Heuristic Approaches for Robust Cloud Monitor Placement

Melanie Siebenhaar, Dieter Schuller, Olga Wenge, and Ralf Steinmetz

Technische Universität Darmstadt, Multimedia Communications Lab (KOM),
Rundeturmstr. 10, 64283 Darmstadt, Germany
Email:{firstname.lastname}@KOM.tu-darmstadt.de

When utilizing cloud-based services, consumers obtain high configurable resources with minimal management effort and eliminate large up-front IT investments. However, this shift in responsibility to the cloud provider is accompanied by a loss of control for the cloud consumer. By offering SLAs and corresponding monitoring solutions, cloud providers already try to address this issue, but these solutions are not considered as sufficient from a consumer's perspective. Therefore, we developed an approach that allows to verify compliance with SLAs from a consumer's perspective in our former work. Since the monitoring infrastructure itself may fail, this approach was enhanced in one of our subsequent works in order to account for reliability. We introduced the Robust Cloud Monitor Placement Problem and a formal optimization model. In this paper, we propose corresponding solution approaches and evaluate their practical applicability, since the problem is NP-complete.

1 Introduction

Utilizing services from the cloud, consumers gain a very high level of flexibility. Configurable computing resources are provided on-demand in a similar manner like electricity or water [4] at a minimal amount of management effort. However, this shift in responsibility to the cloud provider bears several risks for the cloud consumer. Amongst these risks is a loss of control concerning aspects like performance, availability, and security. Quality guarantees in the form of so-called Service Level Agreements (SLAs) offered by cloud providers aim at lowering this risk in favor for cloud consumers. Basically, an SLA represents a contract between a cloud provider and a cloud consumer and defines certain quality levels (e.g., lower bounds for performance parameters such as availability) to be maintained by the cloud provider. In addition, such a contract specifies some penalties for the cloud provider in case of SLA violations. Nevertheless, the cloud consumers' perception still is that providers do not sufficiently measure performance against SLAs [5]. Furthermore, cloud providers often assign the task of violation reporting to their customers [11]. But even despite the fact that some cloud providers also offer consumers corresponding monitoring solutions, these solutions cannot be perceived as an independent evidence base for the detection and documentation of SLA violations from a consumer's perspective.

Our previous works ([15], [14]) addressed this issue. We proposed a hybrid monitoring approach in order to provide reliable means to verify adherence with SLAs from a consumer's perspective. This approach focuses on the consumer-side verification of availability of cloud applications and comprises the placement of monitoring units on provider and consumer side. Furthermore, we argue that not only the monitoring quality of such an approach has to be taken into account, but also the reliability of the monitoring infrastructure itself. This view was emphasized by our evaluation which revealed the sensitivity of our approach to network impairments. Therefore, we extended our approach and explored the placement of monitoring units with respect to current network reliability. As result, we introduced the *Robust Cloud Monitor Placement Problem* (RCMPP) as a new research problem along with a corresponding formal optimization model. Since the RCMPP is a nonlinear, multi-objective optimization problem, we further proposed transformations in order to turn the problem into a linear, single-objective optimization problem and thus, laying the foundation to exactly solve the problem using off-the-shelf optimization algorithms.

In this paper, we investigate the applicability of such exact optimization algorithms and also propose new heuristic solution approaches. Due to the fact that the RCMPP is NP-complete, exact optimization algorithms are very likely to be inapplicable for real-world scenarios exhibiting large-scale data center infrastructures. Hence, we compare the computation time and solution quality of an exact, ILP-based approach against heuristic solution approaches in order to derive recommendations for their applicability in practice. Furthermore, we extend our approach to a broker-based scenario and propose the concept of *Monitoring Level Agreements* (MLAs). In doing so, our extended approach permits an advanced performance control for cloud consumers.

The remainder of this paper is structured as follows: Section 2 gives an overview of related work. Section 3 briefly describes our extended approach and gives an overview of our former work. In Section 4, we propose exact and heuristic solution approaches for the RCMPP and present a corresponding evaluation in Section 5. In Section 6, the paper closes with a summary and future work.

2 Related Work

Only a few approaches exist which consider a reliable placement of monitoring units. A distribution of sensor pods on each cloud node in order to enable tenants to monitor the connectivity between their allocated nodes is proposed by [12]. The authors in [2] follow a different approach and focus on relocatable VMs in the presence of up to k host failures. For this purpose, the authors propose different optimization algorithms in order to determine k independent paths from the monitoring nodes to all other nodes in the network. Concerning networks in general, [10] aim at minimizing the number of monitoring locations while taking a maximum of k node/edge failures into account. A related problem also exists in the field of Operations Research. The fault-tolerant facility location problem, first studied by [8], tries to connect each city with at least a certain

number of distinct facilities in order to account for failures. However, none of the related approaches addresses robust monitor placement by jointly considering current network reliability, monitor redundancy, and resource as well as location constraints.

3 Performance Control for Cloud Consumers

In this section, we give an overview of our extended approach and align our former work as well as our new contributions in the context of a broker-based scenario. Our former work is briefly described in the next sections followed by a detailed description of our new contributions.

This paper focuses on a broker-based scenario that is based on a cloud market model. In such a cloud market, cloud consumers submit their functional and non-functional requirements for their desired cloud services to brokers, which constitute mediators between cloud consumers and providers [4]. The brokers are capable to determine the most suitable cloud providers by querying a service registry residing in the market. Furthermore, such a broker enables consumers to negotiate SLAs to be provided by the cloud provider. For this purpose, a broker acts on behalf of a cloud consumer and conducts SLA negotiations with cloud providers. Now, in order to be able to verify compliance with SLAs from a consumer's perspective later on, the broker can also act on behalf of consumers and apply our proposed monitoring approach during runtime. The advantage of a broker-based perspective for our approach is the exploitation of global knowledge. In case that SLA violations are detected, a broker is aware of the adherence to SLAs of other cloud providers and thus, is able to recommend alternative cloud providers. In addition, the monitoring information gained at runtime can be used to initiate SLA re-negotiations or to adapt the properties of the monitoring approach in order to improve monitoring quality or monitoring infrastructure reliability.

In the following, we focus on an enterprise cloud consumer utilizing a set of applications running in different data centers of a cloud provider. In order to verify the performance guarantees in the form of SLAs obtained from the cloud provider, the enterprise cloud consumer entrusts the broker which conducted the SLA negotiations before with the monitoring of the running cloud applications. As part of such a monitoring service order, we propose the definition of *Monitoring Level Agreements* (MLAs) specifying the properties of the monitoring tasks for each application (cf. Section 3.1 for details). The broker then applies our hybrid monitoring approach and places monitoring units for each cloud application on provider- as well as on broker-side. Besides the provider-side monitoring, the broker-side monitoring permits an assessment of the status of a cloud application from a consumer's perspective. In order to obtain a robust monitor placement, the broker can select one of our proposed monitor placement algorithms according to our investigation in Section 4.

3.1 Hybrid Monitoring Approach and Monitoring Level Agreements

Our hybrid monitoring approach introduced in our former work in [15] focuses on verifying the availability of cloud applications from a consumer's perspective, since availability is one of the very few performance parameters contained in current cloud SLAs. Nevertheless, other performance parameters can be easily incorporated in our monitoring approach as well. In order to allow for an independent assessment of the status of a cloud application and visibility of the end-to-end performance of cloud applications, we proposed a hybrid monitoring approach with placements of monitoring units on provider and consumer side. The latter are now replaced by broker-side placements. Furthermore, such a hybrid approach permits to differentiate between downtimes caused by issues on broker and provider side and thus, enables a broker to filter downtimes that relate to cloud provider SLAs. In our hybrid monitoring approach, each monitoring unit observes predefined services of a cloud application as well as processes of the underlying VM. For each cloud application, the set of services to be invoked by a monitoring unit can be defined in advance. Same applies for the system processes to be observed on VM level. For this purpose, MLAs can specify the consumer's requirements concerning all the cloud applications to be monitored. Besides the services and processes, an amount of redundant monitoring units to be placed for each application can be defined. Higher numbers of redundant monitoring units are reasonable for business critical applications, since the probability that all redundant monitors fail decreases. We also follow this assumption in our monitor placement approach described in the next section.

3.2 Robust Cloud Monitor Placement

As already stated before, our approach must not only consider monitoring quality, but also has to account for downtimes of the monitoring infrastructure itself. Therefore, the monitoring units have to be placed by a broker in the data centers on provider- and broker-side in such a way that maximizes the robustness of the monitoring infrastructure. We introduced this problem, denoted as *Robust Cloud Monitor Placement Problem* (RCMPP), in our former work in [14]. The corresponding formal model is briefly described in the following. Table 1 shows the basic entities (upper part) and parameters (lower part) used in the formal model. Each instance of the RCMPP consists of a set $S = \{1, ..., n\}$ of data center sites comprising the set $S'' = \{1, ..., d\}$ of data center sites on broker side and the set $S' = \{d+1, ..., n\}$ of data center sites on cloud provider side. On each data center site $s \in S$ on provider and broker side, a set $V_s = \{1, ..., i\}$ of VMs is running which constitute candidates for monitor placement. A set of cloud applications $C_{s'v'} = \{1, ..., j\}$ to be monitored is running on each VM $v' \in V_{s'}$ located on a data center site $s' \in S'$ on provider side. A set of links $L = \{l(sv \rightleftharpoons s'v')\}$ interconnects the VMs $v \in V_s$ constituting placement candidates with the VMs $v' \in V_{s'}$ of the cloud applications $C_{s'v'}$. Each cloud application $c \in C_{s'v'}$ has certain requirements concerning the corresponding monitoring units. These requirements comprise a specific resource demand of $rd_{s'v'cr} \in \mathbb{R}^+$ for a specific

Table 1. Used symbols in the formal model

Symbol	Description
$S = \{1, ..., n\}$	set of n data center sites
$S'' = \{1, ..., d\}$	consumer sites, $S'' \subset S$
$S' = \{d+1, ..., n\}$	provider sites, $S' \subset S$
$V_s = \{1, ..., i\}$	VM candidates for monitor placement on site $s \in S$
$C_{s'v'} = \{1, ..., j\}$	cloud applications to monitor on VM $v' \in V_{s'}, s' \in S'$
$L = \{l(sv \rightleftharpoons s'v')\}$	links interconnecting VM monitor candidates V_s and VMs of applications $C_{s'v'}$
$R = \{1, ..., k\}$	set of k considered VM resource types
$rd_{s'v'cr} \in \mathbb{R}^+$	resource demand for monitoring application $c \in C_{s'v'}$ for resource $r \in R$
$rs_{svr} \in \mathbb{R}^+$	resource supply of VM $v \in V_s$ for resource $r \in R$
$rf_{s'v'c} \in \mathbb{N}_{>1}$	redundancy factor for monitoring application $c \in C_{s'v'}$
$p_{l(sv \rightleftharpoons s'v')} \in \mathbb{R}^+$	observed reliability for each link $l \in L$
$p_{sv} \in \mathbb{R}^+$	observed reliability for each VM $v \in V_s$

resource type $r \in R = \{1, ..., k\}$ such as CPU power or memory, and a redundancy factor $rf_{s'v'c} \in \mathbb{N}_{>1}$, indicating that the cloud application c has to be monitored by $rf_{s'v'c}$ different monitoring units. In order to account for the reliability of the monitoring infrastructure, it has to be noted that the broker is not aware of the underlying network topologies of the cloud provider and the Internet service provider. However, we assume that the broker is able to utilize traditional network measurement tools in order to estimate the end-to-end performance between any pair of VMs that are represented by a given link $l \in L$ in order to determine the observed reliability $p_{l(sv \rightleftharpoons s'v')} \in \mathbb{R}^+$ for a given link $l \in L$. Furthermore, we assume that the broker can also utilize such measurement tools in order to estimate the reliability $p_{sv} \in \mathbb{R}^+$ of a given VM $v \in V_s$ on a site $s \in S$. Finally, our model must also consider the respective resource supply of $rs_{svr} \in \mathbb{R}^+$ each VM $v \in V_s$ on a site $s \in S$ is able to provide. The objective of the RCMPP now is to assign $rf_{s'v'c}$ monitoring units for each cloud application to be monitored on broker and provider side, while maximizing the reliability of the whole monitoring infrastructure. Hereby, we express the reliability by the probability that at least one of the monitoring units for each cloud application is working properly. In doing so, the resource constraints of the VMs must not be exceeded and all monitoring units must be placed. Furthermore, we incorporate a set of placement restrictions for the monitoring units. First of all, no monitoring unit is allowed to be placed on the VM of the cloud application to be monitored and second, one monitoring unit must be placed on broker and provider side, respectively. Both restrictions directly follow from our hybrid monitoring approach. Third, for reasons of fault-tolerance, each monitoring unit to be placed for a single application must be placed on a different site.

3.3 Formal Model

The corresponding optimization model for the RCMPP is depicted in Model 1 and serves as a starting point for the development of solution algorithms. The RCMPP constitutes a multi-objective optimization problem, since we want to maximize the reliability of the monitoring units for each cloud application, simultaneously (cf. Equation 1). Each of these potentially conflicting objective functions expresses the probability $p^{mon}_{s'v'c}(x)$, that at least one monitoring unit

Model 1 Robust Cloud Monitor Placement Problem

$$Maximize \quad \{p^{mon}_{s'v'c}(x) | s' \in S', v' \in V_{s'}, c \in C_{s'v'}\} \tag{1}$$

$$p^{mon}_{s'v'c}(x) = 1 - \prod_{s \in S, v \in V_s} (q^{path}_{svs'v'})^{x_{svs'v'c}} \tag{2}$$

$$q^{path}_{svs'v'} = [(1 - p_{sv}) + (1 - p_{l(sv \rightleftharpoons s'v')}) - (1 - p_{sv})\ (1 - p_{l(sv \rightleftharpoons s'v')})] \tag{3}$$

subject to

$$\sum_{s \in S, v \in V_s} x_{svs'v'c} = rf_{s'v'c} \tag{4}$$
$$\forall s' \in S', v' \in V_{s'}, c \in C_{s'v'}, rf_{s'v'c} \geq 2$$

$$\sum_{s' \in S', v' \in V_{s'}, c \in C_{s'v'}} rd_{s'v'cr}\ x_{svs'v'c} \leq rs_{svr} \tag{5}$$
$$\forall s \in S, v \in V_s, r \in R$$

$$\sum_{v \in V_s} x_{svs'v'c} \leq 1 \tag{6}$$
$$\forall s \in S, s' \in S', v' \in V_{s'}, c \in C_{s'v'}$$

$$\sum_{s \in S, v \in V_s} x_{svs'v'c} \geq 1 \tag{7}$$
$$\forall s' \in S', v' \in V_{s'}, c \in C_{s'v'}, s = \{d+1, ..., n\}$$

$$\sum_{s \in S, v \in V_s} x_{svs'v'c} \geq 1 \tag{8}$$
$$\forall s' \in S', v' \in V_{s'}, c \in C_{s'v'}, s = \{1, ..., d\}$$

$$x_{svs'v'c} = 0 \tag{9}$$
$$\forall c \in C_{s'v'},\ s = s' \ and \ v = v'$$

$$x_{svs'v'c} \in \{0, 1\} \tag{10}$$
$$\forall s \in S, v \in V_s, s' \in S', v' \in V_{s'}, c \in C_{s'v'}$$

for the respective application does not fail. Equation 2 represents this probability by 1 minus the probability that all monitors for a specific cloud application $c \in C_{s'v'}$ fail. Equation 3 determines the probability to fail ($q^{path}_{svs'v'}$) for a given monitoring unit of a specific cloud application $c \in C_{s'v'}$. Hereby, the reliability of the VM $v \in V_s$ where the monitoring unit is placed as well as the reliability of the link between this VM and the VM where the cloud application is running on are considered. Equation 10 defines a set $x_{svs'v'c}$ of binary decision variables indicating whether a monitoring unit for a cloud application $c \in C_{s'v'}$ running on VM $v' \in V_{s'}$ on site $s' \in S'$ is placed on VM $v \in V_s$ running on site $s \in S$. The vector x in Equation 1 represents all decision variables $x_{svs'v'c}$. In order to ensure that $rf_{s'v'c}$ redundant monitoring units monitor the corresponding cloud application, Equation 4 has been added to the model. Equations 7 and 8 account for the hybrid monitoring approach and specify that at least one monitoring unit has to be placed on broker and provider side, respectively. Equation 5 prevents the exceeding of the resource supplies of each VM $v \in V_s$ and Equation 6 ensures that each monitoring unit for a given application $c \in C_{s'v'}$ is placed on a different data center site. Last, but not least, a monitoring unit is not allowed to be placed on the VM where the corresponding application is running (cf. Equation 9).

The RCMPP is a multi-objective optimization problem, hence, no unique solution can be obtained. Furthermore, as can be seen from Equation 2 in Model 1, it also constitutes a nonlinear problem. Although approaches exist for solving multi-objective optimization problems, these approaches only yield pareto-optimal solutions and typically require some kind of preference structure concerning the set of solutions as input. Since defining a preference structure, e.g., in the form of a lexicographic preference order with regard to all applications to be monitored, is very exhausting on a large data center scale, we aim for a solution approach focusing on a single-objective.

4 Exact and Heuristic Solution Approaches

This section describes an exact ILP-based solution approach as well as two heuristics in order to solve the RCMPP. The heuristic algorithms are partly inspired from existing solution approaches (e.g., [6], [9]) for the related generalized assignment problem and its bottleneck version. A direct application of existing heuristics is not feasible, since no full mapping exists from the RCMPP to an existing optimization problem. Furthermore, since the RCMPP is NP-complete, ILP-based algorithms will also not be able to find solutions for large-scale problems. Therefore, the development of new heuristics is required. Our heuristics consist of an opening procedure (Greedy), which aims to find a first feasible solution, and an improvement procedure (TSearch) aiming to improve an initial solution. Hence, we obtain two different heuristics in total: Greedy and Greedy+TSearch (denoted as GTSearch).

Model 2 Robust Cloud Monitor Placement Problem after Transformation

$$Minimize \quad z \tag{11}$$

subject to

$$q^{log}_{s'v'c}(x) \leq z \tag{12}$$
$$\forall s' \in S', v' \in V_{s'}, c \in C_{s'v'}, z \in \mathbb{R}$$
$$q^{log}_{s'v'c}(x) = \sum_{s \in S, v \in V_s} x_{svs'v'c} \; log(q^{path}_{svs'v'}) \tag{13}$$
$$q^{path}_{svs'v'} = [(1 - p_{sv}) + (1 - p_{l(sv \rightleftharpoons s'v')}) \tag{14}$$
$$-(1 - p_{sv}) \; (1 - p_{l(sv \rightleftharpoons s'v')})]$$
$$q^{path}_{svs'v'} > 0 \; \forall s \in S, v \in V_s, s' \in S', v' \in V_{s'} \tag{15}$$

4.1 Integer Linear Programming (ILP)-based Approach

In order to turn the RCMPP into a linear, single-objective optimization problem, we proposed a set of transformations in our former work in [14]. The transformations are briefly summarized in the following.

First of all, the maximization problem is turned into a minimization problem. This can be achieved by considering the complementary objectives of the former Model 1, i.e., the probability $q^{mon}_{s'v'c}(x)$ that all monitors fail for each cloud application. The first step enables a subsequent linearization of the problem by taking the logarithm of both sides (this approach is also followed by [1]). Our last step is based on a worst-case analysis, where we aim to minimize the worst possible outcome. For this purpose, we apply a so-called *minimax strategy* [7], which turns the initial set of objective functions into a single objective function that aims to minimize the maximum probability of all $q^{mon}_{s'v'c}(x)$. In doing so, a new decision variable $z \in \mathbb{R}$ expressing this maximum value is introduced. In addition, $|C_{s'v'}|$ new constraints $\forall s' \in S', v' \in V_{s'}$ are added to the constraints of our former Model 1. The resulting linear, single-objective optimization problem is depicted in Model 2. Please note, that the initial constraints have been neglected due to lack of space. Furthermore, we assume $q^{path}_{svs'v'} > 0$, since no system is without failure. The resulting problem represents a mixed-integer linear programming problem that can be solved exactly using off-the-shelf algorithms such as branch-and-bound [6].

4.2 Greedy Algorithm

Algorithm 3 describes our opening procedure. This algorithm is inspired by the *steepest ascent approach* (cf. [6]), which is typically applied in local search procedures. However, although local search algorithms belong to the group of

Algorithm 3 Greedy Heuristic

input: *connections* C, *vm capacities* V, *application monitor requirements* R
output: *monitor placements* P

```
 1: procedure GREEDY(C, V, R)
 2:     P ← {}
 3:     SORTDESCREL(C)
 4:     for all c ∈ C do
 5:         app ← app(c)
 6:         sourcevm ← source(c)
 7:         targetvm ← target(c)
 8:         if app ∈ R then
 9:             violation ← CHECKCONSTRAINTS(app, sourcevm, targetvm, V, R, P)
10:             if violation ≠ TRUE then
11:                 P = P ∪ {c}
12:                 UPDATE(app, sourcevm, targetvm, V, R)
13:             end if
14:             n ← remunits(app, R)
15:             if n = 0 then
16:                 R ← R/ {app}
17:             end if
18:         end if
19:     end for
20:     l ← size(R)
21:     if l > 0 then
22:         P ← NULL                    ▷ no feasible solution could be found
23:     end if
24:     return P
25: end procedure
```

improvement procedures, the idea behind our Greedy algorithm is very similar. In each step, the Greedy algorithm tries to improve the partial solution obtained so far to a maximum extent. For this purpose, the set of connections between each VM (sourcevm) where a cloud application (app) to be monitored is running and each VM (targetvm) constituting a candidate for monitor placement is sorted according to decreasing reliability values (line 3). Afterwards, we explore the connections in descending order (line 4). In each step, if all redundant monitoring units for each application have not been placed so far (line 8), we examine, whether we can place a monitoring unit on the targetvm of the current connection. For this purpose, we check, whether any constraints of the RCMPP are violated when the placement is realized. In case that no violation is detected (line 10), we can add the current connection to the result set of final placements (line 11) and update the auxiliary data structures (line 12). If all redundant monitoring units have been placed for a given application, this application is removed from the set R of monitor requirements (line 16). The Greedy algorithm continues to explore further connections until all monitoring units have been placed for each application (line 21).

Algorithm 4 Tabu Search Algorithm

input: *initial solution* S, *iteration limit* max, *connections* C, *vm capacities* V, *application monitor requirements* R
output: *monitor placements* P

```
 1: procedure TSearch(S, C, V, R)
 2:     T ← {}, mp ← FALSE, r ← 1
 3:     bestobjval ← compobjval(S, C)
 4:     while r <= max do
 5:         update(T)
 6:         if mp = TRUE then
 7:             i ← 0
 8:         else
 9:             i ← i + 1
10:         end if
11:         if i < size(R) then
12:             b ← bottleneckapp(i, S, C)
13:         else
14:             return P                    ▷ no improvement can be found anymore
15:         end if
16:         AM ← {}
17:         PM ← placedmonitors(b)
18:         for all pm ∈ PM do
19:             NB_SWAP ← swapneighbourhood(pm)
20:             NB_SHIFT ← shiftneighbourhood(pm)
21:             N ← NB_SWAP ∪ NB_SHIFT
22:             AM ← admissiblemoves(pm, N, V, R, P)
23:         end for
24:         sortdescobjval(AM)
25:         for all am ∈ AM do
26:             if (am ∉ T)or(objval(am) > bestobjval) then
27:                 domove(am)
28:                 T ← T ∪ {am}
29:                 if (objval(am) > bestobjval) then
30:                     bestobjval ← objval(am)
31:                 end if
32:                 mp ← TRUE
33:                 break
34:             end if
35:         end for
36:         r ← r + 1
37:     end while
38:     return P
39: end procedure
```

4.3 Tabu Search Algorithm

Our tabu search algorithm is depicted in Algorithm 4 and is inspired by the work from Karsu and Azizoglu [9], who proposed a tabu search-based algorithm for

the multi-resource agent bottleneck generalised assignment problem. However, the problem they consider deals with workload balancing over a set of agents over multiple periods.

The TSearch algorithm constitutes an improvement procedure, hence, requires an initial solution S as starting point. Our Greedy algorithm can be applied for this purpose. The TSearch algorithm then tries to improve the initial solution over a number of iterations until a predefined number of iterations is reached (line 4). In each iteration, the TSearch algorithm determines a so-called neighbourhood based on the current solution. Basically, a neighbourhood of a given solution is a set of solutions that can be obtained by performing so-called moves. Thereby, a move typically consists of dropping and adding one or more parts (monitor placements in this case) of the current solution. In the TSearch algorithm, we make use of a combined shift and swap neighbourhood (lines 20 and 19). That is, each move either constitutes in shifting a monitoring unit to a different VM or two monitoring units swapping places. For the determination of the neighbourhoods, the TSearch algorithm starts from the cloud application exhibiting the worst total reliability for its monitoring units placed (line 12) and thus, having a pivotal role for the calculation of the lower bound z in our optimization model. We denote this cloud application as *bottleneck application*. However, in case that no shift or swap of at least one of the monitoring units of the bottleneck application is feasible during an iteration, we select the cloud application with the second worst total reliability for its monitoring units placed in the next iteration (line 9). At the end of each iteration, we determine the set of admissible moves with respect to the constraints of the RCMPP among the solutions in the neighbourhood (line 22). From the set of admissible moves, we then either select the move with the highest improvement or lowest decrease with respect to the objective value of the current solution. For this purpose, the list of admissible moves is sorted in decreasing order of objective values (line 24). This is a typical approach when following a tabu search-based procedure. Also a decrease is accepted in order to obtain a different solution, so that the algorithm does not get stuck in a local optimum. However, in order to prevent that the algorithm is running in circles, a global tabu list is maintained (line 2) that forbids the last moves to be performed again for a predefined number of subsequent iterations. Only in case that a move that is currently part of the tabu list would yield a better solution than the current best solution found so far, this move is performed despite being part of the tabu list (line 26).

5 Performance Evaluation

We have implemented our solution approaches in Java and conducted an evaluation in order to assess their applicability for real-world scenarios. For the implementation and evaluation of the ILP-based approach, we used the JavaILP framework[1] and the commercial solver framework IBM ILOG CPLEX[2].

[1] http://javailp.sourceforge.net/

[2] http://www.ibm.com/software/integration/optimization/cplex-optimizer

Table 2. Independent variables and values used in the evaluation

Independent Variable	Symbol and Values
Number of sites:	$\|S'\|, \|S''\| \in \{2, \underline{3}, 4\}$
Number of VMs:	$\|V_s\| \in \{4, \underline{5}, 6, 7, 8\}$
Number of applications:	$\|C_{s'v'}\| \in \{1, \underline{2}, 3\}$
Redundancy factor:	$rf_{s'v'c} \in \{2, \underline{3}, 4\}$

5.1 Evaluation Methodology

In order to assess the practical applicability of our approaches, we examine the two *dependent variables* computation time and solution quality. Solution quality is expressed by the objective value achieved by a solution approach being transformed into the corresponding downtime in seconds on a yearly basis. Furthermore, we consider four *independent variables*, namely, the total number of data center sites on broker and provider side, the number of VMs on each site, the number of cloud applications concurrently running on each VM, and the redundancy factor for the placement of the monitoring units. As evaluation methodology, we follow a fractional factorial design [3]. That is, at each point in time, we only vary one independent variable while keeping the other independent variables fixed. Hence, the impact of each independent variable on the two dependent variables is measured separately. In total, the evaluation consists of 14 test cases depicted in Table 2, each comprising 100 randomly generated problems. For the generation of the problems, we incorporate realistic data including VM capacities and availability guarantees from the specifications of Amazons EC2 VM offers[3], as well as packet loss statistics from the PingER project[4] to model link reliability. In addition, we consider only one resource type, CPU, since our early experiments have shown that CPU is the primary determinant for placing the monitoring units. Furthermore, we choose each application out of three application types, each exhibiting different CPU requirements for the monitoring units to be placed. We also obtained the CPU requirements from our early experiments. Synthetic VM workloads based on the work by [13] are used to determine the remaining VM resource supplies. Each problem was solved using our three solution approaches. In addition, we added a random solution approach, which conducts a random placement of the monitoring units, while only considering adherence with all constraints. For the solution of each problem and each optimization approach, we set a timeout of 5 minutes, which can be perceived as a realistic value in the context of our broker-based scenario and on-demand cloud service provisioning. For the GTSearch heuristic, we set the maximum number of iterations to 1000 and the tabu tenure to 50 based on [9].

[3] http://aws.amazon.com/ec2/
[4] http://www-iepm.slac.stanford.edu/pinger/

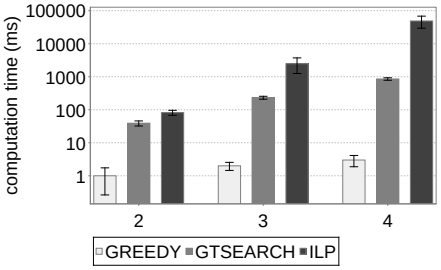

Fig. 1. No. of sites

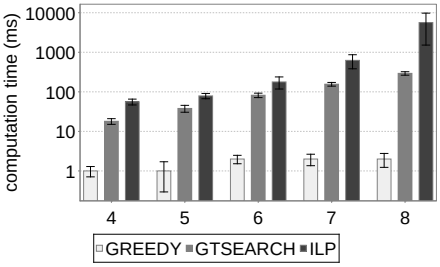

Fig. 2. No. of VMs

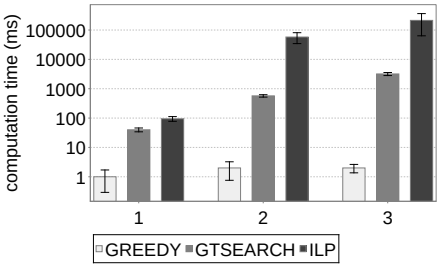

Fig. 3. No. of applications

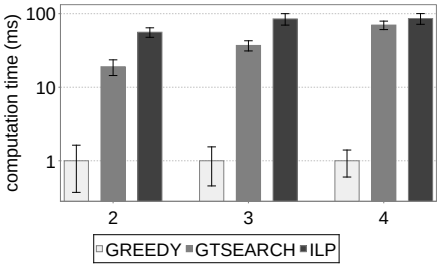

Fig. 4. Redundancy factor

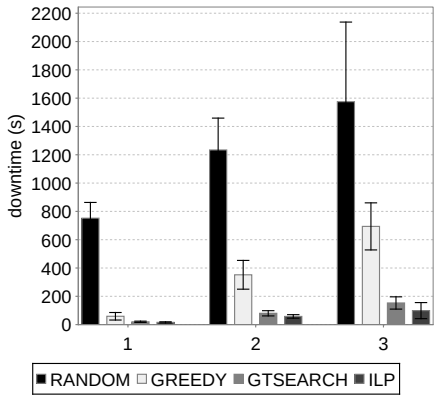

Fig. 5. No. of applications

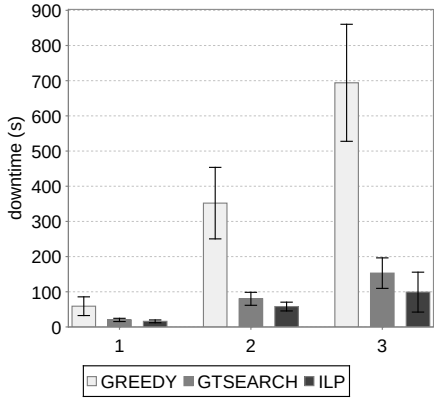

Fig. 6. No. of applications (magnified view)

5.2 Simulation Results

Figures 1 to 6 depict selected results of the evaluation. Please note the logarithmic scale in the first four figures.

When using the ILP approach, the computation time shows an exponential growth with increasing problem size, e.g., 100ms up to 10000ms in Fig. 1. However, this effect is considerably less when increasing the redundancy factor. All in all, the exponential growth underlines the fact that the RCMPP is NP-complete.

Hence, the applicability of the ILP approach in practice is very limited, since the size of the problems considered in the evaluation is relatively small. Nevertheless, the ILP approach can serve as a baseline in order to assess the heuristic approaches. In comparison to the ILP approach and the GTSearch heuristic, the Greedy heuristic performs best with respect to computation time and yields a linear growth with increasing problem size. The GTSearch heuristic also shows smaller values in computation time than the ILP approach. This effect is most pronounced when the number of cloud applications increases (cf. Fig. 3). However, the GTSearch heuristic exhibits no linear growth with respect to problem size like the Greedy heuristic (cf. Fig. 2). Therefore, a further improvement of this heuristic with respect to computation time will be considered in future work, since this heuristic performs best, besides the ILP approach, with respect to solution quality (cf. Fig. 5). In comparison, the Greedy heuristic although showing the best computation times performs worse regarding solution quality with increasing complexity of the problem (cf. Fig. 6 for a magnified view). Nevertheless, the Greedy heuristic still achieves a considerably large improvement over a random placement. The results of a random placement of monitoring units are depicted in Fig. 5 and emphasize the need for heuristic solutions. Without conducting any optimization, the monitoring units would end up, e.g., with a downtime of 25 minutes (on a yearly basis) in contrast to a few seconds when using the other approaches in case of 3 cloud applications deployed on each VM. A result which is unacceptable when business critical applications are utilized.

6 Summary and Outlook

When using resources from the cloud, the shift of responsibility to the cloud provider is attended with a loss of control for cloud consumers. Hence, we have developed an approach to monitor compliance with SLAs from a consumer's perspective in our former work and introduced the *Robust Cloud Monitor Placement Problem* (RCMPP), since the monitoring system itself may also fail. In this paper, we investigated three different solution approaches for the RCMPP: an ILP-based approach, a Greedy heuristic, and the Greedy heuristic in conjunction with a tabu search-based improvement procedure (GTSearch). Our simulation results confirmed the practical inapplicability of the ILP-based approach. Nevertheless, it was used as a baseline for assessing the developed heuristics. All in all, only the GTSearch heuristic is able to achieve near optimal results, but exhibits no linear growth in computation time in contrast to the Greedy heuristic. Therefore, we will explore the improvement of our heuristics in future work.

Acknowledgments. This work was supported in part by the German Federal Ministry of Education and Research (BMBF) under grant no. "01|C12S01V" in the context of the Software-Cluster project SINNODIUM (www.software-cluster.org), the German Research Foundation (DFG) in the Collaborative Research Center (SFB) 1053 – MAKI, and the E-Finance Lab Frankfurt am Main e.V. (http://www.efinancelab.com).

References

1. Andreas, A.K., Smith, J.C.: Mathematical Programming Algorithms for Two-Path Routing Problems with Reliability Considerations. INFORMS Journal on Computing 20(4), 553–564 (2008)
2. Bin, E., Biran, O., Boni, O., Hadad, E., Kolodner, E., Moatti, Y., Lorenz, D.: Guaranteeing High Availability Goals for Virtual Machine Placement. In: 31st International Conference on Distributed Computing Systems (ICDCS). pp. 700–709 (2011)
3. Box, G.E.P., Hunter, J.S., Hunter, W.G.: Wiley, 2nd edn. (2005)
4. 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)
5. CSA, ISACA: Cloud Computing Market Maturity. Study Results. Cloud Security Alliance and ISACA (2012), `http://www.isaca.org/Knowledge-Center/Research/ResearchDeliverables/Pages/2012-Cloud-Computing-Market-Maturity-Study-Results.aspx`, [last access: 30 May 2014]
6. Hillier, F.S., Liebermann, G.J.: Inroduction to Operations Research. McGraw-Hill, 8th edn. (2005)
7. Jensen, P.A., Bard, J.F.: Appendix A: Equivalent Linear Programs. In: Supplements to Operations Research Models and Methods. John Wiley and Sons (2003), `http://www.me.utexas.edu/~jensen/ORMM/supplements/units/lp_models/equivalent.pdf` [last access: 30 May 2014]
8. Kamal, J., Vazirani, V.V.: An Approximation Algorithm for the Fault Tolerant Metric Facility Location Problem. In: Jansen, K., Khuller, S. (eds.) Approximation Algorithms for Combinatorial Optimization, Lecture Notes in Computer Science, vol. 1913, pp. 177–182. Springer (2000)
9. Karsua, z., Azizoglua, M.: The Multi-Resource Agent Bottleneck Generalised Assignment Problem. International Journal of Production Research 50(2), 309–324 (2012)
10. Natu, M., Sethi, A.S.: Probe Station Placement for Robust Monitoring of Networks. Journal of Network and Systems Management 16(4), 351–374 (December 2008)
11. Patel, P., Ranabahu, A., Sheth, A.: Service Level Agreement in Cloud Computing. Tech. rep., Knoesis Center, Wright State University, USA (2009)
12. Sharma, P., Chatterjee, S., Sharma, D.: CloudView: Enabling Tenants to Monitor and Control their Cloud Instantiations. In: 2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013). pp. 443–449 (2013)
13. Shrivastava, V., Zerfos, P., Lee, K.W., Jamjoom, H., Liu, Y.H., Banerjee, S.: Application-aware Virtual Machine Migration in Data Centers. In: Proceedings of the 30th IEEE International Conference on Computer Communications (INFOCOM 2011). pp. 66–70 (April 2011)
14. Siebenhaar, M., Lampe, U., Schuller, D., Steinmetz, R.: Robust Cloud Monitor Placement for Availability Verification. In: Helfert, M., Desprez, F., Ferguson, D., Leymann, F., Muoz, V.M. (eds.) Proceedings of the 4th International Conference on Cloud Computing and Services Science (CLOSER 2014). pp. 193–198. SciTe Press (April 2014)
15. Siebenhaar, M., Wenge, O., Hans, R., Tercan, H., Steinmetz, R.: Verifying the Availability of Cloud Applications. In: Jarke, M., Helfert, M. (eds.) Proceedings of the 3rd International Conference on Cloud Computing and Services Science (CLOSER 2013). pp. 489–494. SciTe Press (May 2013)