Ronny Hans, David Steffen, Ulrich Lampe, Dominik Stingl, and Ralf Steinmetz:
Short Run: Heuristic Approaches for Cloud Resource Selection. In: Proceedings of the 9th International Conference on Cloud Computing (CLOUD 2016), Institute of Electrical and Electronics Engineers (IEEE), June 2016.

Short Run: Heuristic Approaches for Cloud Resource Selection

Ronny Hans, David Steffen, Ulrich Lampe, Dominik Stingl, and Ralf Steinmetz
Multimedia Communications Lab (KOM)
Technische Universitaet Darmstadt
Darmstadt, Germany
Email: ronny.hans@kom.tu-darmstadt.de

***Abstract*—Using cloud computing for IT service provisioning has become a common practice over the last years. Besides basic infrastructure services, more advanced multimedia services with high Quality of Service (QoS) requirements are offered. To fulfill such requirements, appropriate cloud resources must be used for service provisioning. In this paper, we analyze different heuristic approaches to speed up the selection procedure of cloud resources while ensuring a high solution quality at the same time. In the work at hand, we present a Best-of-Breed approach, which is assembled by simple heuristics. Further, we adopt an advanced metaheuristic approach, i. e., tabu search, for the given problem and compare it with the former mentioned approach. With our approaches, we offer the means for a cloud provider to select appropriate resources from a large pool to facilitate QoS-aware multimedia service provisioning.**

***Index Terms*—cloud computing, data center, quality of service, multimedia, service, heuristic, tabu search**

I. Introduction

Over the last decade, cloud computing has developed from the idea to sell a resource surplus of Information Technology (IT) resources to a common paradigm for IT service provisioning. In accordance, the amount of services that are provided via cloud data centers has grown rapidly over the past years. Cisco predicts that the ratio of overall Internet traffic caused by cloud data centers grows from 61% in 2014 to a share of 83% in 2019 [1]. Along with the increasing *quantity* of provided cloud services, the non-functional *quality* requirements, i. e., Quality of Service (QoS), also increase. For a centralized cloud infrastructure with a small number of large data centers, low-latency applications, such as cloud gaming, represent a major challenge [2]. Specifically for multimedia services such as cloud gaming, data centers and compute resources close to the potential users are required, which subsequently leads to a growing number of cloud resources. Regarding the definition of cloud computing, compute resources are assumed to be unlimitedly available [3]. Nevertheless, with increasing requirements regarding functional and non-functional aspects of multimedia services, resources which are able to fulfill these requirements regarding a specific group of users are limited.

Thus, in our research, we analyze approaches for efficiently planning new as well as for selecting existing resources from cloud infrastructures. Thereby, we take the point of view of a single cloud provider which aim to optimize its infrastructure. In our past work we published a heuristic framework to find valid solutions for this specific optimization problem [4]. These heuristics are characterized by a high performance and very good solution quality. Nevertheless, regarding the nature of heuristics, no guarantee whether a solution is close to a optimal resource assignment can be given. Therefore, in the work at hand, we investigate different heuristic approaches to find possibilities to improve and guarantee the solution quality. For that matter, we propose a Best-of-Breed heuristic, as well as an improvement procedure based on tabu search. Further, we systematically compare these approaches regarding performance and solution quality.

The remainder of this paper is structured as follows: In Section II, we give a problem description and briefly present our optimization model. We also provide a brief summary of our previously published approaches. In Section III, we introduce our Best-of-Breed approach, and in Section IV, we present the tabu search heuristic. These approaches are subsequently evaluated in Section V. An overview of related work is given in Section VI. Section VII concludes the paper with a brief summary and an outlook on our future work.

II. Problem Statement and Model

In the following sections, we briefly describe our previously published problem, mathematical model and approaches.

A. Problem Statement

In this work, we consider a provider who aims to use a set of (potential or existing) cloud resources, i. e., data centers, to offer the cloud infrastructure for multimedia service providers. The data centers are located in different geographical areas. Thereby, each data center may provide resource units between a lower and an upper bound. Using a data center and it resources results in certain fixed and variable costs. The resources are characterized by predefined QoS attributes and corresponding QoS guarantees with respect to each user cluster and each QoS attribute.

The amount of provisioned resources are determined by a given demand. Each user cluster constitutes a specific demand. A provider faces the challenge of the cost-minimal and QoS-aware selection of appropriate resources. Thereby, we assume that the overall service demand of all user clusters and the corresponding QoS requirements must be matched. In our

former work, we referred to this problem as *Cloud Data Center Selection Problem* (CDCSP) [5].

B. Mathematical Model

In this subsection, we briefly describe the mathematical model, which is based on our former work [5]. The presented model constitutes a static, single period optimization problem. To start with, we describe the required formal notations, beginning with the basic entities:

- $D = \{1, 2, ..., D^{\#}\}$: Set of data centers
- $U = \{1, 2, ..., U^{\#}\}$: Set of user clusters
- $Q = \{1, 2, ..., Q^{\#}\}$: Set of QoS attributes

With respect to the basic entities, the associated parameters can be defined as follows:

- SD_u: Service demand of user cluster u
- $K_d^{min} \in \mathbb{R}$: Minimal capacity of data center d
- $K_d^{max} \in \mathbb{R}$: Maximal capacity of data center d
- $CF_d \in \mathbb{R}$: Fixed costs of selecting data center d
- $CVO_d \in \mathbb{R}$: Variable operational costs per resource unit in data center d
- $QG_{d,u,q} \in \mathbb{R}$: QoS guarantee of data center d w.r.t. user cluster u for QoS attribute q
- $QR_{u,q} \in \mathbb{R}$: QoS requirement of user cluster u w.r.t. QoS attribute q

To model the CDCSP as an optimization problem, the following decision variables are introduced:

- x_d: Selection of a data center d
- $y_{d,u}$: Capacity provided by data center d to user cluster u

In the model, Equation 1 defines the objective of the problem, i. e., the minimization of total cost C, which depends on the decision variables x_d and $y_{d,u}$ (Equation 7). The *binary* variables x_d indicate if data center d will be used for service provisioning. $y_{d,u}$ are *integer* variables that denote the number of resource units data center d provides to user cluster u.

Equation 2 constitutes the constraint that the service demand of each user cluster needs to be satisfied. Equations 3 and 4 define the given lower bound K_d^{min} and the given upper bound K_d^{max} of the available capacity for each data center. In addition, they represent the link between the decision variables x and y. In Equation 5 and Equation 6, the variables $p_{d,u}$ restrict the resource allocation between data centers and user clusters, depending on the fulfillment of the QoS requirements.

C. Previous Approaches

In the given exact formulation, the problem uses binary and integer decision variables (cf. Equation 7), resulting in an Integer Program (IP). Such an IP can be solved using off-the-shelf algorithms, such as branch-and-bound [6]. However, the computation time for such IPs grows exponentially with the number of decision variables. To overcome this drawback, we also introduced a simple heuristic approach based on the common concept of Linear Program (LP) relaxation in our past research [7]. Thereby, the binary and integer decision variables are substituted by corresponding real variables (cf. Equation 8), thus trading potentially non-optimal solutions against substantial reductions in computation time. Since this basic heuristic approach do not consider the specific structure of the described problem, we introduced and evaluated priority-based approaches in our past work [4]. The published heuristic framework is based on priority and cost allocation rules and puts us into the position to assemble numerous heuristics, each of them with a different solution quality and a different performance, depending on the given test case.

Model 1 Cloud Data Center Selection Problem

$$\text{Min. } C(x,y) = \sum_{d \in D} x_d \times CF_d + \sum_{d \in D, u \in U} y_{d,u} \times CVO_d \tag{1}$$

$$\sum_{d \in D} y_{d,u} \geq SD_u \quad \forall u \in U \tag{2}$$

$$\sum_{u \in U} y_{d,u} \leq x_d \times K_d^{max} \quad \forall d \in D \tag{3}$$

$$\sum_{u \in U} y_{d,u} \geq x_d \times K_d^{min} \quad \forall d \in D \tag{4}$$

$$y_{d,u} \leq p_{d,u} \times K_d^{max} \quad \forall d \in D, \forall u \in U \tag{5}$$

$$p_{d,u} = \begin{cases} 1 & \text{if } QG_{d,u,q} \leq QR_{u,q} \quad \forall q \in Q \\ 0 & \text{else} \end{cases} \tag{6}$$

$$\begin{aligned} x_d &\in \{0,1\} \quad \forall d \in D \\ y_{d,u} &\in \mathbb{N} \quad \forall d \in D, \forall u \in U \end{aligned} \tag{7}$$

$$\begin{aligned} x_d &\in \mathbb{R}, 0 \leq x_d \leq 1 \quad \forall d \in D \\ y_{d,u} &\in \mathbb{R}, y_{d,u} \geq 0 \quad \forall d \in D, \forall u \in U \end{aligned} \tag{8}$$

III. Best-of-Breed Heuristic

With this approach, we aim to ensure a better and more constant solution quality compared to single heuristics while maintaining a favorable performance. The basic idea behind the approach is the efficient application of different heuristics for the same problem. The predefined heuristics are executed subsequently or in parallel for an identical problem instance. Out of all different solutions, we select the one with the best solution quality, i. e., lowest total cost.

Since the input parameters in a real life scenario are unknown in advance, the solution quality between the individual heuristics may differ. In principle, each possible heuristic approach may be included within this *Best-of-Breed* heuristic. Nevertheless, depending on the number of the used heuristics,

their characteristics, and the implementation of the Best-of-Breed approach, the computation time increases. Thus, a wise selection of the included heuristics is required.

In this work, we use the heuristic approaches that we published in our former work [4]. On one hand, the controllable factor for solution quality and performance is the configuration of each heuristic, which is easy to determine for a predefined set of input parameters. On the other hand, input parameters such as demand and resource utilization, which in real life scenarios are out of our control sphere, influence solution quality and performance as well. Our past work shows that these parameters cause different best-fitting configurations depending on the test case. In the subsequent sections, we describe our procedure to assemble appropriate heuristics for this approach.

A. Solution Quality

To determine a set of heuristics that should be considered within the Best-of-Breed approach, we evaluate a set of existing heuristic regarding there solution quality and performance. First, for each evaluated test case, we select the one heuristic with the best average solution quality compared to the exact solution. We compare this heuristic to all other heuristics using the statistical instrument of paired t-tests [8]. Based on this approach, we are able to decide whether a given heuristic approach differs statistically significant from the selected best approach or not. As result, we get a group of heuristics that deliver the *best* solution quality, i. e., a group of heuristics with no statistically significant difference compared to the average best solution.

B. Performance

Out of the previously described group, we select the one with the best performance, i. e., lowest computation time. We followed the same procedure like we did with the solution quality. We compared the heuristic with the best performance to all other heuristics to identify the best *and* the fastest heuristics for each test case. The identified heuristic approaches are included in our Best-of-Breed approach. With this methodology, for a given set of problem instances, it is obvious that the described Best-of-Breed approach should provide a better or equal solution quality compared to a single heuristic. However, it also causes a higher computational effort. In our prototypical implementation, we execute the single heuristics subsequently, thus increasing the computation time with each additional heuristics. In scenarios with high performance requirements, a parallel execution may be favorable. In the evaluation section, we apply the Best-of-Breed approach to different test cases and compare the results with the metaheuristic *tabu search*, which is described in the following section.

IV. Tabu Search Heuristic

The second approach we evaluate is the metaheuristic *tabu search*, which was introduced by Fred Glover in 1986 [9]. It is used to guide a local search procedure with the aim to overcome *local* optima and to find a solution close to the *global* optimum. For the local search procedure, it can make use of various methods, such as approximation procedures [10]. Fig. 1 illustrates the different components of the tabu search approach we used. These core elements and further configuration of our approach are explained in the following subsections.

A. Initial Solution

Since the tabu search heuristic is an improvement procedure, it requires an initial solution as a starting point. Further, it guides a local search procedure. In both cases, appropriate heuristic approaches are required. For that purpose, we make use of one of the heuristic approaches we determined within the statistical analysis presented in Section III.

B. Local Search Procedure

The local search approach requires several steps, such as defining the neighborhood, analyzing the neighborhood, selecting an appropriate solution, updating the corresponding data structure for every iteration, and checking the stop conditions.

1) Defining the Neighborhood: The first step of the local search procedure is the definition of the neighborhood. To determine the neighborhood $\mathrm{NB}(x)$ of the current solution $x \in \mathbf{X}$, specific moves must be defined. Thereby, the size of the neighborhood and next solution depends on the predefined moves. In our case, we use single-attribute-moves, i. e., use or not use a data center. Thus, with a given number of m possible data centers, the neighborhood is restricted to m different possible solutions of the assignment problem.

2) Analyzing of Neighborhood Solutions: To determine the neighborhood, we use candidate lists [11]. The list $D^{\text{unused}}_{x_{curr}}$ includes all currently unused data centers that may be *opened*. The list $D^{\text{used}}_{x_{curr}}$ includes all currently used data centers, which are possible candidates to *close*. Thus, each list stores the required attributes to transform a current solution $x \in \mathbf{X}$ into an adjacent solution $x' \in \mathrm{NB}(x)$. Consequently, an adjacent solution differs in exactly one attribute from the current solution. We sort our candidate lists by total cost per resource unit (cf. Equation 9).

$$\overline{c(d)} = \mathrm{CV}_d + \mathrm{CF}_d \cdot \frac{1}{K_d^{\max}} \tag{9}$$

The candidate list $D^{\text{used}}_{x_{curr}}$ is sorted in descending order, since we assume that more expensive data centers are less likely part of the optimal solution. In contrast, the list $D^{\text{unused}}_{x_{curr}}$, which contains the currently unused data centers, is sorted in ascending order, since we are assume that less expansive data centers are more likely part of the optimal solution.

In each iteration, both lists are analyzed step by step. The analysis of a list stops once the calculated total cost starts to increase. That is, when the first calculated neighborhood solution causes higher cost, the neighborhood search stops and the current solution will be used as a result of the current iteration. Thus, the size of the analyzed neighborhood depends

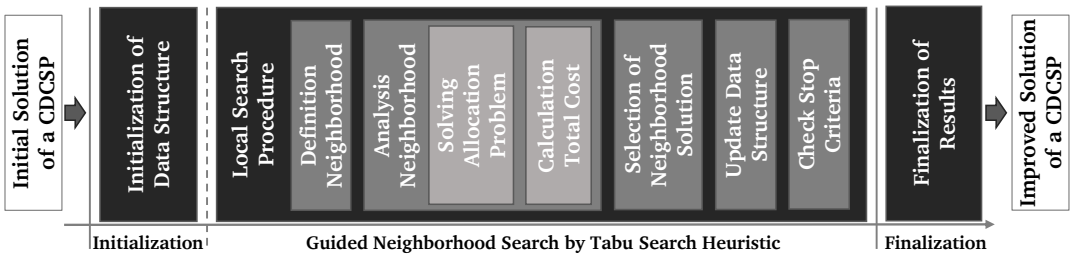

Fig. 1. Components of the Tabu Search Heuristic

Algorithm 2 Analysis of the Neighborhood

Start: Current valid solution $x_{\text{curr}} \in \mathbf{X}$ and sorted candidate lists $D^{\text{used}}_{x_{curr}}$, $D^{\text{unused}}_{x_{curr}}$

1: $x_{\text{fav}} \leftarrow$ Null; costs$(x_{\text{fav}}) \leftarrow \infty$
2: **for all** $d \in D^{\text{unused}}_{x_{curr}}$ **do**
3: $x' \leftarrow$ transformSolution(d, x_{curr})
4: **if** $x \notin$ tabu list *or* satisfiesAspirationCriterium(x') **then**
5: **if** cost(x') $<$ cost(x_{fav}) *or* cost(x_{fav}) $= \infty$ **then** $x_{\text{fav}} \leftarrow x'$
6: **else** stop analyzing candidate list $D^{\text{unused}}_{x_{curr}}$ **end if**
7: **end if**
8: **end for**
9: **for all** $d \in D^{\text{used}}_{x_{curr}}$ **do**
10: $x' \leftarrow$ transformSolution(d, x_{curr})
11: **if** $x' \notin$ tabu list *or* satisfiesAspirationCriterium(x') **then**
12: **if** cost(x') $<$ cost(x_{fav}) *or* cost(x_{fav}) $= \infty$ **then** $x_{\text{fav}} \leftarrow x'$
13: **else** stop analyzing candidate list $D^{\text{used}}_{x_{curr}}$ **end if**
14: **end if**
15: **end for**

on the development of the total cost. Algorithm 2 shows the corresponding pseudo code.

In line 3 and line 10, we open or close a data center, respectively. Thus, a new resource assignment is determined, i. e., assignments are shifted between data centers. The allocation of the resources causes different cost compared to the previous solution. The new costs are compared with the current cost of the solution in line 6 and 13, respectively. To overcome local optima, tabu search allows moves that temporarily lead to worse solutions. So, if no better solution is available, we accept a worse one as well.

The methods *transformSolution* in our pseudo code are simple representations of the (re-)assignment process. In practice, to reduce computation effort in this step, we use different procedures depending on the conducted move: For an additional data center, we can use a simplified calculation procedure for the neighborhood solution because of the relaxed capacity constraints.

First, we add all resource assignments to the list $Y_{x_{curr}}$ and check whether the additional data center is able to satisfy the QoS requirements. Those resource units $y_{d,u}$, that are in line for a new assignment are added to the list $Y^{\text{toCheck}}_{x_{curr}}$. This list is sorted in descending order by variable cost. Afterwards, we compare the variable cost CV_d between present data centers and the new one for the assignments in $Y^{\text{toCheck}}_{x_{curr}}$. If the variable cost for the new data center is less then the current variable cost, the resource assignment is changed in favor of the new data center. This procedure is repeated until the variable cost of a new data center exceeds the current variable cost or until the maximal capacity $K^{\max}_d$ is reached.

Closing a data center reduces the available resources and requires a complete recalculation of the resource assignment. In this case, we use again a heuristic, which was selected following the procedure stated in Section III. Since the amount of available resources is reduced, it is possible that a neighborhood solution does not satisfy condition 2 from our model and causes an invalid solution. In such a case, we assume the cost as infinite and the solution will not be taken into account.

C. Tabu List

The name giving and an essential component of the tabu search is the *tabu list*. Thereby, the tabu list prevents the algorithm from returning to an already visited solution within a given number of iterations. In our approach, we use an attributive tabu list, i. e., a recency-based memory [12]. This type of tabu list only maintains attributes of solutions that have changed in the recent past. In our case, this corresponds to the *id* of an data center $d \in D$, which was recently opened or closed. This data center will be added to the tabu list and its state cannot be changed for a given number of iterations. Regarding the first-in-first-out principle, a stored value is removed from the tabu list when the maximum number of entries is reached [13].

Thereby, the determination of the tabu list size is a major challenge. A tabu list which is too short leads to circles within the local search. A list that is too long results in a big amount of possible solution that are tabu and causes an early termination of the local search and thus, a bad solution quality [11].

For our implementation, we use a tabu list that considers properties of the current problem instance. We set the tabu list size while considering the number of all possible data centers $|D|$. Thereby, the size of the tabu list must be shorter than the number of data centers to avoid that all possible solutions are part of the tabu list. Further, the growth of the list must be

smaller then the number of data centers to permit extensive problem instances. Based on this, we choose a size of $\frac{1}{2} \cdot \sqrt{|D|}$ for our experiments, which corresponds with findings in the literature [14]. The analysis of this parameter is stated in Section V-B1.

Further, there is also a possibility to overrule the tabu list which is named *Aspiration Criteria*. Such criteria can come into force when a so far unknown better solution exists in the neighborhood, even when the corresponding move is prohibited by the tabu list [10]. In Algorithm 2 in Line 4 it is denoted as *satisfiesAspirationCriterium(x')*.

D. Long Term Memory

Besides the short term memory, i. e., tabu list, we use a long term memory. Such a memory stores all solutions or attributes of a solution that have been computed so far. Before calculating a neighborhood solution, the long term memory is consulted to check if the solution has already been calculated during one of the past iterations. Thus, additional computation effort can be avoided. Literature show that the effort for maintaining a long term memory is negligible compared to (re-)solving an assignment problem [15].

E. Stopping Conditions

As a metaheuristic, tabu search does not have a natural stopping condition. To terminate the search process, Glover and Taillard [11] distinguish between four possible stopping conditions:

- No valid solution exists in the neighborhood
- The maximum number of iterations has been reached
- A specific number of iterations was performed without a predefined improvement of the solution
- The optimal solution was found.

The objective in using one or more efficient stopping conditions is to accomplish a good trade-off between solution quality and required computational effort. In our tabu search heuristic, we use a combination of the first three principles of above list, whereby the third one in a adapted manner.

The first one occurs if no valid solution exists in the neighborhood of the current solution or if all possible solutions are part of the tabu list. In this case, the local search is stopped. In the second case, the tabu search ends after a given number of iterations. For our approach, this parameter is based on the number of available data centers of a specific problem instance. The predefined number of static iterations is determined by Equation 10. A brief evaluation of this parameter is stated in Section V-B1.

$$\text{Iter}_{\text{stat}} = \min(0.5 \cdot |D| \ , \ 2 \cdot \sqrt{|D|}) \tag{10}$$

Regarding the last principle, we add further iterations depending on the improvement of the objective. We assume, that a large improvement of the objective value enables further improvements. Hence, if we are able to find a better solution within an iteration, we increase the total number of iterations by Iter_{add}, where the latter is computed using Equation 11.

$$\text{Iter}_{\text{add}} = (1 - c(x')/c(x)) \cdot \sqrt{\text{Iter}_{\text{stat}}} \cdot 10 \tag{11}$$

With this approach, we merge the second and the third stop condition in the list and get an adaptive number of iterations. If one of the stop conditions is fulfilled, the local search is terminated. The best solution that is stored in the *long term memory* becomes the final result of the tabu search.

V. Evaluation

A. Setup

In order to evaluate the previously described heuristic approaches, we prototypically implemented them in Java 8. For the optimal and the LP-relaxed approach, we used IBM ILOG CPLEX 12.5[1], which was accessed through the JavaILP middleware[2]. The evaluation was conducted on a workstation, equipped with an Intel Xeon CPU E5-1620 v3 with 3.50 GHz and 16 GB of memory, operating under Microsoft Windows 7.

Our evaluation focused on two dependent variables, namely, total costs to assess the solution quality and computation time to assess the performance. As independent variables, we considered the number of data centers and the number of user clusters. Problem instances were generated based on the 2010 United States census[3]. Using these data, we set the service demands and different cost parameters according the population of a randomly selected county and its median income. As QoS parameter, we take latency into account and set it corresponding to the requirements for multimedia services.

B. Results and Discussion

This section is divided into three parts. First, we evaluate parameters of the tabu search algorithm, i. e., the size of the tabu list and the number of iterations. Second, we compare the different approaches regarding solution quality and performance. In the last part, we evaluate the approaches regarding *large* test cases.

1) Tabu Search Parameters: The tabu search heuristic can be varied by several parameters. Such settings determine the solution quality and the performance of the heuristic. In this evaluation, we focus on the tabu list size and the number of iterations. In Fig. 2 and 3, fixed values for the tabu list size from one to twenty were chosen to determine its influence on solution quality and performance. To minimize the influence of the number of iterations, we selected a fixed value of twenty iterations for each test case, independent of the number of data centers. Fig. 2 illustrates the influence of the tabu list size regarding the solution quality. Clearly to recognize are the high cost ratios for the tabu list size of one. Subsequently, with an increasing size, the cost ratios for all test cases decrease and thus the solution quality is improved. Afterwards, the cost ratios increase again.

Reasons may include a cycling behavior for small tabu list size and a high amount of forbidden solutions for a high value.

[1] http://www.ibm.com/software/integration/optimization/cplex-optimizer/
[2] http://javailp.sourceforge.net/
[3] http://www.census.gov/geo/maps-data/data/gazetteer.html

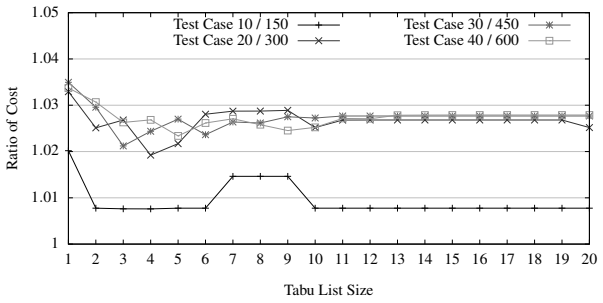

Fig. 2. Ratio of costs between the exact approach and the tabu search heuristic by predefined tabu list size.

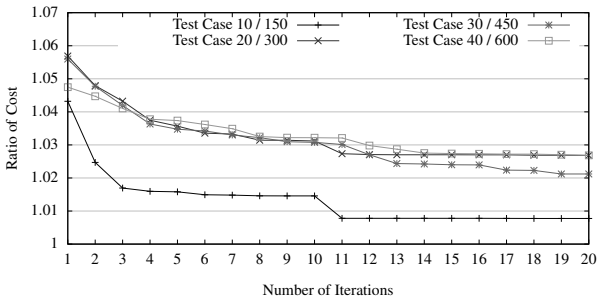

Fig. 4. Ratio of costs between the exact approach and the tabu search heuristic by predefined number of iterations.

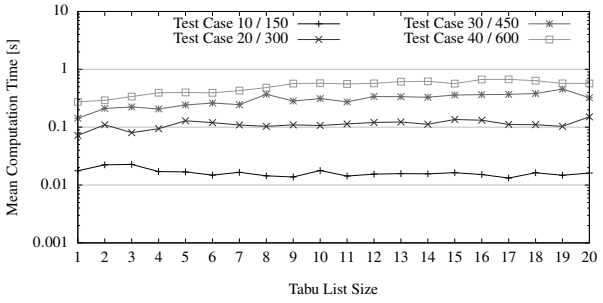

Fig. 3. Observed mean computation times by predefined tabu list size. Please note the logarithmic scaling of the ordinate.

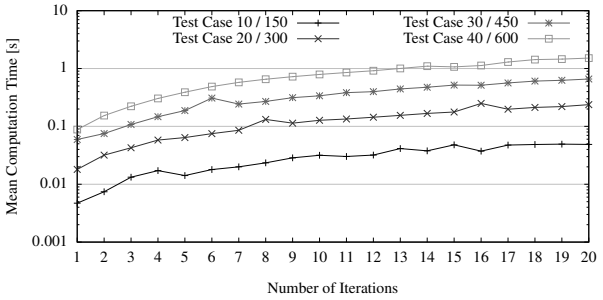

Fig. 5. Observed mean computation times by predefined number of iterations. Please note the logarithmic scaling of the ordinate.

Further, as illustrated in Fig. 3, the tabu list size has just a minor influence on the computation time. Thus, there is nearly no trade-off between solution quality and performance regarding this parameter. It is more important to find a favorable value for each problem instance. The used formula $\frac{1}{2} \cdot \sqrt{|D|}$ meets this criteria quite well, nevertheless, it does not necessarily constitute the ideal point for each test case. Furthermore, we evaluated the number of iterations. For that matter, we determine the value for the tabu list size using the above mentioned formula and vary the number of iterations from one to twenty (cf. Fig. 4 and 5). In the figures, a clear trade-off between solution quality and performance is observable. With increasing number of iterations, the solution quality improves by a decreasing rate. Since the computation time may increase by factor ten or more, a adequate compromise is required (cf. Equation 10).

For both evaluations, tabu list size and number of iterations, the fluctuations in solution quality for the smallest test case (10/150) are outstanding. The reason for this phenomena can be see in the small number of data centers. A change in the used data centers may result in a substantial change in solution quality.

2) Comparison of the Different Approaches: For our analysis, we use our previously published exact approach (CDCSP-EXA.KOM), our LP-relaxed approach (CDCSP-REL.KOM), and our priority-based start heuristic (CDCSP-PBSH.KOM) as benchmarks. We compare these approaches regarding performance and solution quality to the heuristic approaches presented in this paper. For each test case we created 100 problem instances. We subsequently computed the observed mean absolute computation times and the macro-averaged ratio of total cost along with the respective 95% confidence intervals based on a t-distribution.

Fig. 6 shows the solution quality of our approaches, whereby three main observations need to be mentioned. First – except of the first test case (10/150) – the solution quality of all heuristics compared to the exact approach improves with an increasing number of data centers and user clusters and thus tends towards the optimal solution. The second observation refers to the differences between the tabu search heuristic (CDCSP-TS.KOM) and the Best-of-Breed approach (CDCSP-BoB.KOM). Regarding the first test case (10/150), the difference between these two corresponds to 2.4 percentage points. Up to the last test case (40/600), the difference continuously shrinks to 1.0 percentage points. Further, the average solution quality of the tabu search heuristic always exceeds the one of the Best-of-Breed approach. The third observation refers to the differences between the priority-based start heuristic and the approaches presented in the work at hand. Both, the tabu search heuristic as well as the Best-of-Breed approach make use of this simple heuristic, which was selected using the statistical analysis in Section III. Although the chosen heuristic was selected as best and fastest for one test case, the average results of the tabu search heuristic and the Best-of-Breed approach are better for *every* test case. Thus, the solution quality benefits from these approaches.

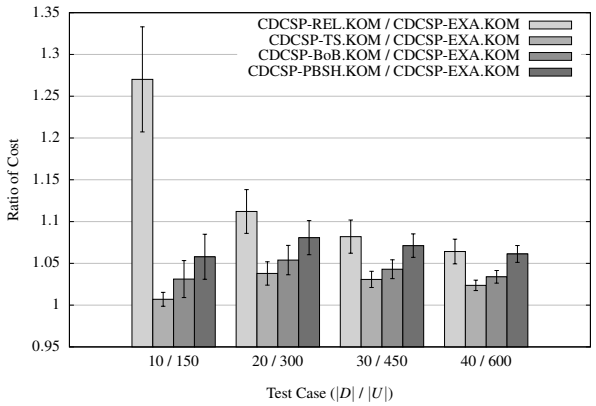

Fig. 6. Ratio of costs (based on macro-average; with 95% confidence intervals) between the exact approach and the heuristic approaches by test case.

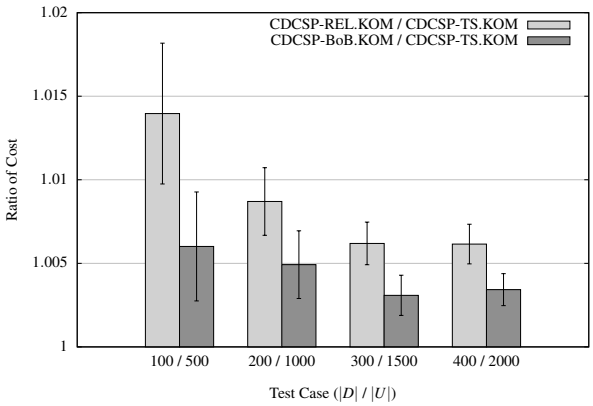

Fig. 8. Ratio of costs (based on macro-average; with 95% confidence intervals) between the tabu search approach and the heuristic approaches by test case.

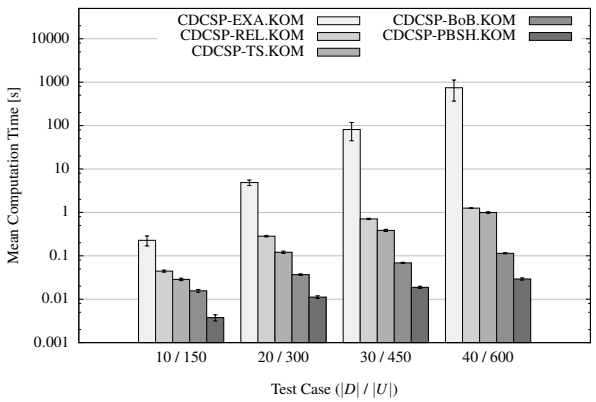

Fig. 7. Observed mean computation times (with 95% confidence intervals) by test case. Please note the logarithmic scaling of the ordinate.

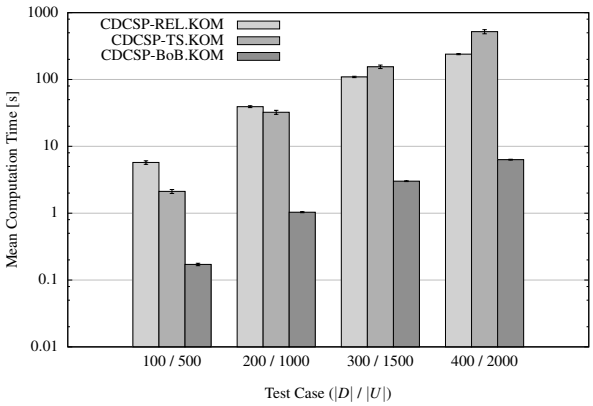

Fig. 9. Observed mean computation times (with 95% confidence intervals) by test case. Please note the logarithmic scaling of the ordinate.

Fig. 7 presents the computation time of five approaches. Clearly recognizable is the increasing difference between the exact approach and each of the heuristic approaches, due to the previously mentioned exponential growth in computation time for the former. Of specific interest to our present work is the growing difference between the tabu search heuristic and the Best-of-Breed approach. For the first test case (10/150), the required computation time for the Best-of-Breed approach is 45.5% less compared to the tabu search. For the last test case (40/600), the Best-of-Breed approach requires 88.5% less computation time. Thus, with an increasing number of data centers and user clusters, the Best-of-Breed approach is more favorable with repsect to computation time.

3) Large Test Cases: In this part, we focus on *large* test cases with hundreds of data centers. Because of the high computation effort the exact solution is not applicable for our evaluation. Thus, we used the LP-relaxed approach for comparison. Furthermore, we only generated 50 problem instances per test case, compared to 100 in the previous parts of the evaluation. Fig. 8 and Fig. 9 show the corresponding results. Regarding the solution quality, the tabu search heuristic achieves the best results. Thus, we used it as benchmark for the cost ratios. As mentioned before, with increasing test case size percentage difference in solution quality decreases and reaches a marginal level. Nevertheless, mapping the percentage difference to absolute cost, the difference may be enormous. A substantial difference between the heuristic approaches is the computation time depicted in Fig. 9. In our configuration, tabu search performs the worst for large test cases. For the last test case (400/2000), tabu search requires in average nearly 38 times longer compared to the Best-of-Breed approach.

To summarize, our tabu search heuristic delivers the best average results for every test case, but with the drawback of significant higher computation time compared to the Best-of-Breed approach. For increasing problem sizes, the advantages regarding the solution quality of the tabu search decreases and the disadvantages regarding the computation time increases. Thus, for calculations with minor time constraints, such as planing scenarios, tabu search may be the best choice. For large infrastructures with a high number of data centers and user clusters as well as fluctuating demand, our Best-of-Breed approach may be more favorable.

VI. Related Work

Cloud resource allocation and data center placement are well studied fields in research. Therefore, in this section, a set of papers is presented which fit best to our current research.

Goiri et al. [16] analyze the placement of data centers with the aim to reduce the total costs of resource provisioning. Therefore, the authors formulate an optimization model to determine cost efficient locations for data centers. The optimization problem is solved by the means of LP relaxation and a simulated annealing heuristic. Thereby, the paper focuses on data center placement and do not consider run time constraints in large environments.

Regarding the current public cloud infrastructure, which is mainly based on a few number of large data centers, Choy et al. [2] demonstrate the need to enhance it by additional resources to improve its multimedia capabilities. The results of their work are based on the analysis of Amazons public cloud infrastructure in the United States. The authors show that only a portion of 70% of the population can use services with stringent latency requirements such as cloud gaming. The authors propose the use of additional data centers or so called Edge Server. Thus, it is possible to improve the coverage by about 30%. In contrast to our research, the authors neither use an optimization approach, nor do they explicitly assign user demands to given cloud resources.

Larumbe and Sansò [17] analyze their so called *Cloud location and routing problem*. For this purpose, the authors formulate an optimization problem which encompasses the location of data centers, the assignment of the software components to servers, and the routing between user and data center. In a subsequent work, Larumbe and Sansò [14] enhanced their analysis regarding resources within data centers to assign server and network resource. Because of the increased complexity of the optimization problem, the authors develop a tabu search heuristic to solve the optimization problem. This approach bears resemblance with our work. However, the authors do not aim to find heuristics with the best trade-off between performance and solution quality to solve the problem under runtime constraints.

In summary, to the best of our knowledge, our work is the first that presents a Best-of-Breed approach based on priority-based heuristics and compare it to a tabu search heuristic. With the focus on QoS-aware services provisioning, we present approaches that manage to maintain a high solution quality while requiring reasonable computational effort at the same time.

VII. Summary and Outlook

In this paper, we presented advanced heuristic approaches to tackle the *Cloud Data Center Selection Problem*. To solve the problem, a variety of different heuristic approaches can be used, which differ in performance and solution quality. To ensure a stable solution quality in combination with high performance, we investigated two major enhancements compared to our former work. Specifically, we presented a Best-of-Breed approach as well as an improvement procedure, i. e., tabu search. We evaluated these approaches and compared them with each other for different problem sizes. Thereby, specifically with the Best-of-Breed approach, we are able to combine the benefits of high solution quality and low computational effort. In the future, we plan to apply this approach for cloudlet-supported environments. These environments are characterized by a high number of small *data centers* and a volatile demand. Furthermore, a combination of the two presented approaches may be promising.

Acknowledgment

This work has been sponsored in part by the German Federal Ministry of Education and Research (BMBF) under grant no. 01IS12054, by E-Finance Lab e.V., Frankfurt a.M., Germany (www.efinancelab.de), and by the German Research Foundation (DFG) in the Collaborative Research Center (SFB) 1053 MAKI.

References

[1] Cisco, "Cisco Global Cloud Index: Forecast and Methodology, 2014-2019," 2015. [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/Cloud_Index_White_Paper.pdf

[2] S. Choy, B. Wong, G. Simon, and C. Rosenberg, "The Brewing Storm in Cloud Gaming: A Measurement Study on Cloud to End-User Latency," in *11th Annual Workshop on Network and Systems Support for Games*, 2012.

[3] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," National Institute of Standards and Technology, Tech. Rep., 2011.

[4] R. Hans, D. Steffen, U. Lampe, B. Richerzhagen, and R. Steinmetz, "Setting Priorities: A Heuristic Approach for Cloud Data Center Selection," in *Proceedings of the 5th International Conference on Cloud Computing and Services Science*, 2015.

[5] R. Hans, U. Lampe, and R. Steinmetz, "QoS-Aware, Cost-Efficient Selection of Cloud Data Centers," in *6th IEEE International Conference on Cloud Computing*, 2013.

[6] W. Domschke and A. Drexl, *Einführung in Operations Research*. Springer, 2004, in German.

[7] R. Hans, "Selecting Cloud Data Centers for QoS-Aware Multimedia Applications," in *PhD Symposium at the 2nd European Conference on Service-Oriented and Cloud Computing*, W. Zimmermann, Ed., 2013.

[8] R. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. Wiley, 1992.

[9] F. Glover, "Future Paths for Integer Programming and Links to Artificial Intelligence," *Computers & Operations Research*, vol. 13, no. 5, pp. 533–549, 1986.

[10] ——, "Tabu Search – Part I," *ORSA Journal on Computing*, vol. 1, no. 3, pp. 190–206, 1989.

[11] F. Glover and E. Taillard, "A User's Guide to Tabu Search," *Annals of Operations Research*, vol. 41, no. 1, pp. 1–28, 1993.

[12] F. Glover, M. Laguna, and R. Marti, "Principles of Tabu Search," *Approximation Algorithms and Metaheuristics*, vol. 23, pp. 1–12, 2007.

[13] T. Grünert and S. Irnich, *Optimierung im Transport: Grundlagen*. Shaker Verlag, 2005.

[14] F. Larumbe and B. Sansò, "A Tabu Search Algorithm for the Location of Data Centers and Software Components in Green Cloud Computing Networks," *IEEE Transactions on Cloud Computing*, vol. 1, no. 1, pp. 22–35, 2013.

[15] M. Sun, "A Tabu Search Heuristic Procedure for the Capacitated Facility Location Problem," *Journal of Heuristics*, vol. 18, no. 1, pp. 91–118, 2012.

[16] Í. Goiri, K. Le, J. Guitart, J. Torres, and R. Bianchini, "Intelligent Placement of Datacenters for Internet Services," in *31st International Conference on Distributed Computing Systems*, 2011.

[17] F. Larumbe and B. Sansò, "Optimal Location of Data Centers and Software Components in Cloud Computing Network Design," in *12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, 2012.