Int. J. Nonlinear Anal. Appl. 12 (2021) No. 1, 87-109
ISSN: 2008-6822 (electronic)
http://dx.doi.org/10.22075/ijnaa.2021.4664

Automatic QoS-aware Web Services Composition based on Set-Cover Problem

Morteza Khani Dehnoi, Saeed Araban*

Department of Computer Engineering, Faculty of Engineering, Ferdowsi University of Mashhad, Mashhad, Iran.

(Communicated by Madjid Eshaghi Gordji)

Abstract

By definition, web-services composition works on developing merely optimum coordination among a number of available web-services to provide a new composed web-service intended to satisfy some users requirements for which a single web service is not (good) enough. In this article, the formulation of the automatic web-services composition is proposed as several set-cover problems and an approximation algorithm has been exploited to solve them. In proposed method, the web-service composition has been carried out within two main phases, the top-down expansion of the composition tree, and the production of composed service by bottom-up traversal of composition tree. In the first phase, the production of a composition tree (similar to the production of tree in problem-solving by searching) is proposed by starting from the output or post-conditions of the requested service towards its input or pre-conditions. Each node or state of the tree is a set of inputs and/or outputs or conditions, and services as tree edges illustrate the transition from one node to another. In the second phase, finding the path from the leaves of the produced composition tree to the root is considered equal to reaching the output of requested service, and this path specifies the involved services and the composition plan. The requested service input set determines the available leaves of the composition tree. To achieve each non-leaf node of the tree, a set-cover problem is produced and solved using a greedy approximation algorithm. If the production and solving of the set-cover problems continues hierarchically until it reaches the root node, the composition plan and cost of the required composition service will be specified. The main focus of this research is the joint sequential and parallel composition with the aim of producing near-optimal and QoS-aware composed services.

Keywords: Web Services, Composed Services, Set-cover Problem, Approximation Algorithm.
2010 MSC: 76T20

*Corresponding Author: Saeed Araban

Received: February 2020 *Revised:* October 2020

1. Introduction

One of the greatest potential abilities of service technology that should be considered is the interoperability between services within or outside the boundaries of the service owner organization. If a service calls other services to perform its functional tasks, the caller service is called the composed service, and the called services are called the basic services. To increase functionality, a composed service is not dependent on the basic services. Since a composed service may invoke other services from heterogeneous systems to perform its tasks, it must be able to deal with various problems such as inconsistency in data pattern, incompatibility in transactions, sequencing constraints in invoking the operations and the security of distributed systems.

2. Literature review

Web-services composition is defined as developing merely optimum coordination among a number of available web-services to provide a new composed web-service intended to satisfy some users requirements for which a single web service is not (good) enough. An important goal of web-service composition is to achieve maximum flexibility in adapting dynamically to an environment in which the available services (whether simple or composed services) are constantly changing in terms of availability, load balancing or application.

2.1. Composed Service Plan

Generally, the composed service plan is produced by two types of methods[1], [2]:

1. Workflow techniques
2. AI (Artificial Intelligence) planning techniques

The workflow techniques consider a composed service as a set of irresolvable (atomic) services along with a workflow that includes the data flow and the flow of control of the execution of the involved services. The workflow techniques provide some automatic methods for binding abstract roles to the actual and objective sources or services.
On the other hand, the production of a composed service plan can be based on the AI planning. In these methods, it is assumed that each service can be defined and identified by preconditions and its executional effects on the environment. Therefore, a plan or process can be generated automatically using AI planning methods, without having any predefined knowledge of workflow. The production of the composed service plan in the method proposed in this article is based on AI planning.

2.2. Composed Service Execution

Composed web-services are usually executed by either of two strategies: Orchestration or Choreography.
In Orchestration, a coordinator, that itself can be a service, controls and coordinates the services involved in the composition. In this method, the services involved in the composition should not be aware that they are parts of a higher-level business process. The central coordinator should be aware of the overall purpose of the process, definitions of the roles, and the order of employing of the web-services involved in the composition.
By contrast, Choreography acts without a central process. In the absence of coordinator, any service involved in the composition operation knows when it should collaborate and with which service it should interact. The collaboration is done based on message reciprocation. Every service involved in the composition should be aware of the general business process (the task of the composed service),

the jobs to be done, the messages to be passed, and the schedule for message passing processes [1], [3]. The execution of the composed service in the proposed method in this article is based on Orchestration.

2.3. Automatic Web-Services Composition

Automatic web-services composition is when generation of the composition chain is done at runtime (on the fly) without any manual interference. In this methods, a service request including the characteristics of the requested service would cause the generation of a service consisting of several components which has not been existed before the request. In other words, in this method, any query instead of being just a contact solicitation for an interface would be considered as a request for a composed system[2], [4]. The method proposed in this article provides an automatic composition of web-services.

2.4. Static or Dynamic Web-Services Composition

From one viewpoint, the methods of web-services composition can be classified into two general divisions, static and dynamic. In static web-services composition, the information of available services which can be used in the composition should be prepared and collected before starting the composition operation. Considering the great number of services and the fact that they get frequent updates by the providers in terms of updating existing services as well as adding new services to the community, collecting information about available services for the composition operation is the main issue for these methods. Therefore, the static methods for the services composition always have an internal inflexibility. Also, due to inability regarding the load balance in the partner services in the composition, scalability has some restrictions using these methods.
In dynamic web-services composition, the composition plan is formed based on the present state of the web world; as a result, it is not necessary to collect comprehensively the information of all available services before the composition operation. The method presented in this article addresses the dynamic web-services composition.

2.5. Quality-Aware Web-Services Composition

In addition to the functional specification, web-services are also known by their non-functional attributes usually referred as Quality of Services (QoS) attributes. The importance of QoS attributes shows itself where among several services which are equal in functionality, the chosen one is the most satisfactory regarding the QoS preferences of the customer. If QoS parameters are the criteria for web-service selection in composition process, it will be QoS-aware web-services composition. The method proposed in this article addresses the QoS-aware web services discovery and composition.

2.6. Sequential and/or Parallel Web-Services Composition

If the functional requirements of the requested composed service can be achieved by consecutive application of two or several services, it will be chain or sequential composition of web-services (Figure 1).
On the other hand, if a part of the functional requirements of the requested service is met by employing one service and another part of its requirements is met by employing one or several other services, the composition is parallel (Figure 2).
The plan of a composed web-service can be described at several sequential and parallel levels (Figure 3).
Most automatic web-services composition methods have addressed the sequential composition of web-services [5]–[7], but there have also been efforts for the parallel web-services composition [8], [9]. In

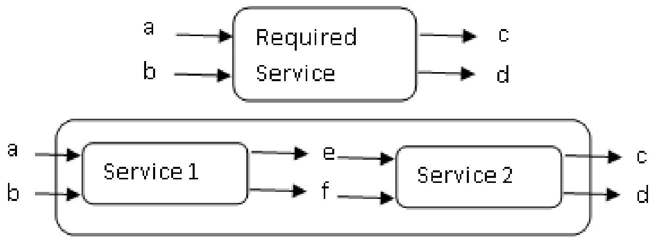


Figure 1: Sequential web-services composition

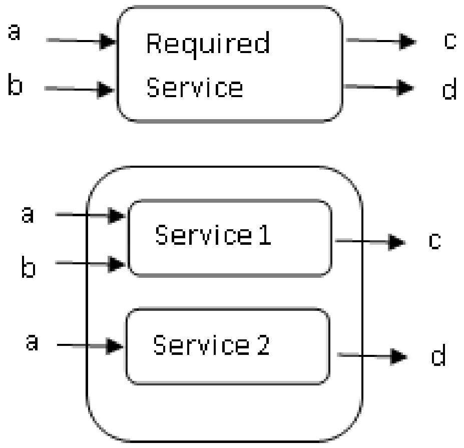


Figure 2: Parallel web-services composition

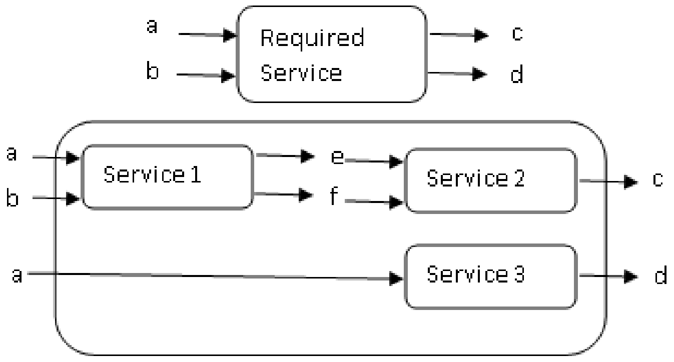


Figure 3: A sequential layer and a parallel layer of web-Services composition

the proposed solution in this article, the possibility of automatic parallel and sequential composition of services has simultaneously been considered.

2.7. Categorization of Web-Services Composition Methods

Table 1 presents a categorization of service composition methods and their characteristics.

3. Proposed Solution

In this section, a solution is proposed for automatic QoS-aware web-services composition by modeling as a minimum set cover problem and using the approximation algorithm. In the first step, the set cover problem and the approximation algorithm for solving it as the main tool in the proposed

Table 1: Web-Services composition methods at a glance

Service Composition plan	Method description		References	Execution of Composed Services	Automation	Supporting Parallel Composition	Dynamic Planning	Supporting composed Service recovery	QoS-Awareness	Considering user preferences	Distribution	Scalability
Workflow based	There is a workflow as part of the request. The main problem is the optimal selection of services for roles in the workflow. The problem is modeled as a multi-objective optimization problem (MOOP).	Using heuristic and meta-heuristic methods for optimization	[10], [11], [20]–[27], [12]–[19][28], [29][30]	Orch	×	✓	Semi-dynamic	Fully (service substitution)	✓	✓	×	✓
		Using Linear or Integer Programming for optimization	[31]–[34]	Orch	×	✓	Semi-dynamic	Partially (service substitution)	✓	✓	×	✓
		Modelling and solving as a Constraint Satisfaction Problem (CSP)	[35]–[37]	Orch	×	✓	Semi-dynamic	Partially (service substitution)	✓	✓	×	✓
		Clustering or indexing available services based on functional or non-functional features by the service brokers.	[38]–[47]	Orch	×	✓	Semi-dynamic	Fully (service substitution)	✓	✓	×	✓
	Workflow is generated.	Design of composition is performed in two phases: 1) workflow generation, 2) optimal selection of services for roles in the workflow. Due to	[48]–[52]	Orch	✓	✓	Semi-dynamic	Partially (service substitution – partial replanning)	✓	✓	×	×

		complexity of workflow generation, these methods are domain specific.										
AI Planning based	Available services are modeled as graphs (Automata, Finite State Transducer (FST) or Colored Petri Net (CPN)). These methods require high preprocessing and model rebuilding. Major problems in this category are: 1) Incompatibilities of the highly dynamic environment of web-services, 2) Imbalance of load on the services involved in the composition.		[53], [54], [63]–[67], [55]–[62][68], [69]	Orch	✓	×	Static	×	✓	×	×	×
	Available services are modeled as rule-based expert systems. These methods require high preprocessing and rebuilding of inference engine.		[8], [9]	Orch	✓	×	Static	×	×	×	×	✓
	The problem is modeled as a tree, which is solved by searching. No preprocessing is required. Using agent technology and users feedbacks is possible. The highest degree of flexibility and loose coupling is available.		[5], [6], [78]–[83], [70]–[77]	Orch	✓	×	Dyna mic	Partially (partial replanning)	×	×	✓	✓
	Proposed method described in section 3.			Orch	✓	✓	Dyna mic	Partially (service substitution – partial replanning	✓	✓	✓	×

method has been reviewed. In the next step a proper representation of the problem and the solution production process have been provided. Finally, a small example of how to implement the proposed solution has been provided.

3.1. Set Cover Problem

The set cover problem is a classic problem in the computer sciences and complexity theory, and is among the 21 famous Karp problems, whose NP-Completeness has been proven in 1972 [84]. Consider the universal set U and the set S including m of the other sets, in a way that the union of m sets inside S is equal to U. That is, the set S covers the set U. The set cover problem is to identify set C as the smallest subset of S whose inside sets union is equal to U. That is, set C also covers set U. For example, if U and S considered as $U = \{1, 2, 3, 4, 5\}$ and $S = \{\{1, 2, 3\}, \{2, 4\}, \{3, 4\}, \{4, 5\}\}$, by solving the set cover problem the result will be $C = \{\{1, 2, 3\}, \{4, 5\}\}$. If the selection of each set within S has a number as a cost, the problem will be of the weighted set cover type.
There are good approximation algorithms such as Johnson's Greedy Algorithm [85] with logarithmic approximation coefficient and polynomial time order for the set cover problem. Heuristic methods such as Genetic Algorithm in [86] and Ant Colony Algorithm in [87] are also used to improve the approximation. According to the surveyed background, in web-service composition, the set cover problem has not been used and has only been used in some researches such as [88] and [89] to prove that the composition problem is NP-Complete.
Johnson's Greedy Algorithm [85] to solve the set cover problem selects a set of S at each step that contains the highest utility or in other words the least cost per not selected member. For this purpose, for each set, it calculates the result of cost divided by the number of members and selects the set with the least dividing result. The selected set is added to C and its members will be eliminated from U and unselected sets. The algorithm is complete when U is empty. It has been proved in [85] that the approximation coefficient of this algorithm is H(n) in which n is the number of members of the reference set U. This means that Johnson's algorithm selects those sets for U cover whose sum of costs may be at most H(n) times greater than the minimum cover cost. It should be mentioned that H(n) is the nth harmonic number and is obtained from the following equation.

$$H(n) = \sum_{k=1}^{n} \frac{1}{k} \leq \ln n + 1 \tag{3.1}$$

3.2. Representing the Problem of Services Composition

The suggested method proposes the composition of web services into two main phases:

1. Top-down expansion of the composition tree
2. Bottom-up traversal of composition tree to production of composed service plan

The general process of the proposed method is shown in the following activity diagram.

3.3. Top-Down Expansion of Composition Tree

Employing each service is considered to convert a pre-execution status (including inputs and preconditions) to a post-execution status (including outputs and post-conditions). In the issue of web-services composition, the output of each service can provide the input of one or several other services. In this way, the space of the web-services composition problem can be used as a tree model and by problem-solving with a search. In this way, each node or the status of a tree is a set of inputs, outputs, and conditions in which services as tree edges provide the possibility of transition from one node to

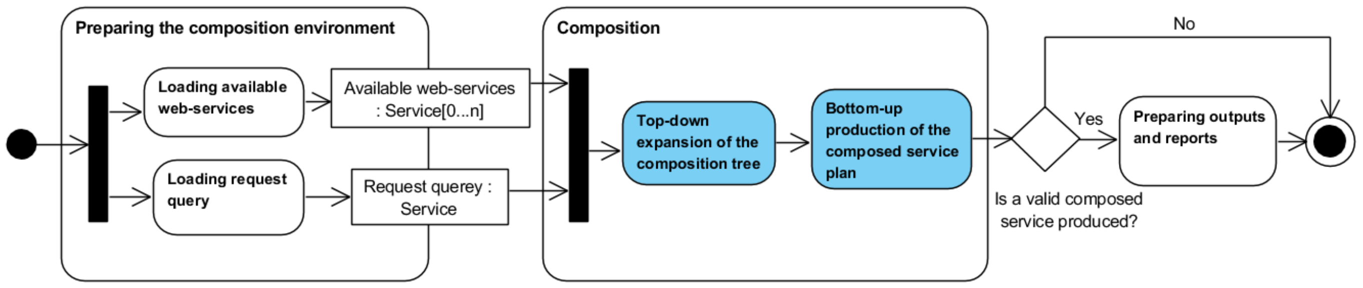

Preparing the composition environment
Loading available web-services
Available web-services : Service[0...n]
Loading request query
Request querey : Service
Composition
Top-down expansion of the composition tree
Bottom-up production of the composed service plan
No
Yes
Is a valid composed service produced?
Preparing outputs and reports

Figure 4: General process of the proposed method

another node. The described tree is called the composition tree. In this research, a composition tree is produced from the output or the post-conditions of the requested service toward the input or its preconditions:
State Space: The state space includes pre-execution and post-execution status of all available services
Tree Root (initial state): The requested service outputs are the content of the root node.
Successor Function: The Successor function takes a node as input and finds all web services that are available at the same time and their output contains a part of the node content that is expanding. The successor function restores the input set of each one of the services it finds as produced nodes (children of current node).
Goal Test (End-Node Production Test in Each Branch): The production of trees in each branch continues up to where we reach a node that its content is subset of input or preconditions of the requested service, or the depth or weight limitation of the path is violated. The depth or weight limitation of the path can be determined depending on the composition time limitation or the services cost limitation by a request.
Composed Service Cost: Composed service cost, is a composite factor of QoS that can be calculated for each valid sub-tree of the composition tree. Mapping QoS attributes in cost is performed by a cost function. The cost function receives the QoS index vector of an employed service and the QoS importance coefficients vector from the request as input, and maps it as a cost by computing similarity of two input vectors. Cost function applies to all employed services of composed service plan and sum of results regards as composed service cost.
The process of top-down expansion of composition tree is shown in the following activity diagram.

3.4. Bottom-Up Production of Composed Service Plan

Reaching from the valid leaves of the composition tree to the root is equal to reaching the outputs of the requested service, and the path to reach the root specifies the composition plan (services involved in the composition). Leaves of a tree whose content is a subset of the requested service input are available. To reach every non-leaf node of a tree, a set-cover problem should be produced and solved. Components of the set-cover problem are defined as follows:
U-set: It will contain the content of the current node.
S-set: For each child of the current node, a set will be placed in S that the members of this set will be the intersection of set U with the current edge service output.
Modeling the Weight of Sets: The weight of each set will be equal to the total cost of reaching the child node, and the cost of corresponding service to transition from the child to the current node. It is remembered that the cost of each service is calculated by calculating the similarity of the QoS index vector of this service and the QoS importance coefficients vector from the request.

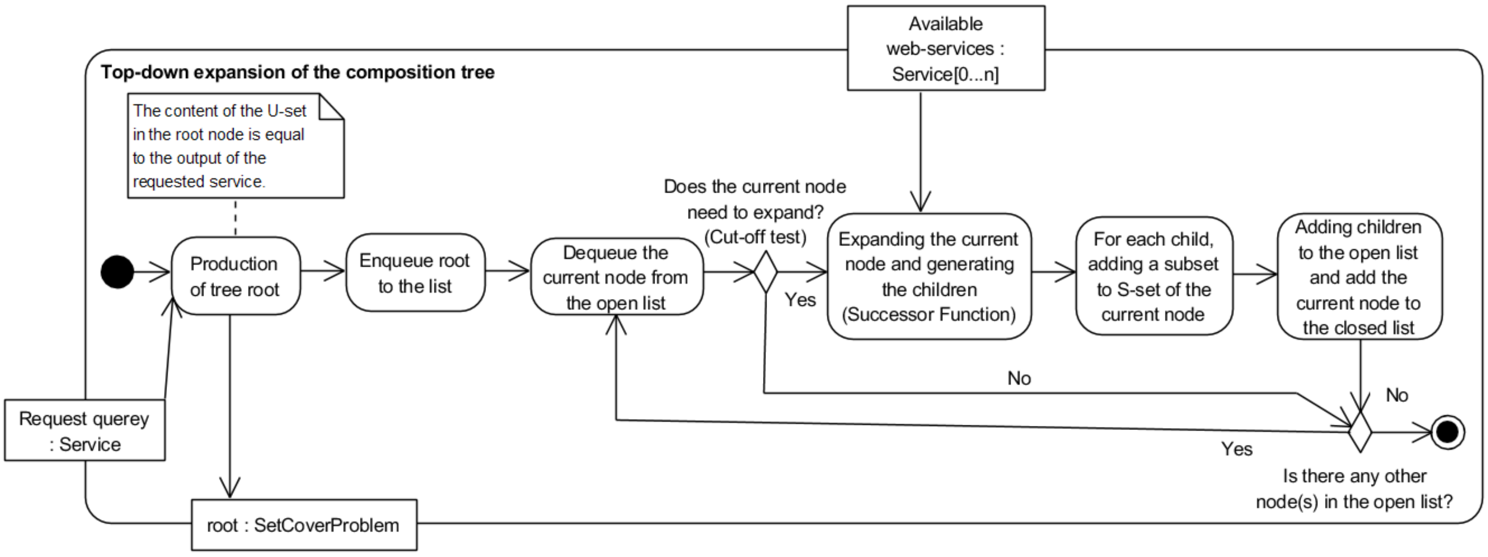

Top-down expansion of the composition tree
Available web-services : Service[0...n]
The content of the U-set in the root node is equal to the output of the requested service.
Production of tree root
Enqueue root to the list
Dequeue the current node from the open list
Does the current node need to expand? (Cut-off test)
Yes
No
Expanding the current node and generating the children (Successor Function)
For each child, adding a subset to S-set of the current node
Adding children to the open list and add the current node to the closed list
No
Yes
Is there any other node(s) in the open list?
Request querey : Service
root : SetCoverProblem

Figure 5: Process of top-down expansion of composition tree

Thus, by solving the above set-cover problem, the cost and plan of reaching the current node are calculated. If the production and problem solving of the set-cover continue hierarchically to reach the root node, the plan and cost of the requested composed service will be produced. In the present research, the Johnson approximation algorithm [85] is used to solve the set-cover problems.
Two points about the proposed method should be mentioned: First, the composition tree produced in the proposed method can also be applied in composed service recovery (partial re-planning and service substitution). Second, there is no need to aggregate the information of available services before the composition operations, and the composition operations can be performed dynamically. The process of bottom-up production of composed service plan is shown in the following activity diagram.

3.5. Solving a Sample of Web-Services Composition Problem

To clarify the method, a very simple and demonstrable sample of the problem is described and solved below. In Figure 7, the problem space including available services and the request of composition service is described.
The result of the first phase of the method, namely top-down expansion of composition Tree is presented in Figure 8.
The execution of the second phase of the method, namely bottom-up traversal of composition tree for the production of composed service plan is presented in Figure 9. The process of hierarchically producing and executing the set cover problem and the calculated value as the cost of reaching each node (red-colored numbers) are specified in the figure.
In Figure 10 the final plan of the requested composed service is illustrated after eliminating unused branches.
In Figure 11, the BPMN diagram of composed service produced for the sample problem is drawn.

4. Implementation and Test

The proposed method of this article is implemented in the Service-Oriented Enterprise Architecture Laboratory (SOEA LAB) of the Ferdowsi University of Mashhad [90], and the QoS-WSC data set [91] has been used to test it. This data set has 18 service repositories for testing QoS-aware web-services

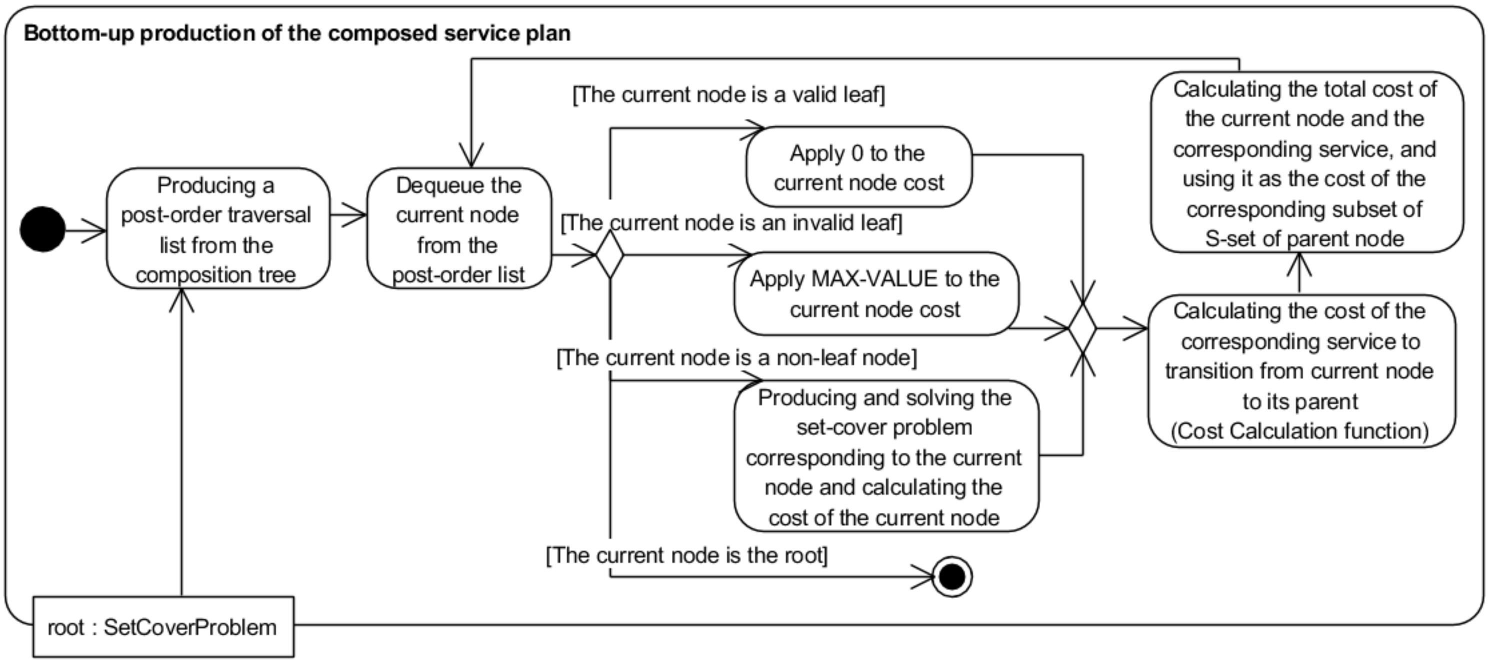

Bottom-up production of the composed service plan
Producing a post-order traversal list from the composition tree
Dequeue the current node from the post-order list
[The current node is a valid leaf]
Apply 0 to the current node cost
[The current node is an invalid leaf]
Apply MAX-VALUE to the current node cost
[The current node is a non-leaf node]
Producing and solving the set-cover problem corresponding to the current node and calculating the cost of the current node
[The current node is the root]
Calculating the total cost of the current node and the corresponding service, and using it as the cost of the corresponding subset of S-set of parent node
Calculating the cost of the corresponding service to transition from current node to its parent (Cost Calculation function)
root : SetCoverProblem

Figure 6: process of bottom-up production of composed service plan

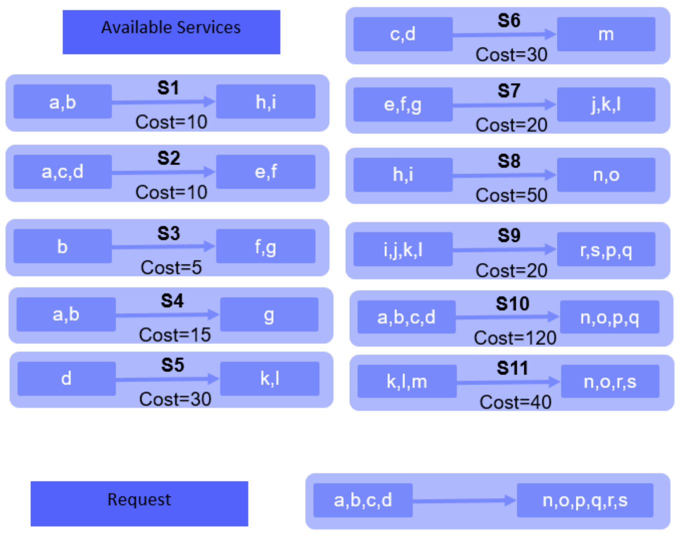


Figure 7: Sample Problem Space

composition. Its smallest data repository contains 2,156 and its largest data repository contains 8,356 WSDL files to describe the space of available services. Each WSDL file contains the functional description of the service (in the form of two request and response messages), a QoS index vector (in the form of QoS tag), as well as the description of the service interface (in the form of a port message). This test data set contains 198 service composition problems in the form of 18 query files. For each query file, one solution file contains all possible solutions and one best solution file contains the best solution in terms of QoS for the issues raised in the related query file. The information of services and solutions is maintained by the BPEL (Business Process Execution Language) standard. The results obtained from the experiment in all composition queries correspond to the results declared in the data set. Examples of the WSDL file, the request query, and the corresponding composition

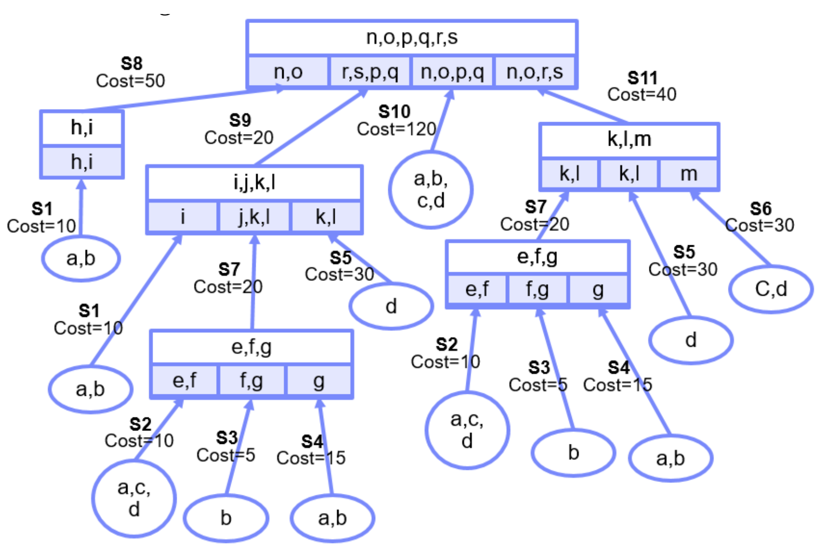


Figure 8: Composition Tree of Sample Problem

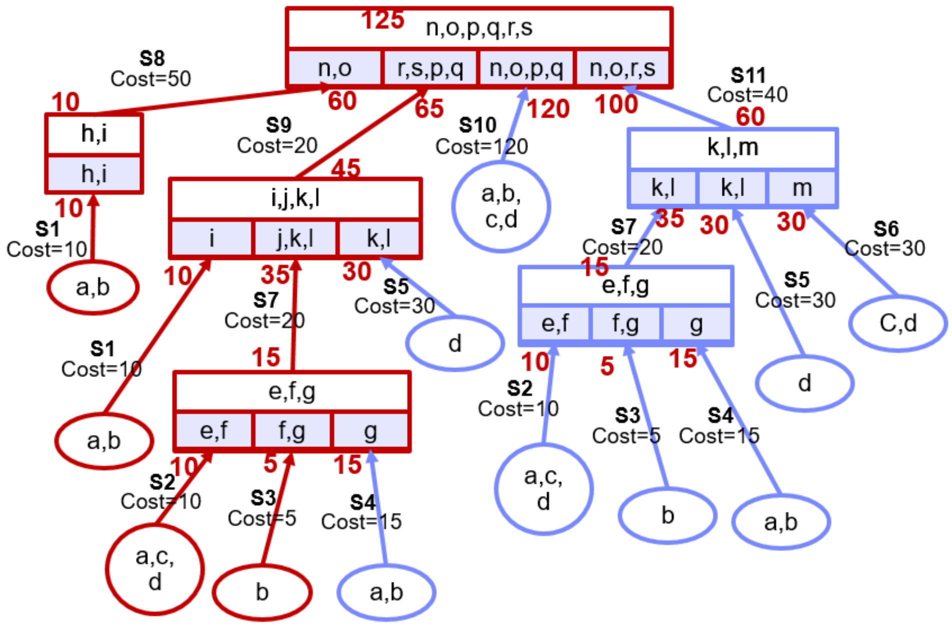


Figure 9: Hierarchical Production and solving of Set-Cover Problems in Sample Problem

tree, solution (composed service plan), and BPMN diagram produced by the proposed method are presented in Appendix.

5. Evaluation of the Proposed Solution

Evaluation of the proposed method for the three criteria of completeness, performance and approximation factor, and from the two perspectives of theory evaluation and experiment-based evaluation are presented in the following.

Completeness: Completeness of the method means that if there is a valid solution, the method guarantees to find it. In the proposed method, since the composition tree is fully produced, if there is a solution, it will be found. If the production of the composition tree is limited to a specified depth

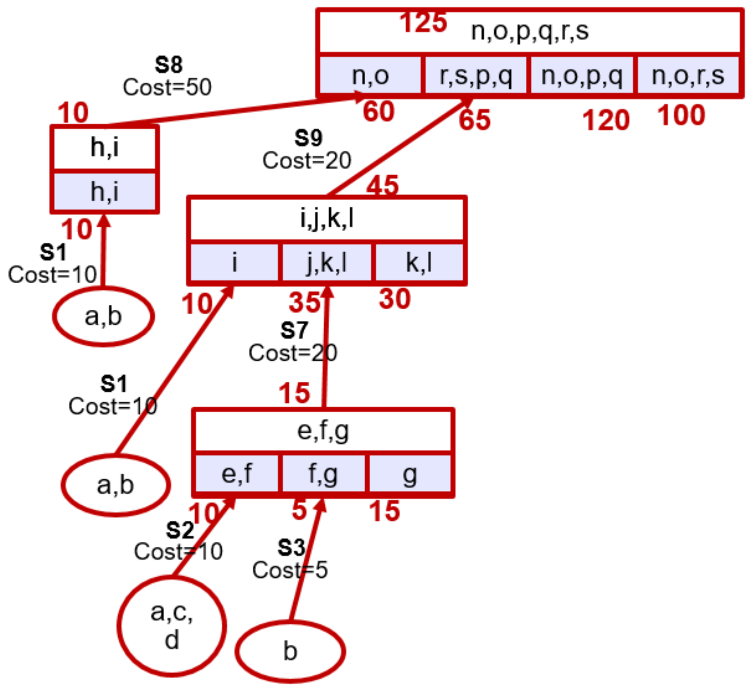


Figure 10: Composition Service Plan of Sample Problem

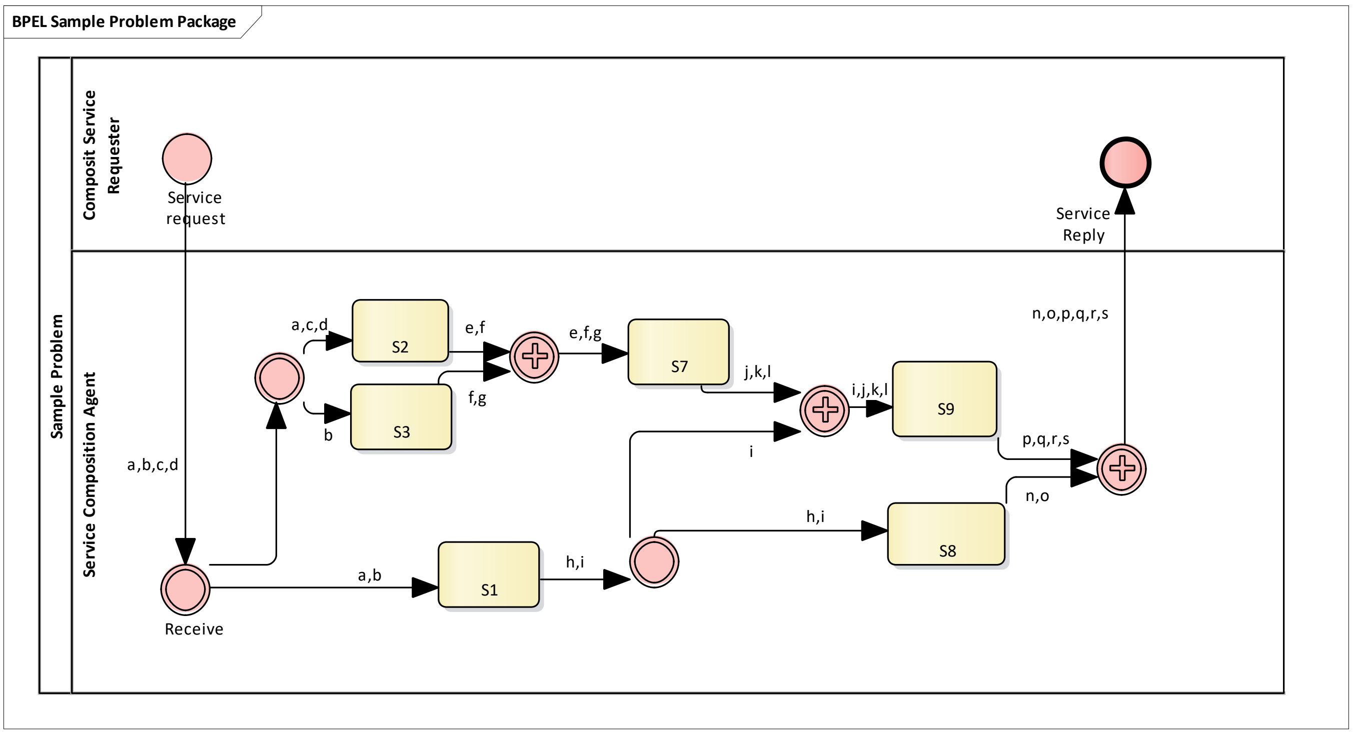


Figure 11: BPMN Diagram of the Produced Composition Service

(or cost), if there is a solution down to that depth (or cost), the proposed method guarantees to find it. In the experiments performed on the QoS-WSC data set, a valid solution has also been found for every composition query.

Performance: The time complexity of the first phase of the proposed method (top-down expansion

of composition tree) is proportional to the number of nodes produced in the composition tree. If the maximum branching factor of tree is b and the maximum depth of the composition tree is d and the number of available services is S, the execution time of the proposed method will be limited to O(Sb^d). In the second phase of the proposed method (bottom-up production of composed service), for each non-leaf node of the composition tree, a set-cover problem has been produced and solved. Since the Johnson algorithm (with polynomial time order) is used to solve each set-cover problem, the time order of the second phase will not be more than the time order of the first phase, so the time order of the presented method will be the same as O(Sb^d). Also in the experiments performed on the QoS-WSC data set, the number of available services and the maximum depth of the valid responses for each query are specified. The maximum branching factor is also the same for all queries. The execution time of the proposed method for each query has also been measured.
The data extracted from the tests show that three independent variables have effect on the proposed method execution time.

1. Number of available services
2. Number of functional requirements
3. Depth of composition tree

Figures 12, 13 and 14 show the effect of these variables on the composition time. The consistency of the results of the evaluation based on the experiment and theoretical evaluation can be observed in these diagrams.

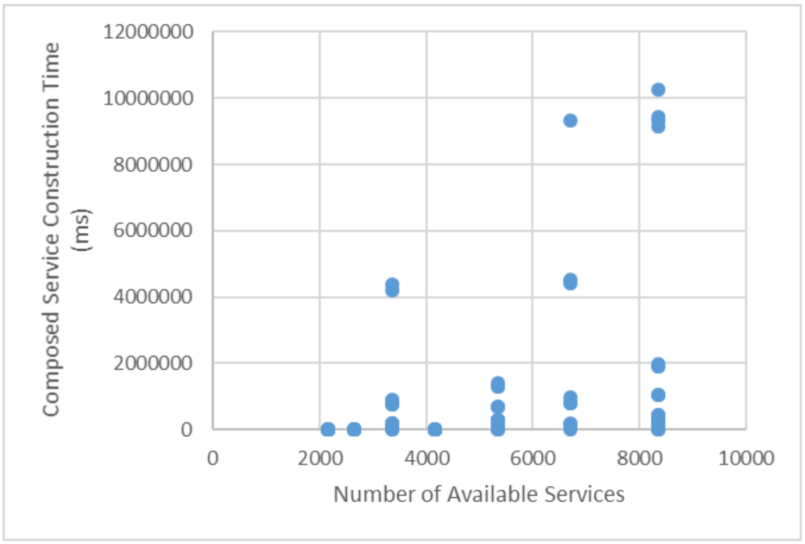


Figure 12: Effect of number of available services on composition time

Figure 2 shows the effect of the number of available services and average of inputs/outputs on the loading time of the available services.
Approximation Factor: The approximation factor is the proximity criterion of the cost of the composed service produced by proposed method to the cost of the optimal composition. In the proposed method the only approximate component is Johnson's algorithm for solving the set cover problems, so the approximation factor of proposed method is proportional to Johnson's algorithm. In the experiments performed on the QoS-WSC data set, for every composition query, the cost of composed service produced by proposed method is near enough to the cost of the optimal composition (i.e. no more than the multiplication of best solution cost by the approximation factor).

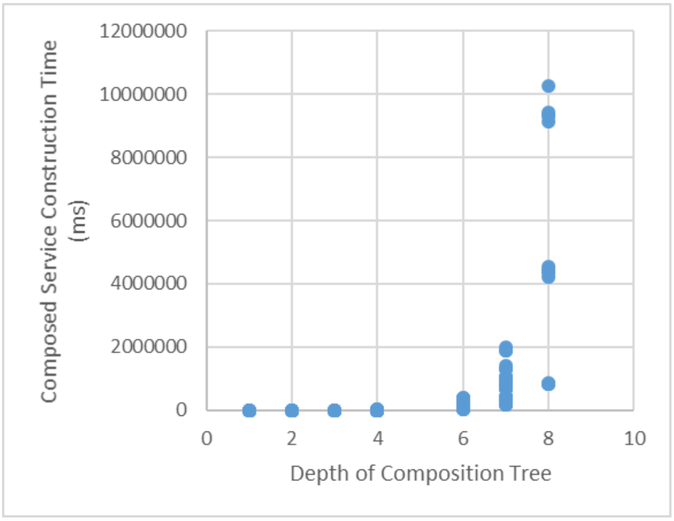


Figure 13: Effect of depth of composition tree on composition time

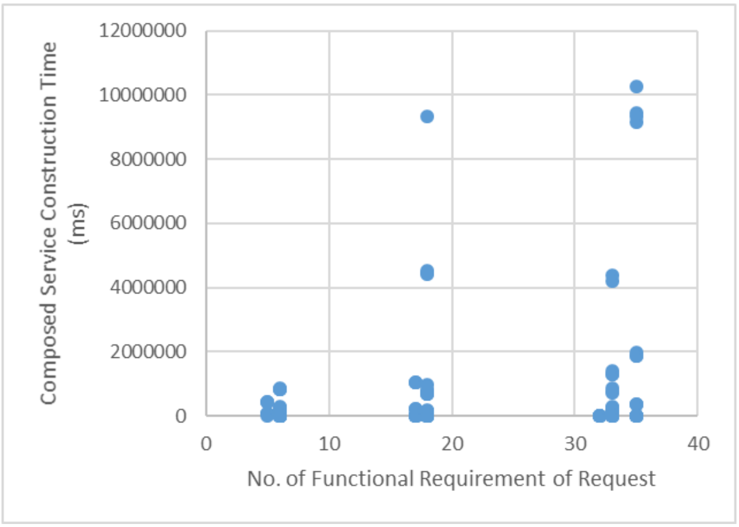


Figure 14: Effect of number of functional requirements on composition time

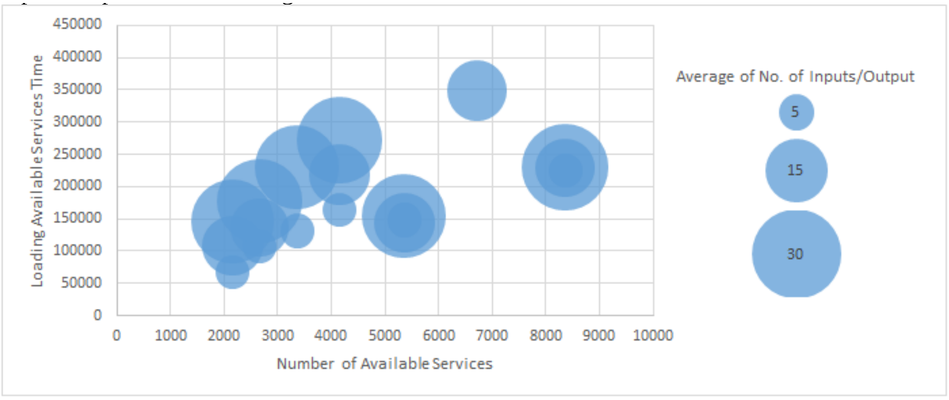


Figure 15: Effect of number of available services and average of inputs/outputs on loading time

6. Conclusion

In this paper, a new method for automatic web-services composition is proposed. In this regard, the problem has been formulated as a several set-cover problems and an approximation algorithm has been applied to solve them. In proposed method, the web-service composition has been performed in two main phases, the top-down expansion of the composition tree, and the production of composed service plan by bottom-up traversal of composition tree. The main focus of this research has been the joint sequential and parallel composition with the aim of producing near-optimal and QoS-aware composed services.

References

[1] "Web Service Choreography Interface (WSCI) 1.0." [Online]. Available: https://www.w3.org/TR/wsci/. [Accessed: 13-Apr-2020].
[2] G. Baryannis and D. Plexousakis, "Automated Web Service Composition?: State of the Art and Research Challenges," ICS-FORTH, Tech. Rep, no. October, 2010.
[3] L. A. F. Leite, G. Ansaldi Oliva, G. M. Nogueira, M. A. Gerosa, F. Kon, and D. S. Milojicic, "A systematic literature review of service choreography adaptation," Serv. Oriented Comput. Appl., vol. 7, no. 3, pp. 199–216, 2013.
[4] J. Rao and X. Su, "A Survey of Automated Web Service Composition Methods," in Proceedings of the First international conference on Semantic Web Services and Web Process Composition, Springer-Verlag, 2005, pp. 43–54.
[5] Y. Chen, J. Huang, and C. Lin, "Partial Selection: An Efficient Approach for QoS-Aware Web Service Composition," in 2014 IEEE International Conference on Web Services, 2014, pp. 1–8.
[6] Q. Wu and F. Ishikawa, "Towards Service Skyline for Multi-granularity Service Composition," in Proceedings of the 2014 International Workshop on Web Intelligence and Smart Sensing - IWWISS '14, 2014, pp. 1–6.
[7] M. Suresh Kumar and P. Varalakshmi, "A Novel Approach for Dynamic Web Service Composition through Network Analysis with Backtracking," in Advances in Computing and Information Technology, 2013, pp. 357–365.
[8] S. R. Ponnekanti and A. Fox, "SWORD?: A Developer Toolkit for Web Service Composition," Proc. Elev. Int. World Wide Web Conf., vol. 45, pp. 1–23, 2009.
[9] Y. Yao and H. Chen, "A Rule-Based Web Service Composition Approach," in 2010 Sixth International Conference on Autonomic and Autonomous Systems, 2010, pp. 150–155.
[10] L. Huang, X. Zhang, Y. Huang, G. Wang, and R. Wang, "A Qos Optimization for Intelligent and Dynamic Web Service Composition Based on Improved PSO Algorithm," in 2011 Second International Conference on Networking and Distributed Computing, 2011, pp. 214–217.
[11] M. Li, T. Deng, H. Sun, H. Guo, and X. Liu, "GOS: A Global Optimal Selection Approach for QoS-Aware Web Services Composition," in 2010 Fifth IEEE International Symposium on Service Oriented System Engineering, 2010, pp. 7–14.
[12] W. Li and H. Yan-xiang, "A Web Service Composition Algorithm Based on Global QoS Optimizing with MOCACO," in Proceedings of the 2011, International Conference on Informatics, Cybernetics, and Computer Engineering (ICCE2011), 2011, pp. 79–86.
[13] H. Rezaie, N. NematBaksh, and F. Mardukhi, "A Multi-Objective Particle Swarm Optimization for Web Service Composition," in NDT 2010: Networked Digital Technologies, 2010, pp. 112–122.
[14] L. Li, P. Cheng, L. Ou, and Z. Zhang, "Applying Multi-objective Evolutionary Algorithms to QoS-Aware Web Service Composition," in ADMA 2010: Advanced Data Mining and Applications, 2010, pp. 270–281.

[15] J. He, L. Chen, X. Wang, and Y. Li, "Web Service Composition Optimization Based on Improved Artificial Bee Colony Algorithm," J. Networks, vol. 8, no. 9, Sep. 2013.
[16] F. Chen, M. Li, and H. Wu, "GACRM: A dynamic multi-Attribute decision making approach to large-Scale Web service composition," Appl. Soft Comput., vol. 61, pp. 947–958, Dec. 2017.
[17] F. Chen, R. Dou, M. Li, and H. Wu, "A flexible QoS-aware Web service composition method by multi-objective optimization in cloud manufacturing," Comput. Ind. Eng., vol. 99, pp. 423–431, Sep. 2016.
[18] H. Wang, B. Zou, G. Guo, D. Yang, and J. Zhang, "Integrating Trust with User Preference for Effective Web Service Composition," IEEE Trans. Serv. Comput., vol. 10, no. 4, pp. 574–588, Jul. 2017.
[19] X. Liang, A. K. Qin, K. Tang, and K. C. Tan, "QoS-aware Web Service Composition with Internal Complementarity," IEEE Trans. Serv. Comput., pp. 1–1, 2016.
[20] Y. Liu, J. Liao, Q. Qi, J. Wang, and J. Wang, "Lightweight approach for multi-objective web service composition," IET Softw., vol. 10, no. 4, pp. 116–124, Aug. 2016.
[21] S. Niu, G. Zou, Y. Gan, Y. Xiang, and B. Zhang, "Towards the optimality of QoS-aware web service composition with uncertainty," Int. J. Web Grid Serv., vol. 15, no. 1, p. 1, 2019.
[22] S.-L. Fan, F. Ding, C.-H. Guo, and Y.-B. Yang, "Supervised Web Service Composition Integrating Multi-objective QoS Optimization and Service Quantity Minimization," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 10966 LNCS, Springer, Cham, 2018, pp. 215–230.
[23] S. Chibani Sadouki and A. Tari, "Multi-objective and discrete Elephants Herding Optimization algorithm for QoS aware web service composition," RAIRO - Oper. Res., vol. 53, no. 2, pp. 445–459, Apr. 2019.
[24] A. Kedia, A. Pandel, A. Mohata, and S. Sowmya Kamath, "An Intelligent Algorithm for Automatic Candidate Selection for Web Service Composition," Springer, Singapore, 2018, pp. 373–382.
[25] A. Sawczuk da Silva, H. Ma, Y. Mei, and M. Zhang, "A Hybrid Memetic Approach for Fully Automated Multi-Objective Web Service Composition," in 2018 IEEE International Conference on Web Services (ICWS), 2018, pp. 26–33.
[26] M. H. Shirvani, "Web Service Composition in multi-cloud environment: A bi-objective genetic optimization algorithm," in 2018 Innovations in Intelligent Systems and Applications (INISTA), 2018, pp. 1–6.
[27] X. Sun et al., "A Fluctuation-Aware Approach for Predictive Web Service Composition," in 2018 IEEE International Conference on Services Computing (SCC), 2018, pp. 121–128.
[28] Y. Cheng and C. Ding, "Optimization of web services composition using artificial bee colony algorithm," in 2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), 2017, pp. 1–6.
[29] D. H. Elsayed, M. H. Gheith, E. S. Nasr, and A. E. D. M. El Ghazali, "Integration of Parallel Genetic Algorithm and Q-learning for QoS-aware Web Service Composition," in 2017 12th International Conference on Computer Engineering and Systems (ICCES), 2017, pp. 221–226.
[30] Y. Zhao, W. Tan, and T. Jin, "QoS-aware Web Service Composition Considering the Constraints between Services," in Proceedings of the 12th Chinese Conference on Computer Supported Cooperative Work and Social Computing - ChineseCSCW '17, 2017, pp. 229–232.
[31] M. Alrifai, T. Risse, and W. Nejdl, "A hybrid approach for efficient Web service composition with end-to-end QoS constraints," ACM Trans. Web, vol. 6, no. 2, pp. 1–31, May 2012.
[32] V. Gabrel, M. Manouvrier, and C. Murat, "Optimal and Automatic Transactional Web Service Composition with Dependency Graph and 0-1 Linear Programming," in ICSOC 2014: Service-Oriented Computing, 2014, pp. 108–122.

[33] B. S, C. Rajendran, and S. RP, “Penalty Based Mathematical Models for Web Service Composition in a Geo-Distributed Cloud Environment,” in 2017 IEEE International Conference on Web Services (ICWS), 2017, pp. 886–889.
[34] M. Ghobaei-Arani and A. Souri, “LP-WSC: a linear programming approach for web service composition in geographically distributed cloud environments,” J. Supercomput., vol. 75, no. 5, pp. 2603–2628, May 2019.
[35] M. Alrifai, D. Skoutas, and T. Risse, “Selecting skyline services for QoS-based web service composition,” in Proceedings of the 19th international conference on World wide web - WWW ’10, 2010, p. 11.
[36] C.-F. Lin, R.-K. Sheu, Y.-S. Chang, and S.-M. Yuan, “A relaxable service selection algorithm for QoS-based web service composition,” Inf. Softw. Technol., vol. 53, no. 12, pp. 1370–1381, 2011.
[37] E. Kaldeli, A. Lazovik, and M. Aiello, “Continual Planning with Sensing for Web Service Composition,” Artif. Intell., pp. 1198–1203, 2010.
[38] D. Darling Jemima and G. R. Karpagam, “Conceptual framework for semantic web service composition,” in 2016 International Conference on Recent Trends in Information Technology (ICRTIT), 2016, pp. 1–6.
[39] F. Wagner, B. Kloepper, F. Ishikawa, and S. Honiden, “Towards robust service compositions in the context of functionally diverse services,” in Proceedings of the 21st international conference on World Wide Web - WWW ’12, 2012, p. 969.
[40] L. Wu, Y. Zhang, and Z. Di, “A service-cluster based approach to service substitution of web service composition,” in Proceedings of the 2012 IEEE 16th International Conference on Computer Supported Cooperative Work in Design (CSCWD), 2012, pp. 564–568.
[41] M. Rathore, M. Rathore, and U. Suman, “A quality of service broker based process model for dynamic web service composition,” Proc. 3RD Int. Work. Model. Enterp. Inf. Syst. (EIS’ 07, pp. 1267–1274, 2011.
[42] N. H. Rostami, E. Kheirkhah, and M. Jalali, “An Optimized Semantic Web Service Composition Method Based on Clustering and Ant Colony Algorithm,” Feb. 2014.
[43] K. Huynh, T. Quan, and T. Bui, “Smaller to Sharper: Efficient Web Service Composition and Verification Using On-the-fly Model Checking and Logic-Based Clustering,” in International Conference on Computational Science and Its Applications, 2016, pp. 453–468.
[44] J. Li, Y. Yan, and D. Lemire, “Full Solution Indexing for Top-K Web Service Composition,” IEEE Trans. Serv. Comput., vol. 11, no. 3, pp. 521–533, 2018.
[45] J. Li, Y. Yan, and D. Lemire, “Full Solution Indexing for Top-K Web Service Composition,” IEEE Trans. Serv. Comput., vol. 11, no. 3, pp. 521–533, May 2018.
[46] V. A. Permadi and B. J. Santoso, “Efficient skyline-based web service composition with QoS-awareness and budget constraint,” in 2018 International Conference on Information and Communications Technology (ICOIACT), 2018, pp. 855–860.
[47] S. Chattopadhyay and A. Banerjee, “QoS constrained Large Scale Web Service Composition using Abstraction Refinement,” IEEE Trans. Serv. Comput., pp. 1–1, 2017.
[48] Z.-Z. Liu, D.-H. Chu, Z.-P. Jia, J.-Q. Shen, and L. Wang, “Two-stage approach for reliable dynamic Web service composition,” Knowledge-Based Syst., vol. 97, pp. 123–143, Apr. 2016.
[49] S. Bansal, A. Bansal, G. Gupta, and M. B. Blake, “Generalized semantic Web service composition,” Serv. Oriented Comput. Appl., vol. 10, no. 2, pp. 111–133, Jun. 2016.
[50] A. Sawczuk da Silva, Y. Mei, H. Ma, and M. Zhang, “Particle Swarm Optimisation with Sequence-Like Indirect Representation for Web Service Composition,” in EvoCOP 2016: Evolutionary Computation in Combinatorial Optimization, 2016, pp. 202–218.

[51] A. Sawczuk da Silva, Y. Mei, H. Ma, and M. Zhang, “A memetic algorithm-based indirect approach to web service composition,” in 2016 IEEE Congress on Evolutionary Computation (CEC), 2016, pp. 3385–3392.

[52] A. Sawczuk da Silva, H. Ma, Y. Mei, and M. Zhang, “A Hybrid Memetic Approach for Fully Automated Multi-Objective Web Service Composition,” in 2018 IEEE International Conference on Web Services (ICWS), 2018, pp. 26–33.

[53] Y. Yan, M. Chen, and Y. Yang, “Anytime QoS optimization over the PlanGraph for web service composition,” in Proceedings of the 27th Annual ACM Symposium on Applied Computing - SAC ’12, 2012, p. 1968.

[54] R. Eshuis, F. Lécué, and N. Mehandjiev, “Flexible Construction of Executable Service Compositions from Reusable Semantic Knowledge,” ACM Trans. Web, vol. 10, no. 1, pp. 1–27, Feb. 2016.

[55] I. Salomie, V. R. Chifu, and C. B. Pop, “Hybridization of Cuckoo Search and Firefly Algorithms for Selecting the Optimal Solution in Semantic Web Service Composition,” in Cuckoo Search and Firefly Algorithm, Springer International Publishing, 2014, pp. 217–243.

[56] C. B. Pop, V. R. Chifu, I. Salomie, and M. Dinsoreanu, “Immune-Inspired Method for Selecting the Optimal Solution in Web Service Composition,” in RED 2009: Resource Discovery, 2010, pp. 1–17.

[57] Z. Zhang, W. Li, Z. Wu, and W. Tan, “Towards an Automata-Based Semantic Web Services Composition Method in Context-Aware Environment,” in 2012 IEEE Ninth International Conference on Services Computing, 2012, pp. 320–327.

[58] Y. Xiao, X. Zhou, and X. Huang, “Automated Semantic Web Service Composition Based on Enhanced HTN,” in 2010 Fifth IEEE International Symposium on Service Oriented System Engineering, 2010, pp. 59–63.

[59] H. Tong, J. Cao, S. Zhang, and M. Li, “A Distributed Algorithm for Web Service Composition Based on Service Agent Model,” IEEE Trans. Parallel Distrib. Syst., vol. 22, no. 12, pp. 2008–2021, Dec. 2011.

[60] X. Tang, C. Jiang, and M. Zhou, “Automatic Web service composition based on Horn clauses and Petri nets,” Expert Syst. Appl., vol. 38, no. 10, pp. 13024–13031, 2011.

[61] A. Bekkouche, S. M. Benslimane, M. Huchard, C. Tibermacine, F. Hadjila, and M. Merzoug, “QoS-aware optimal and automated semantic web service composition with user’s constraints,” Serv. Oriented Comput. Appl., vol. 11, no. 2, pp. 183–201, Jun. 2017.

[62] F. Moo Mena, R. Hernandez Ucan, V. Uc Cetina, and F. Madera Ramirez, “Web service composition using the bidirectional Dijkstra algorithm,” IEEE Lat. Am. Trans., vol. 14, no. 5, pp. 2522–2528, May 2016.

[63] P. Rodriguez-Mier, C. Pedrinaci, M. Lama, and M. Mucientes, “An Integrated Semantic Web Service Discovery and Composition Framework,” IEEE Trans. Serv. Comput., vol. 9, no. 4, pp. 537–550, Jul. 2016.

[64] F. Bey, S. Bouyakoub, and A. Belkhir, “Time-Based Web Service Composition,” Int. J. Semant. Web Inf. Syst., vol. 14, no. 2, pp. 113–137, Apr. 2018.

[65] S.-L. Fan, Y.-B. Yang, and X.-X. Wang, “Efficient Web Service Composition via Knapsack-Variant Algorithm,” Springer, Cham, 2018, pp. 51–66.

[66] L. Ţucăr and P. Diac, “Semantic Web Service Composition based on Graph Search,” Procedia Comput. Sci., vol. 126, pp. 116–125, Jan. 2018.

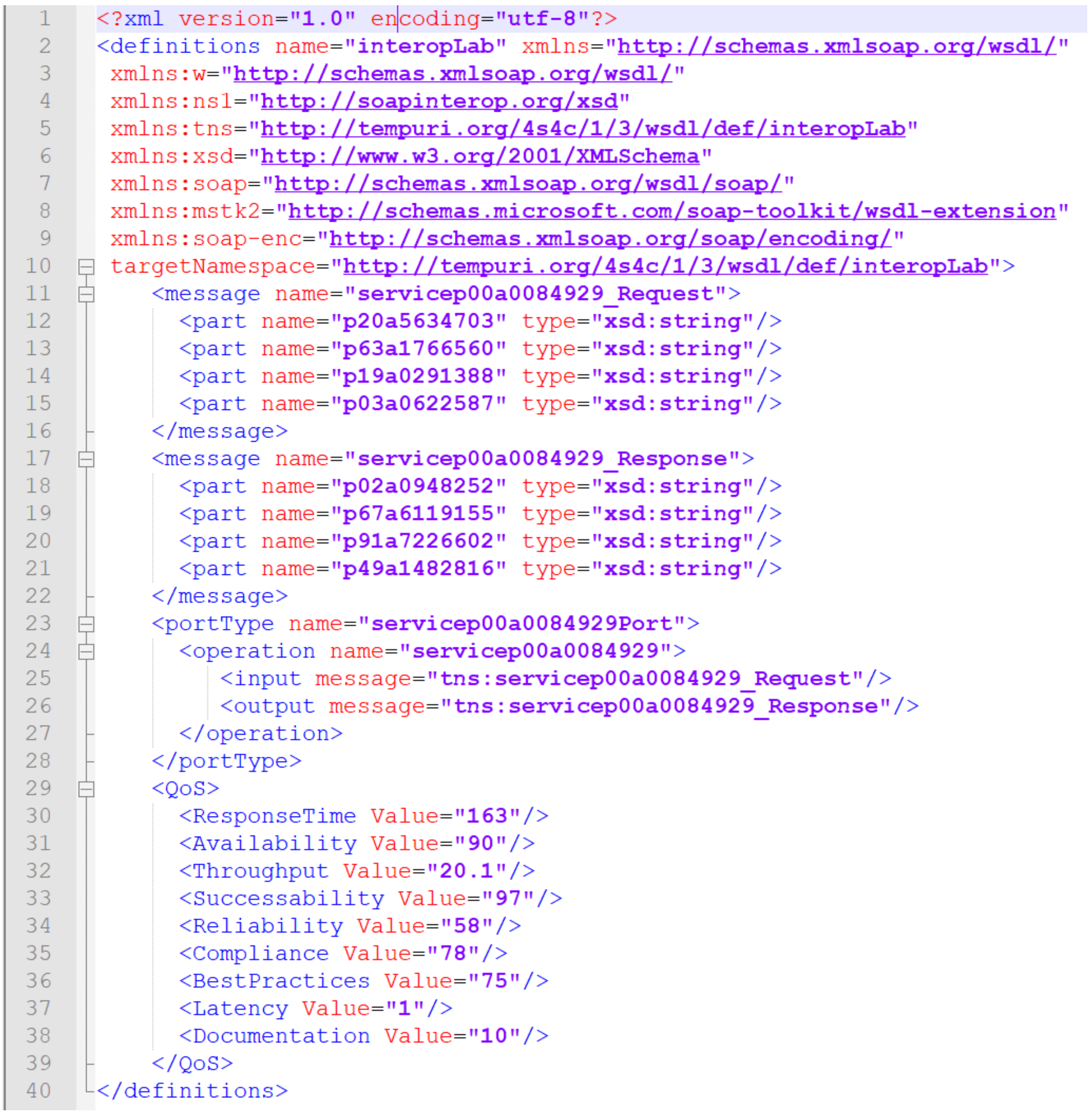
[67] H. Elmaghraoui, L. Benhlima, and D. Chiadmi, “Dynamic web service composition using AND/OR directed graph,” in 2017 3rd International Conference of Cloud Computing Technologies and Applications (CloudTech), 2017, pp. 1–8.

[68] S.-L. Fan, Y.-B. Yang, and X.-X. Wang, "Efficient Web Service Composition via Knapsack-Variant Algorithm," Springer, Cham, 2018, pp. 51–66.
[69] A. Bekkouche, S. M. Benslimane, M. Huchard, C. Tibermacine, F. Hadjila, and M. Merzoug, "QoS-aware optimal and automated semantic web service composition with user's constraints," Serv. Oriented Comput. Appl., vol. 11, no. 2, pp. 183–201, Jun. 2017.
[70] S. Niu, G. Zou, Y. Gan, Z. Zhou, and B. Zhang, "UCLAO* and BHUC: Two Novel Planning Algorithms for Uncertain Web Service Composition," in 2016 IEEE International Conference on Services Computing (SCC), 2016, pp. 531–538.
[71] Bo Zhang, "A heuristic bidirectional search algorithm for automatic Web service composition," in 2010 International Conference on Advanced Intelligence and Awareness Internet (AIAI 2010), 2010, pp. 407–411.
[72] S. Deng, B. Wu, J. Yin, and Z. Wu, "Efficient planning for top-K Web service composition," Knowl. Inf. Syst., vol. 36, no. 3, pp. 579–605, Sep. 2013.
[73] N. Ukey, R. Niyogi, A. Milani, and K. Singh, "A Bidirectional Heuristic Search Technique for Web Service Composition," in ICCSA 2010: Computational Science and Its Applications – ICCSA 2010, 2010, pp. 309–320.
[74] C.-S. Wu and I. Khoury, "Tree-based Search Algorithm for Web Service Composition in SaaS," in 2012 Ninth International Conference on Information Technology - New Generations, 2012, pp. 132–138.
[75] P. Rodriguez-Mier, M. Mucientes, and M. Lama, "Automatic Web Service Composition with a Heuristic-Based Search Algorithm," in 2011 IEEE International Conference on Web Services, 2011, pp. 81–88.
[76] F. D. O. Jr and J. De Oliveira, "QoS-based Approach for Dynamic Web Service Composition.," J. Univers. Comput. Sci., vol. 17, no. 5, pp. 712–741, 2011.
[77] M. Ghobaei-Arani, A. A. Rahmanian, M. S. Aslanpour, and S. E. Dashti, "CSA-WSC: cuckoo search algorithm for web service composition in cloud environments," Soft Comput., vol. 22, no. 24, pp. 8353–8378, Dec. 2018.
[78] H. Fekih, S. Mtibaa, and S. Bouamama, "An Efficient User-Centric Web Service Composition Based on Harmony Particle Swarm Optimization," Int. J. Web Serv. Res., vol. 16, no. 1, pp. 1–21, Jan. 2019.
[79] H. Ye and T. Li, "Web Service Composition with Uncertain QoS: An IQCP Model," Springer, Singapore, 2019, pp. 146–162.
[80] S. Deng, Y. Du, and L. Qi, "A Web Service Composition Approach Based on Planning Graph and Propositional Logic," J. Organ. End User Comput., vol. 31, no. 3, pp. 1–16, Jul. 2019.
[81] E. Shahsavari and S. Emadi, "Semantic Constraint and QoS-Aware Large-Scale Web Service Composition," Shahrood Univ. Technol., vol. 7, no. 1, pp. 181–191, Jan. 2019.
[82] J. Huang, Y. Zhou, Q. Duan, and C. Xing, "Semantic Web Service Composition in Big Data Environment," in GLOBECOM 2017 - 2017 IEEE Global Communications Conference, 2017, pp. 1–7.
[83] J. Alves and J. Marchi, "Web Service Composition: An Agent-Based Approach," in 2017 Brazilian Conference on Intelligent Systems (BRACIS), 2017, pp. 121–126.
[84] "Set cover problem." [Online]. Available: https://en.wikipedia.org/wiki/Set_cover_problem. [Accessed: 16-Jul-2017].
[85] D. S. Johnson, "Approximation algorithms for combinatorial problems," J. Comput. Syst. Sci., vol. 9, no. 3, pp. 256–278, Dec. 1974.
[86] J. . Beasley and P. . Chu, "A genetic algorithm for the set covering problem," Eur. J. Oper. Res., vol. 94, no. 2, pp. 392–404, Oct. 1996.

[87] R. Jovanovic and M. Tuba, “An ant colony optimization algorithm with improved pheromone correction strategy for the minimum weight vertex cover problem,” Appl. Soft Comput., vol. 11, no. 8, pp. 5360–5366, Dec. 2011.
[88] S. C. Geyik, B. K. Szymanski, P. Zerfos, and D. Verma, “Dynamic Composition of Services in Sensor Networks,” in 2010 IEEE International Conference on Services Computing, 2010, no. Scc, pp. 242–249.
[89] V. Gabrel, M. Manouvrier, K. Moreau, and C. Murat, “QoS-aware automatic syntactic service composition problem: Complexity and resolution,” Futur. Gener. Comput. Syst., Apr. 2017.
[90] “Journal Citation Reports - Web of Science Group.” [Online]. Available: https://clarivate.com/webofscien
citation-reports/. [Accessed: 22-Mar-2020].
[91] M. Khani and S. Araban, “QoS-WSC,” Mendeley Data, 2020.

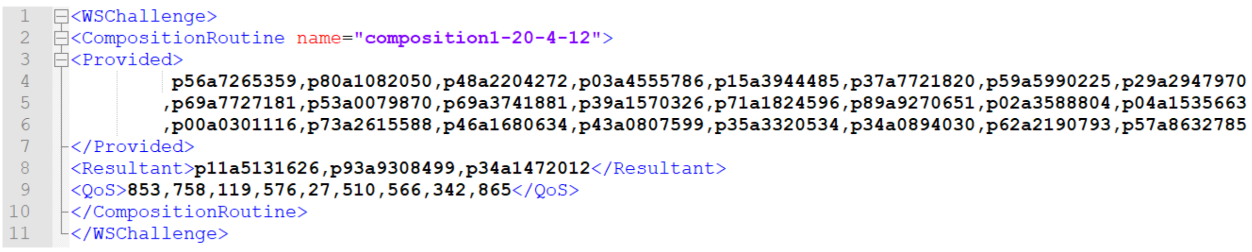
Appendix

Appendix 1: A sample of WSDL file



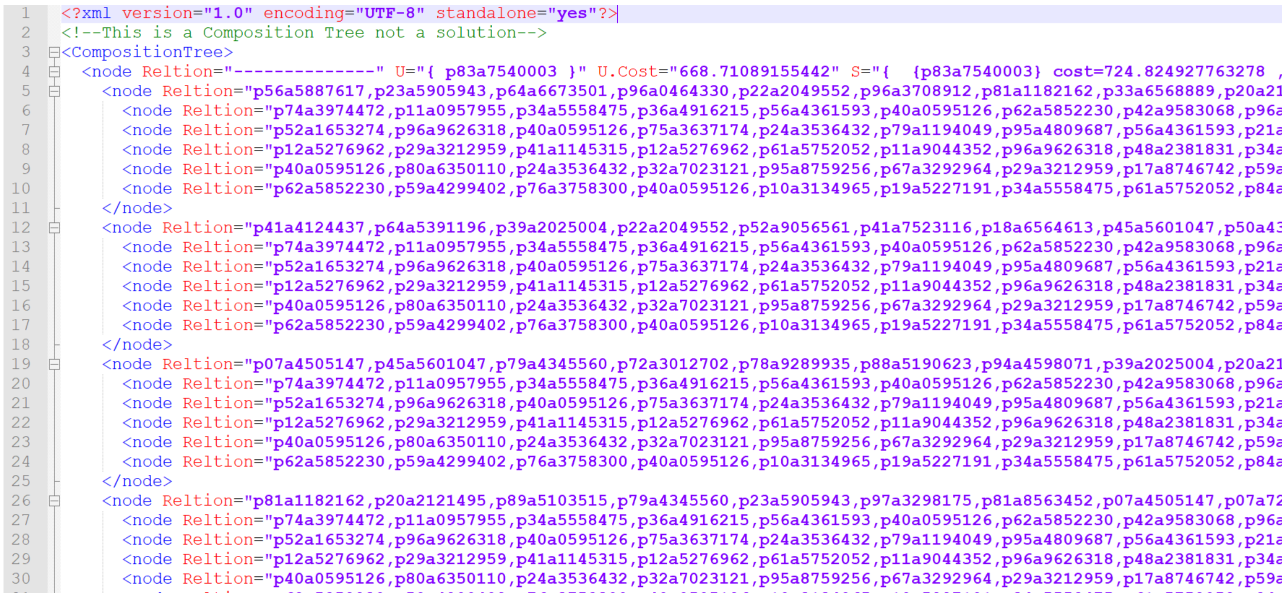
```xml
<?xml version="1.0" encoding="utf-8"?>
<definitions name="interopLab" xmlns="http://schemas.xmlsoap.org/wsdl/"
 xmlns:w="http://schemas.xmlsoap.org/wsdl/"
 xmlns:ns1="http://soapinterop.org/xsd"
 xmlns:tns="http://tempuri.org/4s4c/1/3/wsdl/def/interopLab"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:mstk2="http://schemas.microsoft.com/soap-toolkit/wsdl-extension"
 xmlns:soap-enc="http://schemas.xmlsoap.org/soap/encoding/"
 targetNamespace="http://tempuri.org/4s4c/1/3/wsdl/def/interopLab">
    <message name="servicep00a0084929_Request">
      <part name="p20a5634703" type="xsd:string"/>
      <part name="p63a1766560" type="xsd:string"/>
      <part name="p19a0291388" type="xsd:string"/>
      <part name="p03a0622587" type="xsd:string"/>
    </message>
    <message name="servicep00a0084929_Response">
      <part name="p02a0948252" type="xsd:string"/>
      <part name="p67a6119155" type="xsd:string"/>
      <part name="p91a7226602" type="xsd:string"/>
      <part name="p49a1482816" type="xsd:string"/>
    </message>
    <portType name="servicep00a0084929Port">
      <operation name="servicep00a0084929">
         <input message="tns:servicep00a0084929_Request"/>
         <output message="tns:servicep00a0084929_Response"/>
      </operation>
    </portType>
    <QoS>
      <ResponseTime Value="163"/>
      <Availability Value="90"/>
      <Throughput Value="20.1"/>
      <Successability Value="97"/>
      <Reliability Value="58"/>
      <Compliance Value="78"/>
      <BestPractices Value="75"/>
      <Latency Value="1"/>
      <Documentation Value="10"/>
    </QoS>
</definitions>
```

Appendix 2: A sample of composition query



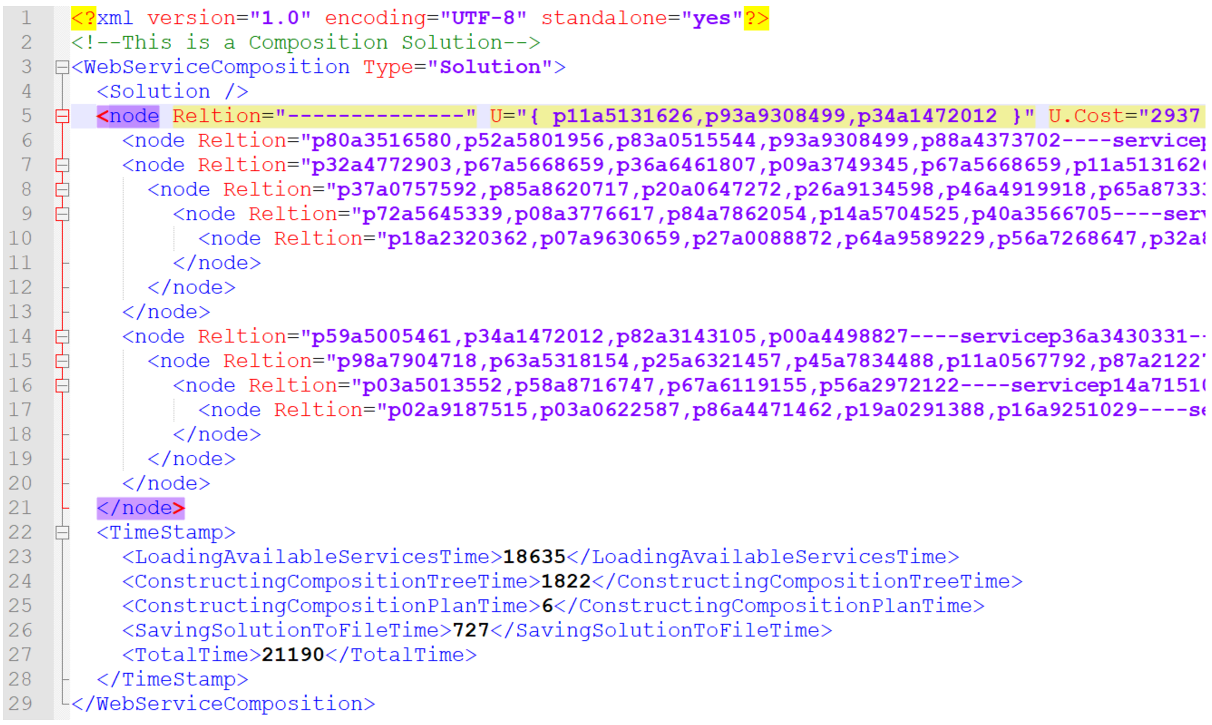
```xml
<WSChallenge>
<CompositionRoutine name="composition1-20-4-12">
<Provided>
            p56a7265359,p80a1082050,p48a2204272,p03a4555786,p15a3944485,p37a7721820,p59a5990225,p29a2947970
           ,p69a7727181,p53a0079870,p69a3741881,p39a1570326,p71a1824596,p89a9270651,p02a3588804,p04a1535663
           ,p00a0301116,p73a2615588,p46a1680634,p43a0807599,p35a3320534,p34a0894030,p62a2190793,p57a8632785
</Provided>
<Resultant>p11a5131626,p93a9308499,p34a1472012</Resultant>
<QoS>853,758,119,576,27,510,566,342,865</QoS>
</CompositionRoutine>
</WSChallenge>
```

Appendix 3: A part of expanded composition tree



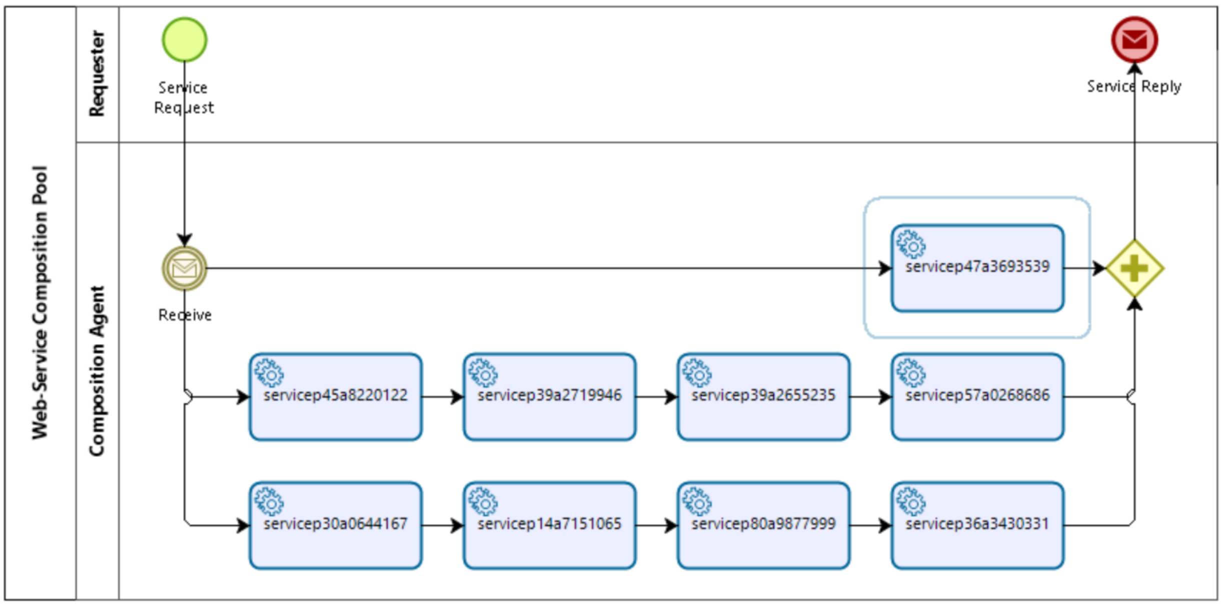
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!--This is a Composition Tree not a solution-->
<CompositionTree>
  <node Reltion="--------------" U="{ p83a7540003 }" U.Cost="668.71089155442" S="{  {p83a7540003} cost=724.824927763278 ,
    <node Reltion="p56a5887617,p23a5905943,p64a6673501,p96a0464330,p22a2049552,p96a3708912,p81a1182162,p33a6568889,p20a21
      <node Reltion="p74a3974472,p11a0957955,p34a5558475,p36a4916215,p56a4361593,p40a0595126,p62a5852230,p42a9583068,p96a
      <node Reltion="p52a1653274,p96a9626318,p40a0595126,p75a3637174,p24a3536432,p79a1194049,p95a4809687,p56a4361593,p21a
      <node Reltion="p12a5276962,p29a3212959,p41a1145315,p12a5276962,p61a5752052,p11a9044352,p96a9626318,p48a2381831,p34a
      <node Reltion="p40a0595126,p80a6350110,p24a3536432,p32a7023121,p95a8759256,p67a3292964,p29a3212959,p17a8746742,p59a
      <node Reltion="p62a5852230,p59a4299402,p76a3758300,p40a0595126,p10a3134965,p19a5227191,p34a5558475,p61a5752052,p84a
    </node>
    <node Reltion="p41a4124437,p64a5391196,p39a2025004,p22a2049552,p52a9056561,p41a7523116,p18a6564613,p45a5601047,p50a43
      <node Reltion="p74a3974472,p11a0957955,p34a5558475,p36a4916215,p56a4361593,p40a0595126,p62a5852230,p42a9583068,p96a
      <node Reltion="p52a1653274,p96a9626318,p40a0595126,p75a3637174,p24a3536432,p79a1194049,p95a4809687,p56a4361593,p21a
      <node Reltion="p12a5276962,p29a3212959,p41a1145315,p12a5276962,p61a5752052,p11a9044352,p96a9626318,p48a2381831,p34a
      <node Reltion="p40a0595126,p80a6350110,p24a3536432,p32a7023121,p95a8759256,p67a3292964,p29a3212959,p17a8746742,p59a
      <node Reltion="p62a5852230,p59a4299402,p76a3758300,p40a0595126,p10a3134965,p19a5227191,p34a5558475,p61a5752052,p84a
    </node>
    <node Reltion="p07a4505147,p45a5601047,p79a4345560,p72a3012702,p78a9289935,p88a5190623,p94a4598071,p39a2025004,p20a21
      <node Reltion="p74a3974472,p11a0957955,p34a5558475,p36a4916215,p56a4361593,p40a0595126,p62a5852230,p42a9583068,p96a
      <node Reltion="p52a1653274,p96a9626318,p40a0595126,p75a3637174,p24a3536432,p79a1194049,p95a4809687,p56a4361593,p21a
      <node Reltion="p12a5276962,p29a3212959,p41a1145315,p12a5276962,p61a5752052,p11a9044352,p96a9626318,p48a2381831,p34a
      <node Reltion="p40a0595126,p80a6350110,p24a3536432,p32a7023121,p95a8759256,p67a3292964,p29a3212959,p17a8746742,p59a
      <node Reltion="p62a5852230,p59a4299402,p76a3758300,p40a0595126,p10a3134965,p19a5227191,p34a5558475,p61a5752052,p84a
    </node>
    <node Reltion="p81a1182162,p20a2121495,p89a5103515,p79a4345560,p23a5905943,p97a3298175,p81a8563452,p07a4505147,p07a72
      <node Reltion="p74a3974472,p11a0957955,p34a5558475,p36a4916215,p56a4361593,p40a0595126,p62a5852230,p42a9583068,p96a
      <node Reltion="p52a1653274,p96a9626318,p40a0595126,p75a3637174,p24a3536432,p79a1194049,p95a4809687,p56a4361593,p21a
      <node Reltion="p12a5276962,p29a3212959,p41a1145315,p12a5276962,p61a5752052,p11a9044352,p96a9626318,p48a2381831,p34a
      <node Reltion="p40a0595126,p80a6350110,p24a3536432,p32a7023121,p95a8759256,p67a3292964,p29a3212959,p17a8746742,p59a
```

Appendix 4: a sample of produced composed service plan



```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!--This is a Composition Solution-->
<WebServiceComposition Type="Solution">
  <Solution />
  <node Reltion="--------------" U="{ p11a5131626,p93a9308499,p34a1472012 }" U.Cost="2937
    <node Reltion="p80a3516580,p52a5801956,p83a0515544,p93a9308499,p88a4373702----service
    <node Reltion="p32a4772903,p67a5668659,p36a6461807,p09a3749345,p67a5668659,p11a513162
      <node Reltion="p37a0757592,p85a8620717,p20a0647272,p26a9134598,p46a4919918,p65a8733
        <node Reltion="p72a5645339,p08a3776617,p84a7862054,p14a5704525,p40a3566705----ser
          <node Reltion="p18a2320362,p07a9630659,p27a0088872,p64a9589229,p56a7268647,p32a
        </node>
      </node>
    </node>
    <node Reltion="p59a5005461,p34a1472012,p82a3143105,p00a4498827----servicep36a3430331-
      <node Reltion="p98a7904718,p63a5318154,p25a6321457,p45a7834488,p11a0567792,p87a2122
        <node Reltion="p03a5013552,p58a8716747,p67a6119155,p56a2972122----servicep14a7151
          <node Reltion="p02a9187515,p03a0622587,p86a4471462,p19a0291388,p16a9251029----s
        </node>
      </node>
    </node>
  </node>
  <TimeStamp>
    <LoadingAvailableServicesTime>18635</LoadingAvailableServicesTime>
    <ConstructingCompositionTreeTime>1822</ConstructingCompositionTreeTime>
    <ConstructingCompositionPlanTime>6</ConstructingCompositionPlanTime>
    <SavingSolutionToFileTime>727</SavingSolutionToFileTime>
    <TotalTime>21190</TotalTime>
  </TimeStamp>
</WebServiceComposition>
```

Appendix 5: BPMN diagram of composed service


Web-Service Composition Pool
Requester
Composition Agent
Service Request
Service Reply
Receive
servicep47a3693539
servicep45a8220122
servicep39a2719946
servicep39a2655235
servicep57a0268686
servicep30a0644167
servicep14a7151065
servicep80a9877999
servicep36a3430331