# Energy-Aware Ant Colony Based Workload Placement in Clouds

Eugen Feller*, Louis Rilling†, Christine Morin*
*INRIA Centre Rennes - Bretagne Atlantique
Campus universitaire de Beaulieu, 35042 Rennes Cedex, France
{Eugen.Feller, Christine.Morin}@inria.fr
†Kerlabs
80 avenue des buttes de Coësmes, Bâtiment Germanium, 35700 Rennes, France
Louis.Rilling@kerlabs.com

***Abstract*—With increasing numbers of energy hungry data centers energy conservation has now become a major design constraint. One traditional approach to conserve energy in virtualized data centers is to perform workload (i.e., VM) consolidation. Thereby, workload is packed on the least number of physical machines and over-provisioned resources are transitioned into a lower power state. However, most of the workload consolidation approaches applied until now are limited to a single resource (e.g., CPU) and rely on simple greedy algorithms such as First-Fit Decreasing (FFD), which perform resource-dissipative workload placement. Moreover, they are highly centralized and known to be hard to distribute. In this work, we model the workload consolidation problem as an instance of the multi-dimensional bin-packing (MDBP) problem and design a novel, nature-inspired workload consolidation algorithm based on the Ant Colony Optimization (ACO). We evaluate the ACO-based approach by comparing it with one frequently applied greedy algorithm (i.e., FFD). Our simulation results demonstrate that ACO outperforms the evaluated greedy algorithm as it achieves superior energy gains through better server utilization and requires less machines. Moreover, it computes solutions which are nearly optimal. Finally, the autonomous nature of the approach allows it to be implemented in a fully distributed environment.**

***Keywords*-Ant Colony Optimization, Combinatorial Optimization, Green Cloud Computing, Multidimensional Bin Packing, Swarm Intelligence, Virtualization**

## I. Introduction

Cloud computing has recently evolved as a new computing paradigm which promises virtually unlimited resources. Customers can rent resources based on the pay-as-you-go model and thus are charged only for as much as they have used. Thereby, resources are transparently provisioned by the cloud provider according to the customers requirements. However, customers growing demands for computing power are now facilitating the cloud service providers (e.g., Amazon, Google, Microsoft, Yahoo!, etc.) to deploy increasing amounts of energy hungry data centers [15]. Consequently, energy costs for operating and cooling the equipment of such data centers have increased significantly up to a point where they are able to surpass the hardware acquisition costs. Studies have shown that data centers around the world consumed 201.8 terawatt hours (TWh) in 2010. This is enough energy to power 19 million average U.S. households and results in approximately $23.3 billion spent for energy costs [2].

From the business perspective, reducing the energy consumption can lead to immense cost reductions. Moreover, besides the huge energy costs, heat dissipation increases inevitably with higher power consumption and doubles the probability of hardware failures [14]. Therefore, reducing the energy dissipation has a significant effect on the overall availability, reliability and productivity of a system. Not least, the way energy is generated influences our environment either directly by the carbon footprint or indirectly by the nuclear waste. Therefore, reducing the energy consumption does not only save a significant amount of money and improves the system reliability, but also helps protecting our environment. According to [16], data centers emit as much $CO_2$ as the whole Argentine, and will quadruple their $CO_2$ emissions by 2020.

Several approaches exist in order to conserve energy. Besides the possibility to replace the hardware with more energy-efficient one, reducing the energy wasted because of hardware over-provisioning is crucial. Today's data centers infrastructure is typically over-provisioned in order to sustain the service availability during periods of peak resource demand. However, resource demand in current data centers is usually of a bursty nature and thus results in a low average utilization of approximately 15-20% [26]. Therefore, a big fraction of the resources can be used to take energy conservation decisions such as suspending or turning off unnecessary servers, while still preserving the customers performance requirements. Several open-source cloud projects have been recently started to provide alternative solutions to public Infrastructure as a Service (IaaS) cloud providers (e.g., Amazon EC2). Examples of such cloud management frameworks include *Eucalyptus, OpenNebula, and Nimbus*.

Given that ubiquitous virtualization solutions are able to *live migrate* the workload (i.e., VMs) and servers can be *turned on and off* at any time, clusters can be turned into dynamic pools of resources by these frameworks. However, one of the main limitations of all current cloud management frameworks next to their high degree of centralization [13], is that they do not provide any advanced energy-aware workload consolidation policies.

Consolidation of virtual machines on the least number of physical nodes is an instance of the well known multi-dimensional bin-packing (MDBP) problem and has been mostly studied by means of simulations in several works (e.g., [8], [23]). Thereby, because of the NP-hard nature of the problem and the need to compute the solutions in a reasonable amount of time, approximation approaches (i.e., heuristics) have shown to provide good results. However, many of the existing approaches nowadays still: (1) *ignore the multi-dimensional character of the problem (e.g., [5], [19])*, (2) *adapt simple greedy algorithms (e.g., FFD)*, which tend to waste a lot of resources [21] and are *highly centralized*.

In this paper, we first accurately model the workload placement problem as an instance of the multi-dimensional bin-packing (MDBP) problem. We then take a *nature-inspired* approach derived from the behavior of real ants and propose a *novel* algorithm based on the Ant Colony Optimization (ACO) meta-heuristic to compute the placement dynamically according to the current load. We apply our algorithm on a number of synthetic test instances and compare it with one frequently applied greedy algorithm (i.e., FFD). The results indicate that the ACO-based algorithm outperforms the evaluated greedy approach as it computes workload placements with superior energy gains through better resource utilization and requires less machines. Moreover, by solving the model utilizing the IBM ILOG CPLEX [1] solver, we show that the solutions computed by our approach are *nearly optimal* (i.e., small deviation of 1.1%). To the best of our knowledge this is the first work to: (1) *apply ACO on the MDBP problem in the context of dynamic workload consolidation* and (2) *utilize ACO in order to conserve energy*.

The remainder of this paper is organized as follows. Section II gives a short introduction to the ACO. Section III provides a formal problem definition of the workload placement problem as an instance of the multi-dimensional bin-packing (MDBP) problem and introduces our workload resource demand estimation approach. Section IV details the design of the ACO-based workload consolidation algorithm. Section V presents the experimental results. Section VI discusses the related work. Finally, Section VII closes the paper with conclusions and future work.

## II. Ant Colony Optimization

Ant Colony Optimization (ACO) is a meta-heuristic, which was initially introduced as Ant Systems (AS) in 1992 within the PhD thesis of the Italian researcher Marco Dorigo [12]. Initially, it was developed to solve the Traveling Salesman Problem (TSP). However, since then it has been successfully adapted to solve many other complex combinatorial optimization problems (e.g., vehicle routing, quadratic assignment, dynamic job scheduling, graph coloring and bin packing).

The main inspiration to develop this system was the natural food-discovery behavior of real ants. Because of the limited abilities of the ants to see and hear their environment they have developed a form of indirect communications (also called Stigmergy) by use of a chemical substance referred as *pheromone*. This substance is deposited by each ant on the path it traverses and evaporates after a certain period of time. Other ants can smell the concentration of this substance and tend to favor paths probabilistically according to the amount of pheromone deposited on them. Surprisingly, after some time the entire ant colony converges towards the shortest path to the food source. This behavior was studied by biologists in numerous controlled experiments [11] and can be explained as follows. At the beginning, when starting from the nest the ants choose a random path to follow. However, on the shortest path to the food source the ants will return faster. Thereby, this path will have a stronger pheromone concentration thus being more attractive for subsequent ants to follow it. When time passes, pheromone concentration on the shortest paths will continue to increase, while on the longer ones it will keep falling, making them less and less attractive.

When applied on combinatorial optimization problems such as TSP or the Bin-Packing Problem (BPP), artificial ants act as a multi-agent system and construct a complex solution based on indirect low-level communication. Thereby, several parts of the algorithm need to be defined in order to imitate real ants. Similarly, as real ants do, a decision on which *path* or *item* to choose next needs to be taken. Therefore, a probabilistic decision rule has to be defined which will be used by the algorithm to guide the ants choice towards the optimal solution. Furthermore, unlike real ants, a memory is necessary for each ant, which will be used to keep track of the local solution constructed so far. Finally, a pheromone update mechanism is required in order to: (1) simulate pheromone evaporation, (2) deposit pheromone either on the visited paths (i.e., TSP) or on the selected item-bin pairs (i.e., BPP), respectively. Thereby, a decision needs to be taken on which ant will perform the pheromone updates. This can be either done *after each ants move*, by the *iterations best* ant, or the *best-so-far* ant. We will describe our design choices in Section IV.

## III. Formal problem definition and resource demand estimation

This section details the assumptions of this work and provides a formal definition of the workload consolidation problem as a multi-dimensional bin-packing (MDBP) problem. Thereby, a binary integer programming (BIP) representation of the problem is introduced. Afterwards, the workload resource demand estimation approach is presented.

### A. Assumptions

We assume a homogeneous environment in which all physical machines have the same capacity. Furthermore, the algorithm requires the knowledge about all the workload and its associated resource requirements in order to compute the placement. These resource requirements can be either seen as static or dynamic. In the static case it is assumed that a batch of VMs is submitted to the system and needs to be placed. Thereby, as no utilization information is available upon initial submission, given VMs resource requirements are considered static and VMs are scheduled according to this information.

On the other hand, when time passes (i.e., sufficiently long) history resource utilization becomes available and can be used to estimate the resource demand. In that case, VM resource requirements can be seen as the dynamic estimates of the maximum resources required by the VMs over the predefined monitoring interval (e.g., week). The proposed algorithm then takes those values as input and dynamically *overbooks* the resources when the workload resource demand permits it. Thereby, the resource utilization is optimized. Consequently, we assume that workload resource utilization can be measured over predefined periods of time $T$ (e.g., weeks) thus allowing the maximum workload resource demand to be estimated. Thereby, for the sake of simplicity the time $t$ at which the maximum resource demand values are computed is not mentioned in the following formal definitions and is assumed to be the same as on which the workload consolidation algorithm is triggered. Moreover, in order to minimize the amount of migrations and limit the degree of performance degradation the algorithm is assumed to be triggered after predefined, sufficiently long periods of time (e.g., weekly basis). More intelligent triggering decisions based on the analysis of workload characteristics (see [21]) are possible but go beyond the scope of this paper. Finally, despite the relative long measurement periods, *overbooking* of resources can lead to performance degradation when workload resource demands suddenly start to increase. It is assumed that such changes can be detected and handled by the appropriate algorithm [17].

### B. Formal Problem Definition

We define the problem of mapping the workload (i.e., VMs) to physical machines as an instance of the MDBP problem, in which the physical machines represent the bins and the workload the items to be packed. Each bin has a predefined static resource (e.g., CPU cycles, CPU cores, RAM size, network bandwidth and disk size) capacity vector and each item is assigned with one time-varying resource demand vector.

Let $B := \{B_0, \ldots, B_v, \ldots, B_{n-1}\}$ denote the set of bins and $I := \{0, \ldots, m-1\}$ the set of items, with $n = |B|$ and $m = |I|$ representing the amounts of bins and items, respectively. Furthermore, available resources (i.e., CPU cycles, CPU cores, RAM size, network bandwidth and disk size) are defined by the set $R$ with $d = |R|$.

Each bin $B_v$ is assigned with a predefined static homogeneous $d$-dimensional bin capacity vector $\vec{C}_v := (C_{v,1}, \ldots, C_{v,k}, \ldots C_{v,d})$, in which each component defines the bin's capacity of resource $k \in R$. Moreover, all item's $i \in I$ are represented by their time-varying $d$-dimensional resource demand vectors $\vec{r}_i := (\overline{r}_{i,1}, \ldots, \overline{r}_{i,k}, \ldots, \overline{r}_{i,d}) \in [0,1]^d$, with each component of the vector being the items maximum demand for resource $k \in R$ over the last measurement period $T$ (e.g., week) relative to the corresponding dimension in the static bin resource capacity vector $\vec{C}_v$. Thereby, without loss of generality we assume that the values of $\vec{C}_v$ have been normalized to 1 in the following definitions.

Finally, in order to complete our binary integer programming (BIP) model, we define the following two decision variables:

1) Bin allocation variable $y_v$, equals 1 if the bin $v$ is chosen, and 0 otherwise.
2) Item allocation variable $x_{i,v}$, equals 1 if the item $i$ is assigned to the bin $v$, and 0 otherwise.

The ultimate goal of the consolidation algorithm is then to place all items such that, the number of bins used is minimized. This is reflected in our objective function (1).

$$\text{Minimize} f(y) = \sum_{v=0}^{n-1} y_v \tag{1}$$

Subject to the following constraints:

$$\sum_{i=0}^{m-1} \overline{r}_{i,k} x_{i,v} \leq C_{v,k} y_v, \forall v \in \{0, \ldots, n-1\}, \forall k \in R \tag{2}$$

$$\sum_{v=0}^{n-1} x_{i,v} = 1, \; \forall i \in \{0, \ldots, m-1\} \tag{3}$$

Constraint (2) ensures that the capacity of each bin is not exceeded and constraint (3) guarantees that each item is assigned to exactly one bin.

### C. Workload Resource Demand Estimation

Given that workloads resource demands are of time varying nature, spikes in utilization are most likely to appear. In order to provide stable values to the workload consolidation algorithm and minimize the amount of performance degradation due to consolidation, workload demand estimations are needed. We base our estimations on the long-term workload resource utilization history (e.g., weeks, months). Thereby, workload resource utilization is captured at predefined measure points thus creating a data set composed out of discrete resource measures. These measures are then analyzed and workload resource demand $\overline{r}_{i,k}$ is estimated by taking the maximum value of the previous measures. These values are then used by the algorithm to perform the consolidation.

## IV. Energy-Aware Ant Colony Optimization Based Workload Consolidation

This section presents the design of our ACO meta-heuristic based algorithm, to solve the previously defined workload consolidation problem. Thereby, all parts of the algorithm are described and the pseudocode is presented. In the proposed algorithm each ant receives all items (i.e., VMs), opens a bin (i.e., physical machine) and starts assigning the items to the bin. This is achieved by the use of a probabilistic decision rule, which describes the desirability for an ant to choose a particular item as the next one to pack in its current bin. This rule is based on the information about the current pheromone concentration on the item-bin pair and a heuristic which guides the ants towards choosing the most promising items. Hence, the higher the amount of pheromone and heuristic information is associated with a particular item-bin pair, the higher the probability is that an ant will choose it. This stochastic nature of the algorithm allows the ants to explore a large number of potential solutions and thus compute better placements than the evaluated state-of-the-art greedy algorithm (see Section V).

Finally, after all ants have constructed their solutions, the amount of pheromone associated with each item-bin pair needs to be updated in order to simulate pheromone evaporation and reinforce item-bin pairs which belonged to the *better* solutions. Consequently, a pheromone update rule is defined.

### A. Probabilistic Decision Rule

We define the probability for an ant to choose an item $i$ as the next one to pack in its current bin $v$ as follows.

$$p_v^i := \frac{[\tau_{i,v}]^\alpha \times [\eta_{i,v}]^\beta}{\sum_{u \in N_v} [\tau_{u,v}]^\alpha \times [\eta_{u,v}]^\beta}, \quad \forall i \in N_v \tag{4}$$

whereby, $\tau_{i,v}$ denotes the pheromone based desirability of packing item $i$ into bin $v$ and $\eta_{i,v}$ the items heuristic information. Moreover, two parameters $\alpha$, $\beta \geq 0$ are used in order to either emphasize more the pheromone or the heuristic information. Finally, $N_v$ defines the set of all items which qualify for inclusion into the current bin $v$. Hence, those are all items which have not been assigned to any bin yet and do not violate the bin capacity constraints in all dimensions.

$$N_v := \{i \mid \sum_{j=0}^{n-1} x_{i,j} = 0 \wedge \vec{b}_v + \vec{r}_i \leq \vec{C}_v\} \tag{5}$$

Thereby, $\vec{b}_v$ is defined as the load vector of the bin $v$. It is computed as the sum of all item resource demand vectors assigned to the bin.

$$\vec{b}_v := \sum_{i \in B_v} \vec{r}_i \tag{6}$$

### B. Heuristic Information

As our objective is to minimize the number of machines (i.e., maximize the resource utilization), we define the heuristic information to favor items which utilize the bins better. This is achieved by defining $\eta_{i,v}$ as the inverse of the scalar valued difference between the static capacity of bin $v$ and the load of bin after packing the item $i \in N_v$.

$$\eta_{i,v} := \frac{1}{|\vec{C}_v - (\vec{b}_v + \vec{r}_i)|_1} \tag{7}$$

In order to compute the ratio defined by equation 7 the resulting $d$-dimensional resource demand vector needs to be mapped to a scalar value. Therefore, the L1-norm is used in this work. However, alternative methods such as taking the arithmetic mean are possible.

### C. Pheromone Trail Update

After all ants have finished building a solution, pheromone trails on all item-bin pairs need to be updated in order to help guiding the algorithm towards the optimal solution. Thereby, a pheromone trail update rule $\tau_{i,v}$ exists and is used in order to simulate pheromone evaporation and reinforce item-bin pairs which belonged to the so far best solution. In this work we follow the MAX-MIN Ant System (MMAS) [24] approach in which only the *iteration's-best* ant (i.e., ant whose solution's objective function value is minimal) is allowed to deposit pheromone. The pheromone update rule is defined in Eq. 8.

$$\tau_{i,v} := (1-\rho) \times \tau_{i,v} + \triangle\tau_{i,v}^{best}, \quad \forall\, (i, B_v) \in I \times B \tag{8}$$

whereby, the constant $\rho$, $0 \leq \rho \leq 1$ is used to simulate pheromone evaporation. Hence, higher values for $\rho$ lead to increased evaporation rate. Moreover, some item-bin pairs need to be reinforced. Thereby, $\triangle\tau_{i,v}^{best}$ is defined as the *iteration's-best* item-bin pheromone amount. Hence, if some item belongs to a bin of the so far best solution $S_{best}$, its pheromone amount is reinforced. Consequently, only item-bin pairs which are part of $S_{best}$ will be reinforced and thus become more attractive. Others, which are not part of $S_{best}$ will continue loosing pheromone according to the pheromone evaporation rate $\rho$. Thereby, a solution $S := [x_{i,j}]_{|I| \times |B|}$ is defined as a binary matrix whose elements represent the mapping of items to bins.

The ultimate goal of our ACO-based algorithm is to minimize the amount of bins, thus increasing the average utilization of each bin. Hence, we target to favor solutions which utilize the least number of bins. Therefore, we define the amount of pheromone iteration's *best* ant deposits on the item-bin pair to be inverse proportional to the value of the objective function $f$ applied on the iteration's best-solution $S_{best}$. Thereby, only item-bin pairs which are marked as allocated in $S_{best}$ will be reinforced.

$$\triangle\tau_{i,v}^{best} := \begin{cases} \frac{1}{f(S_{best})} & \text{if } x_{i,v} = 1 \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

Finally, because only the *iteration's-best* ant is allowed to deposit pheromone, early stagnation of the search is most likely to happen, thus leading to a situation in which all ants always choose the same items. Thereby, the ability of the algorithm to explore alternative solutions is decreased. In order to limit this effect, MMAS [24] introduces lower and upper bounds for the pheromone values $\tau_{i,v}$. Hence, $\tau_{i,v}$ is restricted to the range $[\tau_{min}, \tau_{max}]$. Analogously, we define $\tau_{max}$ as $\tau_{max} := \frac{1}{f(S_{best}) \times (1-\rho)}$ and $\tau_{min}$ as $\tau_{min} := \frac{\tau_{max}}{g}$, respectively with factor $g > 1$.

### D. Formal Algorithm Definition

The pseudocode of the ACO-based algorithm is depicted in Figure 1. The algorithm takes as input the set of items and bins, including their corresponding time-varying resource demand vectors $\vec{r}_i$ and static resource capacity vectors $\vec{C}_v$, respectively. Moreover, a set of parameters (i.e., $\alpha$, $\beta$, $\rho$, $g$, $\tau_{max}$, $nCycles$, $nAnts$) is required for initialization. First, the parameters are initialized and the pheromone trails of the items are set to $\tau_{max}$ (line 4). The algorithm then iterates until the specified number of cycles $nCycles$ (lines 5 to 35). Thereby, for each iterations an ant $a$ opens a bin $v$ and starts building a solution $S_a$ (lines 6 to 20). This is achieved by first initializing the set of items $IS$, the elements of the binary solution matrix

$S_a$ and the *bin-index* variable $v$. The algorithm then enters a loop and starts assigning the items to the bins (lines 9 to 19). Thereby, the current bin $v$ is being filled until its resources are saturated. This is achieved by initializing the set $N_v$ with all items which are not yet assigned to any bin and do not violate the capacity constraints of the current bin (line 10). If this set is not empty, the probabilistic decision rule $p_v^i$ is used to select one item $i$ out of the set to be packed in the current bin $v$, stochastically (line 12). The item is then marked as allocated in the solution matrix by setting the appropriate value in the matrix $S_a$ to 1, removed from the set of items $IS$ and the host utilization is updated (lines 13 to 15). This process is performed as long as there are still items left to be assigned and enough capacity available in the current bin (line 9 and 10). Afterwards, when the bin capacity is saturated (i.e., $N_v$ becomes empty) the bin-index variable is incremented and the packing process is continued until all items are placed (lines 9 to 19). After all ants have constructed their solutions $S_a$, a comparison is performed and the *cycle's best* solution is saved (line 21) as $S_{cycle}$. Thereby, two criteria: *amount of utilized bins* and *amount of failed item allocations* are used in order to judge about the cycle best solution. While the first one seems natural, the second one is a result of two solutions which equal in terms of utilizing all available bins but differ in the utilization efficiently of the bins. For instance, two solutions would use the same number of bins, but the first one would fail allocating resources for 10% of the requests while the second one would satisfy all requests. Finally, if this is the first cycle, the cycle best solution becomes the global best one. Otherwise, a comparison is done with the current global best solution. Thereby, if the cycle best solution yields to an improvement it becomes the new global best one (lines 22 to 24). Afterwards, the values for $\tau_{min}$ and $\tau_{max}$ are computed (line 25) and the pheromone trails on all item-bin pairs $(i, B_v)$ are updated using the pheromone update rule $\tau_{i,v}$ (lines 26 to 34). Thereby, in order to respect the specified lower and upper bounds for $\tau_{i,v}$ two conditions exist. First condition guarantees that the upper bound is respected. Hence, if some item-bin pair received a higher pheromone amount than $\tau_{max}$, it is reinitialized to $\tau_{max}$ (lines 28 to 30). Similarly, when the pheromone amount of some items falls below the predefined lower bound $\tau_{min}$ it is updated accordingly (lines 31 to 33). The algorithm terminates after $nCycles$ and returns the so far global best computed solution $S_{best}$ (line 36).

**Algorithm 1** Energy-Aware ACO-based Workload Consolidation

1: Input: Set of items $I$ and set of bins $B$ with their associated resource demand vectors $\vec{r}_i$ and $\vec{C}_v$ respectively, Set of parameters
2: Output: Global best solution $S_{best}$
3:
4: Initialize parameters, Set pheromone value on all item-bin pairs to $\tau_{max}$
5: **for all** $q \in \{0 \dots nCycles - 1\}$ **do**
6:   **for all** $a \in \{0 \dots nAnts - 1\}$ **do**
7:     $IS := I; v := 0$
8:     $S_a := [x_{i,j} := 0], \forall i \in \{0, \dots, m-1\}, \ \forall j \in \{0, \dots, n-1\}$
9:     **while** $IS \neq \emptyset$ **do**
10:       $N_v := \{i \mid \sum_{j=0}^{n-1} x_{i,j} = 0 \wedge \vec{b}_v + \vec{r}_i \leq \vec{C}_v\}$
11:       **if** $N_v \neq \emptyset$ **then**
12:         Choose item $i \in N_v$ stochastically according to probability $p_v^i := \frac{[\tau_{i,v}]^\alpha \times [\eta_{i,v}]^\beta}{\sum_{u \in N_v} [\tau_{u,v}]^\alpha \times [\eta_{u,v}]^\beta}$
13:         $x_{i,v} := 1$
14:         $IS := IS - \{i\}$
15:         $\vec{b}_v := \vec{b}_v + \vec{r}_i$
16:       **else**
17:         $v := v + 1$
18:       **end if**
19:     **end while**
20:   **end for**
21:   Compare ants solutions $S_a$ according to the objective function $f \rightarrow$ Save cycle best solution as $S_{cycle}$
22:   **if** $q = 0 \vee IsGlobalBest(S_{cycle})$ **then**
23:     Save cycle best solution as new global best $S_{best}$
24:   **end if**
25:   Compute $\tau_{min}$ and $\tau_{max}$
26:   **for all** $(i, B_v) \in I \times B$ **do**
27:     $\tau_{i,v} := (1 - \rho) \times \tau_{i,v} + \triangle\tau_{i,v}^{best}$
28:     **if** $\tau_{i,v} > \tau_{max}$ **then**
29:       $\tau_{i,v} := \tau_{max}$
30:     **end if**
31:     **if** $\tau_{i,v} < \tau_{min}$ **then**
32:       $\tau_{i,v} := \tau_{min}$
33:     **end if**
34:   **end for**
35: **end for**
36: **return** Global best solution $S_{best}$

## V. Experimental Results

This section present the performance evaluation of the proposed ACO-based workload consolidation algorithm. Thereby, in order to gain a first insight into the performance of the algorithm on large-scale before implementing it in a real environment, we have decided to conduct simulation-based experiments. Therefore, because of limitations of existing cloud computing simulation frameworks (e.g., CloudSim [6] only allows to move one VM per event, making it difficult to simulate global re-packing), we have developed our own Java-based simulation toolkit and used it to compare the ACO-based algorithm with the frequently applied FFD heuristic. Thereby, in order to improve the performance a multithreaded version of the ACO-based workload consolidation algorithm was developed. Furthermore, a modified version of the FFD heuristic has been implemented in order to consider the multidimensional character of the problem. Thereby, the VM resource demand vectors were sorted in decreasing order according to the L1-norm.

We simulated a cluster composed of homogeneous hosts with each having a static resource capacity of 10000 MIPS, 24 cores, 50 GB of RAM, 1 TB storage and 10 GBit/sec network connection. Thereby, the amount of hosts was set to the amount of VMs in order to support the worst packing scenario, in which only one VM is assigned per host. In total, up to 600 VMs were simulated with each requiring either 1000, 2000, 3000 or 5000 of MIPS, 2 cores, 4 GB of RAM, 200 GB of storage and 1 GBit/sec of network bandwidth.

In order to estimate the energy consumed by a placement, we approximate the power of a host as a linear function $P(u)$ in its current utilization $u \in [0, 1]$ (see Eq. 10).

$$P(u) = (P_{max} - P_{idle}) \times u + P_{idle} \tag{10}$$

with $P_{idle}$ and $P_{max}$ being the average power values when the system is idle and fully utilized, respectively. Both values have been fixed to *171* and *218 Watt*, for all simulations according to the measurements performed on our own testbed. The testbed we use is equipped with one Dell PowerEdge 1950 server plugged into a Sentry POPS (Per Outlet Power Sensing) switched Cabinet Distribution Unit (CDU). The machine comprises 4 GB of RAM and two Intel Xeon 5148 2.33 GHz CPUs, each having 2 cores. Idle power was derived by measuring the power drawn by the testbed machine when it only hosts the OS and the least amount of required system services (e.g., udev, sshd, etc.). Average peak power consumption was measured by running the *stress* benchmark application, with parameters set to stress all the system components.

Finally, for estimating the energy consumed by a placement a time period $t$ was defined and set to 24 hours. Consequently, the energy values represent the power drawn by the cluster at the utilization given by the placement over the period of 24 hours. Thereby, it was assumed that idle machines are turned off after the workload consolidation. Hence, their idle power is not part of the total placement energy consumption. In particular, energy consumed by a placement was computed according to Eq. 11.

$$E(B) := \begin{cases} t \times \sum_{v=0}^{n-1} P(\frac{|\vec{b}_v|_1}{d}) & \text{if } |\vec{b}_v|_1 \neq 0 \\ 0 & \text{otherwise} \end{cases} \qquad (11)$$

Note, that because of the non-proportional power usage (i.e., high idle power) of traditional servers, no manner which energy model is used, turning off/suspending machines always yields to energy savings assuming that the algorithm is triggered during appropriate time periods (e.g., low utilization). Moreover, since the packing is based on peak resource demands, we assume that consolidated workloads will not suffer from significant performance degradation. Given such assumptions, consolidating the workload of for example two servers at 0.3 and 0.7 utilization, respectively, onto one server running at peak utilization (i.e., 1) is always advantageous.

The parameters of the ACO-based algorithm were derived empirically through numerous simulations and finally set as depicted in Table I. Thereby, the amount of cycles and ants were finally initialized to 2 and 5, respectively, above which no improvement in the solutions could be observed.

TABLE I
PARAMETERS OF THE ACO-BASED WORKLOAD CONSOLIDATION ALGORITHM

| $\alpha$ | $\beta$ | $\rho$ | g | $\tau_{max}$ | $nCycles$ | $nAnts$ |
|---|---|---|---|---|---|---|
| 1 | 2 | 0.7 | 2 | 3 | 2 | 5 |

We run the simulation for up to 600 VMs and measured the amount of provisioned hosts, energy consumption of the placement and the average execution times for both algorithms (i.e., FFD and ACO-based). In addition, in order to judge the quality of the solutions, optimal solutions were computed by integrating the previously introduced BIP model into the high-performance Mixed-integer linear programming (MILP) solver IBM ILOG CPLEX v12.2 [1]. Thereby, the solver was set to emphasis optimality and run in parallel mode with 4 threads.

In order, to derive the actual energy savings, the amount of energy spent for computing the placement was estimated by multiplying the execution time of the algorithm with average power drawn (i.e., 198 Watt) of the system during the simulation. The resulting amount of energy spent for the simulation was included into the final energy consumption of the placement and accounted not more than 400 Wh. Therefore, it did not impact the total energy results of the algorithms which were in the order of *kWh*. The final numerical simulation results are depicted in Table II.

TABLE II
NUMERICAL SIMULATION RESULTS

| VMs | Policy | Hosts | Execution time | Energy (= kWh) | Energy gain (= %) |
|---|---|---|---|---|---|
| 100 | FFD | 30 | 0.39 sec | 139.62 | |
| | ACO | 28 | 37.46 sec | 131.41 | 5.88 |
| | CPLEX | 28 | 0.451 sec | 131.41 | 5.88 |
| 200 | FFD | 59 | 0.58 sec | 275.13 | |
| | ACO | 56 | 4.51 min | 262.83 | 4.47 |
| | CPLEX | 55 | 1.27 sec | 258.71 | 5.96 |
| 300 | FFD | 88 | 0.77 sec | 410.65 | |
| | ACO | 84 | 15.04 min | 394.28 | 3.98 |
| | CPLEX | 83 | 2.86 sec | 390.12 | 4.99 |
| 400 | FFD | 117 | 1.03 sec | 546.16 | |
| | ACO | 112 | 34.23 min | 525.75 | 3.73 |
| | CPLEX | 110 | 5.07 sec | 517.43 | 5.26 |
| 500 | FFD | 146 | 1.39 sec | 681.67 | |
| | ACO | 139 | 1.17 h | 653.17 | 4.18 |
| | CPLEX | 138 | 9.41 sec | 648.84 | 4.81 |
| 600 | FFD | 175 | 1.75 sec | 817.19 | |
| | ACO | 167 | 2.01 h | 784.75 | 3.96 |
| | CPLEX | 165 | 12.95 sec | 776.14 | 5.02 |

Figure 1, provides a graphical representation of the results.

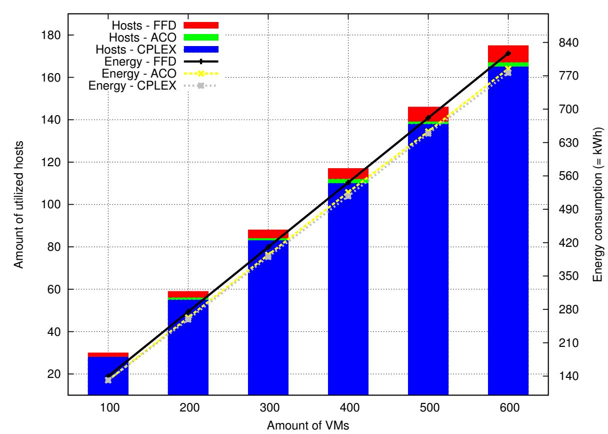


Fig. 1. Amount of utilized hosts and energy consumption

As it can be observed, computation time required to derive the placement and thus the energy spent in computation are higher using the ACO-based approach. This is because of our implementation which is *far from being optimal* while the used LP-solver (i.e., CPLEX) is *highly optimized*. In particular, 1.75 sec were required to compute the placement for the highest amount of VMs (i.e., 600) by the FFD and 2.01 hours by the ACO-based algorithm, resulting in 0.09 Wh and 397.98 Wh of energy spent in computation. Nevertheless, the solutions of the ACO-based approach utilize *significantly lower amounts of hosts* and thus yield to *superior average host utilizations and* *energy gains*. Thereby, on average 4.7% of hosts and 4.1% of energy were conserved by applying the ACO approach. Moreover, the solutions computed by the ACO-based approach are nearly optimal (i.e., small deviation of 1.1%). In addition, complexity of both evaluated algorithms is quadratic in the number of virtual machines, while CPLEX despite being highly efficient is exponential in the worst-case. Finally, it is worth mentioning that under a constrained number of hosts such as it is the case in a real system, FFD would need longer time to schedule the workload as it requires higher amounts of hosts. Consequently, the number of VMs which are required to reside in queues (i.e., *non-allocatable*) is higher when the FFD approach is applied.

## VI. Related Work

A lot of research has been done on designing algorithms for solving instances of bin-packing problems, during the last three decades. Hence, a variety of exact algorithms and heuristics have been designed. Because of the NP-hard nature of the problem and the need to compute the solutions in reasonable time we focus our work and thus the related work on heuristic algorithms. These works can be divided into two categories: greedy algorithms (e.g., Best-Fit (BF), First-Fit (FF), Next-Fit (FF), Best-Fit Decreasing (BFD), First-Fit Decreasing (FFD), Permutation Pack (PP), Choose Pack (CP)) and evolutionary algorithms (e.g., genetic algorithms, ant systems).

In [9], the authors survey the existing greedy algorithms for solving one-dimensional bin-packing problems. The approximability of the multi-dimensional bin-packing (MDBP) problem and the related Vector Bin-Packing (VBP) problem is studied in [8]. Moreover, in [23], the authors model the MDBP *related* resource allocation problem in virtualized service hosting environments and provide simulation-driven results for many state-of-the-art greedy algorithms (e.g., FF, BF, PP, CP). Thereby, the objective of the evaluated algorithms is to maximize the minimum yield over all services. Given such an objective, they identify the CP algorithm to be the most effective one. On the contrary, our objective is to minimize the amount of active servers.

In [18], the first evolutionary algorithm based on the Ant Colony Optimization (ACO) meta-heuristic was designed for solving *one-dimensional bin-packing problems*. The authors have shown that combined with a local search their algorithm could compete with the best known solution methods. This work has been further refined in [4]. Thereby, an algorithm called AntPacking is proposed and shown to perform at least as good as the best genetic algorithm.

In [25], the authors model the workload placement as an instance of the one-dimensional bin-packing problem and apply a modified version of the FFD algorithm to perform the placement. In [19], a framework called EnaCloud is proposed and a modified version of the Better-Fit algorithm is applied. Similar work can be found in [3], where the authors present simulation-driven results for a workload placement algorithm based on a modified version of the BFD algorithm and report substantial energy saving. More energy-aware workload placement approaches, which resort to the adaptation of simple greedy algorithms can be found in [22] and [20].

In [10], a virtual machine *deployment framework* for private and public clouds based on Ant Colony Optimization (ACO) is proposed. Thereby, the authors deal with the problem of how to efficiently deploy virtual machines images in the cloud. Particularly, this work aims at constructing balanced and dependable deployment configurations by the use of replication.

Our approach falls down into the same category of techniques (i.e., ACO-based). However, its objective is contrary as it targets to *unbalance* the workload in order to *conserve energy*. To the best of our knowledge this is the first work to: (1) *apply ACO on the MDBP problem in the context of dynamic workload placement* and (2) *utilize ACO in order to conserve energy*. Hence, unlike most of the introduced approaches dealing with the *dynamic workload placement problem* which consider only one resource (i.e., CPU) and resort to relatively simple resource-dissipative greedy algorithms [21], we accurately *model* the *workload placement* problem as an instance of the MDBP problem and take a *nature-inspired* evolutionary approach to perform the placement. Our simulation results show that the ACO-based approach outperforms the evaluated greedy algorithm (i.e., FFD) as it achieves superior energy gains through improved resource utilization. Moreover, it computes solutions which are within 1.1% from the optimal and unlike CPLEX its complexity is quadratic in the number of virtual machines. Finally, contrary to simple greedy algorithms and LP solvers, the proposed algorithm can be implemented in a distributed manner, thus making it more suitable for scalability and fault-tolerance.

## VII. Conclusions and Future Work

In this paper, we propose a nature-inspired approach for solving the dynamic workload placement problem for present and future energy-aware Infrastructure-as-a-Service (IaaS) cloud computing environments. To the best of our knowledge this is the first work to *apply artificial swarm intelligence on the MDBP problem in the context of dynamic workload placement* and *analyze its energy benefits*. In particular, we have first accurately defined the workload placement problem as an instance of the multi-dimensional bin-packing (MDBP) problem by introducing a binary integer programming (BIP) model and proposed to use long-term history resource utilization measures in order to estimate future workload resource demands. We then have introduced a *novel* dynamic workload placement algorithm based on the Ant Colony Optimization (ACO) meta-heuristic to solve the introduced problem and compared it with one traditional greedy algorithm (i.e., FFD). Both algorithms have been implemented and experimentally validated by means of simulations. The results demonstrate that the artificial swarm intelligence based approach (i.e., ACO) provides superior energy gains than traditional workload placement based on the evaluated greedy algorithm and achieves *nearly optimal* results (i.e., 1.1% deviation). Particularly, on average 4.7% of hosts and 4.1% of energy were conserved by applying ACO. However, the savings came at the cost of increased computation time. Therefore, we conclude that complementarity between the two approaches should be exploited in order to increase the energy efficiency. For example, a FFD-based policy could be used to perform initial schedule of submitted VM batches, while the ACO-based approach could run in background and optimize the placements on regular time intervals (e.g., daily or weekly basis). Moreover, FFD and other heuristics of the same category (e.g., Best-Fit, Next-Fit, etc.) are highly centralized. Hence, they fail to fulfill important properties such as scalability and fault tolerance. Even though our approach is also implemented in a centralized manner and its implementation is far from being optimal, the autonomous nature of ants allows to it be implemented in a fully distributed environment. Hence, avoiding single point of failure (SPOF) and providing good properties such as scalability and fault-tolerance.

Consequently, in the future we plan to design a distributed version of the algorithm and support hardware heterogeneity. In addition, despite the resource isolation properties of virtualization technology, co-location of workload with similar characteristics (e.g., memory intensive) on the same physical machine can lead to performance degradation even if no resource shortage exists as the caches are typically shared between the workload. Hence, we plan to adapt the algorithm in order to take into account workload characteristics. Finally, we plan to investigate approaches for accurate time estimations at which the algorithm should be triggered. Such estimations are necessary in order to prevent possible performance degradation because of migration and thus reduce the risk of Service Level Agreement (SLA) violation. This will enable the application of the algorithm in an real environment. Last but not least, we are currently working on a first prototype implementation of the previously proposed hierarchical and distributed workload (i.e., VM) consolidation manager called Snooze [13]. The algorithm introduced in the present paper is part of this prototype and thus will be experimentally validated in the near future. Experiments will be performed on the Grid5000 testbed [7].

## VIII. Acknowledgment

This research is funded by the french *Agence Nationale de la Recherche (ANR)* project EcoGrappe under the contract number ANR-08-SEGI-000.

## References

[1] Ilog cplex: High-performance software for mathematical programming and optimization, 2011. http://www.ilog.com/products/cplex/.

[2] O. D. C. Alliance. Carbon footprint values, 2011.

[3] A. Beloglazov and R. Buyya. Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers. In *Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science*, MGC '10, pages 4:1–4:6, New York, NY, USA, 2010. ACM.

[4] B. Brugger, K. F. Doerner, R. F. Hartl, and M. Reimann. Antpacking an ant colony optimization approach for the one-dimensional bin packing problem. In J. Gottlieb and G. R. Raidl, editors, *Evolutionary Computation in Combinatorial Optimization*, volume 3004 of *Lecture Notes in Computer Science*, pages 41–50. Springer Berlin / Heidelberg, 2004.

[5] R. Buyya, A. Beloglazov, and J. H. Abawajy. Energy-efficient management of data center resources for cloud computing: A vision, architectural elements, and open challenges. *CoRR*, abs/1006.0308, 2010.

[6] R. N. Calheiros, R. Ranjan, C. A. F. D. Rose, and R. Buyya. Cloudsim: A novel framework for modeling and simulation of cloud computing infrastructures and services. *CoRR*, abs/0903.2525, 2009. informal publication.

[7] F. Cappello, E. Caron, M. Dayde, F. Desprez, Y. Jegou, P. Primet, E. Jeannot, S. Lanteri, J. Leduc, N. Melab, G. Mornet, R. Namyst, B. Quetier, and O. Richard. Grid'5000: A large scale and highly reconfigurable grid experimental testbed. In *GRID '05: Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*, pages 99–106, Washington, DC, USA, 2005. IEEE Computer Society.

[8] C. Chekuri and S. Khanna. On multi-dimensional packing problems. In *Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '99, pages 185–194, Philadelphia, PA, USA, 1999. Society for Industrial and Applied Mathematics.

[9] E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson. *Approximation algorithms for bin packing: a survey*, pages 46–93. PWS Publishing Co., Boston, MA, USA, 1997.

[10] M. J. Csorba, H. Meling, and P. E. Heegaard. Ant system for service deployment in private and public clouds. In *Proceeding of the 2nd workshop on Bio-inspired algorithms for distributed systems*, BADS '10, pages 19–28, New York, NY, USA, 2010. ACM.

[11] Deneubourg, S. Aron, S. Goss, and J. M. Pasteels. The self-organizing exploratory pattern of the argentine ant. *Journal of Insect Behavior*, 3(2):159–168, March 1990.

[12] M. Dorigo, G. Di Caro, and L. M. Gambardella. Ant algorithms for discrete optimization. *Artif. Life*, 5:137–172, April 1999.

[13] E. Feller, L. Rilling, C. Morin, R. Lottiaux, and D. Leprince. Snooze: A Scalable, Fault-Tolerant and Distributed Consolidation Manager for Large-Scale Clusters. In *2010 IEEE/ACM International Conference on Green Computing and Communications (GreenCom-2010)*, Hangzhou, China, 2010.

[14] W.-c. Feng, X. Feng, and R. Ge. Green supercomputing comes of age. *IT Professional*, 10(1):17–23, 2008.

[15] G. International. Make it green: Cloud computing and its contribution to climate change, 2010.

[16] J. M. Kaplan, W. Forrest, and N. Kindler. Revolutionizing data center energy efficiency. Technical report, 2008.

[17] G. Khanna, K. Beaty, G. Kar, and A. Kochut. Application performance management in virtualized server environments. In *Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP*, pages 373 –381, 2006.

[18] J. Levine and F. Ducatelle. Ant colony optimisation and local search for bin packing and cutting stock problems. *Journal of the Operational Research Society. (forthcoming)*, 93:2003, 2003.

[19] B. Li, J. Li, J. Huai, T. Wo, Q. Li, and L. Zhong. Enacloud: An energy-saving application live placement approach for cloud computing environments. In *Proceedings of the 2009 IEEE International Conference on Cloud Computing*, CLOUD '09, pages 17–24, Washington, DC, USA, 2009. IEEE Computer Society.

[20] M. Y. Lim, F. Rawson, T. Bletsch, and V. W. Freeh. Padd: Power aware domain distribution. *Distributed Computing Systems, International Conference on*, 0:239–247, 2009.

[21] T. Setzer and A. Stage. Decision support for virtual machine re-assignments in enterprise data centers. In *Network Operations and Management Symposium Workshops (NOMS Wksps), 2010 IEEE/IFIP*, pages 88 –94, 2010.

[22] S. Srikantaiah, A. Kansal, and F. Zhao. Energy aware consolidation for cloud computing. In *Proceedings of HotPower '08 Workshop on Power Aware Computing and Systems*. USENIX, December 2008.

[23] M. Stillwell, D. Schanzenbach, F. Vivien, and H. Casanova. Resource allocation algorithms for virtualized service hosting platforms. *J. Parallel Distrib. Comput.*, 70:962–974, September 2010.

[24] T. Stutzle and H. Hoos. Improvements on ant-system: Introducing max-min ant system, 1996.

[25] A. Verma, P. Ahuja, and A. Neogi. pmapper: power and migration cost aware application placement in virtualized systems. In *Middleware '08: Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*, pages 243–264, New York, NY, USA, 2008. Springer-Verlag New York, Inc.

[26] W. Vogels. Beyond server consolidation. *Queue*, 6(1):20–26, 2008.