quad (4)$$

$$\sum_{C_i \in F} x_{ij} = r_j, \ (1 \leq j \leq n) \quad (5)$$

$$\sum_{j=1}^{n} x_{ij} \geq 1, \ (\forall C_i \in F) \quad (6)$$

$x_{ij} \geq 0$ ($\forall C_i \in F$ and $1 \leq j \leq n$)

All x_{ij} are integers.

The constraints (2), (3) and (4) are capacity constraints. The constraint (5) indicates that the consumer request has to be satisfied completely. The participation constraint (6) in IP-CFPM conveys that the federation $F \subseteq \mathcal{I}$ is viable only if every cloud service provider in the federation makes a contribution.

In order to understand the factors determining the optimality and stability of a cloud federation, and the payoff distribution scheme, we use the coalitional game theory framework [5, 14].

2.1 Coalitional Game Theory

We use characteristic function games to analyze cloud federation formation mechanisms in the Mashayekhy's cloud brokerage model. We consider games in which the profit of a federation can be divided in any way between the federation members, the so called *transferable utility games*.

DEFINITION 2.1. *A coalitional game in characteristic function form with transferable utility is denoted by $G = (N, v)$. Here, $N = \{1, \cdots, n\}$ represents the players and $v(.)$ is the characteristic function defined as $v : \mathcal{P}(N) \rightarrow \mathbb{R}^+$, where $\mathcal{P}(N)$ is the powerset of N.*

For a coalition $F \subseteq N$, we call $v(F)$ as its *value* and represents the profit obtained through cooperation by the players in F. It is usually the case that $v(\phi) = 0$. The value generated by the *grand coalition* N is denoted by $v(N)$. We call $v(.)$ to be super-additive if the following condition is satisfied, $v(P) + v(Q) \leq v(P \cup Q)$ for any $P, Q \in \mathcal{P}(N)$ and $P \cap Q = \emptyset$. Clearly, in a coalitional game with super-additive characteristic function forming larger coalitions is beneficial for the players.

An outcome of a coalitional game is the payoff vector which distributes the profit of the grand coalition $v(N)$ among the players. We have the following definitions.

DEFINITION 2.2. *A payoff vector $x = (x_1, \cdots, x_n) \in \mathbb{R}^n$ is said to be an imputation if it satisfies the following efficiency and individual rationality conditions.*

1. $\sum_{i \in N} x_i = v(N)$.

2. $x_i \geq v(\{i\}), \forall i \in N$

Let us denote by $x(F)$ the payoff of the coalition/federation $F \subseteq N$, which is defined as $\sum_{i \in F} x_i$. We have the following definition.

DEFINITION 2.3. *The core of a characteristic function game G consists of all payoff vectors such that $x(F) \geq v(F)$ for all $F \subset N$ and $\sum_{i \in N} x_i = v(N)$.*

Clearly, a payoff vector in a non-empty core guarantees that no federation has an incentive to break away from the grand federation, thus ensuring *stability*. We note that core is a set-valued solution concept. There exist single valued solution concepts, based on axioms related to *fairness*, such as *Shapley value* and *Banzhaf index*, see [14] for more details. In simple words, in these two solution concepts the payoff to a player depends upon the marginal contribution of the player to the federation he belongs to. In the context of cloud federations, the notion of fairness as captured by Banzhaf index is more appropriate, see [10], when compared with Shapley value.

DEFINITION 2.4. *Given a characteristic function game $G = (N, v)$, the Banzhaf index of a player $i \in N$ denoted as $\beta_i(G)$ is defined as*

$$\beta_i(G) = \frac{1}{2^{n-1}} \sum_{F \subseteq N - \{i\}} [v(F \cup \{i\}) - v(F)].$$

The Banzhaf index of a player gives his marginal contribution averaged across all coalitions in which he is present. The Banzhaf indices of players are independent of the coalition structure (refer Definition 2.6) and it can be the case that for a coalition $F \in CS$, $\sum_{i \in F} \beta_i(G) \neq v(F)$. This means that the value distribution among the players in coalition need not be efficient. If the coalition structure corresponds to a grand coalition of all players, then we can use *normalized Banzhaf index* to maintain efficiency.

DEFINITION 2.5. *The normalized Banzhaf index of player $i \in N$ in a characteristic function game $G = (N, v)$ is defined as*

$$\eta_i(G) = \frac{\beta_i(G)}{\sum_{i \in N} \beta_i(G)}.$$

We notice that the payoff vector based on normalized Banzhaf index may not belong to the core. The outcome of a cooperative game, in characteristic function form, is mainly concerned with the allocating the worth of the grand coalition $v(N)$, and with the formation of grand coalition. When the characteristic function is not super-additive then grand coalition is not feasible. The possible scenario would be formation of collection of federations, that is, disjoint sets of the player set, also called as a *coalition structure* or federation structure. The central questions then would be related to the formation of a coalition structure and the associated stability issues. To study these aspects, we briefly discuss a class of games [9] in the following subsection.

2.2 Hedonic Games and Merge-Split Algorithm

For analyzing cloud federation formation mechanisms, a special class of coalitional games called *Hedonic games* are appropriate, see [9].

DEFINITION 2.6. *A coalition structure, denoted by CS, is a partition $\{F_1, \cdots, F_k\}$ of the player set N and each F_i, $1 \leq i \leq k$, is called a coalition.*

Let $\mathcal{N}_i = \{F \in 2^N | i \in F\}$ denote all the coalitions in which player i is present. Let $\succeq_i$ denote a total, reflexive and transitive relation over $\mathcal{N}_i$. If $F_1 \succeq_i F_2$ ($F_1 \succ_i F_2$) for $F_1, F_2 \in \mathcal{N}_i$, then it means that the player i prefers (strictly) the coalition F_1 over F_2.

DEFINITION 2.7. *A hedonic game denoted as $G = (N, \succeq_1, \cdots, \succeq_n)$ consists of a set of $N = \{1, \cdots, n\}$ players and their preference relations.*

The outcome of a hedonic game is a coalition structure. Given a coalition structure CS, let $F_i \in CS$ denote the coalition to which the player i belongs to.

DEFINITION 2.8. *A coalition $C \subseteq N$ blocks a coalition structure CS if $C \succ_i F_i$ for all $i \in C$. Such a coalition is called a blocking coalition.*

All the players in the blocking coalition C have an incentive to part from their existing coalitions and form a new coalition between them.

DEFINITION 2.9. *The core of a hedonic game consists of all coalition structures which are not blocked by any coalition.*

DEFINITION 2.10. *A hedonic game is core stable if it has a non-empty core.*

DEFINITION 2.11. *A federation $F \subseteq N$ is individually federation stable if there exists no $i \in F$ such that $F - \{i\} \succeq_j F$ for all $j \in F$.*

Note that this notion of stability defined over an individual federation differs from the notion of individual stability defined over coalition structures conventionally [5].

Mashayekhy et al. [10] modeled the cloud federation formation mechanism as a hedonic game wherein the preference relation for a player i is defined as

$$\forall F, F' \in \mathcal{N}_i \quad F \succeq_i F' \iff v(F) \geq v(F').$$

The value of a coalition $F \subseteq N$ is defined to be equal to the value obtained by solving the integer linear programming problem IP-CFPM.

2.2.1 Merge-Split Algorithm

Given a hedonic game consisting of cloud service providers as players, Mashayekhy et al. [10] proposed a fixed-point iterative Merge-Split algorithm to find an individually stable federation. The algorithm starts with an initial coalition structure $CS_0 = \{\{1\}, \cdots, \{n\}\}$ in which each CSP forms a federation by itself. In each iteration of the algorithm, the current coalition structure CS_i passes through a *merge phase* and a *split phase* to reach a new coalition configuration CS_{i+1}. The algorithm terminates when the coalition structure reaches a fixed point, i.e., $CS_k = CS_{k+1}$ for some $k > 0$. During a merge phase, two federations $F, F' \in CS_i$ are merged if and only if $F \cup F' \succ_i F_i$ for all $i \in F \cup F'$. Note that F_i denotes F or F' depending on whether the player i is in F or F'. During a split phase, a federation F splits into F' and F'' if and only if there exists a player $i \in F$ such that $F' \succeq_i F$ assuming $i \in F'$ with out loss of generality. Note that if $F' \succeq_i F$, then $F' \succeq_j F$ for all $j \in F'$ due to the

way in which the preference relation is defined. A federation can split without any improvement in the value but a merge happens only if the value of the newly formed federation strictly increases. Using this observation, we can argue that the fixed-point algorithm converges in finite number of iterations. Then federation $F_{ms} \in CS_k$ whose value is maximum is chosen to serve the consumer request. The federation F_{ms} is individually stable otherwise a split operation would have happened and the iterative algorithm would have proceed further forming a new coalition structure CS_{k+1}.

In the next section, we analyze the time complexity of the Merge-Split algorithm and propose substantially faster algorithms to arrive at optimal cloud federations which are individually stable.

3. ALGORITHMS FOR FEDERATION FORMATION

Let $\mathcal{CS}_N$ denote the space of all coalition structures over N players. The Merge-Split algorithm is an iterative fixed point algorithm to find a stable coalition from $\mathcal{CS}_N$ and hence in the worst case may take $|\mathcal{CS}_N|$ iterations to converge. The size of $\mathcal{CS}_N$ is equal to the number of ways in which a set of size N can be partitioned and is equal to the Bell number B_N which grows super-exponentially. Further, in each iteration of the Merge-Split algorithm, we have to solve a integer linear programming problem, although we can avoid solving previously encountered problem instances using a memoization table. These factors make it impractical to use Merge-Split algorithm even for a moderately large N.

3.1 Optimal Cloud Federation Formation Mechanism

In the federation formation approach proposed by Mashayekhy et al. [10], after a stable coalition structure $CS = \{F_1, \cdots, F_k\}$ is found using the Merge-Split algorithm, the coalition whose value is maximum, $F_{ms} = argmax_{F_i \in CS}\{v(F_i)\}$, is chosen as the winning federation and the rest of the coalitions remain neutral, with zero payoff, without affecting the value of the winning federation in any way. Based on this observation, we formulate the problem of finding an optimal federation as an integer linear programming problem OPT-CFFM (Cloud Federation Formation Mechanism) wherein we relax the participation constraint (6) from IP-CFPM and consider the federation F to be the grand coalition $\mathcal{I}$.

$$\text{Maximize} \sum_{C_i \in \mathcal{I}} \sum_{j=1}^{n} x_{ij}(p_j - c_{ij}) \tag{7}$$

$$\sum_{j=1}^{n} w_j^c x_{ij} \leq N_i, \ (\forall C_i \in \mathcal{I}) \tag{8}$$

$$\sum_{j=1}^{n} w_j^m x_{ij} \leq M_i, \ (\forall C_i \in \mathcal{I}) \tag{9}$$

$$\sum_{j=1}^{n} w_j^s x_{ij} \leq S_i, \ (\forall C_i \in \mathcal{I}) \tag{10}$$

$$\sum_{C_i \in \mathcal{I}} x_{ij} = r_j, \ (1 \leq j \leq n) \tag{11}$$

$x_{ij} \geq 0$ ($\forall C_i \in \mathcal{I}$ and $1 \leq j \leq n$)

All x_{ij} are integers.

Input: Profit set P, resource capacity matrix R, consumer request $r = (r_1, \cdots, r_n)$.
$F = \{\}$;
while *true* **do**
  `// max operator finds the element which generates maximum profit from the set P.`
  `// max operator can be implemented using any priority queue [3]`
  $(p, i, j) = max(P)$;
  if $r_j = 0$ **then** continue;
  α = maximum number of VM_j instances that can be composed using R_i ;
  if $\alpha \leq r_j$ **then**
    (n, m, s) = Resources required to compose α VM_j instances ;
    $P = P - \{(p, i, j)\}$;
    $r = (r_1, \cdots, r_{j-1}, r_j - \alpha, \cdots, r_n)$;
    $R_i = R_i - (n, m, s)$;
    if $\alpha > 0$ **then** $F = F \cup \{i\}$;
  else
    (n, m, s) = Resources required to compose r_j VM_j instances ;
    $P = P - \{(p, i, j)\}$;
    $r = (r_1, \cdots, r_{j-1}, 0, \cdots, r_n)$;
    $R_i = R_i - (n, m, s)$;
    $F = F \cup \{i\}$;
  end
  if $r = (0, \cdots, 0)$ **or** $P = \emptyset$ **then** break;
end

Algorithm 1: GCFFM: Greedy algorithm for cloud federation formation.

Once we solve the above OPT-CFFM problem, the resulting cloud federation consists of those CSPs who contribute towards serving the consumer request and is defined as follows

$$F_{opt} = \{C_i \in \mathcal{I} \mid \exists x_{ij} > 0\}.$$

Note that in the Merge-Split algorithm, the search space is the set of all partitions of $\mathcal{I}$ whereas in the OPT-CFFM problem it is the power set of $\mathcal{I}$. Solving a single integer linear programming problem (OPT-CFFM) is clearly practical when compared with the Merge-Split algorithm which takes super-exponential time in the worst case. Even if we do not expect encountering such a worst case in practice, we show empirically (refer Section 6) that we gain multiple orders of magnitude speedup using the above proposed approach. Further, we show in Section 4 that the coalition F_{ms} may be individually stable but its value can be sub-optimal.

3.2 Greedy Cloud Federation Formation Mechanism

The optmization problem OPT-CFFM is amenable to a greedy approach. We propose a polynomial time greedy algorithm which finds a near optimal cloud federation. Although, we do not have any theoretical approximation guarantees, in the experiments that we conducted the value of the cloud federation computed by the greedy is within a negligible 0.06% of the optimal on an average (refer Section 6). Further, the greedy algorithm achieves 200x speedup when compared with the Merge-Split algorithm on an average.

Let $P_{m \times n}$ be the profit matrix whose rows and columns are labeled with CSPs and virtual machine instance types

respectively. The entry p_{ij} of the profit matrix is equal to $p_j - c_{ij}$ and denotes the profit generated if a VM_j instance is served by the CSP C_i. Abusing the notation, from now on we consider P as denoting the set $\{(p_{ij}, i, j) \mid 1 \leq i \leq m \text{ and } 1 \leq j \leq n\}$. Let $R_{m\times 3}$ be the resource capacity matrix such that the row R_i (i^{th} row) is equal to the resource capacity (N_i, M_i, S_i) of the CSP C_i. Recall that a consumer request is denoted by the vector $r = (r_1, \cdots, r_n)$.

The greedy algorithm assigns resources to the consumer request iteratively. The remnant resource request at the end of k^{th} iteration is denoted by r^k. Similarly, R^k denotes the available resources with the CSPs at the end of k^{th} iteration. And let $r^0 = r$ and $R^0 = R$. The set P also changes with every iteration and P^k denotes the profit set at the end of k^{th} iteration with $P^0 = P$. We define a max operator on the set P^k as $max(P^k) = (p, i, j)$ where $p \geq p_{ij}$ for all $(p_{ij}, i, j) \in P^k$.

In the first iteration, we compute $max(P^0)$ and let it be (p, i, j). Let α be the maximum number of VM_j instances that can be composed using the resources (n, m, s) from the available resources vector R_i^0 of the CSP C_i. If $\alpha \leq r_j$, then $P^1 = P^0 - \{(p, i, j)\}$, $r^1 = (r_1, \cdots, r_{j-1}, r_j - \alpha, \cdots, r_n)$ and $R_i^1 = R_i^0 - (n, m, s)$. If $\alpha > r_j$, then let (n, m, s) be the resources required to compose r_j instances of type VM_j. Then $P^1 = P^0 - \{(p, i, j)\}$, $r^1 = (r_1, \cdots, r_{j-1}, 0, \cdots, r_n)$ and $R_i^1 = R_i^0 - (n, m, s)$. We continue this iterative procedure until either the resource assignment for the consumer request is done, $r^k = (0, \cdots, 0)$, or the set P^k is empty and $r^k \neq (0, \cdots, 0)$. In the later case, it is not possible to completely satisfy the consumer request with the available resources from the CSPs. Finally, the cloud federation consists of those CSPs which are chosen to serve VM instances in at least one of the iterations of the greedy algorithm. Algorithm 1 summarizes the above discussion. The size of the set P^k decreases by one in every iteration and hence the algorithm converges in at most $|P| = mn$ iterations. In each iteration, the max operation is the most computationally expensive and takes $O(\log |P^k|)$ time. Overall, the time complexity of the greedy algorithm is $O(mn \log mn)$.

4. OPTIMALITY VERSUS STABILITY

In this section, we compare the Merge-Split algorithm and the OPT-CFFM algorithm with respect to the optimality and stability criteria. Note that we have defined two notions of stability in Section 2: core stability (refer Definition 2.9) and individual federation stability (refer Defintion 2.11).

4.1 Merge-Split Algorithm

The Merge-Split algorithm arrives at a coalition structure $CS = \{F_1, \cdots, F_k\}$ and then assigns the consumer request to the federation with the maximal value. Let $F_{ms} = argmax_{F_i \in CS} \{v(F_i)\}$ and F_{opt} be the federation obtained from OPT-CFFM. Then it is easy to observe that $v(F_{ms}) \leq v(F_{opt})$. The federation thus arrived is individually stable but its value could be sub-optimal as illustrated by the following example.

We consider 3 CSPs $\mathcal{I} = \{C_1, C_2, C_3\}$. Let R be a consumer request such that:

- $v(\{C_1\}) = v(\{C_2\}) = v(\{C_3\}) = 0$
- $v(\{C_1, C_2\}) = 12$, $v(\{C_2, C_3\}) = 13$, $v(\{C_1, C_3\}) = 14$
- $v(\{C_1, C_2, C_3\}) = 11$

Input: F_{opt}, resource capacity matrix R, consumer request $r = (r_1, \cdots, r_n)$.
// Inputs R and r are required to compute the value of a federation.
// Value $v(F)$ of a federation F is obtained by solving IP-CFPM.
$F^*_{opt} = F_{opt}$;
$change = true$;
while *change==true* **do**
  $change = false$;
  foreach $C_i \in F_{opt}$ **do**
    if $v(F^*_{opt}) == v(F^*_{opt} - \{C_i\})$ **then**
    $F^*_{opt} = F^*_{opt} - \{C_i\}$;
    $change = true$;
  end
end

Algorithm 2: Fixed point algorithm to compute an individually stable federation F^*_{opt} from F_{opt}.

Clearly, none of the individual CSPs can satisfy the consumer request R by themselves. The OPT-CFFM algorithm selects the federation $\{C_1, C_3\}$ as it generates the maximum payoff. The outcome of the Merge-Split algorithm depends on the initial federation structure and the choices it makes during the merge and split phases of the iterations. Let the initial coalition structure be $\{\{C_1\}, \{C_2\}, \{C_3\}\}$. During the merge phase a coalition structure $\{\{C_1, C_2\}, \{C_3\}\}$ can be formed since $v(\{C_1, C_2\}) > v(C_1)$ and $v(\{C_1, C_2\}) > v(C_2)$. The coalition structure does not change in the following split phase. In the next iteration, the federations $\{C_1, C_2\}$ and $\{C_3\}$ would not merge, since $v(\{C_1, C_2\} > v(\{C_1, C_2, C_3\})$. The algorithm converges in this iteration and finally the sub-optimal federation $\{C_1, C_2\}$ is chosen as it has the maximum value in the coalition structure. If the Merge-Split algorithm has considered the federations C_1 and C_3 for merging in the first iteration, it would have found a federation which is not only stable but also optimum. An interesting case to consider is when the initial coalition structure is of the form $\{\{C_1, C_2, C_3\}\}$. We will leave this as an exercise to the reader.

Another important observation is that the final coalition structure from which a federation is chosen to serve the consumer request need not be core stable. For example the coalition structure $\{\{C_1, C_2\}, \{C_3\}\}$ is not core stable since $\{C_1, C_3\}$ forms a blocking coalition. In fact, if the coalition structure at the end of the Merge-Split algorithm is core stable, then the federation chosen from it, F_{ms}, is not only individually stable but also has an optimal value. Finally, constructing a problem instance such that the resulting characteristic function has the desired properties of the aforementioned example needs to be explored. We conjecture that $v(F_{ms}) \leq c \cdot v(F_{opt})$ for some constant c which is dependent on the price vector.

4.2 OPT-CFFM

The federation F_{opt} obtained using the OPT-CFFM approach may not be individually stable as it could be possible to drop a CSP from F_{opt} while retaining its value. Note that the value of the federation F_{opt} cannot go up any further as it has the optimal value. Based on this observation, we can apply a simple fixed point iterative algorithm (refer Algorithm 2) to find an individually stable federation F^*_{opt} from

F_{opt}. In our experiments, it is always the case that F_{opt} obtained from IP-CFPM is also individually stable. The idea of core stability is not applicable in OPT-CFFM approach as it never considers coalitional structures unlike the Merge-Split algorithm.

5. PAYOFF STRUCTURE

The payoff distribution schemes such as Shapley value and Banzhaf index makes sense when the federation formed is a grand coalition of all the CSPs. In our context, once we arrive at a federation, the rest of the CSPs have no part to play and cannot influence the internal payoff distribution within the formed federation. Hence, we can construct a new characteristic function game $G^* = (F^*_{opt}, v)$ where the value of a federation $F \subseteq F^*_{opt}$ is obtained by solving the integer linear program IP-CFPM. We can use any payoff distribution scheme for the game G^*. Since, the order in which the CSPs join a federation does not matter Banzhaf index is a more reasonable payoff distribution scheme. Using the Banzhaf index scheme, the payoff for each CSP within the federation F^*_{opt} is defined as follows

$$\beta_i(G^*) = \frac{1}{2^{|F^*_{opt}|-1}} \sum_{F \subseteq F^*_{opt} - \{i\}} [v(F \cup \{i\}) - v(F)].$$

Since Banzhaf index is not efficient, i.e., $\sum_{i \in F^*_{opt}} \beta_i(G^*) \neq v(F^*_{opt})$, we use normalized Banzhaf index value which is defined as follows for payoff.

$$\eta_i(G*) = \frac{\beta_i(G)}{\sum_{i \in F^*_{opt}} \beta_i(G)} \times v(F^*_{opt})$$

Mashayekhy et al. [10] proposed a payoff scheme which is obtained by approximating the Banzhaf value of CSPs by considering the grand coalition $\mathcal{I}$. The computation of the approximate Banzhaf value is obtained through the Merge-Split algorithm and hence has the same limitations as that of the Merge-Split. In our approach, we compute the exact Banzhaf value by computing the value of every federation $F \subseteq F^*_{opt}$. Hence we consider only $2^{|F^*_{opt}|}$ distinct sub-coalitions instead of $2^{|\mathcal{I}|}$ sub-coalitions, thereby reducing the time complexity. The size of the federation F^*_{opt} is usually very small when compared with the total number of cloud service providers $|\mathcal{I}|$. Inspite of this, the time taken for payoff computation is substantially larger when compared with federation formation. This can be easily seen as we need to solve only one integer linear program to arrive at F_{opt} and $|F_{opt}|$ integer linear programs (in most cases) to arrive at F^*_{opt}. Then we need to solve $2^{|F^*_{opt}|}$ integer linear programs for payoff computation[1]. We leave the problem of finding time efficient payoff distribution schemes with similar properties to that of Banzhaf index to future work.

6. EXPERIMENTAL RESULTS

For experimental analysis of the proposed algorithms, we considered 10 different Amazon EC2 regions as 10 distinct CSPs ($|\mathcal{I}| = 10$). Each CSP provides four VM instance types: m3.medium, m3.large, m3.xlarge and m3.2xlarge.

[1]In all our experiments, we found that F_{opt} and F^*_{opt} are the same. Hence, if we eliminate the F^*_{opt} computation phase, payoff computation takes roughly $2^{|F_{opt}|}$x times more.

	medium VM1	large VM2	xlarge VM3	2xlarge VM4
w^c_j (vCPU)	1	2	4	8
w^m_j (GB)	3.75	7.5	15	30
w^s_j (GB)	4	32	80	160
p_j (price)	0.1176$	0.2450$	0.5096$	1.0584$

Table 1: VM instance types and their resource configuration.

The resource requirements for each of these VM instance types is given in the Table 1. Table 2 shows the cost at which various CSPs offer these instance types. Let $c^{max}_j = max_{1 \leq i \leq 10}\{c_{ij}\}$ for $1 \leq j \leq 4$ be the maximum price at which a VM_j instance is offered. For example, $c^{max}_1 = 0.098$ from the Table 2. The price at which a broker offers a VM_j instance is determined as $p_j = m_j \cdot c^{max}_j$ where $m_1 = 1.2$, $m_2 = 1.25$, $m_3 = 1.3$ and $m_4 = 1.35$. Each CSP is considered to have 100 physical machines, each with 8 cores, 24 GB of memory and 160 GB of storage to lend for federation formation. This amounts to a capacity of 800 cores, 2400 GB of RAM and 16 TB of storage for each CSP.

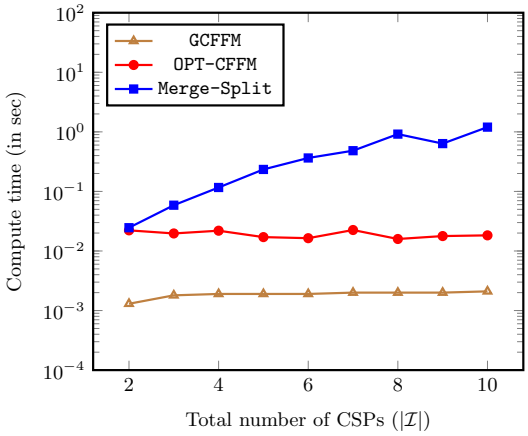
We generate 4 classes of consumer requests i.e., small, medium, large and extra large. Each class consists of 25 randomly generated requests based on a gaussian distribution and the same set of requests are used in all the experiments. Every request is a four tuple $r = (r_1, \cdots, r_4)$. Each coordinate of the tuple is generated using a Gaussian distribution with certain mean and standard deviation. The mean μ of the Gaussian distribution is set to 100, 200, 300 and 400 respectively for small, medium, large and extra large requests. The standard deviation is set to 30 for all the request types. The idea is that, since the mean of large requests is $(300, 300, 300, 300)$, the overall core requirements for such a request would be $300 * (1 + 2 + 4 + 8)$ which is 4500 cores. Hence, a minimum of 6 CSPs are required to form a federation to cater to large requests on an average.

We study the scalability of Merge-Split, OPT-CFFM and GCFFM (greedy) algorithms by varying the available CSPs, $\mathcal{I}$, from 2 to 10. When $|\mathcal{I}| = k$, we pick a random k subset from the 10 CSPs in Table 2. Then we run the three federation formation algorithms on the same set of 100 consumer requests which are already generated. We used IBM ILOG cplex optimization studio [2] [1] in Python to solve the integer programming problems. All the experiments are done on Intel(R) Core(TM) i3-4030U CPU @ 1.90GHz quadcore machine with 4GB of RAM.

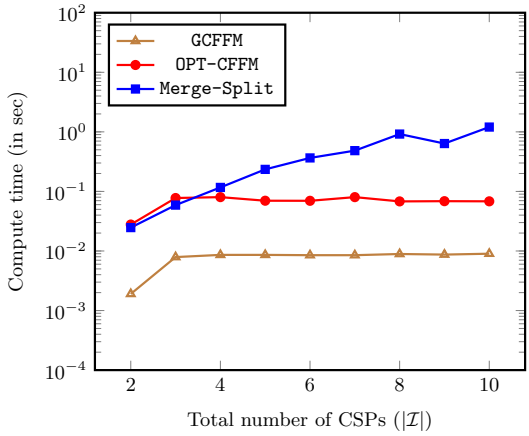
Figures 2(a), 3(a), 4(a) and 5(a) show the average time taken by Merge-Split, OPT-CFFM and greedy algorithms to arrive at a federation with the increasing value of $|\mathcal{I}|$. We can easily notice that the greedy algorithm performs the best, followed by OPT-CFFM and then Merge-Split. The time taken by greedy algorithm almost remains constant as it is a polynomial time algorithm and the range of $|\mathcal{I}|$ is too small to see the asymptotic growth as against the Merge-Split algorithm which has exponential time complexity. The greedy algorithm gives 200x speedup on an average when compared with the Merge-Split algorithm. Recall that both OPT-CFFM and greedy algorithm computes F_{opt} which may not be individually stable. We reduce F_{opt} to F^*_{opt} by applying the fixed point Algorithm 2. However, in our experimental results it has always been the case that both F_{opt} and F^*_{opt} are the same. After we arrive at F^*_{opt},

Amazon-EC2 regions	US East (N. Virginia) C_1	US West (Oregon) C_2	US West (N. California) C_3	EU (Ireland) C_4	EU (Frankfurt) C_5	Asia Pacific (Singapore) C_6	Asia Pacific (Tokyo) C_7	Asia Pacific (Sydney) C_8	South America (Sao Paulo) C_9	AWS GovCloud (US) C_{10}
VM_1 (medium)	0.067$	0.067$	0.077$	0.073$	0.079$	0.098$	0.096$	0.093$	0.095$	0.084$
VM_2 (large)	0.133$	0.133$	0.154$	0.146$	0.158$	0.196$	0.193$	0.186$	0.19$	0.168$
VM_3 (xlarge)	0.266$	0.266$	0.308$	0.293$	0.315$	0.392$	0.385$	0.372$	0.381$	0.336$
VM_4 (2xlarge)	0.532$	0.532$	0.616$	0.585$	0.632$	0.784$	0.77$	0.745$	0.761$	0.672$

Table 2: The cost of VM instance types from various CSPs.

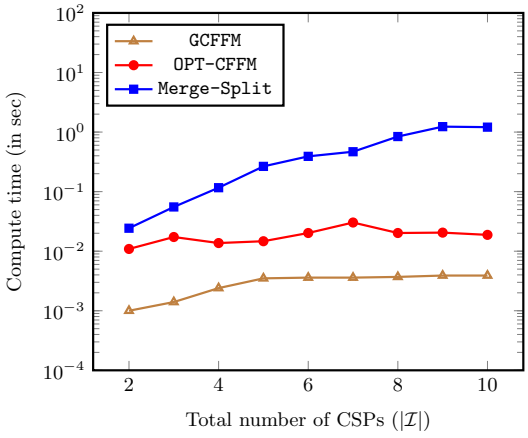


(a) Time taken for federation formation.

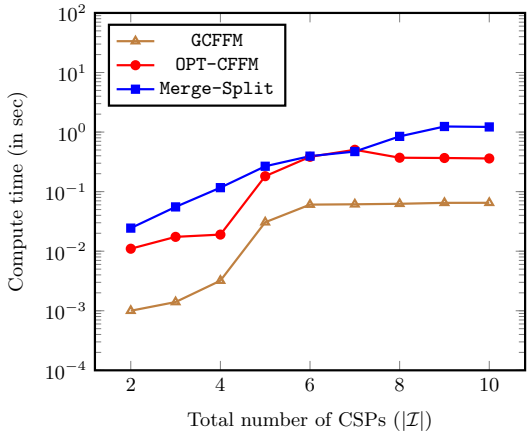


(b) Time taken for federation formation and payoff distribution computation.

Figure 2: Small requests

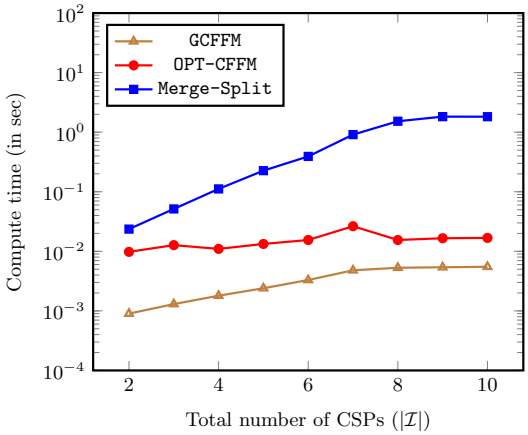


(a) Time taken for federation formation.

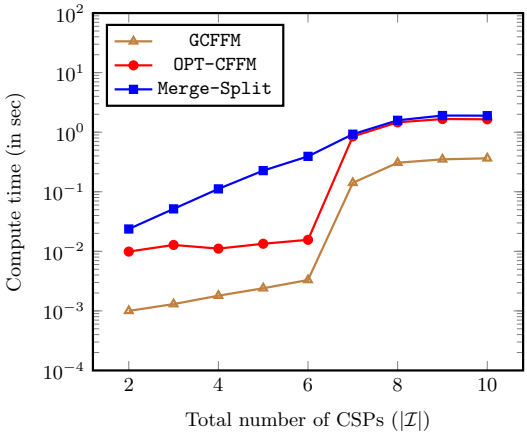


(b) Time taken for federation formation and payoff distribution computation.

Figure 3: Medium requests

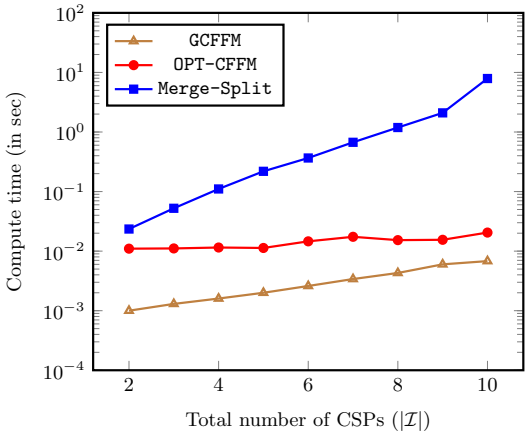


(a) Time taken for federation formation.

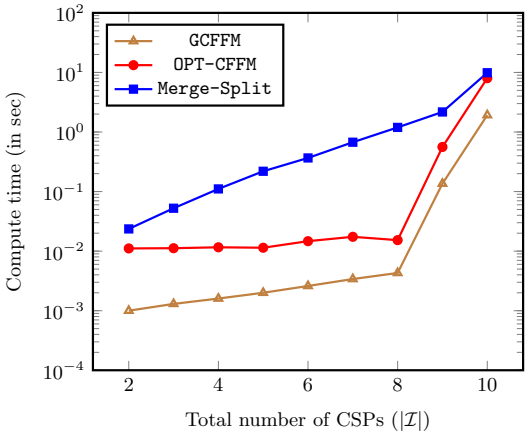


(b) Time taken for federation formation and payoff distribution computation.

Figure 4: Large requests



(a) Time taken for federation formation.



(b) Time taken for federation formation and payoff distribution computation.

Figure 5: Extra large requests

we compute the payoff distribution using exact normalized Banzhaf indices which requires solving $2^{|F^*_{opt}|}$ distinct integer linear programs. The Merge-Split algorithms computes approximate normalized Banzhaf indices as a part of the federation formation algorithm and hence does not require any further computational effort to determine the payoff distribution. Figures 2(b), 3(b), 4(b) and 5(b) shows the time required to arrive at a federation and the corresponding payoff distribution using Merge-Split, OPT-CFFM and greedy algorithms. It can be seen that time required for payoff computation goes up even for greedy especially for large and extra large requests which require large federations to serve them. However, greedy still performs 66x times faster on an average when compared with the Merge-Split algorithm. We leave the problem of arriving at new computationally efficient payoff distribution schemes to the future work. Finally, the value of federation obtained from greedy (and Merge-Split) is within a negligible 0.06 percent of the optimal federation obtained using the OPT-CFFM approach. This shows that the greedy algorithm is very practical as it computes almost optimal federations with two to three orders of magnitude speedup when compared with other approaches.

7. RELATED WORK

Rochwerger et al. [16, 15] proposed the reservoir model for federated cloud computing which addresses issues such as dynamic provisioning of resources, admission control, load balancing policies and cross cloud virtual networks. David Villegas et al. [20] showed the applicability of cloud federations across different layers of cloud services such as IaaS, PaaS and SaaS. Samaan [17] proposed capacity sharing strategies to maximize the revenue of the federations in the long run. Inigo Goiri et al. [6] proposed economic models for resource outsourcing and insourcing in federated clouds. Marco Guazzone et al. [7] proposed a distributed stable federation formation algorithm which attempts to reduce the energy costs of individual cloud providers. Mohammad Mehedi Hassan et al. [8] proposed a cooperative game based resource and revenue sharing mechanism which maximizes the social

welfare of the federation. Marian Mihailescu et al. [11] proposed a strategic proof dynamic pricing for resources in federated clouds. Antonio Celesti et al. [4] proposed a three phase model (discovery, match-making, and authentication) which enables clouds to cooperate with each other in a federated fashion. Dusit Niyato et al. [13] proposed a stochastic linear programming game model to study resource and revenue sharing in federated clouds. Mahyar Movahed Nejad et al. [12] designed efficient and truthful greedy mechanisms for dynamic virtual machine provisioning in clouds.

8. CONCLUSIONS

In this paper, we formulated the optimal federation formation problem as an integer linear programming problem and proposed a near optimal polynomial time greedy algorithm for the same. The value of the federation generated by the greedy algorithm is same as that of the optimal for all practical purposes. However, we do not have any provable approximation guarantees for the greedy algorithm. Once an optimal federation is obtained, then we compute the payoff distribution using the exact normalized Banzhaf index computation. In our future work, we would like to investigate computationally efficient payoff distribution schemes with good fairness properties. Further, whether the ratio between the value of an optimal federation and that of a federation generated by the Merge-Split algorithm is bounded by a constant or not, is an open problem.

9. REFERENCES

[1] IBM ilog cplex optimization studio v12.6.3. User Manual Available: http://www.ibm.com/support/knowledgecenter/SSSA5P_12.6.3/ilog.odms.studio.help/pdf/usrcplex.pdf.

[2] IBM ilog cplex software for academics. Available: https://www.ibm.com/developerworks/community/blogs/jfp/entry/cplex_studio_in_ibm_academic_initiative?lang=en.

[3] Priority queue. Available: https://en.wikipedia.org/wiki/Priority_queue.

[4] A. Celesti, F. Tusa, M. Villari, and A. Puliafito. How to enhance cloud architectures to enable cross-federation. *2010 IEEE 3rd International Conference on Cloud Computing*, pages 337–345, July 2010.

[5] G. Chalkiadakis, E. Elkind, and M. Wooldridge. *Computational Aspects of Cooperative Game Theory (Synthesis Lectures on Artificial Inetlligence and Machine Learning)*. Morgan & Claypool Publishers, 1st edition, 2011.

[6] I. Goiri, J. Guitart, and J. Torres. Characterizing cloud federation for enhancing providers' profit. *2010 IEEE 3rd International Conference on Cloud Computing*, pages 123–130, July 2014.

[7] M. Guazzone, C. Anglano, and M. Sereno. A game-theoretic approach to coalition formation in green cloud federations. *14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pages 618–625, May 2014.

[8] M. M. Hassan, M. A. Al-Wadud, and G. Fortino. A socially optimal resource and revenue sharing mechanism in cloud federations. *Computer Supported Cooperative Work in Design (CSCWD), 2015 IEEE 19th International Conference on*, pages 620–625, May 2015.

[9] J. G. J. H. Drèze. Hedonic coalitions: Optimality and stability. *Econometrica*, 48(4):987–1003, 1980.

[10] L. Mashayekhy, M. M. Nejad, and D. Grosu. Cloud federations in the sky: Formation game and mechanism. *IEEE Transactions on Cloud Computing*, 3(1):14–27, Jan 2015.

[11] M. Mihailescu and Y. M. Teo. Dynamic resource pricing on federated clouds. *10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pages 618–625, 2010.

[12] M. M. Nejad, L. Mashayekhy, and D. Grosu. A family of truthful greedy mechanisms for dynamic virtual machine provisioning and allocation in clouds. *2013 IEEE Sixth International Conference on Cloud Computing*, pages 188–195, June 2013.

[13] D. Niyato, A. V. Vasilakos, and Z. Kun. Resource and revenue sharing with coalition formation of cloud providers: Game theoretic approach. *2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 215–224, May 2011.

[14] G. Owen. *Game Theory*. 3 edition, 1995.

[15] B. Rochwerger, D. Breitgand, A. Epstein, D. Hadas, I. Loy, K. Nagin, J. Tordsson, C. Ragusa, M. Villari, S. Clayman, E. Levy, A. Maraschini, P. Massonet, H. Muñoz, and G. Toffetti. Reservoir - when one cloud is not enough. *Computer*, 44:44 – 51, 2011.

[16] B. Rochwerger, D. Breitgand, E. L. SAP, A. Galis, K. Nagin, I. M. Llorente, R. Montero, Y. Wolfstha, E. Elmroth, J. Cáceres, M. Ben-Yehuda, W. Emmeric, and F. Galán. The reservoir model and architecture for open federated cloud computing. *IBM Journal of Research and Development*, 53(4):535–545, July 2009.

[17] N. Samaan. A novel economic sharing model in a federation of selfish cloud providers. *IEEE Transactions on Parallel and Distributed Systems*, 25(1):12–21, Jan 2014.

[18] A. Singla, K. Desai, S. Purini, and V. Choppella. Distributed safety verification using vertex centric programming model. In *15th International Symposium on Parallel and Distributed Computing*, 2016.

[19] A. N. Toosi, R. N. Calheiros, and R. Buyya. Interconnected cloud computing environments: Challenges, taxonomy, and survey. *ACM Comput. Surv.*, 47(1):7:1–7:47, May 2014.

[20] D. Villegas, N. Bobroff, I. Rodero, J. Delgado, Y. Liu, A. Devarakonda, L. Fong, S. Sadjadi, and M. Parashar. Cloud federation in a layered service model. *JCSS Special Issue: Cloud Computing 2011*, 78:1330–1344, Sept 2012.