Self-optimising Decentralised Service Placement in Heterogeneous Cloud Federation

Emanuele Carlini
Massimo Coppola
Patrizio Dazzi
ISTI – CNR,
Pisa, Italy
Email: {name.surname}@isti.cnr.it

Matteo Mordacchini
Andrea Passarella
IIT – CNR,
Pisa, Italy
Email: {name.surname}@iit.cnr.it

***Abstract*—Clouds have been relevant business drivers and computational backends for a wide range of applications, including IoT, e-health, data analytics. To match the complex needs of such comprehensive set of different kinds of applications, in recent times there is an emerging need for new paradigms and forms of Clouds, organised according to a federated, heterogeneous and distributed structure. To exploit heterogeneity and localisation, in order to enhance the overall performances, ensure energy efficiency, reduce costs for resource providers and in the meantime enhance the user experience, proper service placement solutions are required. However, conducting efficient deployments in such a scenario is complex due to the dynamic nature of applications, resources, users. As a consequence, there the a need for scalable, distributed, adaptive, context-aware solutions characterised by high-efficiency and reduced overhead. We propose a highly distributed, self-adaptive solution aimed at optimising the overall deployment of cloud services by means of point-to-point interactions occurring among clouds and cloudlets belonging to the same federation. The contribution of this paper is the definition of a service exchange mechanism, its Markov-chain based modelling and thorough experimental evaluation.**

I. Introduction

Cloud Computing has deeply transformed the way in which IT resources are delivered to users and customers. Clouds [1] are the *de facto* instances for the provisioning of IT resources to the mass according the utility computing approach. Computing and storage are delivered in an on-demand fashion, in the shape of services, accounted to customers according to the pay per use model.

Large IT behemoths provide their spare resources for renting to other private enterprises as well as to individuals. However, with the evolution of services and applications that can be brought on the cloud, a single cloud solution cannot provide the heterogeneity and functionality required for many business solution. Therefore, the initial concept of cloud computing has rapidly evolved into multi-cloud environments, which gather together multiple and heterogeneous cloud data-centres and service providers.

The earlier protagonist of today's multi-cloud era are the so called *Cloud Federations* [2]–[5]. Cloud federation brought the cloud computing to the next level, realising much more than inter-cloud interoperability, rather providing an unified view on an heterogeneous pool of resources while using a single access point to control applications. In the Cloud Federation model a number of cloud providers voluntary join their resources to collaboratively increase their market and to achieve scale economy that would have been outside their reach. Therefore, in cloud federation an application is submitted via a specific cloud provider but its execution can in principle involve any combination of providers within the federation.

In the scientific and industrial cloud community there is a wide acknowledgement on the features that will characterise the future of clouds, data-centres and their large scale aggregations such as cloud federations, or from a more in general extent, to cyber-infrastructures [6,7]. Actually, next generation cyber-infrastructures are supposed to became more and more distributed, heterogeneous, federated and complex [8,9].

Proper deployments can exploit heterogeneity and localisation to enhance the overall performance, ensure energy efficiency, reduce costs for resource providers and in the meantime enhance the user experience. In fact, a service that is expensive to run with a given resource, can be cheap to run with a different resource.

Even more, there are clear signs [10]–[12] suggesting that a convergence with mobile, fog and edge computing will let clouds to be the computing enablers for a wider set of resources, such as IoT, sensors, smart-cities.

The widening in the spectrum of enabled applications, along with the more complex and distributed structure characterising clouds, calls for specific solutions for their management, able to match such peculiarities.

In particular, there is a call for advanced scalable solutions [13]–[16] that are able to deal with highly distributed heterogeneous systems. These systems do not assume the existence of a central authority, that concentrates all the decisions [17]–[21]. In fact, a proper decentralised solution is a fundamental pre-requisite to match the needs deriving from an highly-dynamic and widely deployed computational system. As matter of fact, decentralisation is required to conduct a proper resource allocation, as a centralised approach would require to gather in a unique place, as well as in a timely and frequent manner, a huge amount of information about the actual status of resources and service dynamics.

In this paper we propose a solution whose foundational

concept is rooted and borrowed from approaches dealing with highly distributed networked systems [22]–[25], such as those focusing on peer-to-peer computing. In such systems, it is usual to avoid the definition of a centralised authority, whose functional activities are instead provided as a result of a collaborative process involving many of (sometimes all) the elements participating in the computation.

The proposed approach aims to achieve an overall optimisation of the allocations of services to clouds, by means of decentralised, point-to-point interactions performed in a step-based discrete fashion. In fact, our solution consists in a highly distributed, self-adaptive system that is able to optimise the overall deployment of applications by means of the aforementioned interactions. These interactions occur among properly defined active entities, named *Cloud Drivers*, that are in charge to represent clouds. Each Cloud Driver contacts the other drivers and proposes an exchange of services, offering those it considers too expensive for itself and requesting the services it can run with a reduced effort.

In presenting our proposed solution, the paper gives the following contributions. An organically presented set of the related works, previously contributed to the scientific literature (Section II). The foundational concept underpinning the proposed solution is presented by outlining its similarities with the *Generalised Assignment Problem*, a well-known problem in the operational research field (Section III). In the very same section, we also present the algorithms, formally describing the control flow of our proposed approach. Our approach is also formalised through a Markov Chain model that organically describes its features and key elements in a conceptual framework (Section IV). Then, we provide the experimental results we obtained by implementing our proposed solution and testing it under different conditions, in order to provide its validity and viability (Section V). Finally, we draw our conclusion (Section VI).

II. Related Work

With respect to the optimisation of the resource utilisation among its participants, many approaches for Cloud Federation employ the concept of a centralised and structured entity that works as the controller of the federation, to drive the resource selection and the brokerage process [15]. In this sense, two different approaches are represented by Contrail [5] and Intercloud [26].

In particular, Contrail focuses both on the vertical and horizontal integration of multiple cloud providers, organising the enforcement of QoS by the definition of federation-level SLAs, which also drivers the resource selection process and can be mapped on cloud providers SLAs. Conversely, Intercloud's idea of Cloud Federation is realised via the definition of a common marketplace in which applications are negotiated among brokers and cloud providers.

Other approaches, such as RESERVOIR [27] and OPTIMIS [28], provided a more distributed view in the landscape of Cloud Federation. In RESERVOIR, providers communicate directly to each other to negotiate the utilisation of resources. Rather, OPTIMIS realises a services toolkit aimed to orchestrate the lifecycle of the applications, which in turn allows the self-management of the cloud federation.

On the other hand, the solution presented by Anastasi *et al.* [13] makes use of a mixed approach based on cross-entropy and lightweight methods, inspired by human cognitive process to drive the resource selection.

All the above solutions are based on centralised, infrastructure-controlled mechanisms. However, growing trends in Cloud computing push towards the evolution to more heterogeneous, decentralised and fog/edge-based approaches. A next generation cyber-infrastructure must be capable of targeting the above-mentioned needs. This fact calls for more distributed, self-organising and adaptive solutions for the optimisation of the resources within a federation of heterogeneous Clouds.

The problem of managing self-adapting resource allocation over federated clouds has been already tackled in the past, aiming at achieving a full autonomic distributed behaviour. For instance, in the paper from Ismail *et al.* [29] an autonomic, decentralised brokering framework is discussed. The paper aims at solving the issues of multiple cloud interactions by means of a reinforcement learning approach. The main difference with our approach is that in the Ismail *et al.* approach the assignment is computed once, according to the simplest brokering metaphor, whereas our approach has an iterative structure based on point-to-point interactions.

Other self-adaptive resource management systems for distributed data centers include the works proposed by Barbagallo *et al.* [30], and by Sedaghat *et al.* [31]. In particular, Barbagallo *et al.* [30] propose self-* techniques for optimising the energy consumption in a distributed data center. Through the interaction of autonomous entities, the system is able to adaptively re-distribute the load in a data center by exploiting some servers to the maximum efficiency, while allowing other to turn off, thus saving energy. On the other hand, Sedaghat *et al.* [31] propose to exploit a peer-to-peer adaptive system in oder to optimise the resource utilisation within a single cloud. In particular, the authors make use of a random gossip-based protocol to let servers in a cloud to exchange Virtual Machines (VMs). These exchanges allow the system to optimise its energy efficiency, considering the constrains on resource utilisation for both servers and VMs.

Our problem can be see akin to the (distributed) generalised assignment problem (DisGAP).

In particular, the case where items (services) are to be assigned to bins (Clouds) by a population of discrete agents via local knowledge can be cast as Distributed GAP. The work of Sun *et al.* [32] describes an asynchronous approach to solution search in this context, i.e. one based on distributed agents with only local knowledge. However, as most of the works in the field, that work focuses on finding exact solutions to complex constraint problems, which necessary involve backtracking at the local and global level. In the settings of a Federated Cloud, actual backtracking is impossible, and waiting for the whole network of clouds to converge toward an optimal solution

before applying it is unpractical.

Another related approach is the one presented by Dubois *et al.* [33]. The authors propose Mycocloud, a system designed to maximise the utilisation of each single node within a cloud computing system. To this end, Dubois *et al.* propose a decentralised, self-organising service placement that is based on a hierarchical peer-to-peer overlay, i.e. Myconet [34]. The goal of Mycocloud is twofold: maximise to resource utilisation of nodes within a cloud system and, at the same time, balance the nodes' computing load. Similarly to our solution, the problem faced by Mycocloud is a version of the Generalised Assignment Problem. Our solution differes from Mycocloud since our focus in on the maximisation of revenues of each single cloud in a federation and, consequently, the revenue of the federation as a whole. Clouds show a behaviour that is partly collaborative, and partly selfish. As explained in details in Sec. III, clouds collaborate with each other, since they exchange information and service instances. However, clouds are also selfish, since they try to perform only those exchanges that increase their own revenues. In doing this, we exploit a flat, unstructured peer-to-peer overlay.

Our work also drew inspiration by re-distribution approaches from other fields, in particular from solutions proposed to face the issues of electric power generation and power grid local management methods.

The distributed approach has been already investigated for the problem of trading and distributing energy on smart power grids. Kuntschke *et al.* [35] discuss the need to aim at both economical optimisation and power grid stability enforcement, where both commercial and customer-owned small power generation points are present. The two sides of the problem are tackled with a hierarchical organisation where many small generators are grouped together into virtual power plants, and where decisions on technical constraint management are by design kept separate from production by introducing a set of decisions entities run by a third party.

III. Concept and Architecture

As we stated in the introduction of this paper, in our envisioned scenario, whose graphical summarisation is given in Figure 1, the first class entities are **clouds** belonging to a federation. Each cloud is characterised by an heterogeneous set of installed resources and an heterogeneous set of hosted **services**. Each cloud is represented by a specific CLOUD DRIVER that is devoted to the interactions with other clouds. Each Cloud Driver has a detailed and up-to-date view of the resources and services belonging of its reference cloud. At the same time, it has only a very limited knowledge about the other clouds belonging to the federation. This vision is essentially limited to the address of their contact points, i.e., their Cloud Drivers.

Given the installed resources, each cloud can obtain a different profit by hosting a certain type of service. In fact, on the one hand we assume that the fee paid by customers for hosting a certain type of service is the same, regardless of the cloud selected for its execution. On the other hand, the cost borne by a cloud for its hosting varies from one cloud to another, depending on the installed resources. This results in a highly variable gain achieved both by each single cloud and by the overall federation, depending on the actual details of the allocation of services to clouds.

Our proposed approach targets the problem of a fair and optimising inter-exchange of services among clouds. Each cloud aims at exchanging services with other clouds in order to increase its own profit. Counterparts accept only when the profit is mutual. As a consequence, from a federation-wide perspective, the conduction of point-to-point exchanges among different pairs of clouds will lead to a global-level optimisation.

A. Similarities with the Generalised Assignment Problem

The problem we are considering, defined in these terms, recalls the maximum Generalised Assignment Problem [36]. It is a well-known problem in combinatorial optimisation. Such problem is defined as a generalisation of the assignment problem [37]. This problem instance assumes a number of agents and a number of tasks. Any agent can be assigned to perform any task, sustaining costs that may vary, depending on the agent-task assignment. It is required to perform all tasks by *assigning exactly one agent to each task and exactly one task to each agent* in such a way that the total cost of the assignment is minimised. The generalisation derives from the specific definition of the problem, in which both tasks and agents have a size. Moreover, the size of each task, and the number of tasks hosted might vary from one agent to the other. The generalised assignment problem, in its most general form is defined as follows. There are a number of agents and a number of tasks, corresponding in our envisioned scenario to clouds and services, respectively. Any agent can be assigned to perform any task, incurring some cost and profit that may vary depending on the agent-task assignment. Moreover, each agent has a budget, which is the upper bound on the sum of the costs of the assigned tasks.

The problem requires to find an assignment in which all agents do not exceed their own budget and the total profit of the assignment is maximised.

From a more formal viewpoint, the problem can be expressed as follows. We have n kinds of items, a_1 through a_n and m kinds of bins b_1 through b_m. Each bin b_i is associated with a budget t_i. For a bin b_i, each item a_j has a profit p_{ij} and a weight w_{ij}.

A solution is an assignment from items to bins. A feasible solution is a solution in which for each bin b_i the total weight of assigned items is at most t_i. The solution's profit is the sum of profits for each item-bin assignment. The goal is to find a maximum profit feasible solution. Thus, mathematically the generalised assignment problem can be formulated as an integer program:

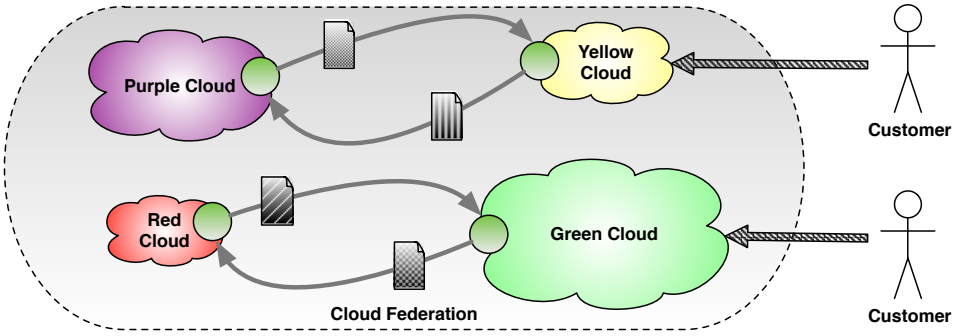


Fig. 1: Graphical Representation of Our Envisioned Scenario

$$\text{Maximise: } \sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}x_{ij}$$

Subject to:

$$\sum_{j=1}^{n} w_{ij}x_{ij} \leq t_i \qquad i = 1,\ldots,m \qquad (1)$$

$$\sum_{i=1}^{m} x_{ij} = 1 \qquad j = 1,\ldots,n \qquad (2)$$

$$x_{ij} \in \{0,1\} \qquad i = 1,\ldots,m, \quad j = 1,\ldots,n \qquad (3)$$

Given this formulation, our problem can be presented in a pretty similar way. In fact, clouds can be modelled as bins and services as items. The different profits depends on the heterogeneity characterising clouds, i.e., different kind of resources can lead to a different efficiency in running services, in turn resulting to different cost for their hosting.

Instead, in our envisioned scenario the constraint on weights is a bit relaxed, considering that they always equal to 1, independently from the service and the target cloud. Even more, the total amount of services assigned to a cloud is fixed. As a consequence, the inequality (1) becomes:

$$\sum_{j=1}^{n} x_{ij} = t_i \qquad i = 1,\ldots,m \qquad (4)$$

B. Our Approach

The Generalised Assignment Problem (GAP) is NP-hard, and it is even APX-hard to approximate. Recently it was shown that an extension of it is $e/(e-1) - \epsilon$ hard to approximate for every ϵ [38]. As a consequence, its application on large, real-world problems requires the exploitation of approximated heuristic approaches. From this viewpoint, our approach is a distributed, and autonomous heuristic for solving the GAP.

In fact, the approach we propose has a completely decentralised nature. As we already mentioned, it is based on point-to-point interactions occurring among cloud representatives, called Cloud Drivers. For each iteration of the protocol, each Driver performs the very same set of operations, devoted to the optimisation of its own set of hosted services. From a formal viewpoint, the process is represented by Algorithms 1 and 2. The control flow of our proposed solution is driven by Algorithm 1.

- Each driver takes the role of "initiator" and starts the activities underpinning the protocol.
- The initiator updates its information about the current status of the resources belonging to the cloud it manages.
- Then the initiator selects a "counterpart" cloud among those participating in the federation. In principle many different approaches can be adopted at this stage. In this paper, we adopt a simple random selection. Random selection is recognized as one of the techniques that avoid that a distributed system get stuck on local optima, rather than tending toward a global optimal solution [39]–[41].
- Thereafter, the initiator contacts the counterpart cloud to get an up-to-date information about the status of the services and the resources belonging to the counterpart.
- The initiator combines all the information gathered to build an overall representation of the status, involving both its own cloud and the counterpart one.
- The initiator analyses the gathered information. If a service exchange with the counterpart can potentially lead to an economic gain for both the involved clouds, the initiator issues to the counterpart an exchange proposal involving two services, one currently in execution on the initiator cloud, the other running on the counterpart cloud. Additional details about this phase are reported later in this section.
- If the counterpart accepts, the exchange takes place. For each cycle, each cloud takes at least the role of "initiator"; it can also behaves as "counterpart" if selected by another "initiator" cloud.

The proposal preparation phase is depicted in Algorithm 2. This phase consists of different steps described in the remainder of this section.

- The proposal preparation starts by extracting the information about the current status of services and resources, considering both the initiator and the counterpart cloud.
- This information is exploited to define two sets. The first set is composed of the pairs of services (one from

```
while not yet converged do
    S_I ← updateServiceStatus();
    R_I ← updateResourceStatus();
    C_part ← SelectCounterpart();
    S_C ← getServices(C_part);
    R_C ← getResources(C_part);
    status ← (S_I, R_I, S_C, R_C);
    proposal ← prepareProposal(status);
    if proposal ≠ ∅ then
        eval ← issueProposal(proposal, C_part);
        if eval then
            consolidateChanges(proposal)
        end
    end
end
```

Algorithm 1: Main Loop of the Proposed Approach

```
function prepareProposal(status: StatusType):
    S_I ← status.S_I;
    R_I ← status.R_I;
    R_C ← status.R_C;
    S_C ← status.S_C;
    A ← { a ∈ S_I, b ∈ S_C | C(a, R_I) > C(b, R_I)};
    B ← { a ∈ S_I, b ∈ S_C | C(b, R_C) > C(a, R_C)};
    potentialExchanges ← A ∩ B;
    if potentialExchanges ≠ ∅ then
        return take(potentialExchanges);
    else
        return ∅;
    end
end
```

Algorithm 2: Proposal Preparation

the initiator cloud, the other from the counterpart cloud) that make an exchange profitable for the initiator cloud. Conversely, the second set contains the couples of services for which an exchange would be beneficial for the counterpart cloud.

- Then, the intersection of the two sets is computed. The result contains the couples that ensure a benefit for both the clouds involved in the service exchange.
- If the intersection is not empty, one of the pairs is taken and returned. In our experiments, we rely on a random choice for this selection. However, more complex solutions can be adopted.

IV. Formalisation of the Approach

In this section, we propose a modelisation of the approach described in the previous sections. The goal of this model is to describe the evolution over time of the number of instances of each service type that are in execution on each of the clouds of the federation.

Symbol	*Definition*
N	Number of clouds in the Federation
C_i	Cloud i inside the Federation
b_i	Maximum number of execution slots available on C_i
M	Total number of service types in the Federation
s_k	Service type k, from the set of service types $S = \{s_1, \ldots, s_M\}$
I_k	Total number of instances of s_k available in the Federation
L_{ik}	Max. number of istances of s_k that could be hosted on a cloud C_i
$\delta_{ij}(a, b)$	Function that evaluates the convience of an exchange of services a and b between clouds C_i and C_j
$P_i(s_k)$	Probability that a cloud C_i has at least one instance of services of type s_k
$P_i(s_k \rightarrow C_j)$	Probability that C_i pass an instance of s_k to C_j
$P_i(s_k \leftarrow C_j)$	Probability that C_i receives an instance of s_k to C_j
$\tilde{\pi}_t^i[s_k]$	Probability distribution vector of the MC associated to s_k on C_i at time t

TABLE I: List of mathematical notations

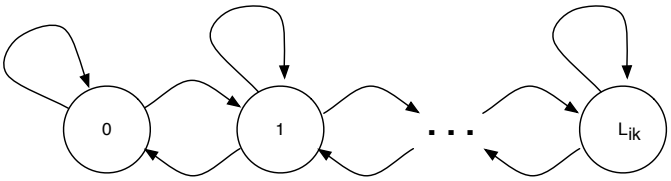


Fig. 2: Markov Chain associated to a service type s_k on a cloud C_i

To this end, we make use of Discrete-Time Markov Chains. In fact, the spreading of services towards the sites that optimise the Federation revenue is based on the interaction of couple of clouds, occurring at discrete times. Markov Chains have been successfully used in the literature to model systems that act in similar conditions [42]–[44]. In the following description, we make use of the symbols reported in Table I.

Specifically, being $C = \{C_1, \ldots, C_N\}$ the set of clouds of the federation and $S = \{s_1, \ldots, s_M\}$ the set of service types, each cloud $C_i \in C$ maintains a separate Markov Chain (MC) for every service type in S. A MC associated to a service type $s_k \in S$ is modeled as the one shown in Fig. 2.

In particular, we assume that a cloud C_i has a maximum of available execution slots b_i. Moreover, service type s_k has I_k instances in execution throughout the whole Cloud Federation. In order to describe the evolution over time of the number of instances of s_k that are hosted simultaneously on C_i, the MC associated with s_k on C_i has $L_{ik} + 1$ states, where $L_{ik} = \min\{b_i, I_k\}$.

From the definition of the proposed approach given in the previous sections, changes in the number of instances of a

service type in execution on a given cloud could occur only as the effect of an exchange of services between two clouds. Therefore, in order to derive the transition probabilities of a MC like the one depicted in Fig. 2, we have to describe the probability for a cloud to give or receive an instance of a service type at any time t, when an interaction between clouds takes place.

To this end, let us consider an interaction at a generic time t, involving a service type s_k, occurring between a cloud C_i and a cloud C_j. To derive the probabilities of C_i to pass/receive an instance of s_k to/from C_j, we define a function $\delta_{ij}(a, b, t)$. This function measures whether it is convenient for both C_i and C_j to pass an instance of a service a from C_i to C_j, in exchange for an instance of service b from C_j to C_i. Formally, $\delta_{ij}(a, b, t)$ is defined as:

$$\delta_{ij}(a,b,t) = \begin{cases} 1 & \text{if } Cost(a, C_i, t) > Cost(b, C_i, t) \\ & \text{AND } Cost(b, C_j, t) > Cost(a, C_j, t) \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where $Cost(\cdot, \cdot, t)$ denotes the cost of running an instance of a service on a given cloud at time t. Note that this definition of cost allows the model to cope with variable conditions, where the cost of running a service is not fixed and could change over time. Let $\tilde{\pi}^i_t[s_k]$ be the probability distribution vector at time t of the MC associated to s_k on C_i, and let also $\tilde{\pi}^i_{u,t}[s_k]$ be the probability associated to the state u of this MC at time t. As a consequence, we can define $P_i(s_k) = \sum_{u=1}^{L_{ik}} \tilde{\pi}^i_{u,t}[s_k]$, the probability that C_i has at least one instance of s_k at time t.

Using these definitions, we can compute the probability $P_i(s_k \rightarrow C_j)$ that C_i will pass an instance of s_k to C_j. This probability depends on the fact that C_j has an instance of at least one other service type s_m, for which the exchange is fruitful for both the clouds. This can be expressed as:

$$P_i(s_k \rightarrow C_j) = 1 - \prod_{m \neq k} (1 - \delta_{ij}(s_k, s_m) P_j(s_m)) \tag{6}$$

On the other end, the probability $P_i(s_k \leftarrow C_j)$ that C_i will receive an instance of s_k from C_j depends on the probability that C_i has at least an instance of another service type for which the exchange could be done. We can can compute this probability as:

$$P_i(s_k \leftarrow C_j) = P_j(s_k) \left(1 - \prod_{m \neq k} (1 - \delta_{ij}(s_m, s_k) P_i(s_m)) \right) \tag{7}$$

While Formulas 6 and 7 describe the exchange probabilities of a service type s_k in an interaction with only one other cloud, the overall probabilities $P^{pass}_{i,t}(s_k)$ and $P^{rec}_{i,t}(s_k)$ for C_i to pass or receive to/from any of the other clouds an instance of s_k at time t, respectively, can be defined as:

Description	*Values*
federation size	$8, 16, 32, 64, 128$
cloud-services cost distribution	Normal, Zipfian, Uniform, Ad-hoc
cloud capacity distribution	fixed value (homogeneous), normal
service types	32
total service instances	1024

TABLE II: Main simulation parameters and their values

$$P^{pass}_{i,t}(s_k) = \sum_{j=1, j \neq i}^{N} \frac{1}{N-1} P_i(s_k \rightarrow C_j) \tag{8}$$

$$P^{rec}_{i,t}(s_k) = \sum_{j=1, j \neq i}^{N} \frac{1}{N-1} P_i(s_k \leftarrow C_j) \tag{9}$$

Thus, each entry $p_{u,v}$ of the transition matrix P_t of the MC of s_k on C_i at time t can be defined as:

$$p_{u,v} = \begin{cases} P^{rec}_{i,t}(s_k) & \text{if } v = u+1, u \in [0, M-1] \\ P^{pass}_{i,t}(s_k) & \text{if } v = u-1,\ u \in [1, M] \\ 1 - \left(P^{rec}_{i,t}(s_k) + P^{pass}_{i,t}(s_k) \right) & \text{if } v = u,\ u \in [1, M-1] \\ 1 - P^{rec}_{i,t}(s_k) & \text{if } v = u = 0 \\ 1 - P^{pass}_{i,t}(s_k) & \text{if } v = u = M \\ 0 & \text{otherwise} \end{cases}$$

The initial condition $\tilde{\pi}^i_0[s_k]$ reflects the initial distribution of the service instances in the whole Cloud Federation.

It is worth noting that clouds having the same associations of costs with services can be described by the very same set of MCs, thus reducing the model complexity.

V. Experimental Evaluation

In order to validate our approach, we simulated a distributed federation using the Peersim [45] simulator. Our simulation is divided into iterations occurring at discrete time intervals.

Each cloud provider is an agent that, once per iteration, acts as initiator. We assume that the clouds belonging to the federation are able to communicate with each other and that there is no penalty in the migration of a service from one cloud provider to another. In addition, we assume that any service can be run on any cloud; potential incompatibilities between cloud and services can be modelled by defining an high cost of execution. All the presented values are the average of 10 independent runs. The only exception is the test focusing on the adherence to the model, where the results represent the mean on 50 independent executions. The services executed on the clouds belong to different types, each characterised by a different resource fingerprint, i.e., differentiated requirements on distinct cloud resources needed for the execution of services. The approach of organising services in categories on the basis of their requirements is quite common in the cloud environment [13,46], as it demonstrated to be a viable approach for representing real-world scenarios.

In order to evaluate the proposed approach, we used the following metrics:

- *gain*; The gain is defined as the sum of the difference between the cost for running a service and its revenue, done for all the services in the federation.
- *overhead*; The overhead is defined as the number of service exchanged among clouds. In the following graphs, we present the cumulative overhead.
- *Gini coefficient*; The Gini coefficient measures the inequality of values in a distribution and it is often used as a metric to evaluate the balance of a value in a distributed system [47]. In our case, it measures the inequalities between the individual gain of clouds (note that when computing the coefficient, we normalise the gain according to the number of services executed by the cloud). The coefficient is defined in the interval $[0, 1)$, with 0 (note that in the following graphs we used the percentage value) representing the perfect equality, i.e. all clouds have the same gain, and 1 the perfect inequality, i.e. one cloud provider has all the possible gain. More formally, in our cloud federation the Gini coefficient at time t is defined as:

$$G^t = \frac{\sum_{i \in N} \sum_{j \in N} |gain_x^t - gain_y^t|}{2n^2\mu} \tag{10}$$

where N is the set containing all the clouds, and n its cardinality.

A. Adherence to the model

In order to validate the simulation against the model, we setup a test to compare the results of the two. The test considered a scenario with 12 clouds, 12 service types and 120 total service instances (10 instances per each type). The service-cloud cost assignment is done considering the ad-hoc assignment (described in Section V-C). Each cloud can manage 10 services and the initial assignment of services to clouds is done randomly. The results obtained with the simulator are the mean of 50 independent runs.

Note that the model has been implemented independently from the simulation, by using a general tool to resolve Markov chains. Figure 3 presents the comparison. The green area is the variance of the simulator results. From this figure, it is evident that, regarding the gain, we can consider the simulation as a good approximation of the model.

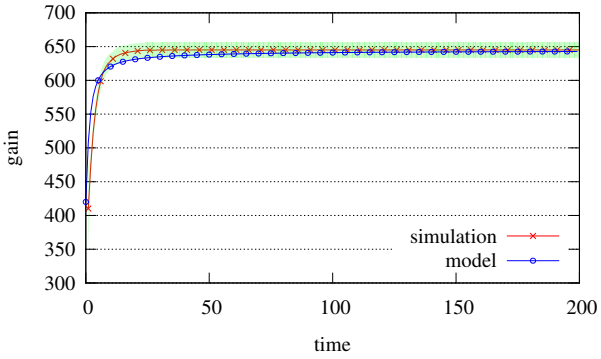


Fig. 3: System gain: model vs simulation

B. Varying the size of the federation

In this experiment we measure how federations of different size converge to a solution. We considered 32 types of services, with each type having 32 instances (in total 1024 services in the federation). The initial assignment of services to clouds is done randomly, with each cloud having the same amount of services. The cost of running a service on a cloud is randomly distributed in the interval $[1, 16]$.We varied the size of the federation with the following values $[8, 16, 32, 64, 128]$.

Figure 4a shows the gain of the federation over time. From the result it is clear how the size of the federation has an impact both in terms of speed of convergence and the total gain reached once converged. In particular, when the number of services types is far more than the number of clouds, there are few clouds that allow for a cost-effective execution of certain type of services. This is evident with a federation with 8 clouds that converges slower that larger federations, and reaches a lower convergence value. The overhead (see Figure 4c) practically follows the same trend as the gain. This similarity is due to the fact that the clouds swap only one service per exchange and therefore the overhead is proportionally related to the gain.

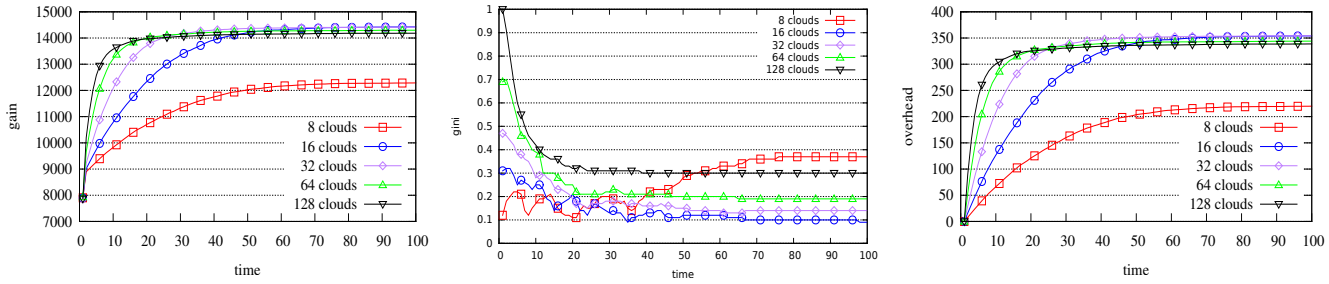
Figure 4b shows the gain of the federation over time. The Gini coefficient tends to decrease over time for most of the sizes of the federation. When the size of the federation is greater than 8, the decrease is smooth and leads to nice values of the Gini coefficient. With a federation composed by 8 clouds, the Gini initially decreases meaning that the exchanges are equally convenient for the parts in the exchange. However, toward the completion of the simulation the Gini coefficient increases, meaning that the exchanges are more valuable for those clouds that can afford a cost-effective execution of certain type of services.

This fact reinforces the idea that for a smooth functioning of the system the size of the federation shall be larger than the number of types of services. In particular, in our setup a federation composed of 64 clouds is the best tradeoff in terms of gain and Gini coefficient.

C. Varying the cost of services

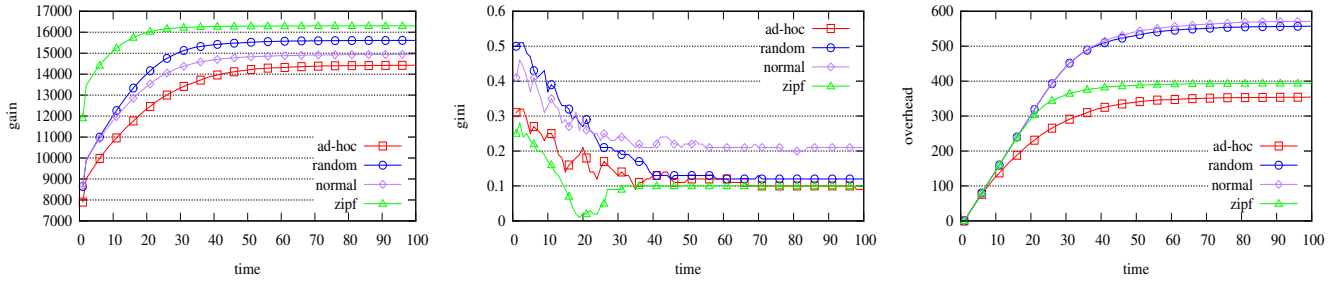
In this experiment we studied how different strategies of assigning costs to services impact on the behaviour of the service exchange algorithm. We setup a federation with 16 clouds and 1024 services (16 types, 64 of each type). The initial assignment of services to clouds is done randomly, with each cloud having the same amount of services. We varied the assignment of the costs to services by using the following distributions: zipfian, normal and uniform. We also used an ad-hoc assignment of costs in which services are grouped in pairs, in such a way that the services in each pair have the the same cost within a cloud, but different clouds have different costs for the same pair.

Figure 5a shows the evolution of the system gain over time. All the cost distributions converge smoothly, and after the 40th iteration a stable convergence is reached. In this experiment the value of the gain once converged is not relevant, due



(a) Total gain of the federation (b) Gini coefficient (c) Overhead

Fig. 4: Varying the size of the federation



(a) Total gain of the federation (b) Gini coefficient (c) Overhead

Fig. 5: Varying the cost assignment of services

to the different cost distributions. For example, the zipfian distribution obtains the higher value since its cost distribution is heavily skewed toward the minimum cost of the service.

Regarding the Gini coefficient (see Figure 5b) the zipfian distribution decreases very rapidly when small-cost services are assigned to the clouds. After that first phases, the exchange of high-cost services is not balanced and this reflects in an increasing value of the Gini coefficient. This also can be confirmed by the overhead (see Figure 5c): initially, with the zipfian distribution the federation executes the larger part of the exchanges, and then it slows down around the 20th iteration, when high-cost services are exchanged.

D. Varying the capacities of the clouds

This experiment evaluates the behaviour in terms of the capacity of the clouds, defined as the number of services they can execute. We consider a federation with 16 clouds and 1024 services (16 types, 64 of each type). The cost of the services follows a normal distribution. We defined two scenarios: (i) homogeneous, all clouds have the same capacity, (ii) heterogeneous, the capacity of each cloud is assigned following a normal distribution with average 32 and variance 16.

Results are presented in Figure 6. Regarding gain, we notice that in the homogeneous scenario the final value is slightly better and convergence is faster than in the heterogeneous case. As a matter of fact, the difference in capacity among clouds does not seem to have a relevant impact on the gain of the system. Similar considerations can be made for the number of exchanges.

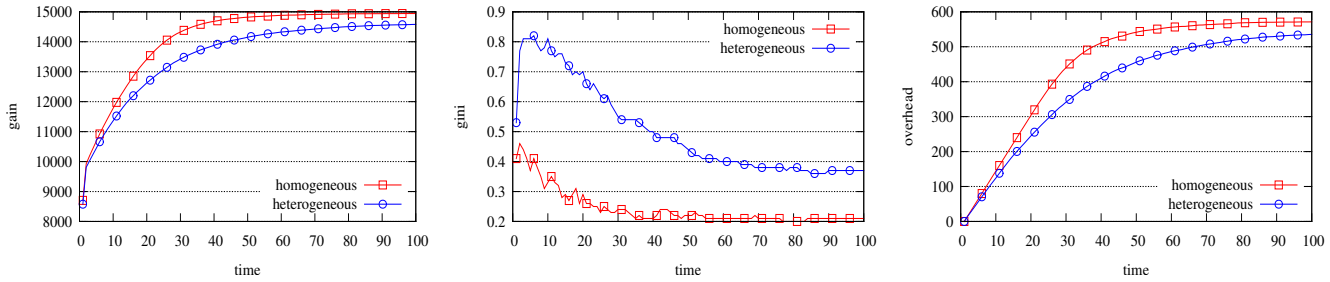
Intuitively, the Gini coefficient is affected by the capacity of the cloud, regardless its computation is normalised according to the number of services.

Evidently, with heterogeneous capacities the gain distribution in the system is more skewed than with the homogeneous one. As a side consideration, it is interesting to note that good values of the total gain are not always associated to an as much as good gini coefficient.

E. Comparison with a GAP solver

As discussed in the related work, our problem can be reduced to a Generalised Assignment Problem (GAP). Therefore, our approach can be seen as a decentralised solver for GAP. We compared our results against a centralised GAP solver [48] as we believe that this comparison can represent a good baseline to evaluate the performances of our approach. The centralised GAP solver employs a greedy heuristics that assure an α-approximation. We considered the same experimental setup as in Section V-C in which services costs are assigned using different distributions. To make for a fair comparison, in our approach we considered the gain of the system after the 100 iterations. Results are presented in Table III.

When considering the normal, random and ad-hoc cost assignments, the GAP solver obtains better results, with an higher gain of around 10%. Interestingly, when considering the zipfian assignment our approach obtains (slightly) better results than the GAP solver, in the order of 0.5%. We explain this behaviour with the fact that the zipfian distribution creates a scenario in which there are few clouds are very convenient



(a) Total gain of the federation (b) Gini coefficient (c) Overhead

Fig. 6: Varying the capacity of the clouds

	Total gain	
cost assignment	*our approach*	*GAP solver*
normal	14942.6	**16223**
zipfian	**16348**	16256
random	15611.4	**17504**
ad-hoc	14424	**16384**

TABLE III: Comparison with GAP solver

for certain services type. Our greedy-like approach spots this assignments easily, as can be also seen in Figure 5a, where the zipfian scenario arrives to the maximum gain in the earlier iterations. Therefore, in terms of total gain, the performance of our approach are comparable with a centralised GAP solver when considering a zipfian assignment of the costs.

VI. Conclusion and Future Work

In this paper we proposed a highly distributed and self-adaptive service exchange protocol. It is designed for optimising the assignment of services by means of point-to-point interactions among clouds belonging to the same federation. We have defined a model based on Markov chains that demonstrates that our approach can converge in a distributed setting. On top of that, we provided additional results via simulations, by varying several parameters of the cloud federation.

From the simulation results, we can conclude that the approach is effective when the number of clouds is relatively large compared to the kind of services accepted by the federation, and that the capacity of the clouds mostly impacts on the distribution of the revenues among clouds, while it keeps the total revenue of the federation practically the same.

As future work, we plan to extend our approach in several directions. We aim toward an extensive theoretical analysis on the hardness of the problem and its relation with the DistGAP problem. We also plan to extend our simulation in order to investigate large-case scenarios and dynamic service arrival/completion conditions. Besides this, we plan to investigate how a more realistic characterisation of services can impact on our proposed approach, e.g., services with a dynamic workload, costs associated with service placement and migration, etc. Finally, we plan to consider more complex approaches for performing the service exchange, e.g., asymmetric exchanges.

Acknowledgement

The authors acknowledge the support of project H2020-723131 BASMATI: Cloud Brokerage Across Borders for Mobile Users and Applications.

References

[1] L. Wang, R. Ranjan, J. Chen, and B. Benatallah, *Cloud computing: methodology, systems, and applications*. CRC Press, 2011.

[2] T. Kurze, M. Klems, D. Bermbach, A. Lenk, S. Tai, and M. Kunze, "Cloud federation," in *CLOUD COMPUTING 2011 : The 2nd Int.l Conference on Cloud Computing, GRIDs, and Virtualization*. IARIA, 2011.

[3] R. Moreno-Vozmediano, R. S. Montero, and I. M. Llorente, "Iaas cloud architecture: From virtualized datacenters to federated cloud infrastructures," *Computer*, no. 12, pp. 65–72, 2012.

[4] M. M. Hassan, M. S. Hossain, A. J. Sarkar, and E.-N. Huh, "Cooperative game-based distributed resource allocation in horizontal dynamic cloud federation platform," *Information Systems Frontiers*, vol. 16, no. 4, pp. 523–542, 2014.

[5] E. Carlini, M. Coppola, P. Dazzi, L. Ricci, and G. Righetti, "Cloud federations in contrail," in *Euro-Par 2011: Parallel Processing Workshops*. Springer, 2011, pp. 159–168.

[6] Gartner, "Eight trends will shape the colocation market in 2016." [Online]. Available: https://www.gartner.com/doc/reprints?id=1-2X9EKB9&ct=160128&st=sb.

[7] L. Schubert, K. G. Jeffery, and B. Neidecker-Lutz, *The Future of Cloud Computing: Opportunities for European Cloud Computing Beyond 2010:–expert Group Report*. European Commission, Information Society and Media, 2010.

[8] R. Moreno-Vozmediano, R. S. Montero, and I. M. Llorente, "Key challenges in cloud computing: Enabling the future internet of services," *Internet Computing, IEEE*, vol. 17, no. 4, pp. 18–25, 2013.

[9] J. Warren, "Equinix says hybrid cloud is the future." [Online]. Available: http://www.forbes.com/sites/justinwarren/2016/03/09/equinix-says-hybrid-cloud-is-the-future/#7b53a602a953

[10] S. R. Carter, "Techniques for dynamic cloud-based edge service computing," Nov. 17 2011, uS Patent App. 12/780,328.

[11] S. Choy, B. Wong, G. Simon, and C. Rosenberg, "A hybrid edge-cloud architecture for reducing on-demand gaming latency," *Multimedia Systems*, vol. 20, no. 5, pp. 503–519, 2014.

[12] M. Satyanarayanan, R. Schuster, M. Ebling, G. Fettweis, H. Flinck, K. Joshi, and K. Sabnani, "An open ecosystem for mobile-cloud convergence," *Communications Magazine, IEEE*, vol. 53, no. 3, pp. 63–70, 2015.

[13] G. F. Anastasi, P. Cassara, P. Dazzi, A. Gotta, M. Mordacchini, and A. Passarella, "A hybrid cross-entropy cognitive-based algorithm for resource allocation in cloud environments," in *Self-Adaptive and Self-Organizing Systems (SASO), 2014 IEEE Eighth International Conference on*. IEEE, 2014, pp. 11–20.

[14] R. Baraglia, P. Dazzi, G. Capannini, and G. Pagano, "A multi-criteria job scheduling framework for large computing farms," in *10th IEEE International Conference on Computer and Information Technology*. IEEE, 2010, pp. 187–194.

[15] G. F. Anastasi, E. Carlini, M. Coppola, and P. Dazzi, "Qbrokage: A genetic approach for qos cloud brokering," in *Cloud Computing (CLOUD), 2014 IEEE 7th International Conference on*. IEEE, 2014, pp. 304–311.

[16] M. Danelutto and P. Dazzi, "A java/jini framework supporting stream parallel computations," in *Proceedings of the International Conference ParCo 2005*, G. R. J. et al., Ed., 2005.

[17] P. T. Endo, A. V. de Almeida Palhares, N. N. Pereira, G. E. Goncalves, D. Sadok, J. Kelner, B. Melander, and J.-E. Mångs, "Resource allocation for distributed cloud: concepts and research challenges," *Network, IEEE*, vol. 25, no. 4, pp. 42–46, 2011.

[18] M. Steiner, B. G. Gaglianello, V. Gurbani, V. Hilt, W. D. Roome, M. Scharf, and T. Voith, "Network-aware service placement in a distributed cloud environment," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 73–74, 2012.

[19] Y. Kessaci, N. Melab, and E.-G. Talbi, "A pareto-based metaheuristic for scheduling hpc applications on a geographically distributed cloud federation," *Cluster Computing*, vol. 16, no. 3, pp. 451–468, 2013.

[20] R. Baraglia, P. Dazzi, M. Mordacchini, L. Ricci, and L. Alessi, "Group: A gossip based building community protocol," in *Smart Spaces and Next Generation Wired/Wireless Networking*. Springer Berlin Heidelberg, 2011, pp. 496–507.

[21] M. Mordacchini, P. Dazzi, G. Tolomei, R. Baraglia, F. Silvestri, and S. Orlando, "Challenges in designing an interest-based distributed aggregation of users in p2p systems," in *2009 International Conference on Ultra Modern Telecommunications & Workshops*. IEEE, 2009, pp. 1–8.

[22] M. Mordacchini, R. Baraglia, P. Dazzi, and L. Ricci, "A p2p recommender system based on gossip overlays (prego)," in *IEEE 10th International Conference on Computer and Information Technology (CIT), 2010*. IEEE, 2010, pp. 83–90.

[23] E. Carlini, M. Coppola, P. Dazzi, D. Laforenza, S. Martinelli, and L. Ricci, "Service and resource discovery supports over p2p overlays," in *Ultra Modern Telecommunications & Workshops, 2009. ICUMT'09. International Conference on*. IEEE, 2009, pp. 1–8.

[24] P. Dazzi, P. Felber, L. Leonini, M. Mordacchini, R. Perego, M. Rajman, and É. Rivière, "Peer-to-peer clustering of web-browsing users," *Proc. LSDS-IR*, pp. 71–78, 2009.

[25] R. Baraglia, P. Dazzi, B. Guidi, and L. Ricci, "Godel: Delaunay overlays in p2p networks via gossip," in *IEEE 12th International Conference on Peer-to-Peer Computing (P2P)*. IEEE, 2012, pp. 1–12.

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

[27] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, I. M. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Caceres *et al.*, "The reservoir model and architecture for open federated cloud computing," *IBM Journal of Research and Development*, vol. 53, no. 4, pp. 4–1, 2009.

[28] A. J. Ferrer, F. Hernández, J. Tordsson, E. Elmroth, A. Ali-Eldin, C. Zsigri, R. Sirvent, J. Guitart, R. M. Badia, K. Djemame *et al.*, "Optimis: A holistic approach to cloud service provisioning," *Future Generation Computer Systems*, vol. 28, no. 1, pp. 66–77, 2012.

[29] A. Ismail and V. Cardellini, *Advances in Service-Oriented and Cloud Computing: Workshops of ESOCC 2014, Manchester, UK, September 2-4, 2014, Revised Selected Papers*. Cham: Springer International Publishing, 2015, ch. Decentralized Planning for Self-Adaptation in Multi-cloud Environment, pp. 76–90. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-14886-1_9

[30] D. Barbagallo, E. Di Nitto, D. J. Dubois, and R. Mirandola, *A Bio-inspired Algorithm for Energy Optimization in a Self-organizing Data Center*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 127–151. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-14412-7_7

[31] M. Sedaghat, F. Hernández-Rodriguez, E. Elmroth, and S. Girdzijauskas, "Divide the task, multiply the outcome: Cooperative vm consolidation," in *Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on*. IEEE, 2014, pp. 300–305.

[32] T. Sun, Y. Xu, and Q. He, "Improving asynchronous search for distributed generalized assignment problem," in *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2012 IEEE/WIC/ACM International Conferences on*, vol. 2, Dec 2012, pp. 38–42.

[33] D. J. Dubois, G. Valetto, D. Lucia, and E. D. Nitto, "Mycocloud: Elasticity through self-organized service placement in decentralized clouds," in *2015 IEEE 8th International Conference on Cloud Computing*, June 2015, pp. 629–636.

[34] P. L. Snyder, R. Greenstadt, and G. Valetto, "Myconet: A fungi-inspired model for superpeer-based peer-to-peer overlay topologies," in *2009 Third IEEE International Conference on Self-Adaptive and Self-Organizing Systems*. IEEE, 2009, pp. 40–50.

[35] R. Kuntschke, M. Specht, M. van Amelsvoort, M. Wagler, M. Winter, and R. Witzmann, "Economic optimization in virtual power plants vs. stable grid operation – bridging the gap," in *2015 IEEE 20th Conference on Emerging Technologies Factory Automation (ETFA)*, Sept 2015, pp. 1–5.

[36] D. P. Hans Kellerer, Ulrich Pferschy, *Knapsack Problems*. Springer-Verlag Berlin Heidelberg, 2004.

[37] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, 1957. [Online]. Available: http://www.jstor.org/stable/2098689

[38] L. Fleischer, M. X. Goemans, V. S. Mirrokni, and M. Sviridenko, "Tight approximation algorithms for maximum general assignment problems," in *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm*, ser. SODA '06. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2006, pp. 611–620. [Online]. Available: http://dl.acm.org/citation.cfm?id=1109557.1109624

[39] M. Jelasity, W. Kowalczyk, and M. Van Steen, "Newscast computing," Technical Report IR-CS-006, Vrije Universiteit Amsterdam, Department of Computer Science, Amsterdam, The Netherlands, Tech. Rep., 2003.

[40] S. Voulgaris, D. Gavidia, and M. Van Steen, "Cyclon: Inexpensive membership management for unstructured p2p overlays," *Journal of Network and Systems Management*, vol. 13, no. 2, pp. 197–217, 2005.

[41] M. Jelasity, R. Guerraoui, A.-M. Kermarrec, and M. Van Steen, "The peer sampling service: Experimental evaluation of unstructured gossip-based implementations," in *Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware*. Springer-Verlag New York, Inc., 2004, pp. 79–98.

[42] R. Bakhshi, D. Gavidia, W. Fokkink, and M. Van Steen, "An analytical model of information dissemination for a gossip-based protocol," *Computer Networks*, vol. 53, no. 13, pp. 2288–2303, 2009.

[43] R. Bruno, M. Conti, M. Mordacchini, and A. Passarella, "An analytical model for content dissemination in opportunistic networks using cognitive heuristics," in *Proceedings of the 15th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems (MSWiM 2012)*. ACM, 2012, pp. 61–68.

[44] D. Chakrabarti, J. Leskovec, C. Faloutsos, S. Madden, C. Guestrin, and M. Faloutsos, "Information survival threshold in sensor and p2p networks," in *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*. IEEE, 2007, pp. 1316–1324.

[45] A. Montresor and M. Jelasity, "Peersim: A scalable p2p simulator," in *Peer-to-Peer Computing, 2009. P2P'09. IEEE Ninth International Conference on*. IEEE, 2009, pp. 99–100.

[46] I. S. Moreno, P. Garraghan, P. Townend, and J. Xu, "An approach for characterizing workloads in google cloud to derive realistic resource utilization models," in *Service Oriented System Engineering (SOSE), 2013 IEEE 7th International Symposium on*. IEEE, 2013, pp. 49–60.

[47] E. Carlini, L. Ricci, and M. Coppola, "Flexible load distribution for hybrid distributed virtual environments," *Future Generation Computer Systems*, vol. 29, no. 6, pp. 1561–1572, 2013.

[48] R. Cohen, L. Katzir, and D. Raz, "An efficient approximation for the generalized assignment problem," *Information Processing Letters*, vol. 100, no. 4, pp. 162–166, 2006.