Optimized Deployment Plans for Platform as a Service Clouds

Benjamin Byholm
Åbo Akademi University
Department of Information Technologies
Turku, Finland
benjamin.byholm@abo.fi

Ivan Porres
Åbo Akademi University
Department of Information Technologies
Turku, Finland
ivan.porres@abo.fi

ABSTRACT

We approximately solve the problem of computing deployment plans of multiple cloud services with soft real-time constraints. We recognize that this is a generalized bin packing problem with fragmentable items. We formalize the problem domain and develop an autonomous deployment planning system with soft real-time constraints. The system incorporates a genetic algorithm with quadratic worst-case time complexity for approximately solving the packing problem, providing a service deployment plan with an optimal number of servers and an approximately optimal number of service instances.

KEYWORDS

Cloud computing, Middleware, Packing and covering problems

1 INTRODUCTION

We approximately solve the $\mathcal{NP}$-hard problem of computing deployment plans for multiple cloud services by presenting approximate and heuristic algorithms that can operate under soft real-time constraints. We wish to find an optimal assignment of services to servers, minimizing the number of servers and service instances, as service instances incur overhead through memory use and complex management, while running servers cost money and energy. We also wish to do this under soft real-time constraints, due to the high volatility of many Internet-scale applications.

We target a platform as a service (PaaS) utility model where the cloud provider offers a computing platform with automatic resource management. This is in contrast to an infrastructure as a service (IaaS) model, where the basic offering is the virtual machine (VM). In the IaaS model, a VM must not require more computing resources than what is provided by a physical machine (PM). In both models, the cloud provider allocates multiple VMs to a single PM and tries to keep its utilization within allowable limits in order to leverage its hardware investment. Commercial PaaS providers include Amazon Elastic Beanstalk, Heroku, Google AppEngine, Microsoft Azure and Oracle Cloud Platform, while commercial IaaS providers include Amazon EC2, Google Compute Engine, Microsoft Azure and Oracle Cloud Infrastructure. We mention Microsoft Azure as both an IaaS provider and a PaaS provider, since it offers both IaaS and PaaS solutions under the same brand.

UCC'17 Companion, December 5–8, 2017, Austin, TX, USA

1.1 IaaS versus PaaS

The distinction between IaaS and PaaS is somewhat blurred [1], but the traditional view of PaaS is that it provides customers access to elastic computing resources without the need for explicit management [20]. The PaaS model suits services which can be deployed in multiple VMs, resulting in increased reliability and performance, since the performance and reliability requirements of a service may exceed that which is offered by a single physical server.

The platform in PaaS can be simply regarded as middleware, which any IaaS user also can leverage to deploy a service in multiple VMs and then use a scaling and load balancing mechanism to distribute the work among dynamically allocated VMs. Indeed, many scholars have studied the problem of horizontal scaling from the perspective of the IaaS user [2, 3, 17–20]. A common theme in these works is that cloud providers leverage existing IaaS clouds to deploy services in combination with an auto-scaling mechanism for deciding how many VMs a given service will require at any given moment, either now or in the future, as well as a load balancer for distributing the load between these VMs. In this way, cloud providers obtain the benefits of a PaaS cloud while using a basic IaaS cloud, giving greater control to the downstream provider. However, from a holistic perspective, it is actually beneficial to leave the responsibility of elastic scaling and resource management to the upstream provider, which can optimize the entire system globally.

For example, assume that a data center hosts three services (a, b and c), from three different customers in three servers. Each server handles 500 million instructions per second (MIPS). The demand is 600 MIPS for service a, 100 MIPS for service b and 300 MIPS for service c. Since the resource demand for service a exceeds the capacity of a server, the downstream provider deploys the service in two VMs: a_1 and a_2, requiring 300 MIPS each (horizontal scaling), using a load-balancing mechanism to spread the requests evenly between the VMs. Table 1a shows the resulting deployment plan, visualized in fig. 1a, which shows that each server is underutilized, partially due to the policy that the load associated with service a is evenly distibuted among its two instances. The downstream provider cannot know that this situation is suboptimal. However, if we take into account the computing needs of all services in the data center, we can find a better allocation of VMs to servers and consolidate the services in only two servers, as shown in table 1b, visualized in fig. 1b. This optimal solution cannot be obtained if the upstream provider is bound by the decisions made by the downstream provider. Only the upstream provider knows the complete picture, so by leaving resource management to the upstream provider, a better outcome is achievable. This leads to lower operational costs for the upstream provider and a lower environmental footprint for all involved entities. The upstream provider can in turn reduce its prices, gaining an

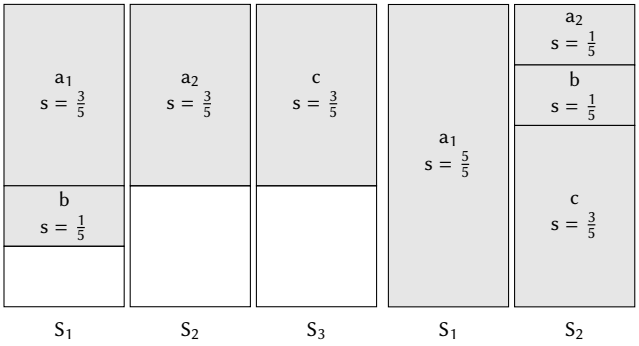
Table 1: Example of deployment plans

(a) Fixed-size plan

	Server		
Service	S_1	S_2	S_3
a	3/5	3/5	
b	1/5		
c			3/5

(b) Variable-size plan

	Server	
Service	S_1	S_2
a	5/5	1/5
b		1/5
c		3/5



(a) Fixed-size services. **(b) Variable-size services.**

Figure 1: Example of service deployment.

edge over its competitors, also benefiting the downstream provider.

1.2 Planning Algorithms

Due to the high volatility in the load of many Internet-scale applications, the algorithms involved in deployment planning must be fast and produce good solutions before they become irrelevant due to changed premises. That is, they must operate under soft real-time constraints. The algorithms must cope with large clusters consisting of millions of services and servers without spending too much resources on producing the deployment plans, since they are merely means to an end and do not possess any intrinsic value.

Several authors have used classic bin packing (BP) as the underlying model for such algorithms, e.g. the approaches proposed by Beloglazov et al. [3] and Gabay and Zaourar [8]. However, algorithms based on classic BP must always produce deployment plans using unaltered services with fixed size, as shown in fig. 1a, since they cannot alter the sizes of the VMs which are to be deployed.

For this reason, we have devised an approach based on fragmentable items bin packing (FIBP), which is a generalization of classic BP and can produce variable-size deployment plans, as shown in fig. 1b. This model enables full utilization of an optimal number of servers, as opposed to a classic BP model, which leaves many servers underutilized. It naturally incorporates services too large to fit in any one server and is approximately solvable under soft real-time constraints, as we will show in the rest of this paper.

This is the main contribution of this article: We present a formal definition of the problem of deployment planning for cloud services in a PaaS context, as well as planning algorithms based on FIBP that work under soft real-time constraints.

2 PRELIMINARIES

In this section we describe the problem domain of service deployment planning, as well our models for cloud servers and computing resources. The main concepts are servers, services and containers.

2.1 Problem Domain

Without loss of generality, consider a set of services $\mathbf{S}$ and a set of servers $\mathbf{M}$. A server $m \in \mathbf{M}$ provides concurrent computing resources to a set of containers $\mathbf{C}_m \subseteq \mathbf{C}$, which implement services $s \in \mathbf{S}$. A container isolates a service instance and provides access to a guaranteed amount of its server's resources. At least $k_s \geq 0$ containers implement any given service $s \in \mathbf{S}$. The size of a container is limited by the resources of its server, taking into account other possible containers on the same server.

2.2 Services

Every service $s \in \mathbf{S}$ requires a nonzero amount of central processing unit (CPU), measured in MIPS. In this paper, we assume that CPU is the bottleneck resource of the system. Determining the resource requirements of services is out of scope for this paper, but there are several options, e.g. deriving an empirical performance model at runtime, constructing an analytic model or directly measuring the system [2].

2.3 Containers

Services are hosted in containers $c \in \mathbf{C}$, which become service instances. A service may use multiple containers. Our approach is agnostic with respect to the underlying technology used for containers. It works for VMs, Docker containers, sandboxes, policy groups, etc. The only two requirements that we impose on such technologies is that it is possible to deploy and concurrently run multiple containers in the same server and that it is possible to adjust the server resources allocated to each container.

2.4 Servers

A server $m \in \mathbf{M}$ hosts a set of containers $\mathbf{C}_m \subseteq \mathbf{C}$ which implement services. A container $c \in \mathbf{C}$ may only be assigned to a single server $m \in \mathbf{M}$. Containers reserve a dedicated amount of their servers' resources. Since each server has a finite amount of resources, it can only host a finite number of containers, depending on their sizes. Servers can be homogeneous or heterogeneous, i.e. all servers may be identical, or their capacities may vary.

2.5 Resources

Resources are discrete quantities. To make the problem tractable, we must quantize resource supply and demand at some resolution. While we might waste some capacity when quantizing resources, there exists a quantum ε_r for which all smaller quanta drown in noise and overhead dominates. Hence, we do not consider this a problem in practice, particularly since cloud systems have high volatility and we can pick an arbitrarily small quantization error. Finer granularity, i.e. smaller quantization error, may result in more containers, since the likelihood of finding a set of services that fill

a server decreases. Performance may also suffer if all container resources are consumed by overhead. However, this can always be alleviated by using more servers, since there is less need to fill them completely. The cloud provider determines the quantization.

Quantization works the same for the homogeneous and the heterogeneous cases: Suppose three servers offer 1200 MIPS, 400 MIPS and 300 MIPS respectively. We also have two services, requiring 1000 MIPS and 800 MIPS respectively. Let us decide to quantize at a resolution of 100 MIPS per unit. This gives 1200 MIPS/100 MIPS = 12, 400 MIPS/100 MIPS = 4 and 300 MIPS/100 MIPS = 3 for the servers. Similarly, the two services yield 1000 MIPS/100 MIPS = 10 and 800 MIPS/100 MIPS = 8. We wish to find an allocation of services to servers with the fewest containers and servers, since containers introduce overhead and complex routing requirements, while running servers cost money and energy.

3 PROPOSED SOLUTION

A deployment planner finds a good deployment of service instances to servers. This should be done under soft real-time constraints, so that obtained plans can be realized before they become outdated.

3.1 Packing Services in Containers and Servers

We represent the problem of assigning services to containers and containers to servers as an instance of the minimum fragmentable items bin packing (Min-FIBP) problem, as presented by Byholm and Porres [4]. FIBP is a generalization of the classic, $\mathcal{NP}$-complete BP problem, which permits cutting items into smaller fragments. In the optimization version of classic BP, we are given n items (services) and n bins (servers) and produce an assignment $\boldsymbol{X}$ of items to bins. Each bin i has integer capacity c and item j requires w_j capacity. The objective (1a) is to minimize the number of used bins z, with the constraint that the total capacity required by the items in each bin does not exceed c (1b). Every item must be assigned to some bin (1c), and the decision variables are boolean, (1d) and (1e). A possible formulation of classic BP is [14]:

$$\min_{z} \quad z = \sum_{i=1}^{n} y_i \tag{1a}$$

$$\text{subject to} \quad \sum_{j=1}^{n} w_j x_{ij} \leq c y_i, \ \forall i \in \mathbf{N} = \{1, \ldots, n\}, \tag{1b}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \ \forall j \in \mathbf{N}, \tag{1c}$$

$$y_i \in \mathbb{B}, \ \forall i \in \mathbf{N}, \tag{1d}$$

$$x_{ij} \in \mathbb{B}, \ \forall i, j \in \mathbf{N}. \tag{1e}$$

Min-FIBP is the optimization version of FIBP. Since cutting items is permitted, we always know how many servers are required in the homogeneous case. In fact, FIBP even works with a heterogeneous set of servers. The objective in Min-FIBP is to minimize the number of fragments for given sets of items and bins. A solution with no cuts is equivalent to a classic BP solution.

The problem comprises an assignment $\boldsymbol{X}$ of services $s \in \mathbf{S}$ to servers $m \in \mathbf{M}$. The objective (2a) is to minimize the number of fragments, i.e. integer entries χ_{sm} greater than zero. The first constraint (2b) is that each service $s \in \mathbf{S}$ must be fully assigned to some servers $m \in \mathbf{M}$, so the sizes of all containers of service s must add up to its quanta $\psi(s)$. The second constraint (2c), states that each server $m \in \mathbf{M}$ has a capacity $\Psi(m)$ we cannot exceed. The third constraint (2d) restricts containers to multiples of unit-sized parts, so the size of each container must be a natural number:

$$\min_{\boldsymbol{X}} \quad \sum \mathbb{1}_{\boldsymbol{X} > 0} \tag{2a}$$

$$\text{subject to} \quad \sum_{m \in \mathbf{M}} \chi_{sm} = \psi(s), \ \forall s \in \mathbf{S}, \tag{2b}$$

$$\sum_{s \in \mathbf{S}} \chi_{sm} \leq \Psi(m), \ \forall m \in \mathbf{M}, \tag{2c}$$

$$\chi_{sm} \in \mathbb{N}, \ \forall (s, m) \in \mathbf{S} \times \mathbf{M}. \tag{2d}$$

We approximately solve the Min-FIBP problem using a grouping genetic algorithm [4]. The grouping genetic algorithm yields a guaranteed 5/4-approximation with complexity $O(|\mathbf{S}|^2)$ in the worst case. After constructing the initial population, the grouping genetic algorithm proceeds to improve the population through grouping crossover and mutation until it finds an optimal solution, stagnates or reaches the maximum number of generations. The length of this phase is variable and the termination condition can be set to a predefined time limit, a fixed number of iterations, or a convergence criterion. In this way, we have a parametrizable trade-off between solution quality and execution time.

3.2 Why FIBP is a Better Model than BP

FIBP is a strictly better model than BP for deployment planning in cloud computing. We shall give three primary reasons for this:

Firstly, in the FIBP model, we always know the minimum number of servers required to deploy the services. In the BP model, we do not know if a deployment of a given set of services is possible with a given set of servers. A natural lower bound on the number of servers necessary for BP is the linear relaxation [14]:

$$L_1 = \left\lceil \sum_{s \in \mathbf{S}} \frac{\psi(s)}{c} \right\rceil. \tag{3}$$

The FIBP model is always solvable with this number of identical servers, while the BP model usually requires more servers.

Secondly, the FIBP model supports heterogeneous servers in addition to homogeneous servers. Although a single data center might well use homogeneous servers, this is unlikely to extend to multiple independent data centers or cloud providers. Moreover, if heterogeneous servers are permitted, a data center can upgrade its hardware in stages, without having to shut down entirely for a full upgrade of the server pool. Thus, support for heterogeneous servers can be highly beneficial for all involved parties.

Thirdly, the FIBP model supports services that cannot be deployed in a single server, while the BP model does not. Several Internet-scale applications offer services that require far more computational resources than what can be offered by a single server.

Thus, the FIBP model is strictly superior to the BP model for deployment planning in cloud computing. We shall further illustrate these points with examples and experiments.

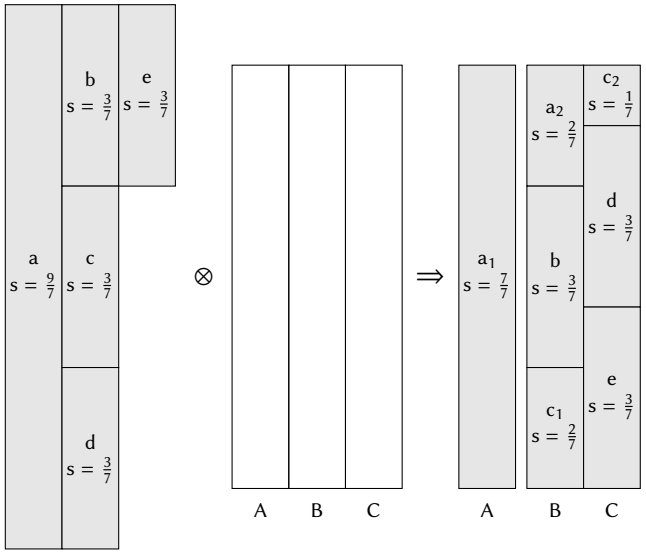


Figure 2: Five services are deployed in three servers. Two services are deployed into two containers each.

3.3 Example

We want to find a good configuration for five different services: a, b, c, d and e. Each service requires some amount of CPU. Service a requires 900 MIPS, while each of services b, c, d and e require 300 MIPS. Each server has a capacity c of 700 MIPS. Together, all services require 2100 MIPS, so $L_1 = \lceil 2100\,\text{MIPS}/700\,\text{MIPS} \rceil = 3$. For FIBP, this is the minimum number of servers necessary to host the services and is always achievable.

Figure 2 depicts the example. Service a requires more capacity than any server can offer, so it must operate in more than one container on more than one server. Preprocessing in the Min-FIBP solver will cut it into one container of 700 MIPS and another of 200 MIPS. The larger fragment will be placed in its own, dedicated server. The remaining fragment goes to another server, together with service b. At this point, said server has 200 MIPS to spare, but none of the remaining services fit in 200 MIPS. Thus, one of them (service c in this example) is cut into two fragments of 200 MIPS and 100 MIPS respectively. The larger fragment fills server B, while the smaller goes in the last server, together with the remaining services. All services have now been deployed in only three servers.

Contrast this elegant solution to that of the classic BP model. As is, the problem has no solution, since service a does not fit in any bin. If permitted to redefine the problem, we might replace service a with two services: a_1 and a_2. Let us also assume that we again make a_1 require 700 MIPS and a_2 require 200 MIPS. Then, a_1 is again placed in a dedicated server, while a_2 goes in server B. All remaining services require 300 MIPS each, so server B can fit one of them, but must leave 200 MIPS unused. Server C fits two more services, leaving 100 MIPS unused. One service has still not been placed, so we require a new server, D, which will have 400 MIPS unused. Now all services have been placed, but it required one server more yet left 700 MIPS, i.e. an entire server's worth, unused. This is significant waste, despite the BP packing being optimal.

3.4 Heterogeneous Servers

As mentioned in sections 2.4 and 3.1, the FIBP model also supports heterogeneous servers, since heterogeneous problem instances can be polynomially reduced to homogeneous instances [4]. As explained in section 2.5, resources provided by servers and resources required by services must be quantized at a given resolution.

The three example servers mentioned in section 2.5 offer a total of 19 CPU units. We pick a value larger than this, say 20, and define this as the unique capacity c for each of the three servers. We then proceed by adding a dummy service requiring $c - \Psi(m_i)$ units for each original server $m_i \in \mathbf{M}$ to the set of services $\mathbf{S}$, that is we add three dummy services requiring $20 - 12 = 8$, $20 - 4 = 16$ and $20 - 3 = 17$ CPU units respectively. After solving the transformed Min-FIBP problem, we simply remove the dummy services and shrink the servers to their original sizes to undo the transformation and proceed with deploying the services to their servers.

4 EXPERIMENT

This section describes an experimental comparison of our approach and a classic BP approach. Unlike our approach, solutions based on classic BP cannot directly support services that require more capacity than the one provided by a single server or servers with heterogeneous capacity. Therefore we focus on homogeneous servers where no service requires more capacity than a server can offer.

4.1 Design

We generated ten random problems with uniform distribution of service sizes and attempted to solve them using our approach and CPLEX. CPLEX is an award-winning, state of the art solver for mathematical programming, widely used in industry and academia. Each problem involved 256 services. Server capacity was quantized at a resolution of 1/32. Each service needed between 1 and 32 units of CPU capacity, inclusively. We consider these problems to be very small, but this is required to allow CPLEX to produce good solutions in a reasonable amount of time, since the branch-and-cut methods used by CPLEX for mixed integer linear programming (MILP) problems require time exponential in the size of the problem.

We conducted two experiments with the generated problem set. The first experiment sought to compare the applicability of FIBP and classic BP in terms of number of servers required in a cloud computing context. As explained in section 3.2, the BP model always requires at least as many servers as the minimum number of servers needed for the FIBP model, usually more, since the classic BP model does not permit cutting items into smaller fragments. For this reason, we also restricted the maximum service size in the generated problems to fit in a single server, since the resulting problems would otherwise be unsolvable under the classic BP model. Additionally, the classic BP model does not support heterogeneous server configurations, which is why we limited ourselves to homogeneous servers.

The second experiment sought to compare the speed and quality of our solver to that of CPLEX under the FIBP model. Since CPLEX is an exact solver and Min-FIBP is strongly $\mathcal{NP}$-hard, it requires computation time exponential in the size of the problem to find and verify that a candidate solution truly is optimal. However, CPLEX does produce valid solutions throughout its execution, so

Table 2: Servers required by the FIBP and BP models. The results of the BP model are approximate

FIBP	132	134	140	130	130	134	133	129	127	136
BP	136	138	143	134	133	138	136	131	129	140

Table 3: Containers used by our solver versus CPLEX

Our	264	262	267	262	262	271	265	261	259	268
CPLEX	350	348	357	344	344	350	343	349	339	341

by limiting its execution with a time budget, we should hopefully produce reasonably good solutions. Since we are operating under soft real-time constraints, it is not reasonable to spend days on computing solutions. It should be possible to produce reasonably good solutions in less than 10 ms for problems of this size. However, already the presolving step used by CPLEX to reduce input problems before commencing with the main optimization actually requires two orders of magnitude more time than this. For this reason, we generously awarded CPLEX a time budget of 100 s to solve each problem for both experiments.

Both experiments were conducted on a regular desktop computer equipped with an Intel Core i7-4770 CPU running at 3.4 GHz and 16 GiB of random access memory (RAM). The processor has 4 physical CPU cores with Hyper-threading, giving 8 logical cores. CPLEX is fully multi-threaded and used all 8 logical cores, while our solver is single-threaded and only used 1 logical core.

4.2 Results

Table 2 shows the results from the first experiment, which sought to compare the applicability of the FIBP model and the BP model. As we can see, the FIBP model used strictly fewer servers than the BP model for every problem. CPLEX did not manage to conclude that any obtained solution for BP was optimal before exhausting the time limit of 100 s. On the other hand, the results for the FIBP model were instantly computed using eq. (3).

Table 3 shows the results from the second experiment, which sought to compare the performance and quality of our solver versus CPLEX under the FIBP model. As we can see, the solutions produced by CPLEX consistently required 30 % more containers than those produced by our solver. Table 4 shows the computation time required for both solvers. CPLEX only terminated because it exceeded the preset time limit of 100 s. We would again like to stress the point that 100 s is far too much time for solving the problems under soft real-time constraints.

4.3 Analysis

Table 2 verifies that the FIBP model allows using fewer servers than the BP model for deploying a given set of services. We have thus experimentally demonstrated the claims in section 3.2.

From table 4, we can tell that problem 0 and problem 7 were solved to optimality while generating the initial population, and that the solver successfully determined that they were optimal.

Table 4 combined with table 3 tells us that CPLEX required five orders of magnitude more computation time to produce solutions which were 30 % worse than those produced by our solver. All but problem 4 were solved to optimality by our solver. We had previously determined that this problem requires only 261 containers.

Table 4: FIBP runtime of CPLEX and our solver

	Time (ms)	
Problem #	CPLEX	Our
0	99690	0.02
1	99690	2.64
2	99680	1.09
3	99700	2.85
4	99810	1.96
5	99720	2.14
6	99810	2.72
7	99730	0.02
8	99700	1.57
9	99660	3.19

5 RELATED WORK

The classic BP problem is a well-known combinatorial optimization problem, first introduced by Johnson et al. [11]. The decision form of BP is strongly $\mathcal{NP}$-complete [9]. The classic BP problem has many variants, one of which is FIBP. The FIBP problem was recently formalized by LeCun et al. [12]. While there have been a couple earlier variants of BP dealing with fragmentable items [13, 15], they have not explicitly attempted to minimize the number of fragments for given sets of items and bins. LeCun et al. [12] also presented approximation algorithms for a special case of minimum fragmentable items bin packing with equal capacities (Min-FIBP-EQ), which we generalized in addition to reducing their computational complexities [4]. In addition to FIBP, Casazza and Ceselli [5] also studied other variants of BP with fragmentation and presented algorithms for exactly solving these problems. However, since FIBP is strongly $\mathcal{NP}$-complete [12], any deterministic, exact algorithm requires time exponential in the size of the input, unless $\mathcal{P} = \mathcal{NP}$. The problems studied by Casazza and Ceselli [5] involve no more than 100 items and require several minutes or hours to solve, which means that they are unsuitable for applications with real-time constraints and even moderately large problem sizes.

Our Min-FIBP solver [4] combines fast approximation algorithms with a metaheuristic known as a grouping genetic algorithm, inspired by the work of Quiroz Castellanos et al. [16] for classic BP. The grouping genetic algorithms form a family of genetic algorithms different from the traditional Holland-style genetic algorithms [10], more suitable for grouping problems. They were first introduced by Falkenauer [6] 25 years ago. Falkenauer [7] found that a certain form of grouping genetic algorithms produce results far better than those obtained from their underlying heuristics in isolation and the work of Quiroz Castellanos et al. [16] is a further refinement thereof. Our Min-FIBP solver [4] is the first application of a grouping genetic algorithm to Min-FIBP.

BP is an intuitive model for service deployment with several variations. For this reason, many scholars have studied service deployment and consolidation under these models. However, we

have not seen any previous works using FIBP, which, as we have shown, is strictly superior to classic BP.

Beloglazov et al. [3] used a variable size BP model to manage data centers in an energy-aware and efficient manner. Gabay and Zaourar [8] used a variable-size vector bin packing model. However, as we have shown in this paper, a model such as FIBP, where services can be divided among several containers naturally allows for denser packings and can be solved much quicker. As noted by Gabay and Zaourar [8], problems with heterogeneous bins can be reduced to problems with uniform bin sizes. Similarly, instances of Min-FIBP can be reduced to equivalent instances of Min-FIBP-EQ. However, Gabay and Zaourar [8] make a fair assessment that this may lead to complications when dealing with multiple resources. This aspect should be closely investigated in possible generalizations of FIBP to multiple dimensions.

Zabolotnyi et al. [20] compared IaaS and PaaS solutions and presented a middleware solution called JCloudScale for the purpose of providing the convenience of PaaS in an IaaS environment. Zabolotnyi et al. [20] claim that one of the problems with PaaS offerings is limited customer control over the elasticity behavior of deployed applications. Utilization of physical servers was not reported in the experiments. We take the opposite point of view and argue that this property is, in fact, a strength of PaaS, since the provider has the best possible information about the status of the cluster and can perform global optimization of the entire system, as opposed to local optimization by individual actors.

6 CONCLUSION AND FUTURE WORK

We presented a new take on the $\mathcal{NP}$-hard problem of computing deployment plans for multiple cloud services. We recognized that the FIBP model is a better fit than the classic BP model, since it enables using fewer servers, supports heterogeneous server configurations and copes with Internet-scale services too large to fit in any single server. This is the first work in cloud computing based on FIBP, which enables denser packings with less wasted capacity. Less waste saves both money and energy.

Future work involves a practical generalization of FIBP to multiple dimensions, allowing us to efficiently handle multiple resources. We are currently working on a combined migration and consolidation algorithm, which can transition a system from an old deployment to a new deployment plan, minimizing the amount of data that must be migrated under soft real-time constraints. A further extension is support for incompatibility constraints, where two or more containers cannot reside in the same server, e.g. because of redundancy concerns, if they implement the same service, or due to possible interference between two containers of highly different natures. We look forward to seeing new works based on FIBP in the field of cloud computing in the future.

ACKNOWLEDGMENTS

Benjamin Byholm received scholarships from the Nokia Foundation and the Finnish Foundation for Technology Promotion (TES).

REFERENCES

[1] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. 2010. A View of Cloud Computing. *Commun. ACM* 53, 4 (April 2010), 50–58. https://doi.org/10.1145/1721654.1721672

[2] Adnan Ashraf, Benjamin Byholm, and Ivan Porres. 2016. Prediction-Based VM Provisioning and Admission Control for Multi-Tier Web Applications. *Journal of Cloud Computing* 5, 1 (2016), 15. https://doi.org/10.1186/s13677-016-0065-9

[3] Anton Beloglazov, Jemal Abawajy, and Rajkumar Buyya. 2012. Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing. *Future Generation Computer Systems* 28, 5 (2012), 755–768. https://doi.org/10.1016/j.future.2011.04.017

[4] Benjamin Byholm and Ivan Porres. 2017. *Fast Algorithms for Fragmentable Items Bin Packing.* Technical Report 1181. TUCS.

[5] Marco Casazza and Alberto Ceselli. 2016. Exactly Solving Packing Problems with Fragmentation. *Computers & Operations Research* 75, C (Nov. 2016), 202–213. https://doi.org/10.1016/j.cor.2016.06.007

[6] Emanuel Falkenauer. 1992. The grouping genetic algorithms — widening the scope of the GAs. *Belgian Journal of Operations Research, Statistics & Computer Science* 33, 1 (1992), 2.

[7] Emanuel Falkenauer. 1996. A hybrid grouping genetic algorithm for bin packing. *Journal of Heuristics* 2, 1 (1996), 5–30. https://doi.org/10.1007/BF00226291

[8] Michaël Gabay and Sofia Zaourar. 2016. Vector bin packing with heterogeneous bins: application to the machine reassignment problem. *Annals of Operations Research* 242, 1 (2016), 161–194. https://doi.org/10.1007/s10479-015-1973-7

[9] Michael Randolph Garey and David Stifler Johnson. 1990. *Computers and Intractability.* W. H. Freeman & Co., New York, NY, USA.

[10] John Henry Holland. 1992. *Adaptation in Natural and Artificial Systems.* MIT Press, Cambridge, MA, USA.

[11] David Stifler Johnson, Alan John Demers, Jeffrey David Ullman, Michael Randolph Garey, and Ronald Lewis Graham. 1974. Worst-Case Performance Bounds for Simple One-Dimensional Packing Algorithms. *SIAM J. Comput.* 3, 4 (1974), 299–325. https://doi.org/10.1137/0203025

[12] Bertrand LeCun, Thierry Mautor, Franck Quessette, and Marc-Antoine Weisser. 2015. Bin packing with fragmentable items: Presentation and approximations. *Theoretical Computer Science* 602 (2015), 50–59. https://doi.org/10.1016/j.tcs.2015.08.005

[13] Chittaranjan A. Mandal, Partha Pratim Chakrabarti, and Sujoy Ghose. 1998. Complexity of fragmentable object bin packing and an application. *Computers & Mathematics with Applications* 35, 11 (1998), 91–97. https://doi.org/10.1016/S0898-1221(98)00087-X

[14] Silvano Martello and Paolo Toth. 1990. *Knapsack Problems.* John Wiley & Sons, Chichester, England.

[15] Nir Menakerman and Raphael Rom. 2001. Bin Packing with Item Fragmentation, Frank Dehne, Jörg-Rüdiger Sack, and Roberto Tamassia (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 313–324. https://doi.org/10.1007/3-540-44634-6_29

[16] Marcela Quiroz Castellanos, Laura Cruz Reyes, José Torres Jiménez, Claudia Gómez Santillán, Héctor Joaquín Fraire Huacuja, and Adriana Cesário de Faria Alvim. 2015. A grouping genetic algorithm with controlled gene transmission for the bin packing problem. *Computers & Operations Research* 55 (2015), 52–64. https://doi.org/10.1016/j.cor.2014.10.010

[17] Weijia Song, Zhen Xiao, Qi Chen, and Haipeng Luo. 2014. Adaptive Resource Provisioning for the Cloud Using Online Bin Packing. *IEEE Trans. Comput.* 63, 11 (Nov. 2014), 2647–2660. https://doi.org/10.1109/TC.2013.148

[18] Andreas Wolke, Boldbaatar Tsend-Ayush, Carl Pfeiffer, and Martin Bichler. 2015. More than bin packing: Dynamic resource allocation strategies in cloud data centers. *Information Systems* 52 (2015), 83–95. https://doi.org/10.1016/j.is.2015.03.003

[19] Timothy Wood, Prashant Shenoy, Arun Venkataramani, and Mazin Yousif. 2009. Sandpiper: Black-box and gray-box resource management for virtual machines. *Computer Networks* 53, 17 (2009), 2923–2938. https://doi.org/10.1016/j.comnet.2009.04.014

[20] Rostyslav Zabolotnyi, Philipp Leitner, Waldemar Hummer, and Schahram Dustdar. 2015. JCloudScale: Closing the Gap Between IaaS and PaaS. *ACM Trans. Internet Technol.* 15, 3, Article 10 (July 2015), 20 pages. https://doi.org/10.1145/2792980