Optimization of a Cloud Resource Management Problem from a Consumer Perspective

Rafaelli de C. Coutinho, Lúcia M.A. Drummond, and Yuri Frota

Institute of Computing – Fluminense Federal University, RJ
{rcoutinho,lucia,yuri}@ic.uff.br

Abstract. Cloud Computing is a distributed computing paradigm in which computing resources are available to users via Internet. Although there are many works on resource management in related literature, few of them tackle the problem from the perspective of commercial cloud consumers. In this paper, the proposed resource management problem selects cloud resources aiming to reduce the payment cost and the execution time of user applications. In order to solve this problem, an integer programming formulation and a heuristic based on Greedy Randomized Adaptive Search Procedure (GRASP) are also introduced. The model and the algorithm were tested over a set of instances constructed from requirements of real applications combined with sets of resources offered by commercial clouds. The obtained results indicate that the presented methods can be an important decision support tool for cloud consumers.

Keywords: Cloud Computing, Resource Management Problem, Optimization.

1 Introduction

Cloud Computing is a distributed computing paradigm in which computing resources are available to consumers via Internet [11]. It delivers infrastructure, platform, and software as services by signing service-level agreements (SLAs) with consumers. In terms of cloud economics, the provider should offer resource-economic services. Novel, power-efficient schemes for caching, query processing, and thermal management are mandatory due to the increasing amount of waste heat that data centers dissipate for application services. Moreover, new pricing models based on the pay-as-you-go policy are necessary to address the highly variable demand for cloud resources. On the other hand, cloud service consumers might have an SLA with a cloud service provider concerning how much bandwidth, CPU, and memory the consumer can use at any given time throughout the day. The cloud consumer has to decide how he or she wants to use cloud services; like whether or not to add in more virtual machines and at what price point that option becomes too expensive to justify the return. In this context, application schedulers have different policies that vary according to the objective function: minimize total execution time, minimize payment cost to execute, minimize the demand for power while maintaining a service-level guarantee for

D. an Mey et al. (Eds.): Euro-Par 2013 Workshops, LNCS 8374, pp. 218–227, 2014.

the consumers, balance the load on resources used while meeting the deadline constraints of the application and so forth. Independently of the objective function, resource allocation problems in clouds are NP-hard problems, so there are no efficient methods to solve them.

As cloud computing becomes more widespread and consequently energy consumption of the network and computing resources grow, more researches have been conducted on energy cost optimization. In [3], the authors analyse the energy consumption in clouds. Some problems that aim to minimize the energy costs can be seen in [7] [13] [14]. A survey about the energy cost for resource allocation and scheduling algorithms based on energy consumption are found in [4] [19]. Concerning the cloud consumer perspective, computing resources can be acquired by reservation or on-demand plans. In [6], it is observed that an interesting reservation plan can be difficult to achieve due to uncertainty of future demands of consumers and resource prices of providers. In this context, the paper proposes an optimal cloud resource provisioning algorithm by formulating a stochastic programming model. Other researches tackle virtualization and the scheduling of virtual resources to physical ones with the objective of minimizing the network costs [20] [22] [2] and the computation costs [22] [18] [28] in clouds or cooperative clouds [21]. In [9], main challenges inherent to the resource allocation problem are highlighted and categorized. An architecture for market-oriented clouds and resource management strategies, as well as an overview of representative clouds platforms, are presented in [5]. Finally, a study that demonstrates the economic feasibility and efficiency of cloud computing for small and medium enterprises is presented in [26]. Most of papers found in the related literature consider the resource allocation problem from the cloud provider view whether in optimization of energy cost or the cost of the services and resources provision. Problems pertinent to cloud consumers, such as optimization of payment costs and execution times to run their applications, are usually neglected in related works.

In this paper, based on the actual way that popular cloud providers, such as *Amazon Elastic Compute Cloud* (EC2) and *Google Cloud Platform* offer resources and services for high performance computing users, a resource management problem that aims to reduce the payment cost and the execution time of the user application is defined. In order to solve it, an integer programming formulation and a heuristic based on Greedy Randomized Adaptive Search Procedure (GRASP) [10] are also introduced. The model and the algorithms were tested over a set of instances constructed from requirements of real applications combined with sets of resources offered by commercial clouds. The obtained results indicate that the presented methods can be an important decision support tool for cloud consumers.

2 Problem Definition and Mathematical Formulation

In literature there are many different definitions of Cloud Computing. According to [27], one reason for the existence of different perceptions is that Cloud

Computing is not a new technology, but a new operational model that brings together a set of existing technologies in a different way. In this work, we consider an Infrastructure-as-a-Service (Iaas) model, i.e., cloud consumers request computing resources as processing power (defined by Gflop[1]), disk storage, memory and architecture type, for a period of time and pays only what he/she uses. For instance, the values charged by a IaaS provider as *Amazon EC2* are mainly based on computer resources allocated for virtual machines (denoted by packages) purchased by the consumer. Cloud providers have a wide variety of package types (e.g. micro, high performance CPU, computers clusters, GPU clusters, high operations I/O, etc.), each one associated with a different cost and configuration, designed to meet different requirements. In this context, consumers have to decide which packages should purchase in order to minimize a specific objective as execution time or payment cost.

To describe this scenario as a mathematical formulation, we set the grounds for all the notation used from here on. Let P be the set of packages types offered by a cloud provider during a set of time periods. We define the set of consumer requirements as the maximum cost C_M, maximum time T_M, disk storage D_S, memory capacity M_C and a processing demand of G_f Gflop. Similarly, each package type $p \in P$ has an associated cost c_p (i.e. the cost of purchasing the package for one period of time) and computing resources as disk storage d_p, memory capacity m_p and a processing power of g_p Gflop per period of time (Gflopt). Moreover, it is common that cloud providers have a maximum limit of N_M purchased packages for each consumer in each period of time. We now define a binary variable x_{pit} for each $p \in P$, $i \in \{1, \ldots, N_M\}$ and $t \in T = \{1, \ldots, T_M\}$, such that $x_{pit} = 1$ if and only if package i of type p is purchased at time t; otherwise $x_{pit} = 0$. Also, define variable t_m as the last time period that a package was purchased by the consumer. The described scenario can be formulated as the following integer programming problem:

$$\text{(CC-IP)} \quad \min(\alpha_1 \sum_{p \in P} \sum_{i=1}^{N_M} \sum_{t \in T} c_p x_{pit} + \alpha_2 t_m) \tag{1}$$

$$\text{subject to} \quad \sum_{p \in P} \sum_{i=1}^{N_M} \sum_{t \in T} c_p x_{pit} \leq C_M \tag{2}$$

$$\sum_{p \in P} \sum_{i=1}^{N_M} d_p \; x_{pit} \geq D_S \; x_{p'i't}, \quad \forall t \in T, \forall p' \in P, \; \forall i' \in \{1, \ldots, N_M\} \tag{3}$$

$$\sum_{p \in P} \sum_{i=1}^{N_M} m_p \; x_{pit} \geq M_C \; x_{p'i't}, \quad \forall t \in T, \forall p' \in P, \; \forall i' \in \{1, \ldots, N_M\} \tag{4}$$

[1] flop - floating-point operations.

$$\sum_{p\in P}\sum_{i=1}^{N_M}\sum_{t\in T} g_p x_{pit} \geq G_f \tag{5}$$

$$\sum_{p\in P}\sum_{i=1}^{N_M} x_{pit} \leq N_M, \quad \forall t \in T \tag{6}$$

$$t_m \geq t\ x_{pit}, \quad \forall t \in T, \forall p \in P, \forall i \in \{1,\ldots,N_M\} \tag{7}$$

$$x_{pit+1} \leq x_{pit}, \quad \forall t \in T, \forall p \in P, \forall i \in \{1,\ldots,N_M\} \tag{8}$$

$$x_{pi+1t} \leq x_{pit}, \quad \forall t \in T, \forall p \in P, \forall i \in \{1,\ldots,N_M-1\} \tag{9}$$

$$x_{pit} \in \{0,1\}, \quad \forall t \in T, \forall p \in P, \forall i \in \{1,\ldots,N_M\} \tag{10}$$

$$t_m \in \mathbb{Z} \tag{11}$$

where $(\alpha_1 + \alpha_2) = 1$.

In the above model, denoted as CC-IP, the objective function 1 seeks both the minimization of costs and total purchased time. The weights α_1 and α_2 define the relevance of each objective established by the consumer. Generally, since these objectives are conflicting, both objectives cannot arrive at optimal levels simultaneously. For example, for values of α_1 close to 1 the formulation prioritizes low-budget solutions with long periods of execution time. Constraint (2) ensures that the paid costs do not exceed the maximum recommended cost. Inequalities (3) and (4) enforce that the storage and memory capacity are, respectively, sufficient large to meet the requirements in each period of time. Similarly, constraint (5) guarantee that the processing power is at least large enough to satisfy the total demand. Inequalities (6) ensures that the number of selected packages does not exceed the cloud providers limit while constraint (7) together with objective function guarantee the correct interpretation of variable t_m. Constraint (8) enforces that there will be no gaps of time in feasible solutions (i.e. if a package is selected at time $t+1$, then it will also be selected at time t). Note that all package permutations yield feasible solutions in CC-IP, thus we add constraints (9) to establish an order between the packages and eliminate symmetry. Constraints (10) and (11) define the binary and integrality requirements on the variables.

3 GRASP for Resource Selection in Cloud Environments

Exact procedures have often proved incapable of finding solutions as they are extremely time-consuming, particularly for real-world problems. Conversely, heuristics and metaheuristics provide sub-optimal solutions in a reasonable time. In this context, we have designed and implemented a Greedy Randomized Adaptive Search Procedure (GRASP) which has been successfully applied in many combinatorial optimization problems [24]. The proposed heuristic *GraspCC* is

composed of two phases: a construction phase *coCC* and a local search phase *lsCC*. The construction phase finds an initial solution that may be later improved by the local search phase.

In order to describe this method, we need some additional notation. We define a *solution* $\{(p_1, i_1, t_1), (p_2, i_2, t_2), \ldots\}$ as a set of 3−tuples (p, i, t) representing that package i of type p was purchased on period t. Let $\mathbb{S}$ be the set of all possible solutions. A solution $s \in \mathbb{S}$ is said to be *feasible* if s respects the consumer requirements. We also denote $t_m(s) = \max_{(p,i,t)\in s} t$ as the last period that a package was selected. Moreover, we define below a *cost function* $F : \mathbb{S} \to \mathbb{R}$, which will measure the quality of the solution. Similar to the objective function (1), the function F attempts to minimize costs and time (12) while penalizes the infeasibility regarding maxima time and cost (13). The parameters λ_1 and λ_2 are penalty coefficients associated with the violation of time and cost requirements, respectively.

$$F(s) = (\alpha_1 \sum_{(p,i,t)\in s} c_p + \alpha_2 t_m(s)) \tag{12}$$

$$+ \lambda_1(\max\{0, t_m(s) - T_M\}) + \lambda_2(\max\{0, \sum_{(p,i,t)\in s} c_p - C_M\}) \tag{13}$$

Algorithm 1: *GraspCC*

```
1  Input: P, C_M, T_M, D_S, M_C, G_f, α1, α2, λ1, λ2
2  Output: solution s*;
3  s* = ∅;  F(s*) = ∞;  i := 0;
4  while i ⩽ iter
5     s = coCC(P, C_M, T_M, D_S, M_C, G_f, α1, α2, λ1, λ2);
6     s = lsCC(s, P, C_M, T_M, D_S, M_C, G_f, α1, α2, λ1, λ2);
7     if (F(s) < F(s*)) and (s is feasible)
8        s* = s;  i := 0;
9     end if
10 end for
11 return s*;
```

We can now state the algorithm *GraspCC* which is shown in Algorithm 1, where the parameter *iter* denotes the maximum number of iterations without improvement in the best solution found. The first task in every iteration of the *GraspCC* is to construct a solution starting from an empty set, in a greedy randomized fashion. This task is performed by algorithm *coCC* which is described in Algorithm 2. In this method, the ordered set L_P is defined, in line 3, as the set of packages $p \in P$ such that its elements appears in descending order of cost and processing power $(\alpha_1 c_p + \alpha_2 g_p)$. At each iteration in lines 4-7 we choose a package $\overline{p}$ among the first β packages to be added to the first period of time. Note that the parameter β defines the degree of randomness the construction phase will have. This process is repeated until the solution satisfies the disk and memory requirements (for $t = 1$), however the maximum time and cost constraints are

relaxed in order to achieve diversity in the initial solution. Moreover, in lines (9)-(12) the chosen packages are replicated in future periods until the processing demand is satisfied.

Algorithm 2: *coCC*

1 **Input:** $P, C_M, T_M, D_S, M_C, G_f, \alpha_1, \alpha_2, \lambda_1, \lambda_2$
2 **Output:** solution s;
3 $s = \emptyset$; $L_P = Order(P)$;
4 **while** $(\sum_{p|(p,i,1)\in s} d_p < D_S)$ or $(\sum_{p|(p,i,1)\in s} m_p < M_C)$
5 Choose package $\overline{p}$ (index $\overline{i}$) randomly among the first β elements of L_P
6 $s = s \cup \{(\overline{p}, \overline{i}, 1)\}$;
7 **end while**
8 $\overline{t} = 2$;
9 **while** $(\sum_{p|(p,i,t)\in s} g_p < G_f)$
10 $s = \bigcup_{(p,i,1)\in s}(p, i, \overline{t}) \cup s$;
11 $\overline{t} = \overline{t} + 1$;
12 **end while**
13 **return** s

There is no guarantee that the construction method returns a feasible or a locally optimal solution with respect to some neighborhood. Therefore, the solution s may be improved by a local search procedure denominated *lsCC* (Algorithm 3). The neighborhood $N_r(s)$ is defined as the family of all solutions obtained by exchanging r tuples in s with another r tuples not in s. The *lsCC* method starts with the solution provided by the construction phase. It iteratively replaces the current solution by that with minimum cost function F within its neighborhood (lines (3)-(9), where "s improving" means that there is some solution in the neighborhood of the current solution with cost function better than the current s). The local search halts when no better solution is found in the neighborhood of the current solution. In this work we employed a neighborhood with $r \leq 2$, since for values of $r > 2$ it is computationally intensive to perform an exhaustive search.

Algorithm 3: *lsCC*

1 **Input:** $s, P, C_M, T_M, D_S, M_C, G_f, \alpha_1, \alpha_2, \lambda_1, \lambda_2$
2 **Output:** solution s;
3 **while** (s improving)
4 **for all** $\overline{s} \in (N_1(s) \cup N_2(s))$
5 **if** $F(\overline{s}) < F(s)$
6 $s = \overline{s}$
7 **end if**
8 **end for**
9 **end while**

4 Experimental Results and Conclusions

The algorithms described in the previous section were implemented in ANSI C and the CPLEX 12.4 [16] was used to solve the mathematical formulation. All simulated experiments were performed on a computer with processor Intel Core i7 3.4Hz and 12Gb of RAM under Linux (Ubuntu 12.04) operating system. The algorithms were tested over a set of instances constructed from the requirements of RAM memory, disk storage, gflop number, execution time and payment cost of real applications combined with the sets of virtual machine packages available in commercial clouds.

These instances use requirements of five real applications and packages of two commercial clouds. The instances are divided in two groups, one for each cloud. The considered applications are a branch-and-bound algorithm for solving the Quadratic Assignment Problem (QAP) [12], three algorithms that tackle the manipulation of biologic sequence problem (RAXML [25], ModelGenerator [17] and Segemehl [15]) and a typical analysis user job for the CMS experiment [1]. The two commercial clouds are *Amazon EC2* [8] and *Google Cloud Platform* [23]. In *Amazon EC2*, package groups of high performance computing (HPC) clusters were employed, while in *Google Cloud Platform*, it was utilized packages from the *Google Compute Engine*. The instances are described in Table 1, where the columns represent the instance name, the requirements of RAM memory, disk storage, gflop number, the informed maximum time execution and maximum payment cost, respectively.

In order to illustrate the use of the different goals in the proposed cost function, weighted by α_1 and α_2 values, we analyse an instance executed with distinct sets of alpha, representing two extreme cases: (i) the cloud consumer is only concerned with payment cost ($\alpha_1 = 1$ and $\alpha_2 = 0$) and (ii) he/she only prioritizes the execution time ($\alpha_1 = 0$ and $\alpha_2 = 1$). For instance, if for the application nug24-cbb using packages of *Amazon EC2* (nug24-cbb_*am*), the cloud consumer only considers the payment cost, the mathematical formulation gives the following solution: 3 time units for execution time and $46.20 for payment

Table 1. Instance Description

Instance	RAM (GB)	Disk Storage (GB)	GFLOP Number	Time (hours)	Package Number	Cost ($)
nug22-sbb	77	51	5067533	12	20	343
nug24-sbb	154	103	14741914	48	20	998
nug25-sbb	214	142	28792800	60	20	1950
nug28-sbb	528	352	67720666	72	20	4586
nug30-sbb	918	612	120929760	84	20	8190
nug22-cbb	77	51	20270131	12	20	343
nug24-cbb	154	154	88451482	72	20	1498
nug25-cbb	214	214	230342400	120	20	3900
nug28-cbb	528	528	541765325	144	20	9173
nug30-cbb	918	918	967438080	168	20	16380
mod-gen	4	2	3317760	24	20	100
raxml	3	2	3317760	24	20	100
segemehl	64	600	28748390	4	20	192
cms-1000	1500	20	216000000	24	30	1728
cms-1250	1875	25	270000000	24	40	2304
cms-1500	2250	30	324000000	24	45	2592

Table 2. Results of *GraspCC* Metaheuristic and CC-IP Mathematical Formulation using CPLEX with $\alpha_1 = 0.5$ and $\alpha_1 = 0.5$

Instances	*GraspCC*					CC-IP			
	Function cost	Solution Value		Total time	Gap (%)	Function cost	Solution Value		Total time
		Time	Payment				Time	Payment	
nug22-sbb_*am*	0.0456	1	4.50	0.01	0	0.0456	1	4.50	0.59
nug24-sbb_*am*	0.0130	1	12.10	0.02	0	0.0130	1	12.10	6.79
nug25-sbb_*am*	0.0113	1	17.10	0.05	0	0.0113	1	17.10	10.98
nug28-sbb_*am*	0.0128	1	40.80	0.07	0	0.0128	1	40.80	27.23
nug30-sbb_*am*	0.0344	3	133.50	0.08	0	0.0344	3	133.50	32.42
nug22-cbb_*am*	0.0519	1	11.80	0.03	0	0.0519	1	11.80	0.77
nug24-cbb_*am*	0.0309	2	46.20	0.17	0	0.0309	2	46.20	5.81
nug25-cbb_*am*	0.0458	3	119.70	0.91	0	0.0458	3	119.70	12.01
nug28-cbb_*am*	0.0123	1	36.90	0.07	0	0.0123	1	36.90	21.20
nug30-cbb_*am*	0.2652	23	1034.70	3.63	0	0.2652	23	1034.70	118.42
mod-gen_*am*	0.0217	1	2.10	0.00	0	0.0217	1	2.10	0.50
raxml_*am*	0.0217	1	2.10	0.00	0	0.0217	1	2.10	0.48
segemehl_*am*	0.1688	1	16.80	0.07	0	0.1688	1	16.80	0.26
cms-1000_*am*	0.1630	4	275.20	0.18	0	0.1630	4	275.20	31.82
cms-1250_*am*	0.1572	4	340.20	0.53	0	0.1572	4	340.20	39.50
cms-1500_*am*	0.1625	4	410.40	0.53	0	0.1625	4	410.40	32.83
nug22-sbb_*go*	0.0497	1	4.90	5.36	0.0066	0.0497	1	4.88	77.72
nug24-sbb_*go*	0.0270	1	14.95	20.74	0	0.0270	1	14.95	9194.37
nug25-sbb_*go*	0.0347	3	29.54	38.90	0	0.0347	3	29.54	28081.33
nug28-sbb_*go*	0.0655	6	87.44	133.66	0	0.0655	6	87.44	35278.98
nug30-sbb_*go*	0.1128	11	202.27	42.98	0	0.1128	11	202.27	73656.27
nug22-cbb_*go*	0.1137	2	18.56	17.97	0	0.1137	2	18.56	246.79
nug24-cbb_*go*	0.1173	8	82.91	177.53	0	0.1173	8	82.91	19695.32
nug25-cbb_*go*	0.2417	20	229.00	1186.05	0	0.2417	20	229.00	64758.31
nug28-cbb_*go*	0.0541	5	70.73	60.00	0.0859	* 0.0552	5	70.64	86400.00
nug30-cbb_*go*	0.8502	81	1573.22	2987.79	0	0.8502	81	1573.22	5085.60
mod-gen_*go*	0.0234	1	3.16	2.97	0	0.0234	1	3.16	32.71
raxml_*go*	0.0234	1	3.16	2.91	0	0.0234	1	3.16	29.34
segemehl_*go*	0.5038	3	26.21	52.40	0	0.5038	3	26.21	23.52
cms-1000_*go*	0.4809	13	384.70	27.56	0	0.4809	13	384.70	4458.83
cms-1250_*go*	0.4341	12	449.52	153.13	0	0.4341	12	449.52	34210.73
cms-1500_*go*	0.4796	13	573.46	53.27	0	* 0.4797	13	573.85	86400.00

cost. However, if the cloud consumer only prioritizes the execution time, the mathematical formulation gives the solution: 2 time units for execution time and $84 for payment cost. Thus, note that the proposed objective function that integrates both objectives can guide the consumer's decision. The specification of proper weights (α_1 and $\alpha 2$) to the objectives allows that the cloud consumers take the correct decision in accordance with their needs.

Experiments, reported in Table 2, are a comparison of the *GraspCC* heuristic with the CC-IP mathematical formulation, in terms of solution quality and execution time. The used values were: $\alpha_1 = 0.5$ and $\alpha_2 = 0.5$, $\lambda_1 = 1000$ and $\lambda_2 = 1000$, β equal to the maximum number of packages offered by each cloud provider and $iter = 50$. In this work, both objectives of the cost function were normalized due to their distinct range values. The normalization procedure updates the objective values so that they share the same minimum and maximum values, 0 and 1, respectively. Thus, the payment cost was divided by the most expensive package cost times the maximum time informed by the cloud consumer. Similarly, the execution time component of the objective function was divided by the maximum time informed by the cloud consumer. Each instance is identified by the label n_c, where n is the application name and c is the cloud

providers (*am* and *go* identify *Amazon EC2* and *Google Compute Engine*, respectively). The next three columns present the best function cost solution found by *GraspCC* and the costs (not normalized) that compose it: execution time and payment cost, respectively. The fifth column shows the execution time in seconds for *GraspCC* to solve the problem. The next column is the corresponding percentage difference from the optimal solution (named Gap in the table). Analogously, the last four columns present the best feasible solution found by CC-IP, the costs that compose it, the execution time and payment costs, and the time in seconds spent to solve the problem.

For some instances, the CC-IP mathematical formulation took a long time to prove the optimality of the solution, therewith we consider in our experiments an execution time limit of 24 hours. The exact method was not able to find the optimal solution in instances marked with (*) in Table 2 within this time limit. Note that *GraspCC* presented an outstanding improvement of the execution time, in average 99.01% less than the execution time of CC-IP, giving optimal or almost optimal solutions in all tests.

This work was supported of CAPES and FAPERJ (E-26/110.552/2010).

References

1. Adolphi, R., Spanier, S.: The CMS experiment at the CERN LHC, CMS collaboration. Journal of Instrumentation 3(08), S08004 (2008)
2. Alicherry, M., Lakshman, T.: Network aware resource allocation in distributed clouds. In: Proceedings IEEE INFOCOM, pp. 963–971. IEEE (2012)
3. Baliga, J., Ayre, R.W., Hinton, K., Tucker, R.S.: Green cloud computing: Balancing energy in processing, storage, and transport. Proceedings IEEE 99(1), 149–167 (2011)
4. Beloglazov, A., Abawajy, J., Buyya, R.: Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Future Generation Computer Systems 28(5), 755–768 (2012)
5. Buyya, R., Yeo, C.S., Venugopal, S., Broberg, J., Brandic, I.: Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation Computer Systems 25(6), 599–616 (2009)
6. Chaisiri, S., Lee, B., Niyato, D.: Optimization of resource provisioning cost in cloud computing. IEEE Transactions on Services Computing 5(2), 164–177 (2012)
7. de Oliveira Jr., F.A., Ledoux, T.: Self-management of applications qos for energy optimization in datacenters. In: Proceedings of the 2nd International Workshop Green Computing Middleware, GCM 2011, pages 3:1–3:6. ACM (2011)
8. Amazon Elastic Compute Cloud (Amazon EC2) (March 16, 2013), `http://aws.amazon.com/pt/ec2/`
9. Endo, P.T., de A. Palhares, A.V., Pereira, N.N., Goncalves, G.E., Sadok, D., Kelner, J., Melander, B., Mangs, J.: Resource allocation for distributed cloud: concepts and research challenges. IEEE Network 25(4), 42–46 (2011)
10. Feo, T.A., Resende, M.G.C.: Resende. Greedy randomized adaptive search procedures. Journal of Global Optimization 6, 109–133 (1995)
11. Foster, I., Zhao, Y., Raicu, I., Lu, S.: Cloud computing and grid computing 360-degree compared. In: Grid Computing Environments Workshop, pp. 1–10. IEEE (2008)

12. Goncalves, A., Drummond, L., Pessoa, A., Hahn, P.: Improving lower bounds for the quadratic assignment problem by applying a distributed dual ascent algorithm. Cornell University Library, Technical Report (2013)
13. Goudarzi, H., Ghasemazar, M., Pedram, M.: Sla-based optimization of power and migration cost in cloud computing. In: 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, pp. 172–179. IEEE (2012)
14. Goudarzi, H., Pedram, M.: Energy-efficient virtual machine replication and placement in a cloud computing system. In: IEEE 5th International Conference on Cloud Computing, pp. 750–757. IEEE (2012)
15. Hoffmann, S., Otto, C., Kurtz, S., Sharma, C.M., Khaitovich, P., Vogel, J., Stadler, P.F., Hackermüller, J.: Fast mapping of short sequences with mismatches, insertions and deletions using index structures. PLoS Computational Biology 5 (2009)
16. S.A. ILOG. Cplex 11 user's manual (2008)
17. Keane, T., Creevey, C., Pentony, M., Naughton, T., Mclnerney, J.: Assessment of methods for amino acid matrix selection and their use on empirical data shows that ad hoc assumptions for choice of matrix are not justified. BMC Evolutionary Biology 6(1), 29 (2006)
18. Li, Q., Guo, Y.: Optimization of resource scheduling in cloud computing. In: 12th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, pp. 315–320. IEEE (2010)
19. Luo, L., Wu, W., Di, D., Zhang, F., Yan, Y., Mao, Y.: A resource scheduling algorithm of cloud computing based on energy efficient optimization methods. In: International Green Computing Conference, pp. 1–6. IEEE (2012)
20. Meng, X., Pappas, V., Zhang, L.: Improving the scalability of data center networks with traffic-aware virtual machine placement. In: Proceedings of IEEE INFOCOM, pp. 1–9. IEEE (2010)
21. Niyato, D., Zhu, K., Wang, P.: Cooperative virtual machine management for multi-organization cloud computing environment. In: Proceedings of the 5th International ICST Conference on Performance Evaluation Methodologies and Tools, pp. 528–537, ICST (2011)
22. Pandey, S., Wu, L., Guru, S.M., Buyya, R.: A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In: 24th IEEE International Conference on Advanced Information Networking and Applications, pp. 400–407. IEEE (2010)
23. Google Cloud Platform (March 16, 2013), `https://cloud.google.com/products/compute-engine`
24. Resende, M.G.C., Ribeiro, C.C.: GRASP with path-relinking: Recent advances and applications. In: Ibaraki, T., Nonobe, K., Yagiura, M. (eds.) Metaheuristics: Progress as Real Problem Solvers, pp. 29–63. Springer (2005)
25. Stamatakis, A.: Raxml-vi-hpc: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models. Bioinformatics 22(21), 2688–2690 (2006)
26. Sultan, N.A.: Reaching for the cloud: How SMEs can manage. International Journal of Information Management 31(3), 272–278 (2011)
27. Zhang, Q., Cheng, L., Boutaba, R.: Cloud computing: state-of-the-art and research challenges. Journal of Internet Services and Applications 1(1), 7–18 (2010)
28. Zhao, J., Zeng, W., Liu, M., Li, G.: Multi-objective optimization model of virtual resources scheduling under cloud computing and it's solution. In: International Conference on Cloud and Service Computing, pp. 185–190 (2011)