A Heuristic Technique for an Efficient Decision Support in Context-aware Service Selection

Completed Research Paper

Lars Lewerenz
University of Regensburg
Universitätsstraße 31
93053 Regensburg, Germany
Lars.Lewerenz@wiwi.uni-regensburg.de

Abstract

In service-oriented systems, context information can be used to select a variety of services to support the execution of processes. Yet, context information causes interdependencies between services, in the sense that they are evaluated differently – regarding their utility and feasibility - when being composed with other services. Such context interdependencies have not been sufficiently addressed by research so far, as approaches either disregard them or propose exact solutions which are hardly applicable due to high time complexity of the selection problem. To alleviate this drawback, we present a heuristic technique considering context interdependencies. In particular, this technique consists of an approach to decompose end-to-end constraints together with a local selection approach. Our evaluation, by means of a real-world data, shows that this technique achieves close to optimal selection results at a fraction of the computation time of an exact solution and thus contributes to an efficient decision support.

Keywords: Context-aware, context interdependencies, service selection, decomposition of constraints, local selection

Introduction

Context information is one of the key factors that heavily influences most real-world processes (cf. Hu et al. 2012; Wang et al. 2012; Zhou et al. 2011). According to Dey (2001), context information can be defined as "any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves". Emerging technologies such as mobile devices (e.g., smart phones, wearables and tablets) further reinforce the influence of context information on processes, as they allow us to capture and to process context information in a convenient way. As a result, more and more context information is available, that can be used to adapt processes to the current needs and situations of a conducting entity.

An architectural paradigm that allows for a flexible adaption of processes to changing environmental needs or business challenges is the one of a service-oriented architecture (cf. Papazoglou 2012; Serrano et al. 2014). Here, the execution of processes can be supported by composing a number of loosely coupled services together (i.e., service composition). A widely discussed problem in this regard concerns the selection of the best services from a set of functionally equivalent services – referred to as a service class – for each process action. Decision support for this problem is provided by means of QoS-aware service selection approaches (cf. Alrifai et al. 2012; Ardagna and Pernici 2007; Yu et al. 2007; Zeng et al. 2004). These approaches basically map the different non-functional properties (NFP) of services – represented by QoS attributes (e.g., price, response time, availability) – onto a single utility value and maximize this value under the consideration of end-to-end constraints comprising user-defined requirements (e.g., an upper price limit for the entire process). Zeng et al. (2004) and Yu et al. (2007), for instance, regarding the QoS-aware service selection problem as a multi-choice, multi-dimensional knapsack problem (MMKP) which is known to be NP-hard (cf. Hwang and Yoon 1981) in the strong sense. As a result, every exact solution to this problem is expected to have an exponential time complexity with respect to the problem size (cf. Alrifai et al. 2012). Here, the main influencing factor, besides the number of service classes and the number of end-to-end constrains, is the number of functionally equivalent services available for each service class.

With the consideration of context information in service selection (cf. Yu and Reiff-Marganiec 2009a; Yuan et al. 2013; Zhang et al. 2013), the time complexity of an exact solution is even increased. In detail, this can be accounted for by the two following aspects. The first one is the existence of *context-aware attributes* of a service (cf. Cuddy et al. 2005; Heinrich and Lewerenz 2015; Lin et al. 2012). These attributes are characterized by the fact that their quantified values are not fixed and given, but rather are dependent on the composition in which the corresponding service is participating as well as on the context information that arises from this particular composition. To give an example: the response time of a service can differ depending on the physical distance to a previously selected service (see e.g., Lin et al. 2012). Thus to quantify this value, we first need to know where the preceding service is located which is context information (e.g., GPS position). Then, this information can be used to calculate the distance to a succeeding service and consequently to quantify the response time (e.g., the lower the distance, the lower the response time and vice versa). The second aspect is the dynamic characteristic of context information (cf. Gu et al. 2005; Henricksen et al. 2002), which is the cause for the following three effects (cf. Heinrich and Lewerenz 2015):

❶ Context information is dependent on the previously selected services, whereas the quantified values of context-aware attributes themselves depend on context information. The context information GPS position, for instance, strongly depends on the GPS position of the previously selected service. As a consequence, different context information can be determined subject to the considered service composition.
❷ As a result of ❶, the quantified values and thus the utility of the same service may be a different for each possible service composition.
❸ The selection of a service can lead to context information in which some or all succeeding services are not feasible with respect to the end-to-end constraints.

In the following ❶-❸ are denoted as *the effects of context interdependencies*. As a result, time complexity of an exact solution for the context-aware service selection is increased as the problem size is additionally increased due to the potentially huge number of possible context information (cf. ❶). Because of the latter, a service no longer has a fixed single utility value (cf. QoS-aware service selection), but rather multiple ones, depending on the service composition which it is part of. The same applies to the feasibility of services

regarding the end-to-end constraints. It is obvious that the effects of context interdependencies need to be considered within a service selection approach, as they directly influence the utility and feasibility of a service composition.

Even though research has dealt with the usage of context information for the selection of services, most approaches either disregard the effects of context interdependencies (cf. Cuddy et al. 2005; Kirsch-Pinheiro et al. 2008; Madkour et al. 2013; Sensoy et al. 2009; Vanrompay et al. 2009; Zhang et al. 2013), leave end-to-end constraints unconsidered (cf. Yu and Reiff-Marganiec 2009b; Yuan et al. 2013), or are expected to be confronted with scalability issues (cf. Heinrich and Lewerenz 2015; Wu and Zhu 2013). We will therefore address the following research question in this paper:

How to design a heuristic technique for the context-aware service selection that provides an efficient decision support by taking into account the effects of context interdependencies, end-to-end constraints and provides a good scalability.

The remainder of this paper is organized as follows: in the next section, we discuss the related literature and our contribution with regard to the identified research gap. In addition to that, we present our model setup. In the third section, we present our novel heuristic technique that is meant to address the research question of this paper. In the fourth section, we provide an evaluation of our heuristic technique with respect to its correctness and its ability to contribute to the stated research question. Finally, we conclude our paper with a short discussion on important limitations and outlook on future research.

Background

The following subsections provide an overview of the literature related to our research and a discussion of our contribution to it, ensued by the presentation of the model setup for context-aware service selection.

Related Work

Our research is related to the literature on QoS-aware service selection and directly contributes to the literature on context-aware service selection. We start by discussing the literature on QoS-aware service selection.

As already described in the introduction, every exact solution to the QoS-aware service selection problem has an exponential time complexity, when being modeled as an MMKP due to the NP-hardness of the problem (cf. Hwang and Yoon 1981). Within the last decade, a number of heuristic techniques (cf. Berbner et al. 2006; Canfora et al. 2008; Li and Yan-xiang 2012; Wang et al. 2010; Yu et al. 2007; Zeng et al. 2004) have been proposed to address this time complexity. Here, it became clear that approaches which use local selection (e.g., Zeng et al. 2004) usually have a superior performance – with regard to computation time – than heuristic techniques such as genetic algorithms (e.g., Canfora et al. 2005) or ant colony optimization (e.g., Li and Yan-xiang 2012). The reason for this is low time complexity of the local selection of $O(j)$, where j is the number of services available for a service class. However, end-to-end constraints cannot be taken into account with local selection. To resolve this issue, several works (cf. Alrifai et al. 2012; Sun and Zhao 2012; Surianarayanan et al. 2014) present approaches to decompose the end-to-end constraints into local constraints. Alrifai et al. (2012), for instance, model their decomposition approach as a MMKP. They specify a number of quality levels (e.g., value of a specific NFP) for each service class and attribute and assign a utility value to each of them. Subject to the end-to-end constraint the accumulated utility of the quality levels is maximized. According to the authors, the problem size of their decomposition approach is determined by the number of service classes, the number of quality levels and the number of end-to-end constraints and thus is independent of the number of services. Consequently, this approach is usually more scalable than every approach aiming at an exact solution (e.g., Zeng et al. 2004).

Based on the evolution of QoS-aware service selection approaches over the last decade, we argue that the idea of using local selection in combination with an approach to decompose the end-to-end constraints also seems like a promising way to reduce the time complexity in context-aware service selection. However, as no consideration of context information and thus no consideration of the effects of context interdependencies (cf. ❶-❸) takes place, these approaches would need to be extended to fit to the context-aware service selection in an appropriate way.

In the following, we discuss current context-aware service selection works according to the applied approach, namely local selection, global optimization and heuristic technique.

Local selection: the basic idea of local selection is to select the best services for each service class individually while considering the context information available for this service class. Here, several approaches (cf. Cuddy et al. 2005; Kirsch-Pinheiro et al. 2008; Madkour et al. 2013; Sensoy et al. 2009; Vanrompay et al. 2009) can be found, which, however, do not take context interdependencies between services into account. In contrast to that is the backwards composition context based service selection (BCCbSS) approach by Yu and Reiff-Marganiec (2009a). Before executing the next action, the BCCbSS approach additionally selects the optimal service for the action after the next action while taking the corresponding context information into account. Then, the selection of services in the next action is reconsidered. The goal is to determine alternative context information potentially resulting in a higher utility of the selected services for the two considered activities. If this is the case, a re-selection is performed. Otherwise the initial selection is maintained.

For the context-aware service selection, local selection definitely has its advantages regarding scalability as time complexity is very low. However, with local selection end-to-end constraints cannot be considered.

Global optimization: it is used in scenarios where an exact solution (i.e., optimal service composition) is required. Heinrich and Lewerenz (2015) use the concept of states to model and organize context information. This allows for a determination of feasible context information regarding a service composition even at planning time. Several quantification functions and context-aware attributes are proposed. The optimal service composition is determined by using integer programming while considering the effects of context interdependencies as well as end-to-end constraints.

For the context-aware service selection, global optimization has its advantages. Particularly regarding the consideration of the effects of context interdependencies as well as end-to-end constraints. However, due to the high time complexity of the exact solution, global optimization is only applicable for very small problem instances. Hence, fast decision support for many real-world problems is not possible with global optimization.

Heuristic Techniques: several approaches (cf. Wu and Zhu 2013; Yuan et al. 2013; Zhang et al. 2013) can be found that address the high time complexity of an exact solution by presenting a number of heuristic techniques. Rather than modelling context information and quantifying context-aware attributes, Zhang et al. (2013) use process mining to determine existing context interdependencies between services in a specific pattern of the process. For each pattern, they determine the service composition with the highest utility by creating a feasible solution and then improve this solution by allowing infeasible upgrades followed by feasible downgrades. Yuan et al. (2013) use a genetic algorithm to determine the service composition with the highest utility. Here, context information is used to provide information about, for instance, price discounts in case certain services are composed together. Yet, the quantification of context-aware attributes is not taken into account. For a single action of the process and the corresponding services, the authors use a matrix to represent different, already quantified values of a single attribute that can emerge due to all possible service selections in the preceding action. Moreover, the authors leave end-to-end constraints unconsidered. Wu and Zhu (2013) propose an approach based on ant colony optimization. Here, the utility of a service composition is determined based on the NFP and transactional properties (i.e., compensatable, retriable, pivot and compensatable & retriable) of the participating services. Context interdependencies between services are considered in a sense that the quantified values of the transactional properties can differ according to the previous selected services. Additionally, the approach takes end-to-end constraints into account.

Research Gap and Contribution to Research

Existing context-aware service selection approaches mainly address the high time complexity of an exact solution, as heuristic or local selection approaches have been widely proposed (cf. Cuddy et al. 2005; Kirsch-Pinheiro et al. 2008; Madkour et al. 2013; Sensoy et al. 2009; Vanrompay et al. 2009, Wu and Zhu 2013, 2013; Yu and Reiff-Marganiec 2009b; Yuan et al. 2013; Zhang et al. 2013). Yet, none of these works offer an approach that takes into account the effects of context interdependencies (cf. ❶-❸) and end-to-end constraints. The only exceptions are the approaches of Heinrich and Lewerenz (2015) and Wu and Zhu (2013). However, these two approaches are usually confronted with scalability issues. For Heinrich and

Lewerenz (2015), this is due to the fact that they use global optimization which has high time complexity. For Wu and Zhu (2013) it can be accounted for by the usage of ant colony optimization which is based on a directed acyclic graph (cf. Dorigo and Caro 1999). For the context-aware service selection, the number of nodes and edges in this graph can be very large (i.e., due to existing context interdependencies) even for small problem instances. Thus, we expect a high time consumption to achieve close to optimal selection results. To conclude, to the best of our knowledge not a single context-aware service selection approach exists that is capable of considering the effects of context interdependencies and end-to-end constraints which at the same time provides a good scalability and thus is able to provide an efficient decision support.

Against this backdrop, we aim at designing an efficient heuristic technique for the context-aware service selection that is based on local selection in combination with a decomposition approach. We thereby contribute to the current body of knowledge in context-aware service selection in two ways: first, we illustrate how the effects of context interdependencies can be considered in a well-founded way for the decomposition of end-to-end constraints. Second, we present a selection approach that makes use of the low time complexity of local selection but at the same time is able to take end-to-end constraints into account. We further show how the idea of local selection can be combined with an efficient re-selection approach to consider the effects of context interdependencies during selection. To sum up, we not only extend current context-aware service selection approaches but also propose a first approach that is able to consider the effects of context interdependencies as well as end-to-end constraints at a very low time complexity.

Model Setup

In the following, we introduce our model setup in line with existing works, which means that we use those definitions and modeling elements that are considered to be a *common* knowledge base.

In our model, we consider a process that consists of a number of actions i (with $i = 1$ to I) that contribute to achieving the intended goal of the process. A service class S_i includes all services s_{ij} (with $i = 1$ to I, $j = 1$ to J_i) that provide an equivalent functionality to implement action i, but who differ in their values of the NFP. Moreover, we take the following three elementary workflow constructs into account: sequential (cf. Cui et al. 2011; Zeng et al. 2004), pick (cf. Wan et al. 2008; Yu et al. 2007) and conditional (cf. Alrifai et al. 2012; Yu et al. 2007). To consider these workflow constructs during service selection, we use the idea of execution routes (cf. Alrifai et al. 2012; Ardagna and Pernici 2007; Yu et al. 2007; Zeng et al. 2004). An execution route Γ is defined as a path of service classes S_i from the start to the end of the process that includes only one branch of each pick and conditional construct.

We further focus on a number of attributes n (with $n = 1$ to N) describing the quantified NFP values of a service. Thus, we introduce $p_{ij} = [p_{ij}^1, \dots, p_{ij}^N]$ as the *quantified* NFP vector for service s_{ij} that consists of the quantified values for each single attribute n. Please note that this vector integrates the quantified values of both non-context-aware attributes M (with $m = 1$ to M) as well as context-aware attributes O (with $o = 1$ to O) so that $M \cup O = N$ and $M \cap O = \emptyset$ hold.

To quantify the value of context-aware attributes, we use the following type-based *quantification functions* (cf. Shen et al. 2012; Yu and Reiff-Marganiec 2009b) (cf. Table 1):

Table 1. Type-based quantification functions

Type	Function
Boolean	$\tau^B = \begin{cases} 1 & if\ p_{ij}^o\ satisfies\ criterion\ G \\ 0 & otherwise \end{cases}$
Discount	$\tau^{CC} = p_{ij}^o * dc\ with\ \ dc\ as\ a\ discount\ factor\ \begin{cases} 0 < dc < 1\ if\ \tau^B = 1 \\ dc = 1 \quad if\ \tau^B = 0 \end{cases}$
Distance	$\tau^D = \begin{cases} R * c & if\ c \geq 1 \\ R * c * \arcsin(1) & if\ c < 1 \end{cases}; with\ R = 6371\ km$ $and\ c = 2 * \arcsin(sin2(\lvert L2(p_{ij}^o) - L1(p_{ij}^o)\rvert * 0.5) + \cos(L1(p_{ij}^o)) * \cos(L2(p_{ij}^o)) * sin2(\lvert \mathrm{G2}(p_{ij}^o) - \mathrm{G1}(p_{ij}^o)\rvert) * 0.5)^{0.5}$
Favorite	$\tau^F = p_{ij}^O - (a_c)^e\ with\ 0 < e < 1; a_c \in \mathbb{N}$

For each context-aware attribute p_{ij}^o, a quantification function and a context information (i.e., G, A, or location represented by latitude $L(p_{ij}^O)$ and longitude $G(p_{ij}^O)$, whereas the index 1 and 2 represent two succeeding services) is needed. Based on these two pieces of information, the value of a context-aware

attribute can be quantified. In the following, the presented types are illustrated by a short example of the *favorite type*. Let us consider a user who evaluates the category X with 4, the category Y with 3 and the category Z with 2 so that $p_{ij}^{O} = [4,3,3]$. For the corresponding context information, we define $A = [a^1, \dots, a^C]$ as a tally vector including a value for each single category c (with $c = 1$ to C) that indicates how often the corresponding category has been selected so far. Consequently, for the example A is given by $A = [2,0,0]$. If e is set to 0.2, then the value of the context-aware attribute will be quantified with $4 - 2^{0.2} = 2.85$ if p_{ij}^{O} equals category X or $3 - 0^{0.2} = 3$ if p_{ij}^{O} equals category Y or Z. A detailed discussion on the other type-based quantification functions can be found in Shen et al. (2012) or Yu and Reiff-Marganiec (2009b).

To model *context information*, we use the concept of states and in particular belief and world states (cf. Heinrich and Lewerenz 2015). A belief state BS_i is defined as a set of belief state tuples b_{ik} (with k as the number of tuples and i as the number of the corresponding service class) with $b_{ik} \coloneqq \big(v(b_{ik}), r(b_{ik})\big)$ where $v(b_{ik})$ is a state variable and the restriction $r(b_{ik})$ is a subset of the predefined domain $dom(b_{ik})$. $ws_{ik} \subseteq b_{ik}$ is a world state tuple where $\forall\, v(b_{ik}) \in b_{ik} \;\; |r(b_{ik})| = 1$ (cf. Ghallab et al. 2004; Heinrich et al. 2009; Heinrich and Schön 2015). Thus, each context information and its corresponding value is represented by exactly one state variable $v(b_{ik})$ in ws_{ik}. Further, BS_1 is defined as the initial state and BS_{I+1} as the respective goal state.

At planning time and for an execution route Γ, we use the state creation algorithm of Heinrich and Lewerenz (2015) to determine the *possible context information* with respect to a certain service composition. The algorithm starts at the initial state BS_1 and from there determines BS_{i+1} with the corresponding world states $ws_{(i+1)k}$ for S_{i+1} and subsequently for all service classes by imitating the selection of each $s_{ij} \in S_i$ (state-transition). For the state creation, we differentiate between two different types of state variables. State variables where the new value of the corresponding context information is only dependent on the last selected service s_{ij} (e.g., GPS position) are called *non-consecutive*. State variables where the value of the corresponding context information is dependent on the last selected service s_{ij} as well as on the last considered world state ws_{ik}, such that $W: ws_{ik} \times s_{ij} \rightarrow ws_{i'k}$ hold (e.g., daytime) are called *consecutive*.

For the selection of services, in which multiple attributes have to be considered, we use, in line with the existing literature (cf. Alrifai et al. 2012; Ardagna and Mirandola 2010; Ardagna and Pernici 2007; Zeng et al. 2004), a utility function. We consider attributes n^+ $(with\ n^+ = 1\ to\ N^+)$ where the corresponding NFP values need to be maximized and attributes n^- $(with\ n^- = 1\ to\ N^-)$ where the corresponding NFP values need to be minimized. To determine the utility value of a service, without loss of any generality (w.l.o.g), we apply simple additive weighting (SAW). In a first step, the values of the NFP of the services are normalized in the interval [0; 1] to ensure comparability between the differently scaled NFP values. Similar to Alrifai et al. (2012), this is achieved by using the aggregated maximum and minimum NFP values over all service classes S_i. For the attributes n the aggregated values are defined as follows:

$$Pmin(n) = \sum_{i=1}^{I}\big(Pmin(i,n)\big);\ Pmin(i,n) = \min_{s_{ij} \in S_i} p_{ij}^{n} \tag{1}$$

$$Pmax(n) = \sum_{i=1}^{I}\big(Pmax(i,n)\big);\ Pmax(i,n) = \max_{s_{ij} \in S_i} p_{ij}^{n} \tag{2}$$

In a second step, the normalized NFP values of the attributes are weighted with the preferences of the user. Hence, the utility U_{ij} of a service s_{ij} is defined as follows:

$$U_{ij} = \sum_{n^-=1}^{N^-}\left(\frac{Pmax(i,n^-) - p_{ij}^{n^-}}{Pmax(n^-) - Pmin(n^-)}\right) * w^{n^-} + \sum_{n^+=1}^{N^+}\left(\frac{p_{ij}^{n^+} - Pmin(i,n^+)}{Pmax(n^+) - Pmin(n^+)}\right) * w^{n^+} \tag{3}$$

Concerning U_{ij}, $p_{ij}^{n^-}$ and $p_{ij}^{n^+}$ are the NFP values for each single attribute n of the NFP vector of service s_{ij}. The user can set up preferences (i.e., w^{n^-}, w^{n^+}) for each attribute n, where $0 < w^{n^-}, w^{n^+} < 1$ and $\sum_{n^-=1}^{N^-} w^{n^-} + \sum_{n^+=1}^{N^+} w^{n^+} = 1$ hold. Based on this, the utility of a service composition can be computed by aggregating the utility of the selected services. To represent the user's requirements regarding the different aggregated NFP values of a service composition, we introduce the end-to-end constraints vector $Q_c = [Q_c^1, \dots, Q_c^N]$ containing a value for each attribute n.

Thus, using the notation presented above our optimization model is defined as:

$$\max_{x_{ij},y_{ik}} \sum_{S_i \in \Gamma} \sum_{s_{ij} \in S_i} \sum_{ws_{ik} \in BS_i} U\left(s_{ij}, p_{ij}(ws_{ik})\right) * x_{ij} * y_{ik} \quad (4)$$

$$Subject\ to: \sum_{s_{ij} \in S_i} x_{ij} = 1\ \forall S_i \in \Gamma;\ with\ x_{ij} \in \{0,1\} \quad (5)$$

$$\sum_{ws_{ik} \in BS_i} y_{ik} = 1\ \forall BS_i \in \Gamma;\ with\ y_{ik} \in \{0,1\} \quad (6)$$

$$\sum_{ws_{ik} \in BS_i} y_{ik} * \sum_{\left(s_{(i-1)j}, ws_{(i-1)k'}\right) \in \Theta_{ik}} \left(x_{(i-1)j} * y_{(i-1)k'}\right) = 1\ with\ i \in [2; I] \quad (7)[1]$$

$$\sum_{S_i \in \Gamma} \sum_{s_{ij} \in S_i} p_{ij}^m * x_{ij} \leq Q_c^n\ \forall m = 1, \dots, M;\ M \subseteq N \quad (8)$$

$$\sum_{S_i \in \Gamma} \sum_{s_{ij} \in S_i} \sum_{ws_{ik} \in BS_i} p_{ij}^o(ws_{ik}) * x_{ij} * y_{ik} \leq Q_c^n\ \forall o = 1, \dots, O;\ O \subseteq N \quad (9)$$

Considering the service classes S_i and belief states BS_i in execution route Γ as well as the respective services $s_{ij} \in S_i$ and world states $ws_{ik} \in BS_i$, the optimization model determines the decision variables (x_{ij}= 1 indicates that service s_{ij} is selected; x_{ij}= 0 that it is not and y_{ij}=1 indicates that world state ws_{ik} is selected to quantify the values of the context-aware attributes of s_{ij}; y_{ij}=0 that it is not) to maximize the accumulated utility of the selected services based on the selected world states (cf. term (4)). For each service class S_i and for each belief state BS_i exactly one service s_{ij} and one world state ws_{ik}, respectively have to be selected (cf. terms (5 - 6)). Further, term (7) assures that only world states are considered which are feasible with respect to at least one selected service world state combination in the preceding service class. Here, Θ_{ik} contains all possible combinations (i.e., $(s_{(i-1)j}, ws_{(i-1)k'})$) that participate in the creation of the world state ws_{ik}. At the same time the aggregated quantified NFP values of the service composition need to satisfy the end-to-end constraints Q_c^n for each attribute m and o (cf. terms (8 - 9))[2]. Please note, that in case functional equivalent routes exist, the services of the execution route creating the highest accumulated utility among all functional equivalent execution routes have to be selected for the execution of the process.

A Heuristic Technique for the Context-aware Service Selection

Based on the *Model Setup*, we present our heuristic technique and thus the contribution of the paper in this section.

In contrast to the size of a QoS-aware service selection problem (cf. *Background*), the size of a context-aware service selection problem needs to take another additional factor into account, which is the number of feasible world states K_i per belief state BS_i (cf. y_{ik} term (4)). Hence, its problem size is determined as $I * J_i * K_i * N$[3]. As a result, we expect every exact solution to the context-aware service selection problem to scale poorly with respect to the problem size. Addressing this challenge, we present our heuristic technique in the following. To design this technique, we use local selection, due to its low time complexity, and combine it with a decomposition approach (cf. Alrifai et al. 2012; Sun and Zhao 2012; Surianarayanan et al. 2014) to allow for a consideration of end-to-end constraints. Thus, our heuristic technique presented hereafter consists of the two major steps *decomposition* and *local selection*. For both steps, we further show how the effects of context interdependencies can be taken into account in a well-founded way. By doing so, we aim at achieving close to optimal selection results.

---

[1] Please note, that this constraint is mandatory as soon as *consecutive* context information is considered. In case that only *non-consecutive* context information is taken into account the following constraint is used instead: $\sum_{ws_{ik} \in BS_i} y_{ik} * \sum_{s_{(i-1)j} \in \Theta_{ik}} x_{(i-1)j} = 1\ with\ i \in [2; I]$ where Θ_{ik} is a set of services which are involved in the creation of ws_{ik}.

[2] Please note that for attributes n^+, the corresponding constraint has to be multiplied by -1 so that it holds that the aggregated NFP value is less than the given constraint.

[3] For simplicity reasons, we assume the same number of world states K_i for every belief state I as well as the same number of services J_i for very service class of the process considered.

A Context-aware Decomposition Approach

In order to make the consideration of end-to-end constraints by means of local selection possible, we first need to decompose them into local constraints. This is the idea of the *decomposition* step of our heuristic technique. For this purpose, we introduce $LQ_i = [LQ_i^1, \dots, LQ_i^N]$ as the local constraint vector for a service class S_i that includes a local constraint value for each attribute n. To determine LQ_i in an efficient manner, we apply two different procedures, where one is based on *bounds* and the other one is based on *global optimization*.

Indeed, situations for the context-aware service selection can exist, where the given end-to-end constraints do not have any effect on the solution space of a particular attribute n. For an attribute n^-, these situations can be characterized by the fact that the maximum NFP value, aggregated over all service classes, is less than the given end-to-end constraint and for an attribute n^+ it is the other way round. Formally speaking, these situations occur when the following conditions hold: $Pmax(n^-) < Q_c^{n^-}$ and $Pmin(n^+) > Q_c^{n^+}$, respectively. To deal with these situations in an efficient manner our first procedure *bounds* is used.

Procedure bounds: this procedure is applied for attributes n^- or n^+ where the corresponding condition $Pmax(n^-) < Q_c^{n^-}$ or $Pmin(n^+) > Q_c^{n^+}$ is satisfied. More precisely, we determine LQ_i^n for every $S_i \in \Gamma$ by using $Pmax(i, n)$ for an attribute n^- and $Pmin(i, n)$ for an attribute n^+, respectively. In doing so, we are able to determine the local constraints for an attribute n not only very fast but can additionally assure that the end-to-end constraints are decomposed in an optimal way.

Procedure global optimization: this procedure is applied for attributes n^- or n^+ where the corresponding condition $Pmax(n^-) < Q_c^{n^-}$ or $Pmin(n^+) > Q_c^{n^+}$ is violated. To assure an optimal decomposition of the end-to-end constraints, we model the decomposition problem as a MMKP, which is pretty similar to the one of a QoS-aware service selection problem. At first, we determine a number of quantified NFP values for each service class i and every attribute n (Step 1). These values represent candidates for being used as a local constraint during the local selection later on. Subsequently, we assign a benefit to each local constraint candidate which reflects its capability to serve as a local constraint (Step 2). Finally, we maximize the benefit of the local constraint candidates while satisfying the end-to-end constraints (Step 3). In the following these steps are described in greater detail.

Step 1: in our approach the local constraint candidates for a service class S_i are represented by the quantified NFP values p_{ij} that can emerge due to the possible combinations of $s_{ij} \in S_i$ and $w_{ik} \in BS_i$. We use these values because they have a direct influence on the feasibility of a service composition (cf. term (8 - 9)). Therefore, we introduce the quantified NFP vector $SP_{in} = [sp_{in}^1, \dots, sp_{in}^V]$ $(with\ V = K_i * J_i)$ for service class S_i that includes all quantified values of attribute n, sorted in ascending order. Based on this vector, we determine for each $S_i \in \Gamma$ and attribute n, a number Υ ($\Upsilon \in \mathbb{N}$) of local constraint candidates which are denoted as lcc_{in}^{φ} $(with\ \varphi = 1\ to\ \Upsilon)$ in the following. The procedure for this is as follows:

I. The first two local constraint candidates are determined by using the minimum and maximum quantified NFP values of an attribute. Consequently, $lcc_{in}^1 = sp_{in}^1$ $(with\ sp_{in}^1 \in SP_{in})$ for the minimum value and $lcc_{in}^{\Upsilon} = sp_{in}^V$ $(with\ sp_{in}^V \in SP_{in})$ for the maximum value.
II. The remaining values of SP_{in} are divided in $\Upsilon - 2$ equal subsets.
III. For each subset a quantified NFP value is randomly selected and used as a local constraint candidate.

Step 2*:* for an attribute n and a service class S_i, we now have to decide which local constraint candidate should serve as a local constraint. For this purpose, we assign a benefit to each local constraint candidate. Without considering any context interdependencies in the first place, this benefit should reflect the capability of the local constraint candidate to allow for close to optimal selection results in each service class during the local selection. Thus, we define the benefit $B(lcc_{in}^{\varphi})$ of lcc_{in}^{φ} initially as:

$$B(lcc_{in}^{\varphi}) = \frac{\widehat{U}_{max}(lcc_{in}^{\varphi})}{U_{max}(i)} \tag{10}$$

Concerning $B(lcc_{in}^{\varphi})$, $\widehat{U}_{max}(lcc_{in}^{\varphi})$ is the maximum utility that can be achieved if lcc_{in}^{φ} is used as a local constraint in S_i. This utility is considered in relation to $U_{max}(i)$ that is the maximum utility in S_i.

For the context-aware service selection this, however, is not sufficient. Setting a local constraint in a service class also affects the utility and the feasibility of services in the succeeding service class. The reason for this can be found in the effects of context interdependencies. More precisely, services contribute to the creation of world states (cf. *Model Setup*) for a succeeding service class. Excluding services due to a local constraint thus automatically reduces the number of feasible world states that can be taken into account during local selection (cf. line 6, Table 2). However, as the quantified values of context-aware attributes and thus the utility of a service dependents on the considered world state (cf. term (4)), setting a local constraint in a service class consequently affects the capability of the local selection to achieve close to optimal selection results in the succeeding service class. Therefore it is obvious that the effects of context interdependencies should already be taken into account during the decomposition. Hence, we extend the benefit of a local constraint candidate (cf. term (10)) in the following way:

$$B\left(lcc_{in}^{\varphi}\right) = \frac{\widehat{U}_{max}\left(lcc_{in}^{\varphi}\right)}{U_{max}(i)} * \frac{\hat{A}_{abs}\left(SP_{in}, lcc_{in}^{\varphi}\right)}{A_{abs}(SP_{in})} * \frac{\widehat{K}_{abs}\left(BS_{i+1}, lcc_{in}^{\varphi}\right)}{K_{abs}(BS_{i+1})} * \frac{\widehat{U}_{max}(i+1)}{U_{max}(i+1)} \tag{11}$$

Concerning $B\left(lcc_{in}^{\varphi}\right)$, $\hat{A}_{abs}\left(SP_{in}, lcc_{in}^{\varphi}\right)$ is the number of quantified NFP values in SP_{in} that would qualify if lcc_{in}^{φ} is used as a local constraint in S_i. This number is considered in relation to $A_{abs}(SP_{in})$ that is the absolute number of quantified NFP values in SP_{in}. With this factor we aim to increase the number of quantified NFP values (i.e., services) available for the local selection. Moreover, the third factor is used to increase to number of feasible world states in the succeeding service class. Here, $\widehat{K}_{abs}\left(BS_{i+1}, lcc_{in}^{\varphi}\right)$ is the number of world states in BS_{i+1} that are feasible if lcc_{in}^{φ} is used as a local constraint in S_i. This number is considered in relation to $K_{abs}(BS_{i+1})$ that is the absolute number of world states in BS_{i+1}. Finally, $\widehat{U}_{max}(i+1)$ is the maximum utility that can be achieved in S_{i+1} with respect to the feasible world states in BS_{i+1}. Again, this utility is considered in relation to $U_{max}(i+1)$ that is the maximum utility that can be achieved in S_{i+1} based on all world states in BS_{i+1}. Please note that for the last service class of the process considered, the usage of term (10) is sufficient, as no further dependencies to a succeeding service class can exist. After defining the benefit of local constraint candidates (cf. term (11)), we can now proceed with the decomposition of the end-to-end constraint.

Step 3: based on the notation presented above, our optimization model is defined as follows:

$$\max_{x_{in}^{\varphi}} \sum_{S_i \in \Gamma} \sum_{n=1}^{N} \sum_{\varphi=1}^{\Upsilon} B\left(lcc_{in}^{\varphi}\right) * x_{in}^{\varphi} \tag{12}$$

$$Subject\ to\text{:} \sum_{S_i \in \Gamma} \sum_{\varphi=1}^{\Upsilon} lcc_{in}^{\varphi} * x_{in}^{\varphi} \leq Q_c^n\ \forall n = 1, .., N \tag{13}$$

$$\sum_{\varphi=1}^{\Upsilon} x_{in}^{\varphi} = 1\ with\ x_{in}^{\varphi} \in \{0,1\}\ \forall\ S_i \in \Gamma;\ \forall n = 1, .., N \tag{14}$$

Considering the service classes S_i included in the execution route Γ as well as the corresponding attributes N and local constraints candidates Υ, the optimization model determines the decision variables x_{in}^{φ} ($x_{in}^{\varphi} = 1$ indicates that local constraint candidate lcc_{in}^{φ} is selected; $x_{in}^{\varphi} = 0$ that it is not) to maximize the accumulated benefit of the selected local constraint candidates (cf. term (12)). For each service class S_i and for each attribute n exactly one local constraint candidate lcc_{in}^{φ} has to be selected (cf. term (14)). At the same time, the aggregated quantified NFP values of the local constraint candidates need to satisfy the end-to-end constraints Q_c^n[4] for each attribute n (cf. term (13)). The result is a set of local constraints for each service class $S_i \in \Gamma$ and attribute n, which is used during the local selection later on.

The resulting problem size of the decomposition problem (cf. term (12 – 14)) is determined by $S_i * N * \Upsilon$ and thus independent of the number of services J_i as well as the number of world states K_i. As a result, we

[4] For attributes that have to be maximized the corresponding constraint has to be multiplied with minus one so that it holds that the aggregated quantified values of local constraint candidates are less than the given end-to-end constraints.

expect our decomposition procedure *global optimization* to be very scalable regarding the problem size of a context-aware service selection problem.

To conclude, in this subsection we presented two decomposition procedures that can be used to decompose the end-to-end constraints into local ones. The procedure *bounds* is not only expected to improve performance. It also prevents the optimization model in the procedure *global optimization* from being biased by attributes for which the solution space is not limited by the end-to-end constraints. Considering these attributes in the optimization can lead to inferior selection results. This would consequently affect our local selection in a negative way, regarding its capabilities of finding a feasible as well as close to optimal solution.

Local Selection

With the local constraints on hand, we can now proceed to select services by means of *local selection*. Besides its low time complexity, local selection comes with another important capability that helps to prevent an unnecessary increase of the problem size. So far, for the creation of world states (cf. *Model Setup*) it was assumed that context information is changed for every considered service (e.g., the GPS position), which we refer to as *continuous* context information. However, in real life there is context information (e.g., context information *A* cf. *Model Setup*, p. 5) where this assumption partially holds. Consequently, we refer to it as *discontinuous* context information. As a result, the representation of discontinuous context information via states (i.e., belief and world states) is still feasible, but comes with the following challenge for the problem size that can be illustrated with a simple example. We consider a sequential process with I service classes and J services for each service class. As context-aware attributes, we use one, represented by the distance type (e.g., GPS position), which is of relevance for each service class. The other is represented by the *Boolean* type, which we require to be of relevance only for a few, say v, of the I service classes. The GPS position is represented using a *non-consecutive* state variable (cf. *Model Setup*), whereas the Boolean type attribute must be represented using a *consecutive* state variable. Hence, in each existing world state the GPS attribute is simply updated, resulting in no additional world states. For the Boolean type attribute, however, two world states are created from each preceding state. One for each value *true* and *false*, although the information carried along in the world state is not necessarily needed for every service class. After using the Boolean type attribute for the v-th service class, we have created 2^*v times as many world states as without the attribute, according to all possible combinations of true and false in the preceding service classes. For a context-aware attribute taking m different values instead of 2, one would create v^*m times more world states than actually necessary. As a result, the number of decision variables (i.e., K_i) in the optimization model (cf. term (4)) would be increased by v^*m which influence the computation time of any selection approach in a negative way. For instance, it would be more useful to consider discontinuous context information right during the selection procedure rather than representing it via states. Local selection offers the needed capabilities for this, as it allows firstly to track emerging context information (e.g., for context information A - how many times has a category been selected so far) and secondly for a quantification of the corresponding context-aware attributes right before the actual selection of services.

For context-aware service selection, the basic idea of using local selection is to determine for each $S_i \in \Gamma$ the world state ws_{ik}^* service s_{ij}^* combination (here "*" symbolizes that a world state or service has been selected) that creates the highest utility $U(s_{ij}^*, p_{ij}(ws_{ik}^*))$ and which is feasible subject to the local constraints $LQ_i \; \forall \, n = 1, \dots, N$. However, selecting services for each service class individually neglects the effects of context interdependencies. We contribute to this challenge by presenting a *re-selection routine* that tries to improve the utility of the service composition by reconsidering the initial selected solution for each service class subject to the feasible context information (i.e., world states). Finally, we illustrate how discontinuous context information can be considered within the local selection. For the latter and w.l.o.g., we use the context information *A* (cf. *Model Setup*).

In the following, the procedure of the local selection and the re-selection routine are discussed in general and illustrated by means of an example (cf. Figure 1). For reasons of comprehensibility, we focus on the GPS position as the only context-aware attribute as well as on the utility of a service s_{ij}. Moreover, as the GPS position is the only attribute considered, we leave out the normalization in the utility function provided in term (3). Nevertheless, local selection can be conducted in the similar way for other non-context-aware and context-aware attributes. The path with the solid black line in Figure 1 represents the feasible result of the local selection so far. An algorithm of the local selection is presented in Table 2.

Table 2. Local selection	
Input	Local constraint vector LQ_i for each service class S_i
	Tally vector A
Output	Service composition
1	**For** $i = 1$ **to** I
2	**If** $i = 1$
3	**Then** select the world state ws_{ik}^* $(\forall k = 1, \dots, K_i)$ service s_{ij}^* combination resulting in the highest utility $U(s_{ij}^*, p_{ij}(ws_{ik}^*))$ and which is feasible subject to $LQ_i^n \ \forall n = 1, \dots, N$
4	Update A considering c of s_{ij}^*
5	Update utility of every $s_{ij} \in S_{i+1}$ taking A into account
6	**Else** select the world state ws_{ik}^* $(W: ws_{(i-1)k}^* \times s_{(i-1)j}^* \rightarrow ws_{ik}^*)$ service s_{ij}^* combination resulting in the highest utility $U(s_{ij}^*, p_{ij}(ws_{ik}^*))$ and which is feasible subject to $LQ_i^n \ \forall n = 1, \dots, N$
7	Update A considering c of s_{ij}^*
8	Update utility of every $s_{ij} \in S_{i+1}$ taking A into account
9	**Trigger** *Re-selection routine* (cf. Table 3)
10	**If** re-selection was successfully
11	**Then** replace the already selected service $s_{(i-1)j}^*$ and world state ws_{ik}^* with substitute service $s_{(i-1)j}$ and world state ws_{ik} determined by the re-selection routine
12	**Else** keep the selected service $s_{(i-1)j}^*$ and world state ws_{ik}^*
13	**End If**
14	**End If**
15	**End For**
16	**Return** service composition

Starting with the initial state and the first service class, the algorithm selects the service s_{ij}^* resulting in the highest utility (cf. line 3 in Table 2). In the illustrating example, this is ws_{11}^* and s_{11}^* with a utility of $U(s_{11}^*, p_{11}(ws_{11}^*)) = -300$. Update routines are triggered after each selection step (cf. line 4, 5 and 7, 8 in Table 2) to consider discontinuous context information. The first routine (cf. line 4, 7) is meant to update the tally vector A under the consideration of the category c of s_{ij}^*. The second routine (cf. line 5, 8) is meant to update the utility values of all services in the succeeding service class while taking context information A into account. It is obvious that the update routines are unnecessary if context-aware attributes are not affected by the corresponding context information or in case the last service class is considered.

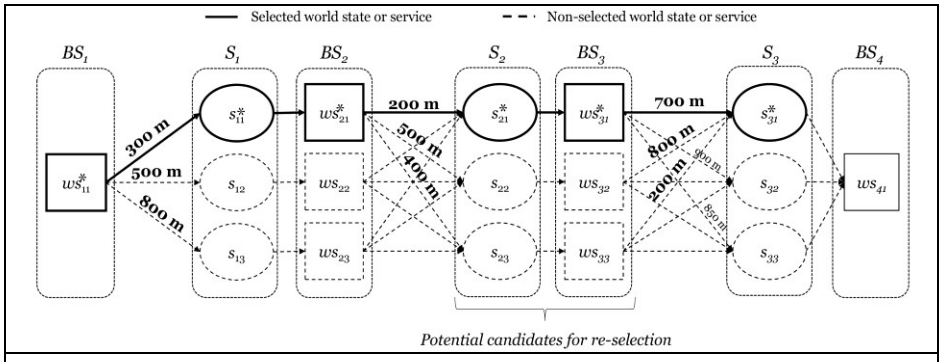

Figure 1. Illustrating Example of the Local Selection

In line 6, $W: ws_{(i-1)k}^* \times s_{(i-1)j}^* \rightarrow ws_{ik}^*$ assures that only the world state is taken into account, which is feasible (cf. *Model Setup,* p. 5) with respect to the already selected world state $ws_{(i-1)k}^*$ service $s_{(i-1)j}^*$ combination. In the example above, this is world state ws_{21}^* due to the initial selected world state service combination in S_1. Based on this, the local selection selects service s_{21}^* in S_2 and subsequently s_{31}^* in S_3 (cf. line 6 in Table 2), as those are the services which result in the highest utility (i.e., $U(s_{21}^*, p_{21}(ws_{21}^*)) = -200$ and $U(s_{31}^*, p_{31}(ws_{31}^*)) = -700$) after considering the feasible world states ws_{21}^* and ws_{31}^*. However, due to the effects of context interdependencies, we cannot assure that the local selection is able to select the world state service combination within a service class having the highest utility. For instance, in our illustrating example above, the selection of service s_{31}^* in combination with ws_{33} would result in a higher utility (i.e., $U(s_{31}^*, p_{31}(ws_{33})) = -200$). Hence, to take the effects of context interdependencies into account, it would be useful for the local selection to reconsider if the initial selected service s_{31}^* results in a higher utility when

being combined with an alternative world state ws_{ik} ($ws_{ik} \neq ws^*_{ik}$). To reach an alternative world state in BS_3 (e.g., ws_{32} or ws_{33}) a re-selection of s^*_{21} is required. Hence, the question arises if a re-selection of ws^*_{21} should take place to potentially achieve even a higher utility for service classes S_2. These recursive extension can be continued until the initial state is reached. The resulting size of such a re-selection problem is given by $I * J_i * K_i * N$. As the re-selection is triggered after every service class, we argue that re-considering the entire initial selected solution (i.e., service composition) is not reasonable with respect to the computation time. Furthermore, in dependence of the considered type of context information most of the context interdependencies mainly occur between two succeeding service classes. For context information, where the corresponding value is represented by *non-consecutive* state variables, context interdependencies exist only with respect to the last selected service $s^*_{(i-1)j}$. For context information, where the corresponding value is represented by *consecutive* state variables, context interdependencies exist with respect to the last selected service $s^*_{(i-1)j}$ and the last selected world state $ws^*_{(i-1)k}$. A re-selection of $ws^*_{(i-1)k}$ is, however, with respect to the computation time not useful, as the discussion above shows.

Consequently, we fix the already selected service s^*_{31} (i.e., s^*_{ij}) and world state ws^*_{21} (i.e., $ws^*_{(i-1)k}$), and under these conditions try to improve the initial utility of the service composition by re-selecting s^*_{21} (i.e., $s^*_{(i-1)j}$) and ws^*_{31} (i.e., ws^*_{ik}). This is the idea of the *re-selection routine* (cf. Table 3).

Table 3. Re-Selection Routine

Input	selected services $s^*_{ij}, s^*_{(i-1)j}$ and world states $ws^*_{ik}, ws^*_{(i-1)k}$; $u_{cur} \leftarrow U(s^*_{(i-1)j}, p_{(i-1)j}(ws^*_{(i-1)k})) + U(s^*_{ij}, p_{ij}(ws^*_{ik}))$
Output	true; false
1	$sws \leftarrow$ feasible substitute world states
2	$higher\_utility\_found \leftarrow false$
3	**If** $sws \neq \emptyset$
4	**Then**
5	**For each** $ws_{ik} \in sws$
6	**If** $U(s_{(i-1)j}, p_{(i-1)j}(ws^*_{(i-1)k})) + U(s^*_{ij}, p_{ij}(ws_{ik})) > u_{cur}$
7	**Then** $u_{cur} \leftarrow U(s_{(i-1)j}, p_{(i-1)j}(ws^*_{(i-1)k})) + U(s^*_{ij}, p_{ij}(ws_{ik}))$; $higher\_utility\_found \leftarrow true$
8	**End If**
9	**End For**
10	**End If**
11	**Return** $higher\_utility\_found$

The algorithm starts by determining all world states ws_{ik} ($ws_{ik} \neq ws^*_{ik}$), which can serve as a substitute for ws^*_{ik} (cf. line 1 in Table 3). For this purpose, world states need to satisfy all of the following three conditions:

a. First, they must be feasible with respect to at least one combination of $ws^*_{(i-1)k}$ and $s_{(i-1)j}$ ($s_{(i-1)j} \neq s^*_{(i-1)j}$), whereas the quantified values that emerge due to this combination must be feasible with respect to $LQ_{(i-1)}$ as well.
b. Second, the quantified values of s^*_{ij} that emerge in combination with the substitute world state ws_{ik} must be feasible with respect to LQ_i.
c. Third, the utility of the new world state ws_{ik} service s^*_{ij} combination must be greater than the initial one $U(s^*_{ij}, p_{ij}(ws_{ik})) > U(s^*_{ij}, p_{ij}(ws^*_{ik}))$.

In our illustrating example above, both of the substitute world states ws_{32} and ws_{33} would satisfy condition *a.*, as world state ws_{32} is feasible with respect to the combination of ws^*_{21} with s_{22}, and world state ws_{33} is feasible with respect to the combination of ws^*_{21} with s_{23}. Moreover, if $LQ^{distance}_2$ is given by 600 m, both world states also satisfy the second part of condition *a.*, as $p_{22}(ws^*_{21}) = 500\ m$ and $p_{23}(ws^*_{21}) = 400\ m$. In a similar way condition *b.* can be evaluated, where we assume that both world states qualify. However, world state ws_{33} violates condition *c.*, as $U(s^*_{31}, p_{31}(ws_{32})) = -800$ is inferior against the utility of the already selected combination $U(s^*_{31}, p_{31}(ws^*_{31})) = -700$. Consequently, it is not added to sws. In contrast to that, ws_{33} qualifies and thus is added to sws. In case no world state is found that satisfies these conditions, the algorithm returns false (cf. line 2, 3 and 11 in Table 3).

In case the condition $sws \neq \emptyset$ is satisfied, the re-selection routine tries for each $ws_{ik} \in sws$ to improve u_{cur} (i.e., $U(s^*_{(i-1)j}, p_{(i-1)j}(ws^*_{(i-1)k})) + U(s^*_{ij}, p_{ij}(ws^*_{ik}))$) by re-selecting $s^*_{(i-1)j}$. In our illustrating example above, the only feasible alternative for ws^*_{31} is ws_{33}. With the usage of ws_{33} in combination with s_{23} the re-selection routine is able to find an alternative world state service combination that improves the current

utility u_{cur} (i.e., before $u_{cur} = U(s_{21}^*, p_{21}(ws_{21}^*)) + U(s_{31}^*, p_{31}(ws_{31}^*)) = -900$; after re-selection $U(s_{23}, p_{23}(ws_{21}^*)) + U(s_{31}^*, p_{31}(ws_{33})) = -600$). Consequently, the local selection replaces the initially selected world state service combination with the alternative one and proceeds by taking the next service class into account. After the last service class the local selection returns the selected service composition (cf. line 16 in Table 2).

Evaluation

In this section, we evaluate the contribution of the proposed heuristic technique to the research question of this paper. We therefore validate in a first step *(i)* the feasibility of the hybrid decomposition approach as well as the local selection algorithm. Then, we evaluate (*ii*) the influence of the number of local constraint candidates on the efficiency (i.e., scalability and solution quality) of the heuristic technique. Finally, we evaluate *(iii)* the efficiency of our heuristic technique.

Feasibility of the Decomposition Approach and Local Selection (i)

We prototypically implemented the proposed hybrid decomposition approach and the algorithm for the local selection. We ensured the validity of our heuristic technique by making a series of tests using the JUnit Framework, including runs with extreme values, regression tests and unit tests. We thereby checked the decomposed end-to-end constraints as well as the selected services and world states in detail for different constellations of non-context-aware and context-aware attributes, different numbers of local constraint candidates as well as different constraint values. Furthermore, persons others than programmers validated the source code via structured walk through. The implemented algorithm did not show any defects at the end of this test phase.

Evaluation Methodology for Step (ii) and (iii)

To evaluate *(ii)* the influence of the number of local constraint candidates on the efficiency as well as *(iii)* the efficiency of our proposed heuristic technique, we conducted numerous simulations which are described in the following.

Datasets: we used two different datasets, an artificial one (AD) and a real-world data set (RD) which originates from the tourism domain. Here, we used *Google Places*[5] to determine suitable services[6] as well as the values for the NFP (which are described in the following). In total the real-world data set comprises 1,400 services.

Non-functional properties: for both datasets, we used the duration and the recommendation value as *non-context-aware attributes*, whereas price (discount type), business hours (Boolean type) and GPS position (distance type) are used as *context-aware attributes*[7].

Selection approaches: to resolve the optimization model given in terms (4 – 9), we used three different selection approaches which are described in the following:

- EXACT: this is a global optimization approach (cf. Heinrich and Lewerenz 2015) which is based on integer programming. The selection approach returns the optimal service composition under the consideration of the effects of context interdependencies as well as end-to-end constraints and thus serves as a benchmark against which the other selection approaches are evaluated.
- HT: this is our heuristic technique that is presented throughout this paper and is expected to enable an efficient context-aware service selection while taking the effects of context interdependencies and end-to-end constraints into account.

---

[5] http://www.programmableweb.com/api/google-places, accessed September 2015

[6] Please note that the services provided by *Google Places* have to be understood as an information/service object (cf. Hinkelmann et al. (2013); Dannewitz et al. (2008)) representing a real-life entity. Those service objects can be further denoted by NFP (e.g., GPS position) of the real life entity (for further information, we kindly refer to Heinrich and Lewerenz (2015)). All used service objects (e.g., "*Die Pinakotheken*", "*Deutsches Musuem",* "*Hofbräuhaus",* etc.) originate from the city of Munich, Germany.

[7] The corresponding services and their NFP can be made available upon request.

- ACO: this is a heuristic technique which is based on ant colony optimization (cf. Dorigo and Caro 1999) and proposed by Wu and Zhu (2013). It is capable of considering the effects of context interdependencies and end-to-end constraints.

Parameterization of heuristics: both heuristic techniques (HT, ACO) need certain input parameters. In our evaluation we used the following parameterization:

- HT: the only parameter our heuristic technique needs is the number of local constraints candidates Υ. For the evaluation, we set this parameter to $\Upsilon = 10$. Please note that the influence of this parameter on the efficiency is evaluated separately (cf. step (*ii*)).
- ACO: for the parameterization of this heuristic technique, we used values of existing literature (cf. Li and Yan-xiang 2012; Qiqing et al. 2009; Wu and Zhu 2013). Consequently, we set *alpha* = 1.0; *beta* = 1.0; *dilution* = 0.3; *iterations* = 100 and the number of ants a = 40.

Efficiency: the efficiency of the heuristic techniques (HT, ACO) is evaluated by means of their *optimality* and *time usage* which are defined in the following. The aim of the *optimality* is to evaluate the capability of HT and ACO to determine close to or even optimal solutions (i.e., optimal service composition). Therefore, it is defined as follows:

$$Optimality_{HT} = \frac{(U_{HT} - U_{min}^{EXACT})}{(U_{max}^{EXACT} - U_{min}^{EXACT})};\ Optimality_{ACO} = \frac{(U_{ACO} - U_{min}^{EXACT})}{(U_{max}^{EXACT} - U_{min}^{EXACT})} \tag{15}$$

Concerning term (15), U_{max}^{EXACT} is the utility of the optimal service composition, whereas U_{min}^{EXACT} is the utility of the worst-case service composition that can be determined when the objective given in term (4) is minimized while satisfying the constraints given in terms (5 - 9). The denominator (i.e., the distance between the optimal and the worst utility) is used to normalize the utility determined by the two heuristics (U_{HT}, U_{ACO}). Thus, the *optimality* equals "1", when the heuristic techniques determine the optimal service composition and equals "0" when the heuristic techniques determine the worst-case service composition. The aim of the *time usage* is to determine the proportion of time needed by the two heuristic techniques to resolve the optimization model given in terms (4 – 9) as compared to the time needed for the EXACT approach. Hence, the *time usage* is defined as:

$$time\ usage_{HT} = \frac{CT_{HT}}{CT_{EXACT}};\ time\ usage_{ACO} = \frac{CT_{ACO}}{CT_{EXACT}} \tag{16}$$

Concerning term (16), CT_{HT}, CT_{EXACT} and CT_{MOACO} are the computation times of the different selection approaches.

Scenarios: we used three different scenarios, in which we changed one parameter of the problem size while keeping all other parameters constant (ceteris paribus). All scenarios are based on a sequential process. The first scenario is used to evaluate the influence of the number of local constraint candidates Υ on the efficiency (cf. step (*ii*)) whereas the next two scenarios are used to evaluate efficiency (cf. step (*iii*)).

I In the first scenario, the number of service classes I is 5, the number of services J_i is 50 and Υ is increased in steps of 5 from 5 to 25.

II In the second scenario, the number of service classes I is 10 and the number of services J_i is increased in steps of 10 from 10 to 100.

III In the third scenario, the number of services J_i is 10 and the number of service classes is increased in steps of 3 from 3 to 30.

For all three scenarios, we used the artificial (AD) as well as the real-world data set (RD). We further used a vector containing five randomly created values to represent the end-to-end constraints regarding the attributes specified above. To determine possible context information regarding a service composition, we used the state creation algorithm presented in Heinrich and Lewerenz (2015). This result serves as a basis upon which the different selection approaches (EXACT, HT and ACO) are applied. Please note that as with the method of integer programming the consideration of discontinuous context information is not possible during the selection procedure, we decided to leave them unconsidered in order to achieve comparability between the selection approaches. We simulated each scenario 100 times and determined the average *time usage* as well as the average *optimality*. All analyses were conducted on a machine with an Intel Core I7 processor with 3.6 GHz, 32 GB RAM, Java 1.8, and Gurobi 6.0.

(ii) Influence of the Number of Local Constraints Candidates on the Efficiency

Based on the optimization model given in terms (12 – 14), we expect the number of local constraint candidates (ϒ) to influence the *optimality* and *time usage* of our heuristic technique in two ways. First, with growing number of ϒ the chances to achieve close to optimal selection results (i.e., *optimality*) are increased. This is due to the fact that with a higher number of ϒ a better consideration of the effects of context interdependencies can be achieved during the decomposition of the end-to-end constraints. Second, with a growing number of ϒ, the computation time and thus the *time usage* of HT will increase. The reason for that can be found in the increased number of decision variables (i.e., x_{in}^{φ}) in the optimization model. Consequently, solving the optimization model will usually take longer. Indeed, the results of our simulation experiment for scenario I (cf. Figure 2) confirm these reflections. With a growing number of ϒ, the optimality but also the time usage increase. Thus, there is a trade-off between optimality and time usage that needs to be considered when setting ϒ.

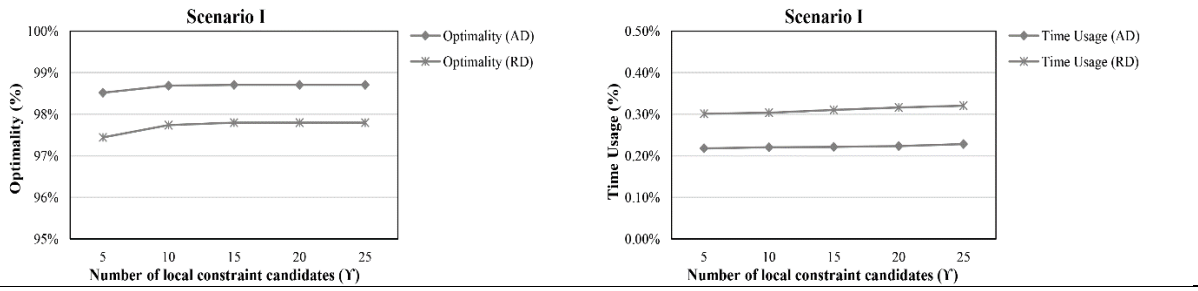

Figure 2. Efficiency of HT in Dependence of the Number of Local Constraints Candidates

(iii) Efficiency of the Heuristic Technique

We start by discussing the results on *time usage* (cf. Figure 3 b, d): the first thing that can be noticed, is that our heuristic technique HT achieves superior results in comparison with ACO with regard to the *time usage*. This could be found for scenario II as well as for scenario III. With a value of at most 8% at the beginning, the *time usage* decreases very quickly to a value under 1% as the process size is increased. For ACO, the results are different. Here, the graphs show that for small process sizes the computation time of ACO is even higher than the one of EXACT (i.e., *time usage* > 100%). Yet, when the process size is increased the *time usage* of ACO steadily decreases, ending up with a value of at most 48%. The differences between HT and ACO may be accounted for by the following. ACO is based on a directed acyclic graph (cf. *Background*) that represents the entire solution space. Thus, searching through the entire solution space even with just a limited number of ants maybe very time consuming (cf. Table 4 in *Appendix*). As opposed to that, our HT benefits from the low time complexity of local selection combined with an efficient re-selection routine to consider the effects of context interdependencies. Here, the results show that our idea to consider just a single service class for re-selection works properly with regard to the *time usage* of HT (effects on *optimality* will be discussed later). Indeed, this is also reflected in the absolute numbers of the computation time of the three selection approaches (cf. Table 4 in *Appendix*). Here, it shows that the computation time of our approach (HT) grows in a moderate way in case the problem size is increased. Even for larger problem sizes our approach only took at most 0.23 sec to determine a result. Thus, the results provide evidence that our heuristic technique is able to scale properly with the process size. Opposed to that, the computation time of EXACT increases in an over-proportional fashion in case the problem size is increased. In the worst case scenario, EXACT took on average over 26 min. (cf. scenario II – artificial data set in Table 4) to determine the optimal service composition.

Next, we discuss the results on *optimality* (cf. Figure 3 a, c): In particular, ACO is able to consider the effects of context interdependencies by the usage of a number of ants that traverse the graph from the start to the end node. Hence, we expect ACO to achieve good selection results for small process sizes. However, when the process size is increased, we expect that ACO loses its ability to achieve close to optimal selection results. One reason for this is that the capability of the approach to find existing context interdependencies is decreased when the solution space is increased (due to an increased process size). As opposed to that, with the usage of the re-selection routine, our heuristic technique is able to consider all existing context

interdependencies between two service classes. Hence, we expect that HT will achieve close to optimal selection results even for bigger process sizes. Indeed, the results of our simulation experiment confirm these ideas. It shows that for small process sizes, ACO is able to determine superior results in comparison to our heuristic technique HT. However, these results come with a *time usage* > 100% (cf. Figure 3 b, d) which contradicts the basic idea of a heuristic. With increasing problem size, our heuristic technique HT is able to achieve superior results in comparison to ACO. Even for the biggest process sizes with many existing context interdependencies its achieved optimality is on average above 98%. As opposed to that, the achieved *optimality* of ACO decreases as the process size increases. These results clearly illustrate that our idea of considering the effects of context interdependencies, by means of our re-selection routine, is not only very fast with respect to the *time usage* but also works appropriately with regard to the *optimality*.

To conclude, the results of our simulation experiment show that our heuristic technique can contribute to the research question of this paper. Besides the consideration of end-to-end constraints, it achieves close to optimal selection results (*optimality > 98%*) by taking into account the effects of context interdependencies at a fraction of computation time of an exact solution (*time usage* < 1%). Thus, it can serve as an efficient decision support in context-aware service selection.

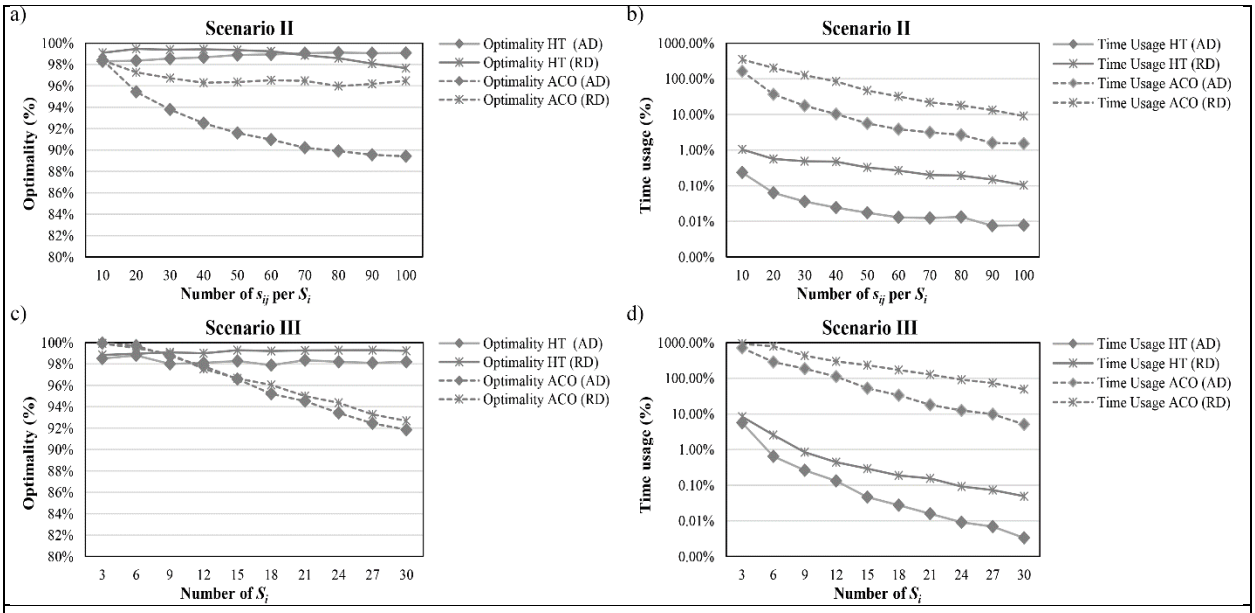

Figure 3. Time Usage and Optimality Heuristic Technique

Conclusion, Limitations and Future Research

In this paper, we presented a heuristic technique that is able to solve the context-aware service selection problem which is known to be NP hard. This technique determines close to optimal selection results at a low computation time while taking the effects of context interdependencies as well as end-to-end constraints into account. Here, existing context-aware service selection approaches do not aim at addressing such a decision support, as they either disregard the effects of context interdependencies, leave end-to-end constraints unconsidered, or provide exact solutions, which are hardly applicable due to their high time complexity.

We address this research gap in this paper. To do so, we designed a heuristic technique that consists of the two steps *decomposition* and *local selection*. For step one, we presented a hybrid decomposition approach based on the procedures bounds and global optimization that aims at decomposing the end-to-end into local constraints. We thereby showed how the effects of context interdependencies can be considered in a well-founded way. For step two, we proposed a local selection approach which uses the decomposed end-to-end constraints as well as a re-selection routine to consider the effects of context interdependencies. Based on that, we feel confident that our approach contributes to an efficient decision support in context-aware service selection. This was also reinforced by our evaluation, where we could illustrate by means of a

real-world data that our heuristic technique is able to determine close to optimal selection results (i.e., optimality > 98%) at a fraction of the computation time of an exact solution (i.e., time usage < 1%).

Moreover, our results also provide important practical implications. As the effects of context interdependencies have a direct influence on the resources (e.g., time and money) used for the execution of processes (cf. Heinrich and Lewerenz 2015), their consideration within a context-aware service selection approach is crucial to avoid an unnecessary resource consumption. In particular, for bigger process sizes, determining the optimal service composition by means of an exact solution can be very time consuming (cf. *Evaluation*). This may not be feasible for many real-world processes, such as travel booking processes (cf. Dai et al. 2009; Hwang et al. 2008; Li et al. 2011; Yang et al. 2009; Zeng et al. 2008). On the one hand, travel agencies want to optimize the use of their resources (e.g., money spent for the execution of the process equivalent to the end-to-end price of the service composition). On the other hand, customers are usually not willing to wait several minutes (i.e., time needed to determine the optimal service composition) before the execution of the travel booking process has been initiated. Overall, our heuristic technique is expected to contribute to such challenges. With an average optimality of 98% (in the observed cases), it avoids an unnecessary resource consumption (e.g., money) by considering the effects of context interdependencies. This is especially important if a process and thus the service composition is executed many times. At the same time, it also allows for a fast decision support (i.e., computation time < 0.3 sec. in the observed cases) and scales very efficiently with the process size.

Besides the highlighted contribution, our approach is also subject to limitations. First, our evaluation has shown that the number of local constraint candidates (Υ) can have an influence on the *optimality* and *time usage* of our heuristic technique. Determining the right value of this parameter thus may be crucial for its efficiency. Due to the very low computation time of our heuristic technique (cf. Table 4 in *Appendix*), a possible solution to this challenge could be to run different instances of the heuristic technique with different number of Υ at the same time. However, further research is needed to clarify how the user can be supported in setting the value of local constraint candidates in a well-founded way. Second, so far our heuristic technique is only able to consider context information that is directly influenced by the considered services of a service composition. As our evaluation of the real-world data illustrates, this is sufficient to provide an efficient decision support. However, endogenous context information (e.g., power failure, network errors or the weather) can exist which is not dependent on services, yet could influence the execution of the process in a negative way. Here, further research is necessary to see how such endogenous context information may be considered within our heuristic technique. This could be achieved either during planning or during runtime. At planning time, methods of simulation (cf. Meredith et al. 1989) could be used to select the optimal service composition based on an expected utility calculus (cf. Heinrich et al. 2015). During runtime, the service composition could be adapted dynamically to endogenous context information by means of re-selection approaches (cf. Canfora et al. 2008; Lin et al. 2010). In case the conduction of the process is support by a mobile device, the logical and physical sensors of these devices (cf. Baldauf et al. 2007) can be used to support such adaptions.

Overall, our research constitutes a first but important step in providing an efficient decision support in context-aware service selection. Beyond that, we hope that it will open doors for further research in this exciting research area.

Appendix

Table 4. Absolute Computation Times in [sec] – EXACT, HT and ACO

Scenario II							**Scenario III**						
	Artificial Data Set			Real-World Data Set				Artificial Data Set			Real-World Data Set		
s_{ij}	*EXACT*	*HT*	*ACO*	*EXACT*	*HT*	*ACO*	S_i	*EXACT*	*HT*	*ACO*	*EXACT*	*HT*	*ACO*
10	1.520	0.004	2.456	0.542	0.006	1.879	3	0.029	0.002	0.209	0.020	0.002	0.186
20	12.875	0.008	4.684	1.921	0.011	3.877	6	0.298	0.002	0.854	0.092	0.002	0.730
30	39.448	0.014	6.875	4.669	0.023	5.860	9	1.050	0.003	1.952	0.376	0.003	1.630
40	89.988	0.022	9.191	9.436	0.045	7.891	12	3.139	0.004	3.484	0.966	0.004	2.877
50	210.121	0.036	11.740	21.353	0.069	9.993	15	10.143	0.005	5.347	1.907	0.006	4.448
60	374.620	0.048	14.317	37.635	0.100	12.077	18	22.880	0.006	7.660	3.835	0.007	6.593
70	535.334	0.066	16.754	65.050	0.130	14.140	21	58.095	0.009	10.377	6.992	0.011	8.879
80	713.984	0.094	18.986	90.147	0.174	16.278	24	107.362	0.010	13.474	12.987	0.012	11.757
90	1366.604	0.103	21.693	139.644	0.207	18.382	27	174.950	0.012	17.119	20.117	0.015	14.789
100	1596.126	0.124	24.117	229.510	0.239	20.641	30	415.964	0.014	21.081	35.644	0.017	17.396

References

Alrifai, M., Risse, T., and Nejdl, W. 2012. "A Hybrid Approach for Efficient Web Service Composition with End-to-End QoS Constraints," *ACM Transactions on the Web* (6:2), pp. 1–31.

Ardagna, D., and Mirandola, R. 2010. "Per-flow Optimal Service Selection for Web Services Based Processes," *Journal of Systems and Software* (83:8), pp. 1512–1523.

Ardagna, D., and Pernici, B. 2007. "Adaptive Service Composition in Flexible Processes," *IEEE Transactions on Software Engineering* (33:6), pp. 369–384.

Baldauf, M., Schahram, D., and Florian, R. 2007. "A Survey on Context-Aware Systems," *International Journal of Ad Hoc and Ubiquitous Computing* (2:4), pp. 263–277.

Berbner, R., Spahn, M., Repp, N., Heckmann, O., and Steinmetz, R. 2006. "Heuristics for QoS-aware Web Service Composition," in *Proceedings of the 2006, IEEE International Conference on Web Services*, IEEE Computer Society (ed.), Chicago, Illinois, pp. 72–82.

Canfora, G., Di Penta, M., Esposito, R., and Villani, M. L. 2005. "An Approach for QoS-aware Service Composition Based on Genetic Algorithms," in *Proceedings of the 2005, Conference on Genetic and Evolutionary computation*, H.-G. Beyer (ed.), Washington, DC, pp. 1069–1075.

Canfora, G., Di Penta, M., Esposito, R., and Villani, M. L. 2008. "A Framework for QoS-aware Binding and Re-binding of Composite Web Services," *Journal of Systems and Software* (81:10), pp. 1754–1769.

Cuddy, S., Katchabaw, M., and Lutfiyya, H. 2005. "Context-aware Service Selection based on Dynamic and Static Service Attributes," in *Proceedings of the 2005, Interantional Conference on Wireless and Mobile Computing, Networking and Communications*, IEEE Computer Society (ed.), Montreal, Canda, pp. 13–20.

Cui, L., Kumara, S., and Lee, D. 2011. "Scenario Analysis of Web Service Composition Based on Multi-criteria Mathematical Goal Programming," *Service Science* (3:4), pp. 280–303.

Dai, Y., Yang, L., and Zhang, B. 2009. "QoS-driven Self-healing Web Service Composition Based on Performance Prediction," *Journal of Computer Science and Technology* (24:2), pp. 250–261.

Dannewitz, C., Pentikousis, K., Rembarz, R., Renault, É., Strandberg, O., and Ubillos, J. 2008. "Scenarios and Research Issues for a Network of Information," in *Proceedings of the 2008, International Mobile Multimedia Communications Conference*, European Alliance for Innovation (ed.), Oulu, Finnland, ICST, pp. 1–7.

Dey, A. K. 2001. "Understanding and Using Context," *Journal of Personal and Ubiquitous Computing* (5:1), pp. 4–7.

Dorigo, M., and Caro, G. D. 1999. "The Ant Colony Optimization Meta-Heuristic," in *New ideas in optimisation*, D. Corne, F. Glover and M. Dorigo (eds.), London: McGraw-Hill, pp. 11–32.

Ghallab, M., Nau, D., and Traverso, P. 2004. *Automated Planning*: *Theory and Practice*, San Francisco: Elsevier Inc.

Gu, T., Pung, H. K., and Zhang, D. Q. 2005. "A Service-oriented Middelware for Building Context-aware Services," *Journal of Network and Computer Applications* (28:1), pp. 1–18.

Heinrich, B., Bolsinger, M., and Bewernik, M. 2009. "Automated Planning of Process Models: The Constructions of Exclusive Choices," in *Proceedings of the 2009, International Conference on Information Systems*, H. Chen and S. Slaughter (eds.), Phoenix, Arizona, AIS.

Heinrich, B., Klier, M., Lewerenz, L., and Zimmermann, S. 2015. "Quality-of-Service-Aware Service Selection: A Novel Approach Considering Potential Service Failures and Nondeterministic Service Values," *Service Science* (7:1), pp. 48–69.

Heinrich, B., and Lewerenz, L. 2015. "Decision Support for the Usage of Mobile Information Services: A Context-Aware Service Selection Approach that Considers the Effects of Context Interdependencies," *Journal of Decision Systems* (24:4), pp. 406–432.

Heinrich, B., and Schön, D. 2015. "Automated Planning of Context-Aware Process Models," in *Proceedings of the 2015, European Conference on Information Systems*, AIS (ed.), Munster, Germany, AIS.

Henricksen, K., Indulska, J., and Rakotonirainy, A. 2002. "Modeling Context Information in Pervasive Computing Systems," in *Proceeding of the 2002, International Conference on Pervasive Computing*, F. Mattern and M. Naghshineh (eds.), Zurich, Switzerland, pp. 167–180.

Hinkelmann, K., Maise, M., and Thönssen, B. 2013. "Connecting Enterprise Architecture and Information Objects Using an Enterprise Ontology," in *Proceedings of the*

2013, International Conference on Enterprise Systems, A. Gerber and P. v. Deventer (eds.), Cape Town, South Africa, IEEE, pp. 1–11.

Hu, B., Wang, Z., and Dong, Q. 2012. "A Modeling and Reasoning Approach Using Description Logic for Context-aware Pervasive Computing," in *Emerging Reseach in Artificial Intelligence and Computational Intelligence*: *International Conference, AICI 2012, Chengdu, China, October 26-28, 2012. Proceedings*, J. Lei, F. L. Wang, H. Deng and D. Miao (eds.): Springer Berlin Heidelberg, pp. 155–165.

Hwang, C.-L., and Yoon, K. 1981. "Methods for Multiple Attribute Decision Making," in *Lecture Notes in Economics and Mathematical Systems*: *Multiple Attribute Decision Making - Methods and Applications A State-of-the Art Survey*, C.-L. Hwang and K. Yoon (eds.): Springer Berlin Heidelberg, pp. 58–191.

Hwang, S.-Y., Lim, E.-P., Lee, C.-H., and Chen, C.-H. 2008. "Dynamic Web Service Selection for Reliable Web Service Composition," *IEEE Transactions on Services Computing* (1:2), pp. 104–116.

Kirsch-Pinheiro, M., Yves, V., and Yolande, B. 2008. "Context-Aware Service Selection Using Graph Matching," in *Proceedings of the 2008, Workshop on Non Functional Properties and Service Level Agreements in Service Oriented Computing*, F. De Paoli, I. Toma, A. Maurino, M. Tilly and G. Dobson (eds.), Dublin, Ireland, CEUR-WS.org.

Li, J., Ma, D., Mei, X., Sun, H., and Zheng, Z. 2011. "Adaptive QoS-aware Service Process Reconfiguration," in *Proceedings of the 2011, IEEE International Conference on Services Computing*, IEEE Computer Society (ed.), Washington, DC, pp. 282–289.

Li, W., and Yan-xiang, H. 2012. "A Web Service Composition Algorithm Based on Global QoS Optimizing with MOCACO," in *Proceedings of the 2011, International Conference on Informatics, Cybernetics, and Computer Engineering*, J. Liangzhong (ed.), Melbourne, Australia, pp. 79–86.

Lin, D., Shi, C., and Ishida, T. 2012. "Dynamic Service Selection Based on Context-Aware QoS," in *Proceedings of the 2012, IEEE International Conference on Services Computing*, L. Moser, M. Parashar and P. Hung (eds.), Honolulu, Hawaii, IEEE, pp. 641–648.

Lin, K.-J., Zhang, J., Zhai, Y., and Xu, B. 2010. "The Design and Implementation of Service Process Reconfiguration with end-to-end QoS Constraints in SOA," *Service Oriented Computing and Applications* (4:3), pp. 157–168.

Madkour, M., Ghanami, D. E., Maach, A., and Hasbi, A. 2013. "Context-aware Service Adaptation: An Approach Based on Fuzzy Sets and Service Composition," *Journal of Information Science and Engineering* (29), pp. 1–16.

Meredith, J. R., Raturi, A., Amoako-Gyampah, K., and Kaplan, B. 1989. "Alternative research paradigms in operations," *Journal of Operations Management* (8:4), pp. 297–326.

Papazoglou, M. P. 2012. *Web services & SOA*: *Principles and technology*, Essex: Pearson Education.

Qiqing, F., Xiaoming, P., Qinghua, L., and Yahui, H. 2009. "A Global Optimzing Web Services Selection Algorithm based on MOACO for Dynamic Web Service Composition," in *Proceedings of the 2009, International Forum on Information Technology and Applications*, Z. Qihai (ed.), Chengdu, China, pp. 37–42.

Sensoy, M., Zhang, J., Yolum, P., and Cohen, R. 2009. "Poyraz: Context-aware Service Selection Under Deception," *Computational Intelligence* (25:4).

Serrano, N., Hernantes, J., and Gallardo, G. 2014. "Service-Oriented Architecture and Legacy Systems," *IEEE Software* (5:31), pp. 15–19.

Shen, Y., Wang, M., Tang, X., Luo, Y., and Guo, M. 2012. "Context-Aware HCI Service Selection," *Mobile Information Systems* (8), pp. 231–254.

Sun, S. X., and Zhao, J. 2012. "A Decomposition-based Approach for Service Composition with Global QoS Guarantees," *Information Sciences* (199), pp. 138–153.

Surianarayanan, C., Ganapathy, G., and Ramasamy, M. S. 2014. "An Approach for Selecting Best Available Services Through a New Method of Decomposing QoS Constraints," *Service Oriented Computing and Applications* (9:2), pp. 107–138.

Vanrompay, Y., Kirsch-Pinhero, M., and Yolande, B. 2009. "Context-Aware Service Selection with Uncertain Context Information," in *Proceeding of the 2009, International Workshop on Context-aware Adaptation Mechanisms for Pervasive and Ubiquitous Services*, R. Rouvoy and M. Wagner (eds.), Lisbon, Portugal, EASST, pp. 1–11.

Wan, C., Ullrich, C., Chen, L., Huang, R., Luo, J., and Shi, Z. 2008. "On Solving QoS-Aware Service Selection Problem with Service Composition," in *Proceedings of the 2008, IEEE International*

Conference on Grid and Cooperative Computing, IEEE Computer Society (ed.), Shenzhen, China, pp. 467–474.

Wang, J., Zeng, C., He, C., Hong, L., Zhou, L., Wong, R. K., and Tian, J. 2012. "Context-aware Role Mining for Mobile Service Recommendation," in *Proceedings of the 2012, Symposium On Applied Computing*, B. R. Bryant, H. M. Haddad, S. Ossowski and R. L. Wainwright (eds.), Trento, Italy, pp. 173–178.

Wang, W., Sun, Q., Zhao, X., and Yang, F. 2010. "An Improved Particle Swarm Optimization Algorithm for QoS-aware Web Service Selection in Service Oriented Communication," *International Journal of Computational Intelligence Systems* (3:6), pp. 18–30.

Wu, Q., and Zhu, Q. 2013. "Transactional and QoS-Aware Dynamic Service Composition Based on Ant Colony Optimization," *Future Generation Computer Systems* (29:5), pp. 1112–1119.

Yang, L., Dai, Y., and Zhang, B. 2009. "Performance Prediction Based EX-QoS Driven Approach for Adaptive Service Composition," *Journal of Information Science and Engineering* (25), pp. 345–362.

Yu, H. Q., and Reiff-Marganiec, S. 2009a. "A Backwards Composition Context Based Service Selection Approach for Service Composition," in *Proceedings of the 2009, IEEE International Conference on Services Computing*, IEEE Computer Society (ed.), Bangalore, India, IEEE, pp. 419–426.

Yu, H. Q., and Reiff-Marganiec, S. 2009b. "Automated Context-Aware Service Selection for Collaborative Systems," in *Proceedings of the 2009, International Conference on Advanced Information Systems Engineering*, P. van Eck, J. Gordijn and R. Wieringa (eds.), Amsterdam, The Netherlands, Springer Berlin Heidelberg, pp. 261–274.

Yu, T., Zhang, Y., and Lin, K.-J. 2007. "Efficient Algorithms for Web Services Selection with end-to-end QoS Constraints," *ACM Transaction on the Web* (1:1), pp. 1–26.

Yuan, Y., Zhang, X., Sun, W., Cao, Z., and Wang, H. 2013. "Optimal Web Service Composition Based on Context-awareness and Genetic Algorithm," in *Proceedings of the 2013, International Conference on Information Science and Cloud Computing Companion*, IEEE Computer Society (ed.), Guangzhou, China, IEEE, pp. 660–666.

Zeng, L., Benatallah, B., Ngu, A. H. H., Dumas, M., Kalagnanam, J., and Chang, H. 2004. "QoS-aware Middleware for Web Services Composition," *IEEE Transactions on Software Engineering* (30:5), pp. 311–327.

Zeng, L., Ngu, A. H. H., Benatallah, B., Podorozhny, R., and Lei, H. 2008. "Dynamic Composition and Optimization of Web Services," *Distributed and Parallel Databases* (24:1-3), pp. 45–72.

Zhang, M., Liu, C., Yu, J., Zhu, Z., and Zhang, B. 2013. "A Correlation Context-aware Approach for Composite Service Selection," *Concurrency and Computation: Practice and Experience* (25:13), pp. 1909–1927.

Zhou, J., Gilman, E., Palola, J., Riekki, J., Ylianttila, M., and Sun, J. 2011. "Context-aware Pervasive Service Composition and its Implementation," *Personal and Ubiquitous Computing* (15:3), pp. 291–303.