Web Service Substitution Based on Preferences Over Non-functional Attributes*

Ganesh Ram Santhanam Samik Basu Vasant Honavar
Department of Computer Science, Iowa State University, Ames, IA 50011, USA
{gsanthan, sbasu, honavar}@cs.iastate.edu

Abstract

In many applications involving composite Web services, one or more component services may become unavailable. This presents us with the problem of identifying other components that can take their place, while maintaining the overall functionality of the composite service. Given a choice of candidate substitutions that offer the desired functionality, it is often necessary to select the most preferred substitution based on non-functional attributes of the service, e.g., security, reliability, etc. We propose an approach to this problem using preference networks *for representing and reasoning about preferences over non-functional properties. We present algorithms for solving several variants of this problem: a) when the choice of the preferred substitution is independent of the other constituents of the composite service; b) when the choice of the preferred substitution depends on the other constituents of the composite service; and c) when multiple constituents of a composite service need to be replaced simultaneously. The proposed solutions to the service substitution problem based on preferences over non-functional properties are independent of the specific formalism used to represent functional requirements of a composite service as well as the specific algorithm used to assemble the composite service.*

1. Introduction

Service-oriented computing [3, 14, 11] offers a powerful approach to the assembly of complex distributed systems from independently developed software components in many applications, e.g., e-Science, e-Business and e-Government. Consequently, there is a growing body of work on specification, discovery, selection, and composition of services.

After a composite service assembled from a repository of component services has been deployed, one or more constituents of the composite service may become unavailable. Hence there arises a need to replace such components with other components from the repository while maintaining the overall functionality of the composite service. Among the candidate substitutions that offer the desired functionality, the user might prefer some substitutions over others based on non-functional attributes of the service, e.g., security, reliability, etc.

Service substitution based on the functional properties of components has been addressed by many authors [2, 4, 12, 13, 15]. This paper is aimed at addressing the service substitution problem taking into account the user preferences over non-functional properties. We associate with each non-functional property, a corresponding domain and assume that the non-functional properties as well as their respective domains are specified by some agreed upon standards. Service substitution in such a setting requires the user to be able to express preferences over non-functional properties of services. For example, a user might prefer a more secure service to a less secure one; or one with a lower cost over one with a higher cost. Furthermore, some attributes may be more important than others, in which case, it is useful to assign *relative importance* to different non-functional attributes (e.g., security being more important than performance).

Such preferences may be *qualitative* or *quantitative*. Qualitative preferences are asserted based on *relative goodness* of two alternatives, whereas quantitative preferences force the user to quantify *by how much* he/she prefers one alternative to another, for example in the form of *utility functions* [9]. While quantitative preferences over multiple attributes can be difficult to elicit from users, qualitative preferences are often easier to elicit [10, 18].

Service substitution considering user preferences over non-functional attributes is complicated by the fact that the value of a particular attribute of the composite service is a function of all its constituent services. For instance, the reliability of a composite service is only as high as the reliability of its *least* reliable constituent. Further, there can be interactions among the user preferences with respect to different non-functional attributes. For example, due to prohibitive cost, the user may prefer higher security when there is low reliability and lower security when reliability is high.

*This work is supported in part by NSF grants CNS0709217, CCF0702758 and IIS0711356.

Against this background, this paper addresses the problem of service substitution based on user specified qualitative preferences over non-functional attributes. The contributions of the paper are as follows:

1. We introduce an approach to service substitution based on user preferences over non-functional properties of services. Our approach utilizes *preference networks* [6] for representing and reasoning about preferences over non-functional properties in this setting.
2. We consider and solve two variants of the service substitution problem, namely, *context-insensitive* and *context-sensitive* substitution. The former assumes that the preferred substitution can be identified independent of the *context*, i.e., the other constituents in the composition, whereas the latter takes into account the context of the substitution.
3. We consider and solve the service substitution problem in a more general setting, wherein multiple constituents of a composition need to be replaced.

The proposed solutions to the service substitution problem based on preferences over non-functional properties are independent of the specific formalism used to represent functional requirements of a composite service as well as the specific algorithm used to assemble the composition.

Organization. Section 2 discusses the preference formalism called TCP-net, a model for representing and reasoning with qualitative preferences and trade-offs. Section 3 describes our solutions for identifying the preferred substitutions for both context-insensitive and context-sensitive substitution problems using the TCP-net preference model. Section 4 describes our algorithm for substitution of multiple services in a composition. Section 5 concludes with a summary, discussion of related work and an outline of future directions for research.

2. Preferences and Trade-offs: TCP-nets

The information about preferences and trade-offs over preference variables (non-functional attributes) can be compactly represented using a graphical model called TCP-net. A TCP-net [5, 6] is a directed graph such that the nodes of the TCP-net represent preference attributes $\mathcal{V}$, and there are three types of edges. The first type of edge is a directed edge (single headed arrow) from X ($\in \mathcal{V}$) to Y ($\in \mathcal{V}$). Such an edge asserts the preferential dependence of an attribute Y on the assignment of its parents $Pa(Y)$ (e.g. $X \in Pa(Y)$). Each node (preference attribute) Y, that has a non empty set of parents $Pa(Y)$ influencing its preferences, is annotated with the conditional preference relation called *conditional preference table* $CPT(Y)$. More formally, we assume for each assignment of $Pa(Y)$, $CPT(Y)$ specifies a total order over $D(Y)$, the domain of Y. The second type of directed edge (double headed arrow) captures the relative

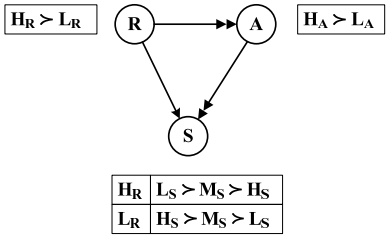

Figure 1: TCP-net: Representing Preferences and Importance

importance among a pair of attributes, i.e., if there is such an edge from X to Y then X is relatively more important than Y. The third type of edge is undirected and captures conditional relative importance between X and Y given Z.

Example 1 *Suppose that a user specifies preferences over three non-functional attributes: reliability (R), security (S) and availability (A). The domains of the attributes are $\{L_R, H_R\}$, $\{L_S, M_S, H_S\}$ and $\{L_A, H_A\}$ respectively, where L_i represents* low *"level" of the attribute i, M_i represents* medium *and H_i represents* high*. The user specifies that reliability is more important than availability, and availability is more important than security. I.e., the user prefers high valuations of reliability and availability. Finally, the user states that his/her preference with respect to the security is* not *independent of the reliability. At lower levels of reliability, the user prefers higher security, but the user tends to prefer lower security when the reliability is high (say, due to prohibitive costs of having higher levels for both attributes). The TCP-net in Figure. 1 represents the above user preferences. It shows that S is preferentially dependent on R (i.e. $Pa(S) = \{R\}$). Double-headed arrow captures the fact that R is relatively more important than A which, in turn, is relatively more important than S. Finally, the conditional preference tables annotate each node presenting the total order of the domain of each attribute (w.r.t. the parents of the node, if any).*

2.1 Preferential Semantics

For completeness of discussion, we describe the formal semantics of TCP-net representation of preferences following [6]. We assume a set of preference variables $\mathcal{V} = \{X_1, \ldots X_n\}$ with finite domains $D(X_1)$, $\ldots D(X_n)$. An *outcome* o is a complete assignment of all variables X_i in $\mathcal{V}$. The set of outcomes is $\mathcal{O} \subseteq D(X_1) \times D(X_2) \times \ldots \times D(X_n)$. A *preference ranking* is a total preorder over the set of outcomes $\mathcal{O}$. We denote the fact that outcome $o_1 \in \mathcal{O}$ is *at least as preferred (strictly preferred)* to outcome $o_2 \in \mathcal{O}$ by $o_1 \succeq o_2 (o_1 \succ o_2)$. We denote the fact that the user is *indifferent* between outcomes o_1 and o_2 by $o_1 \cong o_2$ if neither $o_1 \succeq o_2$ nor $o_2 \succeq o_1$[1].

[1]TCP-nets also allow the expression of *conditional* preferences – preferences over variable valuations conditioned on other variables' valuations.

Preferential Independence. In order to understand the need for preferential independence, we note that the set of possible outcomes is exponential in the number of preference variables n (where $n = |\mathcal{V}|$). Further, the set of possible total preorders is doubly exponential in n. A set of variables $X \subseteq \mathcal{V}$ is *preferential independent* of $Y = \mathcal{V} - X$ if for all possible values of Y, the *preference order* among various assignments to X is the same. Formally, a set of variables X is *preferentially independent*[2] of the set of variables $Y = \mathcal{V} - X$ iff for all $x_1, x_2 \in D(X)$; $y_1, y_2 \in D(Y)$, we have: $x_1 y_1 \succeq x_2 y_1$ *iff* $x_1 y_2 \succeq x_2 y_2$. We say that x_1 is preferred to x_2 *ceteris paribus*[3] (all else being equal). In Figure 1, reliability R is preferentially independent of security S while S is preferentially dependent on R.

Relative Importance. In Figure. 1, we observe that the variables availability and reliability are preferentially independent. However, in order to assert if an outcome with high availability and low reliability is preferred to one with low availability and high reliability, preferential independence information alone is not sufficient. Additional information regarding the relative importance of one attribute over the other will be necessary. In our example, reliability is relatively more important than availability (double headed arrow). As a result, given a choice, the user would settle for lower availability instead of compromising on reliability. This additional relative importance information helps us to infer that an outcome with high reliability and low availability is preferred to one with low reliability and high availability. Formally, let X and Y be a pair of preferentially independent variables given $\mathcal{V} - \{X, Y\}$. We say that X is *relatively more important* than Y, denoted by $X \triangleright Y$, if

$$\forall w.\ w \in D(W),\ where\ W = \mathcal{V} - \{X, Y\}$$
$$\forall x_1, x_2 \in D(X), \forall y_a, y_b \in D(Y) :$$
$$x_1 \succ x_2 \Rightarrow x_1 y_a w \succ x_2 y_b w.$$

The above definition states that the preference $x_1 y_a w \succ x_2 y_b w$ holds even if $y_b \succ y_a$, since any change to the worse in Y is preferred to any change to the worse in X.

In Figure 1, based on the these definitions, R is preferentially independent of A and S; S is preferentially dependent on R and is annotated with a CPT. R is relatively more important than A and A is relatively more important than S.

Remark 1 *We limit our scope of discussion to a class of TCP-nets called* conditionally acyclic *TCP-nets, as only this particular class of TCP-nets have been proved to be satisfiable with a preference relation [6]. We also assume that the preferences over variable domains are total orders for the purpose of this paper, although TCP-nets in general allow specification of partial orders as well. We note that there is another variant of the TCP-net, known as UCP-nets [6] that captures quantitative preferences and relative importance information using utility functions. However, since we are not dealing with quantitative preferences, we stick to the basic qualitative TCP-nets.*

Non-dominated Set of Outcomes. Given a *conditionally acyclic* TCP-net, there exists a total order (that can be obtained using a topological sort) of the set of outcomes $\mathcal{O}$ that is *consistent with* the given TCP-net [6]. However, several orderings of $\mathcal{O}$ can be consistent with a given conditionally acyclic TCP-net (corresponding to distinct topological sorting of the variables in the preference network). In a total preorder, there could be an outcome o such that $\nexists o' : o' \succ o$ with respect to the TCP-net, but one cannot define o as the unique most preferred outcome.

Consider the example in Figure 1. If the user did not provide the information that R is relatively more important than A, then we will not be able to assert the preference between two valuations of attributes: (H_R, H_S, L_A) and (L_R, H_S, H_A). In other words, the user is indifferent with respect to the above outcomes and we say that the outcomes form a *non-dominated* set, one where any element in the set is *not preferred* over any other element in the set. To handle such situations, it is necessary that the reasoning about preferences considers non-dominated outcomes instead of the unique most preferred one.

3. Web Service Substitution

We now proceed to describe the problem of Web service substitution, and how to use TCP-nets to compute the preferred substitutions from a set of functionally feasible alternatives. For this purpose, we will use *dominance queries* [6] of the form $o \overset{?}{\succ} o'$ with respect to TCP-net (in other words whether o is preferred to or dominates o').

Definition 1 (Web Service Composition [17]) *A Web service composition $C = W_1 \oplus W_2 \ldots \oplus W_k$ such that $\forall 1 \leq l \leq k, W_l \in R$ is an assembly of component services from a repository of available services $R = \{W_1, W_2 \ldots W_n\}$ such that C is functionally equivalent* [4] *to a target or goal service G, denoted by $C \equiv G$. In the above, $\oplus$ is the* composition *operator for composing two services.*

The problem of Web service substitution refers to identifying a component service from the repository of services that can suitably replace a particular component in an existing composite service. The identified component service must achieve the desired functionality in the context of the composite service as a whole. Formally, Web service substitution is defined as follows.

[2] Conditional preferential independence can be defined similarly.

[3] Ceteris paribus is a Latin phrase that means "all others being equal".

[4] Functional equivalence can be defined in many ways including bisimulation of labeled transition systems [16, 15]

Definition 2 (Web Service Substitution) *Given an existing composite service $C = W_1 \oplus W_2 \ldots \oplus W_{i-1} \oplus W_i \oplus W_{i+1} \ldots \oplus W_k$ that achieves the functionality of the target or goal service G such that $C \equiv G$, Web service substitution amounts to identifying a replacement, W_s, of a component W_i in the composite service, such that $C' = W_1 \oplus W_2 \ldots \oplus W_{i-1} \oplus W_s \oplus W_{i+1} \ldots \oplus W_k$ and $C' \equiv C$.*

We will also use the notation $C \ominus W$ to denote the partial composition obtained by removing the service W from the composite service C i.e., $C \ominus W_i = W_1 \oplus W_2 \ldots \oplus W_{i-1} \oplus W_{i+1} \ldots \oplus W_k$. The above definition, however, does not take into consideration the preferences over non-functional attributes of the composition and the components.

Given the user-preferences and trade-offs over non-functional attributes in the form of a TCP-net, we can define the *preference valuation* [17] of a service as follows.

Definition 3 (Preference Valuation) *Preference valuation is a function $F : \mathcal{W} \times \mathcal{X} \rightarrow \bigcup(D(X_i))$ $\mathcal{W} = \{W_1, W_2 \ldots W_k\}$, $\mathcal{X} = \bigcup\{X_i\}$. We denote the valuation of an attribute X_i in a Web service W as $F(W)(X_i) = v_i$ where $v_i \in D(X_i)$. We define the valuation of an attribute X_i in a* composition *of two services $W_i \oplus W_j$ as $F(W_i \oplus W_j)(X_i) = F(W_i)(X_i) \odot F(W_j)(X_i)$, where*

$$F(W_i)(X_p) \odot F(W_j)(X_p) = \begin{cases} F(W_j)(X_p) & \text{if } F(W_i)(X_p) \succ F(W_j)(X_p) \\ F(W_i)(X_p) & \text{otherwise} \end{cases}$$

*The **preference valuation** of a composition $P = W_1 \oplus W_2 \oplus \ldots W_l$ with respect to attribute X_p is defined (inductively) as $F(P)(X_p) = F(W_1)(X_p) \odot F(W_2)(X_p) \ldots \odot F(W_l)(X_p)$. We also denote the **complete preference valuation** over all attributes $\mathcal{X}$ of a composition P as the tuple $V_P = \langle F(P)(X_i), F(P)(X_2) \ldots F(P)(X_k) \rangle$.*

The function F defines how the valuations of P's components are aggregated. Our definition of F computes the the least preferred valuation of that attribute among the participating component services in the composition. For example, in the case of reliability, the valuation of a composition can be defined as the valuation of its *least* reliable component. We note that other attributes such as cost may require other ways of aggregating the attribute valuations of the components in a composition, which can be handled by having appropriate definitions of F and $\odot$.

Definition 4 (Sole Dependence) *Given a composition C and its component W, we say that C is* solely dependent *on W with respect to an attribute X_i iff $V_{C \ominus W}(X_i) \succ V_C(X_i)$.*

In other words, improving the valuation $V_W(X_i)$ improves C's valuation of X_i as well, i.e., all other components in C have a strictly better valuation for X_i than W.

	Services	Reliability	Security	Availability
Composite	C	L_R	L_S	H_A
To-replace	W	L_R	L_S	H_A
Substitutes	W_1	L_R	L_S	L_A
	W_2	H_R	H_S	L_A
	W_3	L_R	M_S	H_A
	W_4	L_R	H_S	H_A

Table 1: Preference Valuations

3.1. Computing Preferred Substitutions

Now we address the problem of finding the preferred substitutions from a set of functionally feasible substitutions using a TCP-net model of preferences over non-functional attributes. Given a composite service C and a component W to be replaced, we assume that there exists a mechanism[5] that generates the set of functionally feasible substitutions, $\mathcal{W}$, from the repository of available services. Our goal is to compute the set $\mathcal{W}' \subseteq \mathcal{W}$ of preferentially non-dominated substitutions with respect to the given TCP-net.

We further distinguish between two variants of the substitution problem, namely, *context-insensitive* and *context-sensitive* substitution. The first approach assumes that the preferred substitution can be obtained independent of the *context*, i.e., the non-functional properties of the other components in the composition, while in the second approach, the context of the substitution is taken into account.

Example 2 *Consider a composite service C that and a component W in C that needs to be substituted. Suppose that there is a set of* functionally feasible substitutions *$\mathcal{W} = \{W_1, W_2, W_3, W_4\}$ such that each component $W_i \in \mathcal{W}$ can substitute W in C satisfying all the functional requirements of the substitution. The non-functional attributes of interest to the user are reliability, availability and security, and the user-preferences over these attributes are represented using the TCP-Net in Figure 1. The valuations of the non-functional attributes for C, W and the substitutes are presented in Table 1. The objective is to identify the preferred substitution(s) for W in the composition C.*

3.1.1 Context-Insensitive Substitution

This approach simply computes the preferentially non-dominated substitutions for the component to be replaced, from the set of functionally feasible substitutions, without considering how the non-functional properties of the other components in the composition may affect the valuation of the overall composition.

In Example 2, W_2 is the most preferred substitution in $\mathcal{W}$ as $V_{W_2} \succ V_{W_4} \succ V_{W_3} \succ V_{W_1}$ (see Figure 1 and Table 1).

[5] We refer the reader to [15, 12, 13, 2, 4] for more details on functional aspects of service substitution

If C is *solely dependent* on W with respect to reliability and security, then the choice of W_2 is the most preferred one. As the existing composite service C has the valuation $V_C = (L_R, L_S, H_A)$, following Definition 3, the new composition with W_2 as the substitute will have a valuation $V_C{}' = (H_R, H_S, L_A) \succ V_C$.

However, consider the scenario when with respect to the attributes reliability and security, C is *not solely dependent* on W, e.g., there is some other service in C in addition to W that has low reliability and low security. In that case, W_2 as the preferred substitution would be a bad choice, as the overall valuation of C goes down: $V_C{}' = (L_R, L_S, L_A)$, i.e., $V_C \succ V_C{}'$. This is because the computed substitutions do not take into account the *context* of the composite service as a whole. A better substitution in terms of preference can be obtained, if we take into account some context information, i.e., the preference valuation of the composite service C in the absence of W, $V_{C \ominus W}$. Thus, the context-insensitive approach works well only when all the non-functional attribute valuations of the composition are solely dependent on the component being replaced.

3.1.2 Context-Sensitive Substitution

In this approach, when identifying a replacement, we take into account how the valuation of the identified substitution will affect the overall valuation of the composition. As a result, here we consider the valuation of the composition in the absence of the component to be replaced in order to identify substitutions, i.e., $V_{C \ominus W}$, in contrast to the context-insensitive approach where we instead considered V_C. The following algorithm describes this approach.

1. Compute $V_{W_j}{}' = V_{(C \ominus W) \oplus W_j}, \forall W_j \in \mathcal{W}$
2. Compute the preferentially non-dominated set of valuations $\varphi = \{V_{W_j}{}' \mid \nexists W_k{}' : V_{W_k}{}' \succ V_{W_j}{}'\}$ w.r.t. TCP-net
3. Return the set of substitutions corresponding to each valuation in φ i.e., $\mathcal{W}' = \{W_j \mid V_{W_j} \in \varphi\}$

To see the subtle distinction between the context-insensitive and sensitive approaches, suppose that in Example 2, C is solely dependent on W with respect to security; and not with respect to reliability and availability. Let the valuation of $V_{C \ominus W}$ yield (L_R, H_S, H_A) – assuming all other components in C have high security level. The above algorithm would return the solution as W_4 because $V_{(C \ominus W) \oplus W_4} = (L_R, H_S, H_A)$ clearly dominates $V_{(C \ominus W) \oplus W_3} = (L_R, M_S, H_A)$, as well as $V_{(C \ominus W) \oplus W_1} = (L_R, L_S, L_A)$ and $V_{(C \ominus W) \oplus W_2} = (L_R, H_S, L_A)$. Thus, this approach yields the best solution $\mathcal{W}' = \{W_4\}$ taking into account the context information.

4. Multiple Component Substitution

The solutions we have seen so far are aimed at finding preferred substitutions for one component at a time. We now address the problem of finding substitutes for more than one component at a time.

Consider a composite service C that needs to replace a *set* of n of its components: $W = \{W_1, W_2, \ldots W_n\}$. We again assume that the user-preferences over the non-functional attributes are modeled using a TCP-net, and that there exists a mechanism to find out the set of functionally feasible substitutions for any given component in the composite. Suppose that each W_i has a set of functionally feasible substitutions: $R_{W_i} = \{W_{i_1}, W_{i_2}, \ldots W_{i_k}\}$. The problem is to find one substitution W_{i_j} from R_{W_i} for each component W_i to be replaced. There are many feasible sets of substitutions functionally possible, given by the space $\mathcal{P} = R_{W_1} \times R_{W_2} \ldots \times R_{W_n}$. We denote the composition obtained by making the set of substitutions $S = \{W_1^s, W_2^s, \ldots W_n^s\}$ (where W_i^s denotes a substitute service in the set R_{W_i}) as $(C \ominus R) \oplus S$. We are interested in finding a set of substitutions $S \in \mathcal{P}$ that maximizes the preference of the resulting composite service, i.e., $\nexists S' \in \mathcal{P} : V_{(C \ominus R) \oplus S'} \succ V_{(C \ominus R) \oplus S}$.

One way to search for an optimal solution is by brute force: exploring the entire space of sets of substitutions $\mathcal{P}$, and finding the set of non-dominating set of substitutions from that set. However, given that the number of components to be replaced is n and considering that each component has k functionally feasible substitutes, it is computationally expensive (number of possible sets of substitutions is exponential: $O(k^n)$) to explore the entire space.

A more efficient but naive approach for finding multiple component substitutions according to the user preferences would be to execute the one-component substitution algorithm multiple times, once for each component to be substituted. However, there is no guarantee that this approach would give the best possible set of substitutions, as illustrated by the following example.

Example 3 *Consider a composition C that requires three services to be replaced, $W = \{W_1, W_2, W_3\}$. Let there be functionally feasible substitutions, $R_{W_1} = \{W_{11}, W_{12}, W_{13}\}, R_{W_2} = \{W_{21}, W_{22}, W_{23}\}, R_{W_3} = \{W_{31}, W_{32}, W_{33}\}$ respectively. Table 2 shows their valuations over two non-functional attributes X and Y with domains $\{x_0, x_1, x_2, x_3\}$ and $\{y_0, y_1, y_2, y_3\}$ respectively. The preferences over the variables are given in Figure 2.*

Let $V_{C \ominus W} = \langle x_2, y_3 \rangle$; then the naive substitution approach will execute the single-component substitution for W_1, W_2 and W_3 independently, yielding replacements W_{11}, W_{21} and W_{31} respectively according to the algorithm presented in Section 3.1.2. The resulting preference value $V_{C \ominus W \oplus \{W_{11}, W_{21}, W_{31}\}} = \langle x_0, y_1 \rangle$. However, note that there exists another solution, namely replacements W_{12}, W_{22} and W_{31} for W_1, W_2 and W_3 respectively, which yields a better solution with valuation $V_{C \ominus W \oplus \{W_{12}, W_{22}, W_{31}\}} = \langle x_2, y_1 \rangle$.

$V_{W_{11}} = \langle x_1, y_3 \rangle$	$V_{W_{21}} = \langle x_0, y_2 \rangle$	$V_{W_{31}} = \langle x_3, y_1 \rangle$
$V_{W_{12}} = \langle x_2, y_2 \rangle$	$V_{W_{22}} = \langle x_2, y_1 \rangle$	$V_{W_{32}} = \langle x_2, y_1 \rangle$
$V_{W_{13}} = \langle x_1, y_1 \rangle$	$V_{W_{23}} = \langle x_0, y_1 \rangle$	$V_{W_{33}} = \langle x_1, y_1 \rangle$

Table 2: Valuations of replacements

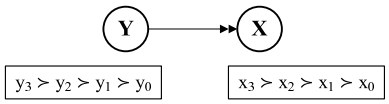

Figure 2: Preferences: Multiple Component Substitution

The reason for the sub-optimal solution obtained by the naive approach in the above example is that it does not consider the effect of the choice of replacement for one component on the choice of replacement for others.

The solution obtained by the naive approach can be improved by having a search procedure that chooses the optimal replacement for a component W_i, *contingent* on the previous replacement choices already made. However, note that even in such an approach, the *order* in which we choose replacements for components plays an important role in determining the solutions. For example, if we choose the replacement for W_1 first, followed by a replacement for W_2 (given the choice for W_1), followed by a choice for W_3 (given the choices for W_1 and W_2), we obtain the sub-optimal solution W_{11}, W_{21} and W_{31} with valuation $\langle x_0, y_1 \rangle$. Instead, if we choose the replacement for W_3 first, followed by W_2 and W_1, then the resulting solution is optimal: the choice for W_3 is W_{31}; the choice for W_2 (*given* the choice for W_3) is W_{22}, and the choice for W_1 (*given* the choices for W_3 and W_2) is W_{12}. The resulting valuation of the substituted composite service is $\langle x_2, y_1 \rangle$ and it is optimal. Thus, the order in which the replacements for the components are chosen impacts the optimality.

In the absence of any information regarding the order in which components have to be replaced, the *only* way to guarantee an optimal solution is to explore all possible orders. In effect, this problem generalizes the known NP-hard traveling salesman problem [8] that involves finding the optimal ordering of points in a plane such that the overall real-valued cost is minimized. The difference in our case is that we deal with qualitative valuations instead of real-valued costs.

We propose an approach to finding the optimal solution by exploiting that fact that the optimal solution corresponds to some *preferred order* in which the services are considered to be replaced. In the above example, such orders are (W_3, W_2, W_1) and (W_3, W_1, W_2). We present an algorithm that organizes the possible orderings in the form of a lattice and obtains the preferred order as the shortest path between two nodes in the lattice.

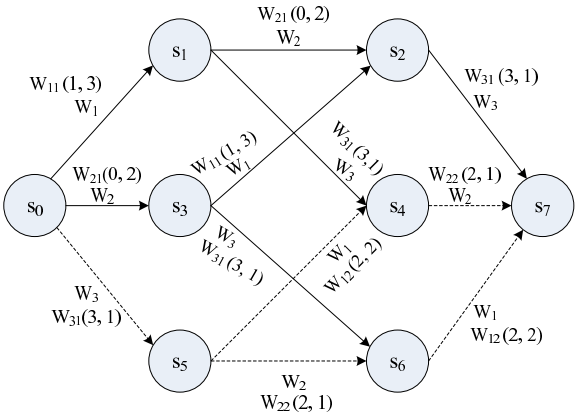

Figure 3: Multiple Component Substitution

4.1 Finding Preferred Order

We construct a lattice with the bottom of the lattice as the partial composition $C \ominus W$, and the top of the lattice as the fully substituted or *repaired* composition, namely $C \ominus W \oplus S$. Each node in level l (i.e., nodes that are l steps away from the bottom) of the lattice is the partial composition $C \ominus W$ composed with l different substitutes. There are a total of $n + 1$ levels in the lattice and at each level l of the lattice, there are $\binom{n}{l}$ nodes. In particular, the level 0 of the lattice is the bottom and level $n+1$ is the top. The lattice for Example 3 is illustrated in Figure 3, where s_0 corresponds to $C \ominus W$ and s_7 corresponds to $C \ominus W \oplus S$, where S is the optimal substitution. We note that each path in the lattice specifies one order of selecting substitutes in W, and we denote the valuation of the fully substituted composition obtained through a path p in the lattice as V_p.

We assign the cost of each node in the lattice as follows. The cost of the bottom of the lattice (s_0 in Figure 3) is $V_{C \ominus W}$. The cost of a path of length l from s_0 to any other node is the given by $V_{C \ominus W \oplus \{\widehat{W_1}, \ldots, \widehat{W_l}\}}$, where $\widehat{W_i}$ is locally-preferred substitute for

$$W_i \in \{W_1, W_2, \ldots, W_n\} - \bigcup_{j<i} \{W_j \mid \widehat{W_j} \text{ substitutes } W_j\}$$

For example, in Figure 3, there are two paths from s_0 to s_4. The cost of substitution along the path s_0, s_1, s_4 is $V_{C \ominus W \oplus \{W_{11}, W_{31}\}} = \langle x_1, y_1 \rangle$; in this path substitution of W_1 is selected before the substitution for W_3. While the cost of substitution along the path s_0, s_5, s_4 is computed from the selection of substitution for W_3 followed by that for W_1. Cost of any node other than the bottom is the most preferred cost among the paths from the bottom of the lattice to itself. For example, cost of node s_4 is $\langle x_2, y_1 \rangle$ due to the cost of the path s_0, s_5, s_4. Finally, the most preferred substitution is given by the path from the bottom to the top of the lattice which corresponds to the cost of the top. Note that, this path corresponds to the preferred-order.

Algorithm 1 PreferredSubstitutions($\succ, G(N, E)$)

```
 1: for all n ∈ N do
 2:    V_n ← ▽
 3:    ρ(n) ← undef
 4: end for
 5: V_{⊥∈N} ← V_{C⊖W}
 6: while |N| > 0 do
 7:    N_1 ← {n ∈ N : ∄m ∈ N : V_m ≻ V_n}
 8:    N ← N \ N_1
 9:    for all n ∈ N_1 do
10:       for all m ∈ N such that (n, m) ∈ E do
11:          if V_{n⊕m} ≻ V_m then
12:             V_m ← V_{n⊕m}
13:             ρ(m) ← n
14:          else if V_m ∼ V_{n⊕m} then
15:             Create a node m' with same edges as m
16:             V'_m ← V_{n⊕m}
17:             N ← N ∪ {m'}
18:          end if
19:       end for
20:    end for
21: end while
```

Algorithm 1, inspired by Dijkstra's shortest path algorithm [8], computes the cost of the top of the lattice. Whereas Dijkstra's shortest path algorithm works for quantitative costs, or cases when the valuations of the paths are totally ordered, Algorithm 1 works for partial orders as well. This is needed because a node can be associated with multiple costs as the cost of the paths from the bottom to that node may be incomparable. Furthermore, unlike Dijkstra's algorithm which works on real value comparison operator, $>$, Algorithm 1 uses dominance relation $\succ$ between non-functional attribute valuations.

Lines 1–4 initializes the cost of a node V_n and its parent $\rho(n)$ to $\triangledown$ and `undef` respectively. The symbol $\triangledown$ denotes the worst valuations of the non-functional attributes in our setting (e.g., $\langle L_R, L_S, L_A \rangle$). At Line 5, the bottom of the lattice is assigned the cost of $V_{C \ominus W}$, i.e., the value of the non-functional attributes of the composition C without the services in W. Line 7 obtains the set of nodes whose associated cost is not dominated by that of any other node. Initially, this set will be singleton containing only the bottom element (unless $V_{C \ominus W} = \triangledown$, in which case any substitution for the services in W is a preferred one). Line 10–13 identifies the one-step neighbors of the current node and updates their cost appropriately if the costs are comparable (Line 11). If there are two paths to m with incomparable costs (specifically with non-dominating costs – Line 14), then the node is split into two (Lines 15–17). In general, if there are p paths leading to a node in the lattice, and p' paths form the non-dominating set with respect to their valuations, then the node is split into p' nodes (replicating all the edges of the original node in the split nodes), one for each of the paths with non-dominated valuations. However, such splitting, which will increase the computational complexity, can be effectively avoided by soliciting information from the user to break the tie due to non-dominance whenever the condition at Line 14 is satisfied. In that case, our algorithm will be able to assign a unique cost to each node, thereby avoiding node splitting.

The algorithm terminates when the minimum cost of all nodes are assigned; At this point, the valuations at the top element in the lattice correspond to the most preferred non-dominated substitutions.

4.1.1 Complexity

Let there be n components to be replaced, and k feasible substitutes for each of them. By our construction, for each edge of the lattice, we make one substitution, i.e., we choose the best substitute from k candidates. Assuming that we obtain unique cost for each node (i.e., costs of all paths leading to a node are comparable), the number of times we make such a selection is equal to the total number of edges in the lattice, which is $n.2^{n-1}$. Hence, we consider $k.n.2^{n-1}$ sets of substitutions in all. It can be shown that $\forall k > 2, \forall n > 4 : k^n > k.n.2^{n-1}$, i.e., our approach will be more efficient than brute force whenever $k > 2$ and $n > 4$.

Remark 2 *If the cost of a node is not unique, as would be the case if there are at least two paths to that node with non dominating costs, then an alternative to splitting the node is to solicit information from the user to break the tie between these costs. This leads to unique cost at each node. In the event, the user fails to provide such information, the algorithm can proceed by splitting the nodes. However, such splitting will lead to increase in computational cost of our method. Let m nodes be split at each level of our lattice representation. Then the complexity for computing the globally-preferred substitutions is*[6]

$$C(n, m) = k.n.2^{n-1} + k.m \sum_{i=2}^{n-1} (i! - 1)(n - i) \qquad (1)$$

Using this, we can design a method that switches from our algorithm to brute-force method when m splits are made and any further splitting will make k^n less than $C(n, m)$.

5. Summary and Discussion

Web service substitution approaches [15, 12, 13, 2, 4] have been developed in the past that identify functionally feasible replacements from a given repository. Regarding non-functional properties, existing techniques have focused on analyzing non-functional properties as hard-constraints

[6] The derivation of Equation 1 is not presented due to space constraint.

(e.g., reliability after substitution *must* be 70%) or modeled them as a minimization/maximization problem (e.g., reliability after substitution must be maximized) [1, 7, 19]. These techniques are either based on constraint satisfiability or require the user to precisely quantify their preferences over non-functional attributes (or both), or they do not address the problem of substitution in detail. To the best of our knowledge, there has been little work on identifying *preferred substitutions* in accordance with user-specified *qualitative* intra-variable and relative importance preferences over non-functional attributes.

We have introduced two different variants of the substitution problem given preferences over non-functional attributes: context-insensitive and context-sensitive. Our treatment of these two variants parallels the work of Pathak et al. [15] on service substitution based on functional attributes. However, it departs from the work of Pathak et al. in an important respect: it offers a solution to the service substitution problem taking into consideration the preferences over non-functional attributes. We have also addressed the problem of substitution of multiple services in a composition. We showed that, in the context-sensitive setting, the order in which services are substituted influences the quality of the solution obtained (relative to user-specified preferences over non-functional attributes). We refer to the substitution order(s) corresponding to the most preferred solution as the preferred order(s). In the absence of any additional information regarding the preferred order, solving the multiple service substitution problem in the context-sensitive setting appears to be NP-hard. We show how to represent all possible substitution orders compactly by arranging the services to be replaced within a lattice structure. Finally, we reduce the problem of obtaining the preferred order(s) to identifying the shortest path(s) between the bottom and top elements of the lattice. We identify the conditions under which our method is computationally more efficient than a brute force search. Our approach to service substitution based on preferences over non-functional properties is independent of the specific formalism used to represent functional requirements of a composite service as well as the specific algorithm used to assemble the composite service.

Work in progress includes experimental studies of the performance and scalability of our algorithms. We next plan to implement our approach in an existing system like MoSCoE that does functional substitution. In future, we would like to extend our framework for service substitution to preference models that allow specification of partial orders over variable domains. We also plan to extend our framework to allow hard constraints over non-functional attribute values in addition to qualitative preferences, so that constrained preferential optimization supported by TCP-nets [6] can be leveraged to obtain preferred substitutions.

References

[1] D. Ardagna and B. Pernici. Adaptive service composition in flexible processes. *IEEE Trans. Softw. Eng.*, 33(6):369–384, 2007.

[2] B. Benatallah, F. Casati, and F. Toumani. Representing, analysing and managing web service protocols. *Data Knowl. Eng.*, 58(3):327–357, 2006.

[3] M. Bichler and K. J. Lin. Service-oriented computing. *Computer*, 39(3):99–101, 2006.

[4] L. Bordeaux, G. Salaun, D. Berardi, and M. Mecella. When are two web services compatible? *Lecture Notes in Computer Science*, 3324:15–28, 2005.

[5] R. I. Brafman, C. Domshlak, and S. E. Shimony. Introducing variable importance tradeoffs into cp-nets. In *Proc. of Uncertainity in Artificial Intelligence*, pages 69–76, 2002.

[6] R. I. Brafman, C. Domshlak, and S. E. Shimony. On graphical modeling of preference and importance. *Journal of Artificial Intelligence Research*, 25:389424, 2006.

[7] D. B. Claro, P. Albers, and J. K. Hao. Selecting web services for optimal composition. In *Proc. of ICWS 2005*.

[8] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 2001.

[9] P. Fishburn. Utility theory for decision making. 1970.

[10] S. French. *Decision theory: an introduction to the mathematics of rationality*. Halsted Press, New York, USA, 1986.

[11] M. P. Huhns and M. P. Singh. Service-oriented computing: Key concepts and principles. *Internet Computing*, 9(1):75–81, 2005.

[12] F. Liu, L. Zhang, Y. Shi, L. Lin, and B. Shi. Formal analysis of compatibility of web services via ccs. In *Proc. of the International Conference on Next Generation Web Services Practices*, page 143. IEEE Computer Society, 2005.

[13] A. Martens, S. Moser, A. Gerhardt, and K. Funk. Analyzing compatibility of bpel processes. In *International Conference on Internet and Web Applications and Services*, 2006.

[14] M. Papazoglou. Service-oriented computing: concepts, characteristics and directions. In *Proc. of the Fourth International Conference on Web Information Systems Engineering*, pages 3–12. IEEE Computer Society, 2003.

[15] J. Pathak, S. Basu, and V. Honavar. On context-specific substitutability of web services. In *Proc. of the International Conference on Web Services*, pages 192–199. IEEE Computer Society, 2007.

[16] J. Pathak, S. Basu, R. Lutz, and V. Honavar. Selecting and composing web services through iterative reformulation of functional specifications. In *Proc. of the 18th International Conference on Tools with Artificial Intelligence*, pages 445–454. IEEE Computer Society, 2006.

[17] G. R. Santhanam, S. Basu, and V. Honavar. Tcp-compose$\star$ - a tcp-net based algorithm for efficient composition of web services using qualitative preferences. In *Proc. of International Conference on Service-Oriented Computing*, volume 5364 of *LNCS*, pages 453–467, 2008.

[18] S. L. Schneider and J. Shanteau. *Emerging perspectives on judgment and decision research*, page 207. Cambridge University Press, 2003.

[19] S. Sohrabi, N. Prokoshyna, and S. McIlraith. Web service composition via generic procedures and customizing user preferences. In *Proc. of ISWC 2006*, pages 597–611.