Optimal Quality of Service Control in Communication Systems

Bosman, J.W.

2014

document version
Publisher's PDF, also known as Version of record

Run-time Optimization of Composite Web Services with Response Time Commitments

6

In this chapter we address dynamic decision mechanisms for composite web services. We represent the composite web-service as a (sequential) workflow of tasks. For each task within this workflow, a number of third-party service alternatives may be available. We assume that the third-party service (task) alternatives offer the same functionality at different price-quality levels. Before a task in the workflow will be executed, a service alternative has to be selected that implements the task functionality. Decisions are represented in a decision strategy which defines decisions for all tasks in the workflow. Our goal is to find a dynamic strategy that maximizes the expected benefit for the composite service providers. For each task, the service-selection decision may be based on information about: observed response times in the current workflow, sub-service costs, response time characteristics of the alternatives, and end-to-end response time objectives with the corresponding rewards and violation penalties. We propose an approach, based on dynamic programming, to determine the optimal, dynamic selection policy. Numerical examples show significant potential gain in expected benefits using the dynamic approach compared to other, non-dynamic approaches.

6.1 Background

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 workflow. The workflow is based on an unambiguous functionality description of a task ("abstract service"), and several alternatives ("concrete services") may exist that implement such a description [93]. With respect to functionality, all concrete services that match the same task service are identical. We refer to [42], and references therein, for an overview of QoS control frameworks.

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. [26, 112, 113]). The main problem addressed in these papers is how to select one concrete service per task for a given workflow. This selection is made with the goal to guarantee the QoS of the composite service (as expressed in the respective SLA) while at the same time

[6] This chapter is based on [120], [122] and [121].

optimizing certain objectives like cost minimization. For the same QoS parameter, different SLAs are possible, ranging from those based on single value (e.g. expected value, median, etc.) to those models that are based on probabilistic models, represented by probability density functions, [117, 7, 122]. The choice of model impacts how QoS of composite services is determined, and consequently, how service selections are made. In static service composition solutions, the composition would remain unchanged the entire life-cycle of the composite web service. In static service composition solutions, the SLA violations could occur relatively often, leading to providers' losses and customer dissatisfaction. This is due to the high variability of the service environment, e.g. response time, and due to the fact that this is not well-reflected in static service composition solutions.

One way to address the problem of SLA violations is the request replication approach, as presented in [111]. Although showing relative potential, the request replication may be inefficient and not scalable compared to other approaches. It is suggested in [27, 114, 77] that, based on observations of the actually realized performance, re-composition of the service may be triggered, in order to mitigate the issue of SOA violations. These "reactive" solutions may select new concrete service(s) for the given workflow during the re-composition phase. 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 [27, 114, 77] describe *when* to trigger such (re-composition) event, and which adaptation actions may be used to improve overall performance. On the other hand, dynamic service compositions consider how to dynamically adapt the service composition at runtime [28].

It seems natural to address the issue of dynamic service composition within SOA by application of Markov Decision Processes (MDPs) and variants thereof. Abundo et al. [1] use MDPs to calculate the admission policy which allows the orchestrator to decide whether to accept or reject a new potential user in such a way to maximize the benefit while guaranteeing non-functional QoS for already admitted users. Wang et al. [106] use MDPs to model service composition with the aim to create automatically an abstract workflow of the service composition that satisfies functional and non-functional requirements, and also to allow the composite service to adapt dynamically to a varying environment. Yet another example of the pro-active solution for SOA-based systems based upon MDP has been considered in [92]. For a sequential workflow, a replacement of the atomic service is initiated when the maximum loss a consumer could bear for each service occurs. The authors do not consider service replacement at each SLA violation. The problem of an adaptive service composition is also addressed in [105], where a solution based on a hierarchical reinforcement learning method has been considered.

All of the mentioned MDP-based solutions show significant improvements over the respective non-dynamic (i.e. static) service composition solutions. However, relatively little is done with respect to a analysis of the services' QoS parameter space.

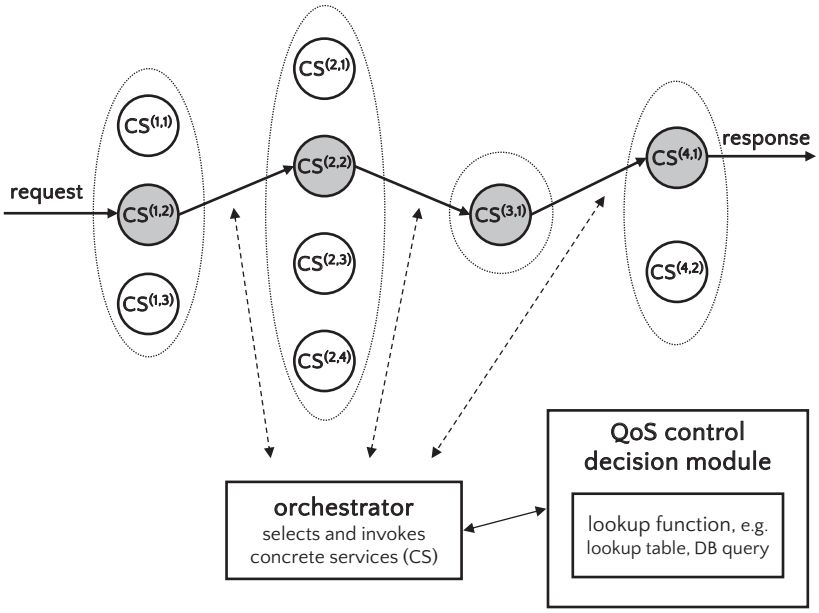


Figure 6.1: *Composite web service depicted by a sequential workflow. Dynamic service composition is based on pre-calculated response-time thresholds using dynamic programming.*

Motivated by this, in this chapter we conduct a thorough empirical evaluation based on our analysis from [122]. In [122] we analyzed our dynamic service composition approach that is based on DP. The objective is to maximize benefit for the composite service provider, while committing to the response-time objective that is part of the agreed end-to-end SLA. The investigation in this chapter aims to fully explore the impact of the reward, penalty and execution cost parameters, relating to the expected value and the variance of the response-time probabilistic models (distributions). To the best of our knowledge, the impact of the number of the concrete services (alternatives) and the positions of the alternatives within the workflow has not been analyzed so far within the context of the MDP-based SOA dynamic service composition. We quantify the answer to the question how many alternatives are advisable at the "beginning", "middle" and "end" of a workflow, and enlight this analysis with respect to the number of services within the workflow.

After illustration of MDP-based solution in Section 6.2 we describe in Section 6.3 the model of the system under consideration. In Section 6.4 we describe the DP approach we abridged for our purposes. We explain the procedure and derive formu-

lae used to determine response time thresholds. Simulation results are presented and discussed in Section 6.5 in which we illustrate the influence of different system parameters. In Section 6.6 we conclude this chapter with directions for further research.

6.2 Motivating example

To illustrate our dynamic service selection approach, we consider the case of sequential workflows. Figure 6.1 depicts a sequential workflow consisting of four tasks (defined as abstract services). Each task maps to a number of concrete services (alternatives). Per task, the number of alternatives is three, four, one and two, respectively. The response times of the concrete web services are represented by random variables for with it is assumed that the probability distributions are known in advance. See Remark 6.3.1 for when this is not the case. The execution costs per concrete service are known in advance as well. Our approach is tailored for sequential workflows. However, our approach is applicable to any workflow that could be aggregated and mapped into a sequential one. This can be done by the calculations of the aggregated services for the most frequently used workflow patterns in which the probabilistic models are used as explained in [122], [117], and [7]. In case of arbitrary workflow patterns, an efficient numerical method could be used, as presented in [34]. See Appendix 6.A for an example how where an (non sequantial) example workflow is aggegrated into a sequential one.

After each execution of a single task within the workflow, the orchestrator decides for the next task which concrete service alternative will be executed. To this end the orchestrator compares the remaining end-to-end deadline time-budget to the pre-calculated response-time thresholds for each service. The thresholds are retrieved from a pre-calculated lookup table. In the most extreme example, after the execution of, e.g. the first task within the workflow, the promised end-to-end deadline may already has been violated. In such a case, the dynamic selection algorithm should choose the cheapest alternatives for all remaining tasks in the workflow. Similarly, when only the last task remains to be executed, and the promised deadline is jeopardized, it may be sensible to select the more expensive service with better deadline-meeting promise than the cheaper service with the much smaller probability of meeting the deadline.

When the client's request meets the agreed end-to-end deadline, the composite service provider 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. The exact relation between the metric of percentile conformance and the monetary penalties charged on the provider due to non-conformance (hereon referred to as the "penalty function") is an important parameter of the SLA [16]. There are multiple ways in

relating the two and thus a variety of penalty functions can be chosen. We have adopted the step-wise penalty function in this case with the desired conforming percentile of 100% [16].

The response-time thresholds are calculated before the first request is admitted to the system, where the DP 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 each client request may be served by a different chain of alternatives.

6.3 Model

A composite service consists of N of tasks or *abstract services*, denoted by $T_1, \dots, T_N$, that have to be performed in sequential order. For each task i there are M_i alternative implementations available, which are called *concrete services*, denoted by $CS^{(i,j)}$, for $j = 1, \dots, M_i$. The composite service processes incoming service requests. To this end, each request is assigned a unique concrete service alternative for each task in the composition chain. In other words, each request is assigned a *workflow of concrete services* $CS^{(1,w_1)} \rightarrow \cdots \rightarrow CS^{(N,w_N)}$ that contains for each task T_i an invocation of a concrete service alternative $CS^{(i,w_i)}$. Each workflow is represented in a workflow vector $\boldsymbol{W}$, defined as:

$$\boldsymbol{W} := [w_1, \dots, w_N]. \tag{6.1}$$

The position of w_i in the vector corresponds to the position in the workflow and the values $w_i = 1, \dots, M_i$ correspond to the concrete service alternative at position i.

Figure 6.2 illustrates the case where $N = 4$, $M_1 = 3$, $M_2 = 4$, $M_3 = 1$, $M_4 = 2$ and $\boldsymbol{W} = [2, 2, 1, 1]$.

Per single composite service request, the orchestrator (illustrated in Figure 6.1) executes services one-by-one as indicated by the workflow. Before task T_i is executed, the orchestrator makes a decision which one of the M_i service alternatives to choose. Decisions from the orchestrator are based on information about response times from the concrete services. 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) [96], or alternatively, as the cumulative distribution function (CDF).

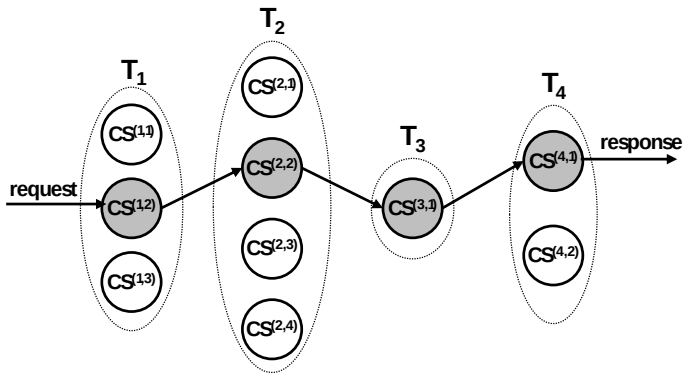


Figure 6.2: *Model with example workflow* $CS^{(1,2)} \rightarrow CS^{(2,2)} \rightarrow CS^{3,1} \rightarrow CS^{(4,1)}$*, corresponding workflow vector* $\boldsymbol{W} = [2, 2, 1, 1]$ *and* $M_1 = 3,\ M_2 = 4,\ M_3 = 1,\ M_4 = 2$.

6.3.1. 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 topic is addressed in Chapter 7.

Each concrete service $CS^{(i,j)}$ (the j-th alternative for task i) has a response time represented by a random variable $D^{(i,j)} \geq 0$ for $i = 1, \ldots, N$ and $j = 1, \ldots, M_i$. From the perspective of the response time, we model each concrete service as a black box, which means that for each random variable $D^{(i,j)}$ the 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. If a concrete service $CS^{(i,j)}$ is invoked a response time realization $d^{(i,j)}$ ($i = 1, \ldots, N,\ j = 1, \ldots, M_i$) is generated and a invocation cost of $c^{(i,j)}$ money units have to be paid. Because $CS^{(i,j)}$ is represented by random variable $D^{(i,j)}$ with given distribution, $d^{(i,j)}$ is a sample from the distribution associated with $D^{(i,j)}$. We assume that response times of concrete service alternatives are mutually independent. Using the mutual independence, for a given workflow $CS^{(1,w_1)} \rightarrow \cdots \rightarrow CS^{(N,w_N)}$ the response time distribution can be determined by taking the convolution of the distributions associated to the respective random variables:

$$D_{\boldsymbol{W}} = D^{(1,w_1)} \star D^{(2,w_2)} \star \cdots \star D^{(N,w_N)}. \tag{6.2}$$

The realization of a end-to-end response time, resulting from invoking workflow $\boldsymbol{W}$, is represented by $D_{\boldsymbol{W}}$. The overall deadline for a request handled by the orchestrator is denoted by δ_p. For each workflow $\boldsymbol{W}$ the probability of a successful response within δ_p is defined by:

$$p_{\boldsymbol{W}} := \mathbb{P}(D_{\boldsymbol{W}} \leq \delta_p). \tag{6.3}$$

The workflow invocation cost $c_{\boldsymbol{W}}$ for workflow $\boldsymbol{W} = (w_1, \ldots, w_N)$ is defined by:

$$c_{\boldsymbol{W}} := \sum_{i=1}^{N} c^{(i,w_i)}. \tag{6.4}$$

As for a fixed workflow $\boldsymbol{W}$ the composition is known, an explicit expression for the expected benefit per request can be formulated. We define $R^*_{\boldsymbol{W}}$ as the expected benefit per request for workflow $\boldsymbol{W}$:

$$R^*_{\boldsymbol{W}} := Rp_{\boldsymbol{W}} - V(1 - p_{\boldsymbol{W}}) - c_{\boldsymbol{W}}. \tag{6.5}$$

When decisions are made in a dynamic fashion, the concrete service workflow $\boldsymbol{W}$ is not fixed, but depends on the response time realizations of concrete services. In our approach the optimization problem is solved by applying DP. The key idea behind DP is that the optimization problem can be separated into smaller subset problems that are easier to solve. Solving the smaller subset problems will lead to the solution of the optimization problem that optimizes expected benefit. Dynamic programming consists of the following components:

- **State space** $\mathcal{S}$, in our case the state space is defined by $\mathcal{S} = \{1, \cdots, N\} \times \mathbb{R}^+$. Each state is a tuple $(i, b) \in \mathcal{S},\ (i \in \{1, \cdots, N\},\ b \in \mathbb{R}^+)$ combining position i in the workflow and remaining time until deadline violation b.
- **Decision epochs**, before execution of each next sub-service we have to select an concrete service alternative based on b.
- **Action space**, the set of possible actions for given state (i, b). At each decision moment at task T_i we have a set of concrete service alternatives $\{CS^{(i,1)}, \ldots, CS^{(i,M_i)}\}$, $i = 1, \ldots, N$.
- **Cost function**, defining cost for each action or state that is reached. If concrete service alternative $CS^{(i,j)}$ than invocation cost $c^{(i,j)}$ has to be paid. At the end of the workflow a reward R is earned is the deadline is not violated. Otherwise a penalty V has to be paid.
- **Value function** $P^{(i,*)}(b)$, representing the expected benefit for state $(i, b) \in \mathcal{S}$.

- **Policy** A, the set of decisions or actions over all states (i, b). The policy can be implemented by a lookup table where for given (i, b) the optimal pre-calculated decision is selected. An example is depicted in Figure 6.3 at Section 6.4. We refer to Section 6.4 for more details.

The combination of state space, decision epochs, action space, cost function and value function defines a DP problem where the solution results in policy that optimizes expected benefit. An optimal policy can be determined by defining a backward recursion on $P^{(i,*)}(b)$. This recursion is called the Bellman Equation [11]. The recursion is defined and implemented in Section 6.4. We denote the dynamic workflow that implements lookup table A as:

$$\boldsymbol{W}^d(A).$$

Generally there is no tractable explicit expression for the end-to-end response time distribution when the orchestrator chooses workflows dynamically like for the static workflow. For the orchestrator the realization of a end-to-end response-time distribution is denoted by D and has corresponding realization d. The fraction of requests within the deadline is

$$p := \mathbb{P}(D \leq \delta_p). \tag{6.6}$$

The orchestrator has to deal with two levels of SLAs:

First, the SLA agreed between the individual service provider (ISP) for the j-th service alternative of the i-th task and the composite service provider (CSP). These consist of the following elements:

- The response-time probability distribution function, $f^{(i,j)}(b)$.
- 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.

Second, the SLA between the CSP and its clients, that contains the following elements:

- The end-to-end response time penalty deadline δ_p [time unit].
- The fraction of response time realizations p_{e2e} that should be within the deadline δ_p.
- 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.

The orchestrator must choose workflows such that it will meet the required fraction p_{e2e} of requests within deadline δ_p:

$$\mathbb{P}(D \leq \delta_p) \geq p_{e2e}. \tag{6.7}$$

6.4 Algorithm description

In this section we describe how to optimize expected CSP benefit by formulating the dynamic service selection as a DP, [11]. Before a request is executed by the CSP a response time budget δ_p is available until response-time deadline δ_p is violated. Each execution of concrete service $CS_i(j)$ in the CSP workflow will result in a response-time realization $d^{(i,j)}$ and a remaining response-time budget B. After execution of $CS^{(i,j)}$, the remaining time budget B will reduce with $d^{(i,j)}$ time units. The term "dynamic" refers to the fact that workflows are selected on-line based on B, the remaining response-time budget until the deadline δ_p is violated. In other words, before the execution of each task, a service alternative must be selected, based on B. The DP optimizes the CSP benefit by taking into account the effect of future decisions. Since the decisions within the DP take in account (possible) effects subsequent decisions, a "backward recursion" method is needed for finding the optimal solution. The application of DP will result in a decision policy. The policy indicates, for given response time realizations, which one of the concrete service alternatives should be chosen in order to optimize the CSP's expected benefit per composite service request. The policy is determined by the current position within the sequential workflow i and the remaining time b ("time budget") till the overall deadline δ_p will be violated.

Define $\overline{P}^{(i,*)}(b) := \mathbb{E}[\text{Benefit} \mid B = b, I = i]$ as the expected benefit under the optimal DP policy. Here $B = b$ is the given response time budget , and i represents the task at the ith position in the workflow. The recursion starts with the terminal reward function for $b \geq 0$:

$$\overline{P}^{(N+1,*)}(b) = \begin{cases} R & \text{if } b > 0, \\ -V & \text{otherwise.} \end{cases} \tag{6.8a}$$

Using this function, we iterate backwards using the following equations, for $i = 1, \ldots, N, \quad j = 1, \ldots, M_i, \quad b > 0$:

$$\overline{P}^{(i,*)}(b) = \max_j \Big\{ -c^{(i,j)} + \overline{R}^{(i,j)}(b) + \overline{V}^{(i,j)}(b) \Big\}, \tag{6.8b}$$

$$\overline{R}^{(i,j)}(b) = \int_0^b f^{(i,j)}(t)\overline{P}^{(i+1,*)}(t-b)\mathrm{d}t, \quad \text{and} \tag{6.8c}$$

$$\overline{V}^{(i,j)}(b) = \int_b^\infty f^{(i,j)}(t)\overline{P}^{(i+1,*)}(0)\mathrm{d}t. \tag{6.8d}$$

Here $f^{(i,j)}(t)$ represents the response-time PDF of concrete service alternative j for task i, while the term $\overline{R}^{(i,j)}(b)$ represents the expected reward, when concrete service j (corresponding to task i) is executed for the given time-budget value $B = b$. The term $\overline{V}^{(i,j)}(b)$ represents the expected penalty for exceeding the overall deadline at task i while executing concrete service j for the given time budget value $B = b$. The expected reward and penalty functions take into account the impact of future decisions as represented by terms relating to $\overline{R}^{(i+1,*)}(b)$ in equations (6.8c) and (6.8d). Once the end of the workflow is reached (i.e., $i = N$) the current request has been processed.

The integrals in Equation (6.8c) will generally not result in tractable expressions. The problem can be solved numerically, by discretizing the distributions. 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 T^* and the size of h corresponds to the accuracy of the discretization. The discretized versions of the PDF $q_k^{(i,j)}$ are therefore defined as follows, for $i = 1, \ldots, N,\ j = 1, \ldots, M_i,\ b = 0, \ldots, T^*$:

$$T^* = \left\lceil \frac{\delta^*}{h} \right\rceil,$$
$$q_b^{(i,j)} = \mathbb{P}\big(D^{(i,j)} \leq h[b+0.5]\big) - \mathbb{P}\big(D^{(i,j)} \leq h[b-0.5]\big),$$

Generally, the larger the number of segments T^* the more accurate the discretization would be, but it would take longer time to calculate the respective PDF.

Using the discretization, the backward recursion can be transformed into a scheme that can be evaluated numerically. For given number of segments T^*, let the terms

$P_b^{(i,*)}$, $R_b^{(i,j)}$, and $V_b^{(i,j)}$ represent discretized versions of $\overline{P}^{(i,*)}(b)$, $\overline{R}^{(i,j)}(b)$, and $\overline{V}^{(i,j)}(b)$, respectively. The backward recursion formulae are then as follows:

$$P_b^{(N+1,*)} = \begin{cases} R & \text{if } b > 0, \\ -V & \text{otherwise.} \end{cases} \tag{6.9a}$$

Using this function, we iterate backwards using the following equations, for $i = 1, \ldots, N, \quad j = 1, \ldots, M_i, \quad b = 0, \ldots, T^*$:

$$P_b^{(i,*)} = \max_j \left\{ -c^{(i,j)} + R_b^{(i,j)} + V_b^{(i,j)} \right\}, \tag{6.9b}$$

$$R_b^{(i,j)} = \sum_{k=0}^{b} q_k^{(i,j)} P_{k-b}^{(i+1,*)}, \quad \text{and} \tag{6.9c}$$

$$V_b^{(i,j)} = \sum_{k=b+1}^{T^*} q_k^{(i,j)} P_0^{(i+1,*)}. \tag{6.9d}$$

While applying formulae (6.9b)-(6.9d), the corresponding decisions (actions) A can be obtained by storing the maximum arguments evaluated for $i = 1, \ldots, N, \quad b = 0, \ldots, T^*$ as:

$$A_b^{(i,*)} = \operatorname*{argmax}_{j=1,\ldots,M_i} \left\{ -c^{(i,j)} + R_b^{(i,j)} - V_b^{(i,j)} \right\}.$$

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 6.3. 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 task within the workflow. 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 6.3, and alternative 1 is to be selected.

(2) We have 4.9 time units remaining the decision is made for the task at position 3. The decision is marked by a cross $\times$, and points that service alternative 3 should be selected.

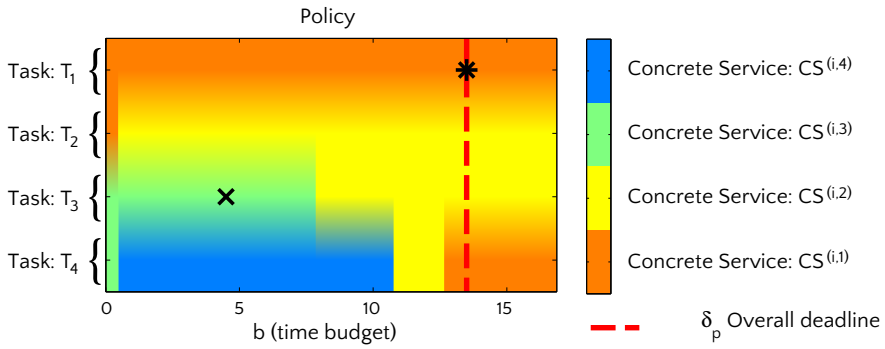


Figure 6.3: *Graphical example of a decision table.*

6.5 Numerical experiments

In this section we investigate the influence of the system parameters on the potential gain that can be obtained by applying dynamic service composition as compared to static service composition. To this end, we have performed a wide variety of numerical experiments. The results of these experiments are outlined below. The

Parameter	Definition
N	Number of tasks (workflow size)
M_i	Number of service alternatives for task i
$f^{(i,j)}$	PDF of response-time distribution $CS^{(i,j)}$
$\mu_{i,j}$	Mean response time of $CS^{(i,j)}$
$\sigma^2_{i,j}$	Variance of response time of $CS^{(i,j)}$
$c^{(i,j)}$	Cost of invocation of $CS^{(i,j)}$
δ_p	End-to-end deadline
p_{e2e}	Required fraction of responses within the deadline
R	Reward per successful request within deadline δ_p
V	Penalty per request not completed within deadline
K_R	Scaled reward
K_V	Scaled penalty

Table 6.1: *Overview of model parameters.*

parameters of the models discussed in Sections 6.3 and 6.4, and their corresponding value ranges, are listed in Table 6.1. The parameter space is highly dimensional and prohibits an exhaustive analysis of model instances. Therefore, we have defined a number of specific model instances that represent certain characteristics of service compositions. First, we study the impact of the input parameters reward, penalty

and cost on the potential gain in expected benefit obtained by dynamic versus static service composition. Second, we investigate the impact of the service-chain length on the potential gain. Third, we consider the impact of the number of available alternatives for the tasks, and their corresponding positions in the service chain, on the expected gain. For convenience, we assume that the response-time distributions for the j-th alternative of task i can be approximated by a log-normal distribution with mean $\mu_{i,j}$ and variance $\sigma^2_{i,j}$, see Table 6.1. We emphasize that this assumption is not restrictive, because our approach supports *all* non-negative probability distributions.

The *gain* G in expected benefit obtained by using optimal dynamic composition compared to the optimal static composition is defined as follows:

$$G := \frac{R^*_{dynamic} - R^*_{static}}{R^*_{static}} \times 100\%, \tag{6.10}$$

where $R^*_{dynamic}$ is the expected benefit per request under the optimal dynamic policy, obtained by applying the algorithm described in 6.4. Moreover, R^*_{static} is the expected benefit per request under the optimal static policy, obtained by an exhaustive search of all possible "paths" traversing the tasks in the workflow (see for example Figure 6.2, where there are 24 possible paths).

For our parameter study we need to choose a deadline δ_p. This has to be done such that experiments will generate sensible results. Therefore, we introduce the reference workflow $\boldsymbol{W}_{ref}$. Using the reference workflow we can relate a sensible deadline to a given end-to-end objective p_{e2e}. The reference workflow $\boldsymbol{W}_{ref}$ is determined by taking

$$\boldsymbol{W}_{ref} = \operatorname*{argmax}_{\boldsymbol{W}} \left[R p_{\boldsymbol{W}} - V(1 - p_{\boldsymbol{W}}) - c_{\boldsymbol{W}} \right], \tag{6.11}$$

$$c_{\boldsymbol{W}} := \sum_{i=1}^{N} c^{(i,w_i)}, \tag{6.12}$$

$$p_{\boldsymbol{W}} := \mathbb{P}(D_{\boldsymbol{W}} \leq \delta_p). \tag{6.13}$$

For given p_{e2e}, we denote by δ_p the deadline such that for the reference composition $\boldsymbol{W}_{ref}$, consisting of $CS^{(i,1)}$, $i = 1, \ldots, N$, it holds that $\mathbb{P}(D_{ref} < \delta_p) = p_{e2e}$.

6.5.1 Impact of reward, penalty and cost parameters

In this subsection we study the impact of concrete service-cost $c^{(i,j)}$, the reward R and the penalty V on the potential gain G, defined in (6.10). To this end, define the overall mean service cost by

$$\overline{c} = \sum_{i=1}^{N} \overline{c}_i, \quad \text{where} \quad \overline{c}_i = \frac{1}{M_i} \sum_{j=1}^{M_i} c^{(i,j)}. \tag{6.14}$$

Note that R, V and $\overline{c}$ are scale-invariant, in the sense that if these parameters are scaled to αR, αV and $\alpha \overline{c}$ for some $\alpha > 0$, then the optimal dynamic and static policies, and hence also the gain G, remain the same. Therefore, it is convenient to define the following two scaled reward and cost parameters:

$$K_R := \frac{R}{\overline{c}} \quad \text{and} \quad K_V := \frac{V}{\overline{c}}. \tag{6.15}$$

As an illustration, consider the following exploratory example with $N = 2$, $M_1 = 1$ and $M_2 = 2$, depicted in Figure 6.4, and where the cost parameters $c^{(i,j)}$ and the distribution parameters $\mu_{i,j}$ and $\sigma^2_{i,j}$ are listed in Table 6.2.

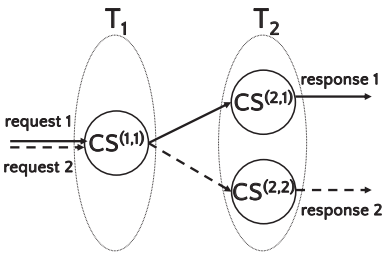


Figure 6.4: *Exploratory example with $M_1 = 1,\ M_2 = 2$.*

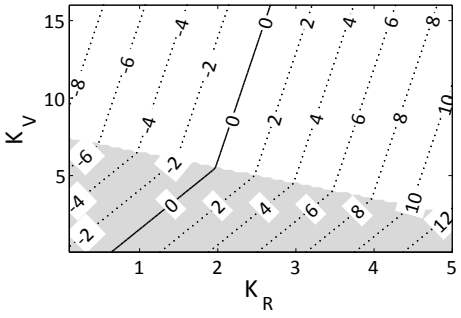
		Task $(i, \cdot)$					
		$(\mathbf{1}, \cdot)$			$(\mathbf{2}, \cdot)$		
		$c^{(i,j)}$	$\mu_{i,j}$	$\sigma^2_{i,j}$	$c^{(i,j)}$	$\mu_{i,j}$	$\sigma^2_{i,j}$
Service alternative $(\cdot, j)$	$(\cdot, \mathbf{1})$	1	5	4	1	5	4
	$(\cdot, \mathbf{2})$	n.a.	n.a.	n.a.	5	2.5	4

Table 6.2: *Global scenario.*

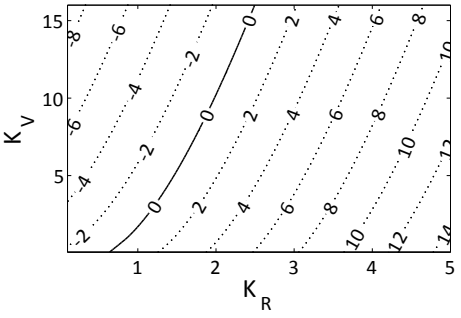
Figure 6.5a shows the contour lines of combinations (K_R, K_V) that lead to the same expected benefit, subject to the constraint that the reference *static* workflow (i.e.,

the workflow $[1, 1]$) leads to an end-to-end response time less than the deadline with probability $p_{e2e} = 80\%$. The values plotted on the dotted lined indicate the expected benefit. The grey area is the set of combinations for which the static workflow is $[1, 1]$, indicated by the solid line in Figure 6.4. The white area represents the combinations for which the static workflow $[1, 2]$ is optimal, indicated by the dashed line in Figure 6.4. Note that in these cases $p_{e2e} > 80\%$, because one chooses a faster alternative while maintaining the same deadline.

(a) *Expected benefit in static set-up.*



(b) *Expected benefit in dynamic set-up.*



(c) *Potential gain in expected benefit.*

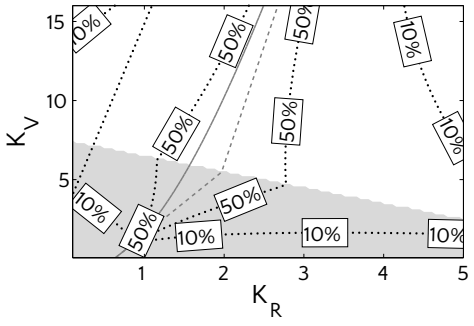


Figure 6.5: *Comparison between static and dynamic set-up for $p_{e2e} = 80\%$. Black contour lines represent equal values. The grey and white area correspond to the different optimal static compositions. Furthermore, the solid grey contour line is where $R^*_{dynamic} = 0$, the dashed grey contour line is where $R^*_{static} = 0$.*

Note that all iso-curves in Figure 6.5a are piecewise linear. More precisely, one may verify using (6.5) that for a given optimal static workflow the iso-curves are of the form $K_V = aK_R + b$, with

$$a = \frac{p_{\boldsymbol{W}}}{1 - p_{\boldsymbol{W}}} \quad \text{and} \quad b = \frac{R^*_{static} + c_{\boldsymbol{W}}}{\overline{c}(p_{\boldsymbol{W}} - 1)}, \tag{6.16}$$

where R^*_{static} is defined in (6.10), and where $c_{\boldsymbol{W}}$ is the cost for the optimal static workflow $\boldsymbol{W}$, $p_{\boldsymbol{W}}$ is probability a request using $\boldsymbol{W}$ will be within deadline δ_p and $\boldsymbol{W}$ is the optimal static workflow for which holds that $\boldsymbol{W} = [1, 1]$ in the grey area and $\boldsymbol{W} = [1, 2]$ in the white area. Moreover, it is readily verified from (6.5) that the switching curve of the optimal workflow (i.e., the border separating the grey and the white area) is given by:

$$K_V + K_R = \frac{c_{[1,1]} - c_{[1,2]}}{\overline{c}(p_{[1,1]} - p_{[1,2]})}, \tag{6.17}$$

where $c_{[1,k]}$ is the cost for workflow $[1, k]$ $(k = 1, 2)$ as defined in (6.4), $\overline{c}$ is defined in (6.14) and $p_{[1,k]}$ is the probability that the response time meets the deadline if the static workflow $[1, k]$ is optimal.

Figure 6.5b shows the results for the dynamic counterpart of Figure 6.5a, and Figure 6.5c shows the iso-curves for the relative gain G, defined in (6.10). Most remarkably, the iso-curves depicted in Figure 6.5c are wedge-shaped with a sharp angle at the switching curve (6.18). This suggests that the relative gain G has the highest values at the switching curve. In other words, *the information about the response times at the first task is particularly crucial to decide about the optimal path.*

The solid grey line is the contour line over the combinations (K_V, K_R) where $R^*_{dynamic} = 0$, and the dashed grey line show the combinations for which $R^*_{static} = 0$. We observe that the distance between these two lines is maximal on the switching curve, as expected. Moreover, note that the green line gives the boundary where positive benefit is expected: combinations to the left of this curve will result in loss in expectation.

Figures 6.6a to 6.6c show similar results to those in Figures 6.5a to 6.5c, but with $p_{e2e} = 90\%$.

6.5.2 Impact of number of alternatives

We will now investigate the impact of the number of available alternatives for each of the tasks on the gain G to be obtained by using dynamic composition. To this end, we consider a service chain of length $N \geq 3$, and with $M_i = 1$ for $i \neq 2, N$, and

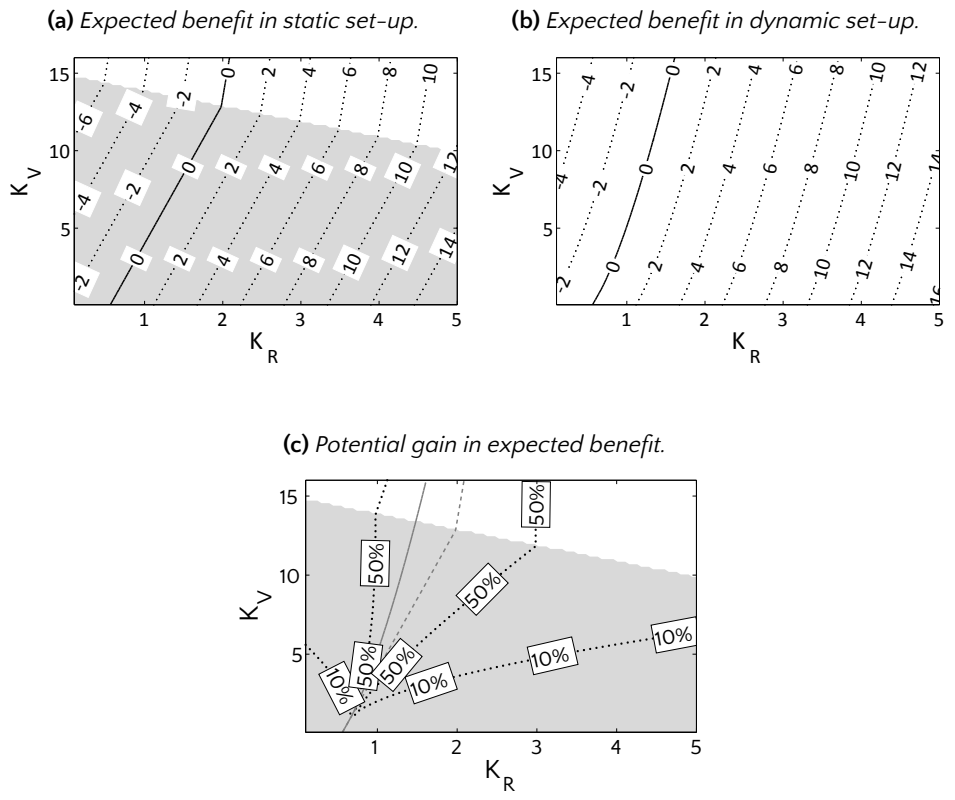


Figure 6.6: *Comparison between static and dynamic set-up for $p_{e2e} = 90\%$. Black contour lines represent equal values. The grey and white area correspond to the different optimal static compositions. Furthermore, the solid grey contour line is where $R^*_{dynamic} = 0$, the dashed grey contour line is where $R^*_{static} = 0$.*

where M_2 and M_N are varied as $\{1, 2, 3, 4\}$. The cost and distribution parameters are listed in Table 6.3, and moreover, we take $R = 26$, and $V = 130$. Note that it is readily verified from (6.14) that $\overline{c} = 26$, and from (6.15) that $K_R = 1$ and $K_V = 5$.

Figure 6.7 illustrates an example with $M_2 = 2$ and $M_N = 3$. For each of the model instances, we have calculated (1) the optimal static composition and its corresponding expected benefit R^*_{static}, (2) the optimal dynamic composition and its corresponding expected benefit $R^*_{dynamic}$, and (3) the relative gain G, defined in (6.10).

In particular we expect that, due to increasing uncertainty when traversing a service composition workflow, more alternatives are desired at the end of the workflow. For our analyses we define a service chain of length N where at the second position and at the last position the number of concrete service alternatives is varied. With this

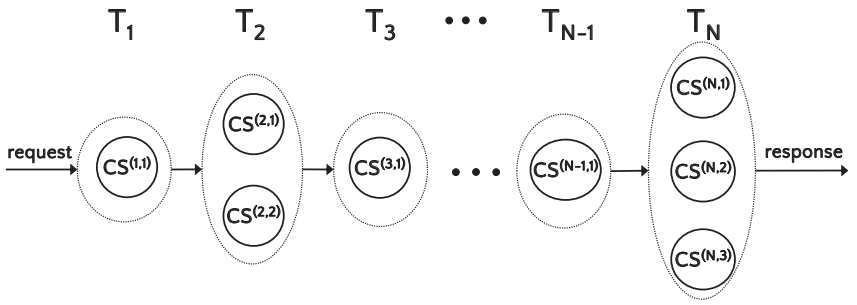


Figure 6.7: *Configuration for analyses on impact of position of alternatives at task T_2 at position 2 and task T_N at position N. In this case $M_2 = 2$, $M_N = 3$ and $M_i = 1$ for $i \in \{1, \ldots, N\} \setminus \{2, N\}$.*

set-up we can compare the relative value of alternatives in the beginning and at the end of the service composition chain. The results are presented in tables where the rows correspond to M_N and columns correspond to M_2. In other words, the rows correspond to the number of concrete service alternatives at position N and the columns correspond to the number of available concrete service alternatives at position 2. The parameter values are presented in Table 6.3. Results of the case where $N = 8$ and $p_{e2e} = 90\%$ for the reference composition are represented in Table 6.4. We observe that increasing the number of alternatives at position 2 in the workflow results in a neglectable increase in gain G. However increasing the number of alternatives at position 8 results in a increase in gain up to 42.1%. If we take a closer look to the expected benefit of the static composition approach in Table 6.4 we observe that the third and fourth alternative are never used as the expected benefit does not increase when the third and fourth alternative are available. Furthermore, we observe that the table is symmetric for the number of alternatives at positions 2 and 8. This is caused by the fact that workflows in the static scenario are fixed and therefore, if the alternatives at both positions are similar, the position where the number alternatives is increased has no effect on expected benefit. However, for the dynamic composition the position where the number of alternatives is increases has a dramatic effect. At the expected dynamic benefit results in Table 6.4, for the case with dynamic workflows, we observe that for position 8 the availability of alternatives 3 and 4 has a significant impact on benefit. At position 2 the availability has hardly any effect as the advantage of optimal dynamic selection is averaged out over the successive workflow response times. At position 8 the dynamic workflow enables the orchestrator to take advantage of the 3th and 4th alternative where the static workflow is not optimally using the 2th alternative and does not even use the 3th and 4th alternative.

		Task $(i, \cdot)$					
		$\mathbf{i} \notin \{\mathbf{2}, \mathbf{N}\}$			$\mathbf{i} \in \{\mathbf{2}, \mathbf{N}\}$		
		$c^{(i,j)}$	$\mu_{i,j}$	$\sigma^2_{i,j}$	$c^{(i,j)}$	$\mu_{i,j}$	$\sigma^2_{i,j}$
Service alternative $(\cdot, j)$	$(\cdot, \mathbf{1})$	1	5	16	1	5	16
	$(\cdot, \mathbf{2})$	n.a.	n.a.	n.a.	3	2.5	4
	$(\cdot, \mathbf{2})$	n.a.	n.a.	n.a.	9	1.25	1
	$(\cdot, \mathbf{2})$	n.a.	n.a.	n.a.	27	0.675	0.25

Table 6.3: *Cost and distribution parameters.*

Gain		N_2			
		1	**2**	**3**	**4**
N_8	**1**	0.0%	0.0%	0.4%	0.4%
	2	16.6%	14.6%	14.8%	14.8%
	3	37.1%	25.1%	25.3%	25.3%
	4	42.1%	27.6%	27.8%	27.8%

Dynamic benefit		N_2			
		1	2	3	4
N_8	**1**	2.44	5.43	5.45	5.45
	2	6.33	8.21	8.23	8.23
	3	7.44	8.97	8.98	8.98
	4	7.71	9.15	9.16	9.16

Static benefit		N_2			
		1	2	3	4
N_8	**1**	2.44	5.43	5.43	5.43
	2	5.43	7.17	7.17	7.17
	3	5.43	7.17	7.17	7.17
	4	5.43	7.17	7.17	7.17

Table 6.4: *Impact on availability of alternatives.*

6.5.3 Length of the composition

In section 6.5.2 we observed that the effect of alternatives at the start of the workflow can vanish due to averaging of response times over the succeeding services. We will now investigate the effect of service-chain length. The service chain used for the experiments is depicted in Figure 6.8 and has length N. There is one alternative at position 1 ($M_1 = 1$) while there are two alternatives at position $i \geq 2$ ($M_i = 2$ for $i = 2, \ldots, N$). The parameter space is large, therefore we limit us to workflows where there are two concrete-service alternatives for each task. Concrete-service

		Task $(i,\cdot)$					
		$(\mathbf{1},\cdot)$			$(\mathbf{2},\cdot)$		
		$c^{(i,j)}$	$\mu_{i,j}$	$\sigma^2_{i,j}$	$c^{(i,j)}$	$\mu_{i,j}$	$\sigma^2_{i,j}$
Service	$(\cdot,\mathbf{1})$	1	5	16	1	5	16
alternative $(\cdot,j)$	$(\cdot,\mathbf{2})$	n.a.	n.a.	n.a.	3	2.5	4

Table 6.5: *Cost and distribution parameters for the impact of the service-chain length.*

alternative parameters are defined in Table 6.5. Reward parameter R, penalty parameter V are related to overall mean service cost $\overline{c}$ as is defined in (6.14) using scaled reward and penalty K_R and K_V which are defined in (6.15). Deadline δ_p is chosen as defined in (6.11). Note that there are N different static workflows as for the static workflow the position of alternatives does not matter (the set of alternatives is equal for all tasks). We define a workflow for a system with N tasks as $\boldsymbol{W}_k$, $k = 0, \dots, N-1$ where k represents the number of faster alternatives that is invoked in the workflow. In our case the reference workflow becomes $\boldsymbol{W}_{ref} = \boldsymbol{W}_0 = [1, 1, \dots, 1]$ (a workflow consisting of only regular (slow) alternatives). We chose to vary parameters K_R and K_V in the ranges $K_R \in (0; 5]$ and $K_V \in (0; 50]$. Figures 6.9-6.13 contain results for service-chain length $N = 3, 6, 8, 15, 20$ respec-

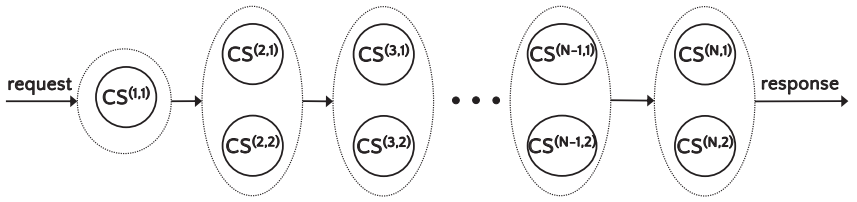


Figure 6.8: *Configuration for analyses on the impact of the service-chain length,* $M_1 = 1,\ M_i = 2,\ i = 2, \dots, N$.

tively with end-to-end probability $p_{e2e} = 90\%$ for the reference workflow. In these figures different gray-scale levels of the contour areas identify different benefit gain G. Each grey-scale area corresponds to a specific range of relative gain G as specified in the color bar. The arrows identify the (dashed) lines that separate different optimal static policies e.g. the number of faster alternatives used. We observe wedge shapes at the edges of the gray-scale contour areas. These wedges correspond to the switching curves where the static policy adds another faster alternative to the workflow, like the wedge shaped curves around the switching curve in section 6.5.1. Red lines identify the switching curves between different optimal static workflows. Arrows, left from the vertical axis, identify what static optimal paths are separated.

Similar to Section 6.5.1, more gain is found around the static optimal path switching curves. Switching curves have an expression similar to (6.18): for $k = 0, \ldots, N-1$,

$$K_V + K_R = \frac{c_{W_k} - c_{W_{k+1}}}{\overline{c}(p_{W_k} - p_{W_{k+1}})}. \tag{6.18}$$

p_{W_k} is the probability that a static workflow with k faster alternatives will generate a response within deadline δ_p. Observe that if the service-chain length N increases then the grey-scale contour areas become wider. This implies that longer workflows potentially have a higher benefit gain G when comparing dynamic and static workflows.

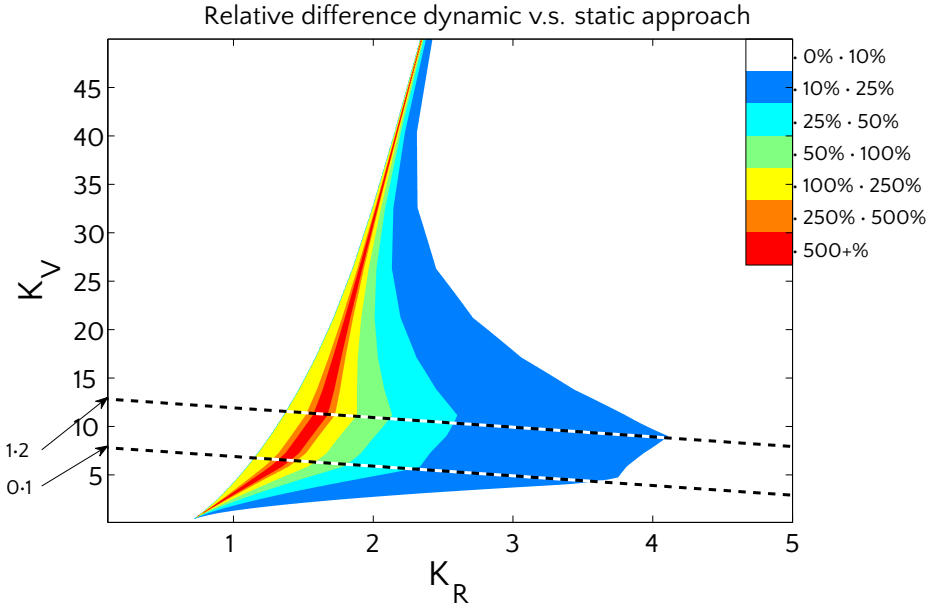


Figure 6.9: *Contour plot for the relative gain G with $N = 3$ and $p_{e2e} = 90\%$.*

6.5.4 Arbitrary alternatives

We consider a service workflow illustrated in Figure 6.14 for our experiments. This sequential workflow consists of $N = 4$ tasks. For each task 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 experimental 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).

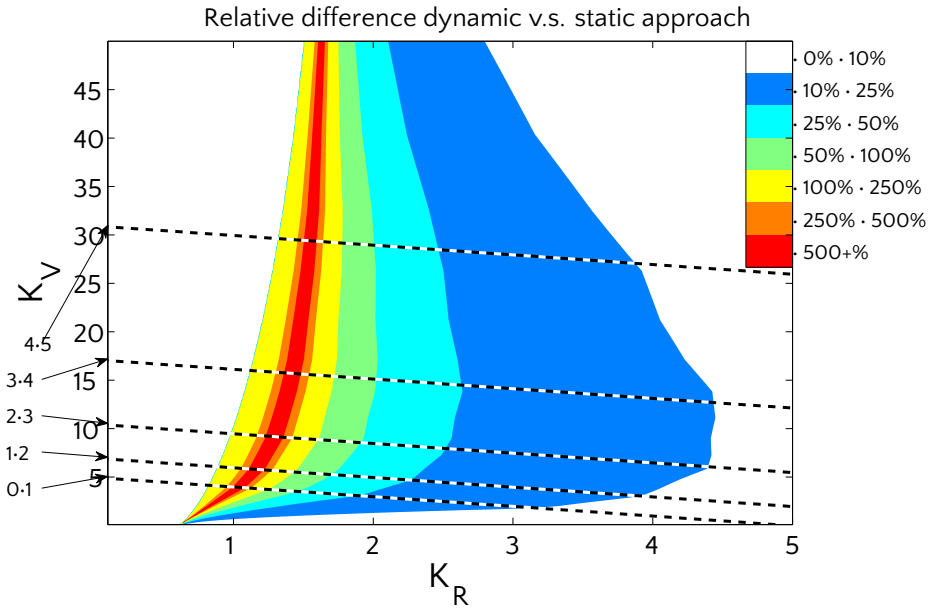


Figure 6.10: *Contour plot for relative gain G with* $N = 6$ *and* $p_{e2e} = 90\%$.

Let $\boldsymbol{W}_{SW} = [w_1, w_2, w_3, w_4]$ represent the service composition for the Static Workflow (SW) strategy, where $1 \leq w_j \leq M_j,\ j = 1, 2, 3, 4$. We want to compare the SW and DP strategies with fixed scaled cost parameter K_V and scaled reward parameter K_R as defined in (6.15). Let $W_{statopt}$ be the optimal static service composition. We choose to dimension the deadline δ_p such that the optimal static composition has at least a fraction of successful requests within the δ_p equal to p_{e2e}:

$$\mathbb{P}(D_{W_{statopt}} < \delta_p) \geq p_{e2e}. \tag{6.19}$$

Furthermore we tailor the reward and penalty parameters for optimal static composition "path" such that the expected reward R^*_{SW} for the static composition equals a predefined value. For the SW strategy R^*_{SW} is given by

$$R^*_{SW} = R \cdot p_{e2e} - V \cdot (1 - p_{e2e}) - c_{\boldsymbol{W}_{SW}}. \tag{6.20}$$

The expression above is similar to (6.5) with workflow cost $c_{\boldsymbol{W}_{SW}}$ as defined in (6.4). In Section 6.5.1 we observed that most relative gain in benefit can be expected where the expected benefit of static composition is small. Therefore, we use the SW strat-

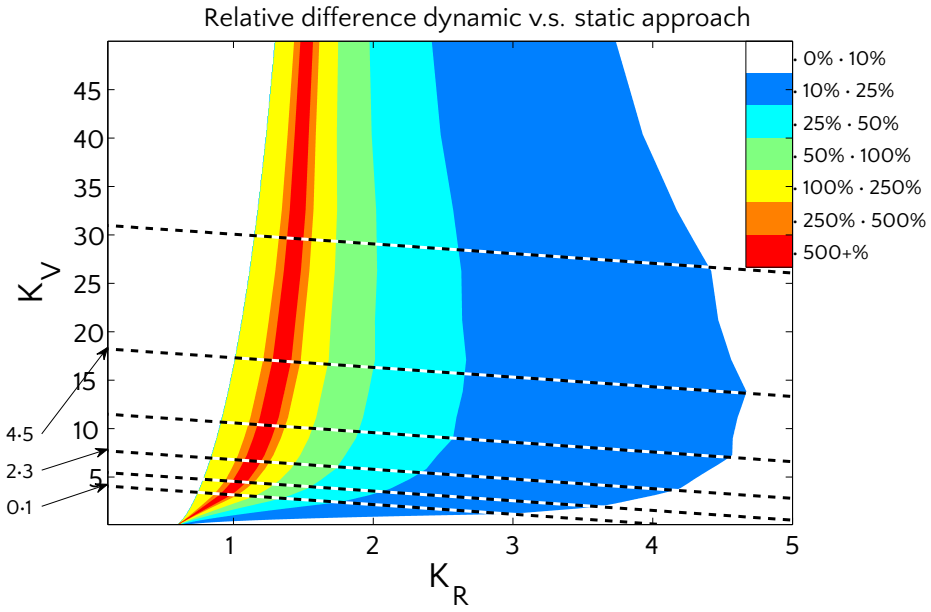


Figure 6.11: *Contour plot for relative gain G with* $N = 8$ *and* $p_{e2e} = 90\%$.

egy for the benchmarking and we set the value $R^*_{SW} = 0.01$. Taking into account (6.20) the values for parameters R and V satisfy the following equations:

$$R = \frac{R^*_{SW} + c_{\boldsymbol{w}_{SW}} + K_V \cdot (1 - p_{e2e}) \cdot \overline{c}}{p_{e2e}}, \qquad (6.21)$$
$$V = \frac{R^*_{SW} + c_{\boldsymbol{w}_{SW}} - K_R \cdot p_{e2e} \cdot \overline{c}}{p_{e2e} - 1}.$$

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 the symmetric scenario simulations is to illustrate the importance of the position of, and the number of, service alternatives within the workflow. The choice of the parameters for symmetric scenario is given in Table 6.6; here the parameters of the concrete service alternatives are the same for all tasks. When the number of alternatives for task is e.g. two, we always consider alternative 1 and alternative 2 in our experiments.

In the asymmetric scenario, which corresponds to parameter Table 6.7, the concrete services have different mean response times for tasks at different positions in the

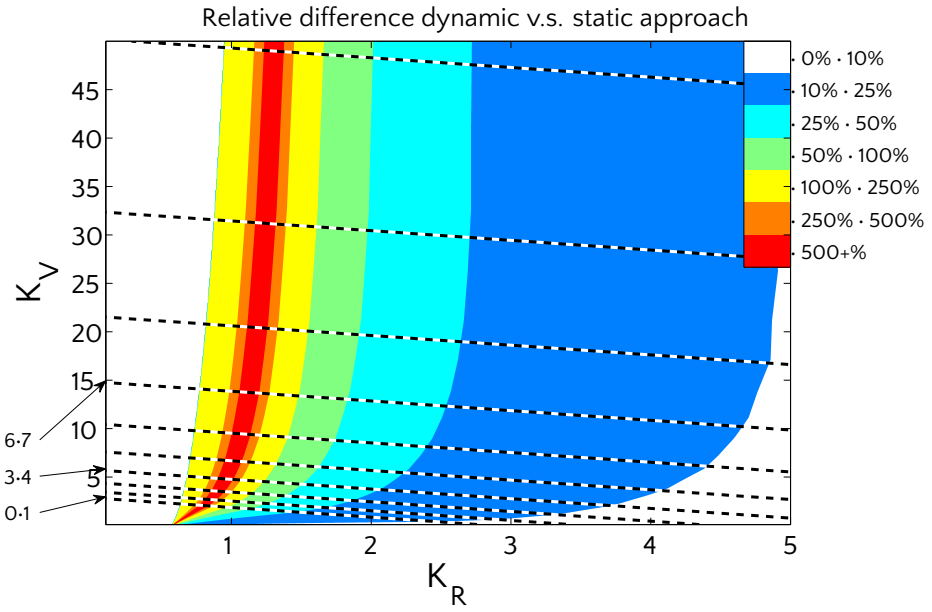


Figure 6.12: *Contour plot for relative gain G with $N = 15$ and $p_{e2e} = 90\%$.*

		Task $(i, \cdot)$		
		$\mathbf{i} \in \{\mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{4}\}$		
		$c^{(i,j)}$	$\mu_{i,j}$	$\sigma^2_{i,j}$
Service alternative $(\cdot, j)$	$(\cdot, \mathbf{1})$	1	5	4
	$(\cdot, \mathbf{2})$	5	2.5	4
	$(\cdot, \mathbf{3})$	10	1.25	16
	$(\cdot, \mathbf{4})$	50	0.5	0.0009

Table 6.6: *Concrete service alternatives symmetric scenario*

.

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.

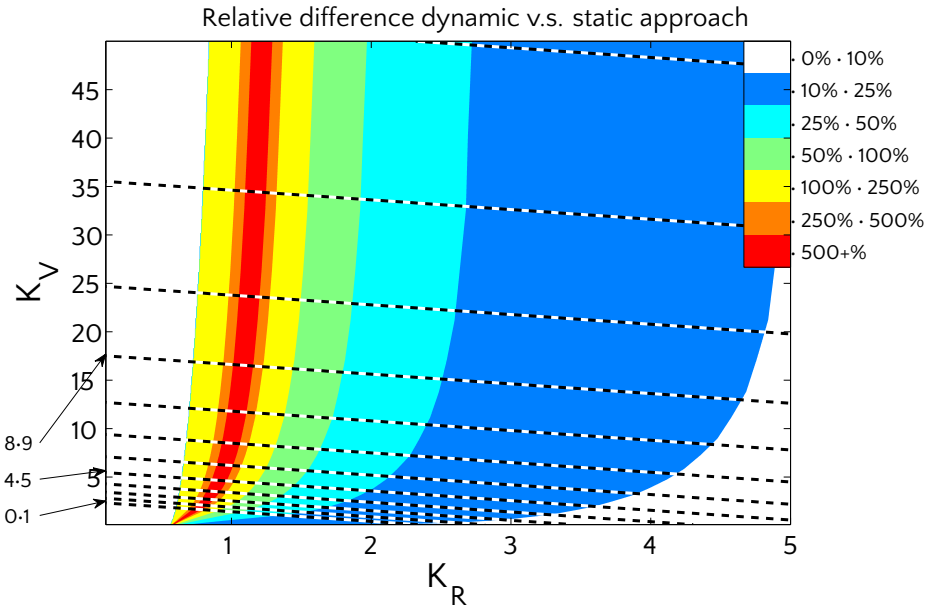


Figure 6.13: *Contour plot for relative gain G with* $N = 20$ *and* $p_{e2e} = 90\%$.

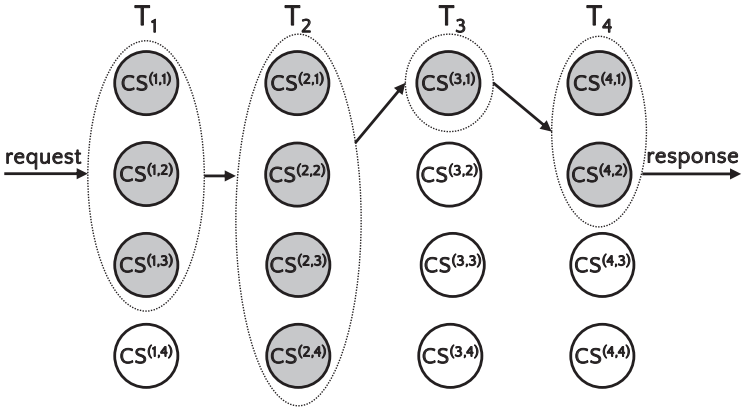


Figure 6.14: *Example workflow where number of alternative concrete services are represented by permutation* $(M_1, M_2, M_3, M_4) = (3, 4, 1, 2)$.

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.

6.5.5 Results

For all possible permutations of the number of concrete service alternatives (see Table 6.8) the expected benefit $\mathbb{E}[R]$ is calculated for both symmetric and asymmetric scenarios and DP and SW algorithms. The results are summarized in Figure 6.15 (symmetric scenario) and Figure 6.16 (asymmetric scenario). For both figures, the alternative count configurations are ordered on expected benefit for the DP algorithm.

In Figure 6.15 we observe that the DP solution achieves the lowest benefit 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 position and no alternatives at the end of the workflow resulting in no possibility to recover from (possible) large service response time accumulated from the previous tasks in the workflow. Furthermore, the DP-based solution always picks the same alternative for the first service in the workflow, not taking any advantage of available alternatives for this service. On the other hand, the highest benefit for the DP solution is achieved for the permutation $(M_1, M_2, M_3, M_4) = (1, 2, 3, 4)$. The largest number of 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. From Figure 6.15 seven regions can be identified labeled by capital letters *A*, *B*, *C*, *D*, *E*, *F*, *G*. Each region is separated by line where the configuration change resulted in a significant increase in expected benefit. The changes that correspond to the most significant increase in benefit are listed in Table 6.10. In this table, labels M_3 and M_4 correspond to the number of alternatives available at the end of the workflow. We observe from Figure 6.15 that six configurations in regions *F* and *G* with the highest benefit are those where the

		Task $(i, \cdot)$					
		$(\mathbf{1}, \cdot)$			$(\mathbf{2}, \cdot)$		
		$c^{(i,j)}$	$\mu_{i,j}$	$\sigma^2_{i,j}$	$c^{(i,j)}$	$\mu_{i,j}$	$\sigma^2_{i,j}$
Service alternative $(\cdot, j)$	$(\cdot, \mathbf{1})$	1	5	4	5	10	16
	$(\cdot, \mathbf{2})$	5	2.5	4	1	9.5	64
	$(\cdot, \mathbf{3})$	10	1.25	16	10	1.5	0.25
	$(\cdot, \mathbf{4})$	50	0.5	0.0009	50	1	0.0025

		Task $(i, \cdot)$					
		$(\mathbf{3}, \cdot)$			$(\mathbf{4}, \cdot)$		
		$c^{(i,j)}$	$\mu_{i,j}$	$\sigma^2_{i,j}$	$c^{(i,j)}$	$\mu_{i,j}$	$\sigma^2_{i,j}$
Service alternative $(\cdot, j)$	$(\cdot, \mathbf{1})$	1	0.5	0.04	1	2.5	1
	$(\cdot, \mathbf{2})$	5	0.4	0.04	5	2.45	2.25
	$(\cdot, \mathbf{3})$	10	0.3	0.0025	10	1	4
	$(\cdot, \mathbf{4})$	100	0.05	0	50	0.25	0.0004

Table 6.7: *Concrete service alternatives asymmetric scenario.*

		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
	2	3	4	2	4	2	3	3	4	1	4	1	3
	3	2	2	3	3	4	4	1	1	3	3	4	4
	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
	2	2	4	1	4	1	2	2	3	1	3	1	2
	3	1	1	2	2	4	4	1	1	2	2	3	3
	4	3	3	3	3	3	3	4	4	4	4	4	4

Table 6.8: *Indices of alternative configurations.*

number of alternatives for task 4 is the highest. Configurations in region *G* perform better than configurations *F* as the number of alternatives for task 3 is 3 in region *G*, and either 1 or 2 in region *F*. Additionally, the benefit "jump" between regions *E* and *F* is due to the number of alternatives (3 and 4, respectively) for task 4. The "jump" between regions *D* and *E* is caused by the fact that in region *D* for abstract service 3 only alternatives 1 and 2 are available while for region *E* 4 alternatives are available for task 3. The largest DP benefit improvement is when for the task $i = 4$, concrete service alternative 4 is considered (see Table 6.6). 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 benefit. Another (smaller) benefit increase can be observed when concrete service alternative 3 becomes available for task at position 3. This can be explained by the fact that the 90th percentile for alternative 3 is still lower than 1 and 2 (see Table 6.9) despite its higher variance.

	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

Table 6.9: *Concrete service alternative 90th percentile.*

In Figure 6.16 the results are given for the asymmetric scenario. For the asymmetric scenario it is harder to observe any 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.

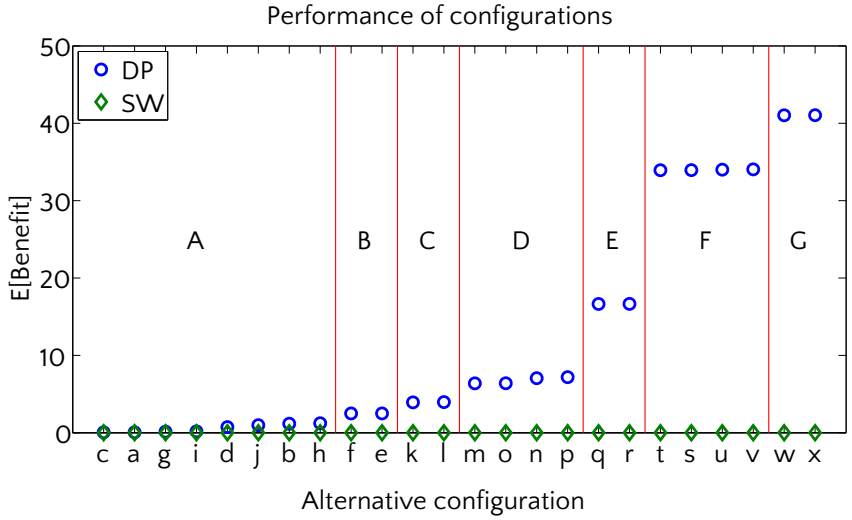


Figure 6.15: *Expected benefits per request in case of asymmetric scenario; comparison between static SW (static workflow) and DP (dynamic programming) for different configurations. See Table 6.10 for quick comparison of characteristics for regions D, E, F, G.*

		Region			
		D	**E**	**F**	**G**
Task *i* #	**M_3**	< 3	4	< 3	3
alternatives: M_i	**M_4**	3	3	4	4

Table 6.10: *Symmetric scenario regions from Figure 6.15.*

For alternative configurations $(M_1, M_2, M_3, M_4) = (4, 3, 2, 1)$ and $(M_1, M_2, M_3, M_4) = (1, 2, 3, 4)$ the lowest expected and the highest benefit are achieved, respectively. The largest benefit 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 6.17 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 6.8.

The main conclusions and some "rules of a thumb" that could be drawn from the Figures 6.15 - 6.17 are as the following:

- Dynamic, on-the-fly service composition results in higher benefits for the CSP, compared to optimal "static" service composition (Figures 6.15 and 6.16). While the expected reward per request for the SW strategy is 0.01, for the

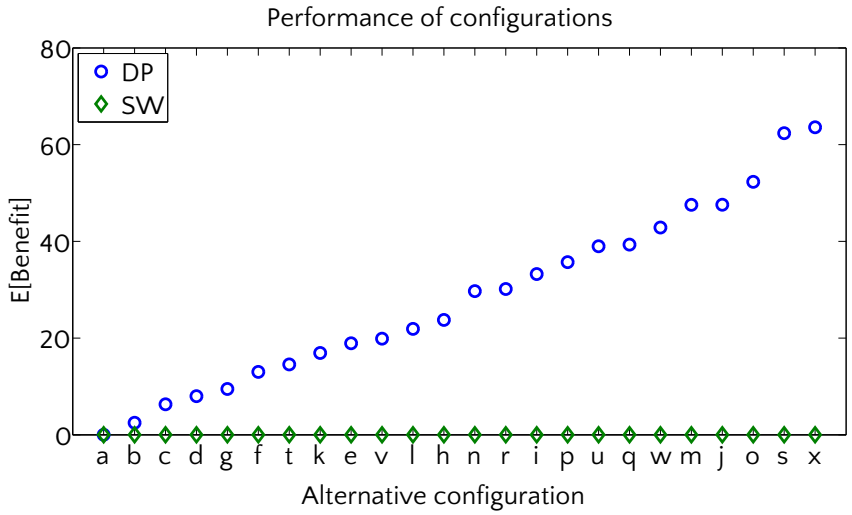


Figure 6.16: *Expected benefits per request in case of asymmetric scenario; comparison between static SW (static workflow) and DP (dynamic programming) for different configurations.*

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 a higher number of concrete service alternatives closer to the end of the sequential workflow (Figures 6.15 and 6.16).
- The variability of the response-times may have significant impact to the benefits achieved. When the end-to-end deadline is in jeopardy, it may be better to select a more expensive service with a smaller response-time variability (and smaller mean) than less expensive one with large response-time variability (Figure 6.16).
- In general it is better to have more response-time versatility with respect to mean and variance, (Figure 6.17). However, this needs to be investigated further.

6.6 Discussion

In this work, we have developed a model to maximize benefit for composite services by on-the-fly dynamic service selection. The selection decisions are based

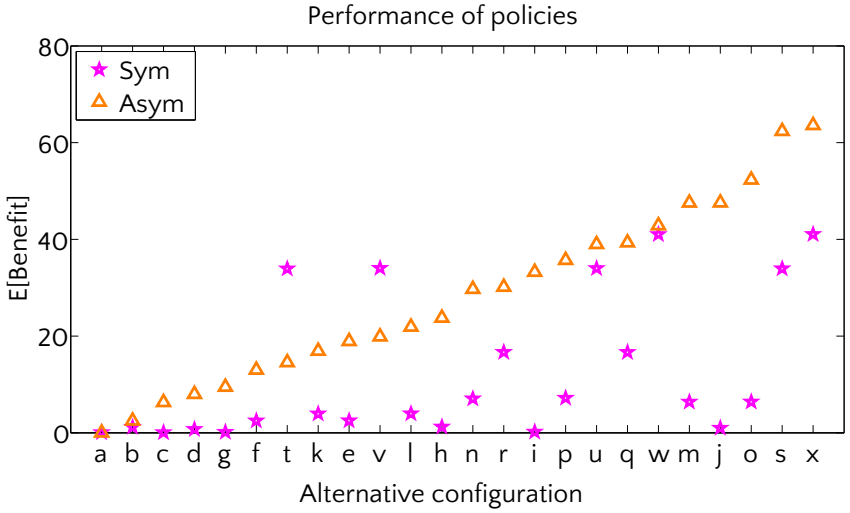


Figure 6.17: *Benefit comparison between symmetric and asymmetric scenarios.*

on observed response times, the response-time characteristics of the alternative, the end-to-end response-time objectives, and the reward and penalty parameters. The results not only indicate *that* there is enormous potential gain compared to other, non-dynamic approaches, but also show *how* one can realize such gain. We believe that this work is a significant step in realizing cost-efficient provisioning of complex composite services.

We end this chapter with a number of challenging areas for follow-up research.

First, in this chapter we have considered the case when SLAs are time invariant, i.e. once established, response-time SLOs, represented by respective PDFs, do not change (see also Remark 6.3.1). However, in practice the response-time distributions may be subject to change over time (e.g., due temporary failure or overload of a concrete service). Moreover, the empirical PDFs are typically determined based on response-time measurements over a finite time horizon, and hence will change over time. In this context, an important next step is to devise and implement models and methods to support *closed-loop control*, where the lookup table is re-calculated in an optimal way (e.g., with respect to the frequency of updates). Note that the results presented in this chapter form an excellent basis for extension towards closed-loop controlled system.

Second, the DP solution may tend to slow down when number of services is extremely large. Therefore, to provide good service quality for complex composite services composed of many services, there is a need for the development of fast yet efficient heuristic solutions, which opens up a challenging area for further research.

Third, in this model a specific reward-and-penalty cost structure was assumed, where the CSP pays an amount of money to the ISP for the execution of a single request, and gets a reward for execution of a single request within penalty deadline but pays penalty to the end customer when the agreed end -to-end deadline is not met. In practice, many other cost structures (e.g., discount structures) are conceivable. Extension of the results to include other cost structures is an interesting area for follow-up research.

Finally, another interesting and practically useful extension of the model is to include the possibility of re-attempts when the response-time of a given individual service exceeds some threshold value. Such reattempts may be particularly beneficial when the response-time distribution has a decreasing hazard rate of failure. Investigation of the potential for cost reduction and the cost-benefit trade-off of reattempts is another promising venue for further research.

Appendix 6.A Workflow aggregation example

We illustrate some basic aggregation rules using an example workflow (represented by Figure 6.18) 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 analysed in many papers (e.g., [96, 29, 66, 63, 118]). However, none of these papers considered stochastic QoS models, except [96], in which the authors calculate the composite service QoS parameters using Monte-Carlo simulations.

The workflow in Figure 6.18 consists of four elementary but frequently used workflow composition patterns, namely sequential, flow, switch and loop. There are two alternatives for tasks 1, 2 and 5, three alternatives for task 3, and tasks 4 and 6 have one alternative. The flow pattern represents (part of a) workflow in which all tasks are executed in parallel, and the response is not available till all services finish their execution (tasks 2 and 3 at Figure 6.18). The switch statement represents workflow that executes one of the tasks with given probabilities. Referring back to Figure 6.18 we identify the switch pattern for tasks 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 task (service 6 in Figure 6.18). Each 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 task i within the given workflow, the PDF and CDF are $f^{(i,j)}(t)$ and $F^{(i,j)}(t)$, respectively. The execution cost in this case is $c^{(i,j)}$, and response-time random

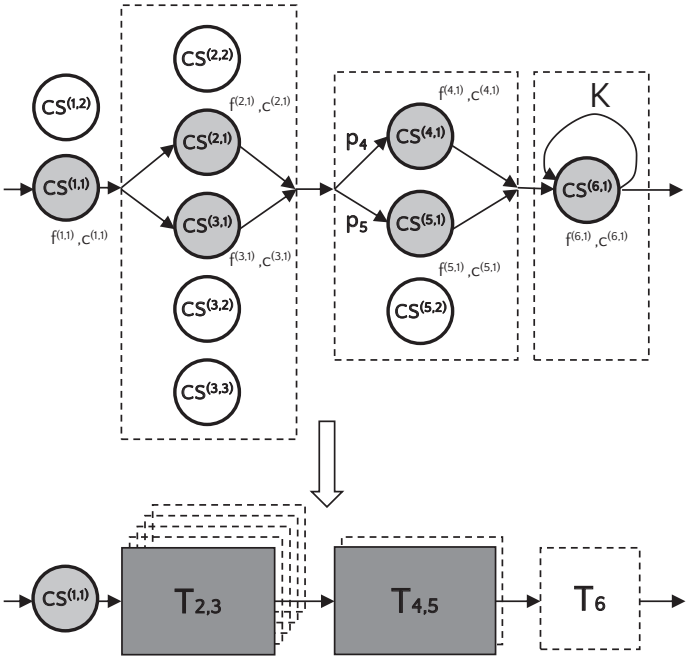


Figure 6.18: *Example workflow reduced and aggregated into the sequential workflow.*

variable is $D^{(i,j)}$. The aggregation rules for concrete services depicted in Figure 6.18 are as follows:

- The resulting response-time PDF for the flow pattern is expressed by the term $f^{T_{2,3}}(t) = F^{(2,1)}(t) \cdot f^{(3,1)}(t) + f^{(2,1)}(t) \cdot F^{(3,1)}(t)$. The execution cost is given as $c^{T_{2,3}} = c^{(2,1)} + c^{(3,1)}$.
- The resulting response-time PDF for the switch pattern (services 4 and 5) is given as $f^{T_{4,5}}(t) = p_4 \cdot f^{(4,1)}(t) + p_5 \cdot f^{(5,1)}(t)$. The execution cost is $c^{T_{4,5}} = p_4 \cdot c^{(4,1)} + p_5 \cdot c^{(5,1)}$.
- The resulting response-time PDF of the loop pattern is expressed as the K-fold convolution of the PDF $f^{(6,1)}(t)$, i.e. $f^{T_6}(t) = f^{(6,1)}(t) * f^{(6,1)}(t) * \cdots * f^{(6,1)}(t) = \left[f^{(6,1)}(t)\right]^K$, where $*$ represents convolution operator. The CDF $F^{T_6}(t)$ is calculated similarly, and the execution cost is $K \cdot c^{(6,1)}$.

For the case where aggregation of the tasks with multiple alternatives the aggregation should take place for each combination of the alternatives for considered tasks. Therefore, aggregation of tasks 2 and 3, $T_{2,3}$ has 6 alternatives, $T_{4,5}$ 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 [34].