Noname manuscript No.
(will be inserted by the editor)

Adaptive and Stable Virtual Machine Placement for Power- and Performance-aware Clouds: A Hybrid Approach with Evolutionary Game Theory and Linear Programming

Yi Cheng Ren · Junichi Suzuki · Shingo Omura · Ryuichi Hosoya

Received: date / Accepted: date

Abstract This paper formulates a power- and performance-aware multiobjective virtual machine (VM) placement problem and approaches the problem with an evolutionary game theoretic algorithm that is augmented by linear programming. The proposed algorithm, called Cielo, aids cloud operators to adapt the resource allocation to VMs and their locations according to the operational conditions in a cloud (e.g., workload and resource availability) with respect to multiple conflicting objectives such as response time and power consumption. In Cielo, evolutionary multiobjective games are performed on VM configuration strategies (i.e., solution candidates) with an aid of linear programming. Cielo theoretically guarantees that each application (i.e., a set of VMs) performs an evolutionarily stable deployment strategy, which is an equilibrium solution under given operational conditions. Simulation results verify this theoretical stability analysis; applications seek equilibria to perform adaptive and evolutionarily stable deployment strategies. Linear programming allows Cielo to gain up to 38% improvement in optimality and up to 5.5x speedup in convergence speed with reasonably acceptable computational costs.

Yi Cheng Ren · Junichi Suzuki
Department of Computer Science
University of Massachusetts, Boston
Boston, MA 02125-3393, USA
E-mail: yiren001@cs.umb.edu

Junichi Suzuki
E-mail: jxs@cs.umb.edu

Shingo Omura · Ryuichi Hosoya
OGIS International, Inc.
San Mateo, CA 94402, USA
E-mail: omura@ogis-international.com

Ryuichi Hosoya
E-mail: hosoya@ogis-international.com

Keywords Cloud computing · virtual machine placement · evolutionary game theory · linear programming · multiobjective optimization

1 Introduction

It is a challenging issue for cloud operators to place applications so that the applications can satisfy given constraints in performance (e.g. response time) while maintaining their resource utilization (CPU and network bandwidth utilization) and power consumptionption. The operators are required to dynamically place applications by adjusting their locations and resource allocation according to various operational conditions such as workload and resource availability. In order to address this challenge, this paper investigates an application placement scheduler, called Cielo, which exhibits the following properties:

– *Self-optimization*: allows applications to autonomously seek their optimal placement configurations (i.e., locations and resource allocation) according to operational conditions (e.g., workload and resource availability), as adaptation decisions, under given optimization objectives and constraints.
– *Self-stabilization*: allows applications to autonomously seek stable adaptation decisions by minimizing oscillations (or non-deterministic inconsistencies) in decision making.

This project approaches the self-optimization and self-stabilization properties with evolutionary computation (EC) and evolutionary game theory (EGT), respectively. Cielo leverages EC, particularly an evolutionary multiobjective optimization algorithm (EMOA), because, in general, EMOAs are robust problem-independent search methods that seek optimal solutions (adaptation decisions) with reasonable computational costs by maintaining a small ratio of search coverage to the entire search space [1,2]. Cielo employs EGT as a means to mathematically formulate adaptive decision making and theoretically guarantee that each decision making process converges to an evolutionarily (or asymptotically) stable equilibrium where a specific (stable) adaptation decision is deterministically made under a particular set of operational conditions.

By integrating EC and EGT, Cielo provides an EGT-backed evolutionary algorithm that allows applications to (1) seek the solutions to optimally adapt their locations and resource allocation and (2) operate at equilibria by making evolutionarily stable decisions for application placement. In Cielo, each application maintains a set (or a population) of placement strategies, each of which indicates the location of and resource allocation for that application. Cielo repeatedly performs evolutionary multiobjective games on placement strategies and evolves them over generations with respect to conflicting optimization objectives including response time, resource utilization and power consumption. In each generation, Cielo runs the simplex linear programming (LP) algorithm for a small portion of the entire search space and leverages the LP-optimal local solution(s) to efficiently search a globally optimal solution.

This paper describes Cielo's algorithmic design and evaluates its adaptability and stability with a cloud data center that supports dynamic voltage and frequency scaling (DVFS) for CPUs. Simulation results demonstrate that Cielo allows applications to seek equilibria to perform evolutionarily stable placement strategies and adapt their locations and resource allocations to given operational conditions. Applications successfully leverage DVFS to balance their response time performance, resource utilization and power consumption. Linear programming aids Cielo to gain up to 38% improvement in optimality and up to 5.5x speedup in convergence speed with reasonably acceptable computational costs. Cielo's performance is evaluated in comparison to existing heuristic algorithms. Cielo outperforms a well-known multiobjective evolutionary optimization algorithm, NSGA-II [3] by 24% in optimality while maintaining 97% higher stability (lower oscillations). Cielo also outperforms two other well-known heuristics, first-fit and best-fit algorithms (FFA and BFA), which have been widely used for adaptive cloud application deployment [4–7].

2 Problem Statement

This section formulates an application placement problem to place N applications on M hosts available in a cloud data center. Each application is designed with a set of server software, following a three-tier application architecture [8, 9] (Fig. 1). Using a certain hypervisor such as Xen [10], each server is assumed to run on a virtual machine (VM) atop a host. A host can operate multiple VMs. They share resources available on their local host. Each host is assumed to be equipped with a multi-core CPU that supports DVFS in each core.

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 toward a user. (Fig. 1). This paper assumes that different applications utilize different sets of servers. (Servers are not shared by different applications.)

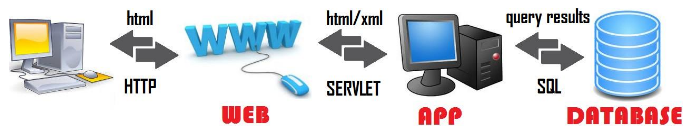


Fig. 1 Three-Tiered Application Architecture

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 the four objectives described below. Every objective is computed on an application by application basis and is to be minimized.

- *CPU allocation* (f_C): A certain CPU time share (in percentage) is allocated to each VM. (The CPU share of 100% means that a CPU core 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.

$$f_C = \sum_{t=1}^{3} c_t \tag{1}$$

  c_t denotes the CPU time share allocated to the t-th tier server in an application.
- *Bandwidth allocation* (f_B): A certain amount of bandwidth (in bits/second) is allocated to each VM. It represents 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.

$$f_B = \sum_{t=1}^{3} b_t \tag{2}$$

  b_t denotes the amount of bandwidth allocated to the t-th tier server in an application.
- *Response time* (f_{RT}): This objective indicates the time required for a message to travel from a web server to a database server:

$$f_{RT} = T^p + T^w + T^c \tag{3}$$

  T^p denotes the total time for an application to process an incoming message from a user at three servers. T^w is the waiting time for a message to be processed at servers. T^c denotes the total communication delay to transmit a message between servers. T^p, T^w and T^c are estimated with the $M/M/M$ 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_t^p denotes the time required for the t-th tier server to process a message.

$$T^p = \sum_{t=1}^{3} T_t^p \tag{4}$$

  T^w is computed as follows.

$$T^w = \frac{1}{\lambda} \sum_{t=1}^{3} \rho_0 \frac{a_t^O}{O!} \frac{\rho_t}{(1-\rho_t)^2} \tag{5}$$

$$\text{where } a_t = \lambda_t \frac{T_t^p}{c_t \cdot q_t/q_{max}},\ \rho_t = \frac{a_t}{O},\ \rho_0 = \left(\sum_{n=0}^{O-1} \frac{\rho_t^n}{n!} + \frac{\rho_t^O}{O!} \frac{1}{1-\rho_t/O} \right)^{-1}$$

λ denotes the message arrival rate for an application (i.e., the number of messages the application receives from users in the unit time). Note that $\lambda = \frac{1}{3}\sum_{t=1}^{3}\lambda_t$ where λ_t is the message arrival rate for the t-th tier server in the application. Currently, $\lambda = \lambda_1 = \lambda_2 = \lambda_3$. ρ_t denotes the utilization of a CPU core that the t-th tier server resides on. q_{max} is is the maximum CPU frequency. q_t is the frequency of a CPU core that the t-th tier server resides on. O is the total number of cores that a CPU contains.
T^c is computed as follows.

$$T^c = \sum_{t=1}^{2} T^c_{t\rightarrow t+1} \approx \sum_{t'=2}^{3} \frac{B \cdot \lambda_{t+1}}{b_t} \tag{6}$$

B is the size of a message (in bits). $T^c_{t\rightarrow t+1}$ denotes the communication delay to transmit a message from the t-th to $(t+1)$-th server. b_t denotes the bandwidth allocated to the t-th tier server (bits/second).

– *Power Consumption* (f_{PC}): This objective indicates the total power consumption (in Watts) by the CPU cores that operate three VMs in an application.

$$f_{PC} = \sum_{t=1}^{3}\left(P^{q_t}_{idle} + (P^{q_t}_{max} - P^{q_t}_{idle}) \cdot c_t \cdot \frac{q_t}{q_{max}}\right) \tag{7}$$

$P^{q_t}_{idle}$ and $P^{q_t}_{max}$ denote the power consumption of a CPU core that the t-th tier server resides on when its CPU utilization is 0% and 100% at the frequency of q_t, respectively.

Cielo considers the following four constraints.

– *CPU core capacity constraint* (C_C): The upper limit of the total share allocation on each CPU core. $c_{i,o} \leq C_C$ for all O cores on all M hosts where $c_{i,o}$ is the total share allocation on the o-th core of the i-th host. The violation of this constraint is computed as:

$$g_C = \sum_{i=1}^{M}\sum_{o=1}^{O}\left(I^C_{i,o} \cdot (c_{i,o} - C_C)\right) \tag{8}$$

$I^C_{i,o} = 1$ if $o_i > C_C$. Otherwise, $I^C_{i,o} = 0$.

– *Bandwidth capacity constraint* (C_B): The upper limit of bandwidth consumption allocated to each host. $b_i \leq C_B$ for all M hosts where b_i is the total amount of bandwidth allocated to the i-th host. The violation of this constraint is computed as:

$$g_B = \sum_{i=1}^{M}\left(I^B_i \cdot (b_i - C_B)\right) \tag{9}$$

$I^B_i = 1$ if $b_i > C_B$. Otherwise, $I^B_i = 0$.

– *Response time constraint* (C_{RT}): The upper limit of response time for each application. $f_{RT}^{i} \leq C_{RT}$ for all applications where f_{RT}^{i} is the response time of the i-th application. The violation of this constraint is computed as:

$$g_{RT} = \sum_{i=1}^{N} \left(I_i^{RT} \cdot (f_{RT}^{i} - C_{RT}) \right) \tag{10}$$

$I_i^{RT} = 1$ if $f_{RT}^{i} > C_{RT}$. Otherwise, $I_i^{RT} = 0$.

– *Power consumption constraint* (C_{PC}): The upper limit of power consumption for each application. $f_{PC}^{i} \leq C_{PC}$ for all N applications where f_{PC}^{i} is the power consumption of the i-th application. The violation of power consumption constraint is computed as:

$$g_{PC} = \sum_{i=1}^{N} \left(I_i^{PC} \cdot (f_{PC}^{i} - C_{PC}) \right) \tag{11}$$

$I_i^{PC} = 1$ if $f_{PC}^{i} > C_{PC}$. Otherwise, $I_i^{PC} = 0$.

3 Background: Evolutionary Game Theory

In a conventional game in the game theory, the objective of a rational player is to choose a strategy that maximizes its payoff. In contrast, evolutionary games are played repeatedly by players randomly drawn from a population [11, 12]. This section overviews key elements in evolutionary games: evolutionarily stable strategies (ESS) and replicator dynamics.

3.1 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

$$U(k,\ x\ell + (1 - x)k) > U(\ell,\ x\ell + (1 - x)k) \tag{12}$$

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

If the payoff function is linear, Eq. 12 derives:

$$(1 - x)U(k, k) + xU(k, \ell) > (1 - x)U(\ell, k) + xU(\ell, \ell) \tag{13}$$

If x is close to zero, Eq. 13 derives either

$$U(k,k) > U(\ell,k) \textit{ or } U(k,k) = U(\ell,k) \textit{ and } U(k,\ell) > U(\ell,\ell) \tag{14}$$

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 other strategies that have lower population shares.

3.2 Replicator Dynamics

The replicator dynamics describes how population shares associated with different strategies evolve over time [13]. 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.

$$\dot{x}_k = x_k \cdot [U(k,X) - U(X,X)] \tag{15}$$

$\dot{x}_k$ is the time derivative of x_k. 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 want to change their strategies on Nash equilibria, every Nash equilibrium is a steady state in the replicator dynamics. As described in Section 3.1, 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.

4 Cielo: An Evolutionary Game Theoretic Scheduler for VMs

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 consists of five parameters to indicate the locations of and the resource allocation for three VMs in a particular application:

$$s(a_i) = \bigcup_{t \in 1,2,3} \left(h_{i,t},\ u_{i,t},\ c_{i,t},\ b_{i,t},\ q_{i,t} \right),\quad 1 < i < N \tag{16}$$

a_i denotes the i-th application. $h_{i,t}$ is the ID of a host that a_i's t-th tier VM is placed to. $u_{i,t}$ is the ID of a CPU core that a_i's t-th tier VM resides on in the host $h_{i,t}$. $c_{i,t}$ and $b_{i,t}$ are the CPU and bandwidth allocation for a_i's t-th tier VM. $q_{i,t}$ denotes the frequency of a CPU core that a_i's t-th tier VM resides on.

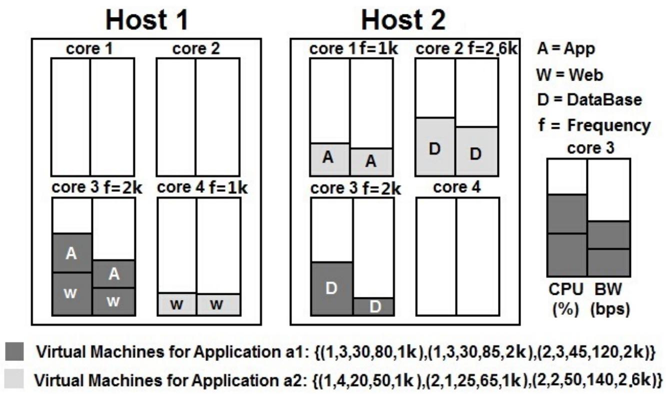

Host 1
Host 2
core 1
core 2
core 3 f=2k
core 4 f=1k
core 1 f=1k
core 2 f=2.6k
core 3 f=2k
core 4
A
W
D
A = App
W = Web
D = DataBase
f = Frequency
core 3
CPU (%)
BW (bps)
Virtual Machines for Application a1: {(1,3,30,80,1k),(1,3,30,85,2k),(2,3,45,120,2k)}
Virtual Machines for Application a2: {(1,4,20,50,1k),(2,1,25,65,1k),(2,2,50,140,2.6k)}

Fig. 2 Example Deployment Strategies

Fig. 2 shows two example strategies for two applications (a_1 and a_2) ($N = 2$). Four cores are available in each of two hosts ($M = 3$ and $O = 2$). a_1's strategy, $s(a_1)$, places the first-tier VM on the third core in the first host ($h_{1,1} = 1$ and $u_{1,1} = 3$). The 30% time share of the CPU core and 80 Kbps bandwidth are allocated to the VM ($c_{1,1} = 30$ and $b_{1,1} = 80$). The VM requires the frequency of 1 GHz for the CPU core ($q_{1,1} = 1k$). The second-tier VM of a_1 is placed on the third core in the first host ($h_{1,2} = 1$, $u_{1,2} = 3$). 30% of the CPU core time and 85 Kbps bandwidth are allocated to the VM ($c_{1,2} = 30$ and $b_{1,2} = 85$). The VM requires the frequency of 2 GHz for the CPU core ($q_{1,2} = 2k$). The third-tier VM of a_1 requires the frequency of 2 GHz ($q_{1,3} =$

$2k$) on the third core of the second host ($h_{1,3} = 2$, $u_{1,3} = 3$). 45% of the CPU core time and 120 Kbps bandwidth are allocated to the VM ($c_{1,3} = 45$ and $b_{1,3} = 120$). If multiple VMs are placed on a CPU core, the core operates at the highest required frequency. For example, on the third core of the first host, two VMs requires 1 GHz and 2 GHz. Thus, the core operates at 2 GHz.

Given $s(a_1)$, a_1's objective values for CPU and bandwidth allocation are 105% (30 + 30 + 45) and 285 kbps (80 + 85 + 120). Assuming the CPU core capacity constraint $C_C = 100\%$ (Eq. 8), it is satisfied on every core ($g_C = 0$). For example, on the third core of the first host, the total share allocation $c_{1,3}$ is 60% (30% + 30%).

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:       **if** random() $\leq P_l$ **then**
7:          $d_i \leftarrow$ linearProgramming($\mathcal{P}_i$)
8:       **else**
9:          **for** $j = 1$ to $|\mathcal{P}_i|/2$ **do**
10:            $s_1 \leftarrow$ randomlySelect($\mathcal{P}_i$)
11:            $s_2 \leftarrow$ randomlySelect($\mathcal{P}_i$)
12:            $\{winner, loser\} \leftarrow$ performGame($s_1$, $s_2$)
13:            $replica \leftarrow$ replicate($winner$)
14:            **for each** parameter $v$ in $replica$ **do**
15:               **if** random() $\leq P_m$ **then**
16:                  $replica \leftarrow$ mutate($replica$, $v$)
17:               **end if**
18:            **end for**
19:            $winner' \leftarrow$ performGame($loser$, $replica$)
20:            $\mathcal{P}_i \setminus \{s_1, s_2\}$
21:            $\mathcal{P}'_i \cup \{winner, winner'\}$
22:         **end for**
23:         $\mathcal{P}_i \leftarrow \mathcal{P}'_i$
24:         $d_i \leftarrow argmax_{s \in \mathcal{P}_i} x_s$
25:         **while** $d_i$ is infeasible **do**
26:            $\mathcal{P}_i \setminus \{d_i\}$
27:            $d_i \leftarrow argmax_{s \in \mathcal{P}_i} x_s$
28:         **end while**
29:         Deploy VMs for the current application based on $d_i$.
30:      **end if**
31:   **end for**
32:   $g = g + 1$
33: **end while**
```

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 of N populations $\{\mathcal{P}_1, \mathcal{P}_2, ..., \mathcal{P}_N\}$ (Line 2). Those strategies may or may not be *feasible*. Note that a strategy is said to be feasible if it violates none of four constraints described in Section 2. A strategy is said to be *infeasible* if it violates at least one constraint.

In each generation (g), under the probability of $1 - P_l$, a series of games are carried out on every population (Lines 8 to 30). 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 2 (Lines 10 to 12). The loser disappears in the population. The winner is replicated to increase its population share and mutated with polynomial mutation [14] (Lines 13 to 18). Mutation randomly chooses a parameter (or parameters) in a given strategy with a certain mutation rate P_m and alters its/their value(s) at random (Lines 14 to 18). Then, another game is performed between the loser and the mutated winner (Line 19). This is intended to select the top two of three strategies (winner, loser and mutated winner).

Once all strategies play games in the population, BitC identifies a feasible strategy whose population share (x_s) is the highest and determines it as a dominant strategy (d_i) (Lines 24 to 28). Cielo deploys three VMs for an application in question based on the dominant strategy (Line 29).

A game is carried out based on the superior-inferior relationship between given two strategies and their feasibility (`performGame()` in Algorithm 1). If a feasible strategy and an infeasible strategy participate in a game, the feasible one always wins the game. If both strategies are feasible, they are compared based on the notion of *Pareto dominance* [15], in which a strategy s_1 is said to dominate another strategy s_2 if both of the following conditions hold:

– 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.

The dominating strategy wins a game over the dominated one. If two strategies are non-dominated with each other, the winner is randomly selected.

If both strategies are infeasible in a game, they are compared based on their constraint violation. An infeasible strategy s_1 wins a game over another infeasible strategy s_2 if both of the following conditions hold:

– s_1's constraint violation is lower than, or equal to, s_2's in all constraints.
– s_1's constraint violation is lower than s_2's in at least one constraints.

In each generation (g), Cielo performs the simplex linear programming (LP) algorithm on each population ($\mathcal{P}_i$) with the probability of P_l (Lines 6 to 7). LP guarantees to find the optimal solution for a single objective function as far as it exists; however, it cannot consider multiple objectives separately. Therefore, Cielo executes LP in one of the following two methods subject to the four constraints described in Section 2 (C_C,C_B,C_{RT} and C_{PC}).

– Single objective method: Cielo executes LP with one of four objectives described in Section 2 (f_C, f_B, f_{RT} or f_{PC}). Another limitation of LP is that objective and constraint functions must be all linear. Since the objective function for response time is non-linear (Eq. 3), Cielo replaces it with a linearly approximated function:

$$f_{RT}^{LP} = \sum_{t=1}^{3} \left\{ T_t^p + \frac{1}{O}(T_t^p \lambda_t - c_t \frac{q_t}{q_{max}}) + (B\lambda_t - b_t) \right\} \tag{17}$$

– Weighted sum method: Cielo executes LP with the following objective function, which aggregates four objective functions (f_C, f_B, f_{RT}^{LP} or f_{PC}) as a weighted sum. w denotes a weight value for a particular objective function. Each objective value (f_i) is normalized.

$$f_{WS} = \sum_{i=1}^{4} w_i f_i \tag{18}$$

Once LP finds the optimal solution for a population, Cielo treats it as the dominant strategy for that population (Line 7). Note that the linear programming rate (P_l) is intended to be low. Cielo executes LP for a small portion of the entire problem (i.e., for a small number of populations: $P_l \times N$ populations) and leverages the LP-optimal solution(s) to boost convergence speed as well as the quality of the set of dominant strategies d_1, d_2, ... , d_N).

5 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 a set of differential equations that describe the dynamics of the population state (or strategy distribution), (2) proving an strategy selection process has equilibria and (3) proving the the equilibria are asymptotically stable (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 is the fitness of a strategy s. It is a relative value determined in a game against an opponent based on the dominance relationship between them. 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 follows.

$$\dot{x}_s = \sum_{k \in S^*, k \neq s} \{x_s p_k^s - x_k p_s^k\} = x_s \sum_{k \in S^*, k \neq s} x_k \{\phi(F_s - F_k) - \phi(F_k - F_s)\} \tag{19}$$

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 $x_1 = 1 - x_2 - \cdots - x_{|S^*|}$

$$\dot{z}_s = z_s[c_{s1}(1 - z_s) + \sum_{i=2, i\neq s}^{|s^*|} z_i \cdot c_{si}], \quad s, k = 2, ..., |S^*| \tag{20}$$

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

$$J_{sk} = \left[\frac{\partial \dot{z}_s}{\partial z_k}\right]_{|Z=Z_{eq}} = \left[\frac{\partial z_s[c_{s1}(1 - z_s) + \sum_{i=2, i\neq s}^{|S^*|} z_i \cdot c_{si}]}{\partial z_k}\right]_{|Z=Z_{eq}} \tag{21}$$
$$\text{for } s, k = 2, ..., |S^*|$$

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

$$J = \begin{bmatrix} c_{21} & 0 & \cdots & 0 \\ 0 & c_{31} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & c_{|S^*|1} \end{bmatrix} \tag{22}$$

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

6 Simulation Evaluation

This section evaluates Cielo, particularly in its optimality and stability, through simulations.

6.1 Simulation Configurations

This paper simulates a cloud data center that consists of 100 hosts in a 10×10 grid topology ($M = 100$). The grid topology is chosen based on recent findings on efficient topology configurations in clouds [16,17]. This paper also assumes five different types of applications. Table 6.1 shows the message arrival rate (i.e., the number of incoming messages per second) and message processing time (in second) for each type of applications. This configuration follows Zipf's law [18,19]. This paper simulates 40 application instances for each application type (200 application instances in total; $N = 200$).

Application type	1	2	3	4	5
Message arrival rate (λ in Eq. 6)	110	70	40	20	10
Web server (T_1^p in Eq. 4)	0.02	0.02	0.04	0.04	0.08
App server (T_2^p in Eq. 4)	0.03	0.08	0.04	0.13	0.11
DB server (T_3^p in Eq. 4)	0.05	0.05	0.12	0.08	0.11

Table 1 Message Arrival Rate and Message Processing Time

Each host is simulated to operate an Intel Core2 Quad Q6700 CPU, which is a quad-core CPU that has five frequency and voltage operating points (P-states). Table 2 shows the power consumption at each P-state under the 0% and 100% CPU utilization [20]. This setting is used in Eq. 7 to compute power consumption objective values.

P-state	Frequency (q)	P^q_{idle}	P^q_{max}
p1	1.600 GHz	82.70 W	88.77 W
p2	1.867 GHz	82.85 W	92.00 W
p3	2.113 GHz	82.91 W	95.50 W
p4	2.400 GHz	83.10 W	99.45 W
p5	2.670 GHz	83.25 W	103.00 W

Table 2 P-states in Intel Core2 Quad Q6700

Table 3 shows the parameter settings for Cielo. Mutation rate is set to $1/v$ where v is the number of parameters in a strategy. ($v = 15$ as shown in Eq. 16). Every simulation result is the average with 20 independent simulation runs.

Comparative study is carried out for the following variants of Cielo.

- Cielo-BASE: Cielo with linear programming (LP) disabled
- Cielo-LP$_\mathrm{C}$: Cielo with LP that uses the CPU consumption objective
- Cielo-LP$_\mathrm{B}$: Cielo with LP that uses the bandwidth consumption objective
- Cielo-LP$_\mathrm{PC}$: Cielo with LP that uses the power consumption objective
- Cielo-LP$_\mathrm{RT}$: Cielo with LP that uses the response time objective
- Cielo-LP$_\mathrm{WS}$: Cielo with LP that uses a weighted sum of objective values as an objective (Eq. 18).

Parameter	Value
Number of hosts (M)	100
Number of CPU cores per host (O in Eq. 6)	4
Number of applications (N)	200
Number of generations (G_{max} in Algo. 1)	500
Population size ($\|\mathcal{P}_i\|$ in Algo. 1)	100
Linear programming rate (P_l in Algo. 1)	0.005, 0.05, 0.1, 0.5
Mutation rate (P_m in Algo. 1)	$1/v$
Reference point for HV computation	f_C=600, f_B=1000, f_{PC}=2000, f_{RT}=1000

Table 3 Parameter Settings for Cielo

Cielo's variants are compared with and without constraints (C_M and C_∞ in Table 4). Constraints are enabled unless otherwise noted.

Constraint Combinations	C_C (%)	C_B (Kbps)	C_{PC} (W)	C_{RT} (ms)
C_∞	∞	∞	∞	∞
C_M	100	1,000	400	40

Table 4 Constraint Combinations

Cielo is evaluated in comparison with the simplex LP algorithm as well as NSGA-II, which is a well-known multiobjective evolutionary algorithm [3]. Simplex is implemented with GNU Linear Programming Kit (GLPK)[1] and Xypron[2]. Cielo and NSGA-II use the same parameter settings shown in Table 3. All other NSGA-II settings are borrowed from [3]. Both Cielo and NSGA-II are implemented with jMetal [21]. Moreover, Cielo is compared to well-known heuristics, first-fit and best-fit algorithms (FFA and BFA), which have been widely used for adaptive cloud application deployment [4–7]. All simulations were carried out with a Java VM 1.7 on a Windows 8.1 PC with a 3.6 GHz AMD A6-5400K APU and 6 GB memory space.

6.2 Simulation Results

Table 5 examines how a mutation-related parameter, called distribution index (η_m in [14]), impacts the performance of Cielo. Cielo-BASE is used in this evaluation. This parameter controls how likely a mutated strategy is similar to its original. (A higher distribution index makes a mutant more similar to its original.) In Table 5, the performance of Cielo is evaluated with the hypervolume measure that a set of dominant strategies yield in the last (500th) generation. The hypervolume metric indicates the union of the volumes that a given set of solutions dominates in the objective space [22]. A higher hypervolume means that a set of solutions is more optimal. As shown in Table 5, Cielo

[1] http://www.gnu.org/software/glpk/
[2] http://glpk-java.sourceforge.net/

yields the best performance with the distribution index value of 40. Thus, this parameter setting is used in all successive simulations.

Distribution Index	HV	Distribution Index	HV
30	0.823	35	0.828
40	0.830	45	0.827
50	0.825		

Table 5 Impacts of Distribution Index Values on Hypervolume (HV) Performance

Table 6 illustrates how the computational costs of Cielo changes according to the linear programming (LP) rate (P_l in Algorithm 1 and Table 3). When the LP rate is 0 %, Cielo works as Cielo-BASE. Its execution time is approximately 6.5 minutes to run 500 generations. This is an acceptable cost for configuring 3,000 parameters for 200 applications (15 parameters/application $\times$ 200 applications).

Table 6 also shows the computational costs of Cielo-LP$_{\text{WS}}$ under different LP rates from 0.5% to 50%. Probabilistically, LP is applied on one of 200 applications in each generation under the LP rate of 0.5%. If Cielo-LP$_{\text{WS}}$ considers all four objectives in its weighted-sum function (Eq. 18), its execution time exceeds one hour even if the LP rate is 0.5%. The computation of response time is a major contribution to this cost. If Cielo-LP$_{\text{WS}}$ considers three objectives besides the response time objective, its computational costs are reasonably acceptable under the LP rates of 10% or lower. As LP rate increases from 0% to 10%, execution time approximately doubles; however, it is still less than 14 minutes. In all successive simulations, Cielo-LP$_{\text{WS}}$ does not consider the response time objective when it runs LP.

Algorithms	LP rate (%)	Execution time (s)	Speedup
Cielo-BASE	0%	393	1.0
Cielo-LP$_{\text{WS}}$ w/ f_{RT}^{LP}	0.5%	>3600	<0.10
	5%	>3600	<0.10
	10%	>3600	<0.10
	50%	>3600	<0.10
Cielo-LP$_{\text{WS}}$ w/o f_{RT}^{LP}	0.5%	435	0.90
	5%	556	0.70
	10%	832	0.47
	50%	>3600	<0.10

Table 6 Impacts of LP Rates on the Execution Time Performance

Figs. 3 to 6 show a series of boxplots for the objective values that Cielo-BASE and Cielo-LP$_{\text{WS}}$ yield in 20 simulation runs. Each boxplot illustrates the maximum and minimum objective values as well as the first, second and third quartiles of objective values. Cielo-LP$_{\text{WS}}$ outperforms Cielo-BASE in all objectives except the response time objective. Note that Cielo-LP$_{\text{WS}}$ does

	$\mathrm{LP_C}$	$\mathrm{LP_{BW}}$	$\mathrm{LP_{PC}}$	$\mathrm{LP_{WS}}$	BASE
f_C (%)	15.04 (38.6%)	23.03	18.13	16.53 (32.5%)	24.48
f_B (Kbps)	155.87	150.12 (3.6%)	151.43	150.44 (3.4%)	155.76
f_{PC} (W)	296.98	294.78	274.23 (6.9%)	283.25 (3.8%)	294.54
f_{RT} (ms)	48.76	47.54	46.30	41.89	39.39
Distance	0.510 (12.8%)	0.574 (1.9%)	0.515 (12.0%)	0.508 (13.2%)	0.585

Table 7 Performance Improvement of Cielo-LP against Cielo-BASE

not consider the response time objective when it runs LP. Cielo-$\mathrm{LP_{WS}}$ yields better/lower objective values as its LP rate increases from 0.5% to 10%. Since the computational cost is reasonably acceptable under the LP rate of 10% (Table 6), LP rate is set to 10% in all successive simulations.

Table 7 shows the average objective value that each of Cielo's variants yields in the last generation. A number in parentheses indicates a performance gain against CIelo-BASE. Cielo always gains performance improvement on an objective(s) that LP is applied to. Cielo-$\mathrm{LP_C}$ gains the highest performance improvement (38.6%). The performance improvement of Cielo-$\mathrm{LP_{WS}}$ is 13.3% on average. Table 7 also depicts the distance from Cielo's solution to the utopian point, which is (0, 0, 0, 0), in the objective space. Manhattan distance is used as a distance metric here. Cielo-$\mathrm{LP_{WS}}$'s solution is 13.2% closer to the utopian point, which means 13.2% better optimized than Cielo-BASE's. Tables 6 and 7 demonstrate that LP successfully aids Cielo to boost the optimality of its solution with reasonable computational costs.

Table 8 compares the objective values of Cielo-$\mathrm{LP_{WS}}$ with the optimal results that LP finds with and without constraints. Here, LP is used to solve the entire problem. Since LP can consider only one objective in a single simulation run, Table 8 shows the optimal objective values in four objectives by running LP four times ("LP only" in Table 8). LP is also configured to use a weighted-sum function that aggregates four objective values. In this configuration, LP runs once to obtain four objective values ("LP only (WS)" in Table 8). Those values are not guaranteed to be optimal. With constraints disabled (C_∞), Cielo-$\mathrm{LP_{WS}}$ and LP with a weighted-sum function produce higher/worse objective values than LP's optimal values because the two algorithms consider multiple objectives simultaneously. Note that Cielo-$\mathrm{LP_{WS}}$'s performance is very close to LP in CPU and bandwidth allocation while it is inferior to LP in power consumption. Cielo-$\mathrm{LP_{WS}}$ outperforms LP with a weighted-sum function in three of four objectives.

With constraints enabled (C_M), LP obtains the optimal objective values in CPU allocation and power consumption. However, it fails to obtain the optimal values in two other objectives within the timeout period of one hour. LP with a weighted-sum function fails to complete its execution within the timeout period. Cielo-$\mathrm{LP_{WS}}$'s performance is very close to LP in CPU allocation while it is inferior to LP in power consumption.

Table 8 also shows the execution time for the three algorithms. LP's execution time indicates the total execution time to run four simulation runs. With constraints disabled, LP and LP with a weighted-sum function are significantly efficient compared to Cielo-LP$_{\text{WS}}$. With constraints enabled, their efficiency dramatically degrades. LP's execution time is five seconds and six seconds to obtain the objective values of CPU allocation and power consumption, respectively. Therefore, LP's execution time is greater than 7,211 seconds. The existence of constraints greatly impacts the execution time of LP and LP with a weighted-sum function while it does not impact Cielo-LP$_{\text{WS}}$. Cielo-LP$_{\text{WS}}$'s execution time increases only two seconds by enabling constraints. In summary, Cielo-LP$_{\text{WS}}$ yields near LP-optimal performance in some objectives and it is robust against the existence of constraints.

	f_C	f_B	f_{PC}	f_{RT}	Exec. time (s)
Cielo-LP$_{\text{WS}}$: C_∞	16.82	150.68	290.33	44.47	832
LP only: C_∞	15.00	150	54.56	8.06	23
LP only (WS): C_∞	41.10	450	499.97	8.12	7
Cielo-LP$_{\text{WS}}$: C_M	16.53	150.44	283.25	41.89	834
LP only: C_M	15.00	—	54.56	—	>7,211
LP only (WS): C_M	—	—	—	—	>3,600

Table 8 Comparison of Objective Values and Execution Time between Cielo-LP$_{\text{WS}}$ and Linear Programming

Fig. 7 measures the hypervolume that a set of dominant strategies produces at each generation in Cielo-BASE and Cielo-LP$_{\text{WS}}$ and illustrates how it changes over generations. Cielo-LP$_{\text{WS}}$ consistently yields higher hypervolume than Cielo-BASE over generations. Consistent with the results in Table 7, Fig. 7 confirms that Cielo-LP$_{\text{WS}}$ is better optimized than Cielo-BASE.

Table 9 compares the convergence speed of Cielo-LP$_{\text{WS}}$ and Cielo-BASE based on the hypervolume measurement in Fig. 7. Cielo-LP$_{\text{WS}}$ and Cielo-BASE spend 8 and 16 generations, respectively, to reach the hypervolume of 0.37. The speedup of Cielo-LP$_{\text{WS}}$ is 2.0. Cielo-BASE reaches the hypervolume of 0.3858 in 500 generations. In contrast, Cielo-LP$_{\text{WS}}$ requires only 91 generations to reach the same hypervolume, thereby yielding 5.5x speedup. These results illustrate that LP successfully aids Cielo to boost its convergence speed.

	Hypervolume				
	0.37	0.375	0.38	0.385	0.3858
Cielo-BASE	16	41	100	361	500
Cielo-LP$_{\text{WS}}$	8	14	37	79	91
Speedup	2.0	2.9	2.7	4.6	5.5

Table 9 Comparison of Convergence Speed between Cielo-BASE and Cielo-LP$_{\text{WS}}$

Table 10 compares Cielo with NSGA-II along with FFA and BFA based on their minimum, average and maximum objective values. Cielo outperforms NSGA-II in all objectives except response time. Considering all four objectives, Cielo yields 24.67% higher performance than NSGA-II. FFA and BFA produce two extreme results. FFA yields the lowest power consumption (59.61 Watts) because it is designed to place VMs on the minimum number of hosts; however, it sacrifices the other objectives. BFA performs the best in CPU allocation (28.28%) because it is designed to place VMs on the hosts that maintain higher resource availability. Cielo maintains balanced objective values in between FFA and BFA while performing better in response time, CPU allocation and bandwidth allocation.

Objectives		Minimum	Average	Maximum
CPU allocation (%/app)	Cielo-LP$_{WS}$	16.43	16.53	16.68
	NSGA-II	28.86	30.29	31.35
	FFA	28.68	28.68	28.68
	BFA	28.28	28.28	28.28
Bandwidth allocation (Kbps/app)	Cielo-LP$_{WS}$	150.35	150.44	150.49
	NSGA-II	278.32	288.27	295.89
	FFA	1186	1186	1186
	BFA	1200	1200	1200
Power consumption (W/app)	Cielo-LP$_{WS}$	274.19	283.25	290.69
	NSGA-II	1245.15	1246	1246.92
	FFA	59.12	59.61	60.02
	BFA	341.74	341.85	341.95
Response time (msec/app)	Cielo-LP$_{WS}$	41.87	41.89	41.91
	NSGA-II	11.56	11.79	12.04
	FFA	109.06	109.06	109.06
	BFA	92.09	92.09	92.09

Table 10 Comparison of Objective Values among Cielo-LP$_{WS}$, NSGA-II, FFA and BFA

Table 11 shows the variance of objective values that Cielo and NSGA-II yield in 20 different simulation runs. A lower variance means higher stability (or higher similarity) in objective values (i.e., lower oscillations in objective values) among different simulation runs. Cielo maintains significantly higher stability than NSGA-II in all objectives. Cielo's average stability is 97.21% higher than NSGA-II's. This result exhibits Cielo's stability property (i.e. ability to seek evolutionarily stable strategies), which NSGA-II does not have.

	Cielo-LP$_{WS}$	NSGA-II	Difference (%)
CPU allocation	0.150	2.160	93.05%
Bandwidth allocation	0.060	5.599	98.92%
Power consumption	0.005	0.747	99.40%
Response time	0.006	0.239	97.48%
Average	0.055	2.186	97.21%

Table 11 Stability of Objective Values in Cielo-LP$_{WS}$ and NSGA-II

Fig. 8 shows two three-dimensional objective spaces that plot a set of dominant strategies obtained from individual populations at each generation. Each blue dot indicates the average objective values that dominant strategies yield at a particular generation in 20 simulation runs. The trajectory of blue dots illustrates a path through which Cielo's strategies evolve and improve objective values. Gray and red dots represent 20 different sets of objective values at the first and last generation in 20 simulation runs, respectively. While initial (gray) dots disperse (because strategies are generated at random initially), final (red) dots are overlapped in a small region. Consistent with Table 11, Fig. 8 verifies Cielo's stability: reachability to at least one Nash equilibria regardless of the initial conditions.

7 Related Work

Numerous research efforts have been made to study heuristic algorithms for application placement problems in clouds (e.g., [4–7, 23–26]). Most of them assume a single-tier application architecture and considers a single optimization objective. For example, in [23–26], only power consumption is considered as the objective. In contrast, Cielo assumes a multi-tier application architecture (i.e., three tiers in an application) and considers multiple objectives. It is designed to seek a trade-off solution among conflicting objectives.

Game theoretic algorithms have been used for a few aspects of cloud computing; for example, application placement [27–29], task allocation [30] and data replication [31]. In [27–29], greedy algorithms seek equilibria in application placement problems. This means they do not attain the stability property to reach equilibria as Cielo does.

Several genetic algorithms (e.g., [32, 33]) and other stochastic optimization algorithms (e.g., [34, 35]) have been studied to solve application placement problems in clouds. They seek the optimal placement solutions; however, they do not consider stability. In contrast, Cielo aids applications to seek evolutionarily stable solutions and stay at equilibria.

This paper is novel in that the EGT-backed evolutionary algorithm in Cielo integrates optimization and stabilization processes to seek optimal and stable solutions. Optimization and stabilization have been studied largely in isolation; few attempts have been made so far to integrate them facilitate them simultaneously, except in a very limited number of work (e.g., [36]). Evolutionary algorithms and other stochastic search algorithms often focus on optimization and fail to seek stable solutions [37–39]. As a result, they can inconsistently yield different sets of solutions in different runs/trials with the same problem settings, especially when a given problem's search space is large [40–43]. Conversely, game theoretic or EGT algorithms are often dedicated to seek stable solutions (i.e., equilibria), which are not necessarily optimal [11, 12, 44].

To the best of the authors' knowledge, this paper is the first attempt to integrate linear programming (LP) with an EGT-backed evolutionary algorithm. There exist a few research efforts to integrate LP with traditional evolutionary algorithms (e.g., [45, 46]). Cielo is similar to [45] in that both work use LP to solve a part of the entire problem and approach the rest of the problem with an eviltionary algorithm based on LP-optimal solutions. In [46], Kumar et al. solve a relaxed version of the problem with LP and use an evolutionary algorithm for the complete problem based on the LP-optimal solutions. Both of the two relevant work focus on optimization only, not stability, and consider a single objective,while Cielo considers both optimization and stability with respect to multiple objectives.

This paper reports a set of extensions to the authors' prior work [47]. For the problem formulation, this paper considers two extra optimization constraints in addition to the two constraints considered in [47]. As for the algorithmic design, this paper investigates the integration of LP with Cielo, which is out of the scope of [47]. For simulation evaluation, this paper conducts more comprehensive comparative study than [47] with extra benchmark algorithms (FFA, BFA and LP).

8 Conclusions

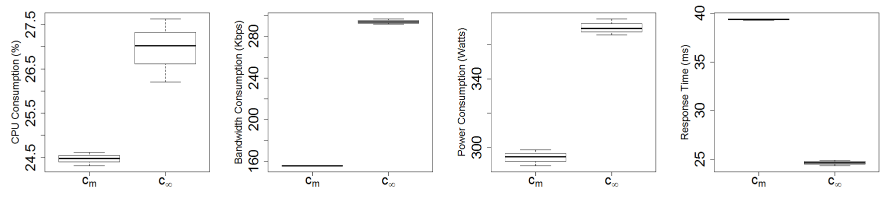
This paper proposes and evaluates Cielo, an evolutionary game theoretic algorithm for adaptive and stable virtual machine (VM) placement in DVFS-enabled clouds. It theoretically guarantees that every application seeks an evolutionarily stable deployment strategy, which is an equilibrium solution under given workload and resource availability. Simulation results verify that Cielo performs VM placement in an adaptive and stable manner. By integrating linear programming, Cielo successfully gains performance improvement in optimality and convergence speed with reasonable computational costs. Cielo outperforms existing well-known heuristics in the quality and stability of VM placement.

References

1. A. Eiben, "Evolutionary computing: the most powerful problem solver in the universe," *Dutch Mathematical Archive*, vol. 5, no. 3, pp. 126–131, 2002.
2. K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms.* John Wiley & Sons Inc, 2001.
3. K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II," in *Proc. Conf. Parallel Problem Solving from Nature*, 2000.
4. 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.
5. 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.
6. H. Goudarzi and M. Pedram, "Energy-efficient virtual machine replication and placement in a cloud computing system," in *Proc. IEEE Int'l Conf. on Cloud Comput.*, 2013.
7. 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.

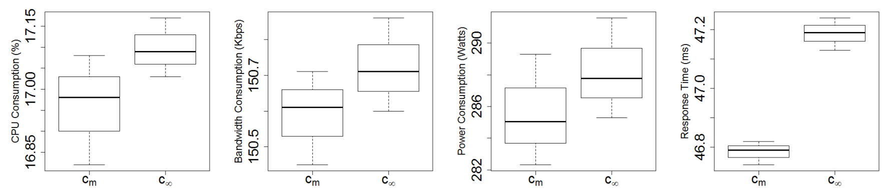
8. B. Urgaonkar, G. Pacifici, P. Shenoy, M. Spreitzer, and A. Tantawi, "An analytical model for multi-tier internet services and its applications," in *Proc. of ACM Int'l Conference on Measurement and Modeling of Computer Systems*, June 2005.
9. T. C. Shan and W. W. Hua, "Solution architecture for n-tier applications," in *Proc. of IEEE Int'l Conference on Services Computing*, September 2006.
10. P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," in *Proc. of ACM symposium on operating systems principles*, October 2003.
11. J. Weibull, *Evolutionary Game Theory.* MIT Press, 1996.
12. M. Nowak, *Evolutionary Dynamics: Exploring the Equations of Life.* Harvard University Press, 2006.
13. P. Taylor and L. Jonker, "Evolutionary stable strategies and game dynamics," *Elsevier Mathematical Biosci.*, vol. 40(1), 1978.
14. K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans Evol. Computat.*, vol. 6, no. 2, 2002.
15. N. Srinivas and K. Deb, "Multiobjective function optimization using nondominated sorting genetic algorithms," *Evol. Computat.*, 2(3),1995.
16. 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.
17. 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.
18. R. Perline, "Zipf's law, the central limit theorem, and the random division of the unit interval," *Physical Review E*, vol. 54(1), 1996.
19. J. Tatemura, W.-P. Hsiung, and W.-S. Li, "Acceleration of web service workflow execution through edge computing," in *Proc. of Int'l WWW Conference*, 2003.
20. T. Guerout, T. Monteil, G. D. Costa, R. N. Calheiros, R. Buyya, and M. Alexandru, "Energy-aware simulation with dvfs," *Simulation Modelling Practice and Theory*, vol. 39, pp. 96–91, 2013.
21. J. Durillo, A. Nebro, and E. Alba, "The jMetal framework for multi-objective optimization: Design and architecture," in *Proc. IEEE Congress on Evol. Computat.*, 2010.
22. E. Zitzler and L. Thiele, "Multiobjective optimization using evolutionary algorithms: A comparative study," in *Proc. Int'l Conf. on Parallel Problem Solving from Nature*, 1998.
23. von Laszewski, L. Wang, A. J. Younge, and X. He, "Power-aware scheduling of virtual machines in DVFS-enabled clusters," in *Proc. IEEE Int'l Conf. on Clusters*, 2009.
24. D. Kliazovich, P. Bouvry, and S. U. Khan, "DENS: data center energy-efficient network-aware scheduling," *Cluster Computing*, 16(1), 2013.
25. 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 Eval. Rev.*, vol. 37, no. 4, 2010.
26. 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.
27. S. U. Khan and C. Ardil, "Energy efficient resource allocation in distributed computing systems," in *Proc. of Int'l Conf. on Distrib., High-Perf. and Grid Comp.*, 2009.
28. 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.
29. 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.
30. R. Subrata, A. Y. Zomaya, and B. Landfeldt, "Game theoretic approach for load balancing in computational grids," *IEEE Trans. Parall. Distr.*, vol. 19, no. 1, 2008.
31. S. Khan and I. Ahmad, "A pure Nash equilibrium based game theoretical method for data replication across multiple servers," *IEEE T. Knowl. Data En.*, vol. 21, no. 4, 2009.

32. H. Wada, J. Suzuki, Y. Yamano, and K. Oba, "E3: A multiobjective optimization framework for sla-aware service composition," *IEEE Trans. Services Computing*, vol. 5, no. 3, 2012.
33. H. A. Taboada, J. F. Espiritu, and D. W. Coit, "MOMS-GA: A Multi-Objective Multi-State Genetic Algorithm for System Reliability Optimization Design Problems," *IEEE Trans. Reliability*, vol. 57, no. 1, 2008.
34. Y. Gao, H. Guan, Z. Qi, Y. Hou, and L. Liu, "A multi-objective ant colony system algorithm for virtual machine placement in cloud computing," *J. Computer and System Sciences*, vol. 79, no. 8, 2013.
35. X. Chang, B. Wang, L. Jiqiang, W. Wang, and K. Muppala, "Green cloud virtual network provisioning based ant colony optimization," in *Proc. ACM Int'l Conference on Genetic and Evol. Computat*, 2013.
36. M. Kodialam and T. Lakshman, "Detecting network intrusions via sampling: A game theoretic approach," in *Proc. of IEEE Conference on Computer and Communications Societies*, 2003.
37. R. Masuchun and W. Ferrell, "Dynamic rescheduling with stability," in *Proc. of IEEE Asian Control Conference*, 2004.
38. S. Kundu, "A note on optimizality vs. stability - a genetic algorithm based approach," in *Proc. of World Congress on Structural and Multidisciplinary Optimization*, 1999.
39. M. Marra and B. Walcott, "Stability and optimality in genetic algorithm controllers," in *Proc. of IEEE Int'l Symposium on Intelligent Control*, 1996.
40. V. Toğan and A. Daloğlu, "An improved genetic algorithm with initial population strategy and self-adaptive member grouping," *Computers and Structures*, vol. 86, no. 11-12, pp. 1204–1218, 2008.
41. O. Yugay, I. Kim, B. Kim, and F. Ko, "Hybrid genetic algorithm for solving traveling salesman problem with sorted population," in *Proc. of IEEE Int'l Conference on Convergence and Hybrid Information Technology*, 2008.
42. X. Li, J. Zhuang, S. Wang, and Y. Zhang, "A particle swarm optimization algorithm based on adaptive periodic mutation," in *Proc. of IEEE Int'l Conference on Natural Computation*, 2008.
43. W. B. Langdon and R. Poli, "Evolving problems to learn about particle swarm optimizers and other search algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 5, pp. 561–578, 2007.
44. N. Nisan, T. Roughgarden, E. Tardos, and V. Vazirani, *Algorithmic Game Theory.* Cambridge University Press, 2007.
45. P. Garbacki and V. Naik, "A hybrid linear programming and evolutionary algorithm based approach for on-line resource matching in grid environments," in *IEEE Int'l Conference on Cluster Computing and the Grid*, 2007.
46. S. K. Garg, P. Konugurthi, and R. Buyya, "A linear programming-driven genetic algorithm for meta-scheduling on utility grids," *Int'l J. of Parallel, Emergent and Distributed Systems*, vol. 26, no. 6, pp. 493–517, 2011.
47. Y. Ren, J. Suzuki, C. Lee, A. V. Vasilakos, S. Omura, and K. Oba, "Balancing performance, resource efficiency and energy efficiency for virtual machine deployment in DVFS-enabled clouds: An evolutionary game theoretic approach," in *Proc. of ACM Genetic and Evolutionary Computation Conference*, 2014.



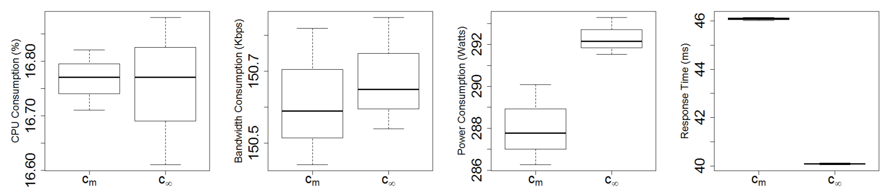
(a) f_C (b) f_B (c) f_{PC} (d) f_{RT}

Fig. 3 Cielo-BASE's Objective Values with and without Constraints (C_M and C_∞)



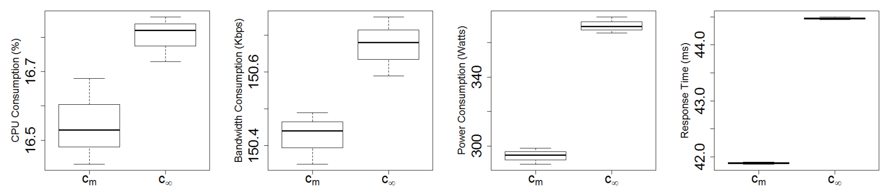
(a) f_C (b) f_B (c) f_{PC} (d) f_{RT}

Fig. 4 Cielo-LP$_{\mathrm{WS}}$'s Objective Values w/ & w/o Constraints (C_M and C_∞). LP rate: 0.5%



(a) f_C (b) f_B (c) f_{PC} (d) f_{RT}

Fig. 5 Cielo-LP$_{\mathrm{WS}}$'s Objective Values w/ & w/o Constraints (C_M and C_∞). LP rate: 5%



(a) f_C (b) f_B (c) f_{PC} (d) f_{RT}

Fig. 6 Cielo-LP$_{\mathrm{WS}}$'s Objective Values w/ & w/o Constraints (C_M and C_∞). LP rate: 10%

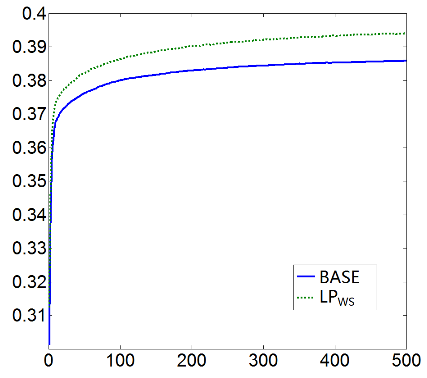
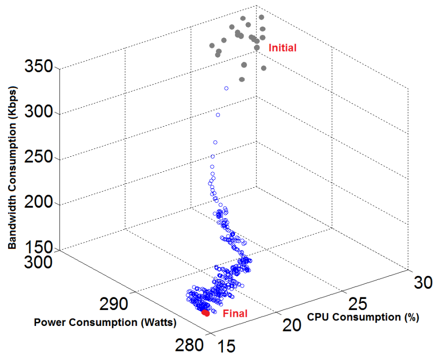
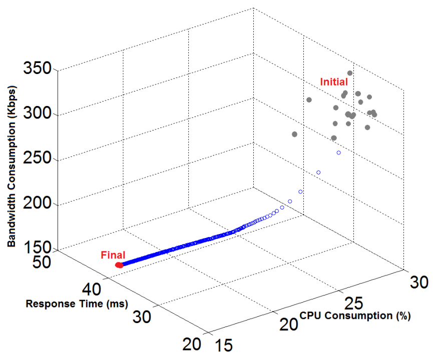


Fig. 7 Comparison of Hypervolume and Convergence Speed between BASE and LP$_{WS}$



(a) CPU allocation, Bandwidth allocation and Energy consumption



(b) CPU allocation, Bandwidth allocation and Response time

Fig. 8 Trajectory of OptCielo's Solution through Generations