On Modelled Flexibility and Service Selection Optimisation

Roland Ukor and Andy Carpenter

School of Computer Science, University of Manchester,
Oxford Road, Manchester M13 9PL, United Kingdom
{roland.ukor,andy}@cs.man.ac.uk

Abstract. Service selection optimisation is concerned with the optimal selection of services for the activities in a business process in such a way that maximises the overall quality of the process. Flexibility built into process models during the design phase enables a single process model to accommodate multiple execution paths any of which may be taken by a process instance during the execution phase. However, a challenge exists as to how services can be selected in a way that is simultaneously optimal for all execution paths.
This paper surveys relevant literature in service selection optimisation and reviews a number of approaches to managing the effects of modelled flexibility on service selection optimisation. It concludes that it is difficult to simultaneously guarantee an exact optimal solution for all paths and presents a characterisation of the solution space in which the existing approaches may provide the closest approximation.

1 Introduction

Business processes can be expressed as web service compositions in notations such as BPEL [1]. To execute service oriented business process, each service activity must be bound to an appropriate service implementation prior to execution. For simple processes, services may be implemented locally and bound to the activities during the definition of the business process. However, the emergence and rapid growth in the adoption of distributed application paradigms such as SOA and web services have facilitated integration of inter-organisational business processes; such that complex processes can easily be provisioned by bringing together functionality provided by web services distributed across different organisations and locations.

Selection of service implementations for activities in a business process often involves the evaluation of one or more candidate services for each activity where these candidates may be looked up in service directories. Typically, service implementations are selected based on functional criteria for each activity in the process. However, in the last decade, there has been various research efforts aimed at extending the selection criteria to include end-to-end quality characteristics and constraints in addition to functional requirements. This is generally referred to as service selection optimisation [2–4].

Modelled flexibility refers to the structural properties of a process that allows it to flexibly respond to different scenarios anticipated by the designers of the process. It is termed modelled flexibility (or flexibility by configuration) because it is built into the process model and supports only variations and exceptions that can be anticipated during the design phase of the process lifecycle. The most common result of modelled flexibility is a distinct set of execution paths that may be taken by any instance of the process during execution.

This paper surveys the literature on service selection optimisation and reviews the effects of modelled flexibility on the ability to simultaneously determine exact solutions to the service selection optimisation problem for all the execution paths in the process. The rest of the paper is structured as follows: section 2 provides a brief introduction to service selection optimisation which is followed by an introduction to modelled flexibility and execution paths. In section 4, a review of the effects of modelled flexibility on approaches to service selection optimisation is presented.

2 Service selection optimisation

Service selection optimisation (SSO) is an approach used to determine the optimal selection of service implementations for activities in a business process based on measurable QoS metrics. It assumes that services are listed in service directories according to the principles of the service oriented architecture and that these services are annotated with QoS metrics such as cost, duration, availability and reliability[1].

Most approaches to finding the optimal selection of services begin by first expressing the problem in terms of a structured problem model on which one or more algorithms are applied to find the optimal solution [2–6]. Given a set of activities $A = \{a_1, a_2, \cdots, a_n\}$ and a set of candidate services S_i for each a_i, a solution to the SSO problem can be represented as a set of pairs $\{(a_i, s_i)\}$ where each pair associates an activity a_i in the process with a candidate service $s_i \in S_i$ selected for that activity. An optimal solution is a solution that provides the desired extreme value for the objective function of the problem model. The objective function is a statement of the value or utility of any given solution.

As this paper focuses on examining the effects of modelled flexibility on service selection optimisation, it does not place emphasis on the efficiency or otherwise of the algorithms or heuristics used to find solutions to the problem. Rather, the discussion that follows employs the simple and well known exhaustive search algorithm for simplicity. The exhaustive search algorithm naively enumerates all possible sets of service selections and computes the statement of

[1] The source of these metrics although important in business terms is not as significant for designing optimisation solutions to the SSO problem in general. The values may either be advertised by the service providers or mined from local records of past service invocations. In addition, trusted third parties may collect and maintain accessible record of metrics values over a period of time

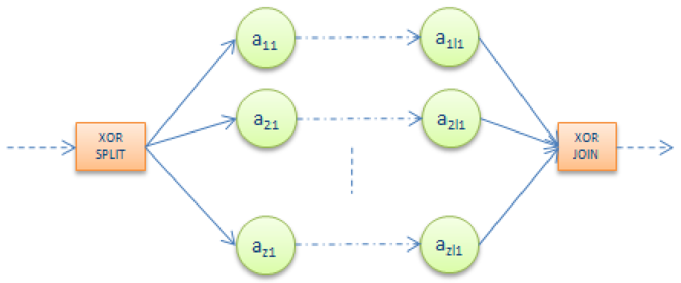

Fig. 1. Flexibility using XOR structural pattern

value for each solution. The solution which has the maximum value is considered the optimal solution. Exhaustive search is usually considered as an exact algorithm that is always expected to find the optimal solution to the problem given unlimited space and time.

While the literature accounts for numerous efforts in service selection optimisation, this paper only examines the aspects that address global optimisation (as opposed to local optimisation), enforce global constraints, support optimisation based on multiple QoS metrics, and supports arbitrary but well structured [7, 8] process models (as opposed to sequential models only).

3 Modelled flexibility and structural patterns

A business process is modelled as a set of activities connected by structural control flow patterns. There is a well established body of knowledge on structural patterns for business processes and workflows [9]. Modelled flexibility in business processes is reflected by the presence of specific structural patterns which provide decision points for conditionally varying the path of business process execution [10, 11]. These patterns make it possible for multiple execution paths to be represented by a single business process model; and their structure implies the ability to make runtime choices as to which set of activities should be executed next based on predefined conditions. These patterns are generally referred to as *choice patterns* and they include the OR and XOR structural patterns. The process fragment in figure 1 depicts the use of an XOR choice pattern to connect z different series of activities. During execution of this fragment, execution control can be passed from the XOR node to one of the activities $a_{11}, a_{21}, \cdots, a_{z1}$ depending on the evaluation of the transition conditions in the XOR node. In other words, there are z paths that could be taken by an instance of this process from the XOR node.

A business process may contain a number of such fragments with choice patterns, the combination of which leads to a number of distinct execution paths any of which may be taken by a process instance based on the evaluation of the decision nodes. Each execution path includes a distinct subset of process activities that will be executed. It is common to find business processes where execution paths have one or more activities in common. This is because the decision points giving flexibility are located after a number of activities have been executed (e.g. an activity to retrieve the profile of client in order to determine whether to execute the path of the process applicable to premium customers or basic customers).

The utility computed by the objective function is used to determine the relative suitability or otherwise of all possible solutions to an SSO problem. Consequently, it is absolutely critical that this statement of value is as precise as possible. In the absence of choice structural patterns in a process, the exhaustive algorithm is exact in that it will always find an optimal solution to the problem. This is because every activity in the process will be executed by all instances. Consequently, it is easy to provide precise statements about the end-to-end QoS of the process by aggregating QoS metrics for each activity in the process. However, this becomes contentious when choice structural patterns are introduced in the process.

In general, let P be a business process represented as a graph $(A \cup Y, E)$ where A is the set of activities, Y is the set of structural patterns and E the set of transitions in the process graph. Also, let $\mathbb{P}(P)$ be the set of z execution paths $\{P_1, P_2, \cdots, P_z\}$ contained in P where $P_i = \{a\colon a \in A$ and a is executed in path $P_i\}$ such that the set of all activities in $P, A = \bigcup_{i=1}^{z} P_i$. The following definitions are in order:

Definition 1. *For any two activities $a_i, a_j \in A$, a_i is a predecessor of a_j written as ($a_i \rightarrow a_j$) if there exists a path $P_k \in \mathbb{P}(P)$ in which a_i will be executed before a_j.*

Definition 2. *For any subgraph L of P, we say that $L_i \in \mathbb{P}(L) = \{L_1, L_2, \cdots, L_m\}$, is an enclosed path in P. Note that P is a subgraph of itself so any definitions in terms of L will also apply to P as a whole.*

Definition 3. *The common activities $C(L)$ of L is the set of process activities common to all execution paths in L.*

$$C(L) = \bigcap_{i=1}^{m} L_i$$

Definition 4. *The activities unique to L written as $\mathbb{U}(L)$ is a set of activities that are only executed in one or more of the execution paths in L*

$$\mathbb{U}(L) = \left(\bigcup_{i=1}^{m} L_i \right) \backslash C(L)$$

Definition 5. *The common pre-path-execution activities $C_{pre}(L)$ of L is the set of process activities common to all execution paths in L that are executed* before *the activities unique to L.*

$$C_{pre}(L) = \{a\colon a \in C(L) \text{ and } a \rightarrow x \ \forall x \in \mathbb{U}(L)\}$$

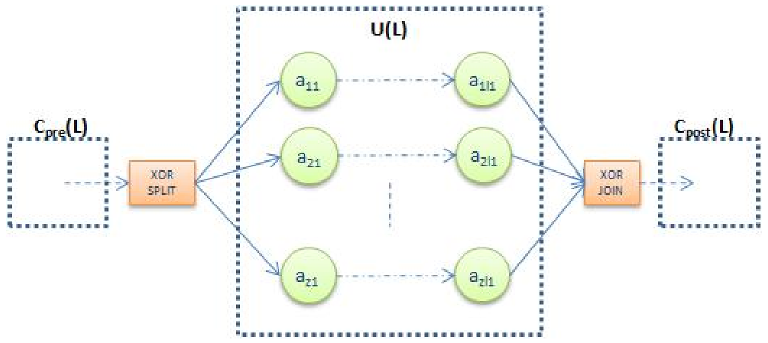

Fig. 2. Flexibility using XOR structural pattern

Definition 6. *The common post-path-execution activities $C_{post}(L)$ of L is the set of process activities common to all execution paths in L that are executed* after *the activities unique to L.*

$$C_{post}(L) = \{a\colon a \in C(L) \ and \ x \rightarrow a \ \forall x \in \mathbb{U}(L)\}$$

The diagram in figure 2 shows the location of the elements defined above in a process with modelled flexibility.

4 Modelled flexibility and approaches to SSO

In an SSO problem model, the objective function is usually a derived from the aggregated end-to-end QoS of the entire process given the QoS values of selected services for each activity in the process. Several authors have listed identical sets of functions for aggregating end-to-end QoS for business processes [2–5]. For each set of selected services associated with a structural pattern, a set of functions are defined to aggregate QoS values for each type of QoS metric (cost, throughput, duration, availability, reliability and reputation). This discussion is primarily concerned with aggregation functions defined for the choice patterns which include OR split/join, XOR split/join and the OR split with m-out-of-k join structural patterns [2]. For this group of patterns, the aggregation of the cost, throughput, duration, availability and reliability metrics were given as the maximum of the aggregate values of all paths associated with the pattern. For any process fragment L conforming to a choice structural pattern with z paths as in figure 1 above, the aggregate value for any of the metric groups will be:

$$Agg(L) = max[agg(L_1), agg(L_2), ..., agg(L_z)]$$

Where L_1, L_2 and L_z refer to the enclosed paths $\{a_{11}, \cdots, a_{1l_1}\}, \{a_{21}, \cdots, a_{2l_2}\}$, and $\{a_{z1}, \cdots, a_{zl_z}\}$ respectively.

If there exist two paths $L_i, L_j \in L$ such that $max(agg(L_i))$ for all combinations of service candidates is greater than $max(agg(L_j))$, then we say L_i

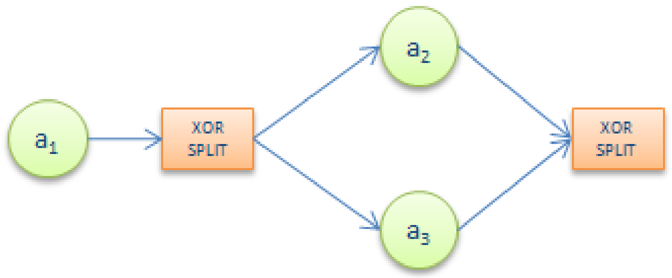

Fig. 3. Example process

dominates L_j in L. The implication of this dominance is that the aggregation function defined above for choice patterns will compute the same aggregate values for L irrespective of the services selected for L_j. As a result, there is a possiblity for the optimisation algorithm to make suboptimal service selections for execution paths which enclose L_j. As an example, consider a simple process shown in figure 3. The process has three activities each with three candidate services as shown by the service directory in table 4.

A naive enumeration of all the 27 possible service allocations showed that because the path containing a_3 dominated the path containing a_2, the aggregate values of the entire process were only determined by the selection of service candidates for activities a_1 and a_3. For instance, when candidate 1 is selected for a_1 and candidate 1 is selected for a_3, the aggregate value for the entire process is $[cost = 22, duration = 36]$ irrespective of whether any of candidates 1 to 3 is selected for a_2. The accumulation of such loss of detail especially in large processes eventually leads to situations where even an exhaustive search algorithm will not be able to take all execution paths into consideration in determining the optimal service selections for activities in the process.

Table 1. Service directory table

Activity	**Candidate**	**Cost**	**Duration**
a_1	1	2	6 secs
	2	3	4 secs
	3	3	8 secs
a_2	1	3	2 secs
	2	5	1 secs
	3	6	3 secs
a_3	1	20	30 secs
	2	10	50 secs
	3	15	40 secs

The significance of this drawback lies in the fact that although a process model can be designed to accommodate multiple execution paths, such provision is usually for the purpose of flexibility; and as such, each instance of the process will only follow one execution path during execution and the most frequent or most significant of which may not be the dominant path.

A second approach that attempts to address the above issue was proposed in [3]. In this approach each execution path $P_1, \cdots, P_z$ are optimised separately. Note that execution paths do not contain choice patterns. As a result, the objective function derived from the aggregation functions defined in [2] are a precise statement of the utility of a given solution which can be used by the exhaustive search algorithm to determine the exact solution to the sub-problem. The result is a set of solutions, one for each execution path in the process.

Let $Sol(P_i) = \{(a_j, s_j)\}$ represent the set of solutions for path P_i in a process where service $s_j \in S_j$ is selected for activity $a_j \in A$. We observe that $Sol(P_i)$ will contain a selection pair (a, s) for each activity $a \in C_{pre}(P)$. In the use of this approach, a challenge arises when for a least one activity a in $C_{pre}(P)$, there exists at least two paths P_j and P_k such that $(a, s_j) \in Sol(P_j)$ and $(a, s_k) \in Sol(P_k)$ and $s_j \neq s_k$. Let $ConflictSol(a)$ be the set of all pairs (a, s) in $\cup_{i=1}^{z} Sol(P_i) \forall a \in C_{pre}(P)$. A decision has to be made as to which pair $(a, s) \in ConflictSol(a)$ should be used to select a service for each activity a in $C_{pre}(P)$. In [3], the authors propose the use of a hot path $P_{hot} \in \mathbb{P}(P)$ which is determined based on the path execution frequency in the previous execution history of the process. In the absence of a history, a user may decide the hot path. In this case, the sevice selected for activity a in $Sol(P_{hot})$ is adopted for the process. The problems associated with this approach has been highlighted in [4]. The main issue is that the use of the hot path as the single decision variable for determining the selection of service implementations for activities in $C_{pre}(P)$ ignores the case for the end-to-end optimality of the other execution paths the result of which could potentially reverse the benefits that had been gained from optimising the other execution paths separately.

In [4], the authors proposed two optimisation approaches. The first approach treats each execution path as an autonomous unit for the purpose of optimisation, yet simultaneously considers all execution paths in the objective function based on the probability of each path. The use of probability here derives from earlier work in [12]. Each execution path is assigned a probability which is computed from the multiplication of the probabilities of all conditional transitions from which the path was derived. Thus the objective function proposed is as follows:

$$max \sum_{i=1}^{z} Prob(P_i).F_i$$

where $Prob(P_i)$ is the probability of path P_i and F_i is the aggregation of the solution vectors for all the activities in the path. This approach therefore considers all the execution paths together in determining the optimal set of selections based on the probability of each execution path being executed which can either be derived from simulation, previous execution history or explicitly asserted by the user.

The second approach presented in [4], utilises an objective function which gives priority to the execution path with the highest probability:

$$max F_{max}$$

where F_{max} is given by the aggregation of the path with the highest probability. This effectively reduces to the hot path solution described above except that the approach goes further to find feasible solutions for other paths in the process which may result in reducing the quality of the selections for the path with the highest probability when necessary to support the feasibility of the other paths.

5 Discussion and Conclusion

From the previous sections, it is evident that none of the approaches reviewed guaranteed an exact solution for service selection optimisation such that the selections for each of the execution paths in the process is optimal. However, they each provide approximations to address the presence of modelled flexibility in business processes. A question then arises as to the existence of an exact optimal solution for flexible processes. A process without modelled flexibility implies that all the activities in the process will be executed by all instances of the process. This certainty makes it possible for the aggregation methods to provide precise statements about the utility of any given solution and as such an algorithm can be able to effectively compare the utility of different solutions to the problem.

However, with the introduction of modelled flexibility by choice structural patterns, different instances of the process may execute different sets of activities. Although the different paths can be enumerated at design time, the activities that are common between these paths are a source of difficulty in guaranteeing that an algorithm could arrive at a solution that simultaneously optimise all the execution paths at the same time.

For any process P, any instance must execute the activities in $C_{pre}(P)$ before the activities in $P_i \backslash C_{pre}(P)$. Given that the exact optimal solution for each execution path in P has been determined during the design phase, one might suggest that these selections be applied for each execution path in the process. However, it is possible for the optimal solutions for the execution paths to select different services for an activity in $C_{pre}(P)$ which will cause assignment conflicts during the design phase. Assuming that a dynamic binding strategy is used at runtime, the selection of services for the activities in $C_{post}(P)$ can be based on the execution path chosen by a process instance but this is not possible for the activities in $C_{pre}(P)$ because the complete execution path of an instance often not known on until all the decision nodes are evaluated. As a result, it is not possible to use this knowledge of optimal service selections for the individual execution paths to assign service implementations for the activities before the decision nodes are evaluated.

In conclusion, we note that 'exact' solutions for service selection optimisation within the context of modelled flexibility may translate to approximate solutions for some execution paths in a process model. The approximations currently realised by the approaches discussed above assumes that the execution path with the highest probability of execution at runtime (usually determined cummulatively from previous execution history) should be the only criteria for addressing the potential conflicits associated with modelled flexibility. In many cases, this assumption may be considered as unrealistic. For example in a process with two paths P_1 and P_2 such that path P_1 caters for premium customers while path P_2 caters for basic customers, it would not be appropriate for the process to be optimised with regard to QoS metrics with the selection scaled preferentially for path P_2 at the expense of path P_1 just because P_2 occurs more frequently than P_1.

Future work will investigate an extension to the approach presented in [4]. The approach will allow for user-controlled biasing of the service selection optimisation based on a set of weighted *meta-metrics* rather than only execution probability. Such meta-metrics may include probability/frequency of execution, business importance of the execution path or any other process metrics. In addition, future work will also address service selection for activities enclosed within exception handling blocks as well investigate the differences between design phase optimisation and execution phase optimisation of service processes.

Acknowledgement The authors wish to thank the anonymous reviewers for their useful feedback. The work of the first author is supported by the Petroleum Technology Trust Fund, Nigeria under grant number PHD08019.

References

1. OASIS. Ws-bpel specification oasis standard. Technical report, OASIS, 2007.
2. Michael C Jaeger. *Optimising Quality-of-Service for the Composition of Electronic Services.* PhD thesis, Technischen Universitat Berlin, January 2007.
3. Liangzhao Zeng, Boualem Benatallah, Anne H. H. Ngu, Marlon Dumas, Jayant Kalagnanam, and Henry Chang. Qos-aware middleware for web services composition. *IEEE Transactions on Software Engineering*, 30(5):311–327, 2004.
4. Tao Yu, Yue Zhang, and Kwei-Jay Lin. Efficient algorithms for web services selection with end-to-end qos constraints. *ACM Trans. Web*, 1(1):6, 2007.
5. Wentao Zhang, Yan Yang, Shengqun Tang, and Lina Fang. Qos-driven service selection optimization model and algorithms for composite web services. In *Computer Software and Applications Conference, 2007. COMPSAC 2007 - Vol. 2. 31st Annual International*, pages 425–431, 2007.
6. Yan Gao, Jun Na, Bin Zhang, Lei Yang, and Qiang Gong. Optimal web services selection using dynamic programming. *iscc*, 0:365–370, 2006.
7. B. Kiepuszewski. *Expressiveness and Suitability of Languages for Control Flow Modelling in Workflows.* PhD thesis, Queensland University of Technology, 2002.
8. Wil M. P. van der Aalst. The application of petri nets to workflow management. *The Journal of Circuits, Systems and Computers*, 8(1):21–66, 1998.

9. Wil M. P. van der Aalst, Alistair P. Barros, Arthur H. M. ter Hofstede, and Bartek Kiepuszewski. Advanced workflow patterns. In *CoopIS*, pages 18–29, 2000.
10. Wasim Sadiq, Shazia Sadiq, and Maria E. Orlowska. Pockets of flexibility in workflow specification, 2001.
11. N.A. Mulyar, M.H. Schonenberg, R.S. Mans, N.C. Russell, and W.M.P. van der Aalst. Towards a taxonomy of process flexibility (extended version). Technical report, 2007.
12. Jorge Cardoso. *Quality of Service and Semantic Composition of Workflows*. PhD thesis, University of Georgia, Athens, Georgia (USA), 2002.