Dynamic Profit Optimization of Composite Web Services with SLAs

M. Živković*, J.W. Bosman†, J.L. van den Berg*, R.D. van der Mei†, H.B. Meeuwissen*, and R. Núñez-Queija†‡,
*TNO, Delft, The Netherlands
†CWI, Amsterdam, The Netherlands
‡University of Amsterdam, The Netherlands

***Abstract*—In this paper we investigate sequential decision mechanisms for composite web services. After executing each sub-service within a sequential workflow, decisions are made whether to terminate or continue the execution of the workflow. These decisions are based on observed response times, expected rewards, and typical Service Level Agreement parameters such as costs, penalties, and agreed response–time objectives. We propose a model for the sequential decision–making process within which we explore a couple of decision algorithms. We benchmarked these algorithms against the profit made when executing the workflow without decision–making. We show that algorithm based on backward recursion principle of dynamic programming is optimal with respect to profit. Next, we analyse the structure of erroneous decisions for both algorithms and show that significant profit gains can be obtained by sequential decision making.**

***Index Terms*—Service Oriented Architecture, Response Time, Percentile Service Level Agreements, Sequential Decision, Backward Recursion, Dynamic Programming.**

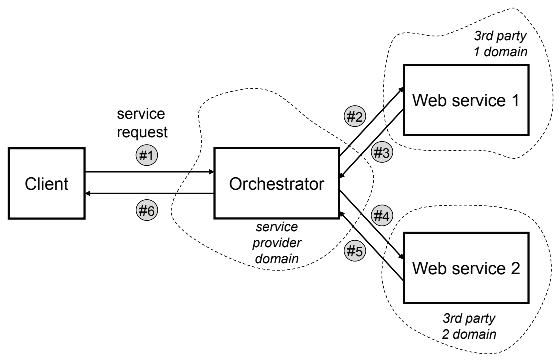

Fig. 1. Composite service orchestration

I. Introduction

Composite web services in a service oriented architecture (SOA) integrate multiple web services (also referred to as sub-services) which may be executed in different administrative domains. Each of these sub-services may be a composite service on its own. The provider of the composite web service typically runs an orchestrator that invokes the sub–services according to the workflow of the composite service. The workflow can be specified with the standardized Business Process Execution Language (BPEL) [1]. This is conceptually illustrated in Figure 1. Upon receiving a client service request (#1), the orchestrator sends a request to web service 1 (#2) in third-party domain 1. Upon receiving the response (#3), the orchestrator sends a request to web service 2 (#4) in 3rd party domain 2. Upon receiving the response (#5), the client receives a response (#6) to its service request.

For the commercial success of the composite web service, it is important that the service provider is able to offer the service at attractive price–quality ratios. To this end, the composite service provider (CSP) negotiates Service Level Agreements (SLAs) with the client and third-party domains. A service level agreement (SLA) is a legal contract that specifies the minimum expectations and obligations that exist between a service provider and a service consumer [2]. A single SLA may contain, among others, service level objectives (SLO), service level evaluation rules, measurements criteria, and ramifications of failing to meet (or indeed exceeding) these objectives (i.e. SLOs). The refund policies (penalties) for service-level violations can be specified relative to the service cost or in absolute terms. We refer to the client–CSP SLA as cSLA while the SLA between CSP and third party domains $i, i = \{1, 2, \ldots, N\}$ is referred to as iSLA$_i$. Both cSLA and iSLA considered in this paper contain response time SLOs as well as costs per composite web service request. Besides, cSLA contains possible penalty per composite service request as well. We assume that, once agreed, SLAs are "static", i.e., do not change during the execution phase.

In current practice, response–time SLOs indicate a hard bound, guaranteeing response times smaller than a certain value. Using hard bounds within SLOs leads to pesimistic response–time targets and may be overly inefficient as shown in [3]. Therefore, we assume percentile–based SLOs within iSLA. We assume that the probability density function (PDF) of response times can be estimated, as shown in, e.g. [13]. Figure 2 depicts the composite web service as a sequential service chain. The random variable D represents the end–to–end response time of the composite service, while the random variable D_i represents the individual response time of web service i. The cSLA also contains a penalty deadline D_p. When CSP responds to a service request within deadline D_p, it receives a reward, otherwise it pays a penalty to its clients. The response–time objective within iSLAs is specified as percentage of service requests that will be served within deadline D_{ip}. There are no iSLA violations as long as the percentage of requests served within deadline is at least as

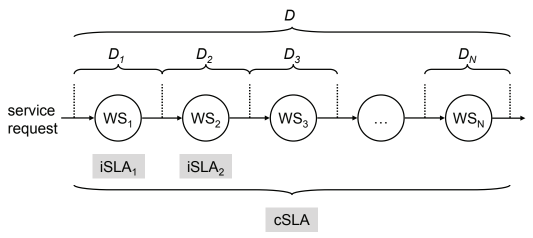

Fig. 2. Sequential composite service orchestration

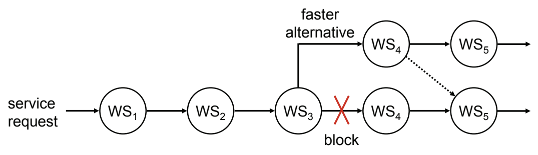

Fig. 3. Possible actions by an intelligent orchestrator.

indicated within the iSLA. Further, we assume no penalties are charged in case of iSLA violations. Thus, cash flows from CSP to the third parties are usage based while the cash flow between CSP and client is determined by the fact whether the response time requirements specified in the cSLA are met or not.

The "intelligent" orchestrator could make additional decisions next to the simple execution of a pre–defined workflow. The main motivation behind such decisions is profit optimization. Examples of such decisions are:

– stop any further processing for a particular service request if it is unlikely that the end–to–end penalty deadline D_p can be met, as indicated by the label "block" in Figure 3.

– proceed with a particular service request, but use an alternative, faster, possibly more expensive, chain, indicated as "faster alternative" in Figure 3.

We focus on the scenario of *sequential* composite service workflows. Our algorithms may directly be applicable in a more general setting, since workflows can often be mapped to the sequential ones, using the aggregation and reduction rules described in [4].

In this paper, we investigate the problem of profit maximization for composite web services by allowing the dispatcher to block requests before invoking the next web service in the chain. In fact, we derive decision logic for request control that uses actual response time measurements as well as parameters in the cSLA and iSLAs to optimize profit for the composite web service provider. The main question we investigate in this paper is whether the orchestrator should ever terminate the execution of a single request within the composite service workflow, and if so, when to do so.

The main contributions of the paper include:

- A model for sequential decision–making within the (arbitrary) composite workflow.
- Estimation of the profit(s) for different cost structures when two types of decision–making algorithms are used:
  – the optimal, dynamic programming backward recursion for different cost structures
  – forward looking algorithms for the decision–making process,

  These profits are also compared with the basic scenario when no decision–making is applied.
- Analysis of the errors made during the decision process for both algorithms.

The paper structure is as follows: After an overview of related work in Section II, we describe the model we use in Section III. In Section IV, the description of the dynamic programming algorithm as well as forward looking algorithms are given. Following the results from experiments on the algorithms presented in Section V, we conclude the paper in section VI and give directions for further research.

II. Related work

In the literature, QoS aspects of composite web-based services have received a lot of attention. A brief overview is given below. Jaeger et al. [4] describe and analyse a mathematical model to calculate QoS properties of a composition. Based on the execution order and the QoS of the invoked sub–services the deduction of costs, availability, reliability and response time is possible. This paper represents a *de-facto* standard, together with the work of Cardoso et al. [5]. However, these papers address aggregation of the workflows and may be used for aggregation of the workflows into sequential–based ones. The vast majority of the current practice Web Service SLAs use *hard contracts* for response times, i.e., where the response times are required to be less than a certain *fixed* value. When composing SLAs, hard rules are used as an addition in case of response times for the architecture as defined in Fig. 2.

In very few papers, the response times within particular SLA have been analysed using *soft contracts* [3], [6]. Rosario et al. [3] propose the use of probabilistic contracts. Probabilistic contracts consist of agreeing on some probability distribution for the QoS parameters in consideration. Using probabilistic contracts for the contracted sub-services, in combination with measurements based QoS estimates for the others, it is possible to synthesize the probabilistic contract of the overall orchestration. In our approach, we explicitly allow for violations of hard bounds for the composite service, taking into account the cost implied by these violations. Our algorithms then compute the expected violation costs and trade these against the potential gains.

Kaiqi and Perros [7] discuss guarantees on percentiles of the response time for a sequential workflow pattern, and analyse the single request. Different from our approach, there is no cost structure involved.

The sequential decision problem and optimization approaches are well-known in literature [8], [9]. However, none of the many papers dedicated to it describe its usage within the context of the composite web services.

Cardellini et al. [10] discuss the solution to dynamically adapt at runtime the composite service configuration. The solution is based on a service selection scheme that minimizes the cost while guaranteeing the negotiated QoS specified within the SLAs. Different form our work, this paper is about composition, not about decision making.

Finally, Shaaban and Hillston [11] propose a cost–based admission control approach with main goal to preserve QoS in Internet commerce systems. Different from rejecting customer requests in a high–load situation, a discount–charge model is used, that depends on system load and internal service structure, in order to encourage customers postponing their requests. They apply a scheduling mechanism based on load forecasting in order to schedule user requests in more lightly loaded time periods. However, this has not been applied within the composite service environment, and the main goal is to preserve the QoS of the system, not to maximize the profit when admitting requests to the system.

III. Model description

In this section we describe our mathematical model for analysis of the problem introduced in Section I, using the representation of service chains in Figure 2, in which the (individual) web services are executed sequentially.

There are in total N individual, transaction–based sub–services in the chain. The response time $D_i \geq 0$ of service i is a random variable (r.v.) for which (an estimate of) the PDF is given. In practice, this PDF may either be estimated from measurements carried out by the CSP, or the third-party domains may publish, or otherwise make available, such information. The PDFs are general (i.e., not assumed to belong to any particular class of distributions) and we denote these with $f_i(x)$ for sub–service i. In our model these PDFs are not time–dependent, but in practice their estimates can be dynamically adjusted. The realisation d_i of the response time is a single value drawn from the given PDF. Let $p_i := \mathbb{P}\{D_i \leq D_{ip}\}$, i.e. the probability p_i that D_i is smaller than the guaranteed value D_{ip} specified within the respective iSLA$_i$. In our mentioned experiments, we also assume that all target probabilities are equal, i.e. $p_i = p$. We assume that response times of individual web services are mutually independent. The r.v. representing the end–to–end response time, D, is given as $D = D_1 + \cdots + D_N$. Since the response time PDFs of individual web services are known, the response time distribution $f^*(x)$ for the composite web service, can be computed by convolving the response time PDFs of the third party domains. The convolution(s) of the probability density functions can be done offline, i.e. before the composite service is deployed, but must be updated if dynamic estimates for the PDFs are used. The realisation of D is represented by d, and let $p_{e2e} := \mathbb{P}\{D \leq D_p\}$ be the probability that D is smaller than the given target D_p.

The individual SLA, iSLA$_i$ $(i = 1, 2, \ldots, N)$, is composed of the following elements:

- The response-time penalty deadline D_{ip} [time unit], i.e. the response time that is "promised" to the CSP.
- The probability that the penalty deadline is met, p_i $(0 < p_i \leq 1)$.
- The reward that the sub-service provider gets for executing a single request within the penalty deadline, c_i [money unit]; from the CSP viewpoint, this value represents cost.

The cSLA (composite SLA) is composed as follows:

- The end-to-end response time penalty deadline D_p [time unit].
- The probability that the end-to-end penalty deadline is met p_{e2e} $(0 < p_{e2e} \leq 1)$.
- The reward R [money unit] that the CSP gets for executing a single request within penalty deadline D_p.
- The penalty that the CSP pays to the end customer when the request has not been completed, or the agreed end-to-end deadline for given request is not met, V [money unit].

The SLAs, once agreed, do not change, i.e. above mentioned parameters do not change either with time. Due to the lack of space we will not consider the case when PDFs change with time.

The CSP obtains the reward when the response time of the request is below the promised value D_p. Otherwise, the CSP pays the penalty to the clients. The promised deadline D_p is met with probability p_{e2e}. Suppose that the request has been served by the composite service within agreed time. Then the profit is $R - Nc$. The biggest loss for the CSP occurs when the complete chain is traversed and the end-to-end deadline is not met. This loss is $V + Nc$. Taking into account probabilities specified in SLAs, in order for the CSP to make profit, the following should hold

$$p_{e2e} \cdot (R - Nc) - (1 - p_{e2e}) \cdot (V + Nc) > 0.$$

However, it may happen that the orchestrator decides to terminate the execution of the composite request after service k $(1 < k < N)$, for example because the promised response time D_p is already breached, and continuation of the service execution would only increase the loss of the composite service provider. Also, depending on the response times and costs involved, the orchestrator may decide to terminate the further execution because the risk of not reaching the promised response time is significantly high.

IV. Sequential decision processes

In this section we describe two decision algorithms. The first algorithm, called the Dynamic Programming Algorithm (DPA), is based on the backward-recursion principle [12]. Next, we describe the Forward Looking Algorithm (FLA), which is based on ideas expressed in [8].

The profit gains for both the DPA and the FLA are calculated in absolute terms, and compared to the "worst case scenario" (WCS) when no decision is taken at all, resulting in each request always traversing the complete composite service chain.

The algorithms considered make decisions whether to further traverse the chain after each sub–service within the chain has been executed. The actions that the orchestrator could then take are:

- `stop` the execution. The CSP pays the costs to the third party providers. Besides, the CSP pays the penalty to the client(s),
- `continue` the execution.

If the last service is executed the CSP pays the actual costs to the third-party providers. When the end–to–end deadline is met the CSP obtains reward from the client(s); otherwise, as for the decision to stop, the CSP pays the penalty to the client(s). For all algorithms, the first service within the chain is always executed.

A. The Dynamic Programming Algorithm

The DPA is based on a one-dimensional recursion that calculates a set of optimal decisions, namely `continue` or `stop`, given the decision is taken before service k and D^* units of time budget are left before the agreed deadline is exceeded. Using the notation introduced in previous sections, the backward recursion is formulated as follows: For $1 < k < N$,

$$\mathbb{E}[R_N \mid D^*] = \max\Big\{-V, -c - \mathbb{P}(D_N > D^*)V + \mathbb{P}(D_N \le D^*)R\Big\}, \tag{1a}$$

$$\mathbb{E}[R_k \mid D^*] = \max\Big\{-V, -c - \mathbb{P}(D_k > D^*)V + \int_0^{D^*} f_k(t)\mathbb{E}[R_{k+1} \mid D^* - t]\mathrm{d}t\Big\}, \tag{1b}$$

$$\mathbb{E}[R_1 \mid D_p] = \max\Big\{-V, -c - \mathbb{P}(D_1 > D_p)V_1 + \int_0^{D_p} f_1(t)\mathbb{E}[R_2 \mid D_p - t]\mathrm{d}t\Big\}. \tag{1c}$$

Here $\mathbb{E}[R_k \mid D^*]$ $(1 < k < N)$ is the expected reward in case the decision made is `continue` and $\mathbb{E}[R_1 \mid D_p]$ is the total expected reward given an overall deadline of D_p time units. In practice, the recursive equations (1a), (1b), (1c) can be solved efficiently by discretizing the response-time distributions.

B. Forward-looking decision process

The Forward Looking Algorithm (FLA) decision mechanism is based on evaluation of the expected reward once each of the services k $(k = 1, \ldots, N-1)$ is executed. The orchestrator decides whether the rest of the chain, i.e. $N_\mathrm{r} := N - k$ services will be executed. At the decision–taking moment, the orchestrator knows:

a) the response times of the services already executed, i.e. the realisations $d_1, d_2, \ldots, d_k$ of $D_1, D_2, \ldots, D_k$, respectively.

b) the response time probability-density functions for each of the N_r services to be executed. The orchestrator calculates the following:

– The remaining "response-time budget", $D^* = D_p - \sum_{i=1}^{k} d_i$.

– $\mathbb{P}_0$, the probability that D^* will be met. The reward of the CSP in this case is $R - Nc$.

– Probability that D^* will not be met, $\mathbb{P}_{\mathrm{LOSS}} = 1 - \mathbb{P}_0$. The loss in this case is $-V - Nc$.

After the service k $(k = 1, 2, \ldots, N)$ has been executed, the algorithm calculates the expected profit (as if the whole chain would be traversed) and compares it to the loss in case the execution is terminated after service k.

- The expected profit is

$$\mathbb{E}Pf_k = \mathbb{P}_0 \cdot (R - Nc) - \mathbb{P}_{\mathrm{LOSS}} \cdot (V + Nc). \tag{2}$$

- The loss if the execution terminates (after execution of the service k) is

$$\mathrm{TH}_k = -V - \sum_{i=1}^{k} c_i = -V - kc. \tag{3}$$

The FLA is then based on the following decision rule:

$$\mathbb{E}Pf_k > \mathrm{TH}_k = \begin{cases} \text{true}, & \texttt{continue} \\ \text{false}, & \texttt{stop}. \end{cases} \tag{4}$$

V. Numerical results

The framework and the two algorithms described in this paper can be used to numerically determine the corresponding decision strategies. In order to test a wide range of (heuristic) strategies, including the WCS without decision control, we also developed a simulation tool. In this section we give an overview results that were obtained with our simulation tool, to illustrate the effectiveness of our algorithms and demonstrate the practical usability of our algorithms. Here, we limit ourselves to comparison of our two algorithms with the WCS. Further experimentation for a wider set of strategies is subject of current research.

In the experiments presented here, we have chosen the following parameter settings:

- Requests consist of a chain of $N = 6$ services.
- The response-time distributions used for experiments were normal (with truncation of negative values) for all services.
- The SLA target probabilities that (individual or composite) service would meet its deadline were identical and set to one of the values 0.75, 0.8, 0.85, 0.9 or 0.95.
- The reward parameter is fixed to a value of 2.5.
- The cost and penalty parameters have been chosen in such a way that the expected profit per request $\mathbb{E}[R_{WCS}]$ made by the CSP is positive in the worst case scenario (WCS) and remains the same. The relationship between V and c is determined by

$$c = \frac{Rp - V(1-p) - \mathbb{E}[R_{WCS}]}{N}.$$

The two main experimental setups are

Setup A: In this setup response time PDFs of the services were identical, and represented by (truncated) normal distributions with the same parameters for expectation and variance. The values selected were $\mathcal{N}(\mu, \sigma^2) = \mathcal{N}(15, 4)$.

Setup B: In this setup the response time PDFs of the services were not identical, although all considered distributions were (truncated) normal. The expectation and variance of the first two services within the chain are chosen to be significantly

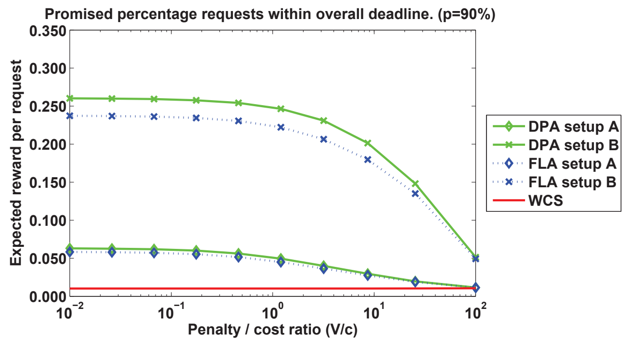

Fig. 4. Comparison expected reward with different algorithms for setup A and B.

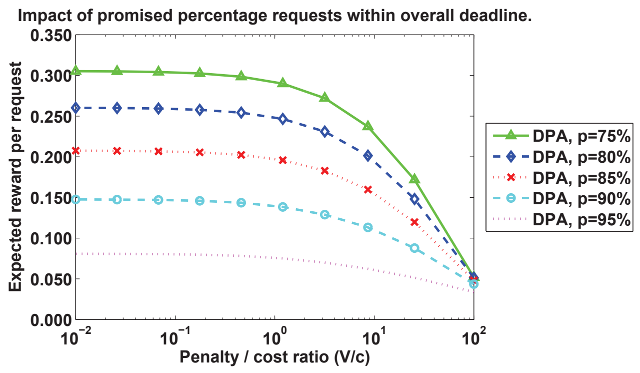

Fig. 5. Comparison expected reward with different p values for setup B.

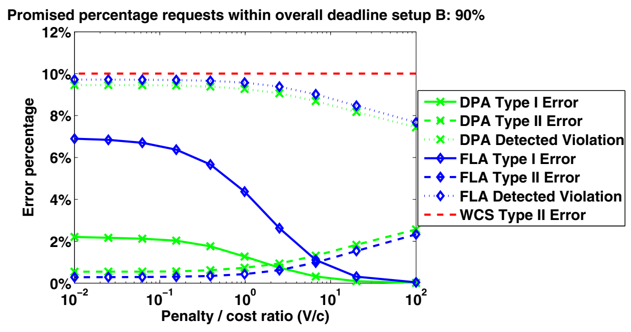

Fig. 6. Comparison of errors made by different algorithms for setup B.

higher than expectation and variance of the services within the rest of the chain. The values selected for the experiments were $\mathcal{N}_1 = \mathcal{N}(150, 25)$, $\mathcal{N}_2 = \mathcal{N}(100, 16)$, and $\mathcal{N}_{3-6} = \mathcal{N}(15, 4)$.

The fact that distributions are identical in setup A allows that delays (i.e. deadlines) that are not met “early” in the chain could be compensated by better than expected performance as the request traverses the chain. The chance of prematurely terminating the execution of the composite request therefore increases. Good algorithms would show quantifiable improvements even for these scenarios.

The setup B has been used to verify the accuracy of the algorithms observed, since delays introduced by services at the beginning of the chain cannot be made up for by the services close(r) to the end of the chain. It may be expected that the algorithms would defer requests at one of the first two services more often than at one of the last four services.

Our results emphasise two performance aspects of the algorithms:

- Increase in the reward (absolute and/or relative) that algorithms yield when compared to the worst case scenario (WCS), i.e. the ”do nothing” scenario.
- Amount of erroneous decisions made.

A. Reward improvements

For each value of the probabilities, cost parameters selection made, and both setup A and B, we have recorded the performance of the DPA and the FLA. Some results are summarized in Figure 4 for setup A and B, and Figure 5 for setup B.

The following conclusions can be made from the experiments:

- Setup A shows less improvement than setup B.
- If the penalty becomes much higher than the cost parameter, the expected reward drops to the WCS value.
- Performance results are better in case when deadline probabilities chosen are smaller. Put simply, the higher probability deadline is met, the less requests could be terminated by the algorithms. The less requests that are terminated, the less profit is gained by the algorithms when compared to WCS.

B. Decision errors

We define two type of decision–making errors: Type I and Type II. **Type I** errors are made within the decision process when execution of the request is terminated before the end of the chain is reached, while the optimal decision is to traverse the whole chain. **Type II** errors are made within the decision process when the request traverses the complete chain, while the optimal decision is to terminate the request.

Suppose that a Type I error has been made by terminating the request once the service k has been executed. The composite service provider is left without reward and also has to pay the penalty. The net loss is therefore $-R - V + (N-k)c$. Similarly, Type II errors reduce the algorithm performance for that request to the worst case scenario, and the net loss reduces to the costs made by redundant invocation of services within the chain. Therefore, in general, reduction of Type I errors leads to larger profit increase than the reduction of Type II errors.

The experimental results are summarized in Figure 6, which shows percentage of errors made for different values of the penalty/cost ratio V/c. We observe that, for both DPA and FLA, the percentage of Type I errors decreases as the ratio V/c increases. This is due to the larger impact of the penalty, as stopping becomes extremely expensive compared to the additional invocation cost of the next sub–service. Similar reasoning explains percentage increase of Type II errors as ratio V/c increases. Besides, DPA is not optimal when it comes to Type II errors, and the percentage of Type II errors for both DPA and FLA is below 2.5% for the observed ratio interval.

VI. Conclusions and Further Research

In this paper we have discussed decision-making mechanisms for the composite web services within service oriented architecture. First, we have considered the model of the composite web service in which the CSP pays a certain amount per request to each third party provider. In addition, the CSP pays a penalty to the customer whenever the execution of the composite service request is prematurely terminated or the agreed end-toend response time is above the threshold promised by the CSP. Further, a stochastic model has been formulated in which the sub–service response times are modeled as stochastic variables with probability distributions known *a priori*. This model allows us to apply our solution to any workflow that could be reduced to the sequential workflow (the vast majority of the current practice workflows). Based on response-time realisations decisions are made for optimizing the expected profit. Based on the derived stochastic model, two different decision-making approaches have been discussed: the DPA and the FLA. The DPA is optimal when it comes to the CSP profit, whereas the latter approach performs better when it comes to type II errors. In the performed simulations, two main scenarios have been examined. Firstly (symmetric scenario) we considered a composite web service consisting of the chain of six sub–services with identical response time distributions. In the second, asymmetric scenario the first two sub–services have a significantly larger expected response time and variance. We have shown that huge profit gains can be made with any of the algorithms used. We have also analysed the structure of the errors made in the decision-making processes. As expected, the performance gains in asymmetric scenario(s) were much higher than in the case of symmetric scenarios.

The results presented have raised several challenging questions for further research. First, we presented numerical results for rather small model instances, with six sub–services, whereas in practice, services chains may be of considerably larger size. This raises the need to address the scalability of the solution approaches presented (in terms of computational complexity), and about the cost reductions that can be obtained in those cases. In this context, also notice that for large N the classical Central Limit Theorem suggests that the optimal policy becomes less sensitive to the response-time performance of the individual sub–services, which opens up the way for CSPs to negotiate much less strict, and hence much cheaper, SLAs with respect to the response times. Second, in this paper we have neglected the dependencies between the response times of subsequent requests to the same sub–service, while in reality such dependence often exists and may have a main impact on the response-time performance of subsequent requests. Taking into account such dependencies may lead to significant cost reductions and opens up a challenging area for further research.

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] (2007, Apr.) Web Services Business Process Execution Language Version 2.0, OASIS. [Online]. Available: http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf

[2] C. Ward, M. J. Buco, R. N. Chang and L. Z. Luan, "A Generic SLA Semantic Model for the Execution Management of e–Business Outsourcing Contracts," in *Proc. Third Int. Conf. on E-Comm. and Web Tech. EC-WEB '02*, Springer–Verlag, London, UK, pp. 249–257, 2002.

[3] S. Rosario, A. Benveniste, S. Haar and C. Jard, "Probabilistic QoS and Soft Contracts for Transaction-Based Web Services Orchestrations," *IEEE Trans. Services Computing*, vol. 1., no. 4, 2008.

[4] M. C. Jaeger, G. Rojec-Goldmann and G. Mühl, "QoS Aggregation for Service Composition using Workflow Patterns," in *Proc. IEEE EDOC*, pp. 149–159, 2004.

[5] J. Cardoso, J. Miller, A. Sheth and J. Arnold, "Quality of Service for Workflows and Web Service Processes," *Jour. Web Sem.*, vol. 1, no. 3, pp. 281–308, 2004.

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

[7] K. Xiong and H. Perros, "Resource Optimization Subject to a Percentile Response Time SLA for Enterprise Computing," in *Proc. IEEE Global Telecommun. Conf. GLOBECOM '06*, San Francisco, CA, USA, 2006.

[8] K. J. Arrow, D. Blackwell and M. A.Girshick, "Bayes and Minimax Solutions of Sequential Decision Problems," *Econometrica*, vol. 17, no. 3 and 4, 1949.

[9] M. H. De Groot, *Optimal Statistical Decisions*. Wiley–Interscience, WCL Edition, 2004.

[10] V. Cardellini, E. Casalicchio, V. Grassi and F. Lo Presti, "Adaptive Management of Composite Services under Percentile-based Service Level Agreements," in *Proc. of 8th Int. Conf. on Service Oriented Computing ICSOC 2010*, San Francisco, CA, Dec. 2010.

[11] Y. A. Shaaban and J. Hillston, "Cost–based admission control for Internet Commerce QoS enhancement," *Jour. Electron. Commer. Rec. Appl.*, vol. 8, no. 3, pp. 142–159, 2009.

[12] R. Bellman, *Dynamic Programming*. Dover Publications, Inc., Mineola, New York, 2003.

[13] T. van Zandt, "How to Fit a Response Time Distribution," *Psychonomic Bulletin & Review*, vol. 7, no. 3, pp. 424–465, 2000.