2012 26th IEEE International Conference on Advanced Information Networking and Applications

Run-time Revenue Maximization for Composite Web Services with Response Time Commitments

Miroslav Živković*, Joost W. Bosman†, Hans van den Berg*, Rob van der Mei†, Hendrik B. Meeuwissen*, Rudesindo Núñez-Queija†

* *TNO, Delft, The Netherlands, e-mail: miroslav.zivkovic@tno.nl*

† *CWI, Amsterdam, The Netherlands, e-mail: joost.bosman@cwi.nl*

Abstract—**We investigate dynamic decision mechanisms for composite web services maximizing the expected revenue for the providers of composite services. A composite web service is represented by a (sequential) workflow, and for each task within this workflow, a number of service alternatives may be available. These alternatives offer the same functionality at different price–quality levels. After executing a sub-service, it is decided which alternative of the next sub-service in the workflow is invoked. The decisions optimizing expected revenue are based on observed response times, costs and response–time characteristics of the alternatives as well as end–to–end response–time objectives and corresponding rewards and penalties. We propose an approach, based on dynamic programming, to determine the optimal, dynamic selection policy. Extensive numerical examples show significant potential gain in expected revenues using the dynamic approach compared to other, non–dynamic approaches.**

Keywords-**service–oriented architecture; response time; service level agreements; revenue maximization; dynamic programming;**

I. Introduction

Composite web services in a service–oriented architecture (SOA) aggregate web services that may be deployed and executed within different administrative domains. The composite web service provider typically runs an orchestrator that invokes the aggregated services according to a pre–defined work-flow. The workflow is based on an unambiguous functionality description of a service ("abstract service"), and several alternatives ("concrete services") may exist that match such a description [1]. With respect to functionality, all concrete services that match the same abstract service are identical.

A lot of attention in the literature has been paid to the problem of QoS–aware optimal service composition of SOA services (see, e.g. [2], [4], [5]). The main problem addressed in these papers is how to select one concrete service per abstract service for a given workflow. This selection is made with the goal to guarantee the QoS of the composite service (as expressed by the respective SLA) while at the same time optimizing certain objectives like cost minimization. Once established, this composition would remain unchanged the entire life–cycle of the composite web service. In reality, SLA violations occur relatively often, leading to losses for providers and customer dissatisfaction. To overcome this issue, it is suggested in [3], [6], [7] that, based on observations of the actually realized performance, re–composition of the service may be triggered. During the re–composition phase, new concrete service(s) may be chosen for the given workflow. Once the re–composition phase is over, the (new) composition is used as long as there are no further SLA violations. In particular, the authors of [3], [6], [7] describe when to trigger such (re–composition) event, and which adaptation actions may be used to improve overall performance. The service re–composition solutions usually take into account the costs of the re–composition as well as the time scale and type of the adaptation involved. Cardellini et al. [8] discuss the solution to dynamically adapt the service composition at runtime. The search for a new solution is triggered by the occurrence of events that make a previously calculated solution void, and the selection is driven by the goal of maximizing a certain utility function. The problem is modeled as an optimization problem that could be efficiently solved on–line.

We investigate full dynamic service selection (i.e. runtime service composition) that maximizes revenues for the composite service provider, while committing to the response–time objective that is part of the agreed end–to–end SLA. In contrast to Cardellini et al. [8], we take into account the option to adapt the service composition during the execution of the workflow, and on a per–request basis. The main research question addressed in this paper is what is the revenue potential of the proposed dynamic service selection compared to non–dynamic, optimal service composition. We use a dynamic programming approach in order to make an optimal service selection at runtime.

To illustrate our dynamic service selection approach, we observe the case of sequential workflows. Fig. 1 depicts a sequential workflow consisting of four abstract services, and each abstract service maps to a number of concrete services (alternatives). The QoS parameters (e.g. service response–time) of concrete web services are modeled by probability distributions. When the client's request meets the agreed end–to–end deadline, the composite service provider

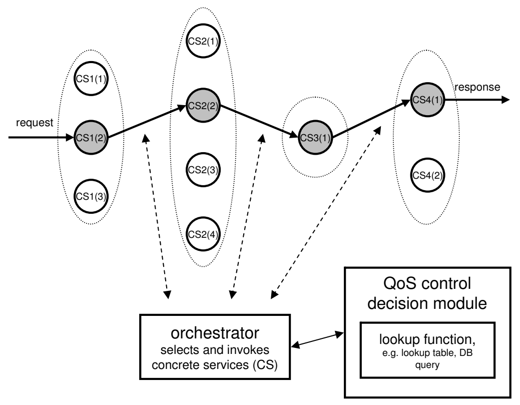

Figure 1. Composite web service depicted by a sequential workflow. Dynamic service composition is based on pre–calculated response–time thresholds using dynamic programming.

is rewarded by the client. Otherwise, when the deadline is not met (i.e. an event of an SLA violation) the provider pays a penalty to the client. After the execution of a single task within the workflow, the orchestrator decides on the next concrete service to be executed. The decision taken is based upon comparison among (1) the remaining time to meet the end–to–end deadline and (2) the pre–calculated response–time thresholds for each service. The response–time thresholds are calculated before the first request is admitted to the system, where the dynamic programming takes the penalty, the reward, the concrete services' level objectives (SLOs) and the execution costs as input. These thresholds are represented by e.g. a lookup table. Depending upon the actual response–times and the thresholds, it is possible that every client request may be served by a different chain of alternatives.

We summarize the main contributions of this paper as follows:

1) Modeling dynamic service composition as a dynamic programming problem. It is shown that this approach yields an optimal service selection policy which can be expressed in terms of a response–time thresholds that are calculated before the service is deployed. At runtime, service selection decisions are not computationally intensive, enabling fast service selection. In practice, service selection can be implemented as a simple lookup function, e.g. lookup table or database query.
2) Numerical investigation of our full dynamic approach as well as a comparison to the optimal a–priori static selection, which shows significant potential gain in expected revenue.
3) Analysis of the impact of the number of available alternatives (and their QoS properties) for each abstract service ("stage") within the sequential workflow. These results show that the expected revenue increases as the number of alternatives increases, especially towards the end of the workflow. In fact, selecting faster but more expensive alternatives occasionally helps to optimize profit.

In Section II, we describe the model of the system under consideration and in Section III we describe the dynamic programming approach we abridged for our purposes. We explain the procedure and derive formulas used to determine response–time thresholds. Simulation results are presented and discussed in Section IV. We illustrate the influence of different system parameters, e.g. the number of concrete services that match an abstract service, etc. Also, we quantify the significant revenue increase when our approach is used. We conclude the paper in Section V, with directions for further research.

II. Sequential workflow decision model

In this section we describe our model based on a composite web service represented by a sequential workflow.

Remark (non sequential workflows)
Our solution is applicable to any workflow that could be aggregated and mapped into a sequential one. We show in Appendix A how to aggregate an example workflow into a sequential one. The aggregation is illustrated for some of basic workflow patterns in case of stochastic (probabilistic) models of services' QoS parameters. The end result is a sequential workflow pattern, in which each service (aggregated or not) has a number of alternatives. However, the aggregation leads to coarser control, since decisions could not be taken for a single service within the aggregated workflow, but rather for the aggregated workflow patterns themselves.

Per single composite service request, the orchestrator executes services one–by–one as indicated by the workflow. There are in total N abstract services in the workflow and the abstract service sequential position is indexed by i, $i = 1, 2, \ldots, N$. Each abstract service i maps onto $M_i \geq 1$ concrete services (alternatives). Before service $i \leq N$ is executed, the orchestrator makes a decision which one of the M_i service alternatives to choose. The decision is made based upon e.g. a lookup table that contains pre–calculated response–time decision thresholds. These thresholds are compared to the time that remains for the workflow execution. The detailed explanation of the procedure and an example of a lookup table are given in Section III.

Once calculated, the decision thresholds are not likely to change as long as agreed SLAs are "static", i.e. time–invariant. The composite service response–time SLO is specified as a "hard" objective, i.e. as a guarantee that all requests would have a response time smaller than a certain deadline. The response–time SLOs of the concrete services are specified as "soft" ones, and in general, "soft" SLOs are expressed as a response–time probability distribution function (PDF) [13], or alternatively, as the cumulative distribution function (CDF).

The response time of alternative service j for abstract service i is denoted as $D_{i,j} \geq 0$, where $i = 1, 2, \dots, N,\ j = 1, \dots, M_i$. From the perspective of the response–time, we model each concrete service as a black box, which means that $D_{i,j}$ is a random variable for which respective PDF (or CDF) is given. The PDFs and CDFs for concrete services are denoted by $f_{i,j}(t)$ and $F_{i,j}(t)$, respectively. From the given PDF $f_{i,j}(t)$, it is easy to determine the expected value $\mu_{i,j}$, variance $\sigma^2_{i,j}$, etc. The realization of the response time $d_{i,j}$ $(i = 1, 2, \dots, N,\ j = 1, \dots, M_i)$ is a single value drawn from the respective PDF. The response times of concrete service alternatives are mutually independent. The realization of end–to–end response time D is represented by d, and $p_{e2e} := \mathbb{P}\{D \leq \delta_p\}$ represents the probability that D is smaller than the given end–to–end deadline δ_p that composite service provider agreed to the clients.

Remark (probability density functions)
In practice, probability density functions may either be estimated from the measurements carried out by the composite service provider, or the third-party domains may publish, or otherwise make available, such information. Since we investigate the potential of our approach, we assume time-invariant SLAs and the PDFs (which are part of the SLAs) do not change either. In case of time–variant PDFs, a recalculation of the response–time thresholds may be occasionally necessary. The recalculation may be triggered when there is a long–term SLA violation, i.e. when the observed PDF differs "significantly" from the initial one. This is beyond the scope of this paper.

The SLA between the individual service provider (ISP) for the j-th concrete service of the i-th abstract service and the composite service provider (CSP) has the following elements:

- The response-time probability distribution function, $f_{i,j}(t)$.
- The cost $c_{i,j}$ [money unit] that the CSP pays to ISP for the execution of a single request. No penalties are imposed. From the ISP viewpoint, this value represents reward.

The SLA between the CSP and its clients has the following elements:

- The end–to–end response time penalty deadline δ_p [time unit].
- The reward R [money unit] that the CSP gets for executing a single request within penalty deadline δ_p.
- The penalty V [money unit] that the CSP pays to the end customer when the agreed end–to–end deadline is not met.

III. Algorithm description

In this section we describe how to optimize expected CSP revenue by formulating the dynamic service selection as a dynamic programming (DP) problem, [14]. Dynamic refers to the fact that before execution of each abstract service in the workflow, depending on the remaining time until deadline δ_p is violated, a concrete service alternative must be selected. The DP optimizes CSP revenue by taking in account the effect of future (optimal) decisions. Since the decisions within DP take in account effects of subsequent decisions, a "backward recursion" method is needed for finding the optimal solution. The application of DP will result in a decision policy. This policy indicates for given response time realizations, which concrete service alternatives should be chosen in order to optimize the CSP's expected revenue per composite service request. The policy is determined by the current position within the sequential workflow i and the remaining time Δ ("time budget") till the overall deadline δ_p will be violated.

The recursion for a set of expected rewards $\mathbb{E}[R_i \mid \Delta = \delta^*]$ is given by:

$$\mathbb{E}[R_i \mid \Delta = \delta^*] = \max_j \Big\{ -c_{i,j} + \mathbb{E}[R_{i,j} \mid \Delta = \delta^*] - \\ -\mathbb{E}[V_{i,j} \mid \Delta = \delta^*] \Big\}, \tag{1}$$

where $i = 1, \dots, N$, $j = 1, \dots, M_i$.
For $j = 1, \dots, M_i$, we have

$$\mathbb{E}[R_{i,j} \mid \Delta = \delta^*] = \begin{cases} \mathbb{P}(D_{N,j} \leq \delta^*)R, \quad i = N, \\ \int\limits_0^{\delta^*} f_{i,j}(t)\mathbb{E}[R_{i+1} \mid \Delta = \delta^* - t]\mathrm{d}t, \\ \qquad\qquad \text{for } i = 1, \dots, N-1. \end{cases} \tag{2}$$

$$\mathbb{E}[V_{i,j} \mid \Delta = \delta^*] = \begin{cases} \mathbb{P}(D_{N,j} > \delta^*)V, \quad i = N, \\ \mathbb{P}(D_{i,j} > \delta^*)\mathbb{E}[R_{i+1} \mid \Delta = 0], \\ \qquad\qquad \text{for } i = 1, \dots, N-1. \end{cases} \tag{3}$$

Here $f_{i,j}(t)$ represents the response time PDF of concrete service alternative j for abstract service i, while the term $\mathbb{E}[R_{i,j} \mid \Delta = \delta^*]$ represents the expected reward, when concrete service j (corresponding to abstract service i) is executed for the given time budget value δ^*. The term $\mathbb{E}[V_{i,j} \mid \Delta = \delta^*]$ represents the expected penalty for exceeding the overall deadline at abstract service i while executing concrete service j for the given time budget value δ^*. The expected reward and penalty functions take into account the impact of future decisions as represented by terms relating to R_{i+1} in equations (2) and (3). Once the end of the workflow is reached ($i = N$), we stop.

The integrals in equation (2) can become rather complicated and will generally not result in tractable expressions. By discretizing the distributions the problem can be solved numerically. For the discretization we split the time interval over which the response–time PDF is defined in segments of the same size h. The number of segments is m^* and the size of h corresponds to the accuracy of the discretization. The discretized versions of the PDF ($p_{i,j,k}$) and CDF ($P_{i,j,k}$) are therefore defined as the following:

$$\begin{aligned} m^* &= \left\lceil \frac{\delta^*}{h} \right\rceil, \\ p_{i,j,k} &= \mathbb{P}\big(D_{i,j} \le h[k+0.5]\big) - \mathbb{P}\big(D_{i,j} \le h[k-0.5]\big), \\ P_{i,j,k} &= \sum_{l=0}^{k} p_{i,j,l}, \\ \text{where } i &= 1,\dots,N,\ j = 1,\dots,M_i,\ k = 0,\dots,m^*. \end{aligned}$$

The larger the number of segments m^* the more accurate the discretization would be, but it would take more time to calculate the respective PDF and/or CDF.

Using the discretization, the backward recursion can be transformed into a scheme that can be evaluated numerically. For given number of segments m^*, let terms R^*_{i,m^*}, R^*_{i,j,m^*}, and V^*_{i,j,m^*} represent discretized versions of $\mathbb{E}[R_i \mid \Delta = \delta^*]$, $\mathbb{E}[R_{i,j} \mid \Delta = \delta^*]$, and $\mathbb{E}[V_{i,j} \mid \Delta = \delta^*]$, respectively. The backward recursion formulas are then as follows:

$$R^*_{i,m^*} = \max_j \Big\{ -c_{i,j} + R^*_{i,j,m^*} - V^*_{i,j,m^*} \Big\}, \tag{4}$$

where $i = 1,\dots,N,\ j = 1,\dots,M_i$,

$$R^*_{i,j,m^*} = \begin{cases} P_{N,j,m^*} R, & i = N, \\ \sum_{k=0}^{m^*} p_{i,j,k} R^*_{i+1,m^*-k}, & i = 1,\dots,N-1, \end{cases} \tag{5}$$

and

$$V^*_{i,j,m^*} = \begin{cases} (1 - P_{N,j,m^*})\, V, & i = N, \\ (1 - P_{i,j,m^*})\, R^*_{i+1,0}, & i = 1,\dots,N-1, \end{cases} \tag{6}$$

where $j = 1,\dots,M_i,\ k = 0,\dots,m^*$.

While applying formulas (4)–(6), the corresponding decisions (actions) A^* can be obtained by storing the maximum arguments evaluated as

$$A^*_{i,m^*} = \operatorname*{argmax}_{j=1,\dots,M_i} \Big\{ -c_{i,j} + R^*_{i,j,m^*} - V^*_{i,j,m^*} \Big\},\ i = 1,\dots,N.$$

The optimal decisions could be represented by a lookup table, and a graphical example of a lookup table for the sequential workflow with $N = 4$ tasks is shown in Figure 2. The horizontal axis corresponds to the time budget left until the overall deadline is breached, while the vertical axis corresponds to the position of the abstract service within the chain. The color corresponds to the decision that has to be taken, e.g. proceed with the alternative j. We illustrate the lookup table with the following examples:

1) We start handling a new request and the overall deadline equals $\delta_p = 13.5$. The decision is marked by an asterisk $*$ at the lookup table shown in Figure 2, and alternative 1 is to be selected.
2) We have 4.9 time units remaining the decision is made for the abstract service at position 3. The decision is marked by a cross $\times$, and points that concrete service alternative 3 should be selected.

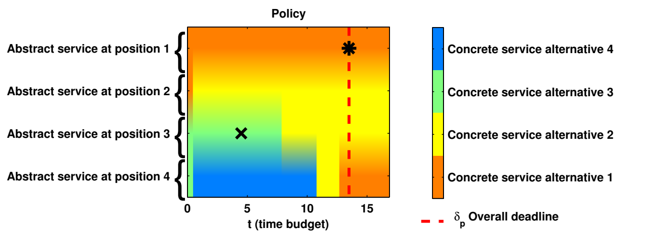

Figure 2. Graphical example of a lookup table. The vertical dashed line represents the overall deadline δ_p.
$*$: Decision at start of request (abstract service 1) for $\delta_p = 13.5$,
$\times$: Decision for remaining time budget $\delta^* = 4.9$ at abstract service 3.

IV. Experiments

A. Setup description

The workflow considered is sequential and there are M_i concrete services that match abstract service i. There are in total $K = \prod_{i=1}^{N} M_i$ possible concrete service composition "paths". In order to evaluate the performance of the dynamic service selection approach, we compare it to the strategy we call "fixed path" (FP). The FP strategy selects the "static" optimal path, i.e. path with the highest expected revenue by exhaustive search for all K possible paths.

Our model leaves a lot of freedom for the selection of response–time distributions and parameter values. These are summarized by Table I. We have selected log–normal response–time distributions, as these distributions are defined only for positive values of response–time, and their parameters can be easily tuned to obtain any desired (finite)

Table I
Definition of parameters used in the experiments

Parameter	Definition	Value
i	Abstract service index	$1,\dots,N$
j	Concrete service alternative index	$1,\dots,M_i$
$F_{i,j}$	Response–time distribution	lognormal$(\mu_{i,j}, \sigma^2_{i,j})$
$\mu_{i,j}$	Response time mean	scenario dependent
$\sigma^2_{i,j}$	Response time variance	scenario dependent
$c_{i,j}$	Service cost	scenario dependent
δ_p	End–to–end deadline	scenario dependent
p_{e2e}	Fraction of required successful responses within the overall deadline $p_{e2e} := \mathbb{P}(D_{FP} < \delta_p)$	0.9
R	Reward per successful request within deadline δ_p	scenario dependent
V	Penalty per request not completed within deadline	scenario dependent
C_{VR}	Penalty/reward ratio	8
$\mathbb{E}[R_{FP}]$	Expected optimal FP revenue	0.01

Table II
CONCRETE SERVICE ALTERNATIVES SYMMETRIC SCENARIO

Abstract service →	Service 1, Service 2, Service 3, Service 4		
Concrete service↓	c	μ	σ
Alternative 1	1	5	2
Alternative 2	5	2.5	2
Alternative 3	10	1.25	4
Alternative 4	50	0.5	0.03

Table III
CONCRETE SERVICE ALTERNATIVES ASYMMETRIC SCENARIO

Abstract service →	Service 1			Service 2		
Concrete service↓	c	μ	σ	c	μ	σ
Alternative 1	1	5	2	5	10	4
Alternative 2	5	2.5	2	1	9.5	8
Alternative 3	10	1.25	4	10	1.5	0.5
Alternative 4	50	0.5	0.03	50	1	0.05

Abstract service →	Service 3			Service 4		
Concrete service↓	c	μ	σ	c	μ	σ
Alternative 1	1	0.5	0.2	1	2.5	1
Alternative 2	5	0.4	0.2	5	2.45	1.5
Alternative 3	10	0.3	0.05	10	1	2
Alternative 4	100	0.05	0	50	0.25	0.02

mean μ and variance σ^2. The selection of distribution is not restrictive, as our solution approach works well for any probability distribution.

We consider a service workflow illustrated at Fig. 3 for our experiments. This sequential workflow consists of $N = 4$ abstract services. For each abstract service i, there are M_i concrete service alternatives, where M_i could take one of the values $\{1, 2, 3, 4\}$, and $M_i \neq M_j$ whenever $i \neq j$. The notation (M_1, M_2, M_3, M_4) depicts the particular experiment setup, and represents one of the possible permutations of the set $\{1, 2, 3, 4\}$. Therefore, there are in total 24 different compositions (experimental setups) to be considered.

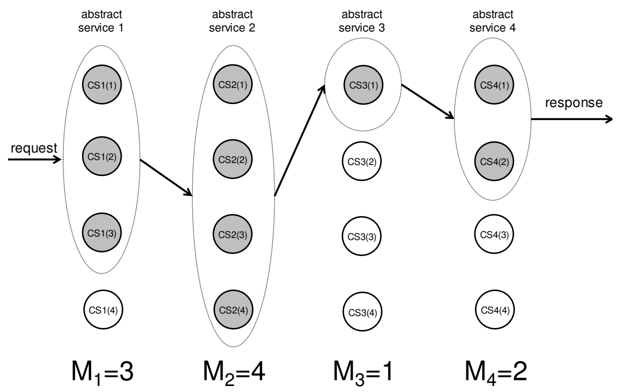

Figure 3. Example workflow where number of alternative concrete services are represented by permutation $(M_1, M_2, M_3, M_4) = (3, 4, 1, 2)$.

We have conducted simulations for two general scenarios, symmetric and asymmetric. These scenarios are defined based on the selection of cost parameters, as well as μ and σ^2 for a concrete service. The goal of symmetric scenario simulations is to illustrate the importance of the position of, and number of, service alternatives within the workflow. The choice of parameters for symmetric scenario is given in Table II; here the parameters of the concrete service alternatives are the same for all abstract services. When the number of alternatives for abstract service is e.g. 2, we always consider alternative 1 and alternative 2 in our experiments.

In the asymmetric scenario which corresponds to parameter Table III, the concrete services have different mean response times for abstract services at different positions in the service composition chain. For example at position 2, alternative $j = 1$ has cost $c_{2,1} = 5$, mean $\mu_{2,1} = 10$ and variance $\sigma^2_{2,1} = 16$ while alternative $j = 2$ is cheaper, i.e. $c_{2,2} = 1$, has a lower mean $\mu_{2,2} = 9.5$ but higher variance $\sigma^2_{2,2} = 64$. Furthermore at position 3, an expensive service with zero variance is added. The goal of the asymmetric scenario is to illustrate the importance of the variance in addition to mean and cost for the service selection, which is usually neglected in state of the art service composition solutions.

Let (j_1, j_2, j_3, j_4) represent the service composition for the FP strategy, where $1 \leq j_k \leq M_k,\ k = 1, 2, 3, 4$. We want to compare the FP and DP strategies when the penalty-to-reward ratio $C_{VR} = \frac{V}{R}$ is set to 8. The expected reward per request for the FP strategy $\mathbb{E}[R_{FP}]$ is given by

$$\mathbb{E}[R_{FP}] = p_{e2e} \cdot R - (1 - p_{e2e}) \cdot V - \sum_{k=1}^{4} c_{k,j_k}. \qquad (7)$$

We use the FP strategy for the benchmarking, so we set the value $\mathbb{E}[R_{FP}] = 0.01$. Taking into account (7) the values for parameters R and V satisfy the following equations:

$$\begin{aligned} R &= \frac{\mathbb{E}[R_{FP}] + \sum_{k=1}^{4} c_{k,j_k}}{(C_{VR} + 1)p_{e2e} - C_{VR}}, \\ V &= C_{VR}R. \end{aligned} \qquad (8)$$

The calculated values for reward and penalty parameters (and given $p_{e2e} = 0.9$) are then used for the simulations of the DP strategy, for both symmetric and asymmetric scenarios.

B. Results

For all possible permutations of number of concrete service alternatives (see Table IV) the expected revenue $\mathbb{E}[R]$ is calculated for both symmetric and asymmetric scenarios and DP and FP algorithms. The results are summarized in Figure 4 (symmetric scenario) and Figure 5 (asymmetric scenario). For both figures, the alternative count configurations are ranked on expected revenue for the DP algorithm.

In Figure 4 we observe that the DP solution achieves the lowest revenue for the permutation $(M_1, M_2, M_3, M_4) = (4, 3, 2, 1)$. In this case there are many alternatives at the first

Table IV
INDICES OF ALTERNATIVE CONFIGURATIONS

label	a	b	c	d	e	f	g	h	i	j	k	l
position 1	4	3	4	2	3	2	4	3	4	1	3	1
position 2	3	4	2	4	2	3	3	4	1	4	1	3
position 3	2	2	3	3	4	4	1	1	3	3	4	4
position 4	1	1	1	1	1	1	2	2	2	2	2	2
label	**m**	**n**	**o**	**p**	**q**	**r**	**s**	**t**	**u**	**v**	**w**	**x**
position 1	4	2	4	1	2	1	3	2	3	1	2	1
position 2	2	4	1	4	1	2	2	3	1	3	1	2
position 3	1	1	2	2	4	4	1	1	2	2	3	3
position 4	3	3	3	3	3	3	4	4	4	4	4	4

position and no alternatives at the end of the workflow resulting in no possibility to recover from (possible) large service response time accumulated at the begin of the workflow. Further, DP–based solution makes always the same choice for the first service in the workflow, not taking any advantage of available alternatives for this service. On the other hand, the highest revenue for the DP solution is achieved for the permutation $(M_1, M_2, M_3, M_4) = (1, 2, 3, 4)$. The most alternatives are then available at the last service within the workflow, thus increasing the possibility to recover from (possible) large response times accumulated at the beginning of the workflow.

We also see from Figure 4 that six configurations with the highest revenue are those where the number of alternatives for abstract service 4 is the highest, i.e. 4. Configurations w, x perform better than configurations s, t, u, v as the number of alternatives for abstract service 3 for the former configurations is 3, and for the latter configurations is either 1 or 2. Additionally, the revenue "jump" between configurations r and t is due to the number of alternatives (3 and 4, respectively) for abstract service 4. The largest DP revenue improvement is when for the abstract service $i = 4$, concrete service alternative 4 is considered (see Table II). When this alternative with low mean and variance is considered, enough certainty to proceed with the workflow execution exists and the high price of concrete alternative 4 will be compensated by the increase in expected revenue. Another (smaller) revenue increase can be observed when concrete service alternative 3 becomes available for abstract service at position 3. This can be explained by the fact that the 90–th percentile for alternative 3 is still lower than 1 and 2 (see Table V) despite its higher variance.

Table V
CONCRETE SERVICE ALTERNATIVE 90TH PERCENTILE.

Alternative	1	2	3	4
Percentile	7.61	4.81	2.74	0.54
μ	5	2.5	1.25	0.5
σ	2	2	4	0.03

In Figure 5 the results are given for the asymmetric scenario. For the asymmetric scenario it is harder to observe any

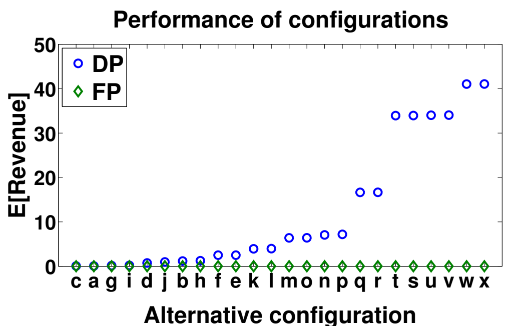

Figure 4. Expected revenues per request in case of symmetric scenario; comparison between FP and DP for different configurations.

structure, because the different alternatives have different impact on the response time and the DP takes advantage of the properties of all available concrete service alternatives. For alternative configurations $(M_1, M_2, M_3, M_4) =$

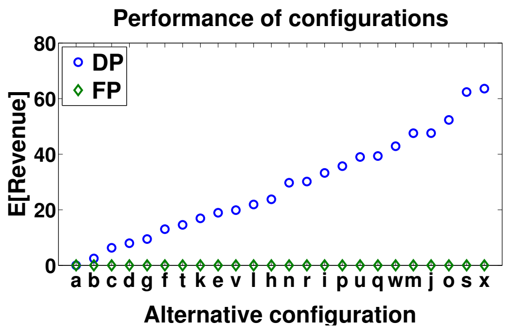

Figure 5. Expected revenues per request in case of asymmetric scenario; comparison between FP and DP for different configurations.

$(4, 3, 2, 1)$ and $(M_1, M_2, M_3, M_4) = (1, 2, 3, 4)$ the lowest expected and the highest revenue are achieved, respectively. The largest revenue increase is achieved when the near–zero variance service alternative is considered at position 4 and when the cheaper second alternative at position 2 is included despite its higher variance.

In Figure 6 we show the comparison of the rewards for the symmetric and asymmetric scenarios. The comparison is done when the same service configuration is used for both scenarios. The indices of service configuration are shown in Table IV.

The main conclusions and some "rules of a thumb" that could be drawn from the Figures 4 - 6 are as the following:

- Dynamic, on–the–fly service composition results in higher revenues for the CSP, compared to optimal "static" service composition (Figures 4, 5). While the expected reward per request for the FP strategy is 0.01, for the symmetric and asymmetric DP scenarios, the expected rewards per request, depending of the configuration, may be greater than 40 or greater than 60, respectively.

- It is more beneficial to have higher number of concrete service alternatives closer to the end of the sequential workflow, (Figures 4, 5).
- The variability of the response–times may have significant impact to the revenues achieved. When the end–to–end deadline is in jeopardy, it may be better to have more expensive service with small response–time variability (and smaller mean) than less expensive one with large response–time variability (Figure 5).
- It is in general better to have more response–time versatility with respect to mean and variance, (Figure 6). However, this needs to be investigated further.

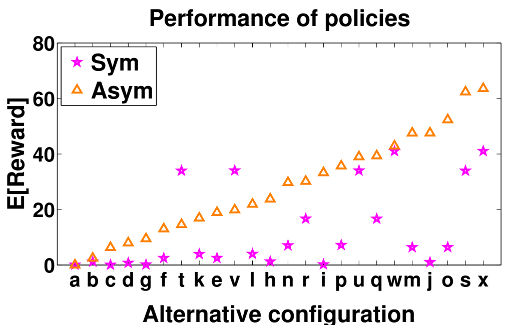

Figure 6. Revenue comparison between symmetric and asymmetric scenarios.

V. Conclusions

In this paper, we investigated the problem of revenue maximization for runtime service composition with end–to–end response–time commitments. We used the dynamic programming approach for dynamic, on–the–fly service selection during the execution of the composite service workflow. The (selection) decisions to be executed are calculated during the service composition, and may be implemented as a lookup table.

A significant amount of extra revenues is generated by our dynamic programming approach compared to the static optimal service composition. These revenues are mainly due to the following scenarios:

- When services at the beginning of the chain are performing better than expected it may be possible to select one of the cheaper but tardy services as the workflow is further traversed, therefore reducing the cost of the invocation.
- When services at the beginning of the chain are performing worse than expected it may be possible to select one of the more expensive but fast services as the workflow is further traversed. Such decisions make end–to–end deadline achievable, however at higher execution costs. Therefore the dynamic service selection results in revenue for CSP in such scenarios. Without the dynamic service selection in place, CSP would have to pay the penalties.

In principle, the number of concrete services considered for service composition becomes more significant as the workflow is traversed. Further, the variability of the response–times has significant impact to the revenues as well.

We have considered the case where SLAs are time–invariant, i.e. once established, response–time SLOs (represented by respective probability density functions) do not change. The possible extensions of this paper would discuss the practical scenarios when such objectives (i.e. PDFs) deviate from values indicated by the SLAs. The re–calculation of the lookup table in optimal way (w.r.t. scalability, frequency of updates) may be necessary in such a case. Further, the dynamic programming solution may tend to slow down when number of services is extremely large. This suggests the development of fast yet efficient heuristic solutions, and opens up a challenging area for further research.

Appendix A.
An example of the workflow aggregation for probabilistic SLAs

We illustrate some basic aggregation rules using an example workflow (represented by Fig. 7) and show how it can be mapped into the (relatively simple) sequential workflow. The proposed aggregation could be done *as long as there is no data dependence among the services*. The aggregation rules are given for the case when probabilistic (i.e. stochastic) QoS models are used for different web service QoS parameters, e.g. response–time. The calculation of the composite service QoS has been analyzed in many papers, e.g. [9]–[13]. However, none of these papers considered stochastic QoS models, except [13], in which the authors calculate the composite service QoS parameters using Monte–Carlo simulations.

The workflow in Figure 7 consists of four elementary, but frequently used workflow composition patterns, namely sequential, flow, switch and loop. There are two alternatives for services 1, 2 and 5, three alternatives for service 3, and services 4 and 6 have one alternative. The flow pattern represents (part of a) workflow in which all services are executed in parallel, and the response is not available till all services finish their execution (services 2 and 3 at Figure 7). The switch statement represents workflow that executes one of the services with given probabilities. Referring back to Figure 7 we identify the switch pattern for services 4 and 5, which are executed with probabilities p_4 and p_5, respectively, where $p_4 + p_5 = 1$. In general, the loop control statement consists of K consecutive invocations of the single service (service 6 in Figure 7). Every service within the workflow is represented by an probabilistic response–time SLO, i.e. the response–time probability density function (PDF) and/or response–time cumulative distribution function (CDF). For concrete service j of abstract service i within the given workflow, the PDF and CDF are $f_{i,j}$ and $F_{i,j}$, respectively.

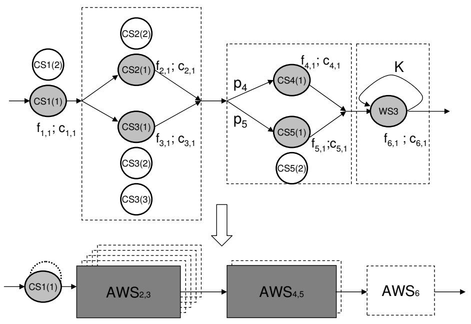

Figure 7. Example workflow reduced and aggregated into the sequential workflow.

The execution cost in this case is $c_{i,j}$, and response–time random variable is $D_{i,j}$. The aggregation rules for concrete services depicted in Figure 7 are as follows:

- The resulting response–time PDF for the flow pattern is expressed by $f_{AWS_{2,3}} = F_{2,1} \cdot f_{3,1} + f_{3,1} \cdot F_{2,1}$. The execution cost is given as $c_{AWS_{2,3}} = c_{2,1} + c_{3,1}$.
- The resulting response–time PDF for the switch pattern (services 4 and 5) is given as $f_{AWS_{4,5}} = p_4 \cdot f_{4,1} + p_5 \cdot f_{5,1}$. The execution cost is $c_{AWS_{4,5}} = p_4 \cdot c_{4,1} + p_5 \cdot c_{5,1}$.
- The resulting response–time PDF of the loop pattern is expressed as K-fold convolution of the PDF $f_{6,1}$, i.e. $f_{AWS_6} = f_{6,1} * f_{6,1} * \cdots * f_{6,1} = f_{6,1}^{*K}$, where $*$ represents convolution operator. The CDF F_{AWS_6} is calculated similarly, and the execution cost is $K \cdot c_{6,1}$.

In case of aggregation of the services with multiple alternatives, the aggregation should take place for each combination of the alternatives for considered services. Therefore, aggregation of services 2 and 3, AWS$_{2,3}$ has 6 alternatives, AWS$_{4,5}$ has 2 alternatives, and so on. The response–time PDFs for the aggregation of more complex workflow patterns could be efficiently numerically calculated using the methods described in [15].

Acknowledgment

This work has been carried out in the context of the IOP GenCom project Service Optimization and Quality (SeQual), which is supported by the Dutch Ministry of Economic Affairs, Agriculture and Innovation via its agency Agentschap NL.

References

[1] C. Preist, “A conceptual architecture for semantic web services,” in *Proc. International Semantic Web Conference (ISWC 2004)*, 2004.

[2] G. Canfora, M. Di Penta, R. Esposito, and M. L. Villani, “An approach for QoS-aware service composition based on genetic algorithms,” in *GECCO ’05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, 2005, pp. 1069–1075.

[3] P. Leitner, “Ensuring Cost-Optimal SLA Conformance for Composite Service Providers,”, in *Proceedings of the PhD Symposium of the International Conference on Service-Oriented Computing 2009 (ICSOC 2009).*

[4] T. Yu, Y. Zhang, and K.-J. Lin, “Efficient algorithms for web services selection with end-to-end QoS constraints,” *ACM Trans. Web*, no. 1, May 2007.

[5] 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*, vol. 30, no. 5, pp. 311–327, 2004.

[6] G. Canfora, M. Dipenta, R. Esposito, and M. Villani, “A framework for QoS-aware binding and re-binding of composite web services,” *Journal of Systems and Software*, vol. 81, no. 10, pp. 1754–1769, Oct. 2008.

[7] L. Zeng, C. Lingenfelder, H. Lei, and H. Chang, “Event-driven quality of service prediction,” in *Proceedings of the 6th International Conference on Service-Oriented Computing*, ser. ICSOC ’08, 2008, pp. 147–161.

[8] V. Cardellini, E. Casalicchio, V. Grassi, and F. L. Presti, “Adaptive management of composite services under percentile-based service level agreements,” in *Proceedings of International Conference on Service-Oriented Computing 2010 (ICSOC 2010)*, 2010, pp. 381–395.

[9] J. Cardoso, J. Miller, A. Sheth, and J. Arnold, “Quality of service for workflows and web service processes,” *Journal of Web Semantics*, vol. 1, pp. 281–308, 2004.

[10] M. C. Jaeger, G. Rojec-Goldmann, and G. Mühl, “QoS aggregation for web service composition using workflow patterns,” in *Proceedings. Eighth IEEE International Enterprise Distributed Object Computing Conference, (EDOC 2004)*, 2004, pp. 149–159.

[11] S.-Y. Hwang, H. Wang, J. Tang and J. Srivastava, “A probabilistic approach to modeling and estimating the QoS of web-services-based workflows”, *Inf. Sci.*, vol. 177, no. 23, pp. 5484–5503, 2007.

[12] H. Zheng, W. Zhao, J. Yang, and A. Bouguettaya, “QoS Analysis for Web Service Composition,” in *Proceedings of the 2009 IEEE International Conference on Services Computing, (SCC ’09)*, 2009, pp. 235–242.

[13] S. Rosario, A. Benveniste, S. Haar, and C. Jard, “Probabilistic qos and soft contracts for transaction-based web services orchestrations,” *IEEE T. Services Computing*, vol. 1, no. 4, pp. 187–200, 2008.

[14] R. Bellman, *Dynamic Programming*. Mineola, NY: Dover Publications, Inc., 2003.

[15] G. L. Choudhury, D. J. Houck, “Combined queueing and activity network based modeling of sojourn time distributions in distributed telecommunication systems,” in *Proceedings of 14th International Teletraffic Congress, (ITC 14)*, 1994, pp. 525–534.