# Cost-driven Optimization of Complex Service-based Workflows for Stochastic QoS Parameters

Dieter Schuller, Ulrich Lampe, Julian Eckert, Ralf Steinmetz
*Multimedia Communications Lab (KOM)*
*Technische Universität Darmstadt*
*Darmstadt, Germany*
*Email: {firstname.lastname}@KOM.tu-darmstadt.de*

Stefan Schulte
*Distributed Systems Group*
*Vienna University of Technology*
*Vienna, Austria*
*Email: s.schulte@infosys.tuwien.ac.at*

***Abstract*—The challenge of optimally selecting services from a set of functionally appropriate ones under Quality of Service (QoS) constraints – the *Service Selection Problem* – has been extensively addressed in the literature based on deterministic parameters. In practice, however, Quality of Service QoS parameters rather follow a stochastic distribution. In the work at hand, we present an integrated approach which addresses the Service Selection Problem for complex workflows in conjunction with stochastic Quality of Service parameters. Accounting for penalty cost which accrue due to Quality of Service violations, our approach reduces the impact of stochastic QoS behavior on total cost significantly.**

***Keywords*-Service Selection, Stochastic Quality of Service, Optimization, Simulation**

## I. Introduction

The selection of services from a set of appropriate ones that are able to provide the required functionality and thereby best meeting cost and Quality of Service (QoS) requirements – the Service Selection Problem (SSP) – is widely recognized in the literature, e.g., [1]–[4]. The optimization of the SSP is thereby based on deterministic QoS values. A solution to the SSP describes an execution plan, i.e., an assignment of services to certain tasks of a workflow, which satisfy the mentioned cost and QoS constraints.

But QoS, e.g., the response time of a service or its availability, is not always deterministic in reality. Due to network latency or server load, response times of services may change dynamically. I.e., when the execution of the computed execution plan actually takes place, the perceived QoS might differ from the expected QoS which has previously been used for the calculation of the execution plan. Thus, although having computed an optimal solution to the SSP during design time which satisfies the constraints, it still is possible that these constraints are violated during runtime.

The work at hand addresses this issue. Based on a service broker scenario, which is presented in Section II, we describe how QoS violations due to stochastic QoS behavior negatively impacts total cost. In order to account for this impact of stochastic QoS parameters, we present an integrated approach comprising an optimization, a simulation, and an adaptation step.

During the optimization step, we compute an optimal solution to the SSP, i.e., an optimal execution plan, satisfying the QoS constraints based on the deterministic QoS values denoted by the respective service providers. In the simulation step, we observe the expected runtime behavior of the computed execution plan in terms of QoS. This way, we can assess potentially occurring constraint violations. We thereby assume that violating QoS constraints is penalized, i.e., penalty fees become due in addition to service invocation cost. According to the results of the simulation, we apply a greedy adaptation heuristic in order to reduce the impact of potentially occurring constraint violations.

The remainder of this work is structured as follows. In Section II, we present a motivating scenario, which will be used throughout the paper. In Section III, we describe our solution to the SSP, which is based on our previous work in [5]. The applied simulation approach is presented in Section IV. Based on the simulation results, we apply our greedy adaptation heuristic which is described in Section V and evaluated in Section VI. Finally, after having distinguished our approach from related work in Section VII, we draw conclusions and discuss future work in Section VIII.

## II. Scenario

In this section, we present a scenario, which is used as an example in the work at hand in order to illustrate the impact of stochastic QoS parameters. The application of our approach is not limited to this scenario.

Imagine a service broker who receives requests from its customers. Paying the broker a fixed amount of money, the customers require certain tasks and workflows, respectively, to be executed. For this, they provide the broker with a document which specifies the required tasks from a functional perspective and indicates the ordering of the tasks, i.e., the structure of the workflow. One of the broker's customers asks for instance for the workflow in Figure 1. The process steps $PS_i$ thereby indicate the tasks which have to be accomplished. Each task can be executed by a single service.

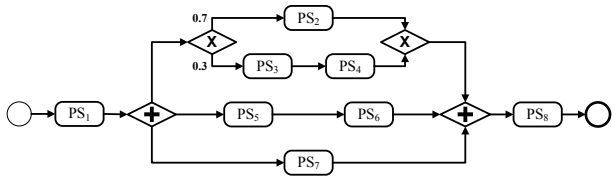



Figure 1: Example Workflow (in BPMN)

In addition to these functional requirements, the customers also specify their QoS needs regarding the execution of the respective workflow. For this, they provide restrictions in the form of upper or lower bounds for specific QoS parameters, the so-called Service Level Objectives (SLOs). With this information, the broker tries to select those services among functionally appropriate ones which satisfy the customers' QoS requirements, as a violation of these requirements will be penalized by the broker's customers. Having selected respective services, the broker pays and invokes these services in order to execute the customers' tasks and workflows, respectively. If the customers' QoS requirements are violated, penalty fees become due, which also have to be paid by the broker. In order to reach an optimized decision, the broker models the selection of services as an optimization problem aiming at minimizing invocation cost and satisfying QoS constraints. I.e., the broker formulates an SSP, which is described in the following Section III.

## III. Service Selection Problem

In order to formulate an SSP and therewith to compute an execution plan, it is necessary to aggregate the QoS and cost values of eligible candidate services according to the regarded workflow structures. For this, we specify a system model in Section III-A and discuss respective aggregation functions in Section III-B. Finally, we utilize the presented aggregation functions and provide an optimization model in Section III-C.

### A. System Model

In this section, we describe the system model utilized in the work at hand. The set of all tasks of a workflow is labeled with $I$, $i \in I = \{1, ..., i^{\#}\}$. The set of services appropriate to accomplish a certain task $i$ is labeled with $J_i$, $j \in J_i = \{1, ..., j_i^{\#}\}$. The decision variables $x_{ij} \in \{0, 1\}$ indicate whether a service $j$ is selected to accomplish task $i$. According to our running example, we consider cost $c$ (charge for invoking a service in cent applying a pay-per-use pricing model), response time $r$ (time elapsed between invoking a service and receiving its response), availability $a$ (probability that a service is available), and throughput $d$ (number of requests a service is able to serve within a certain time interval). These parameters – in fact, even a subset of these parameters – are sufficient to cover the aggregation types summation, multiplication, and min/max-operator (cf. Section III-B). Further QoS parameters can be integrated into the optimization problem straightforwardly. Bounds for these parameters are labeled with $b_c$, $b_r$, $b_a$, $b_d$.

Regarding branchings, we label the set of paths with $L$, in which $l \in L = \{1, ..., l^{\#}\}$ indicates the respective path number. Referring to the workflow in Figure 1, there are two paths $l$ within the AND-block, thus $L = \{1, 2\}$. Different sets of paths will be distinguished by utilizing additional indices, i.e., $L_a$, $L_x$ for AND/XOR. We refer to them as branching $L_a$, $L_x$. The tasks within a branching are covered by the set $I_L \subseteq I$, whereas $I_l \subseteq I_L$ represents the set of tasks within path $l$. We label the set of the remaining tasks, which are not located within a branching, with $I_s = I \setminus (I_l \mid l \in L)$. Utilizing this system model, we develop aggregation functions in the following.

### B. Aggregation Functions

As previously stated, it is necessary to aggregate the QoS and cost values of candidate services according to regarded workflow patterns in order to compute the overall cost and QoS for a specific workflow. Regarding our example scenario, this actually is a prerequisite for comparing workflow QoS with respective bounds issued by the broker's customers. As previously stated, violation of the customers' bounds will result in additional penalty cost. Thus, the broker has a high interest in making sure that the customers' bounds are satisfied. For this, the broker performs a worst-case analysis as opposed to a best-case or average-case analysis. While probabilities for the execution of certain paths of a branching would be considered in an average-case analysis, the worst (best) paths of each branching in terms of aggregated cost and QoS are considered in a worst-case (best-case) analysis for the computation of an optimal solution to the SSP. Respective aggregation functions for sequences, AND-blocks, and XOR-blocks are indicated in Table I.

Table I: Worst-Case Aggregation Functions

| Sequence | AND-block | XOR-block |
|---|---|---|
| $\sum_{i \in I_s} \sum_{j \in J_i} r_{ij} x_{ij}$ | $\max_{l \in L} \sum_{i \in I_l} \sum_{j \in J_i} r_{ij} x_{ij}$ | $\max_{l \in L} \sum_{i \in I_l} \sum_{j \in J_i} r_{ij} x_{ij}$ |
| $\prod_{i \in I_s} \sum_{j \in J_i} a_{ij} x_{ij}$ | $\prod_{l \in L} \prod_{i \in I_l} \sum_{j \in J_i} a_{ij} x_{ij}$ | $\min_{l \in L} \prod_{i \in I_l} \sum_{j \in J_i} a_{ij} x_{ij}$ |
| $\min_{i \in I_s} \sum_{j \in J_i} d_{ij} x_{ij}$ | $\min_{l \in L}(\min_{i \in I_l} \sum_{j \in J_i} d_{ij} x_{ij})$ | $\min_{l \in L}(\min_{i \in I_l} \sum_{j \in J_i} d_{ij} x_{ij})$ |
| $\sum_{i \in I_s} \sum_{j \in J_i} c_{ij} x_{ij}$ | $\sum_{l \in L} \sum_{i \in I_l} \sum_{j \in J_i} c_{ij} x_{ij}$ | $\max_{l \in L} \sum_{i \in I_l} \sum_{j \in J_i} c_{ij} x_{ij}$ |

For a sequence, the cost and QoS values of all services selected to accomplish certain tasks $i$ have to be aggregated according to the respective aggregation type, e.g., summed up for cost. Regarding AND-blocks, it has to be noted that for response time $r$ only the path with the highest aggregated response time – the critical path – requires consideration, as the tasks within the different paths are executed in parallel.

Regarding the other non-functional parameters, the cost and QoS values of all services have to be aggregated as all selected services are executed in the end (similar to a sequence). For XOR-blocks, where only one of the potential paths is executed, we consider the path with the worst aggregated cost and QoS values according the respective aggregation types as we pursue a worst-case analysis.

In the work at hand, we stick to these patterns for the sake of simplicity. The interested reader may refer to our former work in [5] for further structured workflow patterns (OR-blocks, Repeat Loops) as well as for unstructured patterns of Directed Acyclic Graphs and respective aggregation functions, which could have additionally been utilized.

Applying the presented aggregation functions, we formulate the SSP as an optimization problem in the following.

### C. Optimization Model

In this section, we formulate the SSP in Model 1 – accounting for the example workflow in Figure 1. For this, we specify an objective function in (1) and a set of restrictions in (2)–(10) by applying the aggregation functions from Table I.

**Model 1** Service Selection Problem for Example Workflow

Objective Function

$$\text{minimize} \sum_{i\in I_s}\sum_{j\in J_i} c_{ij}x_{ij} + \sum_{l\in L_a}(c'_l + \sum_{i\in I_l}\sum_{j\in J_i} c_{ij}x_{ij}) \quad (1)$$

so that

$$\sum_{i\in I_s}\sum_{j\in J_i} r_{ij}x_{ij} + \max_{l\in L_a}(r'_l + \sum_{i\in I_l}\sum_{j\in J_i} r_{ij}x_{ij}) \leq b_r \quad (2)$$

$$(\prod_{i\in I_s}\sum_{j\in J_i} a_{ij}x_{ij}) \cdot (\prod_{l\in L_a}(a'_l \cdot \prod_{i\in I_l}\sum_{j\in J_i} a_{ij}x_{ij})) \geq b_a \quad (3)$$

$$\min(\min_{i\in I_s}\sum_{j\in J_i} d_{ij}x_{ij}, \min_{l\in L_a}(d'_l, \min_{i\in I_l}\sum_{j\in J_i} d_{ij}x_{ij}))) \geq b_d \quad (4)$$

$$\max_{l_x\in L_x}(\sum_{i\in I_{l_x}}\sum_{j\in J_i} r_{ij}x_{ij}) = r'_l \quad \forall l \in L_a| \text{ interlaced} \quad (5)$$

$$\min_{l_x\in L_x}(\prod_{i\in I_{l_x}}\sum_{j\in J_i} a_{ij}x_{ij}) = a'_l \quad \forall l \in L_a| \text{ interlaced} \quad (6)$$

$$\max_{l_x\in L_x}(\min_{i\in I_{l_x}}(\sum_{j\in J_i} d_{ij}x_{ij})) = d'_l \quad \forall l \in L_a| \text{ interlaced} \quad (7)$$

$$\max_{l_x\in L_x}(\sum_{i\in I_{l_x}}\sum_{j\in J_i} c_{ij}x_{ij}) = c'_l \quad \forall l \in L_a| \text{ interlaced} \quad (8)$$

$$\sum_{j\in J_i} x_{ij} = 1 \qquad \forall i \in I \quad (9)$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in I, \forall j \in J_i \quad (10)$$

Note that the considered workflow in Figure 1 contains an XOR-block within the AND-block. As the aggregation functions indicated in Table I assume the tasks within splits and corresponding joins to be arranged sequentially (cf. [6]), we have to account for this interlaced structure. Referring to our former work in [6], we abstract from the interlacing and insert additional variables $r'_l$, $c'_l$, $a'_l$, $d'_l$ and their respective aggregation functions (i.e., for XOR-blocks in this case) into Model 1. As the broker receives a fixed amount of money from his/her customers for satisfying their needs, (s)he maximizes his/her profit by selecting and invoking the services with minimal cost. Thus, the objective function in (1) aims at minimizing service invocation cost. The customers' restrictions on cost and QoS are indicated in (2)–(7). In (9), we make sure that only one service is selected for each task, and (10) represents the integrality condition for the decision variables.

Model 1 constitutes a non-linear optimization problem as it contains non-linear aggregations of decision variables $x_{ij}$, i.e., multiplication and min/max-operator in (2), (3), (4) for instance. We transform it into a linear one by linearizing the non-linear restrictions in (3)–(7). Due to space restrictions, we omit describing the linearization in the work at hand and refer the interested reader to our former work in [5], [6]. The optimal solution to the obtained linear optimization problem can then be computed by applying Integer Linear Programming (ILP) techniques from the field of Operations Research [7].

Thus, having computed an optimal execution plan minimizing service invocation cost and (presumably) satisfying the customers' constraints, the broker would apply this solution and invoke the respective services while assuming that they actually hold the constraints. But as the invoked services may show a different behavior during runtime than expected during design time, the application of the computed execution plan could lead to violations of the customers' constraints. This would result in additional cost for the broker as, in this case, penalty fees will become due.

In order to assess the impact of stochastic QoS values, we propose to perform an additional simulation step, which is described in the following Section IV.

## IV. Simulation

### A. General Concept

The simulation approach presented in the work at hand is substantially inspired by findings from the domain of project management. In our former work in [8], we outlined the conceptual similarities between workflows and project networks, specifically, *generalized activity networks* [9]: a complex workflow can easily be interpreted as a project network, which consists of comparable entities, such as tasks and branches.

Based on the work in [10], we argued in [8] that simulation provides the best means to assess the risk of breaking

given QoS constraints in practical application. Referring to [10], deterministic methods regularly fail to correctly quantify such risks, specifically if large and complex networks and workflows, respectively, are concerned.

Thus, regarding our example scenario, an optimization approach based on deterministic QoS values does not capture the risk of violating QoS constraints. For instance, if a selected service experiences unusually high demand, it may not be able to provide a certain response time, even if a corresponding bound has been guaranteed by the service provider. Thus, in turn, an execution plan may not satisfy the QoS requirements in some cases. The existence of substantial QoS fluctuations – specifically with respect to Web service response times – has been empirically shown, for instance by Rosario et al. [11] and Miede et al. [12]. In the work at hand, we quantify these inherent risks through simulation, i.e., repeated "virtual" execution of a workflow.

### B. Stochastic QoS Parameters

As previously stated, it is assumed in most related approaches addressing the SSP that a "hard" – *deterministic* – QoS guarantee is specified for each service candidate and QoS parameter. In order to perform a simulation, we further assume that *probabilistic* QoS specifications are available.

In accordance with the notation in Section III-A, $R_{ij}$, $A_{ij}$, and $D_{ij}$ represent *random variables* for the QoS parameters response time, availability, and throughput of service $j$ for task $i$. These random variables may follow arbitrary *probability distributions*. For instance, the response time of a service may be modeled using a normal distribution, i.e., $R_{ij} \sim N(100, 20)$. For an overview of common probability distributions, we refer to corresponding compendiums [13].

Probability distributions can be deduced in at least two principle ways: First, they may be explicitly provided by a service provider as part of contracted guarantees in terms of Service Level Agreements (SLA), following the idea of "soft contracts" [11]. Second, they may be inferred by mining the monitoring data from past service executions.

### C. Execution of the Simulation

For the actual simulation, we virtually execute the previously computed execution plan a predefined number of times. In each iteration, we draw a specific realization of each QoS parameter for each selected service, based on the given random variables $R_{ij}$, $A_{ij}$, and $D_{ij}$. In addition, depending on the branching probabilities of XOR-splits, we draw random variables in order to determine which paths are actually executed in the current iteration. The realized values of the individual services are aggregated afterwards according to the respective workflow structure applying the aggregation functions from Table I. This way, we compute the overall QoS values for the whole workflow.

Based on the workflow in Figure 1, we provide an example in Figure 2. It represents an execution plan where

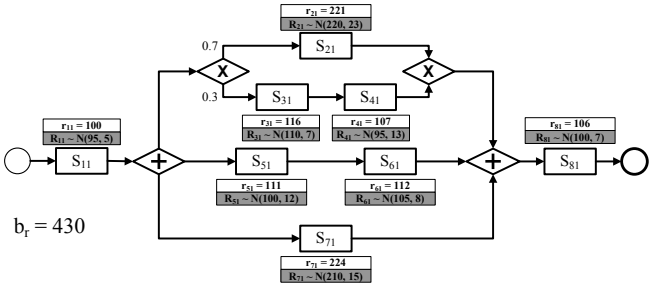



Figure 2: Workflow execution plan with services' QoS guarantees, serving as input to the simulation.

possible, appropriate services have been identified for the realization of all tasks. Each service is associated with deterministic QoS bounds, as specified by respective service providers (boxes with light-gray background). Further, probabilistic QoS specifications – as observed by the broker – are indicated (boxes with dark-gray background). Note that we assume conservative service providers as the deterministic QoS values specified by the service providers are higher than the expected values. For simplicity reasons, we only incorporate the QoS parameter *response time* in this example. As can be quickly validated, the workflow will meet the QoS constraint, namely $b_r = 430$, according to the given deterministic QoS values.

However, performing a simulation with $10,000$ iterations, i.e., drawing $10,000$ corresponding realizations of the regarded QoS parameter response time, reveals that the specified QoS constraint will be violated in approximately $15\%$ of all executions – although we assumed conservative service providers. Figure 3 indicates the respective aggregated response time for the workflow. The QoS violation can be attributed to one or more services exceeding their deterministic QoS guarantees, which cannot be sufficiently captured by the initial service selection and, thus, will result in potentially severe penalities for the broker. In order to reduce the impact of stochastic QoS behavior and therewith the accruing penalties, the boker applies our greedy adaptation heutistic, which is discribed in the following Section V.

## V. Greedy Adaptation Heuristic

In the previous section, we have outlined how a simulation step may reveal the risk of violating certain QoS constraints in practice. This knowledge can be exploited in order to minimize the risk of penalties and therewith the total cost for the broker in our example scenario comprising cost for invoking the selected services and cost according to expected penalty fees. For this, we present a greedy adaptation heuristic in this section aiming at reducing the total cost for the broker.

In this context, adaptation denotes excluding and replacing, respectively, those services from the formerly computed execution plan for other functionally appropriate services,

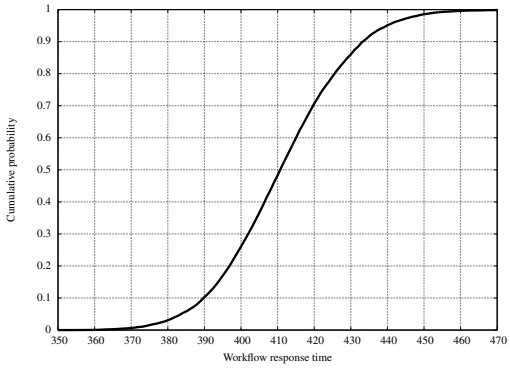



Figure 3: Cumulative probability function of the workflow's response time, according to the simulation.

which cause high penalty cost due to unexpected runtime behavior. One may argue that we could have also performed an adaptation by trying to improve QoS for the services of the computed execution plan, i.e., asking the respective service providers to either enhance resource deployment in terms of, e.g., main storage and CPU, or to (generally) improve the implementation of the service providers' services. As both is out of the broker's control sphere, we do not focus on nor account for such possibilities. We further assume SLAs to be fixed and do not account for SLA negotiations, neither between the broker and its customers, nor between the broker and respective service providers – which could also be seen as an alternative adaptation in order to tackle QoS violations – as (optimal and efficient) SLA negotiation is a research topic on its own and would go beyond the scope of the paper at hand.

Regarding penalty cost, we assume linear penalty fees per *unit* of QoS violation, e.g., cents per second the execution took longer than restricted by the respective bound, or cent per percent point the availability was lower than restricted. It would also be imaginable to assume variable penalty fees, which increase quadratically or exponentially with the size of the violation. As this would only influence the calculation of the actual total penalty cost depending on the chosen penalty model but does not change our approach, we stick to linear penalty fees for the sake of simplicity.

Our greedy adaptation heuristic is indicated in Algorithm 1, using pseudocode. The heuristic is split into four steps. First, we determine the total cost of the current solution. Second, we compute the *critical QoS parameter*, i.e., the QoS parameter which causes the highest penalty cost, as this parameter might bear the highest penalty cost savings. Third, we determine for which of the tasks $i \in I$ a potential improvement of the respective *critical QoS parameter* would be highest. Fourth, we finally perform the adaptation.

More details on these steps are provided in the following. Referring to Algorithm 1, we compute an optimal solution $cs$, which represents the current solution, based on deterministic QoS values, perform the previously mentioned simulation step, and calculate the QoS violation in lines 2 to 4. In order to determine the total cost of the current solution, we aggregate the invocation and accruing penalty cost as indicated in line 5. By computing and comparing the penalty cost for each QoS parameter, we determine *critical QoS parameter* in lines 7 to 13. In order to compute the *critical task*, we compute in line 16 the potential *benefit* for each task $i$. In this context, the potential *benefit* of a task $i$ corresponds to the highest possible reduction in standard deviation $\sigma$. Referring to (11), we compute the difference of the selected service's $\sigma_s$ to the $\sigma_j$ values of the other candidate services of task $i$.

$$benefit = \max_{j \in J_i}(\sigma_s - \sigma_j) \tag{11}$$

The highest possible reduction in $\sigma$, which corresponds to the highest possible reduction in uncertainty and risk regarding stochastic QoS parameters, thereby represents the potential (absolute) benefit for that task. In line 17, we compute the weight $\omega$ for task $i$. For this, we divide the number of the simulation runs where task $i$ has been virtually executed *and* the restriction for the *critical QoS parameter* has been violated by the total number of runs. As indicated in lines 18 to 20, the task with the highest weighted benefit becomes the *critical task*, which is required in order to perform the actual adaptation.

As stated earlier, our adaptation comprises excluding and replacing services while other adaptation techniques and mechanisms also would have been possible and supported by our heuristic. Insofar, our approach is flexible and extensible for supporting further adaptation techniques. We determine the currently selected service $j_s$ in line 24. We ban in lines 25 to 27 all services of the *critical task*, which have a negative benefit, i.e., whose $\sigma$ and therewith the risk of QoS violation is larger than or equal to the $\sigma_{j_s}$ of the currently selected service $j_s$. This way, we adapt the list of available candidate services for the *critical task*.

Utilizing this adapted list, we rerun the optimization and obtain a new execution plan, which is optimal under the adapted circumstances. Afterwards, we conduct the simulation step again and calculate both the invocation and penalty cost for the lastly computed execution plan (cf. the first step). By comparing the total cost of this new solution with the previous – formerly known as current best – solution, we determine whether applying our adaptation heuristic has been advantageous from the broker's point of view. Via a parameter *greed*, we specify the algorithm's degree of greed, i.e., whether and how often the described steps are repeated as long as past iterations reduced total cost. Via a further parameter *anneal*, we control for allowing worse solutions temporarily as starting point for a continuous application of the algorithm. An evaluation for this approach is presented in the following Section VI.

**Algorithm 1** Greedy Adaptation Heuristic

```
1: //First step – determine current cost
2: cs = computeCurrentSolution();
3: sim = simulate(cs);
4: v = computeQoSViolation(sim);
5: totalCost = computeInvCost(cs) + computePanalty(v);
6: //Second step – determine the critical QoS parameter
7: for all p ∈ QoSparameters do
8:     penaltyCost = computePenalty(p);
9:     if penaltyCost ≥ highestPenaltyCost then
10:        highestPenaltyCost = penaltyCost;
11:        criticalQoSparameter = p;
12:    end if
13: end for
14: //Third step – determine the critical task
15: for all i ∈ I do
16:    benefit = computeBenefit(i);
17:    ω = computeWeight(i);
18:    if ω · benefit ≥ highestBenefit then
19:        highestBenefit = ω · benefit;
20:        criticalTask = i;
21:    end if
22: end for
23: //Fourth step – perform the adaptation
24: j_s = getSelectedServiceOf(i);
25: for all j ∈ J_i do
26:    if σ_j ≥ σ_{j_s} then
27:        setBanned(j);
28:    end if
29: end for
```

## VI. Evaluation

As a proof of concept, we implemented our greedy adaptation heuristic. For the computation of an optimal solution to the SSP based on deterministic values, we utilized the linear programming solver CPLEX[1]. In this section, we evaluate the impact of QoS violation on total cost with respect to our broker scenario. For this, we conducted a set of experiments in order to assess the effects of different configurations regarding the adaptation parameters *greed* and *anneal*, which allow to control for the number of iterations the algorithm performs in order to achieve improved solutions and for the number of thereby temporarily accepted worse solutions.

In the following, we describe our experimentation setup. We consider the workflow in Figure 1, which contains $PS_1$-$PS_8$ indicating tasks $i \in \{1, ..., 8\}$. In order to determine QoS values of the respective services, we draw realizations of the random variables $R_{ij}$, $A_{ij}$, and $D_{ij}$. We assume these random variables to follow a normal distribution, i.e., $R_{ij} \sim N(\mu_r, \sigma_r)$, $A_{ij} \sim N(\mu_a, \sigma_a)$, $D_{ij} \sim N(\mu_d, \sigma_d)$.

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

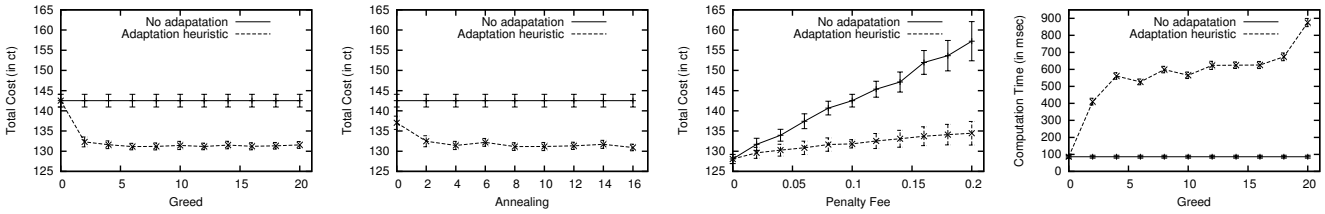
Table II: Stochastic QoS

| **QoS** | $PS_2$, $PS_7$ | $PS_1$, $PS_3 - PS_6$, $PS_8$ |
|---|---|---|
| $R_{ij}$ | $\mu_r \sim U(160, 240)$<br>$\sigma_r \sim U(0, 40)$ | $\mu_r \sim U(80, 120)$<br>$\sigma_r \sim U(0, 20)$ |
| $A_{ij}$ | $\mu_a \sim U(0.92, 1.0)$<br>$\sigma_a \sim U(0.0, 0.08)$ | $\mu_a \sim U(0.94, 0.98)$<br>$\sigma_a \sim U(0.0, 0.04)$ |
| $D_{ij}$ | $\mu_d \sim U(80, 120)$<br>$\sigma_d \sim U(0, 30)$ | $\mu_d \sim U(80, 120)$<br>$\sigma_d \sim U(0, 30)$ |
| $c_{ij}$ | $U(0.8, 1.2) \cdot (40 +$<br>$(0.03 \cdot (\mu_d - \mu_r)) \cdot \mu_a^2)$ | $U(0.8, 1.2) \cdot (20 +$<br>$(0.03 \cdot (\mu_d - \mu_r)) \cdot \mu_a^2)$ |

We could have equally utilized other distribution functions or inferred the respective distributions by mining the monitoring data from past service executions, which we actually envisage in our future work, but we stick to normal distributions in the work at hand for the sake of simplicity. The parameterization of the random variables $R_{ij}$, $A_{ij}$, and $D_{ij}$ is indicated in Table II. We assume that the invocation cost of a service partly depends on its QoS, i.e., *good* QoS values in terms of low response time $r$, high availability $a$, and high throughput $d$ result in higher invocation cost. For this, as also indicated in Table II, we compute the invocation cost of a service according to its QoS, utilizing an additional, uniform distributed random variable $U(a, b)$.

In order to assess the impact of the number of beneficial iteration steps – as indicated by the parameter *greed* – on total cost and computation time, we varied *greed* in Figure 4a and Figure 4d from 0 to 20 step two, utilizing *anneal* $= 4$. Regarding the influence of the number of temporarily allowed worse solutions, we varied the *anneal* parameter in Figure 4b from 0 to 16 step two, utilizing a fixed *greed* value of 10. For these experiments, we set the violation cost to $10\%$ of the respective service invocation cost – per unit of QoS violation (cf. Section V). In Figure 4c, we account for different penalty cost by varying the penalty cost percentage from $0\%$ to $20\%$, utilizing *greed* $= 10$, *anneal* $= 4$. Finaly, in Figure 4d, the computation time for computing respective solutions is indicated. The experiments were performed on an Intel Core 2 Quad processor at 2.66 GHz, 4 GB RAM, running Microsoft Windows 7.

The evaluation results show that the application of our greedy adaptation heuristic to the considered service broker scenario leads to a cost reduction of 9 ct to 12.5 ct in relation to total cost of 142.5 ct for the whole workflow, which corresponds to a reduction of $6\%$ to $8.5\%$. Thus, the broker could save $6\%$ to $8.5\%$ of the total cost. But, Figure 4d reveals that reducing the cost actually "costs" computation time – up to 10 times as much. As Figure 4a indicates, additional reduction in total cost decreases with additional adapatation steps, i.e., higher *greed* values. Therefore, utilizing a *greed* value of 4 to 6, for which the computation time is roughly 6 times higher, seems to be a good compromise between cost reduction and additional computation time. Also values greater than 4 for *anneal* do not improve the cost reduction




(a) Impact of greed (b) Impact of anneal (c) Impact of penalty cost (d) Computation time

Figure 4: Evaluation Results

significantly. Thus, using the parameterization *greed* $= 6$, *anneal* $= 4$, which leads to a cost reduction of $7.3\%$ and a 6-times magnified computation time, appears sensible.

Having described our approach, we discuss related approaches in the following Section VII.

## VII. Related Work

As previously stated, the SSP is widely recognized in the literature. A survey of current approaches can be found in [4]. In principle, current approaches can be divided into two categories: heuristic approaches which try to find rather good solutions within a reduced amount of computation time, e.g., [2], [14], [15], and approaches aiming at finding an optimal solution to the SSP, e.g., [1], [3], [16]. All those approaches presume that the utilized QoS parameters are deterministic. Relevant related work in the area of stochastic QoS parameter, however, is rather sparse.

In their work in [11], Rosario et al. consider probabilistic QoS values, but not for the purpose of services selection. They rather focus on SLA and contract composition, respectively, using *soft probabilistic contracts* as already stated in Section IV-A. The authors in [17] pay insofar attention to stochastic QoS parameters as they try to achieve an accurate prediction of QoS values based on historic data which then is utilized for service selection rather than predefined values guaranteed by service providers. In [18], Hwang et al. utilize Probability Mass Functions (PMFs) for QoS instead of deterministic values. They describe approaches for aggregating the PMFs of single services. The authors thereby utilize a preselected set of services with discrete PMFs, i.e., they do not perform service selection based on PMFs, but rather aim at computing and estimating QoS for service-based workflows. In contrast to this, our approach targets service selection accounting for stochastic QoS parameters. Cardellini et al. consider stochastic QoS parameters for the SSP insofar as they use an $\alpha$-percentile (with $\alpha = 95\%$) for the QoS parameter *response time* [19]. Accordingly, instead of utilizing deterministic *response time* values for the optimization, the authors integrate a restriction demanding the probability of violating the bound for *response time* to be lower or equal to $1-\alpha$, i.e., $1-0.95 = 5\%$. Projected to our broker scenario, this means that the broker can assume satisfying the respective bound with a probability of $95\%$. But Cardellini et al. thereby fail to account for the arising penalty cost due to QoS violations in the remaining $5\%$ of the cases. Our approach, on the other hand, proceeds one step beyond and considers the impact of QoS violations in terms of penalty cost. Depending on the ratio between invocation and penalty cost, it could be beneficial for the broker selecting rather cheap services, which statistically cause QoS violations more often, and paying the penalty cost rather than selecting expensive services with low probabilities of violating QoS constraints. Thus, our approach enables the broker to select those services that bear the lowest cost. In the approach in [20], which probably comes closest to ours, Leitner et al. assume a fixed service composition and a fixed set of possible adaptations to improve the service composition in terms of, e.g., "utilize *Express Shipping*" instead of " utilize *Standard Shipping*". The aim is to select and apply those adaptations that minimize cost comprising invocation cost, penalty cost for QoS violation, and cost for applied adaptations, which aim at avoiding QoS violation. If we abstract from the term *adaptation* and interpret available adaptations as alternative services, then Leitner et al. are solving a SSP with the aim of minimizing total cost, at which penalty cost for QoS violation are considered as well. In order to account for non-deterministic QoS behavior during runtime, the authors utilize a predictor component which predicts prospective QoS values and therewith expected QoS violation. Thus, Leitner et al. estimate the impact of stochastic QoS behavior for each service separately and perform a optimization with these estimated, deterministic QoS values allowing for QoS violations and accounting for their impact on total cost. Our approach, however, goes one step further, as we do not consider the stochastic QoS behavior of the services independently of each other, but account for the whole workflow during our simulation step. Thus, potential reverse QoS deviations of different services from expected behavior can be considered, which is not possible in [20] due to their isolated consideration of expected QoS per service, independently of other services.

In summary, our approach extends related work as it considers, on the one hand, the impact of QoS violation in terms of accruing penalty costs. On the other hand, we do not

only regard isolated stochastic QoS behavior for individual services, but account for probably compensating reverse QoS deviations of different services. Thus, we consider the impact of stochastic QoS behavior for the whole workflow. Conclusions are drawn in the following Section VIII.

## VIII. Conclusion

The SSP is widely recognized in the literature and has been discussed in several scientific papers – based on deterministic QoS parameters. In the work at hand, we addressed the SSP in conjunction with stochastic QoS parameters which has been considered as yet only insufficiently in the literature. For this, we presented an integrated approach comprising an optimization, a simulation, and an adaptation step which aims at reducing the impact of stochastic QoS behavior on total cost. The evaluation shows that the application of our approach leads to a cost reduction up to $8.5\%$, utilizing the described service broker scenario. Thus, the actual, absolute level of cost reduction depends on the concrete paramerization and the regarded scenario. For this, we will extend the evaluation in our future work by considering further workflow structures and QoS distribution functions as well as different degrees of conservative, deterministic QoS values issued by service providers. In addition, we focus on improving the scalability of our greedy adaptation heuristic.

## Acknowledgment

This work is supported in part by the Commission of the European Union within the ADVENTURE FP7-ICT project (Grant agreement no. 285220) and by E-Finance Lab e. V., Frankfurt am Main, Germany (http://www.efinancelab.com).

## References

[1] D. Ardagna and B. Pernici, "Adaptive service composition in flexible processes," *IEEE Transactions on Software Engineering (TSE)*, vol. 33, no. 6, pp. 369–384, 2007.

[2] D. A. Menascé, E. Casalicchio, and V. K. Dubey, "A Heuristic Approach to Optimal Service Selection in Service Oriented Architectures," in *Workshop on Software and Performance (WOSP)*. ACM, 2008, pp. 13–24.

[3] A. F. M. Huang, C.-W. Lan, and S. J. H. Yang, "An Optimal QoS-based Web Service Selection Scheme," *Information Sciences (ISCI)*, vol. 179, no. 19, pp. 3309–3322, 2009.

[4] A. Strunk, "QoS-Aware Service Composition: A Survey," in *European Conf. on Web Services (ECOWS)*. IEEE Computer Society, 2010, pp. 67–74.

[5] D. Schuller, A. Polyvyanyy, L. García-Bañuelos, and S. Schulte, "Optimization of Complex QoS-Aware Service Compositions," in *Int. Conf. on Service Oriented Computing (ICSOC)*. Springer, 2011, pp. 452–466.

[6] D. Schuller, A. Miede, J. Eckert, U. Lampe, A. Papageorgiou, and R. Steinmetz, "QoS-based Optimization of Service Compositions for Complex Workflows," in *Int. Conf. on Service Oriented Computing (ICSOC)*. Springer, 2010, pp. 641–648.

[7] H. Taha, *Operations Research – An Introduction*, 8th ed. Pearson Prentice Hall, London, 2007.

[8] U. Lampe, D. Schuller, J. Eckert, and R. Steinmetz, "Optimizing Service Selection for Probabilistic QoS Attributes," in *Workshop on Architecture, Concepts and Technologies for Service Oriented Computing (ACT4SOC)*. SciTePress, 2010, pp. 42–51.

[9] C. W. Dawson and R. J. Dawson, "Generalised Activity-on-the-Node Networks for Managing Uncertainty in Projects," *Int. Journal of Project Management*, vol. 13, no. 6, pp. 353–362, 1995.

[10] R. J. Schonberger, "Why Projects are 'always' late: a Rationale based on Manual Simulation of a PERT/CPM Network," *Interfaces*, vol. 11, no. 5, pp. 66–70, 1981.

[11] S. Rosario, A. Benveniste, S. Haar, and C. Jard, "Probabilistic QoS and Soft Contracts for Transaction-Based Web Services Orchestrations," *IEEE Transactions on Services Computing (TSC)*, vol. 1, no. 4, pp. 187–200, 2008.

[12] A. Miede, U. Lampe, D. Schuller, J. Eckert, and R. Steinmetz, "Evaluating the QoS Impact of Web Service Anonymity," in *European Conf. on Web Services (ECOWS)*. IEEE Computer Society, 2010, pp. 75–82.

[13] M. McLaughlin, "A Compendium of Common Probability Distributions," http://www.causascientia.org/math_stat/Dists/Compendium.html, 2001.

[14] F. Lécué, "Optimizing QoS-Aware Semantic Web Service Composition," in *Int. Semantic Web Conf. (ISWC)*. Springer, 2009, pp. 375–391.

[15] N. B. Mabrouk, N. Georgantas, and V. Issarny, "A Semantic End-to-End QoS Model for Dynamic Service Oriented Environments," in *Workshop on Principles of Engineering Service-oriented Systems (PESOS)*. IEEE Computer Society, 2009, pp. 34–41.

[16] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-Aware Middleware for Web Services Composition," *IEEE Transactions on Software Engineering (TSE)*, vol. 30, no. 5, pp. 311–327, 2004.

[17] M. Li, J. Huai, and H. Guo, "An Adaptive Web Services Selection Method Based on the QoS Prediction Mechanism," in *Int. Joint Conf. on Web Intelligence and Intelligent Agent Technology (WI-IAT)*. ACM, 2009, pp. 395–402.

[18] S.-Y. Hwang, H. Wang, J. Tang, and J. Srivastava, "A probabilistic approach to modeling and estimating the qos of web-services-based workflows," *Information Sciences (ISCI)*, vol. 177, no. 23, pp. 5484–5503, 2007.

[19] A. Bellucci, V. Cardellini, V. D. Valerio, and S. Iannucci, "A Scalable and Highly Available Brokering Service for SLA-Based Composite Services," in *Int. Conf. on Service Oriented Computing (ICSOC)*. Springer, 2010, pp. 527–541.

[20] P. Leitner, W. Hummer, and S. Dustdar, "Cost-Based Optimization of Service Compositions," *IEEE Transactions on Services Computing (TSC)*, 2011, prePrint.