Speeding Up Web Service Composition with Volatile External Information

John Harney
LSDIS Lab, Dept. of Computer Science
University of Georgia, GA 30602
jfh@cs.uga.edu

Prashant Doshi
LSDIS Lab, Dept. of Computer Science
University of Georgia, GA 30602
pdoshi@cs.uga.edu

ABSTRACT

Existing methods for composing Web services assume that compositions are fixed. However, in practice, service environments are often *volatile* and parameters of participating services may change during the composition. This paper introduces a novel method for composing Web services in the presence of external volatile information. Our approach, which we call the *informed-presumptive* is compared to previous state-of-the-art approaches for Web service composition in volatile environments. We show empirically that the informed-presumptive strategy produces compositions in significantly less time than the other strategies with lesser backtracks.

Categories and Subject Descriptors

H.3.5 [**Online Information Systems**]: Web-based Services

General Terms

Algorithms, Theory

Keywords

Web services, Adaptation, Volatile environments, Expiration times

1. INTRODUCTION

Composing Web services is an important area of study in services oriented computing. It offers the potential for automatically formulating processes in an efficient and timely manner. Web service compositions (WSCs) are predominantly modeled as being *static* – the participating services are assumed to exhibit both fixed quality of service (QoS) parameters and service output values (once invoked) during the composition and execution. However, in practice, WSC environments are often *volatile* – changes in the service parameters may occur during the composition. Therefore, WSC procedures must operate in the presence of this volatility. As a concrete example, consider a trip planning process realized as a WSC. It involves booking an airline ticket, reserving a rental car and obtaining a hotel room for a weekend vacation. The objective of the WSC is to assemble the least expensive itinerary possible among a group of eligible services. The WSC first selects an airline ticket based on the cost of the ticket. Next, the WSC selects a rental car, again based on the cost. While finding a rental car, the price of the airline ticket chosen previously increases. A traditional WSC continues to select the vacation at a greater cost, while a less expensive ticket available on another airline could be selected using another service.

The above example demonstrates that the static WSC's failure to adapt to volatile information will yield a suboptimal process. It reveals a common phenomenon affecting Web service compositions called *data volatility*. Data volatility manifests when the cost (as well as other parameters) of the Web service change during composition. In an earlier line of work, Au et al. [2, 3] introduced a framework that composes processes in the presence of data volatility. If the relevant data of participating services are no longer valid or known to have changed, the proposed adaptive composition techniques query for the changed parameters and decide when to incorporate the changes to create a new composition.

Three techniques [3] were grouped together in a framework called the *reactive query policy*. A reactive query policy is a set of rules used by the WSC procedure to decide when to incorporate the queried parameters that had expired. Three policies called the *eager*, *lazy*, and *presumptive* were proposed. The approaches, although disparate in many ways, all compare the new value of a parameter obtained from a query to the previous value of that parameter in the WSC. If the new parameter is different in value, the procedure backtracks to the step in the composition where the service with the revised parameter is being used. A significant limitation of these approaches is that some backtracks are not necessary – changes in a parameter value do not necessitate changes in the composition. These redundant backtracks lead to longer process composition times, and, for extremely volatile environments, compositions that may never complete.

In this paper, we introduce a new reactive query policy, which we call the *informed-presumptive*. Our approach predicts whether the newly obtained parameter information will cause significant changes in the composition. If a change in the composition is *expected* to occur due to the revised information, the strategy backtracks to the point in the process at which the changed parameter is used. Otherwise, it ignores the changed parameter value, as the change does not produce any revision in the composition. Thus, we seek to find the parameter values that are expected to induce a change in the composition. Utilizing the technique of *gradient descent* [11], we find these value ranges with minimal computational overhead.

Using the informed-presumptive policy with our underlying WSC procedure, we form compositions in less time compared to the previous best approaches mentioned above. This is because we effectively eliminate many of the backtracks that are not required. We

CSSSIA 2008, April 22, 2008, Beijing, China.

theoretically show that the informed presumptive approach does at least as good as the existing approaches and, in many cases, better. We empirically show using a simulated volatile environment that it incurs a lower average number of backtracks than the other policies by eliminating unnecessary ones. Invariably, the reduction in backtracks leads to better average composition times and for extremely volatile environments, a better chance of completing compositions.

2. RELATED WORK

Much of the research in the area of managing Web service compositions in volatile environments stem from previous work in developing dynamic workflows. These early models for adaptive workflows focused on volatility at different levels. In Adeptflex [13], graph based techniques were used to evaluate the feasibility and correctness of changes at the process instance level. Muller et al. [12] propose a workflow adaptation strategy based on pre-defined event-condition-action rules that are triggered when a change in the environment occurs. While the rules provide a good basis for performing contingency actions, they seldom account for all possible adaptive scenarios that may arise in complex workflows. Stohr and Zhao [14] devise the Business Process Adaptation Model, which seeks to decide how changes in business technologies may affect the needs of business process automation. They focus on categorizing the changes and choosing new technologies that benefit the general needs of an organization. Van der Aalst et al. [15, 16] also addressed the need for more flexible business processes. The focus, however, was on adaptation to changes in the functionality of the process, rather than adaptation to volatile data.

The Web services paradigm offered new research opportunities for designing adaptive workflows in Web environments. Desai et al. [6, 7] focus on adapting Web service compositions using handcrafted protocols. These business process protocols were created primarily to alleviate problems of heterogeneity and to support autonomy among different WS providers. The protocols may be changed to adapt to processes that require external events such as exception handling and frequent changes in participating providers' business models. However, the focus is on adapting to varying process functionality rather than data volatility.

Preliminary research on adaptation in volatile WSC environments focuses on interleaving service composition with execution. Doshi et al. [8] offer a technique that manages the dynamism of Web process environments using Bayesian learning. The process model parameters are updated based on previous interactions with the individual Web services and the composition plan is regenerated using these updates. This method suffers from being slow in updating the parameters, and the approach may result in plan (process flow) re-computations that do not bring about any change in the Web process. Gotz and Mayer-Patel [9] incorporate multidimensional data adaptation in the context of multimedia applications. Here the authors use a metric similar to the value of information (VOI) to determine if new information may impact the utility of the application. In a similar vein, previous work [10], used a metric analogous to the VOI called the value of changed information (VOC). The VOC is a statistical measure that predicts how poorly a composition will perform in a changed environment. The VOC is used to determine if it is worthwile to query for new information and recompose the process. In Chafle et al. [4, 5] several alternate plans are pre-specified at the logical level, physical level and the runtime level. Depending on the type of changes in the environment, alternative plans from these three stages are selected. While capable of adapting to several different events, many of the alternative prespecified plans may not be used making the approach inefficient. Further, there is no guarantee of optimality of the resulting WSC.

In a somewhat different vein, Verma et al. [17] explore adaptation in Web processes in the presence of coordination constraints between different WSs. This work is complementary as we do not consider such constraints here.

We introduce a new approach that improves on previous techniques by Au et al. [2] for composing with volatile external information. In [3], they introduce reactive query policies as ways to handle data volatility. We explain these techniques in greater detail in Section 5.

3. MOTIVATING SCENARIO

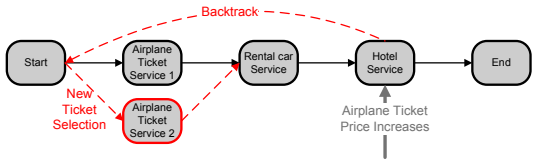


Figure 1: A 3-step travel plan composition with backtracking.

In order to illustrate our approach, we present a travel planning WSC, where a user may create a vacation package. The goal of the WSC is to provide the user with the least expensive travel itinerary chosen from a given set of candidate Web services.

Fig. 1 illustrates three steps that are needed in the composition. The WSC begins by searching through several candidate airline services (such as Delta, Continental, etc) that offer tickets for their flights at a given time specified by a user. The WSC selects the service that offers the flight with the lowest overall cost – an aggregation of both the price of the ticket of the flight and a service usage fee – in addition to other factors such as the availability. Suppose that Airline Ticket Service 1 is selected because the ticket it offers is cheaper than Airline Ticket Service 2. Upon completion, the WSC progresses to finding a rental car. Analagous to selecting the flight, the WSC chooses a rental car service from a set of candidate services based on overall cost. The WSC finishes by selecting an optimal hotel service. Ultimately, the WSC selects the best combination of airline, rental car and hotel room services. Once the selection is completed, the booking is carried out.

Suppose that as the WSC selects the hotel room service, the price of the ticket that Airline Ticket Service 1 had offered suddenly increases. The cost of using Airline Ticket Service 1 may now be more expensive than using another airline service, say Airline Ticket Service 2, who offer a ticket at a lower price. The WSC will have to backtrack to the point where the airline service was chosen, so that it may have the opportunity to select Airline Ticket Service 2. The WSC may now continue from that point onward, effectively maintaining a lower itinerary cost.

4. GREEDY COMPOSITION OF WEB SERVICES

In this paper, we utilize a greedy method of composing Web services although our approach is not limited to any particular method of composition. The WSC process, P, is defined as follows:

$$P = \{S, s_0, A, \hat{A}, s_g, C, Av\}$$

where: • S is the set of all states in the composition;
• $s_0 \in S$ is the initial state of the composition;
• A is the set of all candidate Web services that may be used in the composition;
• $\hat{A} : S \rightarrow 2^A$, where 2^A is the power set of A, is the set of Web

services that could be invoked from a given state;
• $s_g \in S$ is the goal state of the composition;
• $C : S \times A \rightarrow \mathbb{R}$ is the overall cost of using each Web service; and
• $Av : SA \rightarrow [0, 1]$ is the availability of each Web service.

Algorithm for composing Web services
Input: s_0
$s \leftarrow s_0$
while s_g not reached
For all $a \in \hat{A}(s)$
Use Eq. 2 to find V^a
end for
Use Eq. 1 to find $\pi^*(s)$
$a_{max} \leftarrow \pi^*(s)$
Using a_{max}, construct next state, s'
$s \leftarrow s'$
end while
end algorithm

Figure 2: Composition of Web services.

For the sake of simplicity, we assume that C and Av are a function of individual Web service $a \in A$ only; thus we denote that each Web service $a \in A$ has an associated cost, C^a, and availability, Av^a. We also note that although we focus on service cost and availability metrics, other parameters such as response times may also be included.

In our scenario, we let C^a be the sum of both the cost of using service a and the cost retrieved by the ouput of service a (eg the price of an airline ticket). We assign Av^a in an analagous manner.

For each $s \in S$, there exists a subset of Web services $\hat{A}(s) \subseteq A$ which may be invoked at state s. The optimal service, $a_{max} \in \hat{A}(s)$, is the service which results in the maximum expected value at the state s:

$$\pi^*(s) = \underset{a \in \hat{A}(s)}{argmax} \ V^a \tag{1}$$

In Eq. 1, π^* is the optimal strategy which is a mapping from states to the optimal service to invoke, $\pi^* : S \rightarrow A$.

The formulation of our value function, V^a, borrows from [1], where an integer linear program (ILP) finds the services to invoke from a particular state. For a greedy composition, we need not use the more complex ILP – a simple maximization over the value function suffices. We define a value function that is linear over the service parameters.

$$V^a = W_C * \bar{C}^a + W_{Av} * \bar{A}v^a \tag{2}$$

where: $W_C + W_{Av} = 1$

W_C, $W_{Av} \in [0, 1]$ are the importance weights for the parameters, cost and availability, respectively; $\bar{C}^a$ and $\bar{A}v^a$ are the cost and availability for a service a, respectively, normalized between 0 and 1 as follows:

$$\bar{C} = \begin{cases} C^{Max} - C^a / C^{Max} - C^{Min} \\ \qquad if \quad C^{Max} - C^{Min} \neq 0 \\ 1 \\ \qquad if \quad C^{Max} - C^{Min} = 0 \end{cases} \tag{3}$$

Here, C^a is the cost of service a and C^{Max} and C^{Min} are the maximum and minimum service costs of all the services in A, respectively. We scale the availability in an analagous manner.

Fig. 2 outlines the algorithm for the greedy WSC. The process starts at s_0 and evaluates the value of each Web service that could be invoked from s_0. The optimal service a_{max} is then found, leading to the next state, s'. The process iteratively forms the composition until the goal state is reached.

5. BACKGROUND: REACTIVE QUERY POLICIES FOR VOLATILE INFORMATION

Au et al. [3] introduced multiple approaches for performing service composition in volatile environments. These approaches associate expiration times with the service parameters, possibly through SLA's defined by standards like WS-Agreement [18]. When the parameters of the component services expire, an event is triggered that prompts the WSC to issue a query to an information source for the updated information. If the parameter data is different from the previous ones, the procedure backtracks to the step in the composition where the service with the revised parameters is being used.

Au et al. use *reactive query policies* to decide when the queried information should be incorporated within the composition. Three reactive query policies, the *eager*, *lazy* and *presumptive* are used, which we briefly explain below.

When service parameters expire, the eager approach immediately issues a query to the service and suspends the composition until the responses are obtained. We illustrate the eager policy in the context of our trip planning example in Fig 3(a). After selecting the rental car, let the airline ticket cost change. The composition halts until the new price of the ticket is obtained. When the queried information arrives, the WSC checks if the new price is the same as the previous one. If they are different, the WSC backtracks to the airline service composition step, otherwise it continues its composition. This continues until the goal state is reached and no expired parameters remain.

A WSC that uses the lazy approach will issue new queries when the information expires, but does not immediately introduce the changes. Instead, it assumes that the previous information remains valid and continues composing until the process completes. At that point, the lazy approach suspends execution of the completed WSC until all of the responses are received. If there is no change in the parameter value, the composition is completed. Otherwise, it will backtrack to the step in the composition where the service with the revised parameter is being used. As we show in Fig 3(b), the lazy approach will continue until all steps of the composition have been completed, despite the airline ticket expiring in the middle of the composition.

The presumptive policy (Fig 3(c)) performs similar to the eager, except that the composition process is not halted while waiting for the revised information. Backtracks, if any, occur when the revised information is received. For our example, the process on finding the appropriate car rental service receives the query results while attempting to select a hotel reservation service. When it receives the result of the query, it may backtrack if needed.

The advantage of the presumptive policy is that it merges the strengths of both the lazy and eager approaches. Similar to the eager approach, presumptive compositions capture and introduce revised information quickly. The presumptive approach does not suffer from the time lag needed to wait for revised information unlike the eager approach. In this, it resembles the lazy approach because it will continue the composition as if the information is valid, and only backtrack, if needed, when the revised information is received. An empirical analysis of these methods is presented in [3].

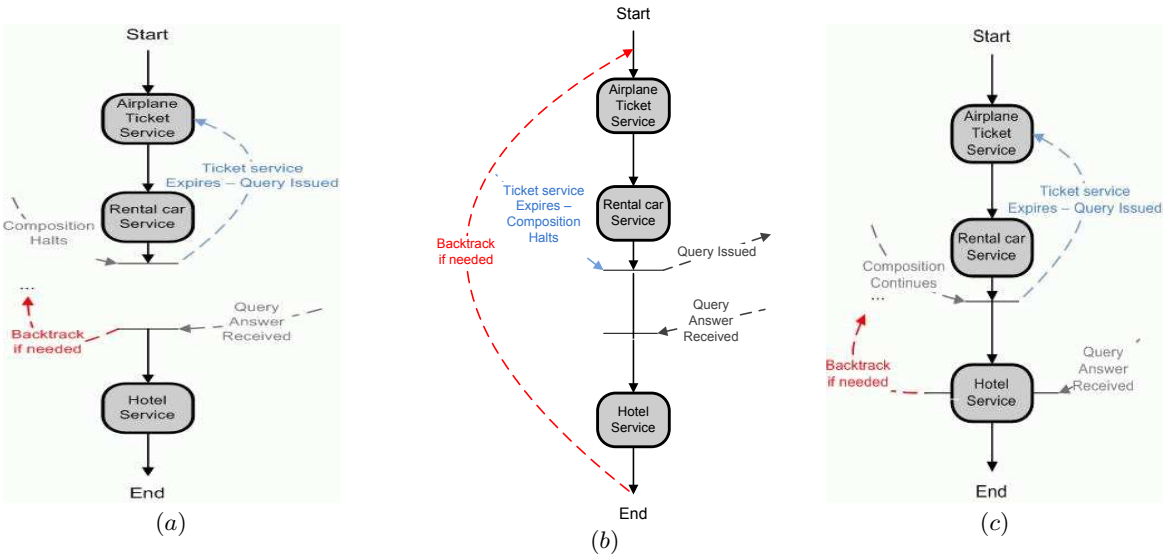


Figure 3: (a) **Eager policy halts the WSC until a response to its query is received.** (b) **Lazy and** (c) **Presumptive approaches do not halt the composition. However, the *lazy* policy does not incorporate the changes until the end of the WSC is reached.**

6. IMPROVED QUERY POLICY: INFORMED-PRESUMPTIVE

A significant limitation of the previous approaches is that some of the backtracks are redundant – changes in the parameter values do not necessitate changes in the composition. For example, an increase in the cost of an airline ticket is not likely to affect the WSC until the cost exceeds that of a ticket from another provider. The previous strategies, however, will backtrack and recompose, because the value of the cost parameter has changed. These redundant backtracks hinder the progress of the WSC and increase the completion time.

6.1 Description

To avoid the excessive backtracking, we must determine if the revised parameters are expected to impact the composition. Let $\vec{p}^a$ denote the *revised* parameter value vector for some service a:

$$\vec{p}^a \stackrel{def}{=} [\bar{C}'^a, \bar{A}v'^a] \tag{4}$$

where $\bar{C}'^a$ and $\bar{A}v'^a$ are the revised parameters obtained in response to a query of service a. (Of course, $\vec{p}^a$ can be extended to account for other parameters as well.)

Because a value function that remains unchanged in the presence of revised parameters indicates no change in the composition, we seek those revised parameter vectors for each service that do not change the value. Let $V^{\pi}(\vec{p}^a)$ be the value of the original composition, π, in the presence of revised parameter vector $\vec{p}^a$. Note that π is optimal for the original parameter values. Let $V^{\pi^*}(\vec{p}^a)$ denote the value of following the new optimal strategy, π^*, in the presence of $\vec{p}^a$. A difference between $V^{\pi^*}(\vec{p}^a)$ and $V^{\pi}(\vec{p}^a)$ implies that the original strategy, π, differs from the new optimal π^*. If π and π^* are different, the revised parameters have changed the recommended actions in the composition.

Our new approach, called the *informed-presumptive*, extends the previously described presumptive approach by identifying the parameters values for each service that are not expected to change the composition. Specifically, we determine all $\vec{p}^a$ where:

$$V^{\pi^*}(\vec{p}^a) - V^{\pi}(\vec{p}^a) = 0. \tag{5}$$

Note that Eq. $5 \geq 0$. Of course, there exists at least one set of values, $\vec{p}^a$, for which Eq. 5 will be true.

THEOREM 1. *For a WSC, there exists at least one parameter vector, $\vec{p}^a = \{\bar{C}'^a, \bar{A}v'^a\}$, for which $V^{\pi^*}(\vec{p}^a) = V^{\pi}(\vec{p}^a)$.*

PROOF. Let $\vec{p}^a$ be equal to the current value(s) of the parameters of service a. In other words, the revised values for service a after expiration are the same as the previous ones. Given that all other parameters remain unchanged, the new optimal strategy π^*, when $\vec{p}^a = \{\bar{C}'^a, \bar{A}v'^a\}$ is equal to the current one. Therefore, $V^{\pi^*}(\vec{p}^a) = V^{\pi}(\vec{p}^a)$. □

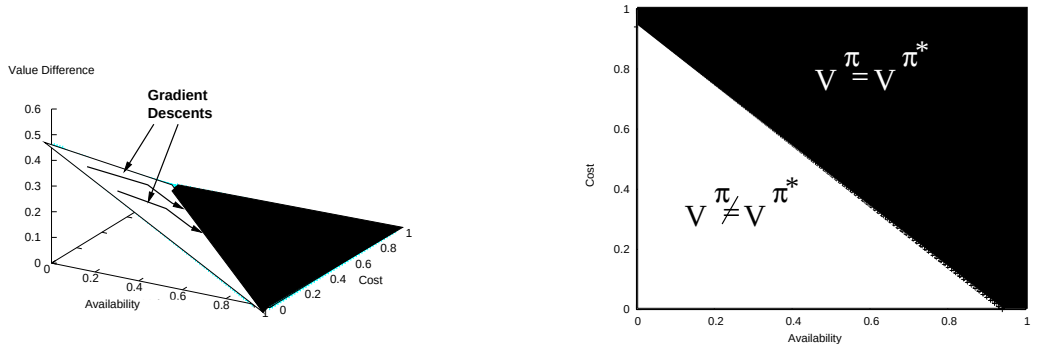
While Theorem 1 points to the existence of at least one $\vec{p}^a$ that trivially satisfies Eq. 5, typically this set of vectors tends to be larger.

We refer to Fig $4(a)$, which shows a plot of Eq. 5 for an arbitrary service. We used $W_C = 0.45$ and $W_{Av} = 0.55$ in generating the plot. The plot is of three dimensions, $\bar{C}'^a$ (cost), $\bar{A}v'^a$ (availability), and the value difference, $V^{\pi^*}(\vec{p}^a) - V^{\pi}(\vec{p}^a)$, on the z-axis. We show a projection of the surface in Fig. $4(b)$. Notice the shaded region of the surface where, $V^{\pi^*}(\vec{p}^a) - V^{\pi}(\vec{p}^a) = 0$. The values of cost and availability that lie in this region do not induce a change in the composition. Hence, no backtrack and subsequent recomposition is needed if the revised, $\vec{p}^a$, lies in this region.

The existence of the shaded region in Fig. $4(b)$ leads us to the following corollary.

COROLLARY 1. *The total composition time required for a WSC that uses the informed-presumptive policy will be less than or equal to the composition time required for the same WSC using the presumptive policy.*

PROOF. We begin by considering the converse – WSC using the presumptive approach takes less time (backtracks less) than when the informed-presumptive approach is used – and show that it is not possible. Thus, when $\vec{p}^a$ expires, there is at least one vector for which the informed-presumptive based WSC will backtrack and presumptive based WSC will not. As we mentioned in Section 5, the presumptive approach will not backtrack only if the new $\vec{p}^a$ is identical to the current parameter values. However, from Theorem 1 we see that if $\vec{p}^a$ is the same as the current, the composition is unchanged and the informed-presumptive will not backtrack either.



(a) (b)

Figure 4: (a) **Gradient descent on the surface** $E(\vec{p}^a)$**.** (b) **The projection of the gradient on the parameter (**$x - y$**) axes. Here, vectors** $\vec{p}^a$ **that lie in the region denoted by** $V^a_{\pi^*}(\vec{p}^a) \neq V^a_{\pi}(\vec{p}^a)$ **induce a change in the composition while the** $\vec{p}^a$ **that lie in the other region denoted** $V^{\pi^*}(\vec{p}^a) = V^{\pi}(\vec{p}^a)$ **do not.**

Hence, there is no $\vec{p}^a$ where the informed-presumptive will backtrack and the presumptive will not. Thus, the presumptive approach cannot take less time than the informed-presumptive approach, and the converse is not true.

We now show that there exists at least one case where the informed-presumptive will yield a lower composition time than the presumptive approach due to lesser backtracking. This occurs when the revised $\vec{p}^a$ is not the same as the original one, but satisfied Eq. 5. Such a vector appears in the shaded region in Figure 4(b). For this case, the presumptive policy based WSC backtracks thereby consuming more time than the informed presumptive, which does not backtrack. □

Thus, for any revised parameter vector $\vec{p}^a$ satisfying Eq. 5, the informed-presumptive policy will avoid an unnecessary backtrack because there is no change in π^*.

6.2 Gradient Descent

We observe that a direct solution of Eq. 5 to find the parameter vectors is not trivial because of the presence of the maximization operator in computing π^*. Alternately, we may view the difference in Eq. 5 as the error in using the original composition given the revised parameter vector, $\vec{p}^a$, and denote the difference as $E(\vec{p}^a)$. This allows us to model the problem of finding the vectors satisfying Eq. 5 as a *gradient descent* [11] where we descend down the error surface, $E(\vec{p}^a)$, until we reach the minimum plateau ($E(\vec{p}^a) = 0$). To perform the gradient descent, we update $\vec{p}^a$ as follows:

$$\vec{p}^a \leftarrow \vec{p}^a + \Delta\vec{p}^a \tag{6}$$

where

$$\Delta\vec{p}^a = -\eta \nabla E(\vec{p}^a) \tag{7}$$

Here, η is the step size, $0 \leq \eta \leq 1$. The negative sign indicates that we take a step in the direction of the reducing gradient. For our application, definition of $\nabla E(\vec{p}^a)$ is the vector:

$$\nabla E(\vec{p}^a) = [\frac{\partial E(\vec{p}^a)}{\partial \bar{C}^a}, \frac{\partial E(\vec{p}^a)}{\partial \bar{A}v^a}] \tag{8}$$

Fortunately, Theorem 2 simplifies the task of finding ∇E($\vec{p}^a$).

THEOREM 2. *Given that V^a is a linear function of the parameters of a service, a, the gradient $\nabla E(\vec{p}^a)$ is constant.*

We show the proof of this theorem in Appendix 1. Theorem 2 allows us to perform gradient descent with minimal computational overhead, as we show in our experiments. Furthermore, the typical drawback of gradient descent of getting stranded on a local rather than the global minima is not a concern here because of the fixed gradient leading to a single minima in the error landscape.

Observe that a single gradient descent generates one vector at which $E(\vec{p}^a) = 0$. However, as we see in Fig. 4(b), the boundary separating the shaded region from the unshaded one is a line, which requires us to find at least two points. Subsequently, we perform another gradient descent from a starting point that is distinct from the previous one. This provides the second parameter vector at which $E(\vec{p}^a) = 0$, enabling us to formulate the boundary. Given the boundary line, we simply need to check whether the revised parameter values, $\vec{p}^a$, lie on the side of the unshaded region in order to backtrack.

We compute the line boundaries separating the regions that induce a backtrack from those that do not, for each service participating in the composition. Note that if we utilize more than two parameters, the boundary is a hyperplane.

7. EXPERIMENTS

We implemented our travel planner in a simulated volatile environment using each of the four reactive query policies to form a WSC. We view each component of the travel plan (ie. selecting a service to book flight, renting the car and reserving a hotel room) as a single step (state) in the composition. The WSC chooses among 10 vendor services at each step, with varying cost of providing the service and availability. Each data point in our plots is the average of 1000 runs of the WSC. We let the time needed to respond to a query be fixed.

Let t^s_{comp} be the time a step s in the composition takes to select a service, and t^a_{exp} be the expiration time of service a. We introduce two important measurable characteristics of volatile environments. First, is the *volatility ratio*, $\frac{t^s_{comp}}{t^a_{exp}}$. This ratio is a measure of how often a selected service will expire while composing the next step. Hence, it is indicative of the volatility of the environment. The second characteristic is the likelihood (probability) that upon expiration, a service's parameter will change.

We first measured the average composition time for each of the four approaches. Figures 5(a), (b) and (c) show the times of these simulations for volatility ratios of 1 (highly volatile environment), 0.5 (medium volatility), and 0.33 (low volatility). Lower composition times indicate better performance. We note that the composition times include the time taken to perform the gradient descents

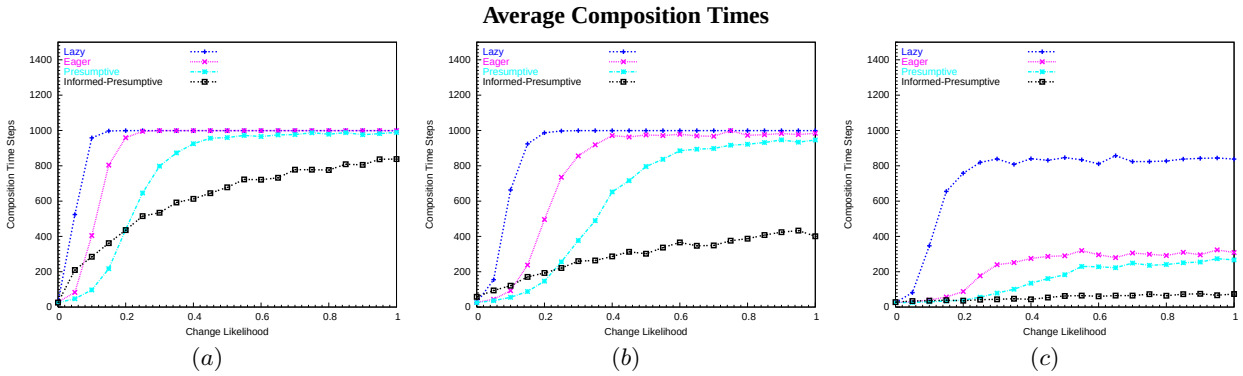


Figure 5: Average composition times for the four approaches with volatility ratios of (a)1, (b)0.5, (c)0.33. The informed-presumptive policy obtains significantly lower composition times and all times reduce as the environment becomes less volatile (lower volatility ratio).

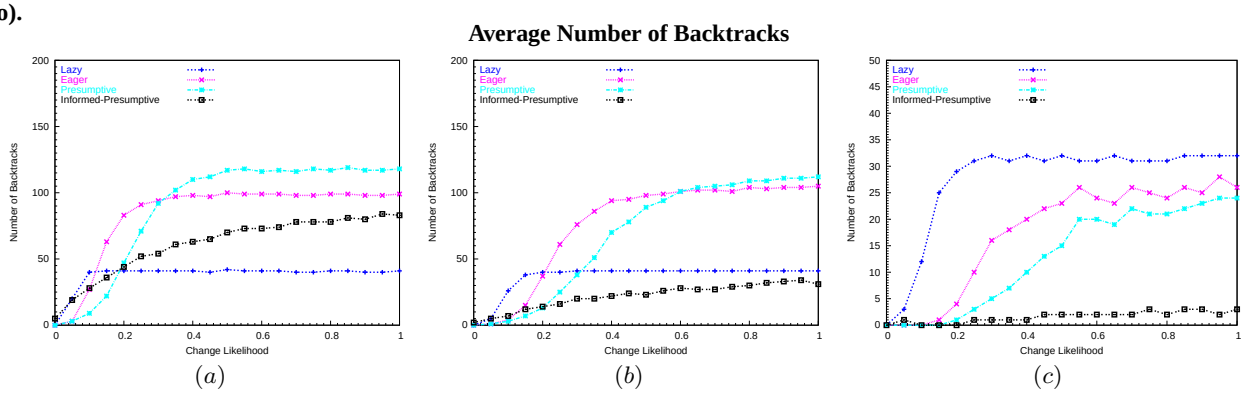


Figure 6: Average number of backtracks for the four approaches with volatility ratios of (a)1, (b)0.5, (c)0.33. Lesser number of backtracks is better. Notice that the lazy backtracks less than the informed-presumptive when the environment is highly volatile (a), although its composition time remains higher.

for the participating services when the informed-presumptive policy is used. In highly volatile environments (Fig. 5(a)), a WSC using the informed-presumptive policy forms in significantly less time than the other policies. However, at a low likelihood of change (≤ 0.2), the informed-presumptive performs slightly worse as precision errors due to overstepping in the gradient descent become visible. In Figs. 5 (b) and (c), we show the results for lower volatility ratios 0.5 and 0.33, respectively. Here, the informed-presumptive approach achieves a significantly lower average composition time than the others.

As we show in Fig. 6, the informed-presumptive performs lesser number of backtracks on average during its composition than some of the other approaches. In particular, at lower ratios of volatility (Figs. 6 (b), (c)), the informed-presumptive backtracks significantly less than the other methods (the exception being at a low likelihood of change). The lesser numbers of backtracks are an important factor in the lower composition times of the informed-presumptive based WSC.

However, Fig. 6 (a) demonstrates that a WSC using a lazy approach performs less backtracks than the informed-presumptive. This is because backtracks are deferred until the end of the WSC as per the lazy policy. Yet, it does not translate to lower composition times as we see in Fig. 5(a). This is because in waiting for the *entire* travel plan to be composed before checking for revised parameters, the lazy often backtracks over more steps in the composition than the informed-presumptive. On measuring the average number of steps per backtrack, we found that the lazy's average exceeded that of the informed-presumptive by a wide margin. Consequently, the lazy performs many redundant step compositions only to be changed at the end.

Finally, we measure the number of incomplete compositions, as shown in Figure 7. We set a ceiling of 1000 time steps for the WSC to complete – if the composition has not terminated by then, we assume that it will never terminate. Such situations are common in volatile environments as the parameters change often and subsequently, backtracks are frequent. Our results indicate that the informed-presumptive approach is more likely to complete the composition (by a factor of 5 in most situations), even in highly volatile environments, in comparison to the other approaches.

In summary, our experimental results conclusively show that the informed-presumptive improves on the previous policies for WSC in volatile environments. Due to Theorem 2, we expect that the computational overhead in performing the gradient descent will remain low when more than two parameters are used. We will further explore this issue empirically by considering multiple parameters.

8. CONCLUSION

Previous approaches for managing WSC in volatile environments often suffer from long composition times and delays, due to unnecessary backtracking. We have shown that, in using the informed-presumptive approach, we may eliminate these backtracks by predicting whether changed parameters induce changes in the composition. These predictions were achieved by applying gradient descent and accomplished with minimal overhead. As a result, we obtained significant speed up in composing the WSC.

When complex non-linear value functions and dependencies are

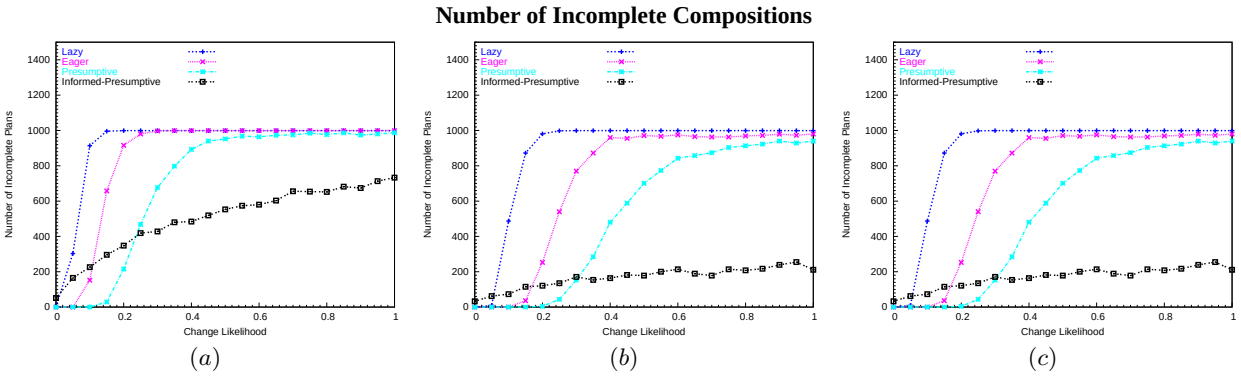


Figure 7: The number of incomplete compositions for the four approaches with volatility ratios of (a)1, (b)0.5 and (c)0.33. A WSC with the informed-presumptive policy is more likely to complete in comparison to when other policies are used.

introduced to the composition, performing gradient descent becomes an arduous task. Our future work will explore implementing our approach on more complex compositions.

9. REFERENCES

[1] V. Agarwal, G. Chafle, K. Dasgupta, N. Karnik, A. Kumar, S. Mittal, and B. Srivastava. Synthy: A system for end to end composition of web services. *JWSR*, 3(4), 2005.

[2] T.-C. Au, U. Kuter, and D. S. Nau. Web service composition with volatile information. In *ISWC*, 2005.

[3] T.-C. Au and D. Nau. Reactive query policies: A formalism for planning with volatile external information. In *CIDM*, 2007.

[4] G. Chafle, K. Dasgupta, A. Kumar, S. Mittal, and B. Srivastava. Adaptation in web service composition and execution. In *ICWS, Industry Track*, 2006.

[5] G. Chafle, P. Doshi, J. Harney, S. Mital, and B. Srivastava. Improved adaptation of web service compositions using value of changed information. In *ICWS, Industry Track*, 2007.

[6] N. Desai, A. K. Chopra, and M. P. Singh. Business process adaptations via protocols. In *SCC*, pages 601–608, 2006.

[7] N. Desai, A. U. Mallya, A. K. Chopra, and M. P. Singh. Owl-p: A methodology for business process modeling and enactment. In *Agent-Oriented Information Systems-III, LNCS, Volume 3529*, pages 79–94, 2006.

[8] P. Doshi, R. Goodwin, R. Akkiraju, and K. Verma. Dynamic workflow composition using markov decision processes. *JWSR*, 2(1):1–17, 2005.

[9] D. Gotz and K. Mayer-Patel. A general framework for multidimensional adaption. In *ICME*, pages 612–619–126, 2004.

[10] J. Harney and P. Doshi. Speeding up adaptation of web service compositions using expiration times. In *WWW*, 2007.

[11] T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[12] R. Muller, U. Greiner, and E. Rahm. Agentwork: a workflow system supporting rule-based workflow adaptation. *DKE*, 51(2):223–256, 2004.

[13] M. Reichert and P. Dadam. Adeptflex-supporting dynamic changes of workflows without losing control. *JIIS*, 10(2):93–17, 1998.

[14] E. Stohr and J. Zhao. A technology adaptation model for business process automation. In *HICSS*, 1997.

[15] W. van der Aalst. Generic workflow models: How to handle dynamic change and capture management information. In *ICCIS*, pages 115–126, 1999.

[16] W. van der Aalst. Exterminating the dynamic change bug: A concrete approach to support workflow change. In *Information Systems Frontiers*, pages 297–317, 2001.

[17] K. Verma, P. Doshi, K. Gomadam, J. Miller, and A. Sheth. Optimal adaptation in web processes with coordination constraints. In *ICWS*, 2006.

[18] Alain Andrieux, Karl Czajkowski, Asit Dan, Kate Keahey, Heiko Ludwig, Toshiyuki Nakata, Jim Pruyne, John Rofrano, Steve Tuecke, and Ming Xu. WS-Agreement Specification. In 2005.

10. APPENDIX: PROOF OF THEOREM

PROOF. We begin by substituting $\mathrm{E}(\vec{p}^{\,a})$ with $V^{\pi^*}(\vec{p}^{\,a}) - V^{\pi}(\vec{p}^{\,a})$ as per Eq. 7 and use this substitution in Eq. 8:

$$\nabla E(\vec{p}^{\,a}) = [\frac{\partial(V^{\pi^*} - V^{\pi})}{\partial \bar{C}}, \frac{\partial(V^{\pi^*} - V^{\pi})}{\partial \bar{A}v}] \tag{9}$$

Then, we separate each of the terms:

$$\nabla E(\vec{p}^{\,a}) = [\frac{\partial(V^{\pi^*})}{\partial \bar{C}} - \frac{\partial(V^{\pi})}{\partial \bar{C}}, \frac{\partial(V^{\pi^*})}{\partial \bar{A}v} - \frac{\partial(V^{\pi})}{\partial \bar{A}v}] \tag{10}$$

Let us focus on the V^{π^*} term in the gradient vector.

Case 1 : Let a be a_{max}, recommended by π^*. If revised parameters do not induce a change in the composition, then, because $\pi = \pi^*$, $V^{\pi} = V^{\pi^*}$ and the difference (and thus its derivative) will be 0. If the revised parameters induce a change in the composition, then $V^{\pi} \neq V^{\pi^*}$. However, V^{π^*} will be a constant term. This is because a_{max} is no longer the service recommended by π^* and another candidate service, a_θ, will now be selected, as it is the service that yields the greatest value. The parameters of a_θ will remain constant, as we are varying the parameters of only a_{max}, thus V_{π^*} will remain constant.

Case 2 : Let a now be a service other than a_{max}. If the revised parameters do not induce a change in the composition, the V^{π^*} will remain constant, as a_{max} remains the optimal service. If the revised parameters induce a change in the composition, then $V^{\pi} = V^{\pi^*}$, and the resulting difference (and its derivative) will be 0.

In eliminating the $\frac{\partial(V^{\pi^*})}{\partial \bar{C}}$ and $\frac{\partial(V^{\pi^*})}{\partial \bar{A}v}$ terms, we have:

$$\nabla E(\vec{\mathbf{p}}^{\mathbf{a}}) = [-\frac{\partial(V^{\pi})}{\partial \bar{C}}, -\frac{\partial(V^{\pi})}{\partial \bar{A}v}] \tag{11}$$

Then, expanding V^{π} using Eq. 2:

$$\nabla E(\vec{\mathbf{p}}^{a}) = [-\frac{\partial(W_C \cdot \bar{C} + W_{Av} \cdot \bar{A}v)}{\partial \bar{\mathbf{C}}}, -\frac{\partial(W_C \cdot \bar{C} + W_{Av} \cdot \bar{A}v)}{\partial \bar{\mathbf{A}}\mathbf{v}}] \tag{12}$$

Finally, we differentiate and obtain:

$$\nabla E(\vec{\mathbf{p}}^{a}) = [-W_C, -W_{Av}] \tag{13}$$

As per Eq. 13, the gradient is constant.

□