4

Web Service Composition Using an AI Planning Technique

Lalit Purohit[1]* and Satyendra Singh Chouhan[2]

[1]Shri G. S. Institute of Tech & Science, Indore, India
[2]Dept. of Computer Engineering, Malviya National Institute of Technology (MNIT), Jaipur, India

Abstract

The concept of web service has become an emerging paradigm with the fast development of the Internet and number of applications increasing over the Web. It provides another way to distributed computing by the interoperability between the heterogeneous applications regardless of the implementation language and platform. Each web service offers functionality of its own. If a single service is not sufficient to offer the required functionality, multiple web services offering different functionalities are threaded together to offer desired functionality by the end user. With the increase in growth of the web services and web service based applications from various organizations, clustering of web services is desired to enable efficient web services searching. This leads to improving the performance of the overall composition process. The composition of services involves efficient and effective arrangement of services to complete the given customized tasks as per the needs of the end user. Nowadays, user's needs are changing from time to time. To deal with the dynamically changing requirements from the end user, a need of dynamic composition of web services arises. In this work, the clustering process has been employed for reduction in the search space and AI planning based technique is used to deal with the problem of dynamic composition of web services. To achieve dynamic and automatic composition, a plan based on the user's request is to be generated that describes the execution sequence of web services. The AI planning is helpful in achieving this objective. Planning is defined as a process of finding a sequence of actions (plan) such that if an agent performs the plan on the given initial state, it will achieve the goal state. By considering web service

**Corresponding author*: purohit.lalit@gmail.com

Anamika Ahirwar, Piyush Kumar Shukla, Manish Shrivastava, Priti Maheshwary and Bhupesh Gour (eds.) Innovative Engineering with AI Applications, (65–82) © 2023 Scrivener Publishing LLC

composition as a planning problem, the web services are the task/functionality provided and the composition as a goal.

***Keywords*:** Artificial intelligence planning, web services, dynamic web service composition, clustering

4.1 Introduction

In the present scenario, a large number of service providers are offering Web Services over the Internet. The enterprise business applications are taking advantage of these services and offering specific services to the end users. The web services are software components with the property of loose coupling, modular, self-described, and are interoperable. Over the Web these components can be easily published, discovered, and invoked by other software components [1]. The applications available over the Web take the advantage of available web services to easily exchange the data with other web applications. This is primarily due to the basic nature of web services of being programming language and platform independent. This has motivated many enterprises to publish their business models as service(s) and can be accessed over the Web [2]. The web services are designed with a goal to offer a specific service to the end user. The primary advantage of web service is that they are reusable i.e. can be used multiple times by any software component. A web service offers limited functionality on its own. However, if a single web service is not able to fulfill the requirements of the application/end user, multiple such web services can be combined together. This process of aggregating various web services together to offer a value added service to the application is termed as Web Service Composition [3].

A large number of web services available over the Web. For business-to-business interaction, the potential of web services can only be realized when many services and businesses are composed together and executed to form a value added composite service [3]. It also enables customers to compose services automatically and on demand. For example, in the travel domain, a tour planning web service can be obtained by integrating a book flight web service, hotel booking service, cab booking service, travel insurance service, payment gateway web service, etc., together.

Web service compositions are of two types – (I) Static composition (II) Dynamic composition. These two types of compositions are defined based on time of composition of web services. The composition is known as static when the task of composition is performed offline i.e. before the start of

当网络服务的执行顺序在运行时动态（自动）确定时，就实现了网络服务的动态组合。

execution of composite web service. The dynamic composition of web services is achieved when the order of web service execution is determined dynamically (at run time) and automatically. The dynamic composition is highly flexible and is most suitable for a continuously growing web repository [2]. However, the task of composing web services dynamically is a challenging task and this chapter focuses on this approach of composition.

In today's times, the Web environment has become dynamic in nature with the drastic increase in the number of services and constant updates. The user's goals might change during the run time due to these ongoing updates in the web service environment. Then, the composition should adapt to these changes and make decisions according to the up to date information. The task of effective and efficient arrangement of web services according to the changing requirements from the end user introduces the need for dynamic web service composition. Many solutions are available from existing state-of-the-art for web service composition. Figure 4.1 represents a broad categorization of various approaches used in the past to efficiently achieve web service composition.

The existing techniques of composition are aggregating the web services with a static approach [4]. In the static composition, a pre-evaluated composition plan is generated offline. Whenever the end user requests for

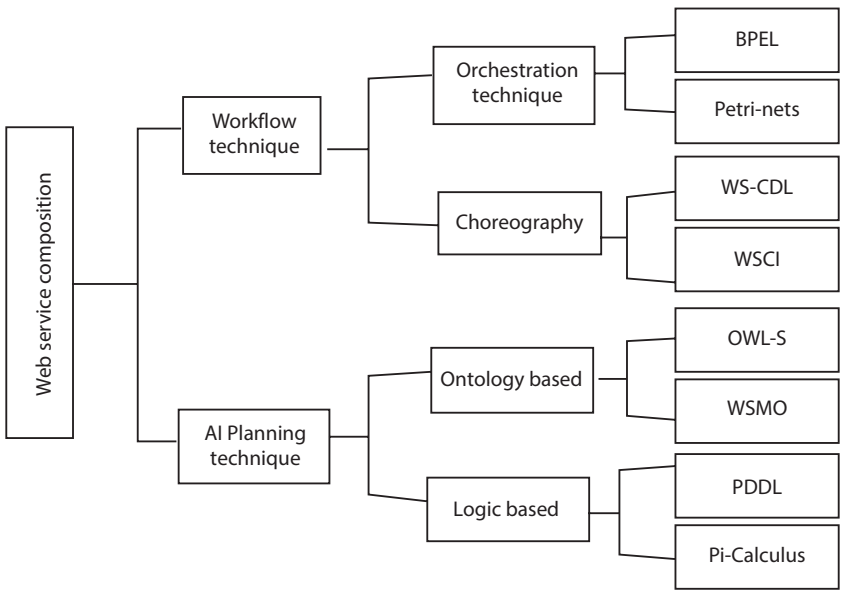


Figure 4.1 Categorization of web service composition techniques.

composite service, the pre-computed composition plan is executed. The composition plan execution remains the same for each end user request. But, many times there is a need to change the plan of composition as per the request of the end user. In such cases, web services are to be composed dynamically. The dynamic composition of web service offers flexibility to the end user in terms of generating output as per the input provided.

For the composition process, a huge repository of web services has to be searched for choosing the required web services. While finding the services relevant to the user's requested query, one needs to check each and every web service present in the huge search space. To deal with this huge task, a clustering approach is applied before applying the composition. The clustering approach help in reducing the workload by not matching each web service with the requested query [5–7]. The dynamic web service composition mainly involves arrangement or ordering of tasks automatically and dynamically to fulfill the user's goal. A plan of web services needs to be generated to achieve the required composite service as output. Thus, the problem of service composition can be modeled as a planning problem. 服务组合的问题可以被建模为一个规划问题。

The primary objective here is to obtain the composition which is able to adapt itself with modifications in customer's requirement and with minimal user intervention. Since, static approaches are suitable for the process in which the business partners are fixed and the functionalities are unlikely to get modified in the short term. Therefore, to overcome this problem we perform the dynamic composition of web services. The clustering technique used in the proposed approach improves the performance of the composition process. This can be observed by the fact that the web services are large in number and on applying clustering, the search of web services for composition tasks is performed on the clusters and not on each and every web service. The planning task involved in the composition process also aims to generate the concurrent plan but not a predefined sequence of tasks [8, 9].

In this chapter, we discuss web service composition using AI planning techniques. First, we present the background information that contains formal definitions and planning related terminologies. It also discusses the service composition as AI planning with a suitable example (Section 4.2). In Section 4.3, we discuss the methodology for web service composition using AI planning. Next we present the performance evaluation of the proposed methodology with respect to a case study in Section 4.4. Conclusions and possible future directions are given in Section 4.5.

4.2 Background

This Section gives a brief introduction to Artificial Intelligence (AI), AI planning, formal definitions related to AI planning and formal introduction of web service composition problems as AI planning problems.

4.2.1 Introduction to AI

Artificial Intelligence (AI) is the study of automated machines or devices with the capability of automatic adoption with respect to changing environments. The machines or devices can discern the external environment and take more effective actions to maximize the chances of successfully achieving the goal. A typical AI based system is shown in Figure 4.2. Any typical AI system has a rule base to take action. The feedback obtained by accessing external environment data is used by AI systems to improve the rule base. This further improves the action taken and decisions made by the AI system [10].

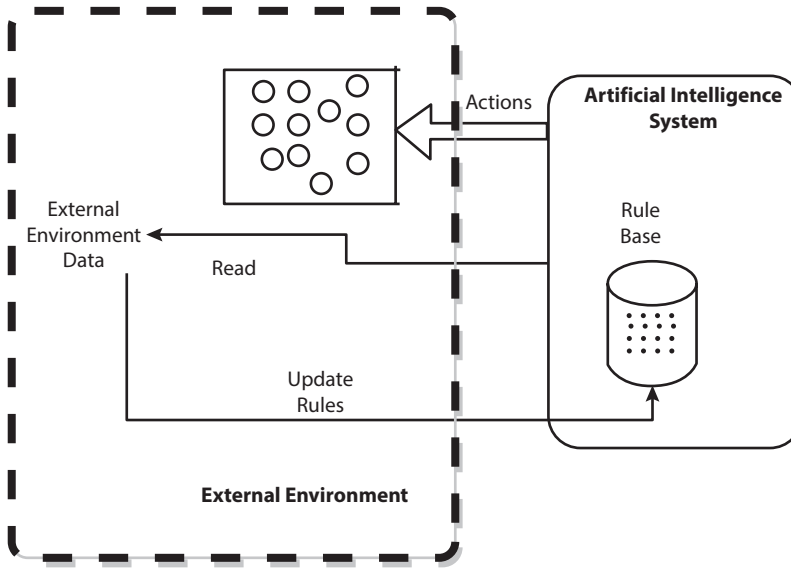


Figure 4.2 Artificial intelligence system.

4.2.2 AI Planning

AI Agents are robots or automatic programs with the capability of interacting with each other and other resources over the Web. The way an artificial agent acts in an environment is called its behavior. The reasoning side of action is known as planning and it is an important ingredient of rational behavior of an agent [9]. The planning problem in AI is considered as decision making accomplished by an intelligent agent [11, 12]. Agents are usually robots or computer programs which are trying to achieve some goal. Planning comes before the acting and in a fully observable environment planning can be done offline and beforehand. Planning is one of the key research areas in AI. The inputs to a planning problem are: Planning has found various real-world applications in robotics [13], games [14], logistics [15], search and exploration, military operations, and web services [15]. The notations and formal definitions are as follows.

物流 勘探

Definition 1 (*Plan*) Given a set of action '*A*', a plan is represented by a sequence of actions of '*A*'. Formally, if '*P*' represents the plan then where each $a_i \in A$, is a transition from a state in *S* to any other state in *S*, for any $s \in S$. Moreover, for, P=*null*, and otherwise. P holding the *null* value represents the empty plan.

Definition 2 (Action) An action "α" can be defined as a mapping from and '*A*' represents the set of actions by any agent such that $a \in A$. An agent can build a plan from the set of actions '*A*'. Each of the action has associated precondition (P) and effect (E). Preconditions are necessary conditions for executing an action. Effects are the conditions (facts/fluents) that will be true after executing the action.

Definition 3 (*Planning problem*) In AI domain, a planning problem is represented as - where set of fluents (also called predicates) is symbolized as *F*, *I* is the initial state (a subset of predicates that are true), set of actions *are denoted by* '*A*', and goal state *as* '*G*' (a subset of predicates that are desired to be true).

在人工智能领域，规划问题表示为 - 其中流动性的集合（也称为谓词）用 F 表示，I 是初始状态（是真的谓词的子集），动作的集合用 'A' 表示，目标状态用 'G' 表示（是希望为真的谓词的子集）。

4.2.3 AI Planning for Effective Composition of Web Services

The Web Service composition problem is considered as a planning problem. The dynamic composition method is required for generating the plan automatically. Such methods can be accomplished using AI planning. AI planning and service composition is much similar. The problem is to determine a flow of actions which leads to the desired goal upon execution and this is common in both. AI planning focuses on the preconditions and effects of the actions and the same goes with service composition [8].

在应用组合之前，首先应用网络服务聚类，因为它有助于在组合过程之前进行必要的预处理

Before applying composition, web service clustering is applied as it helps in pre-processing required before the composition process. Clustering Technique is applied for reducing the search space while searching for web services which are needed for composition [16–18]. It helps in minimizing the total number of services available for finding the services relevant to a user's query. A suitable example is given below.

Example 1. (*The traveling domain* [19]) A researcher is required to travel to another city to present a research paper in an international conference. She needs to plan a tour using a web service based system which satisfies all the constraints such as cost, time, dates, preference of a particular hotel type, airlines, etc. For this purpose, various web services from the travel domains are available such as air tickets booking, accommodation service, cab booking service, insurance service, etc. These web services can be integrated/composed together in such a manner that all the end user constraints are also satisfied by the tour planning web service. This can be done by modeling web service composition problems as an AI planning problem. In this case, for the given request from the application/end user, '*I*' represents the initial states and '*G*' denotes the goal states. The available services are represented as the set '*A*'.

4.3 Proposed Methodology for AI Planning-Based Composition of Web Services

The proposed approach for achieving web service composition using AI planning is presented in Figure 4.3. The actor is the end user/application program to request service. Candidate web services are assumed to be available and are accessible from the repository. Each step in the proposed approach is discussed in detail.

4.3.1 Clustering Web Services

The task of grouping the objects such that the more similar ones are in the same group (i.e. cluster) and less similar ones are in other groups. Clustering of web services can be performed by using QoS parameters. For clustering web services, Partition based Clustering (such as K-means and Self Organizing Feature Maps (SOM)), and Model based Clustering are popular techniques. In this work, K-Means clustering is used to perform clustering of web services. K-means needs the cluster number (K) and the selection of centroids [7]. The points (Web services in this case) which are nearest to the centroids are placed in the same cluster. To measure the

K均值需要聚类数（K）和质心的选择[7]。与质心最近的点（在这种情况下是网络服务）被放置在同一簇中。

"Centroids" 是指质心，在K均值聚类等算法中，质心是每个簇的中心点，它代表着该簇内所有数据点的平均位置

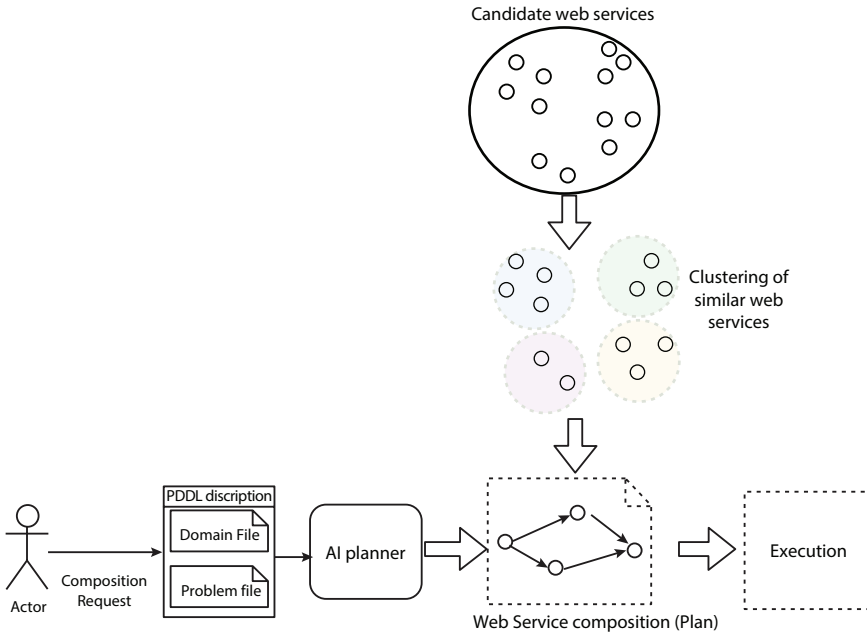


Figure 4.3 Overview of the proposed solution approach.

distance between points and centroids, Euclidean distance is used. The web services in multi-dimensional space are represented using various associated QoS parameters. The Euclidean distance among web services is obtained by using QoS parameters. Based on the Euclidean distance values and '*K*', clusters of web services are formed [5].

The web services are organized in such a way that web services with similar QoS parameter values are in a cluster. When the request of the desired web service from the end user is made, the web services with similar QoS parameter values can be found in a cluster. With this, the web services can be accessed quickly. As the number of web services is growing rapidly in the repositories, it becomes a challenge to find a web service of interest. Clustering helps in categorizing the web services which allow consumers to find related services easily.

4.3.2 OWL-S: Semantic Markup for Web Services *(For Composition Request)*

OWL-S is the ontologically descriptive language to descriptive semantic web services. It was developed to describe services using Web Ontology

OWL-S（Ontology Web Language for Services）是用于描述语义网络服务的本体描述语言。

https://zh.m.wikipedia.org/wiki/OWL-S OWL-S使得能够自动发现、组合、调用和监视特定的网络服务提供的服务。

Language (OWL). The OWL-S enables automatic discovery, compose, invoke and monitor a particular web service offering service. The web services can be described in a way amenable to planning. It consists of a service profile, process model and groundings. It describes the inputs, outputs, preconditions, and effects. Inputs are input needed by the web service, outputs are output produced by the web service, and Preconditions are the logical conditions which should be satisfied before the service begins its execution. Effects are the changes brought in the database/files/associated resource/etc., after the execution of the web service. In OWL-S, operations can be represented as processes. The web services are either atomic process, composite process or simple process. In this work an AI planner is used for web service composition. OWL-S representation is transformed into a PDDL (Planning Domain Description Language) description. PDDL is widely used as an input description language for AI planners.

4.3.3 PDDL: Planning Domain Description Language

为了正式描述一个规划问题，PDDL是标准的编码语言。几乎每个规划器都支持它。使用PDDL来指定规划领域和问题，并将其作为输入提供给规划器。

To formally describe a planning problem, PDDL is the standard encoding language. It is supported by almost every planner. Planning domain and problem are specified using PDDL and given as input to the planner. Component of PDDL language are;

- Objects: Things in the domain which will be used.
- Predicates: Properties of objects that can be false or true.
- Initial state
- Goal state
- Actions/Operators: through which we can change the state of the system.

PDDL is mainly based on the STRIPS [20] syntax and also represents the actions, using pre-conditions, post-conditions and the effects. Here, the action's applicability is represented using pre-conditions i.e., a particular action is applicable on a state if and only if preconditions are satisfied.

Planning task is generally described in two separate PDDL files.

在这里，动作的适用性是使用前提条件表示的，即当且仅当前提条件得到满足时，特定动作对状态才适用。

1. A domain file for specifying predicate and actions
2. A problem file for specifying objects, initial state, and goal state

The separation of domain and problem description is useful because we can have multiple problem files with the same domain file. The domain file contains the types of objects, describes a set of predicates and actions

each has some parameters, a pre-condition (P) and effect (E). From the web service composition perspective, each action can be assumed to be a web service. Thus, domain files consist of all the web services offered in the domain. The problem files consist of the initial state, user's query, and the goal state of the problem.

Given below an example of domain and problem definition:

Domain	Problem
(define (domain travel) (:requirements :strips) (:predicates (road ?from ?to) (person ?p) (vehicle ?v) (at ?thing ?place) (travel ?p ?v)) (:action goto :parameters (?p ?from ?to) :precondition (and (road ?from ?to) (at ?p ?from) (not (?from ?to)) (person ?p)) :effect (and (at ?p ?to) (not (at ?p ?from))))	(define (problem travel-1)) (:domain travel) (:objects p1-person home oce- location car bus- vehicle) (:init (at p1 home) (road home office)) (:goal (at p1 office))

In the above example, "road" is a route from one place(?from) to another(?to). The predicate "at" tells that something (?thing) is present on place(?place). Other predicates like "person" is for any person and "vehicle" is for car, bike etc. The predicate "travel" shows that a person(?p) travels by vehicle(?v). The action "goto" means that a person needs to go from a place(?from) to another(?to). The problem file consists of a set of objects and description of initial and goal states. The propositions listed in the initial state description are true and the ones which are not present are assumed as false. The goal is a logical expression similar to the precondition and it is a conjunction of positive propositions.

初始状态描述中列出的命题为真，而未列出的命题被假定为假。

In the problem part (example), "p1", "home", "oce", "car", "bus" are all objects of the problem file. In the initial state, person 'p1' is present at home and there is a road from home to office. The goal of the problem is that person 'p1' is at the office.

4.3.4 AI Planner

The presented methodology uses Blackbox AI planning system for planning [21]. The blackbox planner uses two approaches: planning as satisfiability and planning as graph. First, Blackbox converts the web service composition problem, described in PDDL, into a Boolean satisfiability problem. Next, it solves using various satisfiability approaches that are specified during execution. The front-end of the planner uses the graph plan system [22]. The Graph plan generates a planning graph. Graph plan is a compact structure where the encoding of all possible plans is done up to a predefined length. Next, the search process is guided by planning graph. Planning graph approach take a mid-path in between state-space search based and partial order planning (POP) techniques and explore the search space defined by the planning graph. The planning graph structure has been an influencing approach in state-based heuristic planning. In satisfiability based approach, a planning problem is converted into a propositional satisfiability (SAT) problem. The problem is represented as a propositional logic formula and then gives as an input to a SAT solver. SAT solver solves it by determining that there existing a model for the propositional logic formula or not.

图规划是一种紧凑的结构，其中对所有可能的计划的编码都是在预定义的长度上完成的。

规划图方法在基于状态空间搜索和部分顺序规划（POP）技术之间取得了折中

在基于满足性的方法中，规划问题被转换为命题满足性（SAT）问题

SAT求解器通过确定命题逻辑公式是否存在模型来解决问题

Blackbox planner provides a lot of flexibility in such a way that we can use WALKSET for 30 seconds and if it fails, then can use SATZ for 900 seconds. Due to this, it is capable of solving a wide range of problems efficiently.

Mapping to Web Service Composition. The plan generated by the Blackbox planner is used and web services are mapped according to that plan. Hence, the web service composition is achieved.

4.3.5 Flowchart of Proposed Approach

The process of composition starts with the input of WSDL documents. These documents are parsed so as to obtain the feature that is 'web service name' for further process. In the next step the feature is used for the purpose of similarity calculation of web services. With the help of this calculation, a similarity score gets generated. K-means clustering techniques are applied on the web services similarity scores. After this, the web services are chosen for the process of composing the PDDL files. These PDDL files are fed as input to the AI planner and in the end the plan gets generated. The plan is then mapped to the web services and the process of composition gets completed. Figure 4.4 shows the flowchart of the proposed approach.

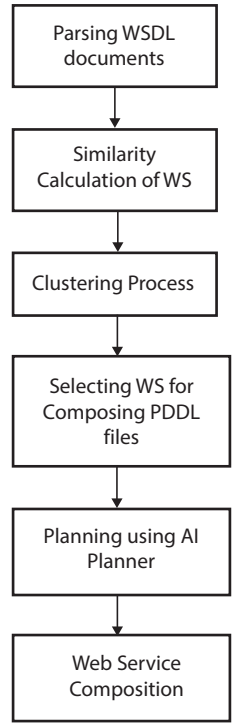


Figure 4.4 Flowchart.

4.4 Implementation Details

The implementation work is performed in various steps which include parsing of WSDL files, similarity calculation between web services, applying clustering process over the web services, PDDL les generation, plan generation and lastly the composition of web services. The presented approach is implemented using JAVA language. All the experiments were performed using Intel Core i7-6300 2.7 GHz with 8 GB RAM. Blackbox planner is used for planning purposes.

4.4.1 Domain Used

We use the travel domain as a case study for the implementation of the presented approach. The domain and problem description is as given below. It is defined in terms of different entities such as person, locations etc. The example travel domain has several locations such as restaurant, ATM,

airport, national park, hotel, beach, and researcher's home. The domain has the following actions:

-*gotohotel(p,x,y)* : person 'p' goes from location 'x'to hotel 'y'

-*gotohome(p,x,y)* : 'p'goes from any location 'x'to researcher's home 'y'

-*gotorestro(p,x,y)* :'p' goes from any location 'x' to restaurant 'y'

-*gotoatm(p,x,y)* : 'p' goes from any location 'x' to atm 'y'

-*gotobeach(p,x,y)* :'p' goes from any location 'x'to hotel 'y'

-*gotopark(p,x,y)* :'p' goes from any location 'x' to hotel 'y'

-*orderfood(p,x)* : 'p' orders food at location 'x'

-*withdrawmoney(p,x)* : person 'p' withdraws money at location 'x'

Based on the above description of the domain, problem instances described in the next section are used.

4.4.2 Case Studies on AI Planning

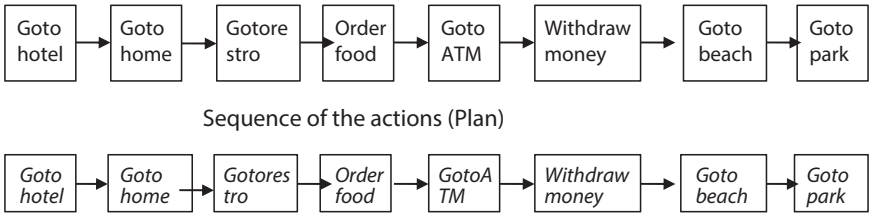
In this section, two case studies are described which leads to two different execution sequences.

Case Study 1

A student P (referred as person) is conducting research work. She is willing to visit another city to meet another researcher. The visit is to be planned so that she can explore the knowledge and research field by interacting with the researcher by duly considering the mentioned constraints. Following sequence of actions (inputs) are considered to plan a tour.

The visit to another city is to be planned by flight. From the airport of the destination city, she wants to go to the hotel of her choice using a cab. After reaching the hotel, she will immediately get ready to visit the researcher at home by 11 AM. By 1:00 PM, her meeting with the researcher will be finished. She will leave the researcher's home to have lunch at the restaurant. Her return flight is scheduled at 11 PM on the same day, thus, for the remaining time she would like to explore various tourist spots across the city. To visit various spots of interest around the city, firstly an ATM close

to the restaurant for withdrawing some money is to be visited. Since she likes water-sports, a nearby beach is also to be visited. A famous national park in the city for sightseeing is also to be visited at last, in the evening. By observing the available inputs, the initial states and goal states, a possible plan is generated as follows.



Similarly case 2 with different scenario can be taken as:

Case Study 2: A student (referred as person P) is doing research work. She wants to visit another city to meet another researcher. The visit is to be planned so that she can explore the knowledge and research field by interacting with the researcher by duly considering the mentioned constraints. Following sequence of actions (inputs) are considered to plan a tour.

The visit to another city is to be planned by flight. From the airport of the destination city, she wants to go to the hotel of her choice using a cab. Her meeting with the researcher is scheduled in the evening at 8:30 P.M. Meanwhile, after reaching the hotel at 9:00 AM, she decided to visit various famous places of interest across the city after getting ready. To visit various places of interest around the city, firstly an ATM close to the restaurant for withdrawing some money is to be visited. Since she likes water sports, a nearby beach is also to be visited. A famous national park in the city with a beautiful sun set view is also to be visited at last, in the evening. Afterwards, she visited the researcher at home by 7:00PM. Afterwards, she went to the restaurant to have dinner.

4.4.2.1 Experiments and Results on Case 1 and Case 2

For experimental purposes, we changed the requirements of the user in each case by changing the initial state, goal states and the actions to be performed during the traveling of the person. Also, in each case, we have experimented by increasing the number of each instance in the problem. For example, the problem instance is defined as (P, A, H, RH, R, AL, B, NP) where Ꝑ denotes person, *A* is airport, *H* is hotel, *RH* is home of researcher,

R represents restaurant, *AL* is the location of ATM, *B* is beach and *NP* is national park. While performing the experiments, we have increased the number of instances for each of *P, A, H, RH, R, AL, B,* and *NP*. The problem instances are denoted by 1 to 6. For each of the problem instances, the experiments are repeatedly performed *'n'* times and reported the average results for each metrics. The results show the various performance metrics of Blackbox planner such as: planning time, total elapsed time, Nodes created (in the graph created by Blackbox planner), No. of grounded action variables, Total no. of variables, no of actions in the plan.

The results of experiments performed by using Blackbox for the Travel domain are presented in Figure 4.5. It can be observed from the experiment that the numbers of objects are increasing with the increasing number of combinations. In order to obtain the results, each combination on the planner is executed 20 times and then averaged. In Figure 4.5, the first figure shows the average time taken for each case. Second figure represents the total number of nodes created by Blackbox planning in the planning graph. This metric is associated with the size of the graph and it is directly proportional to memory use in the system. Fourth figure shows the number

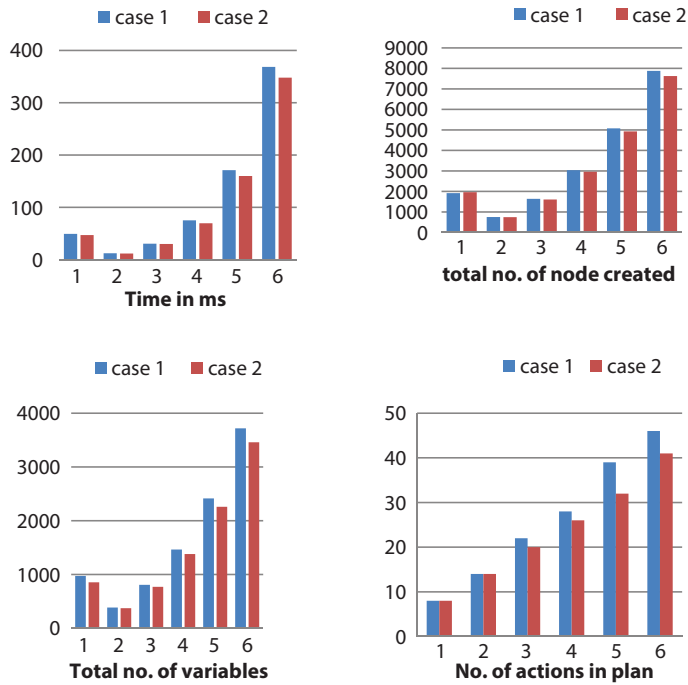


Figure 4.5 Experimental results of Blackbox planner with respect to various metrics.

of actions in a plan generated by the Blackbox planner. In summary we can observe that by increasing the number of instances such that *A, H, RH, R, AL, B,* and *NP* the complexity of the proposed approach increases. Moreover, from the experimental analysis we observe that the proposed approach is effective in case of small size problems.

4.5 Conclusions and Future Directions

In this chapter, we reviewed the research being done previously in dynamic web service composition and after knowing the various techniques and work done. We performed the project in two steps: first to have an effective and efficient selection of web services based on the user's query, and a concept of clustering is applied in this step. Clustering helps in reducing the efforts required to search for the web services in the huge repository. K-means Clustering is used to prune the dataset and reduce the search space of the repository. After getting the clusters we created the scenario according to the user's query and in the next step, we performed the overall web service composition. In the second phase we have performed experimental evaluation by using some domains available such as travel, food, geography and using them to obtain the PDDL domain and problem files. For an example we took a travel scenario domain description and created the PDDL files of this domain problem. We worked on various problem instances of the different case studies. We attempted to work on Blackbox planner which mainly uses planning as satisfiability and planning as graph. The PDDL domain and problem files were given as input to the Blackbox planner. We also noted the resultant plan generated and various other variations in the output when given different problem instances. In future we would like to see the applicability of other state-of-the-art planners for dynamic web service decomposition.

References

1. Hwang, S.Y., Hsu, C.C., Lee, C.H., Service selection for web services with probabilistic QoS. *IEEE Trans. Serv. Comput.*, 8, 3, 467–480, 2015.
2. Cui, L., Li, J., Zheng, Y., A dynamic web service composition method based on viterbi algorithm. *IEEE 19th International Conference on Web Services*, 2012.
3. Purohit, L. and Kumar, S., A classification based web service selection approach. *IEEE Trans. Serv. Comput.*, 14, 2, 315–328, 2021.

4. Lemos, A.L., Daniel, F., Benatallah, B., Web service composition: A survey of techniques and tools. *ACM Comput. Surv. (CSUR)*, 48, 3, 1–41, 20152015.
5. Ratnakar, A., Sharma, P., Gupta, S., Purohit, L., Web service clustering on the basis of QoS parameters. *2019 1st International Conference on Artificial Intelligence and Data Sciences (AiDAS)*, Ipoh, Perak, Malaysia, pp. 12–17, 2019.
6. Purohit, L. and Kumar, S., Clustering based approach for web service selection using skyline computation. *IEEE International Conference on Web Services (ICWS)*, Milan, Italy, pp. 260–264, 2019.
7. Kumar, S. and Purohit, L., *Exploring K-Means Clustering and Skyline for Web Service Selection*, pp. 603–607, 2016.
8. Kuzu, M. and Cicekli, N.K., Dynamic planning approach to automated web service composition. *Appl. Intell.*, 36, 1, 1–28, 2010.
9. Ghallab, M., Nau, D., Traverso, P., *Automated Planning: Theory and Practice*, Elsevier, Amsterdam, Netherlands, 2004.
10. Ertel, W., *Introduction to Artificial Intelligence*, 2nd ed., Springer, Cham, Switzerland, 2017.
11. Wooldridge, M., *An Introduction to Multiagent Systems*, John Wiley & Sons, West Sussex, England, 2009.
12. Chouhan, S.S. and Niyogi, R., Multi-agent planning with collaborative actions, in: *Australasian Joint Conference on Artificial Intelligence*, pp. 609–620, 2016.
13. Parker, L.E., Distributed intelligence: Overview of the field and its application in multi-robot systems. *J. Phys. Agents*, 2, 1, 5–14, 2008.
14. Brafman, R., II, Domshlak, C., Engel, Y., Tennenholtz, M., Planning games, in: *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 73–78, 2009.
15. Cirillo, M., Pecora, F., Andreasson, H., Uras, T., Koenig, S., Integrated motion planning and coordination for industrial vehicles, in: *Proceedings of the 24th International Conference on Automated Planning and Scheduling (ICAPS)*, vol. 2126, 2014.
16. Chouhan, S.S. and Niyogi, R., MAPJA: Multi-agent planning with joint actions. *Appl. Intell.*, 47, 4, 1044–1058, 2017.
17. Zheng, K., Xiong, H., Cui, Y., Chen, J., Han, L., User clustering based web service discovery, in: *Proceedings of International Conference on Internet Computing for Science and Engineering*, pp. 276–279, April 2012.
18. Peer, J., *Web Service Composition as AI Planning: A Survey*, pp. 1–63, University of St. Gallen, Switzerland, 2005.
19. Purohit, L. and Kumar, S., Web services in the internet of things and smart cities: A case study on classification techniques. *IEEE Consum. Electron. Mag.*, 8, 2, 39–43, March 2019.
20. Strips: A new approach to the application of theorem proving to problem solving. *Artif. Intell.*, 2, 3, 189–208, 1971.

21. Blackbox: A new approach to the application of theorem proving to problem solving, in: *Workshop on Planning as Combinatorial Search (AIPS98)*, vol. 58260, pp. 58–60, 1998.
22. Blum, A.L. and Furst, M.L., Fast planning through planning graph analysis. *Artif. Intell.*, 90, 1, 281–300, 1997.