Journal of Theoretical and Applied Information Technology
20th August 2015. Vol.78. No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

WEB SERVICES COMPOSITION FOR CONCURRENT PLAN USING ARTIFICIAL INTELLIGENCE PLANNING

[1]MUHAMMAD AKMAL REMLI, [2]SAFAAI DERIS, [3]MAMOUN JAMOUS, [4]MOHD SABERI MOHAMAD, [5]AFNIZANFAIZAL ABDULLAH
Software Engineering Department,
Faculty of Computing
Universiti Teknologi Malaysia,
81310 UTM Johor Bahru
Malaysia
E-mail: [1]akmalmuhd@gmail.com, [2]safaai@utm.my, [3]mamounjamous@gmail.com, [4]saberi@utm.my, [5]afnizanfaizal@utm.my

ABSTRACT

Automatic planning of web services composition is a challenging problem both in academia and real-world application. Artificial Intelligence (AI) planning can be applied to automate web services composition by depicting composition problem as AI planning problem. Web services composition would combine multiple services whenever some requirements cannot be fulfilled by a single service. Subsequently, many of the planning algorithms to detect and generate composition plan would focus only on sequence composition thus, neglecting concurrent composition. The aim of this paper is to develop an approach to generate a concurrent plan for web services composition based on semantic web services (OWL-S) and Hierarchical Task Network (HTN) Planning. A Bioinformatics case study for pathway data retrieval is used to validate the effectiveness of proposed approach. The planning algorithm extend Hierarchical Task Network (HTN) algorithm to solve the problem of automatic web service composition in the context of concurrent task planning. Experimental analysis showed that the proposed algorithms are capable of detecting and generating concurrent plan when compared with existing algorithms.

Keywords: *Web service composition, Semantic web service, Artificial Intelligent (AI) Planning, Hierarchical Task Network (HT) Planning,*

1. INTRODUCTION

Service Oriented Architecture (SOA) nowadays is gaining more attention in many sectors including education, business, science and the government [1]. Web service is an example of SOA implementation that operates transparently and seamlessly. Web Service Description Language (WSDL), Simple Object Access Protocol (SOAP) and Universal Description Discovery and Integration (UDDI) are the core components in web service. The services based application has promising technologies to solve various activities such as in business, travel trips and hotel booking. In web service application, there are several tasks need to be handled by both user and provider. These web service tasks include service discovery, selection, composition, execution and monitoring. Among these tasks, the composition task is widely applied in the industry and academia research. Web service composition is a combination of several services that can produce new compound services to fulfill user's objectives. Therefore, solving web service composition problem in automatic way is difficult and challenging [21].

In recent years, several AI Planning techniques have been proposed to solve automatic web services composition problems [14] [17] [20] [24]. The problems can be depicted as AI planning problem which allow various AI planning techniques in facilitating service composition problem. AI planning techniques are operated by generating plans before execution and utilized in web service composition problem when a planner is specified by its preconditions and effects. The preconditions and effects are input and output of web services. A planner will generate a sequence of plans when the initial states and goals with a particular task network are given. In addition, web services

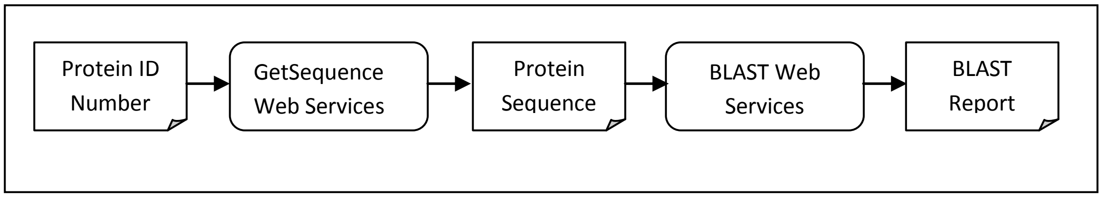

Figure 1: Sequence Of Web Service Composition In Bioinformatics Application

description will be used to translate into planning operator that allows the planner to produce composition goals based on user's requirements [24].

In previous works, HTN planning based algorithm has various advantages to overcome automatic web services composition problems. One of the main reasons HTN planning is suitable in web service composition is OWL-S description structure is similar to hierarchical structure of HTN planning [24]. Hence, the composite process in OWL-S can be mapped into HTN operator and method. The operator and method allow planning algorithm to identify sequential plan for a given problem domain.

However, the limitation that has been neglected in recent years by existing HTN planning based algorithm for web services composition is the inability to allow concurrent plan for task execution. Existing HTN planner (i.e. SHOP2) only allows for a sequence of plans to be generated. This scenario generally occurs in some domains that have concurrent execution of services including in bioinformatics domain. Thus, the capability and usage of this technology (semantic web services) are not fully leveraged. Therefore, this paper focuses on overcoming the stated problem and limitation, which can be described in the following statement:

"Given the problem of web service composition, it is a challenging task to develop AI planning algorithm to enhance the capability in generating concurrent plan for automatic web service composition"

The next section presents the background of web service composition in Bioinformatics and overview of Artificial Intelligence planning. Section 3 shows the applicability of semantic web services in Bioinformatics data retrieval. Section 4 discusses a proposed approach to generate a concurrent plan and numerical evaluations. The last section is the conclusion and future works.

2. BACKGROUND

2.1 Web Service Composition in Bioinformatics

Web services composition scenarios have been studied in bioinformatics field [16] [20] [24] [26] [27] [29]. Applying web service composition in bioinformatics allows new composite services to be offered. Most of bioinformatics database and tools (web services) are exposed publicly on the web [3]. Biologists might want to perform some experiments using distributed data and tools provided by many organizations. Using web services is useful as there are many web services in life science [19]. Typically, biologists have to search for their desired services to help them in choosing applicable service. However, searching for accurate web services is under web service discovery problem, which is not covered in this study [25]. Meanwhile, the services found by biologists will be utilized to perform experiments using Basic Local Alignment Searching Tools (BLAST) [22]. There are some cases where biologists will use other services together to fulfill their requirements. Therefore, output resulted from one service will be used as input to another services manually. Figure 1 illustrates the sequence combinations of services example as discussed in [22]. This task is time consuming and a domain expert is needed to assist biologists in web services environment. Thus, automatic web service composition can help them to achieve the objectives of the experiments.

2.2 Artificial Intelligence (AI) Planning

An automatic process of web service composition is needed to overcome web service composition problems. The environment of service-based application depends on time consumption and also cost effectiveness [21]. Thus, automatic service composition seems suitable to solve time and cost consumption in web service application. In addition, AI planning is capable to overcome the problem of automatic web service composition. In order to use AI planning technique, semantic web community has investigated on the problem of how to synthesize complex actions when given an initial state, goal and possible state of transition [24]. In web service composition, AI planning depicts services as actions while service composition problem depicts as planning problem. Thus, it enables AI planning to find a suitable plan and order them in an appropriate sequence.

AI planning problem can be described as tuples $\langle S, S_0, G, A, T \rangle$ where:

- S is the set of all possible states of the world;
- $S_0 \subset S$ denotes the initial state of the world
- $G \subset$denotes the goal state of the world that the planner attempts to reach;
- A is the set of actions the planner can perform to achieve a desired goal;
- The translation relation $T \subseteq S \ x\ A\ x\ S$ defines the precondition and the effects of the execution of each action.

Based on the AI planning described above, representation of web service composition problem is as follows:

- S_0 and G represents the initial state and goal state that are specified by the service requester,
- A is the set of available services and T denotes the current states of each service.

3. APPLYING SEMANTIC WEB SERVICES FOR PATHWAY DATA RETRIEVAL

The processes of applying semantic web services using OWL-S in bioinformatics field consist of four phases. Figure 2 shows the process taken in this experiment. The first phase consists of determination of WSDL collection provided by several life science web services such as KEGG, myGrid and Bio-Catalogue. The documentation and details on how to use the services are provided in the manual and documentation for each provider respectively. The documentation includes the type of input needed and expected output that will be generated. After determining which service to be used, the development and converting process of syntactic web services into semantic web services is taken. Tools such as Protégé and OWL-S API are used to convert WSDL web services into OWL-S semantic web services. The next part is the development of domain ontology. By using OWL-S, the annotation of domain ontology is crucial as it determines the complexity of OWL-S development. This includes input, output, precondition and effects that will be employed in domain ontology. The most complete domain ontology for life science web services is Gene Ontology (GO) and myGrid Ontology. In domain ontology, OWL-S description containing *serviceProfile*, *serviceModel* and *serviceGrounding* that can be constructed based on user's requirements.

However, to perform this development process, a domain expert in biology is required in order to analyze and perform a specific biological process. The biological process includes data retrieval of basic information of gene and pathway, genome assembly and annotation, sequence alignment and many others. This research mainly focused on data retrieval of basic information of gene and pathway. Data retrieval for basic gene and pathway is chosen due to knowledge constraint and lack of other complex bioinformatics task to be performed. In addition, this is a good starting point to explore more complex applications to be applied in this field.

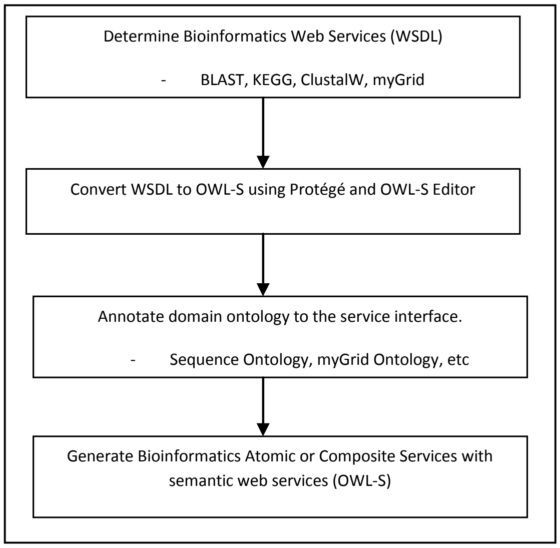

Figure 2. The Procedure Of Developing Semantic Web Services In Bioinformatics

4. GENERATING CONCURRENT PLAN

SHOP2 algorithm generates a sequence of operators that solves a task in problem domain. Biological question, "*what pathway my gene of interest is evolved*?" is used as the case study. The information needed give significant information and very important for biologists to study and analyze in the first phase of systems biology research. In this phase, biologists will identify the related information by searching and querying in Kyoto Encyclopedia of Gene and Genomes (KEGG) database. The process of searching in KEGG consist of two concrete services; getPathway and getGeneDescription. Thus, instead of generating sequence of these services, executing in parallel way and combine the services as a compound services is more efficient. The information contains of pathway information including name of gene and pathway. The sequence of plans generated from SHOP2 is utilized to find and generate concurrent plans.

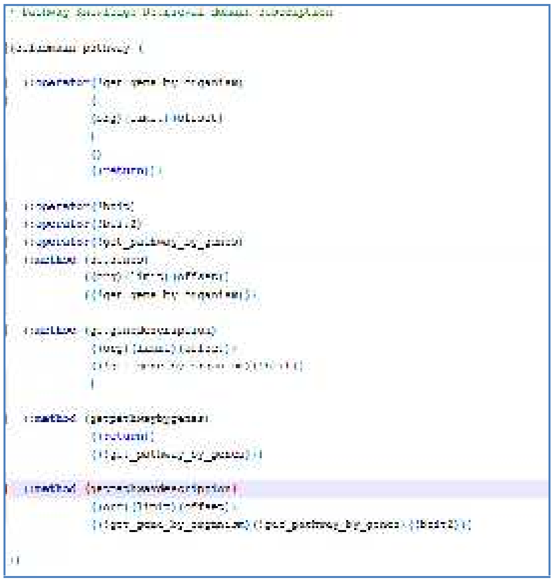

Figure 3 Part Of Domain Description In Pathway Information Retrieval

Figure 4 illustrates the algorithm to find and generate concurrent plan by using pathway information retrieval dataset that expressed in domain description file (Figure 3). The domain description file is generated using procedure in Figure 2. Input is applied at first in SHOP2 algorithm to generate list of sequential plans before it is used to find and detect concurrent plan. The decisions made by the algorithm, on which tasks may be executed concurrently, are based on the required resources (inputs). Lines 6-10 of algorithm prepare resources allocation that match operator, assign the *r_list* and create the action list. The *r_list* holds a list of operators which matches the following syntax:

[!*operator_name* [‘Resource$_1$’,…,’Resource$_n$’]]

The action lists hold operator, resources and task in the task network. Then, the resources are checked whether the same resources have been used, then the concurrent plan may be initialized. Lines 17 – 26 of the algorithm generate the concurrent plan. Firstly, the number of tasks in problem domain is determined. This will allow the algorithm to calculate the number of operators taken in to solve the planning problem. Every task included in Action will be matched after the first task is determined. If the same resources are used in both task lists, the plan is determined as a sequence of plans. Otherwise, when the resources are same, the plan may have concurrent plans.

Algorithm: Find concurrent plan

Inputs: Set of operator *o* in sequential plan {*P*}

Output: Concurrent and Sequential plan generated, $P = \{p\}$, $CP = \{p_1 \parallel p_2 \parallel p_n\}$

```
1:  sp = {p} // initialize sequential plan
2:  cp = { } // initialize concurrent plan to empty list
3:  operator = { } // initialize current plan to empty list
4:  plan ← ParsePlanToList
5:  while p ≠ 0 do
6:  begin ResourceAllocation(plan)
7:    // match operators o in plan with resources (res)
8:      r_list ← (($o_1$, [$rest_1$,…,$res_n$])),….($o_n$,[$res_1$,…$res_n$]))
9:    // create action object from each item in r_list
10:     a_list ← ($Action_1$($t_1$, $o_1$, $res_1$),…,$Action_n$($t_n$, $o_n$,$res_n$))
11:     begin ConcurrentInitialize(a_list)
12:        //create links between operators from resources
13:        l_list ← Link(a_list)
14:        //remove duplicates from l_list
15:        l_list ← Unique(l_list)
16: //check resources. If not same resource applied, made concurrent plan
17: foreach Action a in l_list do
18:   //get task resources
19:     Task t = { }
20:      t ← t.getNum // get set of task in T in problem domain
21:     foreach task t in Action a do     //match first task with next task list
22:       if $task_1$.res ==  $task_2$.res            // same resources used, plan sequential
23:           sp ← {$t_1$ }                      //return sequential plan
24:       else if $task_1$.res ≠ $task_n$.res
25:           cP ← {$t_1$,.. $t_n$}       //return concurrent plan
26:      end for
27: end while
```

Figure 4. Algorithm For Finding Concurrent Plan

4.1 Evaluation on Pathway Retrieval Dataset

The implementation developed allows user to retrieve information from KEGG using semantic web services OWL-S. For this study, suppose biologists want to know which pathway that gene *preprotein translocase subunit SecA (secA)* evolved in *L. lactis* organism (KEGG identifier for this organism is *llm*). The input specified is offset, limit and organism value. The output produced for pathway description from this composition is *path:llm03060* Protein export and *path:llm03070 Bacterial secretion system - Lactococcus lactis subsp. cremoris* MG1363. For gene description, the outputs are *llm:llmg_0124 secA; preprotein translocase subunit SecA* and *K03070 preprotein translocase subunit SecA*. Figure 5 shows the workflow of input and output produced from one service to other services generated from OWL-S. The concurrent services generated in this experiment are *GetPathway* and *GetGeneDescription* which can be executed concurrently.

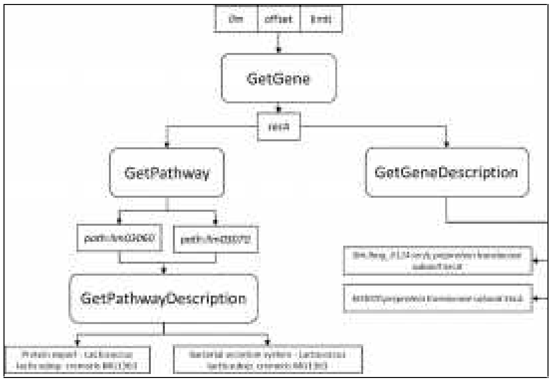

Figure 5 Workflow Generated For Pathway Information Retrieval In KEGG

In automated planning for web service composition, time taken to find plan is crucial to determine the time consumption. Two objectives are identified by performing evaluation; to see how the performance of runtime is affected with concurrent reasoning capability and to measure whether the proposed algorithm is efficient to allow the execution of concurrent task. Operator's cost is used as a measurement scale of the proposed algorithm because the previous HTN algorithm does not allow concurrent task execution.

The measurement of computational runtime, *t*, for executing plans *P*, is given by the following equation:

$$\text{Runtime} = t(P) = \frac{t2 - t1}{1000} \qquad (1)$$

Where $t1$ represents initial time before planning algorithm is performed and $t2$ represents time after the execution of planning algorithm is completed. The exact time taken to perform planning processes is divided by 1000 in order to obtain more precise unit in milliseconds. Figure 6 denotes the result from the experiment (using pathway information retrieval). The upper section (*sequence*) illustrates the overall execution time and the execution time of each task in sequential order (previous algorithm). The lower section (concurrent) shows the overall execution time, execution time of each task and parallel ordering of concurrent plan. A significant difference of 8 milliseconds or respectively 33.3 % is observed after optimizing the sequential plan into the concurrent plan. This result proves that the algorithm is effective to handle concurrent task in web service composition scenario. In this scenario, the pathway information retrieval is tested.

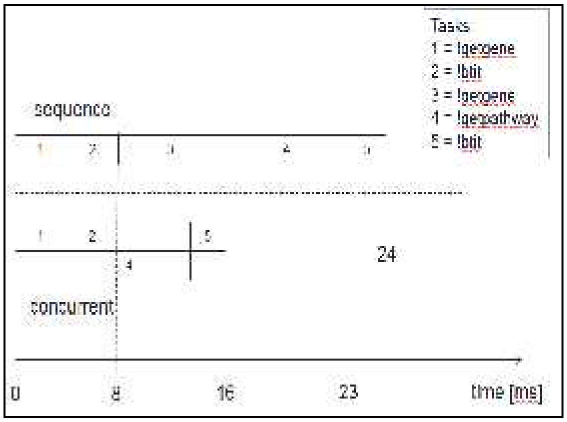

Figure 6 Runtime Plan Measurement For Sequential And Parallel

4.2 Evaluation on Logistic Dataset

The logistic industry involves transporting package from one location to another by using truck to transfer in the same city and plane to transfer in different cities. The operations of logistics transportation will rule out the correctness of the plan using the proposed algorithm. This is because logistic dataset commonly used in classical AI scenario and it is not applied in web service problem. For example, a truck is able to pick up one package at a time. Otherwise, using the proposed algorithm, the truck might have the possibility to pick up multiple packages at certain times. However, the planning performance is needed to compare with the previous planner in terms of computational time and costs.

Dataset are obtained from standard benchmarking planning from Artificial Intelligence Planning Competition (AIPC). This dataset is used due to the fact that domain problem can be simulated based on concurrent situation for moving the vehicles. The components of logistics dataset include vehicles, packages and also locations. The packages can be transferred by vehicles concurrently if there are no conflicts of other vehicles that performing the same tasks. There is no standard benchmark problem dataset and evaluation method for web service composition [25]. Thus, the logistic dataset is suitable to be used in AI planning for web service composition in this research.

Figure 7 shows the results of the proposed algorithm using logistics dataset. Three problems which have different complexity are created. Problem 1 consists of two locations and two packages that need to be transferred from one location to another location. Problem 2 involves three locations and two packages and Problem 3 has four locations with six packages. Each problem is set in the planner to find 100, 200, 300 and 400 plans. The result shows that JSHOP2 has better performance in terms of computational runtime (milliseconds) in order to find the plan compared with the proposed algorithm. The reason for such behavior is due to the proposed algorithm which has the concurrent plan capability that consume slightly more computational time to find and generate the concurrent task. JSHOP2 does not have any mechanism to handle the concurrent process. Thus, in the proposed algorithm, the mechanism to handle the concurrent process has affected planner performance.

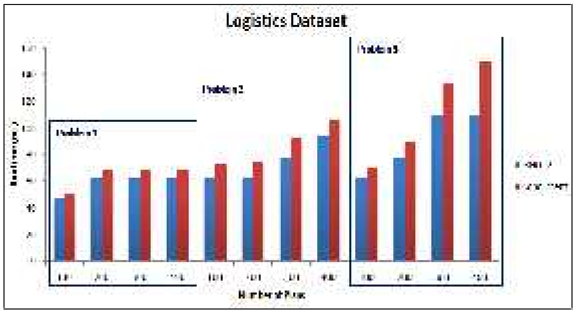

***Figure 7** Experimental Results With Runtime Measurement*

Figure 8 shows the different perspective of experiments in terms of cost measurement when performing the algorithm. The cost is assigned to operators by default value of 1.0. Means that, multiple operators found in certain time will assigned as value 1.0. The result shows that the proposed algorithm is efficient to handle the concurrent task. The cost is reduced by some operators when the similar multiple operators are pruned where they are already applied in the task network. The existing JSHOP2 algorithm is a partial-order planner. Thus, the planner can produce all possible combinations of task sequences when given more than one plan number. For instance, by using :*unordered* task control, the algorithm will find any combination task with any-order combination which results to the redundancy of generated plan. In order to avoid this, the problem number is set to 6 which mean the plan just returns one plan containing all operator instances that transfer 6 packages to deliver. Then, the operators' cost that solves the planning problem can be determined by the generated output.

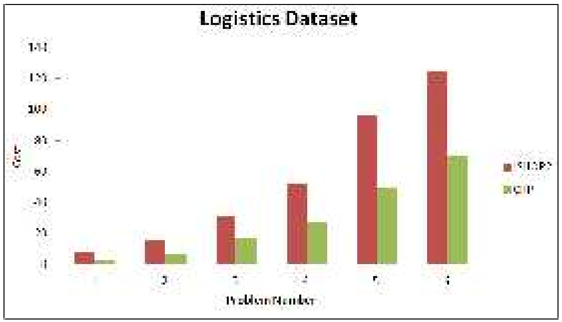

Figure 8 Experimental Results With Cost Measurement

Figure 9 shows the examples of output generated from the planner for 2 plans .Two packages need to be transferred from one location to another location. Package one is transferred from *loc8-3* to *loc8-1* and package two from *loc1-1* to *loc1-2*. The location is in the same city where there is no conflict between operators. The planner detects the first two operators as the merge operator. The next three operators are the concurrent operators which has different operator for transferring two packages in the two plans.

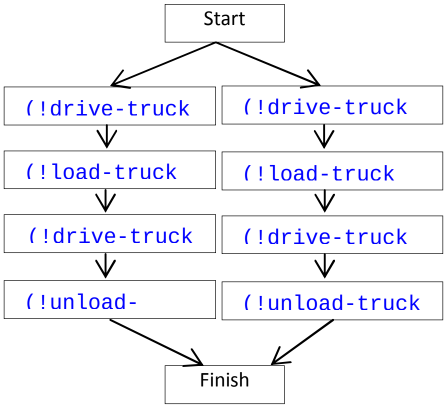

Figure 9 Two Packages Delivered By Different Locations

5. CONCLUSION

This paper proposes a new approach to extend the limitation of the previous algorithm by enabling their capability to generate a concurrent plan. The proposed algorithm based on Hierarchical Task Network (HTN) enhances SHOP2 planning system to detect and generate a concurrent plan based on the output of SHOP2 (sequence plan). To trigger the availability of concurrent planning, allocation of resources based on web services inputs is used. The inputs (resources) are compared with SHOP2 operator, and concurrent plan will initialize if the instance of inputs equal to SHOP2 operator. To evaluate our approach, we perform two experiments using pathway information retrieval and logistic dataset from SHOP2 benchmark problem. The result of pathway information retrieval shows that this approach is able to find and generate a concurrent plan, but it takes longer computational time. Meanwhile, by using logistic dataset, the proposed algorithm is efficient to handle concurrent tasks based on cost reduction by some pruned operators. Therefore, in our future work, we intend to examine the approach in other complex Bioinformatics and system biology workflow which widely used web services as their analysis tools.

ACKNOWLEDGEMENT

This work is partially supported by Universiti Teknologi of Malaysia with Research University Grant (GUP), Vot Number: 04H34 and ScienceFund 4S102. The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation

REFRENCES:

[1] Abramowitz, W., & Eymann, T. (2008). “Service oriented architecture and web services”. Wirtschaftsinformatik, 50(1), 1–3.

[2] Berners-Lee, T., Hendler, J., & Lassila, O. (2001). “The semantic web - A new form of web content that is meaningful to computers will unleash a revolution of new possibilities”. *Scientific American*, 284(6), 34.

[3] Cardoso, J., & Sheth, A. (2006). “Semantic web services: Processes and applications”. New York, NY: Springer.

[4] Dong, Z., Qi, C., & Xu, X. F. (2004). “Ontology based intelligent sensing action in golog for web service composition”. *Grid and Cooperative Computing*, 3032, 964–971

[5] Erol, K., Hendler, J., & Nau, D. S. (1994). “HTN planning: Complexity and expressivity”. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, (pp. 1123-1128). Seattle, WA: NCAI.

[6] Ghallab, M., Nau, D., Traverso, P. (2004). “Automated Planning: Theory and Practice”, Armsterdam Morgan Kaufmann Publisher, 2004.

[7] Garcia, S. F., Fernandez, J. T., Valencia, G. R., Gomez, J. M., & Martinez, B. R. (2008). “Combining semantic web technologies with multiagent systems for integrated access to biological resources”. *Journal of Biomedical Informatics*, 41(5), 848–859.

[8] Hossny, E., AbdElrahman, S., Badr, A. (2010). “An Intelligent Web Service Workflow: A Petri Net Based Approach”. *The Open Information Systems Journal*, 4, 1-11.

[9] Kown, J., Lee, D. (2012). “Non-redundant web services composition based on a two-phase algorithm”. *Data & Knowledge Engineering*, 71, 69-91

[10] Mali, A. D. (2000). “Hierarchical task network planning as satisfiability”. *Recent Advances in AI Planning*, 1809, 122-134.

[11] Martin, D., Paolucci, M., McIlraith, S., Burstein, M., McDermott, D., McGuinness, D., & Sycara, K. (2005). “Bringing semantics to web services: The OWL-S approach”. *Semantic Web Services and Web Process Composition*, 3387, 26-42. Retrieved from http://eprints.ecs.soton.ac.uk/13000/1/OWL-S-SWSWPC2004-final.pdf

[12] Medeiros, C. B., Perez-Alcazar, J., Digiampietri, L., Pastorello, G. Z., Santanche, A., Torres, R. S., & Bacarin, E. (2005).

WOODSS and the web: Annotating and reusing scientific workflows. *SIGMOD Record*, 34(3), 18–23.

[13] Mcllraith, S., Son, C. (2002). “Adapting Golog for Composition of Semantic Web Services”, In *Proceedings of the 8th International Conference on Principles and Knowledge Representation and Reasoning*. 482-496.

[14] Nariai, K., Paik, I., & Shinozawa, M. (2005). “Planning and composition of web services with dynamic constraints using situation calculus”. In *Proceedings of the Fifth International Conference on Computer and Information Technology*, (pp. 1009-1013). Shanghai, China: IEEE Press. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1562790

[15] Nau, D., Au, T. C., Ilghami, O., Kuter, U., Murdock, J. W., Wu, D., & Yaman, F. (2003). “SHOP2: An HTN planning system”. *Journal of Artificial Intelligence Research*, 20, 379–404.

[16] Nau, D., Au, T. C., Ilghami, O., Kuter, U., Wu, D., Yaman, F., & Murdock, J. W. (2005). “Applications of SHOP and SHOP2”. *IEEE Intelligent Systems*, 20(2), 34– 41.

[17] Oh, S.-C., Lee, D., & Kumara, S. R. T. (2006). “A comparative illustration of AI planning-based web services composition”. *SIGecom Exchange*, 5(5), 1–10

[18] Pasquier, C. (2008). “Biological data integration using semantic web technologies”. *Biochimie*, 90(4), 584–594

[19] Potter, S., & Aitken, S. (2005). “A semantic service environment: A case study in bioinformatics”. *Semantic Web: Research and Applications*, 3532, 694–709

[20] Qiu, S., Wang, Y., Liu, Y & Cheng, L. (2011). “Bioinformatics Web Services Composition with Case Based Reasoning”. *Journal of Convergence Information Technology*, 6(2), 240 - 253

[21] Rao, J., Su, X. (2004). “A Survey of Automated Web Service Composition Methods, Semantic Web Services and Web Process Composition”: *First International Workshop, SWSWPC*, vol 3387 / 2005. 43

[22] Sahoo, S. S., Sheth, A (2006). “Bioinformatics Applications of Web Services, Web Processes and Role of Semantic, in Semantic Web Services, Processes and Applications”, J. Cardoso and A.P. Sheth, Editors. Springer US. p. 305-322

[23] Sirin, E., Parsia, B., & Hendler, J. (2004). “Filtering and selecting semantic web services with interactive composition techniques”. *IEEE Intelligent Systems*, 19(4), 42–49.

[24] Sirin, E., Parsia, B., Wu, D., Hendler, J., & Nau, D. (2004). “HTN planning for web service composition using SHOP2”. *Web Semantics Science Services and Agents on the World Wide Web*, 1(4), 377-396.

[25] Sirin, E. (2006). “Combining Description Logic Reasoning WIth AI PLanning for Composition of Web Services”. PHD Thesis, Department of Computer Science, University Of Maryland.

[26] Tiwari, A., & Sekhar, A. K. T. (2007). “Workflow based framework for life science informatics”. *Computational Biology and Chemistry*, 31(5-6), 305–319.

[27] Xu, K., Yu, Q., Liu, Q., Zhang, J., & Bouguettaya, A. (2011). “Web service management system for bioinformatics research: A case study”. *Service Oriented Computing and Applications*, 5(1), 1–15

[28] Zheng, G., & Bouguettaya, A. (2009). “Service based analysis of biological pathways”. *BioMed Central Bioinformatics*, 10(6).