Cielo: An Evolutionary Game Theoretic Framework for Virtual Machine Placement in Clouds

Yi Ren*, Junichi Suzuki*, Athanasios Vasilakos†, Shingo Omura‡ and Katsuya Oba‡
* University of Massachusetts, Boston
Boston, MA 02125-3393, USA
Email: {yiren001,jxs}@cs.umb.edu
† University of Western Macedonia
GR 50100, Kozani, Greece
Email: vasilako@ath.forthnet.gr
‡OGIS International, Inc.
San Mateo, CA 94402, USA
Email: {omura,oba}@ogis-international.com

Abstract—**This paper studies an evolutionary game theoretic framework for adaptive and stable application deployment in clouds. The framework, called Cielo, aids cloud operators to adapt the resource allocation to applications and their locations to the operational conditions in a cloud (e.g., workload and resource availability) with respect to multiple conflicting objectives (e.g., response time and power consumption). Moreover, Cielo theoretically guarantees that each application performs an evolutionarily stable deployment strategy, which is an equilibrium solution under given operational conditions. Simulation results verify this theoretical analysis; applications seek equilibria to perform adaptive and evolutionarily stable deployment strategies. Cielo outperforms well-known existing heuristics.**

Keywords*—*Cloud computing, virtual machine placement, evolutionary game theory

I. Introduction

One of key features in cloud computing is elastic scaling. In order to ensure this feature, cloud operators are required to dynamically (re-)deploy applications by adjusting their locations and resource allocation. This paper investigates two important properties of application deployment in clouds:

- *Adaptability*: Adjusting the locations of and resource allocation for applications according to operational conditions (e.g., workload and resource availability) with respect to given performance objectives.
- *Stability*: Minimizing oscillations (non-deterministic inconsistencies) in making adaptation decisions.

Cielo is an evolutionary game theoretic framework for adaptive and stable application deployment in clouds. This paper describes its design and evaluates its adaptability and stability. In Cielo, each application maintains a set (or a population) of deployment strategies, each of which indicates the location of and resource allocation for that application. Cielo theoretically guarantees that, through a series of evolutionary games between deployment strategies, the population state (i.e., the distribution of strategies) converges to an evolutionarily stable equilibrium, which it always converges to regardless of the initial state. (A dominant strategy in the evolutionarily stable population state is called an *evolutionarily stable strategy*.) In this state, no other strategies except an evolutionarily stable strategy can dominate the population. Given this theoretical property, Cielo aids each application to operate at equilibria by using an evolutionarily stable strategy for application deployment in a deterministic (i.e., stable) manner. Simulation results verify this theoretical analysis; applications seek equilibria to perform evolutionarily stable deployment strategies and adapt their locations and resource allocations to given operational conditions. Cielo outperforms existing heuristics, FFA (first-fit algorithm) and BFA (best-fit algorithm), which have been widely used for adaptive cloud application deployment [1]–[4].

II. Problem Statement

This section formulates an application deployment problem where M hosts are available to operate N applications. Each application is designed with a set of server software, following a three-tier application architecture. Three different types of servers operate at different tiers (Fig. 1). Using a certain hypervisor, each server is assumed to run on a virtual machine (VM) atop a host. A host can operate multiple VMs. Collocated VMs share resources available on their local host.

Each message is sequentially processed from a Web server to a database server through an application server. A reply message is generated by the database server and forwarded in the reverse order (Fig. 1). This paper assumes that different applications utilize different sets of servers. (Servers are not shared by different applications.)

The goal of this problem is to find evolutionarily stable strategies that deploy N applications (i.e., $N \times 3$ VMs) on M hosts so that the applications adapt their locations and resource allocation to given workload and resource availability with respect to five performance objectives described below. (All objectives are to be minimized.)

CPU allocation: A certain CPU time share (in percentage) is allocated to each VM. (The CPU share of 100% means that a CPU is fully allocated to a VM.) It represents the upper limit for the VM's CPU utilization. This objective is computed as the sum of CPU shares allocated to three VMs of an application.

Bandwidth allocation: A certain amount of bandwidth (in kbps) is allocated to each VM. It is the upper limit for the

VM's bandwidth consumption. This objective is computed as the sum of bandwidth allocated to three VMs of an application.

Response time: This objective is computed as the time required for a message to travel from a web server to a database server and from the database server back to the web server: $T^p + T^w + T^c$ where T^p denotes the total time for an application to process an incoming message from a user at three servers, T^w denotes the waiting time for a message to be processed at servers, and T^c denotes the total communication delay to transmit a message among servers. Response time is estimated based on an $M/M/1$ queuing model, in which message arrivals follow a Poisson process and a server's message processing time is exponentially distributed.

T^p is computed as follows where T^p_t denotes the time required for the t-th tier server to process a message.

$$T^p = \sum_{t=1}^{3} T^p_t \tag{1}$$

T^w is computed as follows.

$$T^w = \frac{1}{\lambda} \sum_{t=1}^{3} \frac{\rho_t^2}{1-\rho_t} \quad \text{where } \rho_t = \lambda_t \frac{T^p_t}{c_t} \tag{2}$$

λ denotes the message arrival rate for an application (i.e., the number of messages the application receives from users in the unit time). ρ_t is the CPU utilization of the t-th tier server. Note that $\lambda = \frac{1}{3}\sum_{t=1}^{3} \lambda_t$. (Currently, $\lambda = \lambda_1 = \lambda_2 = \lambda_3$.) c_t denotes the CPU time share allocated to the t-th tier server.

T^c is computed as follows:

$$T^c = \sum_{t=1}^{2} T^c_{t \to t'} \approx \sum_{t'=2}^{3} \frac{B \cdot \lambda_{t'}}{b_t}, \quad t' = t+1 \tag{3}$$

B denotes the size of a message (in bits). b_t denotes the bandwidth allocated to the t-th tier server (bits/second).

Power Consumption: This objective indicates the total power consumption (in Watts) by three hosts in an application.

$$P = \sum_{t=1}^{3} \{P_{idle} + (P_{max} - P_{idle}) \cdot c_t\} \tag{4}$$

P_{idle} and P_{max} denote the power consumption of a host when its CPU utilization is zero and 100%, respectively.

Workload distribution: This objective indicates how evenly CPU utilization is distributed over hosts:

$$\sqrt{\frac{1}{L}\sum_{i=1}^{L}(w_i - \bar{w})^2}, \quad 1 < L < 3, \quad 0 < w_i < 1 \tag{5}$$

L denotes the number of hosts that run three servers of an application. $L = 1$ when all three servers are collocated on a host. $L = 3$ when they are deployed on three different hosts. w_i is the total CPU share allocated to the i-th host of those L hosts. $\bar{w} = (\sum_{i=1}^{L} w_i)/3$.

Cielo considers a CPU capacity constraint: $w_i \leq 1$ for all M hosts. The violation of this constraint is computed as:

$$c^v = \sum_{i=1}^{M} (I_i \cdot (w_i - 1)) \tag{6}$$

$I_i = 1$ if $w_i > 1$. Otherwise, $I_i = 0$.

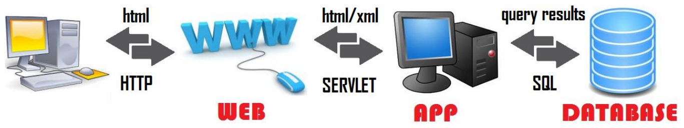


Fig. 1: Three Tiers of Web, Application and Database Servers

III. Background: Evolutionary Game Theory

In a conventional game, the objective of a player is to choose a strategy that maximizes its payoff. In contrast, evolutionary games are played repeatedly by players randomly drawn from a population This section overviews key elements in evolutionary games: evolutionarily stable strategies (ESS) and replicator dynamics.

A. Evolutionarily Stable Strategies (ESS)

Suppose all players in the initial population are programmed to play a certain (incumbent) strategy k. Then, let a small population share of players, $x \in (0,1)$, mutate and play a different (mutant) strategy ℓ. When a player is drawn for a game, the probabilities that its opponent plays k and ℓ are $1-x$ and x, respectively. Thus, the expected payoffs for the player to play k and ℓ are denoted as $U(k, x\ell + (1-x)k)$ and $U(\ell, x\ell + (1-x)k)$, respectively.

Definition 1: A strategy k is said to be evolutionarily stable if, for every strategy $\ell \neq k$, a certain $\bar{x} \in (0,1)$ exists, such that the inequality

$$U(k,\ x\ell + (1-x)k) > U(\ell,\ x\ell + (1-x)k) \tag{7}$$

holds for all $x \in (0, \bar{x})$.

If the payoff function is linear, Equation 7 derives:

$$(1-x)U(k,k) + xU(k,\ell) > (1-x)U(\ell,k) + xU(\ell,\ell) \tag{8}$$

If x is close to zero, Equation 8 derives either

$$U(k,k) > U(\ell,k) \ or \ U(k,k) = U(\ell,k) \ and \ U(k,\ell) > U(\ell,\ell) \tag{9}$$

This indicates that a player associated with the strategy k gains a higher payoff than the ones associated with the other strategies. Therefore, no players can benefit by changing their strategies from k to the others. This means that an ESS is a solution on a Nash equilibrium. An ESS is a strategy that cannot be invaded by any alternative (mutant) strategies that have lower population shares.

B. Replicator Dynamics

The replicator dynamics describes how population shares associated with different strategies evolve over time [5]. Let $\lambda_k(t) \geq 0$ be the number of players who play the strategy $k \in K$, where K is the set of available strategies. The total population of players is given by $\lambda(t) = \sum_{k=1}^{|K|} \lambda_k(t)$. Let $x_k(t) = \lambda_k(t)/\lambda(t)$ be the population share of players who play k at time t. The population state is defined by $X(t) = [x_1(t), \cdots, x_k(t), \cdots, x_K(t)]$. Given X, the expected payoff of playing k is denoted by $U(k, X)$. The population's average payoff, which is same as the payoff of a player drawn randomly from the population, is denoted by $U(X,X) = \sum_{k=1}^{|K|} x_k \cdot U(k,X)$. In the replicator dynamics, the dynamics of the population share x_k is described as follows. $\dot{x}_k$ is the time derivative of x_k.

$$\dot{x}_k = x_k \cdot [U(k,X) - U(X,X)] \tag{10}$$

This equation states that players increase (or decrease) their population shares when their payoffs are higher (or lower) than the population's average payoff.

Theorem 1: If a strategy k is strictly dominated, then $x_k(t)_{t\to\infty} \to 0$.

A strategy is said to be strictly dominant if its payoff is strictly higher than any opponents. As its population share grows, it dominates the population over time. Conversely, a strategy is said to be strictly dominated if its payoff is lower than that of a strictly dominant strategy. Thus, strictly dominated strategies disappear in the population over time.

There is a close connection between Nash equilibria and the steady states in the replicator dynamics, in which the population shares do not change over time. Since no players change their strategies on Nash equilibria, every Nash equilibrium is a steady state in the replicator dynamics. As described in Section III-A, an ESS is a solution on a Nash equilibrium. Thus, an ESS is a solution at a steady state in the replicator dynamics. In other words, an ESS is the strictly dominant strategy in the population on a steady state.

Cielo maintains a population of deployment strategies for each application. In each population, strategies are randomly drawn to play games repeatedly until the population state reaches a steady state. Then, Cielo identifies a strictly dominant strategy in the population and deploys VMs based on the strategy as an ESS.

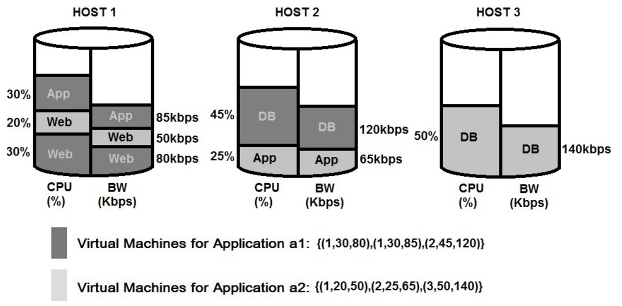

HOST 1
HOST 2
HOST 3
30%
20%
30%
App
Web
Web
App
Web
Web
85kbps
50kbps
80kbps
45%
25%
DB
App
DB
App
120kbps
65kbps
50%
DB
DB
140kbps
CPU (%)
BW (Kbps)
Virtual Machines for Application a1: {(1,30,80),(1,30,85),(2,45,120)}
Virtual Machines for Application a2: {(1,20,50),(2,25,65),(3,50,140)}

Fig. 2: Example Deployment Strategies

IV. Cielo

Cielo maintains N populations, $\{\mathcal{P}_1, \mathcal{P}_2, ..., \mathcal{P}_N\}$, for N applications and performs games among strategies in each population. A strategy s is defined to indicate the locations of and resource allocation for three VMs in an application:

$$s(a_i) = \bigcup_{t\in 1,2,3} (h_{i,t}, c_{i,t}, b_{i,t}), \quad 1 < i < N \tag{11}$$

a_i denotes the i-th application. $h_{i,t}$ is the ID of a host that operates a_i's t-th tier VM. $c_{i,t}$ and $b_{i,t}$ are the CPU and bandwidth allocation for a_i's t-th tier VM. Fig. 2 shows two example strategies for two applications (a_1 and a_2) ($N = 2$ and $M = 3$). a_1's strategy ($s(a_1)$) places the first-tier VM on host 1 ($h_{1,1} = 1$) and allocates the CPU share of 30% and 80 kbps bandwidth for the VM ($c_{1,1} = 30$ and $b_{1,1} = 80$). The second-tier VM is placed on host 1 ($h_{1,2} = 1$) and allocates the CPU share of 30% and 85 kbps bandwidth for the VM ($c_{1,2} = 30$ and $b_{1,2} = 85$). The third-tier VM is placed on host 2 ($h_{1,3} = 2$) and allocates the CPU share of 45% and 120 kbps bandwidth for the VM ($c_{1,3} = 45$ and $b_{1,3} = 120$).Given $s(a_1)$, a_1's objective values for CPU allocation, bandwidth allocation and workload distribution are 105 (30 + 30 + 45), 285 kbps and 7.5 ($\sqrt{\frac{1}{2}((60 - 52.5)^2 + (45 - 52.5)^2)}$).

Algorithm 1 Evolutionary Process in Cielo

```
 1: $g$ = 0
 2: Randomly generate the initial $N$ populations for $N$ applications:
    $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, ..., \mathcal{P}_N\}$
 3: while $g < G_{max}$ do
 4:     for each population $\mathcal{P}_i$ randomly selected from $\mathcal{P}$ do
 5:         $\mathcal{P}'_i \leftarrow \emptyset$
 6:         for $j = 1$ to $|\mathcal{P}_i|/2$ do
 7:             $s_1 \leftarrow$ randomlySelect($\mathcal{P}_i$)
 8:             $s_2 \leftarrow$ randomlySelect($\mathcal{P}_i$)
 9:             $winner \leftarrow$ performGame($s_1$, $s_2$)
10:             $replica \leftarrow$ replicate($winner$)
11:             if random() $\leq P_m$ then
12:                 $replica \leftarrow$ mutate($winner$)
13:             end if
14:             $\mathcal{P}_i \setminus \{s_1, s_2\}$
15:             $\mathcal{P}'_i \cup \{winner, replica\}$
16:         end for
17:         $\mathcal{P}_i \leftarrow \mathcal{P}'_i$
18:         $d_i \leftarrow argmax_{s\in\mathcal{P}_i} x_s$
19:         while $d_i$ is infeasible do
20:             $\mathcal{P}_i \setminus \{d_i\}$
21:             $d_i \leftarrow argmax_{s\in\mathcal{P}_i} x_s$
22:         end while
23:         Deploy VMs for the current application based on $d_i$.
24:     end for
25:     $g = g + 1$
26: end while
```

Algorithm 2 Game between Strategies (`performGame()`)

```
Input: $s_1$ and $s_2$: Strategies to play a game
Output: Winner of the game
 1: if $s_1$ and $s_2$ are feasible then
 2:     if $s_1 \succ s_2$ then
 3:         return $s_1$
 4:     end if
 5:     if $s_2 \succ s_1$ then
 6:         return $s_2$
 7:     end if
 8:     return randomlySelect($\{s_1, s_2\}$)
 9: end if
10: if $s_1$ is feasible and $s_2$ is infeasible then
11:     return $s_1$
12: end if
13: if $s_2$ is feasible and $s_1$ is infeasible then
14:     return $s_2$
15: end if
16: if $s_1$ and $s_2$ are infeasible then
17:     return $argmin_{s\in\{s_1,s_2\}} c^v_s$
18: end if
```

Algorithm 1 shows how Cielo seeks an evolutionarily stable strategy for each application through evolutionary games. In the 0-th generation, strategies are randomly generated for each population (Line 2). In each generation (g), a series of games are carried out on every population (Lines 4 to 24). A single game randomly chooses a pair of strategies (s_1 and s_2) and distinguishes them to the winner and the loser with respect to performance objectives described in Section II (Lines 7 to 9). The loser disappears in the population. The winner is replicated to increase its population share and mutated with a certain mutation rate P_m (Lines 10 to 15). Mutation randomly chooses one of three VMs in the winner and alters its $h_{i,t}$, $c_{i,t}$ and $b_{i,t}$

values at random (Line 12).

Once all strategies play games in the population, Cielo identifies a feasible strategy whose population share (x_s) is the highest and determines it as a dominant strategy (d_i) (Lines 18 to 22). A strategy is said to be feasible if it never violates the CPU capacity constraint ($c^v = 0$, Eq. 6). It is said to be infeasible if $c^v > 0$. Cielo deploys three VMs for an application in question based on the dominant strategy.

Algorithm 2 shows how to select the winner in a game (c.f. `performGame()` in Algorithm 1). This selection process depends on the dominance relationship between given two strategies and their feasibility. A strategy s_1 is said to dominate another strategy s_2 (denoted by $i \succ j$) if both of the following conditions are satisfied [6]:

- s_1's objective values are superior than, or equal to, s_2's in all objectives.
- s_1's objective values are superior than s_2's in at least one objectives.

If given two strategies are feasible and they are non-dominated with each other, the winner is randomly chosen (Line 8). If both of them are infeasible, the one with lower constraint violation is chosen as the winner (Line 17).

V. Stability Analysis

This section analyzes Cielo's stability (i.e., reachability to at least one of Nash equilibria) by proving the state of each population converges to an evolutionarily stable equillibrium. The proof consists of three steps: (1) designing differential equations that describe the dynamics of the population state, (2) proving an strategy selection process has equilibria, and (3) proving the the equilibria are asymptotically (or evolutionarily) stable. The proof uses the following terms and variables.

- S denotes the set of available strategies. S^* denotes a set of strategies that appear in the population.
- $X(t) = \{x_1(t), x_2(t), \cdots, x_{|S^*|}(t)\}$ denotes a population state at time t where $x_s(t)$ is the population share of a strategy $s \in S$. $\sum_{s \in S^*}(x_s) = 1$.
- F_s denotes the fitness of a strategy s. It is a relative value that is determined in a game against an opponent based on the dominance relationship between them (Algorithm 2). The winner of a game earns a higher fitness than the loser.
- $p_k^s = x_k \cdot \phi(F_s - F_k)$ denotes the probability that a strategy s is replicated by winning a game against another strategy k. $\phi(F_s - F_k)$ is the probability that the fitness of s is higher than that of k.

The dynamics of the population share of s is described as:

$$\begin{aligned} \dot{x}_s &= \sum_{k \in S^*, k \neq s} \{x_s p_k^s - x_k p_s^k\} \\ &= x_s \sum_{k \in S^*, k \neq s} x_k\{\phi(F_s - F_k) - \phi(F_k - F_s)\} \end{aligned} \quad (12)$$

Note that if s is strictly dominated, $x_s(t)_{t\to\infty} \to 0$.

Theorem 2: The state of a population converges to an equilibrium.

Proof: It is true that different strategies have different fitness values. In other words, only one strategy has the highest fitness among others. Given Theorem 1, assuming that $F_1 > F_2 > \cdots > F_{|S^*|}$, the population state converges to an equilibrium: $X(t)_{t\to\infty} = \{x_1(t), x_2(t), \cdots, x_{|S^*|}(t)\}_{t\to\infty} = \{1, 0, \cdots, 0\}$. ■

Theorem 3: The equilibrium found in Theorem 2 is asymptotically stable.

Proof: At the equilibrium $X = \{1, 0, \cdots, 0\}$, a set of differential equations can be downsized by substituting $x_1 = 1 - x_2 - \cdots - x_{|S^*|}$

$$\dot{z}_s = z_s[c_{s1}(1 - z_s) + \sum_{i=2, i\neq s}^{|s^*|} z_i \cdot c_{si}], \quad s, k = 2, ..., |S^*| \quad (13)$$

where $c_{sk} \equiv \phi(F_s - F_k) - \phi(F_k - F_s))$ and $Z(t) = \{z_2(t), z_3(t), \cdots, z_{|S^*|}(t)\}$ denotes the corresponding downsized population state. Given Theorem 1, $Z_{t\to\infty} = Z_{eq} = \{0, 0, \cdots, 0\}$ of $(|S^*| - 1)$-dimension.

If all Eigenvalues of Jaccobian matrix of $Z(t)$ has negative real parts, Z_{eq} is asymptotically stable. The Jaccobian matrix J's elements are

$$\begin{aligned} J_{sk} &= \left[\frac{\partial \dot{z}_s}{\partial z_k}\right]_{|Z=Z_{eq}} \\ &= \left[\frac{\partial z_s[c_{s1}(1 - z_s) + \sum_{i=2, i\neq s}^{|S^*|} z_i \cdot c_{si}]}{\partial z_k}\right]_{|Z=Z_{eq}} \\ & for\ s, k = 2, ..., |S^*| \end{aligned} \quad (14)$$

Therefore, J is given as follows, where $c_{21}, c_{31}, \cdots, c_{|S^*|1}$ are J's Eigenvalues.

$$J = \begin{bmatrix} c_{21} & 0 & \cdots & 0 \\ 0 & c_{31} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & c_{|S^*|1} \end{bmatrix} \quad (15)$$

$c_{s1} = -\phi(F_1 - F_s) < 0$ for all s; therefore, $Z_{eq} = \{0, 0, \cdots, 0\}$ is asymptotically stable. ■

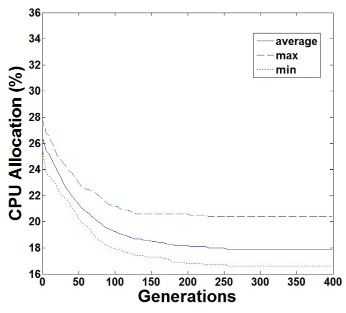
VI. Simulation Evaluation

This paper uses a simulated cloud data center that consists of 100 hosts in a 10×10 grid topology. The grid topology is chosen based on recent findings on efficient topology configurations in a cloud [7], [8]. This paper also assumes five different types of applications. Table I shows the message arrival rate (the number of incoming messages per second) and message processing time (in second) for each application type. This configuration follows Zipf's law. This paper simulates 40 application instances for each type (200 instances in total). In Cielo, the number of strategies is 100 in each population. Mutation rate (P_m in Algorithm 1) is set to 0.01. The maximum number of generations is 400. Every simulation result is the average with 20 independent simulation runs.

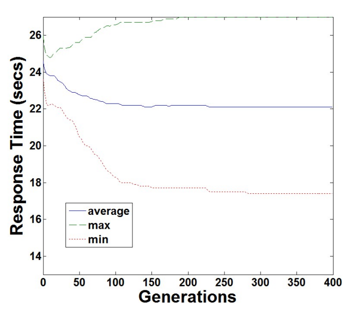
TABLE I: Message Arrival Rate and Processing Time

Application type	1	2	3	4	5
Message arrival rate	110	70	40	20	10
Web server	0.02	0.02	0.04	0.04	0.08
App server	0.03	0.08	0.04	0.13	0.11
DB server	0.05	0.05	0.12	0.08	0.11

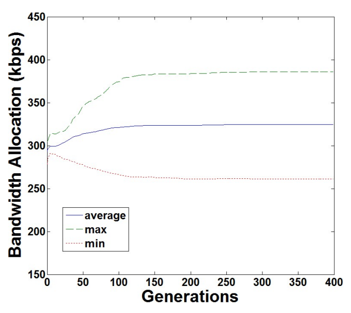
Figs. 3 to 12 illustrate how Cielo evolves deployment strategies through generations for applications and improve



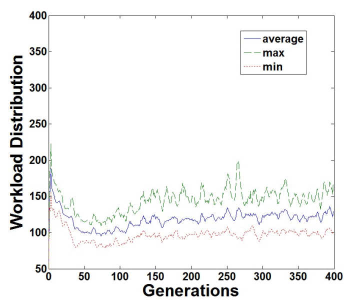
(a) CPU allocation



(b) Response time

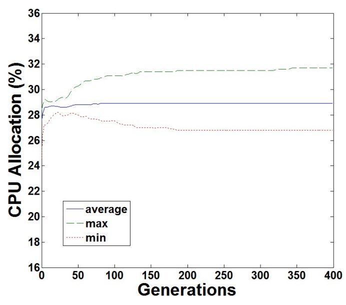


(c) Bandwidth allocation

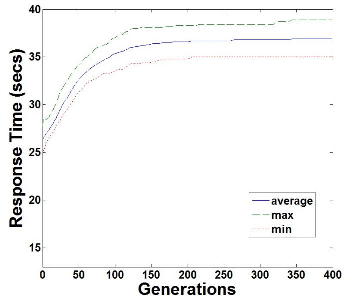


(d) Workload distribution

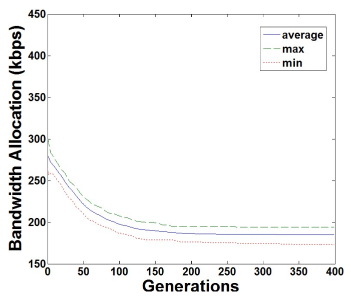
Fig. 3: With the CPU Allocation Objective Considered



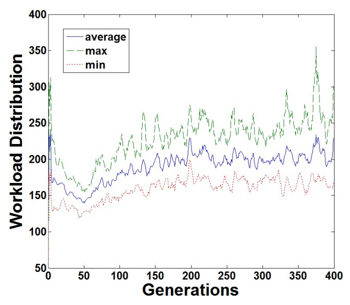
(a) CPU allocation



(b) Response time

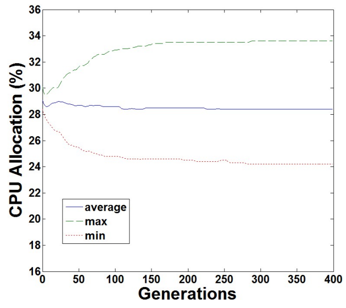


(c) Bandwidth allocation

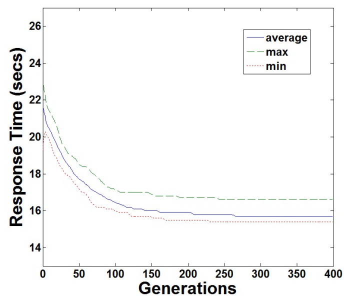


(d) Workload distribution

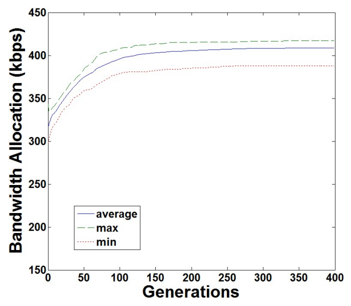
Fig. 4: With the Bandwidth Allocation Objective Considered



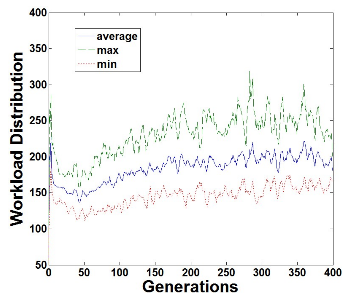
(a) CPU allocation



(b) Response time

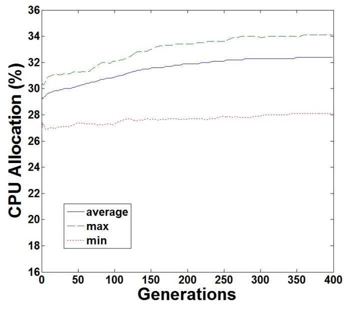


(c) Bandwidth allocation

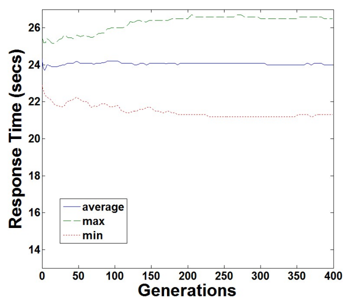


(d) Workload distribution

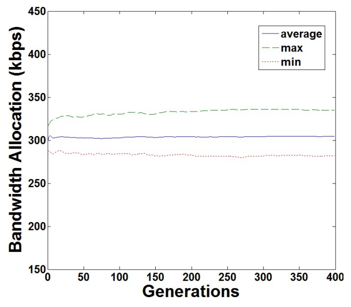
Fig. 5: With the Response Time Objective Considered



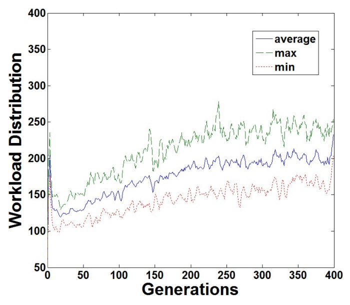
(a) CPU allocation



(b) Response time



(c) Bandwidth allocation



(d) Workload distribution

Fig. 6: With the Power Consumption Objective Considered

their performance (i.e., objective values). Figs. 3 to 7 show Cielo's performance when one of five objectives is considered. For example, Cielo considers the CPU allocation objective in Fig. 3. Figs. 11 and 12 focus on power consumption results.

In Fig. 3, CPU allocation decreases through generations because it is considered as the objective. Its average reaches 18% in the last generation, which is the best performance among Figs. 3a, 4a, 5a, 6a and 7a. The other objective values do not improve because they are not considered.

In Fig. 4, bandwidth allocation improves over time because it is considered as the objective. Its minimum value reaches 180 kbps in the last generation. This is the best performance among Figs. 3c, 4c, 5c, 6c and 7c. The improvement in bandwidth allocation contributes to the increase of response time because these two objectives conflict with each other.

In Fig. 5, response time improves over time because it is considered as the objective. Its minimum reaches 15.8 seconds in the last generation, which is the best result among Figs. 3b, 4b 5b, 6b and 7b. As response time decreases, bandwidth allocation increases because they are conflicting.

In Fig. 6, workload distribution increases because the power consumption objective is enabled. They conflict with each other. For reducing power consumption, Cielo attempts to collocate as many VMs as possible on some hosts and turn off the other hosts that operate no VMs. This is against improving the degree of workload distribution. Fig. 13 confirms this. It shows how many hosts Cielo turns off over time. When the power consumption objective is considered, the number of turned-off hosts increases up to 29. It decreases as low as 10 when the workload distribution objective is considered.

In Fig. 7, workload distribution improves because it is considered as the objective. Figs. 3 to 7 demonstrate that Cielo successfully evolves deployment strategies so that applications improve their performance with respect to a given objective.

In Figs. 8 and 9, two objectives are considered simultaneously. All five objectives are considered simultaneously in Fig. 10. As these figures show, Cielo successfully balance objective values by following on the trade-offs among objectives.

Figs. 11 and 12 illustrate how power consumption changes with different objectives considered. In all the objective settings except the workload distribution objective, power consumption decreases through generations. (As discussed above, power consumption and workload distribution conflict.) When the power consumption objective is considered, power consumption reaches 242 watts in the last generation (Fig. 11a). This is the best performance in Figs. 11 and 12.

Table II compares Cielo with well-known existing heuristics, FFA (first-fit algorithm) and BFA (best-fit algorithm), which have been widely used for VM placement in clouds [1]–[4]. FFA yields the lowest power consumption because it is designed to deploy VMs on the minimum number of hosts; however, it sacrifices performance in the other objectives. BFA is the best in workload distribution and the worst in power consumption because it is designed to deploy VMs on the hosts that maintain higher resource availability. With all five objectives considered, Cielo maintains balanced performance in between FFA and BFA while yielding the best performance in response time and bandwidth allocation.

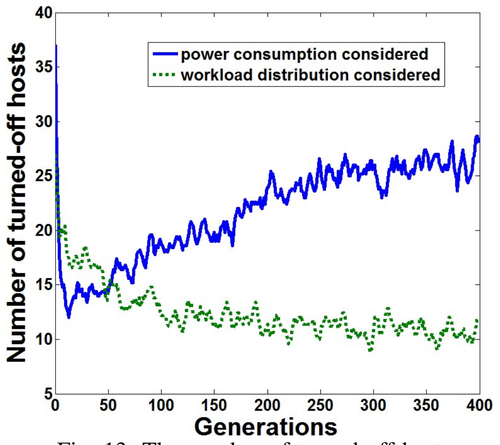


Fig. 13: The number of turned-off hosts

TABLE II: Performance of Cielo, FFA and BFA

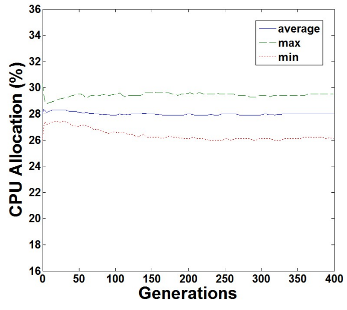
		Min	Avg	Max
Total power consumption (Watts)	Cielo	7,566	8,033	8,800
	FFA	5,742	5,766	5,769
	BFA	11,355	11,366	11,408
CPU allocation (%/host)	Cielo	68	71.6	76.6
	FFA	98	98	98
	BFA	40.8	40.8	40.8
Bandwidth allocation (Kbps/host)	Cielo	721	736	745
	FFA	821	821	821
	BFA	816	816	816
Response time (seconds)	Cielo	15.5	15.8	16.3
	FFA	21.77	21.77	21.77
	BFA	19.3	19.3	19.3
Workload distribution	Cielo	38.2	43	50
	FFA	74.6	74.8	74.9
	BFA	15	15.8	16
# of turned-off hosts	Cielo	26	29	31
	FFA	58	58	58
	BFA	0	0	0

VII. Related Work

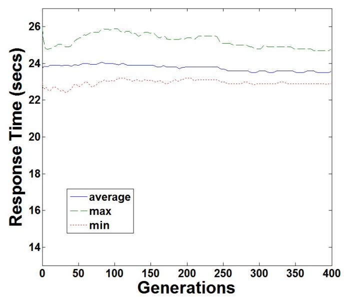
Numerous research efforts have been made to study heuristic algorithms for application placement problems in clouds; e.g., [1]–[4], [9]–[12] to name just a few. Most of those existing work assume single-tier application architecture and considers a single performance objective. For example, in [9]–[12], only energy saving is considered as the objective. In contrast, Cielo assumes a multi-tier application architecture and considers multiple objectives. It is intended to reveal the trade-off relationships among conflicting objectives.

Game theoretic algorithms have been used for a few aspects of cloud computing; e.g., application placement [13]–[15], task allocation [16] and data replication [17]. In [13]–[15], greedy algorithms seek equilibria in application placement problems with respect to multiple performance objectives. This means they do not attain the stability to reach equilibria as Cielo does.

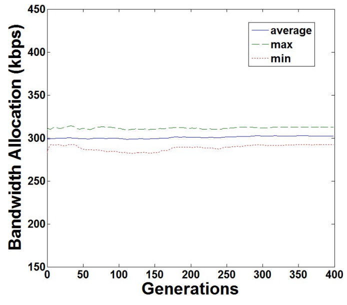
This paper reports a set of extensions to the authors' prior work [18]. For the problem formulation, this paper considers two extra objectives (bandwidth allocation and power consumption) and an extra parameter in each deployment strategy (bandwidth allocation), all of which are not studied in [18]. This paper also considers an optimization constraint in CPU allocation and investigates a constraint-handling algorithm (Algorithm 2) while no constraints are assumed in [18].



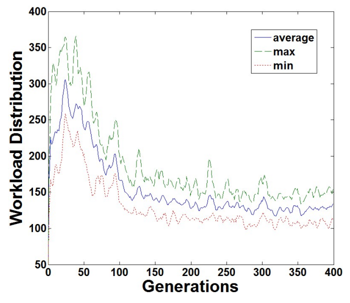
(a) CPU allocation



(b) Response time

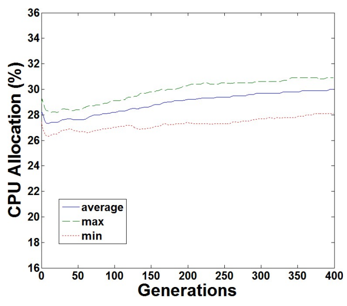


(c) Bandwidth allocation

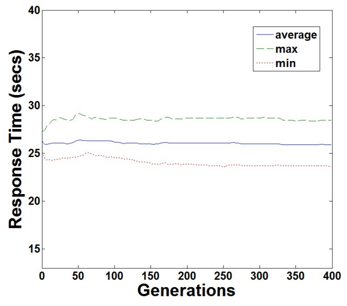


(d) Workload distribution

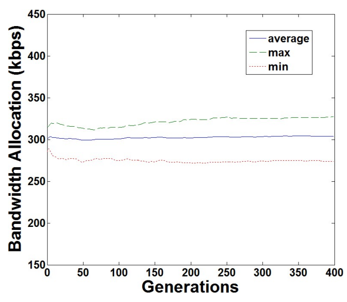
Fig. 7: With the Workload Distribution Objective Considered



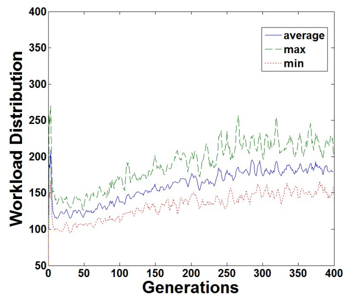
(a) CPU allocation



(b) Response time

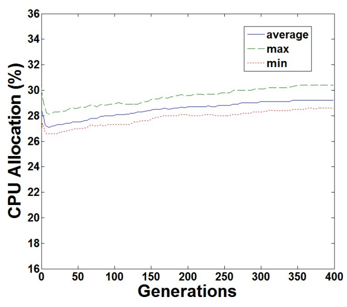


(c) Bandwidth allocation

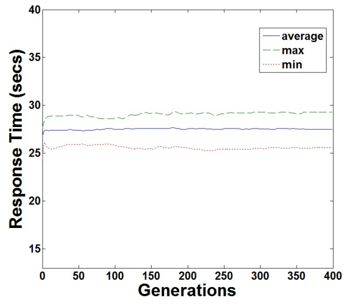


(d) Workload distribution

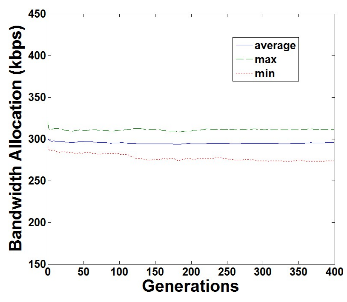
Fig. 8: With the Power Consumption and Workload Distribution Objectives Considered



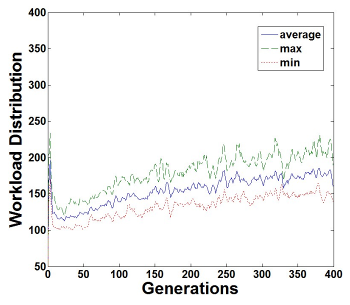
(a) CPU allocation



(b) Response time

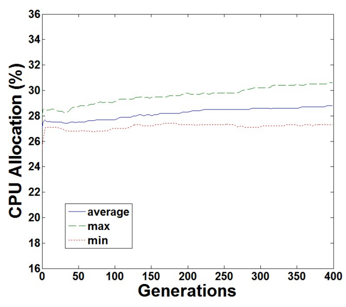


(c) Bandwidth allocation

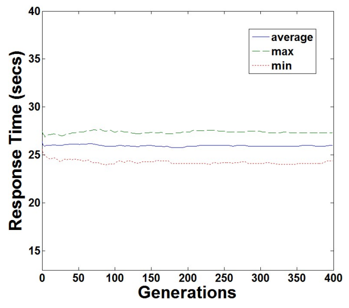


(d) Workload distribution

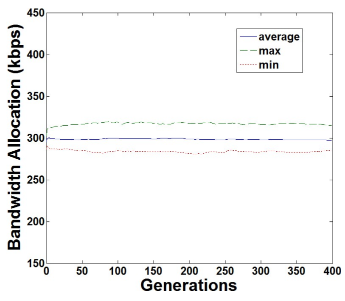
Fig. 9: With the Response Time and Bandwidth Allocation Objectives Considered



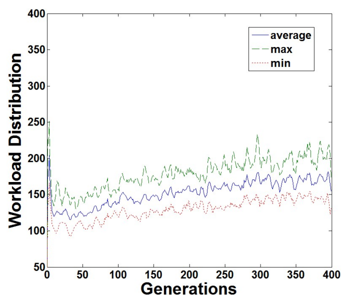
(a) CPU allocation



(b) Response time

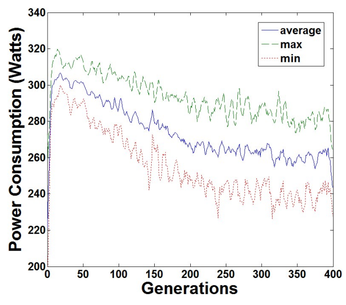


(c) Bandwidth allocation

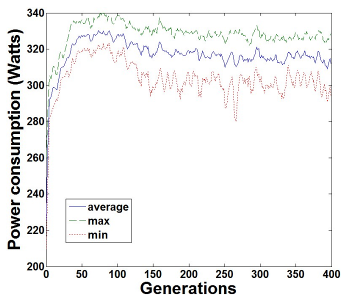


(d) Workload distribution

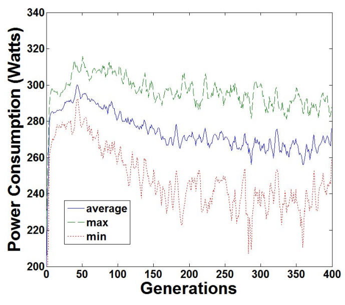
Fig. 10: With All Five Objectives Considered



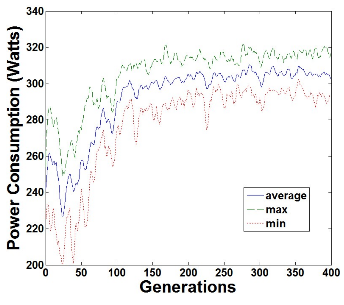
(a) Power Consumption



(b) CPU Allocation Considered

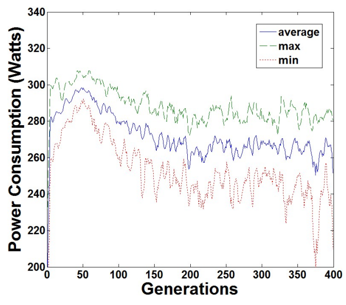


(c) Response Time Considered

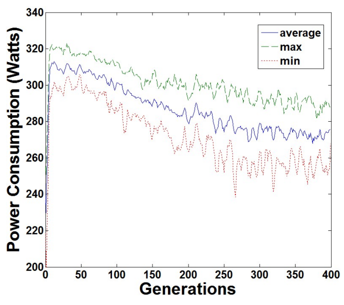


(d) Workload Considered

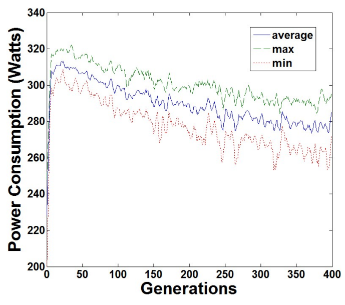
Fig. 11: Power Consumption with an Objective Considered



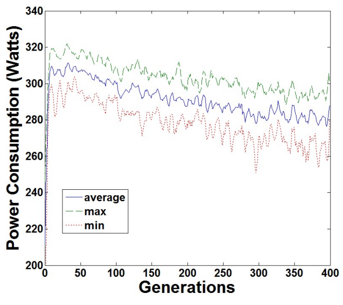
(a) Bandwidth Allocation Considered



(b) Power Consumption and Workload Considered



(c) Response Time and Bandwidth Allocation Considered



(d) All Five Objectives Considered Simultaneously

Fig. 12: Power Consumption with Multiple Objectives Considered Simultaneously

Moreover, this paper carries out larger-scale simulations with more realistic configurations than [18], which uses only five hosts and simple topologies among them.

VIII. Conclusions

This paper proposes and evaluates Cielo, an evolutionary game theoretic framework for adaptive and stable VM deployment in clouds. It theoretically guarantees that every application seeks an evolutionarily stable deployment strategy, which is an equilibrium solution under given workload and resource availability in a cloud. Simulation results verify that Cielo performs VM deployment in an adaptive and stable manner. Cielo outperforms well-known existing heuristics that have been widely used for VM placement in clouds.

References

[1] X. Lia, Z. Qiana, S. Lua, and J. Wu, "Energy efficient virtual machine placement algorithm with balanced and improved resource utilization in a data center," *Mathematical and Computer Modelling*, 58(5-6), 2013.

[2] F. Ma, F. Liu, and Z. Liu, "Multi-objective optimization for initial virtual machine placement in cloud data center," *J. Infor. and Computational Science*, vol. 9, no. 16, 2012.

[3] H. Goudarzi and M. Pedram, "Energy-efficient virtual machine replication and placement in a cloud computing system," in *Proc. IEEE Int'l Conference on Cloud Computing*, 2013.

[4] H. Casanova, M. Stillwell, and F. Vivien, "Virtual machine resource allocation for service hosting on heterogeneous distributed platforms," in *Proc. IEEE Int'l Parallel & Distributed Processing Symposium*, 2012.

[5] P. Taylor and L. Jonker, "Evolutionary stable strategies and game dynamics," *Elsevier Mathematical Biosciences*, vol. 40(1), 1978.

[6] N. Srinivas and K. Deb, "Multiobjective function optimization using nondominated sorting genetic algorithms," *Evol. Computat.*, 2(3),1995.

[7] C. Guo, H. Wu, K. Tan, L. Shiy, Y. Zhang, and S. Lu, "Dcell: A scalable and fault-tolerant network structure for data centers," in *Proc. of ACM SIGCOM*, 2008.

[8] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "Bcube: A high performance, server-centric network architecture for modular data centers," in *Proc. of ACM SIGCOM*, 2009.

[9] von Laszewski, L. Wang, A. J. Younge, and X. He, "Power-aware scheduling of virtual machines in DVFS-enabled clusters," in *IEEE International Conference on Clusters*, September 2009.

[10] D. Kliazovich, P. Bouvry, and S. U. Khan, "DENS: data center energy-efficient network-aware scheduling," *Cluster Computing*, 16(1), 2013.

[11] S. Chen, K. R. Joshi, M. A. Hiltunen, R. D. Schlichting, and W. H. Sanders, "Blackbox prediction of the impact of DVFS on end-to-end performance of multitier systems," *ACM SIGMETRICS Performance Evaluation Review*, vol. 37, no. 4, 2010.

[12] Q. Wang, Y. Kanemasa, J. Li, C. A. Lai, M. Matsubara, and C. Pu, "Impact of DVFS on n-tier application performance," in *Proc. ACM Conference on Timely Results in Operating Systems*, 2010.

[13] S. U. Khan and C. Ardil, "Energy efficient resource allocation in distributed computing systems," in *Proc. of Int'l Conference on Distributed, High-Performance and Grid Computing*, 2009.

[14] G. Wei, A. V. Vasilakos, Y. Zheng, and N. Xiong, "A game-theoretic method of fair resource allocation for cloud computing services," *J. Supercomputing*, vol. 54, no. 2, 2009.

[15] N. Doulamis, A. Doulamis, A. Litke, A. Panagakis, T. Varvarigou, and E. Varvarigos, "Adjusted fair scheduling and non-linear workload prediction for qos guarantees in grid computing," *Elsevier Computer Comm.*, vol. 30(3), 2007.

[16] R. Subrata, A. Y. Zomaya, and B. Landfeldt, "Game theoretic approach for load balancing in computational grids," *IEEE Trans. Parallel and Distributed Systems*, vol. 19, no. 1, 2008.

[17] S. Khan and I. Ahmad, "A pure Nash equilibrium based game theoretical method for data replication across multiple servers," *IEEE Trans Knowledge and Data Engineering*, vol. 21, no. 4, 2009.

[18] C. Lee, J. Suzuki, A. V. Vasilakos, Y. Yamano, and K. Oba, "An evolutionary game theoretic approach to adaptive and stable application deployment in clouds," in *Proc. IEEE Workshop on Bio-Inspired Algorithms for Distributed Systems*, 2010.