An integer programming-based algorithm for optimising the WS-BPEL scenario execution adaptation process

Dionisis Margaris*

Department of Informatics and Telecommunications,
University of Athens,
Athens, Greece
Email: margaris@di.uoa.gr
*Corresponding author

Dimitris Spiliotopoulos,
Apostolos Kardiasmenos and
Dimitrios Pantazopoulos

Department of Informatics and Telecommunications,
University of the Peloponnese,
Tripoli, Greece
Email: dspiliot@uop.gr
Email: kardiasm@gmail.com
Email: d.pantazopoulos@yahoo.gr

Abstract: In this work we present an integer programming-based algorithm for adapting the execution of WS-BPEL scenarios, through the dynamic selection of the services to be invoked, according to criteria and policies set by the user. The proposed algorithm is experimentally evaluated both in terms of adaptation quality and adaptation computation overhead. The experimental results demonstrate that the proposed approach achieves to considerably improve adaptation speed, as compared to the exhaustive search algorithm which is considered as a baseline, while at the same time maintaining adaptation quality.

Keywords: web services; WS-BPEL; business processes; personalisation; adaptation; integer programming; quality of service; QoS; evaluation.

Reference to this paper should be made as follows: Margaris, D., Spiliotopoulos, D., Kardiasmenos, A. and Pantazopoulos, D. (2020) 'An integer programming-based algorithm for optimising the WS-BPEL scenario execution adaptation process', *Int. J. Web Engineering and Technology*, Vol. 15, No. 3, pp.307–332.

Biographical notes: Dionisis Margaris is a Postdoc Researcher at the Department of Informatics and Telecommunications at the University of Athens and an Adjunct Lecturer in the Department of Digital Systems at the University of the Peloponnese, in the area of Information Systems. He has received his degree, his Master's degree and his PhD from the Department of Informatics and Telecommunications at the University of Athens. He has published more than 50 papers in international journals, conferences and books and his research interests include recommender systems, information systems, computer programming, databases systems and social networks.

Dimitris Spiliotopoulos received his BSc, MA, and MPhil degrees from the University of Manchester, UK, and PhD degree from the Department of Informatics and Telecommunications, University of Athens, Greece, in 2009. He works in information systems at the Department of Informatics and Telecommunications, University of Peloponnese, Greece. His research interests include information systems, social media analysis, usability, accessibility, and recommender systems. He has served as a PC member of many international journals and conferences, while he has also participated in more than ten international and national projects.

Apostolos Kardiasmenos has been working in the banking sector and in particular in the field of electronic payment systems for over a decade as a developer/analyst. He has acquired his Bachelor's degree in Informatics and Telecommunications from the University of Peloponnese and a Master's degree in the field of computational linguistics in an inter-university course between the national and Kapodistrian University of Athens and the National Technical University of Athens. His main area of interest covers programming and administration of online transaction processing and switching systems, transaction key management applications and online transaction fraud prevention and detection solutions.

Dimitrios Pantazopoulos is a secondary education school teacher in the area of Mathematics, also has degree in Informatics and Telecommunications from the University of Peloponnese and postgraduate degree in Information Systems from the Hellenic Open University. He also has extensive experience in mobile communication networks as he was held different positions in big telecommunications companies. His research interests include information systems, probabilities, stochastic processes, cryptography and educational software.

1 Introduction

Contemporary web services (WSs) are distributed applications where a single user can perform complex computing processes, without needing to consider the technological details on the part of the service provider and Web Services Business Process Execution Language (WS-BPEL) is the standard language for designing and executing business processes that encompasses the execution of multiple WSs (Fu et al., 2004; Hallwyl et al., 2010). The communication specifications of such applications are usually based on widespread and common protocols, such as HTTP for transport and JSON&XML and its derivatives for parameter encoding, and the execution of the application is routed through a distributed execution engine available to the single user. These widespread protocols allow system designers to organise individual services in order to build higher-level business services. For example, users who plan their vacations abroad and visit the website of a travel agency, are required to fill in the appropriate forms for the destination of the trip and the means of transport of their choice (airplane, boat, bus, etc.), they may select the category of the hotel they would like to stay in (e.g., a three-star hotel) and any venues they would like to attend, such as a football or basketball match, visiting a museum or attending a concert or theatre performance; then, after a user submits the request, the software of the travel agency communicates with each of the information systems that perform individual purchases (e.g., airline company, hotel, theatre, etc.) to complete the reservations designated by the user.

The procedure described above necessitates that the WSs that will be invoked in the WS-BPEL scenario are known in advance. However, in the modern internet, an individual service (e.g., booking a flight or checking for available rooms in a hotel) is typically offered by multiple providers, and each provider makes a service available under (generally) different service quality parameters (quality of service – QoS), such as cost, response time, reliability and so forth (O'Sullivan et al., 2002). In this context, it is highly desirable for users to be able to specify qualitative criteria that should be fulfilled by the service they receive; and subsequently, the execution engine should adapt the realisation of the service provision to each user, by selecting WSs that conform to the QoS specifications set by the user.

For example, in a holiday planning WS-BPEL scenario such as the one described above, the user may request an airline recommendation (limiting thus the choice of transportation means to air travel only), while at the same time requesting a reservation at a particular hotel *H* and also indicating that she wants to attend an art performance; and in the QoS domain, the user may specify that the price of the flight is 'low', whereas the quality of the art performance is 'high', with the total price of the trip being 'moderate'. Then the system should find WSs realising the requested functionalities (air ticket purchase, art performance ticket purchase), subject to the restrictions designated by the user ('low' air fare, 'high' performance quality), combine them with the (rigidly specified) reservation to hotel *H*, and ascertain that the cost of the whole combination is 'moderate'.

A straightforward approach to implementing the identification of the service combination that most closely matches the user-designated functionality and QoS specification would be to generate all possible combinations that the available WSs can form and compute a fitness score for each one (Margaris et al., 2013), under an exhaustive search (ES) approach. However, in real-world scenarios, the number of WSs that realise each functionality is high, hence the cost of the generation of all possible service combinations is prohibitively high. The solution to this problem can be given by using integer programming (IP) techniques (Schrijver, 1998), which are widely used for optimising constraint problems, by restricting some or all of the objective function's variables to be integers.

In this work, we present an IP-based (Schrijver, 1998) recommendation algorithm for optimising the WS-BPEL scenario adaptation process that accommodates QoS criteria set by users/clients. The main objective is to create an algorithm that can efficiently compute the adaptations that satisfy the QoS criteria set by users, while at the same time maintaining the optimality of the computed adaptations, i.e., guaranteeing that the computed adaptations are the optimal ones, under the specified QoS restrictions.

The rest of the paper is structured as follows: Section 2 overviews related work. Section 3 presents the necessary foundations for our work, which include QoS concepts, subsumption relations and IP. Section 4 presents the proposed IP algorithm, while for self-containment purposes we briefly present the ES algorithm introduced in Margaris et al. (2013), which also performs QoS-based adaptation of WS-BPEL scenario execution. The ES algorithm is used as a baseline in the experimental evaluation. In Section 5 we report on experiments conducted to assess performance of the proposed technique, both in terms of efficiency (the time needed to compute the requested recommendations) and effectiveness (adaptation quality), aiming to substantiate the utility of the presented algorithm and its practical applicability. Finally, Section 6 concludes the paper and outlines the future work.

2 Related work

Nowadays, information systems in the internet are very frequently implemented by composing WSs offered by individual providers, since the adaptation process of WS scenarios is a topic that has attracted considerable research efforts over the last years, ranging from QoS (de)composition and optimisation to dynamic prediction and uncertainty, and from exception resolution to fault-tolerance approaches. In the next paragraphs an overview of works in the aforementioned areas is presented.

Several works exist which aim at optimising WS composition process. Chen et al. (2016) present a QoS-aware WS composition method by multi-objective optimisation to assist users make variable decisions. The problem of QoS-aware WS composition is formulated to a multi-objective optimisation model where the individual optimisation objective is either QoS performance or QoS risk, while an efficient e-dominance multi-objective evolutionary algorithm is developed to solve the presented model. Rodriguez-Mier et al. (2016) present a theoretical analysis of graph-based service composition utilising its dependency with service discovery. Their work defines a composition framework integrating fine-grained I/O service discovery that enables graph-based composition generation. The composition includes the semantically relevant set of services for input-output requests. Their proposed framework also presents an optimal composition search algorithm for extracting the optimal composition from the graph by minimising the number and the size of services, as well as several graph optimisations for improved system scalability. The approach proposed by Xia et al. (2011) allows for specification of QoS constraints, which drive the adaptation process; the adaptation process aims to minimise an objective function for the entire web service orchestration (WSO), employing either heuristic (OPTIM_HWEIGHT) or brute force (OPTIM_S) algorithms. Hammas et al. (2015) propose a WS architecture supporting global QoS optimisation, as well as dynamic WS composition. They also propose an ant colony optimisation-based algorithm for QoS aware WS selection. Liang et al. (2019) focus on WS composition for internal complementarity likelihood cases. More specifically, they address the problem of QoS-aware web service selection with internal complementarity (WSS-IC). This is achieved by transforming this problem into a multi-dimensional multi-choice knapsack problem (MMKP), proving that their proposed transformation has non-polynomial time complexity. They demonstrate that existing approaches to MMKPs are not computationally feasible to resolve QoS-aware WSS-IC by performing complexity analysis. Finally, they propose an iteratively improving framework for deriving the solution, iteration by iteration, while taking into account both solution structure and QoS constraints, where, at each iteration, the current solution gets improved by solving a disjunctively constrained knapsack problem. Margaris et al. (2013) perform a horizontal QoS-based adaptation, supporting both the sequential and parallel execution structures within the BPEL scenario. However, this approach uses very limited QoS-based criteria (optionally specifying lower and upper bounds for each QoS attribute), hence running the risk of inferior overall QoS of the formulated solutions when compared to the possibly attainable optimal composition QoS, especially for the cases where CF has to address known issues such as grey sheep and cold start. Lu and Xu

(2017) propose and implement a web-based system that utilises distributed knowledge for intelligent service composition and adaptive resource planning. Gupta et al. (2018b) present a literature review of different tools and approaches, which are available for WSO. Furthermore, they compare earlier WSO approaches, taking into account different benchmarks, and identify the needs of the improvements. Machorro-Cano et al. (2020) propose the design of a language for internet-of-things (IoT) WSs composition, which considers the WSO and choreography in order to define business processes in a supply chain in the healthcare domain, while at the same time discuss important literature on business processes, IoT WSO, IoT WSs choreography and IoT WSs coordination. Furthermore, they propose the design of a language for IoT WS composition, presenting the expressivity of the language, the WSs orchestrating process and choreography characteristics. Gupta et al. (2018a) propose a QoS-based fault-detection and fault-tolerance approach which uses the dynamic WSO. Their approach also considers the users' preferences and WSs' QoS at runtime. Their approach uses reliability in two phases: in the first phase a trust-based WS filtering mechanism is used, achieving reliability at component level before a fault occurs, while in the second phase, a decision for dynamic recovery is taken, whenever a fault is detected in a process.

Zeng et al. (2004) present a QoS-aware middleware platform, namely AgFlow, which supports quality driven WS compositions, by maximising user satisfaction expressed as utility functions over QoS attributes, while at the same time satisfying the constraints set by both the user and the structure of the composite WS. In this work, the overall execution time of each candidate solution is computed using the critical path algorithm, an approach that limits the amount of optimisation that can be achieved through the use of IP techniques and thus resulting in increased adaptation computation overheads. Cardellini et al. (2017) present a software platform supporting QoS-driven adaptation of WS-oriented systems, named MOSES, which integrates within a unified framework different adaptation mechanism, achieving a greater flexibility in facing various operating environments and the possibly conflicting QoS requirements of several concurrent users. The per flow runtime adaptation presented in this work and detailed in Cardellini et al. (2012) also employs a linear programming model to compute the optimal adaptation, the model adopted in MOSES however groups users into *service classes*, where for each user class the importance of different QoS attributes is pre-determined and spans across all WS-BPEL processes invoked by users within this class. This arrangement allows statistics from previous executions to be used for future adaptations, but on the other hand limits the flexibility available to users, who may want vary QoS attribute importance settings according to individual WS-BPEL scenario invocation needs, e.g., maximise cost importance for one invocation, while maximising reliability importance for another invocation.

In order to model services with dynamically changed attributes and uncertain effects, Wang et al. (2016) propose an approach that introduces branch structures into composite solutions to cope with uncertainty in the service composition process. Liu et al. (2015) propose a WS composition method based on QoS dynamic prediction and global QoS constraints decomposition. Their approach includes two critical phases. The first phase happens before service composition, by decomposing global QoS constraints into local constraints, thereby transforming the WS dynamic composition problem to a local optimisation one. The second phase happens at runtime, using predicted QoS values to select the optimal WS for the current abstract service. Zhang and Lyu (2017) present a WS search engine, namely WSExpress that is able to find the desired WS. WSExpress ranks the publicly available WS by both functional similarities to user queries and non-functional QoS characteristics of the WSs. Furthermore, WSExpress can automatically replace a failed WS with a suitable one. This work employs a greedy algorithm to determine the adaptation that best fits the optimisation goal, and therefore the approach proposed therein is prone to calculating suboptimal solutions, due to the presence of local maxima in the search space. Zatout et al. (2018) present an architecture for the adaptation and monitoring of WS composition using a set of components that cooperate in order to ensure the proper operation of WSO; the presented architecture is based on the dynamic selection of alternative WSs. Furthermore, they provide a case study of Bookstore process to illustrate their proposal and discuss some implementation results, with mainly focusing on the throughput and response time attributes.

Some works in the area explicitly address the reliability/fault-tolerance aspects that are critical in a number of domains, given the distributed and error-prone nature of the internet. Kareliotis et al. (2009) present the ASOB middleware, which integrates QoS-based adaptation with exception resolution. The presented middleware intercepts WS invocation failures and distinguishes business system faults from logic faults, remedying the first category through replacing failed WSs by 'next best' solutions; exceptions stemming from business logic faults are left to the WS-BPEL scenario designer to resolve through appropriate handlers, since they cannot be automatically addressed. Song and Tilevich (2019) enhance compiler-generated execution WSO with equivalence to efficiently increase reliability. They also introduce a dataflow-based domain-specific language, whose dataflow specifications include the implicit declarations of equivalent micro-services, as well as their execution patterns. Furthermore, their work introduces new equivalence workflow constructs, in order to automatically generate reliable workflows and execute them efficiently.

The approach proposed in this paper extends the state-of-the-art in the WS-BPEL scenario execution adaptation by providing collectively the following features:

- formulating the task of finding an optimal adaptation as a multicriteria IP optimisation problem, which can be efficiently solved using standard IP problem solver software
- supporting both sequential and parallel service compositions within the WS-BPEL scenario
- considering multiple QoS dimension in an extensible fashion, allowing the accommodation of additional QoS dimensions as needed
- providing flexibility for users to select the WS-BPEL scenario adaptation goals at a per-invocation granularity.

However, none of the aforementioned works present an IP recommendation algorithm that takes into account QoS criteria set by users, aiming at both minimising the adaptation formulation time and maximising the recommendation adaptation quality.

3 Background on QoS attributes, subsumption relationships and IP concepts

The following sub-sections include a summary of the concepts and underpinnings from the domains of WS QoS attributes, WS subsumption relationships and IP that are used in our work.

3.1 WSs QoS attributes

WS QoS attributes are quantities that are able to describe measurements of the overall performance of a WS. Gouscos et al. (2003) define QoS attributes for service latency (time elapsing between invocation and receiving of the reply) and throughput (number of requests that are executed per time unit), while O'Sullivan et al. (2002) and Comerio et al. (2007) consider invocation cost (monetary) and reliability (the degree of being capable of maintaining the service and service quality, measured, e.g., as number of failures per time unit). For conciseness purposes and without loss of generality, in this paper we will consider only the attributes *cost* (c), *response time* (rt) and *reliability* (rel), adopting their definitions from O'Sullivan et al. (2002) and Comerio et al. (2007). Lower-level QoS aspects, e.g., network speed, reliability and latency, are taken indirectly into account, since they reflect on higher-level aspects, e.g., response time (affected by network speed and latency) and service reliability (affected by the network-level reliability). The detailed investigation of the interplay between network-level QoS aspects and service-level QoS aspects is considered part of our future work.

WS-BPEL is the standard language for designing and executing business processes that encompass the execution of multiple WSs. In the context of a WS-BPEL scenario execution, multiple WSs, s_1, s_2, …, s_n are invoked. A user may want to specify constraints on the QoS delivered by each individual WS executed in the context of a specific invocation; to this end, the user may provide QoS specifications regarding the upper and lower bounds for each QoS attribute, for each of the services s_j. In other words, for a s_j service included in an WS-BPEL scenario, the user may provide the following two vectors:

- $\mathrm{MIN}_j(\mathrm{min}_{rt,j}, \mathrm{min}_{c,j}, \mathrm{min}_{rel,j})$
- $\mathrm{MAX}_j(\mathrm{max}_{rt,j}, \mathrm{max}_{c,j}, \mathrm{max}_{rel,j})$.

where vector MIN_j designates the lower bounds of the QoS attributes for the service implementation that will be selected to realise the functionality of s_j (e.g., booking a hotel), while vector MAX_j designates the corresponding upper bounds. In addition, the user provides a weight vector $W(rt_w, c_w, rel_w)$ that indicates the importance of each attribute in the context of the particular invocation of the WS-BPEL scenario. Effectively, the weight factor value is multiplied by the value of the corresponding QoS attribute, and these products are then aggregated in order to produce an overall score for the scenario. If the invocation request does not include a QoS attribute weight specification, then all attributes are considered of equal weight, i.e., the weight of each QoS attribute is set to $1/\mathrm{N_q}$, where $\mathrm{N_q}$ denotes the number of qualitative attributes considered. If, for a service, the minimum value for some qualitative attribute is not provided, then the lower bound is set to 0. Correspondingly, if the maximum value for some qualitative attribute is not provided, the upper bound is set to 1. Note that while the upper and lower bound vectors apply to each individual service s_i, the weights apply *to the whole composition*, rather than to individual services, since the weight reflects the importance attached to each QoS characteristic for the process as a whole, and not to its constituents.

As far as attribute values are concerned, clearly different attributes may have different measurement units (e.g., requests/month for throughput vs. euros for cost), and different magnitudes (e.g., the requests/month attribute may scale to a range of billions, while a price may be in the range of thousands). In order to have meaningful results from combining such attribute values, we consider that all attributes are normalised in the range [0, 1], through the application of a standard normalisation formula (He and Wu, 2008):

$$norm(v) = \frac{v - \min(v)}{\max(v) - \min(v)} \tag{1}$$

where v is the value to be normalised, while $\min(v)$ and $\max(v)$ are the minimum and maximum values of the corresponding attribute in the database extension. Furthermore, we consider that larger values correspond to higher QoS levels: for attributes where the inverse holds (such as price and latency), it is straightforward to transform their values to a 'higher is better' scheme, by altering the normalisation formula above to

$$norm(v) = 1 - \frac{v - \min(v)}{\max(v) - \min(v)} \tag{2}$$

This approach is typically followed for the handling of QoS attribute values in the context of WS selection and composition (Halfaoui et al., 2015).

WS-BPEL scenarios compose multiple individual WSs to composite business processes, which are themselves (composite) WSs. The execution of individual WSs may proceed sequentially (i.e., serially, where the execution of the next service commences when the execution of the previous one has concluded) or in parallel (services are executed concurrently). To calculate the quality of composite WSs consisting of individual WSs $ws_1, \ldots, ws_n$, which are executed sequentially or in parallel and have QoS features equal to $(rt_1, c_1, rel_1), \ldots, (rt_n, c_n, rel_n)$, respectively, the formulas outlined in Table 1 (Canfora et al., 2005) are employed.

Table 1 Computation of composite services QoS

	QoS attribute		
	Response time	*Cost*	*Reliability*
Serial composition	$\sum_{i=1}^{n} rt_i$	$\sum_{i=1}^{n} c_i$	$\prod_{i=1}^{n} rel_i$
Parallel composition	$\max_i(rt_i)$	$\sum_{i=1}^{n} c_i$	$\prod_{i=1}^{n} rel_i$

As can be seen in Table 1, the response time of a serial structure is equal to the sum of the response time of its individual components, while the response time of a parallel structure is equal to the maximum value of its individual components. This difference is important for the subsumption process, as different search strategies should be used to optimally tailor the script to the user's QoS specifications.

Table 2 Example of repository contents.

Service	*Response time*	*Cost*	*Reliability*
A_1	0.6	0.6	0.6
A_2	0.8	0.8	0.8
B_1	0.2	0.2	0.2
B_2	0.7	0.7	0.7

For example, let us consider a WS-BPEL scenario involving the sequential execution of services *A* and *B*. For an execution of this scenario, the weight vector for QoS features has been set to *W* = (1, 1, 0), while at the time of the WS-BPEL scenario execution, the services listed in Table 2 are known. Then, the adaptation engine should select services A_2, B_2, with a score of 2, since for these services we have:

$$score = \left(sum\left(rt_{A_2}, rt_{B_2}\right) * 1 + sum\left(c_{A_2}, c_{B_2}\right) * 1\right) = 2 \tag{3}$$

It is clear that this score is higher than any other combination. In a scenario, however, involving parallel execution, the executing engine should select services A_2 and B_1, since they provide an overall score equal to

$$score_2 = \left(\max\left(rt_{A_2}, rt_{B_2}\right) * 1 + sum\left(c_{A_2}, c_{B_2}\right) * 1\right) = 1.7 \tag{4}$$

which is higher than the score of 1.3 achieved by the combination (A_2, B_2).

3.2 Subsumption relationships

When a WS-BPEL scenario is adopted to match the QoS designations provided by the user, the execution adaptation process selects WS service implementations to realise functionalities that appear in the context of the WS-BPEL scenario. Therefore, the adaptation software needs to be able to test whether within a particular execution, a WS service implementation *A* can realise some requested functionality *B*. In this work, we use the service matchmaking relationships known as *subsumption relationships* (Bellur and Kulkarni, 2007) to address this issue. More specifically, the subsumption relationships used and the rules defining which subsumption relationships dictate service replaceability, are as follows:

1. *A exact B*, if and only if *A* provides the same functionality as *B*. Clearly in this case service *A* can be used to deliver the functionality *B* requested in the WS-BPEL scenario.
2. *A plug-in B*, if and only if *A* provides a more specific function than *B*. For example, service *B* could belong in the 'travel' category, whereas *A* could belong to the more specific 'air travel' (sub)category. In this case, whenever the functionality of *B* is required, the functionality of *A* could be used, since it provides a specialisation of the functionality provided by *B*.
3. *A subsume B*, if and only if *A* provides more generic functionality than *B*. In this case *A* cannot be directly used whenever *B*'s functionality is required. For example, if B corresponds to the 'sea travel' functionality, while *A* implements a 'trip' functionality, then we cannot (always) use service A to realise functionality B, as

service A may only realise 'air travel' and 'car travel', failing thus to deliver the requested 'sea travel'. We note here that in some cases service *A could be used to realise functionality B*, however this cannot be determined automatically by the adaptation algorithm.

4 *A fail B*, in all other cases. When A fail B holds, A cannot be used to deliver the functionality B.

For more information on subsumption relationships, the interested reader is referred to (Bellur and Kulkarni, 2007).

3.3 Integer programming

IP is a methodology for optimising problems where integer values are assigned to decision variables (Schrijver, 1998). IP programming is a sub-domain of linear programming where decision variables may be assigned with generic real values. The additional requirement for computation of integer solutions only makes IP problems harder to solve than linear programming problems, increasing their complexity; however, a number of optimisation techniques which limit the solution search space (Bixby et al., 2004) render IP problem solution computation highly efficient. Furthermore, IP solvers employ a multitude of techniques to speed up the computation of solutions to IP problems. Indicatively, such techniques include cutting planes, presolve, branching rules and branch and bound methods, heuristics, node presolve and probing on dives (Bixby et al., 2004), with each speed up factor providing a speed up ranging from 53.7% (cutting planes) to 1.1% (probing on dives). The potential to apply individual methods and explore the effectiveness of each of them on the speed and quality of the solution computation will be considered in the context of our future work.

IP problems involve a set of decision variables, $x_1, x_2, \ldots, x_n$, the values of which need to be computed, in order to maximise (or minimise) an *objective function* which takes the following form:

$$c_1x_1 + c_2x_2 + \ldots + c_nx_n \tag{5}$$

where $c_1, c_2, \ldots, c_n$ are problem-specific coefficients. The solution, which effectively resolves to a set of value assignments to decision variables ($x_1 = v_1, x_2 = v_2, \ldots, x_n = v_n$) where $v_i \in Z$, $\forall_i$: $1 \leq i \leq n$, may be subject to constraints on the values assigned to variables, which take the following form:

$$\begin{aligned} &a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n relOp_1 b_1 \\ &a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n relOp_2 b_2 \\ &\ldots \\ &a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n relOp_m b_m \end{aligned} \tag{6}$$

where a_{ij}, b_i are problem-specific coefficients and $relOp_i$ is a relational operator ($>, \geq=, =, \leq, <, \neq$), $\forall_i$: $1 \leq i \leq n$, $\forall_j$: $1 \leq j \leq m$.

The formulation of the WS-BPEL scenario adaptation problem as an IP problem and its solution is presented and analysed in the following section.

4 The IP algorithm

The proposed algorithm utilises the information described in Sections 3.1 and 3.2 and formulates an invocation-specific IP problem, the solution of which corresponds to the desired adaptation of the adaptation of the execution of the WS-BPEL process to match the QoS specifications designated by the user. The steps taken by the algorithm are as follows:

1 Firstly, the algorithm extracts the functionalities for which recommendations are requested for.

2 Subsequently, for each such functionality, it retrieves the WSs that are candidate to be used for the realisation of this functionality in the context of the particular WS-BPEL scenario invocation.

3 Finally, it formulates and solves the IP problem, determining the optimal WS-BPEL scenario adaptation for the particular scenario invocation request. The QoS specifications given by the user are effectively used as problem-specific coefficients of the IP problem (c.f. Section 3.3).

In the Section 4.3, we will detail the three steps of the proposed IP algorithm. To illustrate the proposed approach, the case of adapting a travel planning business process will be used; the example WS-BPEL scenario, together with the dataset that will be used in the example, are described in Section 4.1. For completeness, in Section 4.2 we present the exhaustive adaptation algorithm, which every possible service combination, calculates the overall QoS of the corresponding composition, and finally selects as a solution the service combination providing the highest overall QoS (Margaris et al., 2013); the exhaustive algorithm is used as a baseline in the experimental evaluation, presented in Section 5.

4.1 Example WS-BPEL scenario and dataset

To exemplify the proposed approach, we will use the case of adapting the invocation of a simple WS-BPEL scenario. This scenario describes a business process of booking a trip (modelled as a WS-BPEL scenario *BookTrip*), which includes the functionalities of finding an airline ticket, booking a luxury hotel room and attending an athletic event. The booking of air tickets is performed first, and afterwards the hotel reservation and the athletic event ticket purchase proceed in parallel, as indicated by the SEQ and FLOW constructs in Figure 1.

Figure 1 Business process execution request example

```
BookTrip
    SEQ
        (name=bookAirTravel, operation=AirTravel)
        FLOW
            (name=bookLuxury, operation=Luxury)
(name=bookAthletic, operation=Athletic)
```

In our example, we consider a particular scenario invocation where the QoS weights and the upper and lower bounds for the QoS attributes for each functionality are as shown in Table 3; the respective specifications in the form of an XML file are depicted in Figure 2. In Table 3 we can observe that the user does not impose any restrictions about the minimum or maximum response time for book flight (*bookAirTravel*), however s/he sets minimum and maximum bounds of 0.29 and 0.89, respectively, for the cost of the particular operation. Similarly, the user requests that the reliability of the *bookAirTravel* operation would not be less than 0.45, while no restrictions are imposed for the maximum value of reliability. Analogously, constraints are defined for the other two services (*bookLuxury* and *bookAthletic*).

Table 3 Specifications of QoS weights and bounds

QoSWeights = (rt: 0.2, c: 0.3, rel: 0.5)	
Functionality	*QoS bounds vectors*
bookAirTravel	$\text{Min}_{\text{bookAirTravel}}$ = (rt: null, c: 0.29, rel: 0.45)
	$\text{Max}_{\text{bookAirTravel}}$ = (rt: null, c: 0.89, rel: null)
bookLuxury	$\text{Min}_{\text{bookLuxury}}$ = (rt: null, c: 0.51, rel: 0.41)
	$\text{Max}_{\text{bookLuxury}}$ = (rt: null, c: 0.98, rel: null)
bookAthletic	$\text{Min}_{\text{bookAthletic}}$ = (rt: null, c: null, rel: 0.31)
	$\text{Max}_{\text{bookAthletic}}$ = (rt: null, c: 0.9, rel: null)

Figure 2 XML representation of the QoS weights and bounds

```
<?xml version="1.0" encoding="utf-8">
<invokeProcess target="BookTrip">
<QoSweightsrespTime="0.2" cost="0.3" reliability="0.5" />
<serviceInvocationQoS>
<invoke name="bookAirTravel" minCost="0.29"
            maxCost="0.89" minReliability="0.45" />
<invoke name="bookLuxury" minCost="0.51"
            minReliability="0.41" maxCost="0.98" />
<invoke name="bookAthletic" minReliability="0.31"
            maxCost="0.9"/>
</serviceInvocationQoS>
</invokeProcess>
```

In Figure 2, the *QoSWeights* entity corresponds to the weight vector that indicates the importance of each attribute in the context of the particular adaptation. Then, each WS-BPEL *invoke* construct is enhanced with attributes which convey the QoS minimum and maximum limits for the relevant QoS attributes of the individual services: for each invocation corresponding to an *invoke* construct, the service that will be selected to realise the particular functionality should adhere to these limits. Note that the QoS attribute weights have a *scenario-wide scope*, while the scope of the QoS attribute value limits is one single service invocation.

Figure 3 Hierarchy of services and service implementations for the travel planning WS-BPEL scenario (see online version for colours)

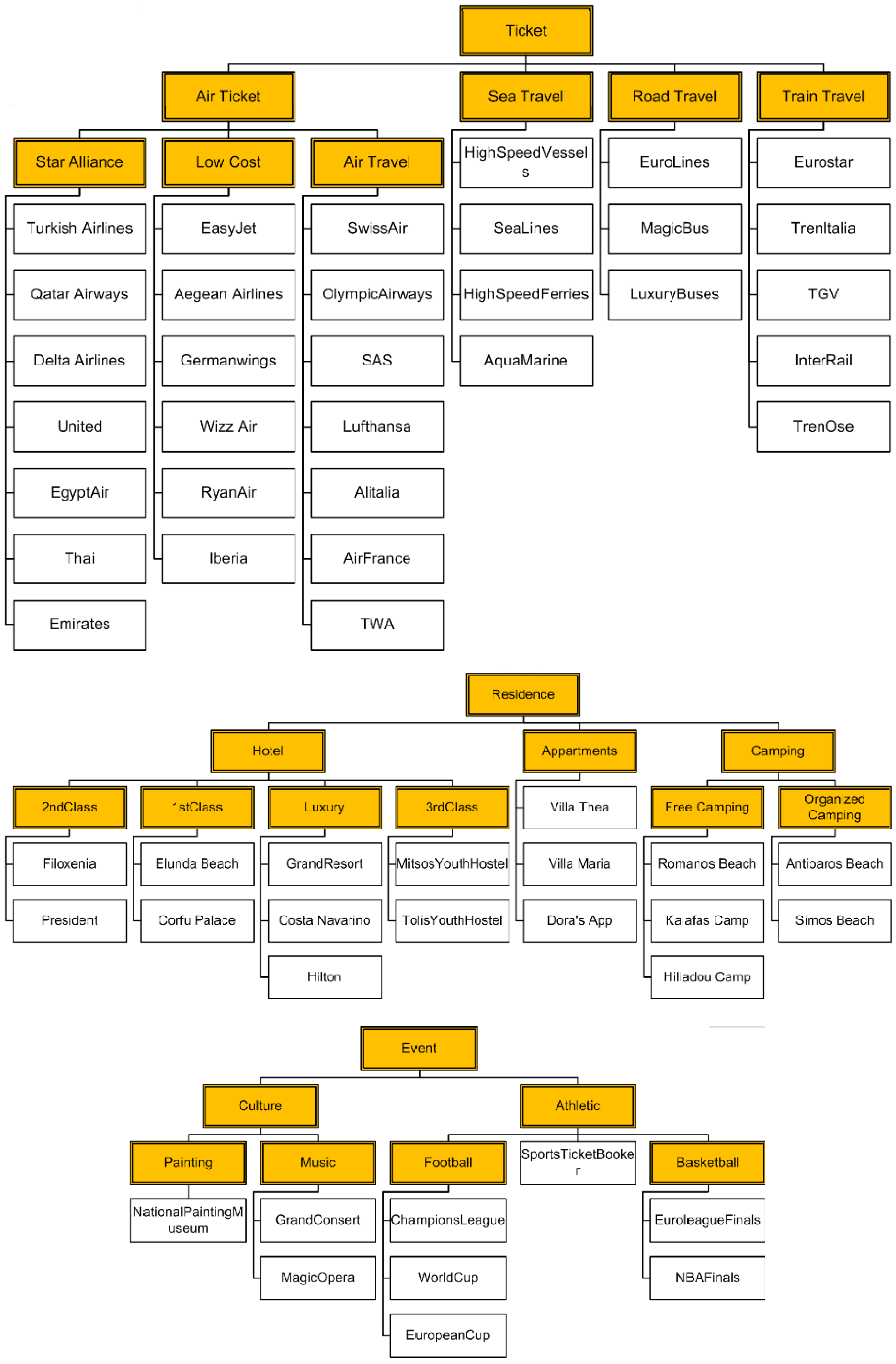


Figure 3 illustrates the taxonomy used in the travel planning WS-BPEL scenario (SoDa Lab, 2020). The taxonomy contains three first-level branches, namely ‘ticket’, ‘residence’ and ‘event’, which are displayed as separate trees in Figure 3 to promote readability. Nodes representing categories are represented as shaded rectangles with double-lined borders, while nodes corresponding to individual service implementations are denoted as clear rectangles with single-lined borders. A line between two rectangles denotes that the lower-level node belongs to the category represented by the higher-level node.

Each node corresponding to a service implementation bears information regarding the QoS aspects of the corresponding implementation. An excerpt of this information is depicted in Figure 4.

Figure 4 Excerpt of the service taxonomy extension

```
Category: Ticket
 Category: Air Ticket
  Category: Air Travel
Implementation: SwissAir (rt=0.75, cost=0.7, rel=0.8)
   Implementation: OlympicAirways (rt=0.8, cost=0.55, rel=0.78)
...
Category: Residence
 Category: Hotel
  Category: Luxury
Implementation: GrandResort (rt=0.91, cost=0.62, rel=0.61)
Implementation: Costa Navarino (rt=0.95, cost=0.92, rel=0.95)
   Implementation: Hilton (rt=0.98, cost=0.88, rel=0.39)
...
Category: Event
Category: Athletics
Category: Football
   Implementation: EuropeanCup (rt=0.75, cost=0.55, rel=0.9)
   Implementation: ChampionsLeague (rt=0.45, cost=0.94, rel=0.7)
Implementation: SportsTicketBooker (rt=0.65, cost=0.6, rel=0.8)
...
 Category: Culture
  Category: Music
Implementation: MagicOpera (rt=0.29, cost=0.54, rel=0.82)
```

4.2 The ES algorithm

The ES algorithm initially reads the BPEL scenario and identifies the respective branches of the taxonomy under which the service implementations which are candidate for use in the particular scenario adaptation are located. According to the rationale of the subsumption relationships, only nodes delivering the *exact* and *plug-in* functionalities, in relation to the one designated in the scenario, are eligible for use to deliver the functionality in the composition. Consequently, if a service category is specified in the scenario under adaptation, only service implementation nodes that are descendants (either direct or indirect) of the category within the taxonomy tree are selected; if a concrete service implementation is designated in the scenario, then the direct parent of the designated service implementation is first located, and subsequently its descendants (either direct or indirect) within the taxonomy tree are selected.

In our example, the realisation of the 'Luxury' service designated within the scenario can be performed by any of the services under the 'Luxury' category within the taxonomy tree of Figure 3 (Grand Resort; Costa Navarino; Hilton), whereas the realisation of the 'Athletic' service can be performed by either the 'SportsTicketBooker' service (a direct descendant of the 'Athletic' category within the taxonomy tree), or any of the descendants of the 'football' and 'basketball' service categories.

Once the candidate services delivering the required functionalities have been identified, they are filtered considering the QoS attribute limits designated by the user for the particular invocation. In our example, service 'Hilton' would be filtered out because it does not meet the minimum threshold for reliability specified by the user, and service 'ChampionsLeague' would be discarded because its cost exceeds the maximum cost threshold. After this step has concluded, for each of the functionalities f_i specified in the scenario under adaptation, a set of candidate functionality implementations C_{f_i} is formulated.

As stated above, the ES algorithm proceeds in formulating all possible service implementation combinations, considering service implementations that fulfil the user-provided thresholds. This is equivalent to the Cartesian product of the candidate functionality implementations sets C_{f_i}, i.e.:

$$allCombinations = C_{f_1} \times C_{f_2} \times \ldots \times C_{f_n}$$

where n is the number of functionalities within the scenario being adapted.

In our example, the set of all combinations would be

$$allCombinations = C_{AirTravel} \times C_{Luxury} \times C_{Athletic}.$$

Subsequently, for each of the combinations, the QoS attribute values of the compositions are computed, using the formulas presented in Section 3.1, and the overall score of the composition is calculated, by applying a weighted sum formula on the combination's QoS attribute values, where the weights for the QoS attributes are given by the *QoSweights* vector (c.f. Figure 2). Finally, the combination having the highest overall score is selected.

Table 4 QoS attribute value and overall score computation formulas for the trip booking scenario

Computation target	*Computation formula*
Response time (rt)	$rt_{AirTravel} + \max(rt_{Luxury}, rt_{Athletic})$
Cost (cost)	$cost_{AirTravel} + cost_{Luxury} + cost_{Athletic}$
Reliability (rel)	$rel_{AirTravel} \times rel_{Luxury} \times rel_{Athletic}$
Overall score	$0.2 \times rt + 0.3 \times cost + 0.5 \times rel$

Considering the trip booking example, Table 4 depicts the formulas used to calculate the QoS attribute values of the compositions, as well as the compositions' overall score (as far as the weights assigned to the three attributes are concerned, these are the ones set by the user in the example – Table 3 and Figure 2), whereas Table 5 illustrates the calculation results of some of the combinations that are generated formulated by the algorithm. Among the combinations considered, the combination <*SwissAir*, *CostaNavarino*, *EuropeanCup*> attains the highest score and is therefore selected to perform the particular adaptation.

Table 5 QoS attribute value and overall score computation for service candidate combinations

Combination	*QoS attribute*			*Overall score*
	rt	*cost*	*rel*	
<SwissAir, GrandResort, EuropeanCup>	1.66	1.87	0.44	1.113
<SwissAir, GrandResort, SportsTicketBooker>	1.66	1.92	0.39	1.103
<SwissAir, CostaNavarino, EuropeanCup>	*1.7*	*2.17*	*0.68*	*1.333*
…	…	…	…	…
<OlympicAirways, CostaNavarino, SportsTicketBooker>	1.75	2.07	0.59	1.267

4.3 The IP-based algorithm

As noted above, the IP-based algorithm formulates an IP problem, the solution of which is the adaptation to be performed on the scenario. In more detail, the IP problem formulation proceeds as follows:

1 Initially the functionalities to be adapted in the original scenario are identified, the corresponding branches in the taxonomy containing potential candidates for the realisation of the functionalities are identified, and the potential candidates are filtered according to the QoS limits designated by the user. This step is identical to the corresponding procedure used in the exhaustive algorithm; when this step concludes, for each of the functionalities f_i specified in the scenario under adaptation, a set of candidate functionality implementations C_{f_i} is formulated.

  Recall that in our example, the functionalities to be adapted are 'AirTravel', 'Luxury' and 'Athletics'. For brevity in notations, we will use the ordinals 1, 2 and 3 to denote the corresponding functionalities. Let us assume that $C_1 = \{SwissAir, OlympicAirways\}$, $C_2 = \{GrandResort, CostaNavarino\}$ and $C_3 = \{EuropeanCup, SportsTicketBooker\}$.

2 For each set of candidate functionality implementations C_{f_i}, a set of variables $\{x_{i,1}, x_{i,2}, \ldots, x_{i,m_i}\}$ is introduced, where m_i denotes the cardinality of C_{f_i}, i.e., $m_i = |C_{f_i}|$. Variable $x_{i,j}$ effectively corresponds to the jth candidate for realising functionality f_i and may be assigned either the value 1, if the jth candidate for realising functionality f_i is actually selected to realise the functionality f_i in the context of the adaptation, or 0, otherwise. Since exactly one of the candidates within C_{f_i} will be selected for the realisation of f_i, it follows that the sum of all variables $x_{i,j}$ must be equal to 1.

  According to the above, for each set of candidate functionality implementations C_{f_i}, the following elements are added to the IP problem:

a A set of variables $\{x_{i,1}, x_{i,2}, \ldots, x_{i,m_i}\}$, where $m_i = |C_{f_i}|$.

b A set of constraints, $\{0 \le x_{i,j} \le 1\}, \forall j : 1 \le j \le |C_{f_i}|$; note that since we employ IP, $x_{i,j}$ can only be assigned integer values and therefore possible assignments to each $x_{i,j}$ are only $x_{i,j} = 0$ and $x_{i,j} = 1$. In environments supporting mixed IP problems, i.e., problems where some of the variables accept only integer values whereas other variables may be assigned any real value, the constraint that the $x_{i,j}$ variables are integers is added using a declaration of the form 'integer $x_{i,j}$'.

c a constraint $\sum_{j=1}^{|C_{f_i}|} x_{i,j} = 1$, for each of the categories C_{f_i}.

In our example, the set of variables would be $\{x_{1,1}, x_{1,2}, x_{2,1}, \ldots, x_{3,2}\}$, with $x_{1,1}$ corresponding to the first candidate service of the first functionality (i.e., *SwissAir*), $x_{1,2}$ corresponding to the second candidate service of the first functionality (i.e., *OlympicAir*) and so forth. The set of constraints stemming that are added according to item 2, above, would be $\{0 \le x_{1,1} \le 1, 0 \le x_{1,2} \le 1, 0 \le x_{1,1} \le 1, \ldots, 0 \le x_{3,2} \le 1\}$. Finally, the set of constraint introduced due to item 3 above would be $\{x_{1,1} + x_{1,2} = 1, x_{2,1} + x_{2,2} = 1, x_{3,1} + x_{3,2} = 1\}$.

3 The next step focuses on the formulation of the objective function. The objective function is built according to the following algorithm:

a *A partial objective function POF_{cost} is formulated for the cost QoS attribute:* The use of a particular service implementation $s_{i,j} \in C_{f_i}$ to realise functionality f_i in the final adaptation would entail the payment of the cost associated with that service implementation; however, if the service is not selected, the cost is not entailed (recall that all QoS attributes, including cost, are normalised in the range [0, 1] with higher values being more preferable for the user). Since 1 each service $s_{i,j}$ is associated with a corresponding binary variable $x_{i,j}$ introduced in step 2 above, and 2 according to the formulas introduced in Section 3.1, the overall value of the cost QoS attribute in a composition is the sum of the costs of the constituent parts, formula (7) reflects the overall value for the cost QoS attribute, taking into account both the service selections and the associated costs (n denotes the number of functionalities to be adapted in the scenario).

$$\sum_{i=1}^{n} \sum_{j=1}^{|C_{fi}|} \left(x_{i,j} * \cos t\left(s_{i,j}\right) \right) \tag{7}$$

In our example, the partial objective function corresponding to the *cost* QoS attribute would be:

$$POF_{cost} = 0.7 \times x_{1,1} + 0.55 \times x_{1,2} + 0.62 \times x_{2,1} + \ldots + 0.6 \times x_{3,2} \tag{8}$$

b *A partial objective function POF_{rel} is formulated for the reliability QoS attribute:* Similarly to the case of the cost presented above, and taking into account that according to the formulas introduced in Section 3.1, the overall value of the reliability QoS attribute in a composition is the product of the reliabilities of the constituent parts, the formula reflecting the overall value for the reliability QoS attribute is:

$$\prod_{i=1}^{n}\sum_{j=1}^{|C_{fi}|}\left(x_{i,j} * rel\left(s_{i,j}\right)\right) \tag{9}$$

However, this (partial) objective function is multiplicative, and thus cannot be used in an IP problem formulation, since IP requires that objective functions are linear (c.f. Section 3.3). To address this issue, the approach suggested in (Cardellini et al., 2017; Zeng et al., 2004) is followed, according to which the property

$$\log\left(\prod_{i} a_i\right) = \sum_{i} \log\left(a_i\right) \tag{10}$$

is exploited to transform the product to a sum, effectively substituting all occurrences of *rel*(*i*) by log(*rel*(*i*)) and all products by sums. Effectively:

$$\begin{aligned} POF_{rel} &= \log\left(\prod_{i=1}^{n}\sum_{j=1}^{|C_{fi}|}\left(x_{i,j} * rel\left(s_{i,j}\right)\right)\right) \\ \Rightarrow POF_{rel} &= \sum_{i=1}^{n}\log\left(\sum_{j=1}^{|C_{fi}|}\left(x_{i,j} * rel\left(s_{i,j}\right)\right)\right) \end{aligned} \tag{11}$$

and since exactly one of the $x_{i,j}$ variables will be equal to one for each *i*, formula (11) can be rewritten as:

$$POF_{rel} = \prod_{i=1}^{n}\sum_{j=1}^{|C_{fi}|}\left(x_{i,j} * \log\left(rel\left(s_{i,j}\right)\right)\right) \tag{12}$$

which adheres to the requirements for IP objective functions.

Following the procedure described above, in the case of the trip booking example the partial objective function corresponding to the *reliability* QoS attribute would be equal to:

$$\log(0.8)\times x_{1,1} + \log(0.78)\times x_{1,2} + \log(0.61)\times x_{2,1} + \ldots + \log(0.8)\times x_{3,2} \tag{13}$$

c *A partial objective function POF_{rt} is formulated for the response time QoS attribute:* In this formulation, the different elements are handled differently, depending on whether they are composed under a sequential or a parallel execution structure, according to the formulas listed in Section 3.1. The procedure for formulating the partial objective function for the response time QoS attribute proceeds in a bottom-up fashion as follows:

- For each a set of candidate functionality implementations $C_f = \{i_{f,1}, i_{f,2}, \ldots, i_{f,n(f)}\}$, the term:

  $$RT(f) = rt\left(i_{f,1}\right)\times x_{f,1} + rt\left(i_{f,2}\right)\times x_{f,2} + \ldots + rt\left(i_{f,n(f)}\right)\times x_{f,n(f)} \tag{14}$$

  is formulated, modelling the response time of the functionality, taking into account the choice of a particular service implementation that will be made to realise functionality *f*.

- If two simple or composite functionalities f_1 and f_2 are combined by means of a *SEQ* construct into a composite functionality comp(f_1, f_2, *SEQ*), the terms expressing the response time of f_1 and f_2 are added up together to form the term conveying the runtime of composite service *comp*(f_1, f_2, *SEQ*), as shown in formula (15):

$$RT\left(comp\left(f_1, f_2, SEQ\right)\right) = RT\left(f_1\right) + RT\left(f_2\right) \tag{15}$$

- If two simple or composite functionalities f_1 and f_2 are combined by means of a *FLOW* construct into a composite functionality *comp*(f_1, f_2, *FLOW*), the maximum of the terms expressing the response time of f_1 and f_2 is calculated to form the term conveying the runtime of composite service *comp*(f_1, f_2, *FLOW*), as depicted in formula (16):

$$RT\left(comp\left(f_1, f_2, FLOW\right)\right) = \max\left(RT\left(f_1\right) + RT\left(f_2\right)\right) \tag{16}$$

The formula corresponding to the root composition of the WS-BPEL scenario represents the computation of the partial objective function is formulated for the response time QoS attribute, POF_{rt}.

Cases 1 and 2 effectively create a formula that is directly compatible with the IP problem formulation (a sum of terms, where each term is of the form $\alpha_{ij}x_i$; c.f. Section 3.3). However, case 3 corresponding to the parallel composition entails the application of the *max* function, which is not permissible in an IP problem. This issue is addressed following the method described in Bisschop (2020), according to which an objective entailing the *max* function (termed a *minimax objective*) undergoes a transformation procedure to assume a form compliant with the IP problem formulation specifications. In more detail, an objective function containing the term

$$\max_{p \in P}\left(\sum_{q \in Q} c_{p,q} x_q\right) \tag{17}$$

is transformed as follows:

- a new variable z is introduced, which represents the above term
- a set of constraints of the form

$$\sum_{q \in Q} c_{p,q} x_q \le z, \forall p \in P \tag{18}$$

  are added to the problem, to ensure that the newly introduced term z (representing the *max* function) is greater than or equal to each individual argument of the *max* function.

Notably, this transformation can be applied either directly in the problem modelling stage, or internally by the software that is employed to calculate the solution to the IP problem. In the solution implemented in the context of our research, we have used the IBM ILOG CPLEX integrated software environment (IBM, 2013a, 2013b), which performs this transformation internally, hence the integer problem is formulated using the *max* function where needed, as dictated by the formulas in Section 3.1, and presented to the IBM ILOG CPLEX software, which applies any necessary transformations and yields the solution to the problem.

In the trip booking example, the partial objective function formulated for the response time QoS attribute is as shown in formula (19):

$$0.75\times x_{1,1}+0.8\times x_{1,2}+\max\left(0.91\times x_{2,1}+0.95\times x_{2,2}, 0.75\times x_{3,1}+0.65\times x_{3,2}\right) \quad (19)$$

d *The overall objective function OF is calculated:* The three partial objective functions, POF_{rt}, POF_{cost} and POF_{rel} are synthesised into a single objective function using a simple weighted sum formula. More specifically:

$$\begin{aligned} OF &= QoSWeights_{rt}\times POF_{rt}+QoSWeightd_{cost}\times POF_{cost} \\ &\quad +QoSWeights_{rel}\times POF_{rel} \end{aligned} \quad (20)$$

where *QoSWeights* is the vector expressing the weights of the QoS attributes are scenario level (c.f. Table 3 and Figure 2).

The performance assessment of the two algorithms (ES and IP-based), in terms of online adaptation time and quality of the produced results, is presented in the next section.

5 Experimental evaluation

In this section, we present our experiments, aiming to evaluate the proposed algorithm, in terms of:

a the quality of the formulated adaptations, i.e., how close the score of the adaptation formulated by the proposed algorithm is to the optimal one

b the efficiency of the algorithm, corresponding to the time needed to compute the requested adaptations.

Notably, the exhaustive algorithm presented in Margaris et al. (2013) computes the optimal adaptation, since it completely enumerates all possible adaptations and selects the one attaining the maximum score.

For the experimental evaluation, we utilised a machine equipped with one Celeron N2840@2.16GHz with 4 GB of RAM, which hosted the BPEL scenarios (XML files) and the service repository. The latter was the host for the representation of the subsumption relationships and QoS characteristics of the services. For the purposes of the evaluation we generated 30 WS-BPEL scenarios of varying complexity and size. More specifically, we formulated a search grid considering as parameters the number of functionalities per scenario (3, 4, 7 and 9), number of implementations per functionality (ranging between 3, 6, 9 and 12) and number of parallel constructs within a scenario (ranging between 0 and 4, with the number of parallel constructs being less than N/2), and the vertices of this grid were used to create the search scenarios. The properties of the set of generated WS-BPEL scenarios are listed in Table 6, while the 30 WS-BPEL scenarios used in this experiment are provided in SoDa Lab (2020).

Table 6 Properties of the generated WS-BPEL scenarios set

Property	*Value*
Number of WS-BPEL scenarios	30
Minimum number of functionalities	3
Maximum number of functionalities	9
Mean number of functionalities	5.8
Minimum number of implementations for a functionality	3
Maximum number of implementations for a functionality	12
Mean number of implementations for a functionality	7.4
Minimum number of alternative compositions (exhaustive enumeration)	153
Maximum number of alternative compositions (exhaustive enumeration)	2,177,280
Mean number of alternative compositions (exhaustive enumeration)	485,956.6
Median of alternative compositions (exhaustive enumeration)	95,040
Minimum number of parallel constructs within a scenario	0
Maximum number of parallel constructs within a scenario	3
Mean number of parallel constructs within a scenario	1.23

Each scenario was run 100 times and, in each execution, the weights of the three QoS attributes (runtime, cost, reliability) were randomly set in the range [0, 1] and subsequently normalised in order to ensure that $rt_w + c_w + rel_w = 1$.

Figure 5 Execution time required by the two algorithms

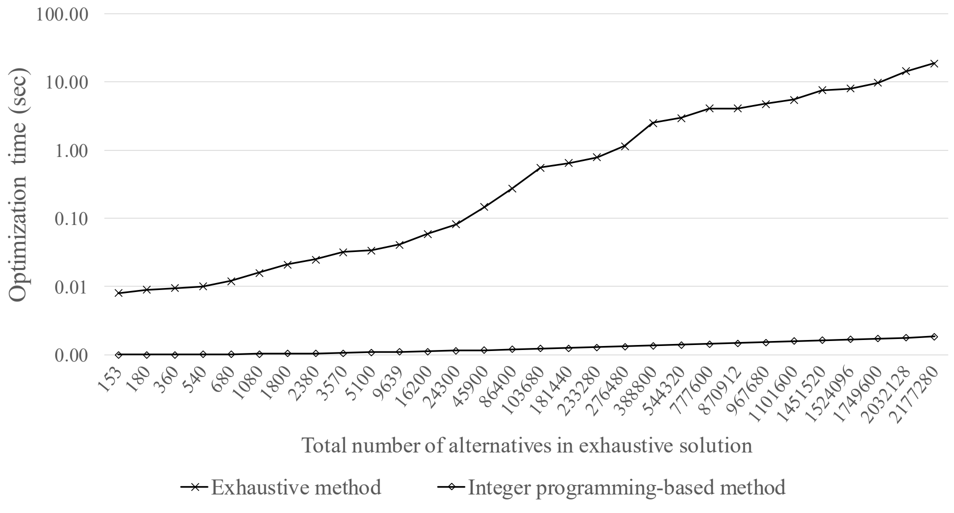


Figure 5 illustrates the execution time required by the two algorithms used in our work. We can clearly see that, while the processing time of the ES algorithm (which creates and evaluates all possible combinations between WSs, in order to select the one achieving the highest score) increases exponentially when the number of combinations that need to be tested increases (note that the y-axis follows a logarithmic scale), the IP-based algorithm, on the other hand, manages to keep the execution time at very low levels, even in the most demanding scenarios, consisting of nine services to be recommended, with over 2 M alternative compositions. This indicates that the IP-based algorithm can be used in online environments, since the overhead introduced is practically negligible.

Figure 6 Optimisation time required by the ES algorithm (see online version for colours)

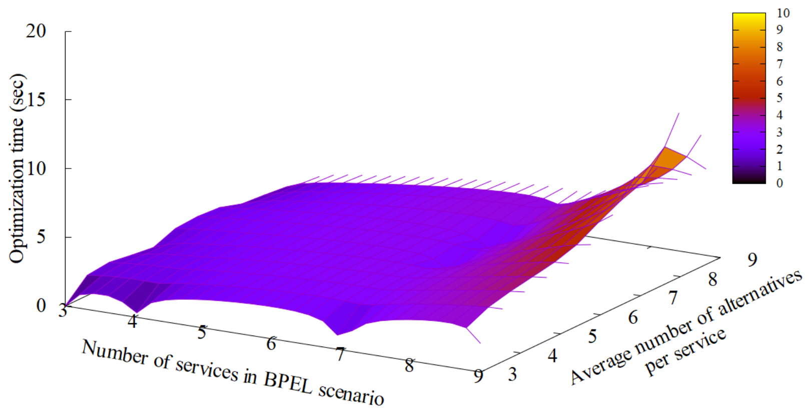
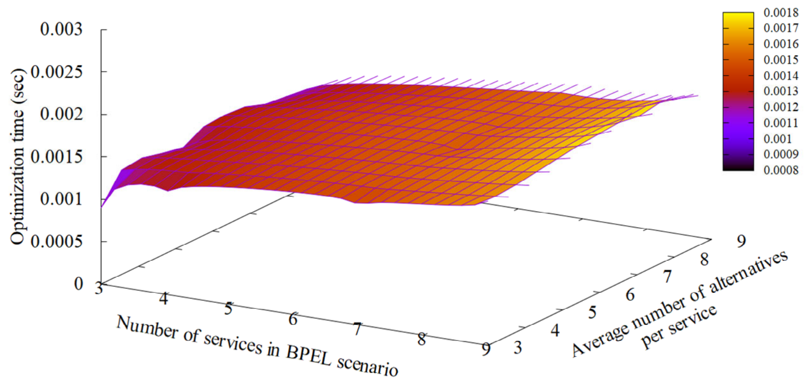


Figure 6 elaborates on the optimisation time required by the ES algorithm, considering the number of services in the BPEL scenario and the average number of alternatives per service. We can observe that the optimisation time increases exponentially with both parameters, while for scenarios having a moderate number of services (seven or more) and a moderate number of alternatives per service (exceeding six alternatives on average from scenarios with seven services or exceeding 3–4 on average for scenarios with nine services), the optimisation time surpasses 1 second.

Figure 7 Optimisation time required by the IP-based algorithm (see online version for colours)



Correspondingly, Figure 7 elaborates on the optimisation time required by the IP-based algorithm, considering the number of services in the BPEL scenario and the average number of alternatives per service. We can observe that the optimisation time increases very slightly as the number of services and/or the number of alternative implementations per service increases.

Figure 8 Comparison of the scores of the solutions produced by the two algorithms

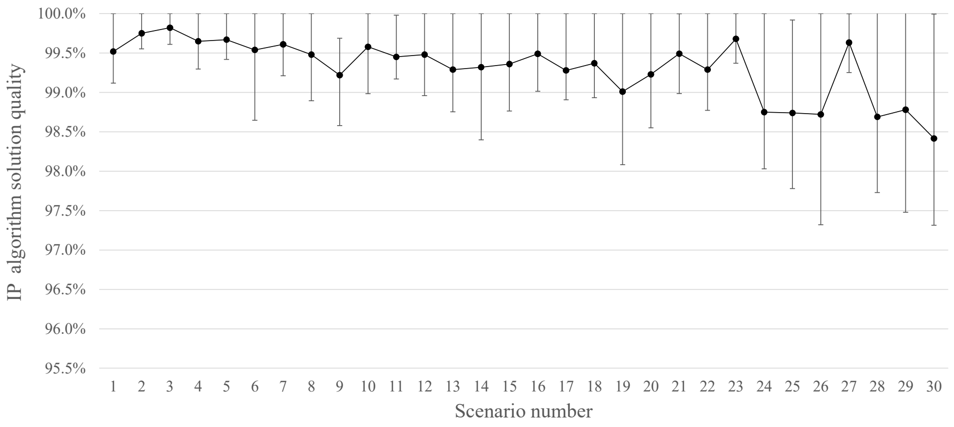


Figure 8 illustrates our experimental findings regarding the quality of the execution score achieved by the IP algorithm, in relation to the quality of the solution produced by the ES algorithm (which is the optimal one, since the ES algorithm considers all candidate solutions). For each scenario (represented as an item on the horizontal axis), the average score quality over the 100 runs with randomised QoS attribute weights is depicted, while the error bars represent the minimum and maximum score quality observed for that particular scenario. We can observe that the quality of the solution computed by the IP algorithm is greater than or equal to 97% for all cases, while the global average score quality of the IP algorithm (i.e., the average quality of the solutions computed by the IP algorithm over the 100 randomised runs of each scenario in relation to the respective scores of produced by the exhaustive algorithm) is 99.37%. For 24 scenarios, the IP algorithm was able to compute an optimal solution for at least one QoS attribute weight combination. Overall, in 62% of the tested cases, the solution computed by the IP algorithm was identical to the one computed by the exhaustive algorithm.

In general, scenarios comprising of higher numbers of constituent services, exhibit a larger deviation between the score of the solution computed by the IP algorithm and the optimal one, computed by the ES algorithm. However, the performance gains, when the number of services increases, are more substantial.

In comparison with other algorithms presented in the literature, the proposed IP-based algorithm provides a speedup of 2–3 orders of magnitude against the OPTIM_HWEIGHT algorithm (Comes et al., 2010), 1–2 orders of magnitude as compared to the QSSAC algorithm (Xia et al., 2011) and 2–4 orders of magnitude against the WSC−CGA algorithm (Liu et al., 2015).

Overall, the experiments clearly demonstrate that the IP-based algorithm can provide very substantial speedup over both the exhaustive algorithm, as well as other adaptation algorithms presented in the literature, with practically negligible effects on the quality of the formulated solutions. Since the overhead introduced by the proposed algorithm is small, the proposed algorithm can be directly integrated into WS-BPEL execution engines, to cater for online adaptation of user requests.

6 Conclusions and future work

In this work, an IP-based algorithm for adapting the execution of WS-BPEL scenario invocations, recommending services according to criteria and policies set by the user was presented. The proposed algorithm exploits multiple characteristics of the WSs and the WS-BPEL language to formulate an IP problem which is then solved to optimise the WS-BPEL scenario execution. The characteristics exploited are clearly domain dependent. The algorithm design process, however, can be used to formulate analogous algorithms in other domains. Each of those algorithms should be verified and validated, considering appropriate characteristics and datasets from the relevant domains. The proposed algorithm was validated through a set of experiments, using scripts with varying size and complexity. These experiments showed that the IP-based algorithm's execution time remains low, even in the most demanding scenarios, addressing thus an important shortcoming of the ES algorithm, which guarantees solution optimality on the one hand, however its execution time grows exponentially with the number of functionalities within the scenario to be adopted, as well as with the number of candidate services that can be used to realise a functionality. The proposed algorithm was also compared to other algorithms that are presented in the literature and was found to reduce the adaptation computation time by 1–4 orders of magnitude. As far as the adaptation quality is concerned, the solutions produced by the IP-based algorithm are very close to the optimal one, with their overall score lagging behind the score of the optimal one less than 2.9% in all cases and less than 0.5% on average. The above results clearly demonstrate the feasibility of the proposed algorithm, as well as its suitability for an online usage, since adaptations are computed in less than 0.003 seconds, therefore any incurred overhead is negligible.

Our future work will focus on handling of loops and conditional execution constructs in WS-BPEL scenarios; this can be supported through branch prediction and loop unrolling techniques (Bernstein, 1996), as well as by gathering statistical information from prior scenario executions and using it as input to the adaptation process.

Acknowledgements

The authors would like to sincerely thank Prof. Costas Vassilakis for his supervision, guidance, support and scientific input for the work presented in this paper.

References

Bellur, U. and Kulkarni, R. (2007) 'Improved matchmaking algorithm for semantic web services based on bipartite graph matching', *Proceedings of the IEEE International Conference on Web Services, 2007 (ICWS 2007)*, IEEE, Salt Lake City, UT, pp.86–93.

Bernstein, A.J. (1996) 'Analysis of programs for parallel processing', *IEEE Transactions on Electronic Computers*, Vol. 15, No. 5, pp.757–763, IEEE.

Bisschop, J. (2020) 'Linear programming tricks', chapter in *AIMMS Optimization Modeling*, AIMMS [online] https://download.aimms.com/aimms/download/manuals/AIMMS3_OM.pdf (accessed 17 January 2020).

Bixby, R.E., Fenelon, M., Gu Z., Rothberg, E. and Wunderling, R. (2004) 'Mixed integer programming: a progress report', chapter in Martin Grötschel (Ed.): *The Sharpest Cut: The Impact of Manfred Padberg and his Work*, pp.309–325, MPS-SIAM Series on Optimization, Berlin-Dahlem, Germany.

Canfora, G., Di Penta, M., Esposito, R. and Villani, M.L. (2005) 'An approach for QoS-aware service composition based on genetic algorithms', Beyer, H-G. and O'Reilly, U-M. (Eds.): *Proceedings of the 2005 Conference on Genetic and Evolutionary Computation*, ACM, pp.1069–1075.

Cardellini, V., Casalicchio, E., Grassi, V., Iannucci, S., Presti, F. and Mirandola, R. (2012) 'MOSES: a framework for QoS driven runtime adaptation of service-oriented systems', *IEEE Transactions on Software Engineering*, September/October, Vol. 38, No. 5, pp.1138–1159.

Cardellini, V., Casalicchio, E., Grassi, V., Iannucci, S., Presti, F. and Mirandola, R. (2017) 'MOSES: a platform for experimenting with QoS-driven self-adaptation policies for service oriented systems', *Software Engineering for Self-Adaptive Systems III. Assurances. Lecture Notes in Computer Science*, Springer, Vol. 9640, pp.409–433.

Chen, F., Dou, R., Li, M. and Wub, H. (2016) 'A flexible QoS-aware web service composition method by multi-objective optimization in cloud manufacturing', *Computers & Industrial Engineering*, Elsevier, Vol. 99, pp.423–431.

Comerio, M., De Paoli, F., Grega, S., Maurino A. and Batini, C. (2007) 'WSMoD: a methodology for QoS-based web services design', *Web Services Research*, Vol. 4, No. 2, pp.33–60.

Comes, D., Baraki, H., Reichle, R., Zapf, M. and Geihs, K. (2010) 'Heuristic approaches for QoS-based service selection', *Proceedings of the International Conference on Service-Oriented Computing, (ICSOC'2010), LNCS 6470*, Springer, pp.441–455.

Fu, X., Bultan, T. and Su, J. (2004) 'Analysis of interacting BPEL web services', *Proceedings of the 13th international conference on World Wide Web (WWW '04)*, ACM, pp.621–630.

Gouscos, D., Kalikakis, M. and Georgiadis, P. (2003) 'An approach to modeling web service QoS and provision price', *Proceedings of the Fourth International Conference on Web Information Systems Engineering Workshops*, IEEE, pp.121–130.

Gupta, R., Kamal, R. and Suman, U. (2018a) 'A QoS-supported approach using fault detection and tolerance for achieving reliability in dynamic orchestration of web services', *International Journal of Information Technology*, Vol. 10, No. 1, pp.71–81, Springer.

Gupta, R., Kamal, R. and Suman, U. (2018b) 'Designing of an orchestration mechanism for the efficient web-services composition', *Progress in Computing, Analytics and Networking. Advances in Intelligent Systems and Computing*, Vol. 710, pp.171–182, Springer, Singapore.

Halfaoui, A., Hadjila, F. and Didi, F. (2015) 'QoS-aware web services selection based on fuzzy dominance', in Amine, A., Bellatreche, L., Elberrichi, Z., Neuhold, E. and Wrembel, R. (Eds.): *Computer Science and Its Applications (CIIA 2015), IFIP Advances in Information and Communication Technology*, Springer, Cham, Vol. 456.

Hallwyl, T., Henglein, F. and Hildebrandt, T. (2010) 'A standard-driven implementation of WS-BPEL 2.0', *Proceedings of the 2010 ACM Symposium on Applied Computing (SAC '10)*, ACM, pp.2472–2476.

Hammas, O., Yahia, S.B. and Ahmed, S.B. (2015) 'Adaptive web service composition insuring global QoS optimization', *Proceedings of the 2015 International Symposium on Networks, Computers and Communications*, IEEE, pp.1–6.

He, D. and Wu, D. (2008) 'Toward a robust data fusion for document retrieval', *Proceedings of the IEEE 4th International Conference on Natural Language Processing and Knowledge Engineering, NLP-KE 2008*, Beijing, China, IEEE, pp.19–22.

IBM (2013a) *IBM ILOG CPLEX Optimizer* [online] http://www.ibm.com/software/commerce/optimization/cplex-optimizer/ (accessed 23 September 2019).

IBM (2013b) *Using CPLEX to Examine Alternate Optimal Solutions* [online] http://www.ibm.com/support/docview.wss?uid=swg 21399929 (accessed 23 September 2019).

Kareliotis, C., Vassilakis, C., Rouvas, S. and Georgiadis, P. (2009) ‘QoS-driven adaptation of BPEL scenario execution’, *Proceedings of the 2009 International Conference on Web Services*, IEEE, pp.271–278.

Liang, X., Qin, A., Tang, K. and Tan, K. (2019) ‘QoS-aware web service selection with internal complementarity’, *IEEE Transactions on Services Computing*, Vol. 12, No. 2, pp.276–289, IEEE.

Liu, Z.Z., Jia, Z.P. and Xue, X. (2015) ‘Reliable web service composition based on QoS dynamic prediction’, *Soft Computing*, Vol. 19, No. 5, pp.1409–1425, Springer.

Lu, Y. and Xu, X. (2017) ‘A semantic web-based framework for service composition in a cloud manufacturing environment’, *Journal of Manufacturing Systems*, Vol. 42, pp.69–81, Elsevier.

Machorro-Cano, I., Alor-Hernández, G., Olmedo-Aguirre, J.O., Rodríguez-Mazahua, L. and Segura-Ozuna, M.G. (2020) ‘IoT services orchestration and choreography in the healthcare domain’, *Techniques, Tools and Methodologies Applied to Global Supply Chain Ecosystems*, Vol. 166, pp.429–454, Intelligent Systems Reference Library, Springer, Cham.

Margaris, D., Vassilakis, C. and Georgiadis, P. (2013) ‘An integrated framework for QoS-based adaptation and exception resolution in WS-BPEL scenarios’, *Proceedings of the 28th ACM Symposium on Applied Computing (ACM SAC 2013)*, ACM, pp.1900–1906.

O’Sullivan, J., Edmond, D. and Ter Hofstede, A. (2002) ‘What is a service?: Towards accurate description of non-functional properties’, *Distributed and Parallel Databases*, Vol. 12, pp.117–133, Kluwer Academic Publishers, The Netherlands.

Rodriguez-Mier, P., Pedrinaci, C., Lama, M. and Mucientes, M. (2016) ‘An integrated semantic web service discovery and composition framework’, *IEEE Transactions on Services Computing*, Vol. 9, No. 4, pp.537–550, IEEE.

Schrijver, A. (1998) *Theory of Linear and Integer Programming*, John Wiley & Sons, New York, USA.

SoDa Lab (2020) *WS-BPEL Scenarios for Adaptation Benchmarking* [online] https://soda.dit.uop.gr/?q=node/774 (accessed 2 May 2020).

Song, Z. and Tilevich, E. (2019) ‘Equivalence-enhanced microservice workflow orchestration to efficiently increase reliability’, *Proceedings of the 2019 IEEE International Conference on Web Services*, IEEE, pp.426–433.

Wang, P., Ding, Z., Jiang, C., Zhou, M. and Zheng, Y. (2016) ‘Automatic web service composition based on uncertainty execution effects’, *IEEE Transactions on Services Computing*, Vol. 9, No. 4, pp.551–565, IEEE.

Xia, Y., Chen, P., Bao, L., Wang, M. and Yang, J. (2011) ‘A QoS-aware web service selection algorithm based on clustering’, *Proceedings of the 2011 International Conference on Web Services*, IEEE, pp.428–435.

Zatout, S., Berkane, M. and Boufaida, M. (2018) ‘An architecture dedicated to dynamic adaptation for services orchestration’, *Proceedings of the 8th International Conference on Computer Science and Information Technology*, IEEE, pp.219–224.

Zeng, LB., Benatallah, A.H.N., Dumas, M., Kalagnanam, J. and Chang, H. (2004) ‘QoS-aware middleware for web services composition’, *IEEE Transactions on Software Engineering*, Vol. 30, No. 5, pp.311–327, IEEE.

Zhang, Y. and Lyu, M.R. (2017) ‘QoS-aware web service searching’, chapter in *QoS Prediction in Cloud and Service Computing*, pp.81–103, Springer, Singapore.