Applied Intelligence (2022) 52:1979–2000
https://doi.org/10.1007/s10489-021-02351-0

Reinforcement learning infused intelligent framework for semantic web service composition

RL infused intelligent framework for SWSC

N. G. Swetha[1] · **G. R. Karpagam**[1]

Accepted: 12 March 2021 / Published online: 2 June 2021

Abstract
Web Services being the predominant aspect of the web, plays an inevitable role in everyday digital life.With an upsurge in web services, the process of combining them to solve a user query has become complicated. Investigators have proposed the usage of various techniques like Artificial Intelligence, Machine Learning and others to solve the problem of service composition which till date has serious unaddressed flaws. This leads to the need of a new intelligent framework capable of reducing the problem dimension which leads to a well structured composition process. This article proposes a novel framework that incorporates the usage of Formal Concept Analysis and Reinforcement Learning to compose the semantic web services thereby providing an efficient solution to the user query. The novelty of the work lies in the usage of Formal Concept Analysis which reduces the complexity of the composition search space thereby making the composition process effective. This article also utilizes the Reinforcement Learning technique with a relatively new reward model which encompasses the semantic input and output to determine the underlying pattern. The proposed framework is tested for the best Reinforcement Learning strategy through rigorous experimentation and the best Reinforcement Learning Algorithm is incorporated into the Intelligent Framework. The novel framework is evaluated using various queries belonging to varied domains to test its reliability and robustness. It is evident from the results that the proposed framework is efficient when compared with the state of art works and is more suitable for real-time service composition.

Keywords Semantic web service · Formal concept analysis · Learning automata · Reinforcement learning · Markov decision process

1 Introduction

The introduction of the World Wide Web in the late 19^{th} century paved way for the digital era which simplified the lifestyle of the people around the globe. It turned out to be the most powerful tool of the current world as anything and everything can be achieved through internet. When a user wants to accomplish an operation via internet, he inputs the query, based on which the most appropriate tool to solve the operation is reverted which will be used to solve the user query. Till date, web services serve as the predominant tool to perform computation over the internet which makes it an integral part of the World Wide Web. The major industries like Amazon, eBay, Flipkart and many others across the globe use web services to solve the real-world crisis. [6] represents the statistics of web service usage developed for Drupal, a content management system as illustrated in Fig. 1. From Fig. 1, it is apparent that the usage of web services (in this case specifically web services developed for Drupal) per day is not less than 6 lakhs, which is a clear indication of how popular web services are in the current scenario.

The web services till date are used in various real time applications. A web service as such can be used to process raw data over the network. It can also be embedded onto any web based application/mobile applications to perform specific tasks. It can also be used for data exchange across

✉ N. G. Swetha
swetha4694@gmail.com

G. R. Karpagam
grk.cse@psgtech.ac.in

[1] Department of Computer Science & Engineering, PSG College of Technology, Coimbatore, 641004, India

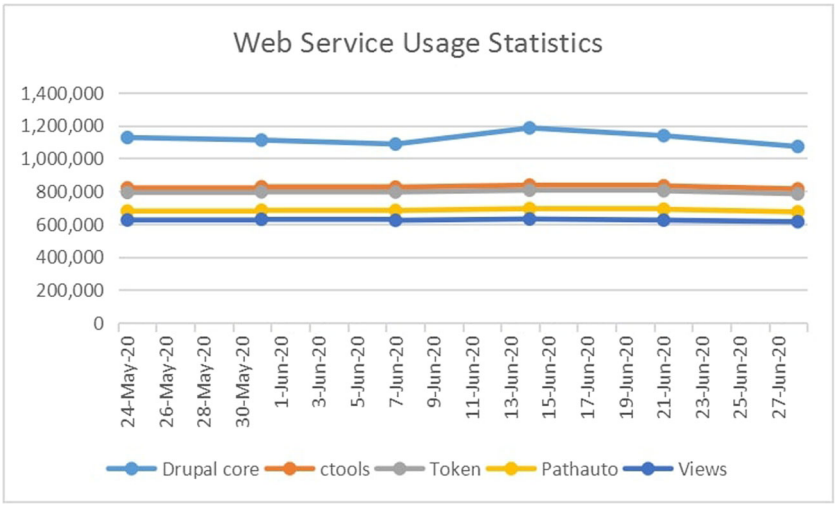


Fig. 1 Web Service Usage Statistics (Drupal.org) [6]

various platforms. The role of web based applications and mobile based applications are increasing in day to day life and almost every real time problem has a mobile app or a web based application as a solution. For example the traditional shopping now a days is replaced with e-shopping facilities after the introduction of mobile based apps and web based applications. The companies which facilitate the e-commerce generally develop a mundane framework for online shopping. This framework has multiple services developed from scratch (Fully Developed Application) which results in repeated development of existing services as shown in Fig. 2. Pre developed and deployed web services which facilitate this very process are available as a part of the registry. The developer of the application generally queries the registry which composes a set of web services that will be useful for the client and gives it as an output, in this case, a composed plan of services that facilitate the e-shopping scenario is returned to the application developer. He then incorporates these web services into the framework (Static Half Composed Application) that is designed. This process reduces the effort and time taken for the development of online portals as it brings in re usability of code but at the same time since the composition is static, the services once incorporated into the framework cannot be changed further. This posses a non availability threat to the composed services which may reduce the reliability of the framework developed. This lead to the development of a Dynamic Half Composed Applications as shown in Fig. 2, whose framework is capable of dynamically composing web services during

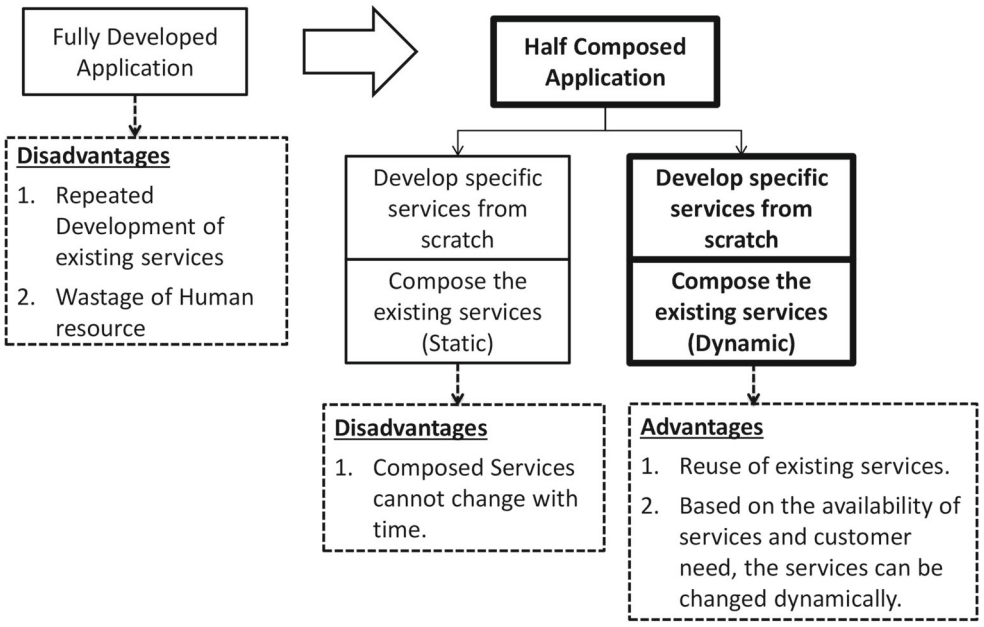


Fig. 2 Evolution of Application Framework

the times of need. Incorporating developed and tested web services as a part of the web/mobile based applications dynamically increases the reliability of the entire framework developed. This approach is widely used by a number of startups as well as established companies in almost every domain. Thus, the dynamic web service composition tools are of significant importance which determines the reliability and robustness of the half composed dynamic applications and as a result of the end to end automation done these days, the importance and need of the web service composition tools are increasing day by day.

Web Services are generally granular which is designed to solve smaller tasks at a time. But in the real-world, the user query is complex which cannot be solved by the execution of a single granular web service. For example, consider the real-world domain of Retail Merchandise. Here, the web services like S1: View Item, S2: Select item for purchase, S3: Add to cart service, S4: Place Order and many others are available to perform online sales. The service S1 can only be used to view the list of items on sales. Similarly, the service S2 can be used to select items for purchase. The Service S3 can be used to place the items on the cart for purchase. The Service S4 enables the user to place an online purchase order. As observed, to perform a single task of online shopping, more than one web services have to be integrated in a specific order as the web services are granular. This process of identifying the order in which a set of web services has to be executed to fulfill the user request is termed as web service composition which remains to be a complex process. It involves the processing of thousands of web services and hence it remains to be a research challenge termed as the Web Service Composition Challenge. This process of dynamic composition involves the usage of semantics to understand the user query and to compose a set of services and so the Web Service Composition Challenge can also be termed as Semantic Web Service Composition (SWSC) Challenge.

Every web service has three important aspects namely, Web Service Provider, Web Service Requester and Registry. A proper synchronization among these elements is essential to satisfy a user query. The Web Service Provider is the one who creates a web service for public usage. Once the web services are created and tested, it is deployed on a server for public access. The information regarding how to access the deployed service is available as a part of the Web Service Description Language (WSDL) file [13, 14, 18]. The Web Service Provider shares the WSDL file to the Registry who in turn provides the information to the requesting client. The Web Service Requester or the client can then access this deployed Web Service through the WSDL file. The WSDL file contains the necessary end to end connectivity details of how a web service can be accessed. In addition to these the various inputs needed for the functioning of a web service and the format of the end result are described in the WSDL file [13, 14, 18].

The Registry is the most essential aspect of any web service. The Web Service Provider on creation of the web service, registers it in the registry using the WSDL file. These registered web services becomes available to all the Web Service Requester or Clients for usage. The Web Web Service Requester issues the query based on his need to the registry. With respect to the query issued, the registry discovers the web services and returns it to the client. The client can then select the list web services needed to satisfy his request manually which is a tedious process. Universal Description, Discovery, and Integration (UDDI) is one such registry that holds services related to the business process [22]. UDDI has a standard mechanism for service discovery which can be utilized by the Web Service Requester to find the relevant web services for a given user query [22]. The proposed framework utilizes this discovery mechanism and composes the web services and returns the composed web services to satisfy the client.

The rest of the paper is organized as follows: Section 1 provides a brief introduction about the problem statement and the various concepts utilized in the work. Section 2 highlights the various literature works carried out in the area of service composition. The research gap, proposed mathematical model and the proposed Intelligent Framework are explained in Section 3. Section 4 illustrates an E-Commerce based Case Study. The experimental evaluation and the results of the proposed framework are depicted in Section 5 followed by the conclusion of the proposed work.

1.1 Formal concept analysis

Formal Concept Analysis (FCA) is a numerical framework for investigating data through the idea of concepts [7]. A collection of objects with a set of common attributes is known as a concept. The objects in a concept is called the extent and the attributes are known as the intent. Thus, a concept can be characterized as the set of intents and extents of a common hierarchy. Group of concepts form a Formal Context. Every Formal context is represented through line diagram known as concept lattice.

Definition 1 Formal Context A Formal Context can be arithmetically interpreted as a tuple (O,A,R) where, O denotes a collection of objects, A denotes a collection of Attributes and $R \subseteq O \times A$ (R represents the binary relationship between O and A) [8]. The binary relationship (R) between O and A is represented in the form of a cross table as explained in [8].

Definition 2 Formal Concept A Formal Concept can be numerically interpreted as pair of (O1,A1) where, $O1 \subseteq$

$O, A1 \subseteq A, O1' = A1\ and\ A1' = O1$ as suggested by [7]. $O1'$ and $A1'$ can be represented as follows,

- If $O1 \subseteq O, then\ O1' = \{a \in A | \forall\ o \in O1 : oRa\}$
- If $A1 \subseteq A, then\ A1' = \{o \in O | \forall\ a \in A1 : oRa\}$

Definition 3 Extent and Intent For a Formal Concept (O1,A1) where $O1 \subseteq O$ and $A1 \subseteq A$, set O1 is called its extent and set A1 is called its intent.

Definition 4 Super Concept Consider the Formal Concepts (O1,A1) and (O2,A2) where $O1, O2 \subseteq O$ and $A1, A2 \subseteq A$. Here, the concept (O1,A1) is said to be a Super Concept of (O2,A2) if $A2 \subset A1$ and $O2 \subset O1$.

1.2 Reinforcement learning

Reinforcement Learning (RL) is a reward based learning scheme which focus on solving the given problem by constantly interacting with the world [16]. RL scheme generates its own training data by interacting with its environment which implies that learning is done by trail and error method. Reinforcement Learning is a part of Machine Learning whose learning depends on the rewards generated for each action taken. As RL is a dynamic learning scheme which adjusts itself to the changing environment, its key aspect is to maximize the rewards achieved.

Elements of reinforcement learning Every problem to be solved using RL technique have an Agent and an Environment [16]. Agent is considered to be the controller who is in charge of every step taken to solve the problem. The Agent during every step taken learns a better way to reach the goal. Environment is the world with which the agent interacts to eventually reach the goal. Every problem to be solved under an environment has an initial state or the start state and a goal state. Apart from these a number of intermediate states are also possible which depicts a particular condition of the problem in the given environment. Action is a path that determines the transition from one state to another state. During this transition through an action, the environment generates a reward evaluating the action. Reward is always a scalar value that indicates whether the action taken by the agent is desirable or not. A reward given by the environment is mainly of three types,

1. A positive reward indicates, the action taken by the agent is desirable. This type of reward tends to push the agent in choosing the action more frequently.
2. A Negative reward indicates, the action taken by the agent is not desirable. This reward emphasizes the fact that, if the same action is chosen frequently, the goal state may not be achieved efficiently. So, the negative rewards tends to push the agent away from choosing this action again in the future.
3. A Neutral reward indicates, the action chosen leads to a neutral state which has no positive as well as no negative influence in achieving the goal.

Q Function A state action pair known as the Q function denoted by $Q(s, a)$ indicates the value of the state 's' taken the action 'a' under the given environment [16].

Policy A policy (π) influences the agents behavior of choosing an action at a particular time instance [16]. In other words, a policy can be defined as a mapping between the current state and the action to be taken in that particular state.

Markov Decision Process Markov Decision Process(MDP) portrays two important aspects of the problems [16]. First, different situations of an environment has to be responded differently. Second, a particular action taken in an environment influences the reward obtained. Based on these two aspects of MDP it is evident that almost every real world problem can be formulated using MDP. The Markov property highlights the fact that, the future state and the reward only depends on the current state and the action taken.

Reinforcement Learning Algorithms There are several Reinforcement Learning Algorithms [16] that can be applied on the real world problems to obtain an efficient solution. Among the various RL algorithms available, five Algorithms are found to be suitable for the Semantic Web Service Composition Challenge as described in Section 1. The Algorithms namely SARSA, λ-SARSA, Expected SARSA, Q-Learning and λ-Q-Learning are applied to the service composition as suggested in Section 5 and the best performing algorithm for the composition challenge is chosen for the proposed framework.

2 Literature review

In spite of the challenge posed by web service composition, number of methods and techniques are proposed and experimented by researchers throughout the world. This section highlights some of the research works that are currently carried out in the domain of service composition.

In 2015, the authors of [17] have proposed a novel framework that utilizes the most simple data structure 'graph' for both discovery and composition purposes. According to the authors of [17], usage of simple data structure for composition process reduces the complexity of the challenge and helps in unraveling the underlying

patterns that exists among the web services. The service composition research applying the Reinforcement Learning (RL) techniques was ignited by the authors of [11, 21]. In [11], the authors have formulated the service composition as a partially observable Markov Decision Process (MDP). The model suggested was compared using various RL algorithms like Monte Carlo, Q-Learning and others. The authors have concluded that, usage of RL algorithms significantly increases the efficiency of composition task. The research work [21] also formulates the composition challenge as MDP and applies various other RL algorithms like Policy Iteration,Value Iteration and others. The authors through this work has concluded that, Policy Iteration is most effective for service composition as its complexity is lesser than the other algorithms taken for study. But the process of updating the value function followed in Policy Iteration Algorithm does not utilize the Q value which is found to be a major set back for composition challenge.

In 2016, the authors of [12] have utilized the Genetic Algorithm(GA) to compose the given set of web services. A variation of GA known as the Culture GA is applied, which translated the global constraints to local constraints for processing. Though this process is found to be effective, the complexity of this technique is much higher. The authors of [4] have conducted an extensive survey of the various techniques used for chaining of web services. The authors clearly indicate the various Artificial Intelligence based framework. This work does not include any RL techniques which clearly indicates that usage of RL in the field of Service composition is very low in spite of its efficiency. The research work [25] focuses on Multi Agent RL model for solving the composition challenge. A distributed model which utilizes the Q-Learning is applied here. The authors have taken no effort to analyze which RL learning algorithm suits the model proposed. In the similar fashion, the authors of [23] have utilized Q-Learning to compose services. The author initially have decomposed the composition plan into composition hierarchy which is then given to the Q-Learning algorithm for further processing. Clearly, the authors have made no experimental analysis to conclude why Q-Learning suits there need.

In 2017, the researchers of [5] have utilized a variation of GA called as Harmony Search Algorithm for composition process. Although the performance of modified GA is good, the complexity is very high making it unsuitable for composition process.The authors of [20] have used Fluent Calculus a Logical programming approach to solve the semantic service composition problem. Although Fluent Calculus serves as the state of art technique, the time complexity is found to be high. The authors of [15] have made a noteworthy contribution of adding constraints the the MDP. The authors of this work have utilized Q-Learning algorithm to support there developed model but have failed to experiment the most suited RL algorithm for there framework.

In 2018, the authors of [9] have proposed Elite-guided Multi-objective Artificial Bee Colony (EMOABC) Algorithm to solve the service composition challenge. It makes use of fast sorting combined with population selection and elite guide strategy for fitness calculation. This method achieves high accuracy at the cost of execution time.

In 2019, the authors of [10] have employed the basic Genetic Algorithm (GA) with modified crossover and selection strategy. The authors of [3] has employed AI based graph planning composition approach with FCA. Although usage of FCA for discovery significantly reduces the complexity of discovery aspect, focus is not laid on the reduction of composition complexity.

In 2020, the authors of [26] have employed the usage of Deep Q Learning to dynamically compose the web services. But the usage of Deep Q Learning has found to increase the complexity of the problem as 2 different Deep Neural Networks are employed on behalf of Q Table. This imposes additional complexity to the existing problem in terms of memory and time. In [24], the authors have used Recurrent Neural Network for QoS prediction and Q-Learning for Composition. The execution time is found to increase during the training phase of the RNN which will have a negative impact on the user satisfaction rate. This article aims to overcome the flaws of the literature by analyzing various RL algorithms first and then incorporating the most suitable algorithm into the framework to solve the Semantic Web Service Composition (SWSC) Challenge.

3 Proposed system

3.1 Research gap

From the Literature Survey in Section 2 it can be observed that multiple Artificial Intelligence based techniques and various other Optimization methods were applied to solve the web service composition problem which has multiple unaddressed issues as follows,

1. Since there is an upsurge in the development and deployment of web services in 20^{th} century, the number of web services relevant to the respective user query is very high. Similarly to complete a task thousands of web services are available. As a result, the total web services which are processed by the composition engine are found to be in multiples of thousand. This increases the complexity of the composition process as all of these thousand services has to be taken into consideration.

From the literature survey it is evident that no effort has been made to reduce the complexity of the problem.

2. To compose multiple web services various techniques like Graph based composition, Fluent Calculus based composition, Multi objective optimization techniques, Genetic Algorithms and various other techniques have been employed till date. Each of these techniques have their own advantages and disadvantages. But the most common flaw observed among all these techniques is that they are highly time inefficient. Reduction of execution time is predominant in providing solution to the SWSC problem, as high execution time always leads to low user satisfaction rate. The user is always concerned in obtaining accurate results in a shorter duration. So, there is a need to improve the time efficiency of the plan generation process.
3. The AI techniques employed in the literature explore all the available services for composition process. This exploration is least significant most of the time which considerably increases execution time. Hence, there is a need to employ a technique which can compose the given SWSC problem with higher accuracy and moderate exploration rate.

This article aims in solving the above said research gap. To reduce the complexity of the problem a Filtration Engine is employed. This reduces the number of web services to be composed. As, the complexity is reduced, a significant improvement in the execution time is observed. Reinforcement Learning is a technique where the percentage of exploration and exploitation can be controlled and is more suitable for dynamic problem solving as the training of the model takes place based on the user data provided at that instance. Although RL is more suitable for service composition, from the literature it can be observed that various RL algorithms which excel in different problem domains were considered in a random fashion for modeling. There is a need to identify the most suitable RL algorithm to solve the service composition issue.The proposed work identifies and utilizes the most suitable RL technique to dynamically compose web services with respect to the user query.

3.2 Mathematical model for composition challenge

As illustrated in Section 1 Semantic Web Service Composition remains to be a research challenge which involves various components and phases. Thousands of web services are available in today's world to solve various real time issues. Thus, the web services can be represented using a universal set S as shown in (1) where t represents the total number of services available globally.

$$S = \{Sx \ / \ x \in \mathbb{N} \ and \ x \leq t\} \tag{1}$$

For each web service available three major components such as Domain, Input and Output are utilized during the service composition process. $Sx(I)$ denotes the various Inputs used by the web service Sx. Similarly, $Sx(O)$ and $Sx(D)$ denotes the various Outputs and the Domain to which the web service Sx belongs respectively.

The most predominant component involved in the Semantic Web Service Composition is the Registry. A Registry is a collection of web services from different domains, which is formulated as shown in (2).

$$R = \left\{ \bigcup_{i=1}^{n} \bigcup_{j=1}^{m} Sx_{ij} \right\} \tag{2}$$

A registry can contain multiple services from multiple domains which is depicted in (2). In (2), 'n' represents the total number of domains in the registry and 'm' denotes the total services in a single domain. Thus, Sx_{ij} indicates that the global service Sx is the j^{th} service belonging to i^{th} domain of the registry.

The composition challenge starts with the user query. Each user query is denoted as Q_y where y represents the query number. Collection of all such queries processed by the registry is represented in (3). Query domain is depicted as $Q_y(D)$ which represents the total number of domains to which the query Q_y belongs to such that $Q_y(D) \in \mathbb{N}$ and $Q_y(D) \leq n$. Every query Q_y corresponds to a specific input and output which are represented by the notion $Q_y(I)$ and $Q_y(O)$ respectively.

$$Q = \{Q_y \ / \ y \in \mathbb{N}\} \tag{3}$$

A query Q_y generally can be satisfied by executing atomic or composite services in a specific order belonging to the $Q_y(D)$. Thus the solution for each query is viewed as a set of services from various query domains as shown in (4). The services belonging to the set Q_y can have one or more services with semantically same input and output interfaces as depicted in (5). These services are collectively termed as Alternate Services which is mathematically represented in (6).

$$Q_y = \left\{ \bigcup_{i=1}^{Q_y(D)} \bigcup_{j=1}^{m} Sx_{ij} \right\} \ where \ Q_y \subseteq R \tag{4}$$

$$\forall Sx \in Q_y, \exists \ Sp, Sq \in Q_y \ s.t \ Sp(I) =_s Sq(I) \ and \ Sp(O) =_s Sq(O) \tag{5}$$

$$A_y = \left\{ \bigcup_{i=1}^{k1} \bigcup_{j=1}^{m1} Sx_{ij} \right\} \ where \ A_y \subset Q_y \ and \ k1 \leq Q_y(D) \ and \ m1 < m \tag{6}$$

When these set of Alternate Services are removed from the original query set, Unique Web Services (Q'_y)

is obtained as shown in (7), which can then be used to construct the anatomy of service execution.

$$Q'_y = Q_y - A_y \tag{7}$$

To fulfill a user query, the set of services in Q'_y has to be formulated into a plan which consists of two important steps. First, the composable service pairs has to be identified. The composable services are represented as C in (8), and each element in C indicates a pair of composable services Sab (Sa can be linked with Sb) which is determined by applying the operation $\circ$ (Semantic Subset comparison $\subseteq_s$) on two services as shown in (9).

$$C = \{Sab/\ Sab = Sa \circ Sb\ \forall Sa, Sb \in Q'_y\} \tag{8}$$

$$Sa \circ Sb = \begin{cases} Sab, & \forall Sa, Sb \in Q'_y, Sb(O) \subseteq_s (Sa(I) \cup Q_y(I)) \\ \emptyset, & \forall Sa, Sb \in Q'_y, Sb(O) \nsubseteq_s (Sa(I) \cup Q_y(I)) \end{cases} \tag{9}$$

From (8), different plans can be obtained by semantically chaining the composable services. Thus, the set of generated plans on execution will provide a better solution to the user query.

3.3 Role of FCA and RL in solving the research gap

As explained in Section 3.2, the user query can be solved by composing multiple web services from Q'_y. To formulate Q'_y, the set of alternate services A_y must be eliminated (filtered) from Q_y to reduce the complexity of the problem. From (5) it is evident that the alternate services have similar interfaces which imply that the the services have semantically same parameters. In Concept Lattice, the objects are represented through the attributes they posses. So, the Concept Lattice becomes the natural fit to represent the services through parameters. Thus Concept Lattice makes the process of retrieval and elimination of web services simpler. So, the usage of FCA proves to be a better fit to design the filtration engine which boosts the performance of the composition engine.

Reinforcement Learning can be applied to the problems with the following characteristics,

- The reward given must be delayed or delivered through sequential steps.
- The problem must have opportunity for active exploration.
- The states are partially observable.
- The agent must be able to learn multiple tasks to reach the goal.

The problem of SWSC challenge as explained in Section 1 satisfies the properties which are synonymous with the characteristics of RL mentioned above. According to the SWSC challenge, the reward must be given to the agent in a step by step fashion before reaching the Goal state. Since there are numerous web services available for composition, the exploration space is high and the states are partially observable as the decision to semantically compose the services has to be taken based on the current state. Based on these observations it is found that, RL applies best for the SWSC challenge which can provide dynamic composition result based on the user query.

3.4 RLAP intelligent framework

Reinforcement Learning based Action Planner (RLAP) Intelligent Framework utilizes the power of Formal Concept Analysis for filtration and Reward based learning to build a composition plan. Every Web Service developer registers the service developed to a registry. The user who wishes to utilize the power of web services, provides a user query to the registry. Every registry has a discovery mechanism which returns only the services that match the user request. The framework as depicted in Fig. 3 utilizes this inbuilt discovery mechanism and builds a composition engine on top of the registry. As seen from Fig. 3 the framework has two prominent processing components namely Filtration Engine and Plan Generator.

Filtration Engine gets the set of discovered services Q_y from the registry as input, removes the redundant services / alternate services A_y and returns the unique set of services Q'_y for plan generation. The Plan Generator then generates the composable plan for this unique set of services using the reward based learning scheme of Expected SARSA. Thus, the RLAP Intelligent Framework returns a plan of services to the user query, which when executed in the same order solves the user request.

3.5 Concept lattice infused filtration engine

Filtration Engine being one of the main component of RLAP Intelligent Framework utilizes the power of Formal Concept Analysis to isolate the alternate services. This process is of prime importance as it significantly reduces the size of web services to be composed. The Concept Lattice infused Filtration Engine has important parts to enable efficient filtration as given below,

- Representation of Web Services in Concept Lattice (explained briefly in Section 3.5.1)
- Separation of Alternate Services from Q_y (explained in Section 3.5.2)

3.5.1 Formal concept analysis for alternate service

Formal Concept Analysis as explained in Section 1.1 is a mathematical structure for analyzing the data with common

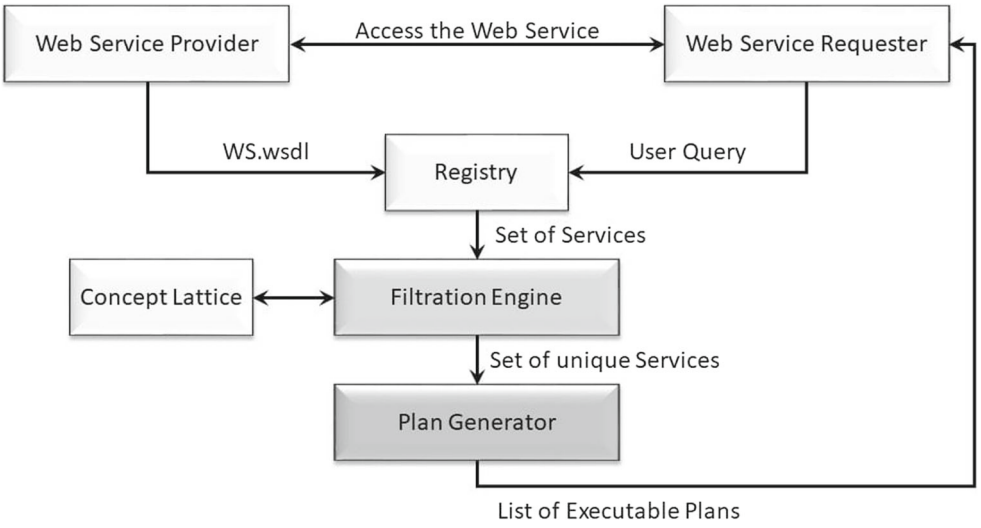


Fig. 3 Reinforcement Learning based Action Planner Intelligent Framework

attribute set termed as concepts. This mathematical structure can be easily applied to web services as each service has a finite set of attributes like Input Parameters and Output Parameters which implies that every web service can be a part of two concepts namely Input Concept and Output Concept. For Example let us consider a web service $WS1$ with Input parameters "ISBN,Count" and Output parameter "Price". The Concept Lattice contains the following objects,

- Object 1: $WS1\#1$; Concept: "Input,ISBN,Count"
- Object 2: $WS1\#2$; Concept: "Output,Price"

The Object 1 belongs to the Input concept and Object 2 belongs to the Output concept and both the objects represents the web service $WS1$. Thus, a single web service is represented as two objects in the concept lattice. Table 1 represents the sample of five services and Table 2 depicts the cross table generated from Table 1 as explained in Section 1.1. Figure 4 represents the Concept Lattice generated using Table 1. Figure 5 represents a dense concept lattice with 10 web services and 20 objects.

3.5.2 Isolation of alternate services

As depicted in (5), every registry will have one or more services with semantically same input and output parameters and those services are termed as Alternate Services A_y. As a result of alternate services in a registry, the discovered service set for a user query also tends to have alternate services which during the phase of composition increases the complexity of the process. So, the Filtration Engine aims in reducing this inborn complexity of composition by eliminating the alternate services for the composition phase.

The set of discovered services returned by the registry Q_y is given as the input to the Filtration Algorithm and the set of unique services Q'_y is returned as shown in Algorithm 1. Initially, a concept lattice is constructed with the set of services from Q_y with 2 dominating concepts namely the Input Concept and the Output Concept. A set of extent with similar input concepts are extracted and stored in list $l1$. For this extraction, the extents are checked for the availability of a super concept which is depicted in Algorithm 2. If a super concept exists then the current extent is ignored. If not, the extents are added to the list. In a similar way, the set of extents with similar output concepts are extracted and stored in list $l2$.

A service can be termed as an alternate service if and only if both the input as well as the output parameters match. So, a match for extents in list $l1$ and $l2$ is looked for. If an exact match is obtained, the extents as such are added to the A_y, if a subset match is found, then only the matching subset is added to the A_y and the rest of the extents are split into separate service sets and are added to the set A_y. Now,

Table 1 Sample Web services for concept lattice depiction

Web service	Input parameter	Output parameter
S1	Author	Book_list, Genre_list
S2	Book_list, Genre_list, Genre, Year	Title
S3	Title	ISBN
S4	ISBN	Count_a (Count Available)
S5	Author, Genre, Year, Count_a,Count	Price

Table 2 Cross table for construction of concept lattice

	Input	Output	author	book_list	genre_list	genre	year	title	ISBN	count_a	count	price
s1#1	X		X									
s1#2		X		X	X							
s2#1	X			X	X	X	X					
s2#2		X						X				
s3#1	X							X				
s3#2		X							X			
s4#1	X								X			
s4#2		X								X		
s5#1	X		X			X	X			X	X	
s5#2		X										X

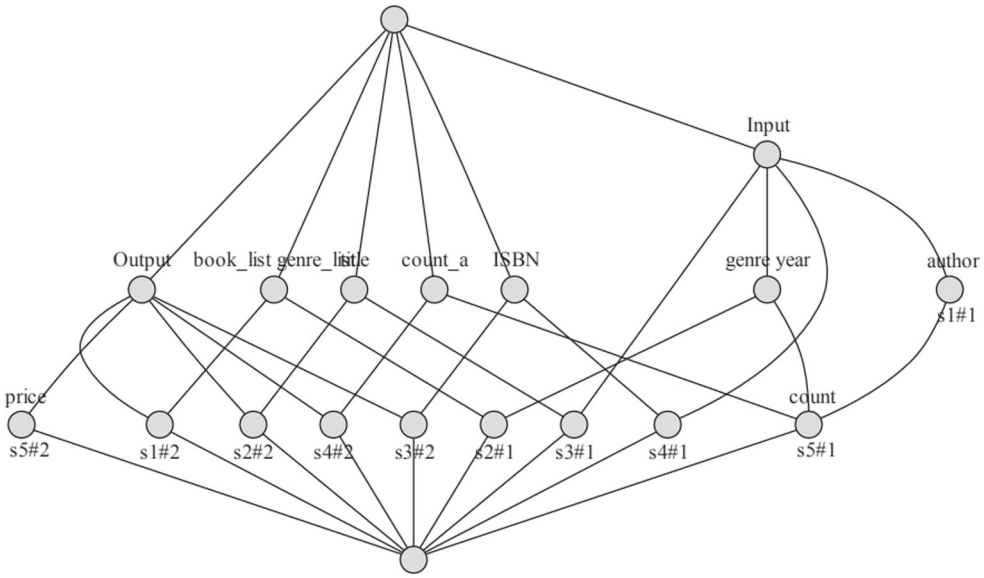


Fig. 4 Sample Concept Lattice

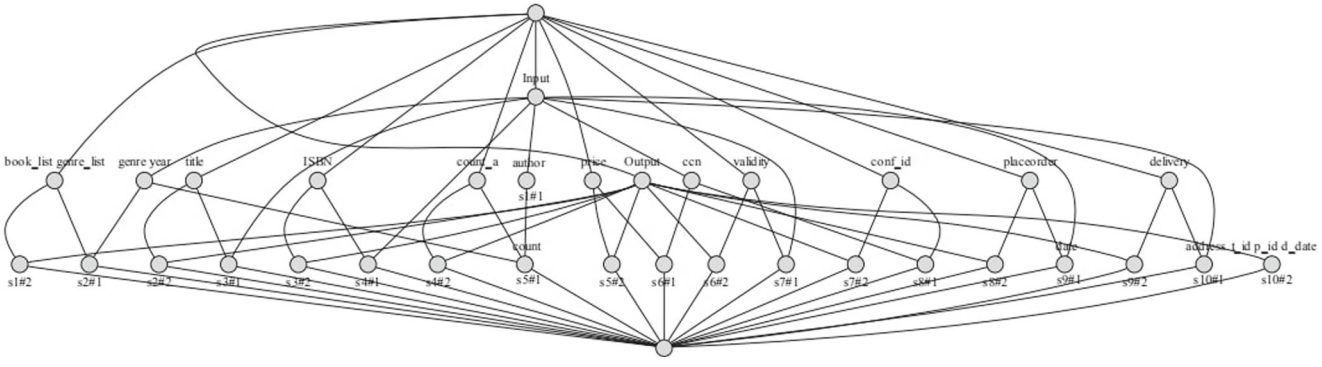


Fig. 5 Sample dense Concept Lattice

A_y contains set of extents with matching input and output parameters.

Algorithm 1 Filteration algorithm.

Input: Q_y
Output: Q'_y

```
l1 ← ∅ ; l2 ← ∅ ; A_y ← ∅ ; Q'_y ← ∅
Lattice ← Construct Concept Lattice from Q_y
foreach input_concept in Lattice do
    e ← extract extent of input_concept
    cc ← Check_Supremum(Lattice,e,1)
    if cc!=∅ then
        Add cc to list l1
    end
end
foreach output_concept in Lattice do
    e ← extract extent of output_concept
    cc ← Check_Supremum(Lattice,e,2)
    if cc!=∅ then
        Add cc to list l2
    end
end
foreach S1 in l1 do
    foreach S2 in l2 do
        if S1 == S2 then
            Add S1 to A_y
        end
        if S1 ⊂ S2 then
            a1 ← combine the matching services into a
            Set
            Add a1 to A_y
            Add all the non matching services as
            seperate set to the A_y
        end
    end
end
foreach Sp in A_y do
    foreach Sq in A_y do
        if Sp ⊂ Sq then
            Flag Sp
        end
    end
end
Remove all Flagged Services from A_y
A_y ← Process_Alternate_Service(A_y)
Q'_y = Q_y − A_y.Value
Return Q'_y
```

From A_y, key web service has to be identified with the best availability in order to ensure user satisfaction and this processing is performed by Algorithm 3. Algorithm 3 ensures that, the service with the best availability becomes the key service ($A_y.Key$) and the rest of the service becomes alternate to the key service ($A_y.Value$). These alternate services ($A_y.Value$) are removed from the set Q_y which yields Q'_y. This Q'_y is then given to the next component of the RLAP Intelligent Framework (Plan Generator) as shown in Fig. 3.

Algorithm 2 Check_supremum algorithm.

Input: Concept Lattice, extent({Sx}), integer
Output: {Sx}

```
if integer == 1 then
    if super input concept exists for the given extent
    then
        Return ∅
    else
        Return extent
    end
end
if integer == 2 then
    if super output concept exists for the given extent
    then
        Return ∅
    else
        Return extent
    end
end
```

Algorithm 3 Process_alternate_service algorithm.

Input: A_y
Output: $A1_y$

```
A1_y ← ∅
foreach S in A_y do
    foreach Sx in S do
        Extract the Availability QoS Value for each Sx
    end
    Key ← Sx with maximum Availability
    Value ← {Sy}
    Add Key in A1_y.Key
    Add Value in A1_y.Value
end
Return A1_y
```

3.6 Expected SARSA powered composition engine

The unique set of services Q'_y from the Filteration Engine is given as the input to the Plan Generator. This section explains in detail about how Expected SARSA is used in the process of composition and the reward model applied to achieve an efficient composition process.

3.6.1 Learning automata for semantic web service composition

Every Reinforcement Learning scheme needs a specific learning automata to induce dynamic learning. Figure 6 depicts the sample learning automata used to compose two web services.

The Reinforcement Learning (RL) environment contain $(n+2)$ states and $(n+2)$ actions where n represents the total number web services. If the value of $n = 2$, then according to the rewarding scheme, there are 4 states and 4 actions in the RL environment which is shown in Fig. 6. Every service is added as a separate state along with the Initial State 'S_0' and the Goal State '$Goal$'. Similarly, transition to every service including 'S_0' and '$Goal$' is marked as action. Thus, the (10) and (11) represents the total states and actions for a composition model of two services.

$$State = \{S_0, Service1, Service2, Goal\} \tag{10}$$

$$Action = \{Move\_to\_S_0, Move\_to\_Service1, Move\_to\_Service2, Move\_to\_Goal\} \tag{11}$$

The reward values given for every transition as shown in Fig. 6 is given below,

- Transition from a service to itself, $Sx \rightarrow Sx : -10$
- Transition from a service to another service is depicted below,

$$Sx \rightarrow Sy : \begin{cases} 10 & , Sx(O) =_s Sy(I) \\ 10 & , (Sx(O) \cup Q_y(I)) =_s Sy(I) \\ -10 & , \text{No Semantic Match} \end{cases} \tag{12}$$

In Fig. 6 the condition 'a' is depicted in Case 1 of (12), condition 'b' is depicted in Case 2 of (12) and condition 'c' is depicted in Case 3 of (12).

- Transition from Initial State to any service is depicted below,

$$S_0 \rightarrow Sx : \begin{cases} 100 & , Q_y(I) =_s Sx(I) \\ \text{Partial Reward} & , Q_y(I) \subset_s Sx(I) \end{cases} \tag{13}$$

In Fig. 6 the condition 'd' is depicted in Case 1 of (13) and condition 'e' is depicted in Case 2 of (13).

- Transition from Service Sx to Goal State is depicted as follows,

$$Sx \rightarrow Goal : \tag{14}$$
$$\begin{cases} 100 & , (Q_y(I) \cup Q_y(O)) \text{ is Satisfied} \\ \text{Partial Reward} & , (Q_y(I) \cup Q_y(O)) \text{ is Partially Satisfied} \end{cases}$$

In Fig. 6 the condition 'f' is depicted in Case 1 of (15) and condition 'g' is depicted in Case 2 of (15).

- All other Transition gets a reward of -1.

The same learning automata can be extended for n number of services. Partial Reward is awarded based on percentage of match obtained.

3.6.2 Expected SARSA and composition

Five Reinforcement Learning algorithms as depicted in Section 1.2 are applied with the proposed reward model explained in Section 3.6.1. The performance of the algorithms are compared as shown in Section 5.2 and it is concluded that the performance of Expected SARSA is better than every algorithm taken for comparison. So, Expected

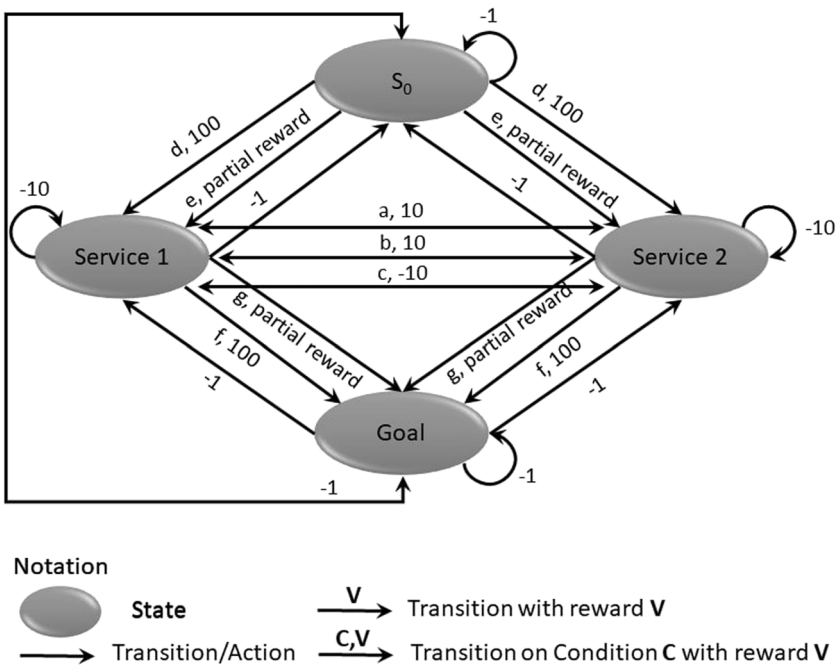


Fig. 6 Learning Automata to compose two web services

SARSA is utilized in the Plan Generator as depicted in Section 3.4. On execution of Expected SARSA on Q'_y a Q-Table is generated from which the plan p_i can be extracted by following the path of highest Q value. Every service Sx in plan p_i can be replaced with its alternate service A_y and multiple plans can be generated. The generated list of plans is given to user. The user can then select a plan from the list of plans based on his preference and the services can be executed in the suggested order to fulfill his request.

4 Case study on online retail merchandise (E-Commerce)

Although Composition of Web Services remain to be a challenge, multiple companies from multiple domains use web services to satisfy their client request. One such promising domain for the usage of web services is the retail merchandise. Multi National Companies such as Amazon, Marks & Spenser and many others utilize web services to enhance their online retail merchandise service. This section focuses on a sample service dataset used for online merchandise and its composition output obtained on applying the RLAP Intelligent framework. The services from the dataset OWL-S [1] (164 web services) and OWLS-SLR [2] (6042 web services) are used for the purpose evaluation. Additional services related to online shopping are also created to support the evaluation purpose. Altogether 7706 web services are used to access the performance of the proposed framework which is explained in Section 5. A sample of 20 services as shown in Table 3 is used to illustrate the working of the framework.

As per the mathematical model suggested in Section 3.2, the sample web services shown in Table 3 are considered for further processing. The total number of services (t) is 20 as per Table 3. The universal set of Services S is depicted in (15).

$$\begin{aligned} S &= \{Sx \;/\; x \in \mathbb{N} \; and \; x \leq t\} \text{ where } t = 20 \\ &= \{S1, S2, ..., S20\} \end{aligned} \tag{15}$$

Let us assume that the UDDI registry holds only the sample services from various domains. According to the data given in Table 3, the value of $n = 9$ and the number of services in each domain vary with a minimum value of 1 and a maximum value of 10. Thus the registry is depicted as shown in (16).

$$\begin{aligned} R &= \left\{ \bigcup_{i=1}^{n} \bigcup_{j=1}^{m} Sx_{ij} \right\} \text{ where } n = 9 \text{ and} \\ m &= \text{number of services in domain} \\ &= \{S1, S2, ..., S20\} \end{aligned} \tag{16}$$

Let the 1^{st} user query given to the registry R be related to the domain Online Retail Merchandise. The registry returns the services related to the query as depicted in (17).

$$\begin{aligned} Q_1 &= \left\{ \bigcup_{i=1}^{Q_1(D)} \bigcup_{j=1}^{m} Sx_{ij} \right\} \; where \; Q_1 \subseteq R \text{ and } Q_1(D) = 1 \\ &= \{S1, S2, ..., S10\} \end{aligned} \tag{17}$$

According to the RLAP Intelligent Framework, the set of services returned by the registry on processing the query (Q_1) is given to the Filteration Engine which will return the set of unique services denoted by Q'_1. To obtain this output, Filteration Algorithm is applied on the set Q_1 and the intermediate values of $l1, l2$ and A_1 are illustrated in (18),(19) and (20) respectively. The value of A_1 on applying Algorithm 3 is shown in (21),(22) and (23). The final set of unique services given by the Algorithm 1 is shown in (24).

$$l1 = \{\{S1, S6\}, \{S2, S7\}, \{S3, S8\}, \{S4, S9\}, \{S5, S10\}\} \tag{18}$$

$$l2 = \{\{S1, S6\}, \{S2, S7\}, \{S3, S8\}, \{S4, S9\}, \{S5, S10\}\} \tag{19}$$

$$A_1 = \{\{S1, S6\}, \{S2, S7\}, \{S3, S8\}, \{S4, S9\}, \{S5, S10\}\} \tag{20}$$

$$\begin{aligned} A_1 &= \{S1 : \{S6\}, S2 : \{S7\}, S3 : \{S8\}, S4 : \{S9\}, \\ &\quad S5 : \{S10\}\} \end{aligned} \tag{21}$$

$$A_1.Key = \{S1, S2, S3, S4, S5\} \tag{22}$$

$$A_1.Value = \{S6, S7, S8, S9, S10\} \tag{23}$$

$$\begin{aligned} Q'_1 &= \{S1, S2, S3, S4, S5, S6, S7, S8, S9, S10\} \\ &\quad -\{S6, S7, S8, S9, S10\} \\ &= \{S1, S2, S3, S4, S5\} \end{aligned} \tag{24}$$

This set of services Q'_1 is given as the input to Plan Generator. The Plan Generator as depicted in Section 3.6 utilizes the power of Reinforcement Learning. On applying Expected SARSA for the set Q'_1, the Q-table as depicted in (25) is obtained. From the Q-table, the Composition plan $S1 \rightarrow S2 \rightarrow S3 \rightarrow S4 \rightarrow S5$ is generated. This indicates that, the services "View item → Select item for purchase → Add to cart Service → Place Order → Track Order" must be executed in the given order to fulfill the user request Q_1.

$$\begin{pmatrix} -1.42 & 99.15 & 0.008 & 66.67 & 33.34 & 3.43 & -1.0 \\ 7.57 & -10.84 & 10.00 & -9.99 & -9.99 & -6.56 & 0.0 \\ 7.57 & 9.15 & -9.99 & 10.00 & -9.99 & -6.56 & 0.0 \\ 7.57 & 9.15 & -9.99 & -9.99 & 10.00 & -6.56 & 0.0 \\ 7.57 & 9.15 & -9.99 & -9.99 & -9.99 & 13.43 & 0.0 \\ 7.57 & 9.15 & -9.99 & -9.99 & -9.99 & -6.56 & 100.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \end{pmatrix} \tag{25}$$

Table 3 Sample Web service details

Service Id	Service name	Service domain	Service input	Service output
S1	View item	Online retail merchandise	Login Token	List of items
S2	Select item for purchase		List of selected items	Selection id
S3	Add to cart Service		Selection id	Cart id
S4	Place Order		Cart Token	Order id
S5	Track Order		Order id	Location details, Order status
S6	View product		Login Token	List of items
S7	Select product		List of selected items	Selection id
S8	Add to cart Service1		Selection id	Cart id
S9	Place Order1		Cart Token	Order id
S10	Track Order1		Order id	Location details, Order status
S11	Book Profile Service	Book	Nil	List of Books
S12	Coffee Tea Report Service	Beverages	Nil	Coffee,Tea,Report
S13	Drought Report Service	Disaster Management	Year	Drought, Report
S14	Food Export Service	Food Mangement	Food	FoodExportability
S15	DestionationService	Location and Mangement	Organization, Surfing	Destination
S16	Email Validator	Connectivity Management	Email,LicenseKey	Body
S17	Synonym	Lexicon Service	LoginToken, word	Synonym
S18	Temperature Convertor Service	Unit Convertor	Value, From Unit, To Unit	Value
S19	Area Convertor Service	Unit Convertor	Value, From Unit, To Unit	Value
S20	Celcius to Farenheit Convertor	Unit Convertor	Value	Value

5 Results and discussions

RLAP Intelligent Framework as explained in Section 3.4 is implemented in an intel i5 core system with 4GB RAM. Anaconda Environment is used for python 3. UDDI registry is used in the RLAP Intelligent framework. The UDDI registry is coupled with python using the UDDI4Py module. WordNet Lexicon is used for semantic comparisons and Concepts module is used in the creation of Concept Lattice in Python. Wu & Palmer's similarity measure is used to find the semantic similarity between the words.

The parameters namely Precision, Recall, F1-Measure and Accuracy are used to analyze the performance of the framework created. For evaluation purposes as described in Section 4, OWL-S dataset created by Andreas Hess in 2007 [1] is used. This dataset contains 164 web services from various domains. As the size of this dataset is very small, 6042 wsdl files from OWLS-TC [2] dataset are annotated using Assam WSDL annotator [1]. In addition to the web services available 1500 web services are created and annotated. Thus a total of 7706 web services from 15 domains are used for the evaluation of RLAP Intelligent framework. A total of 100 queries belonging to 10 domains are obtained from the requester and their respective outputs are acquired from RLAP Intelligent framework. For the cumulative result obtained, confusion matrix is derived and the evaluation parameters are calculated from the confusion matrix. Following formulas are used for calculation of evaluation parameters.

$$Precision = \frac{T_P}{(T_P + F_P)}$$

$$Recall = \frac{T_P}{(T_P + F_N)}$$

$$F1 - Measure = 2 * \frac{(Precision * Recall)}{(Precision + Recall)}$$

$$Accuracy = \frac{(T_P + T_N)}{T_S}$$

Where, T_P indicates True Positive, T_N indicates True Negative, F_P indicates False Positive, F_N indicates False Negative and T_S indicates Total Samples. The evaluated results obtained are of threefold,

1. Optimizing the parameters(explained in Section 5.1) for RL Algorithms under study
2. Comparing the efficiency of RL Algorithms under study
3. Computing the efficiency of RLAP Intelligent Framework

5.1 Optimizing the parameters for RL algorithms

As explained in Section 1.2, the RL Algorithms like SARSA, λ-SARSA, Expected SARSA, Q-Learning and λ-Q-Learning are considered for study purposes with Epsilon-Greedy

method used for action selection. Each of these algorithms have various parameters such as Learning Rate (α), Discount Factor (γ), Credit Assignment for Eligibility Trace (λ) and Greedy Rate (ϵ) which influences the learning ability of the algorithms. The values of these parameters change with respect to the environment in which learning has to be done. So, finding the optimal value of these parameters are of prime importance. While applying SARSA for the suggested service model, the parameters namely ϵ, α and γ play a very important role and the update equation of the algorithm is depicted in [16]. To determine the values for α, ϵ and γ intensive experimentation was conducted with values ranging from 0.1 to 0.9 for each parameter. But there is no significant pattern change observed in terms of Episodes and Rewards for α and ϵ. So, the change in Episodes between consecutive parametric values is calculated and plotted as in Fig. 7a and c. For the estimation of γ a significant pattern of improvement in terms of cumulative rewards is observed. Hence, the value of γ is estimated based on the maximal rewards achieved.

It is observed from Fig. 7a that the change in Episodes is minimal at the values of 0.4 and 0.5 of the Learning Rate. Thus an average of the values 0.4 and 0.5 is computed and the Learning rate (α) is fixed as 0.45 for SARSA. Similarly from Fig. 7c it is noted that, the change in episodes is minimal between the points 0.5 and 0.6. So, arithmetic average of 0.5 and 0.6 is computed and the value 0.55 is fixed as the Greedy Rate (ϵ). For experimentation of both α and ϵ the values ranging from 0.1 to 0.6 are considered, as the values higher than 0.6 produce undesirable results. For the determination of γ which is the rate of Discount, total rewards earned are taken into consideration. Here from the Fig. 7c it is evident that at the value 0.3, the SARSA reaches its maximum reward. So, the γ is assigned with a value of 0.3.

In the similar fashion, the parameters for other four algorithms are determined after intensive experimentation and are depicted in Table 4.

5.2 Observations from various RL algorithms

The optimized parameter values as shown in Table 4 are applied for the respective RL Algorithms and the output in terms of Episodes, Iteration, Time and Rewards are obtained as depicted in Fig. 8 (As a considerable amount of randomness is involved in the RL algorithms, the experimentation is repeated for 100 times and the median value obtained is projected as results here). As, there are multiple parameters involved,a multi criteria decision making approach TOPSIS is used to find the optimal algorithm among the five. Among the parameters in Fig. 8, Rewards obtained has the highest importance in deciding the efficiency of the algorithm. So, Rewards is given a weightage of 0.4 and rest of the parameters namely Episode, Iteration and Time are given equal weightage of 0.2. On applying the TOPSIS algorithm, the performance score of each RL algorithm is obtained and ranked from the highest to the lowest as shown in Table 5.

It is evident from Table 5 that Expected SARSA performs better when compared with the other algorithms taken for study.

5.3 RLAP intelligent framework efficiency

The RLAP Intelligent Framework implemented is tested for its efficiency using the evaluation parameters. The experimental results of the RLAP Intelligent Framework are obtained in such a way that the components like the Filtration Engine and Plan Generator can be evaluated for their efficiency and also the over all efficiency of the system can be evaluated.

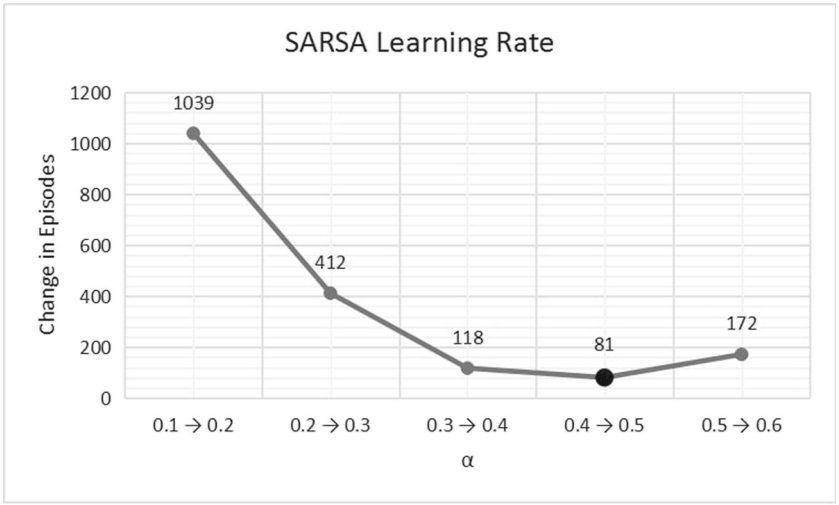
5.3.1 Assessment of concept lattice infused filtration engine

The evaluation of Filtration Engine with respect to the evaluation parameters along with execution time for construction of concept lattice and filtration of services are shown in Fig. 9. From Fig. 9a it is found that, the Filtration Engine using Concept Lattice has an Accuracy of 96.7%, Precision of 99.8%, Recall of 95.9% and F1-Measure of 97.8%. The results are sampled with 100 queries over 10 domains. A higher level of Accuracy and Precision indicates that the Filtration Engine proposed is consistent and reliable.

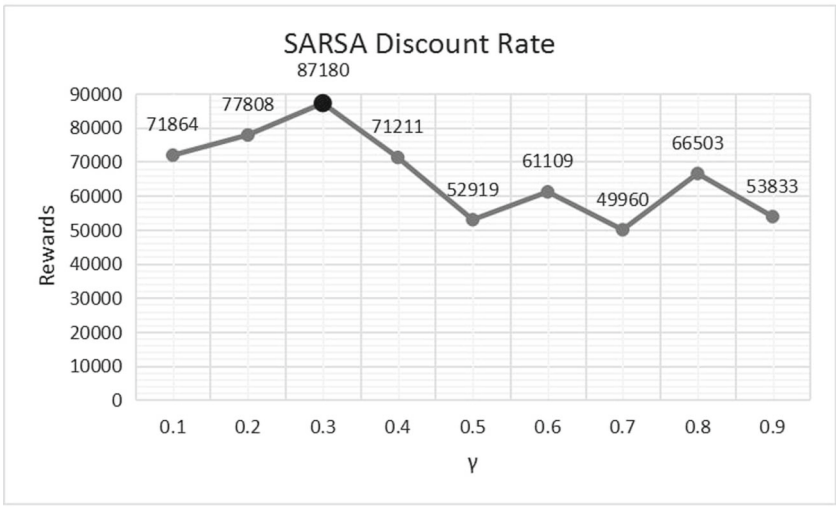
Also the time taken for construction of concept lattice which plays a major role in the Filtration Engine is depicted in Fig. 9b. It is observed that for creating a lattice with 10 services (20 objects in lattice) it takes 0.05 seconds. Similarly, to create a lattice with 100 services (200 objects in lattices) 0.1 seconds is consumed. As the total time taken is less than 1 second usage of concept lattice for filtration proves to be an effective way to maintain the user satisfaction level.

Figure 9c depicts the time taken by the Filtration Engine collectively to filter the alternate services. It can be observed that total filtration time to filter an input of 100 services is around 0.4 seconds inclusive of lattice creation time which is found be acceptable.

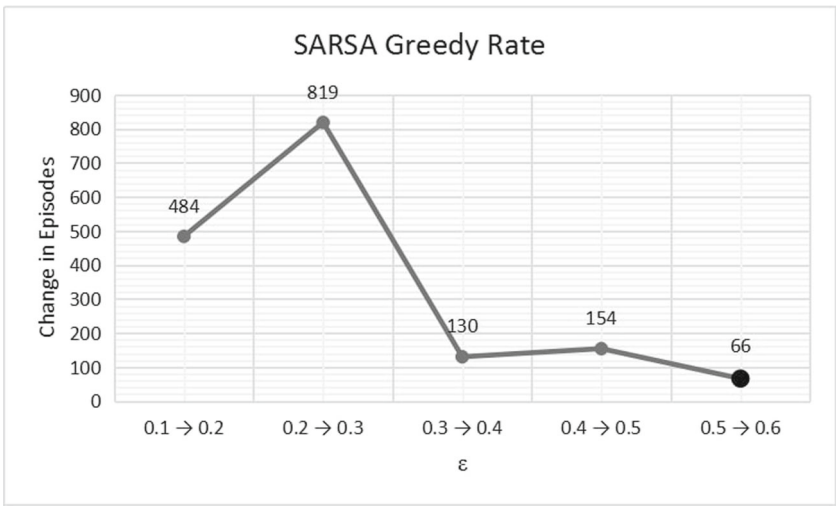
Also from the experimentation done, it is seen that, the percentage of alternate services vary between 25% to 75% for every query. Hence, the role of Filtration Engine is inevitable as it eliminates the alternate services by 92.5% there by greatly reducing the dimensionality of the problem.



(a)



(b)



(c)

Fig. 7 Parameter Tuning for SARSA Algorithm. **a** Estimation of Learning Rate. **b** Estimation of Discount Factor. **c** Estimation of Greedy rate

Table 4 Parameter values for RL algorithms

Algorithm	λ	α	ϵ	γ
SARSA	nil	0.45	0.55	0.3
SARSA Lambda	0.9	0.65	0.45	0.25
Expected Sarsa	nil	0.65	0.45	0.3
Q Learning	nil	0.75	0.55	0.2
Q Learning Lambda	0.8	0.65	0.65	0.15

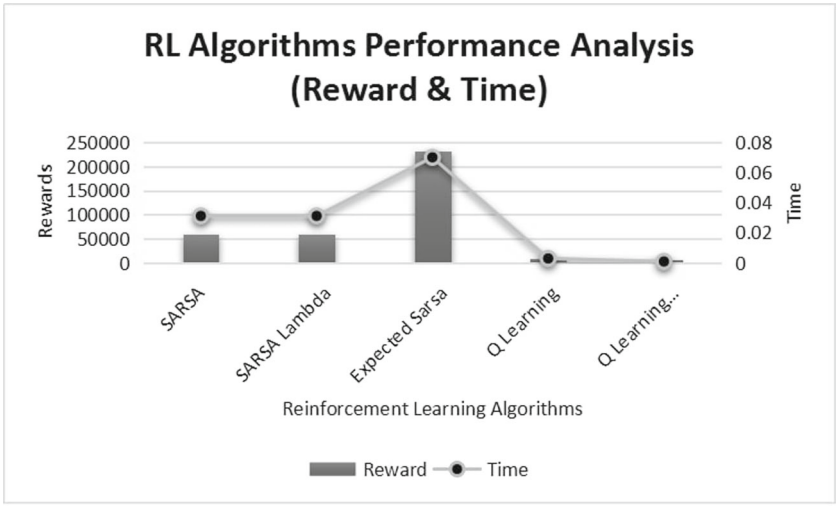
Table 5 TOPSIS performance score of comparison

Algorithm	Performance Score	Rank
SARSA	0.294098471	5
SARSA Lambda	0.29565509	4
Expected SARSA	0.608991445	1
Q Learning	0.380977721	3
Q Learning Lambda	0.391008555	2

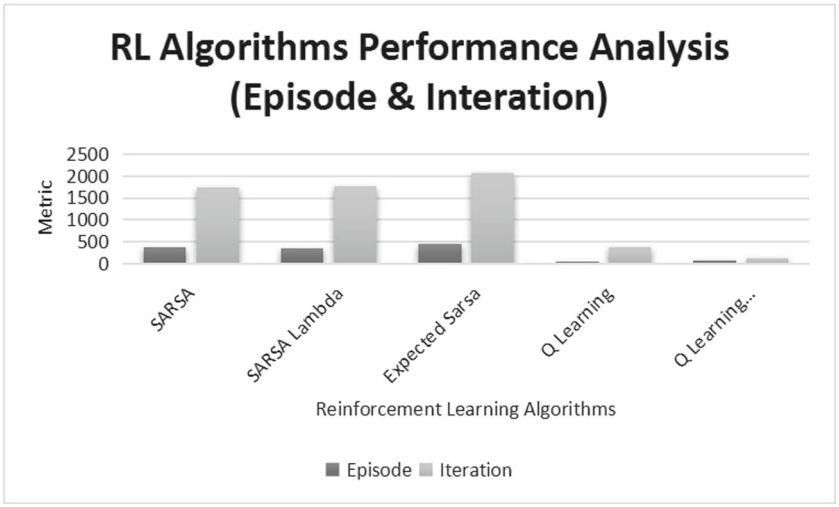
5.3.2 Outcome of expected SARSA powered composition engine

Evaluation of the Plan Generator is shown in Fig. 10. Figure 10a indicates the Execution time of different number of web services with different composition lengths. Composition lengths ranging from 2 to 10 are considered for evaluation purposes. It is found that, for composing a plan of length 10, with 100 web services from the Filtration Engine takes around 0.25 seconds which is better for maintaining user satisfaction and user retention rate.

Figure 10b represents the efficiency of the Plan Generator with respect to the evaluation parameters.The results are sampled using 100 queries over 10 domains. The Plan Generator operates with an Accuracy of 92.5%, Precision of 89.9%, Recall of 99% and F1-Measure of 94%.

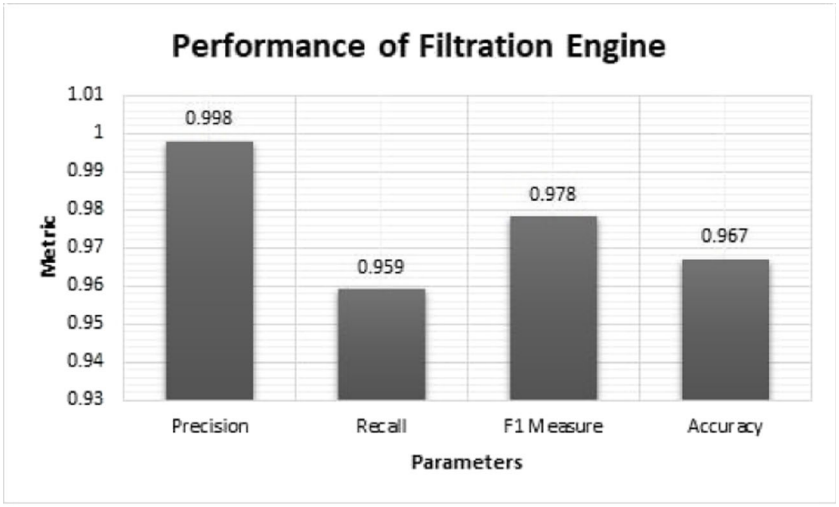


(a)

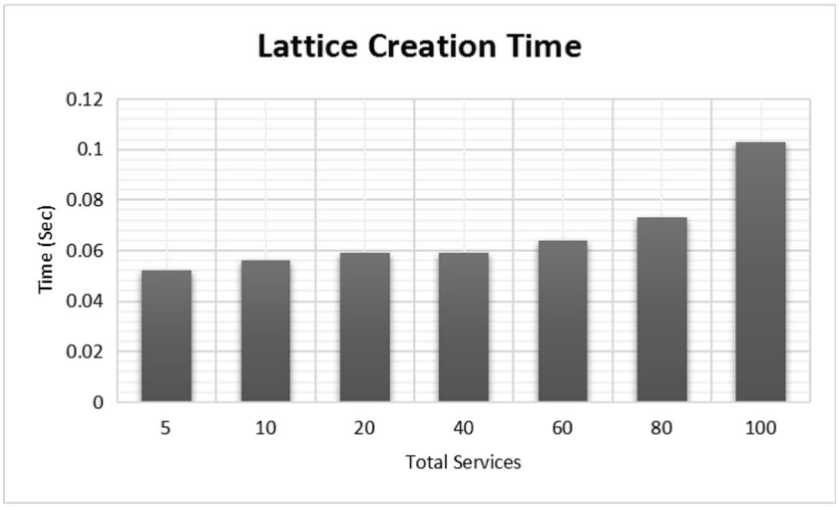


(b)

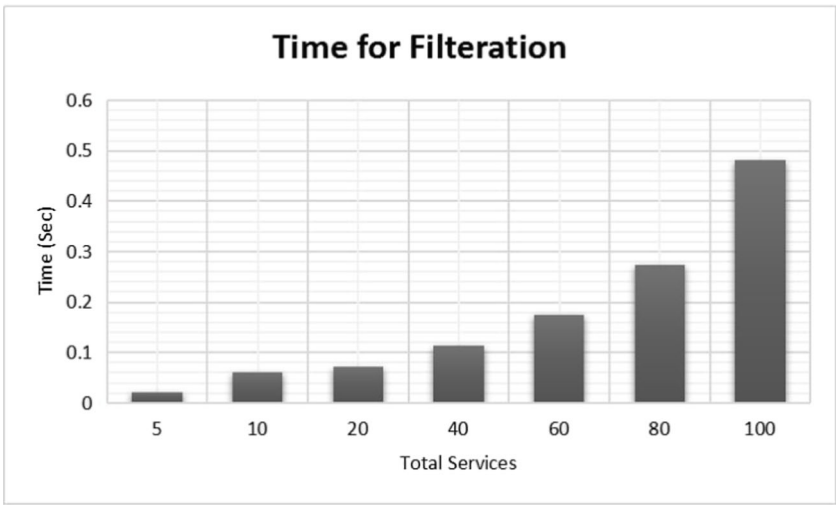
Fig. 8 Comparison of RL Algorithms. **a** RL Algorithm performance Analysis - Reward & Time. **b** RL Algorithm performance Analysis - Episode & Iteration



(a)

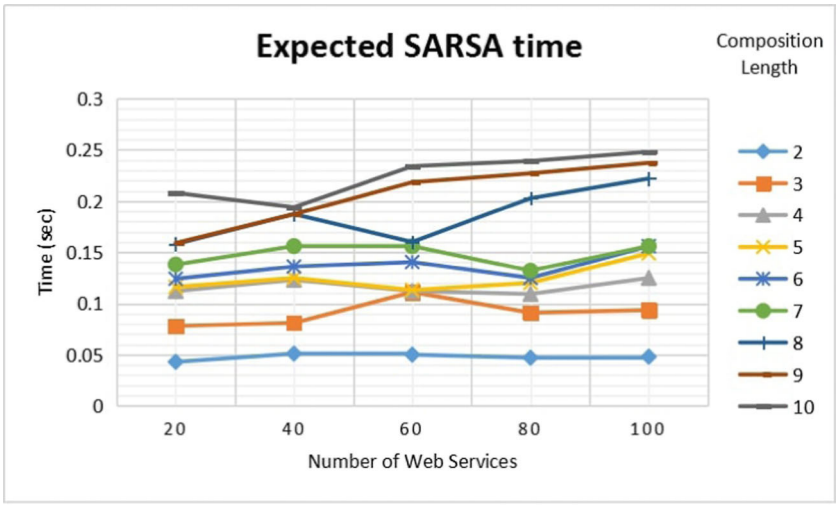


(b)

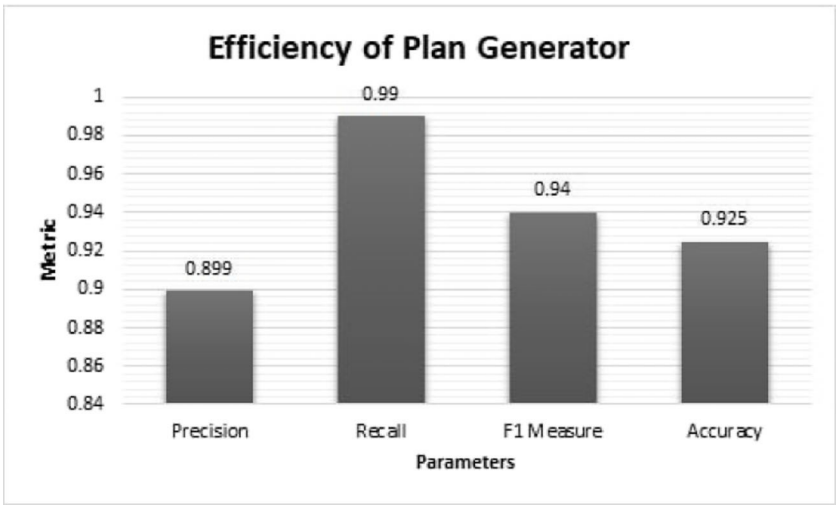


(c)

Fig. 9 Evaluation of Filteration Engine. **a** Efficiency of Filteration Engine. **b** Time duration for creation of Concept Lattice. **c** Time duration for ltration of services



(a)



(b)

Fig. 10 Evaluation of Plan Generator. **a** Execution Time of Plan Generator. **b** Efficiency of Plan Generator

Balanced values of Accuracy and the F1-Measure proves the reliability of the component.

5.3.3 Verification and validation of the plan generated

The authors of [19] have conducted an extensive survey on different state of art verification methods. They portray that among various verification methods available, model checking is the most popular due to its agility and robustness. The authors of [19] also portray that, NuSMV is the most popular tool used by the recent researchers to verify the plan generated. Hence, this article utilizes the tool named NuSMV for verification purpose.

The generated plan is said to be valid if the following conditions are satisfied,

1. Every service which is a part of the composed plan must be reachable.
2. The plan must be free from deadlocks.
3. Every plan generated must satisfy the global requirement as follows, "Every plan generated must consume the input given by the user and the output obtained must satisfy the user query".

The NuSMV tool is used to check all of these three constraints. Figure 11 illustrates the validation of a sample plan of composition length 5. The global requirement of the composed plan is formulated using a simple rule which checks whether the user input is consumed by the initial service and the output obtained on execution is similar to the user requested output. This formulation is carried out manually for all the 100 queries and it is found out that, 95% of the plan generated is Reachable, Deadlock Free and satisfies the Global Requirement.

All the generated plans were subjected to user satisfaction test. A group of 55 peoples were subjected to the

Fig. 11 Verification of the generated plan using NuSMV

```
D:\composition\composition verification\Software\NuSMV-2.5.4-i386-pc-mingw32\bin
>NuSMV -int
*** This is NuSMV 2.5.4 (compiled on Fri Oct 28 14:06:40 UTC 2011)
*** Enabled addons are: compass
*** For more information on NuSMV see <http://nusmv.fbk.eu>
*** or email to <nusmv-users@list.fbk.eu>.
*** Please report bugs to <nusmv-users@fbk.eu>

*** Copyright (c) 2010, Fondazione Bruno Kessler

*** This version of NuSMV is linked to the CUDD library version 2.4.1
*** Copyright (c) 1995-2004, Regents of the University of Colorado

*** This version of NuSMV is linked to the MiniSat SAT solver.
*** See http://www.cs.chalmers.se/Cs/Research/FormalMethods/MiniSat
*** Copyright (c) 2003-2005, Niklas Een, Niklas Sorensson

NuSMV > read_model -i plan1.smv
NuSMV > flatten_hierarchy
NuSMV > encode_variables
NuSMV > build_model
NuSMV > print_reachable_states
###############################################################################
system diameter: 6
reachable states: 6 (2^2.58496) out of 6 (2^2.58496)
###############################################################################
NuSMV > check_fsm

############################################################
The transition relation is total: No deadlock state exists
############################################################
NuSMV > check_ltlspec -p "G (s0.i = s1.i & s0.o = s5.o)"
-- specification  G (s0.i = s1.i & s0.o = s5.o)  is true
NuSMV > _
```

test result. It was found that, all the successfully verified plans were executed online without flaws which led to a user satisfaction rate greater than 85%. This level of user satisfaction was measured using a questionnaire with the following questions.

1. Did the execution of web services provide satisfactory results?
2. Would you prefer to use the same set of web services in the near future to accomplish similar tasks?
3. Please elaborate on the problems faced during the service execution.

The first 2 questions were evaluated for 5 points and a median of the result obtained was utilized in the calculation of User Satisfaction Rate ($UserSatisfactionRate = \left(\frac{V1+V2}{10}\right) * 100$ where V1 and V2 denotes the median value obtained for Question 1 and 2 respectively). The question 3 was utilized to point out the flaws that occurred during the execution. It was observed that, the successfully verified services had no flaws.

5.3.4 Overall performance of the RLAP intelligent framework

AI techniques can be classified into 3 major categories namely,

- Planning Techniques
- Learning Techniques
- Search and Optimization Techniques

The proposed RLAP Intelligent Framework is categorized under AI based Learning Techniques. Expected SARSA is found to be the best under various compared Learning strategies as shown in Section 5.2. So, the proposed framework is now compared across other AI techniques namely the Planning and Optimization techniques. The results are sampled using 100 queries over 10 domains.The RLAP Intelligent Framework operates with an Accuracy of 95%, Precision of 95%, Recall of 98% and F1-Measure of 96%. It is evident that, the Framework performs extremely well with balanced Precision and Recall.

The overall efficiency of the RLAP Intelligent Framework in comparison with AI Planning is shown in Fig. 12. The results of Graph based composition [17], Fluent Calculus based composition [20] and IDECSE [3] are taken in analogy to the RLAP Intelligent Framework. The Graph based Composition has an Accuracy of 91%, Precision of 100%, Recall of 89% and F1-Measure of 94%. The Fluent Calculus based Composition has an Accuracy of 92%, Precision of 90%, Recall of 99% and F1-Measure of 94%. The IDECSE Framework has an Accuracy of 92%, Precision of 92%, Recall of 93% and F1-Measure of 92%. The execution time of RLAP Intelligent Framework, Graph based Composition, Fluent Calculus based Composition and IDECSE Framework for composing a plan of length 10 for 100 web services is around 0.73 seconds, 1.2 seconds, 0.69 seconds and 1.45 seconds respectively. It can be well noted that the proposed RLAP Intelligent Framework has an improved accuracy over the other AI Planning methods. Also, the balance between the Precision and Recall is well balanced when compared with other methods. From the results it is seen that FC based composition [20] has the lowest execution time. But the proposed framework has a well balanced Accuracy, F1 - Measure with acceptable execution

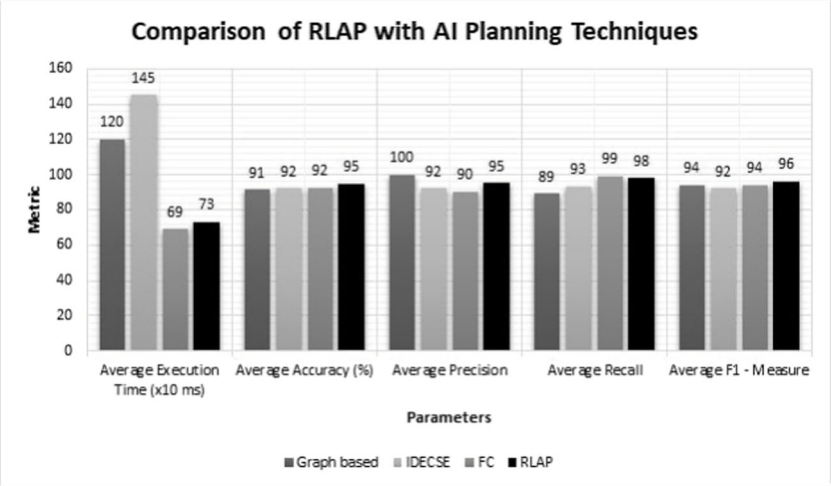


Fig. 12 Comparison of RLAP with state of the art AI planning techniques

time. Hence, the RLAP Intelligent Framework establishes its superiority over the compared AI Planning Techniques.

Similarly the efficiency of the RLAP Intelligent Framework in comparison with Optimization Techniques is shown in Fig. 13. The EMOABC Algorithm [9] has an execution time of 1.3 seconds and converges at 766 iterations with an accuracy of 94.2%. The modified GA technique [10] has an execution time of 0.92 seconds and converges at 714 iterations with an accuracy of 92.3%. The proposed RLAP Framework is found to converge at 489 complete iterations. From the obtained results, it is evident that the convergence of RLAP Framework is much faster when compared with other optimization techniques.

The execution time plays a vital role in the RLAP intelligent framework as every microsecond taken for composing a set of service has a direct impact on the user satisfaction level. Every user in today's world expect every process to be accomplished quicker with atmost accuracy and precision. So, the framework is said to efficient iff the execution time taken to compose the given set of web services is faster. Figure 14 illustrates the overall time efficiency of the proposed system with the literature techniques taken for study. From Fig. 14 it is evident that Fluent Calculus based Composition has a significantly lesser execution time. The RLAP proposed intelligent framework has relatively comparable execution time with FC method and reasonably lesser execution time when compared with the other methods taken for study. From Fig. 14 it is also evident that the maximum execution time taken by the RLAP framework to compose 1000 web services is less than 2 seconds which has a positive effect on user satisfaction level. The maximum execution time taken to compose 1000 web services for GA is around 2.5 seconds which also has a positive effect on user satisfaction level. But for the methods namely Graph based Composition, EMOABC method and IDECSE method the maximum execution time taken to compose 1000 web services are around 3.2 to 4 seconds which is found to have a negative effect on the user satisfaction level.

Also, from Figs. 12, 13 and 14 it is seen that the Average Accuracy, Average Precision, Average Recall and Average F1-Measure of the proposed RLAP Intelligent

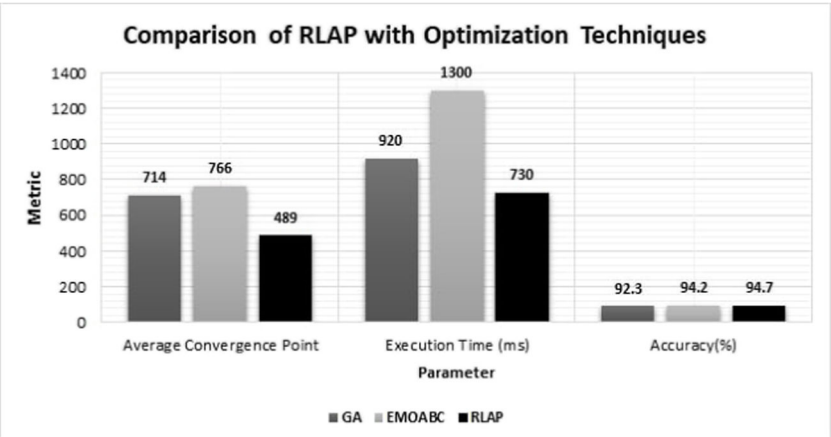


Fig. 13 Comparison of RLAP with state of the art optimization techniques

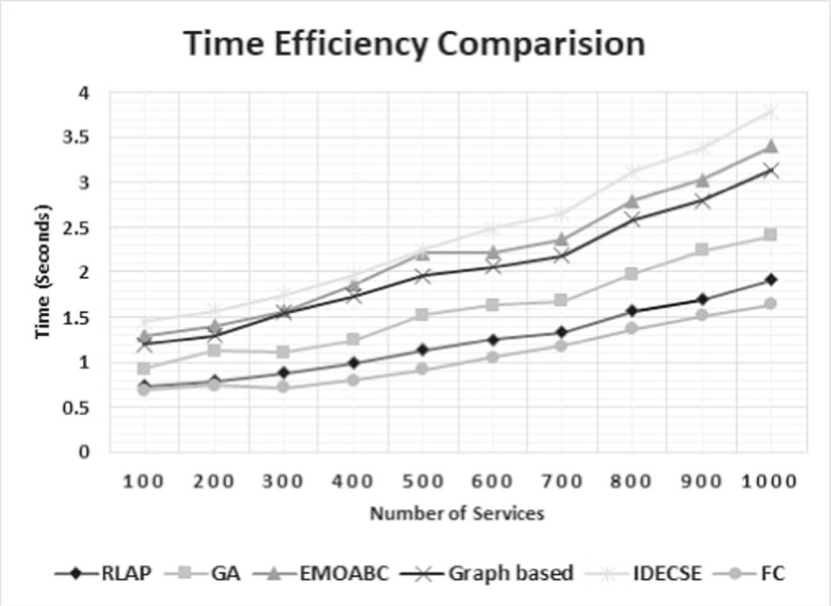


Fig. 14 Overall time efficiency comparison

Framework is better than the FC method with a better user satisfaction level. From the results obtained it is evident that the proposed work (RLAP) establishes its superiority over the solutions provided in the literature. Also Accuracy and F1-Measure of the proposed work are balanced which indicates that the RLAP intelligent framework is a well grounded design with improved performance.

6 Conclusion

A novel framework that utilizes the power of Formal Concept Analysis for Filteration of Alternate services and Reinforcement Learning for Composition of semantic web services is proposed as an auxiliary solution for the Semantic Web Service Challenge that prevails. An intensive evaluation is carried out to determine the most suitable Reinforcement Learning Algorithm for the proposed service model. The obtained results prove that, among the algorithms taken for study, Expected SARSA proves to be the best in terms of performance and reliability. Hence, the Expected SARSA is utilized in the proposed RLAP Intelligent Framework. The RLAP Intelligent Framework initially filters the services using Concept Lattice and the results obtained from the filtration phase proves that Concept Lattice is more efficient in filtering the Alternate services. Also the overall assessment of the RLAP Intelligent Framework indicates that both Accuracy and F1-Measure are well balanced making the framework more suitable for real time service composition. The comparison with the literature works imply that the efficiency of the proposed work is relatively higher making it more suitable for dynamic composition. It is concluded that, Reinforcement Learning based composition coupled with Formal Concept Analysis infused Filtration serves to be an excellent strategy to over come the Semantic Web Service Composition Challenge.

References

1. Collection of owl-s web services (2006). http://www.andreas-hess.info/projects/annotator/owl-ds.html
2. Collection of owls-slr web services (2009). http://lpis.csd.auth.gr/systems/OWLS-SLR/datasets.html
3. Abid A, Rouached M, Messai N (2020) Semantic web service composition using semantic similarity measures and formal concept analysis. Multimed Tools Appl 79(9):6569–6597
4. Amudhavel J, Prabu U, Inbavalli P, Moganarangan N, Ravishankar V, Baskaran R, Dhavachelvan P (2016) Survey and analysis of web service composition strategies: A state of art performance study. Indian J Sci Technol 9(11):1–10
5. Bekkouche A, Benslimane SM, Huchard M, Tibermacine C, Hadjila F, Merzoug M (2017) Qos-aware optimal and automated semantic web service composition with user's constraints. SOCA 11(2):183–201
6. Drupal: Project usage overview (2020). https://www.drupal.org/project/usage
7. Ganter B, Obiedkov S (2018) Conceptual exploration. Springer, Berlin
8. Ganter B, Wille R, Franzke C (2006) Formal concept analysis: mathematical foundations. Springer-Verlag, Berlin
9. Huo Y, Qiu P, Zhai J, Fan D, Peng H (2018) Multi-objective service composition model based on cost-effective optimization. Appl Intell 48(3):651–669
10. Kashyap N, Kumari AC, Chhikara R (2020) Service composition in iot using genetic algorithm and particle swarm optimization. Open Comput Sci 10(1):56–64
11. Lei Y, Jiantao Z, Fengqi W, Yongqiang G, Bo Y (2015) Web service composition based on reinforcement learning. In: 2015 IEEE International conference on web services, pp 731–734. IEEE

12. Liu ZZ, Chu DH, Jia ZP, Shen JQ, Wang L (2016) Two-stage approach for reliable dynamic web service composition. Knowl-Based Syst 97:123–143
13. Luis Felipe Cabrera CK (2005) Web Services Architecture and Its Specifications: Essentials for Understanding WS-*. Pro-Developer. Microsoft Press
14. Redbooks I (2004) Patterns: Service Oriented Architecture And Web Services Ibm
15. Ren L, Wang W, Xu H (2017) A reinforcement learning method for constraint-satisfied services composition. IEEE Transactions on Services Computing
16. Sutton RS, Barto AG (2018) Reinforcement Learning: An Introduction, 2 edn. Adaptive Computation and Machine Learning, The MIT Press
17. Rodriguez-Mier P, Pedrinaci C, Lama M, Mucientes M (2015) An integrated semantic web service discovery and composition framework. IEEE Trans Serv Comput 9(4):537–550
18. Ruby LRS (2007) RESTful web services O'Reilly
19. Souri A, Rahmani AM, Jafari Navimipour N (2018) Formal verification approaches in the web service composition: a comprehensive analysis of the current challenges for future research. Int J Commun Syst 31(17):e3808
20. Swetha NGG (2017) Web service composition using fluent calculus and verification using casual link matrix. In: National conference on ambient intelligent and smart environments, pp 39–44. PSG CT
21. Uc-Cetina V, Moo-Mena F, Hernandez-Ucan R (2015) Composition of web services using markov decision processes and dynamic programming. Sci World J 2015
22. UDDI: Universal description, discovery, and integration (uddi) (2020). http://xml.coverpages.org/uddi.html
23. Wang H, Huang G, Yu Q (2016) Automatic hierarchical reinforcement learning for efficient large-scale service composition. In: 2016 IEEE international conference on web services (ICWS), pp 57–64. IEEE
24. Wang H, Li J, Yu Q, Hong T, Yan J, Zhao W (2020) Integrating recurrent neural networks and reinforcement learning for dynamic service composition. Futur Gener Comput Syst 107:551–563
25. Wang H, Wang X, Hu X, Zhang X, Gu M (2016) A multi-agent reinforcement learning approach to dynamic service composition. Inform Sci 363:96–119
26. Yu X, Ye C, Li B, Zhou H, Huang M (2020) A deep q-learning network for dynamic constraint-satisfied service composition. Int J Web Serv Res (IJWSR) 17(4):55–75

N. G. Swetha currently working as Assistant Professor in the Department of Computer Science and Engineering, PSG College of Technology. She has completed her Master degree at PSG College of Technology, Coimbatore in 2017. She completed her Bachelor degree at Sri Krishna College of Technology, Coimbatore in 2015. She is currently pursuing Ph. D under Anna University. Her areas of research interest include Web Service Composition, Artificial Intelligence and Multi Criteria Decision Making Problems.

Dr G. R. Karpagam currently is a Professor and Associate Head with 23 years of experience in Computer Science and Engineering at PSG College of Technology, Coimbatore. She obtained her B.E, M.E, Ph.D in Computer Science and Engineering. Her areas of specialization include Database Management System, Data Structures and Algorithms, Service Oriented Architecture, Cloud Computing, Artificial Intelligence and Security.