International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 8 (2017) pp. 1765-1771

PSO-inspired BIRCH and Improved Bipartite Graph for Automatic Web Service Composition

Dr.K. Meenakshi Sundaram
Associate Professor
Department of Computer Science
Erode Arts and Science College, Erode, Tamilnadu, India

T. Parimalam
Research Scholar
Department of Computer Science
Erode Arts and Science College, Erode, Tamilnadu, India

Abstract

Web services are published by the service providers through the internet as independent software components by fulfilling the requirements of the customer requests. As the customer requests are not always satisfied by a single service, the methods of services composition were evolved in which a chain of services are composed together. However the longer search time for the discovery of composed services is a challenging problem causing the customer to wait long time especially while using larger web service set. In this paper, an efficient automatic web service composition approach is proposed to improve the performance by introducing domain-ontology-based PSO-inspired Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) clustering and Improved Bipartite graph in the pre-processing and service discovery & ranking phases respectively. Domain-ontology-based PSO-inspired BIRCH incrementally and dynamically clusters the web services in hierarchical manner with optimizing the clustering results to reduce the processing time. The approach improves clustering quality by additional scans and thus creating sophisticated environment for service match-making. In discovery phase, the Improved Bipartite graph produces maximal bicliques by pruning the non-maximal bicliques in the search space without restricting the inputs and reduced computations for elimination. Thus the best service clusters is discovered and ranked which ensures the selection of best composition plan in the planning, verification & execution phase. Experimental results performed in terms of accuracy, runtime, recall and precision, shows that the proposed framework improves the efficiency of web service composition.

Keywords: Web service composition, BIRCH, PSO, Bipartite graph, bicliques

INTRODUCTION

Automatic web service composition is viewed as the modified description of the general web service composition developing from just a manually reachable applications and web of information to the formation of Service Oriented Architecture (SOA) web of involuntary services. Web service composition aims at finding a service to fulfill the given service request by the customer as in maximum situations, a single service in the web service repository cannot fulfill the requirements of the requester. Thus it becomes necessary to find a sequence of services that can be connected functionally as a composite service to provide complete fulfillment of the customer requests. This problem of finding related and composable services is called as web service composition problem, which can be resolved by using two types of approaches. The first approach is the generation of workflow business model using GUI-based modeling software while the second approach is the automatic composition of existing services. However the first approach is plagued with errors and highly impractical, when infinite services are in the warehouse.

In this paper, a proficient automatic web service composition framework is proposed, that consists of three Phases: Pre-processing Phase, Service Discovery and Ranking Phase and Planning, Verification and Execution Phase. In the pre-processing phase, the hierarchical agglomerative clustering is utilized for clustering the services registered in the directory based on the chosen similarity measure, in a bottom-up approach. Through a greedy manner the clusters are either partitioned or combined and each service initiates its own cluster and pairs of clusters are combined in the hierarchy. However it has clustering complexity and reduces the computation speed for larger service sets. Hence the domain-ontology-based PSO-inspired BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies [6]) clustering is developed to minimize the processing time for clustering by incrementally and dynamically partitioning the web services in hierarchical manner along with optimization of the clustering results. In the second phase, service matching and ranking processes are performed using bipartite graph matching. As bipartite graph does not provide effective scaling and relatively inefficient due to the restrictions on the inputs, an improved bipartite graph is proposed. Improved Bipartite graph [12] does not increase the size of the problem and does not have any restrictions on the inputs, thus effectively producing maximal bicliques and ensures optimal matching. Then ranking is done based on the QoS parameters and the third phase of Planning, Verification and execution is carried out. By utilizing the proposed approach in the automatic service composition framework reduces the processing time and improves the service discovery efficiency.

RELATED WORKS

In this section, research methodologies proposed in the recent past related to the automatic web service composition are discussed. Alfredo Cuzzocrea et al [1] proposed a graph-based method for discovering semantic web services via matching composite services inorder to overcome the complexity in matching process. The proposed method compares both the atomic services of the composite process using their service

profiles, as well as the approach in these services are composed to generate the composite process. Matchmaking is then performed based on node similarities and identifying common frameworks between the two graphs that characterize the requested and the existing composite process. However the approach considers only graph structures and not the graph nodes and graph indexing which may result in below-par performance.

Deivamani Mallayya et al [2] introduced an automatic web service composition approach with a QoS based ranking algorithm for enhanced web service discovery. The proposed algorithm called as user preference based web service ranking (UPWSR) algorithm ranks web services depending upon the user requirements and QoS factors of the web service The proposed framework allows user to offer feedback about the composite service which enhances the reputation of the services. Though the approach achieved the lower execution time by fulfilling user preferences, the variability of service set reduces the performance.

Dimitrios Skoutas et al [3] proposed a combined application method for ranking and clustering the web services using the multi-criterion dominance relationships between the services. the matching is followed by ranking which is done by three algorithms for ranking the search results, and two algorithms for selecting the most representative services for clustering, so that the produced clusters reflect the trade-offs between the matched parameters.

Gopal N. Rai et al [4] proposed the approach of algebraic modeling and verification for effectual web service composition. The proposed algebra provides a sound foundation for verification and reasoning. However the approach does not possess efficient matching technique which may reduce the performance of the web service discovery.

Guobing Zou et al [5] introduced a dynamic composition approach using planners for the large size service repository. The proposed approach reduces the response time by combining the artificial intelligence planners to translate the Web service repository into a planning domain. However there are many limitations such as the support of semantic Web service composition, re-planning of composite service during the execution failure, and efficiently updating WSC planning domain.

Junming Zhang et al [7] introduced a neural network based schema matching method for the service matching in the discovery process. Neural network based schema matching method (NNSM) combines the matching results generated from different semantic matchers. However the approach only depends on word similarity but most of the web services also contain lot of phrases, hence phrase matching is required.

Pablo Rodriguez-Mier et al [8] introduced A* algorithm for resolving the issue of semantic input-output message arrangement matching for web service composition. When a customer requests, a service dependency graph in the company of a subset of the original services from an exterior warehouse is dynamically created. Then, the A* search algorithm is employed to discover a minimal composition that fulfills the user request. The graph is optimized to reduce the ineffective and alike services. However more competent optimization techniques can considerably enhance the matching process.

Prasanth S H [9] proposed a backtracking based approach for the semantic matchmaking in the web service discovery process. The proposed approach generates a sorted list of available operation interface annotations, which is used to back-tracking the services that are matching with the query service. Though the backtracking approach provides netter precision with reduced search time, the distribution pattern of the interface annotations are similarity degree scale is found to alter the average number of states per State List and thus alters the runtime.

Qiang Yu et al [10] proposed ant colony based optimization (ACO) technique in the web service composition techniques in the cloud computing environment. ACO has advantages such as allowing positive feedback, distributed computing, and a constructive greedy heuristic search. Based on ACO, the web service composition algorithm chooses the combination that has reduced number of clouds. However the semantic data is not included in the web service composition algorithm. Tao Wen et al [11] analyzed the possible effective clustering for the web service discovery.

PROPOSED SYSTEM

A web service composition framework consists of three Phases: Pre-Processing Phase, Service Discovery and Ranking Phase and Planning, Verification and execution phase.

A. Pre-processing phase

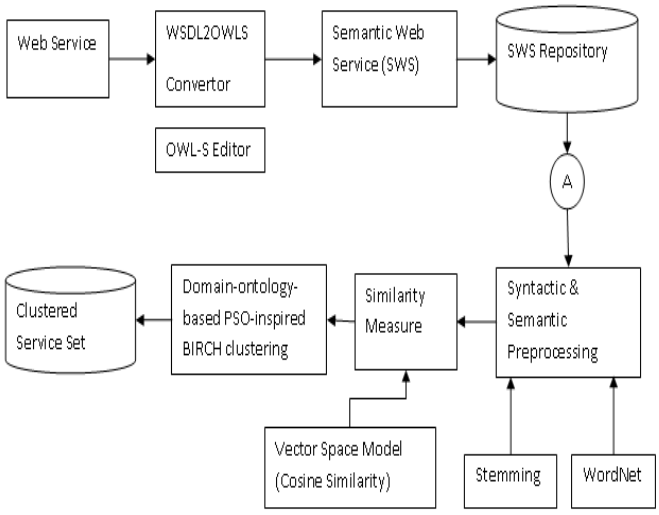

Figure 1. Pre-processing phase

In the pre-processing phase, the necessary pre-requirements for the efficient service discovery are carried out from the repository. The WSDL2OWLS converter tool is used to convert the web services from the WSDL format to the OWL-S format. The OWL-S descriptions are generated from the converter might miss the semantic and control flow informations and hence the OWL-S is edited using the OWL-S editors which add the semantic and control flow informations. Thus formed semantic web services are stored in the repository and after advertisements (A) the clustering begins, the semantic web services are to be clustered for discovering the efficient web service composition. For this purpose, the similarity measures are computed using the semantic/syntactic components like semantic inputs/outputs, syntactic

inputs/outputs, texts in natural language, tags and preconditions. Usually, for semantic components the domain ontologies are used while for the syntactic informations, WordNet is used as ontology. Using the similarity measures, the web services are clustered to generate the clustered service set. Figure 1 shows the processing steps of pre-processing phase.

Hierarchical agglomerative clustering performs service clustering based on the chosen similarity measure, in a bottom-up approach. With the help of greedy manner, the clusters are partitioned or combined together and each service initiates self cluster and pairs of clusters are combined in the hierarchy. However, the complexity of Hierarchical agglomerative clustering consumes more time for clustering the large datasets. Hence the domain-ontology-based PSO-inspired BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) clustering is proposed.

B. Domain-ontology-based PSO-inspired BIRCH clustering

The domain-ontology-based PSO-inspired BIRCH clustering consists of two stages of processing. In the first stage, the clustering process is carried out using the BIRCH clustering for establishing the set of original clusters by grouping very similar instances and also filters the noisy data objects based on the similarity metrics. In the second stage, the original clusters are optimized by the PSO algorithm and near optimal results are obtained which are then labeled.

BIRCH clustering

BIRCH clustering is performed in four phases as in figure 2. Clustering Feature (CF) is a triple summarizing the data that is maintained around a cluster. The informations are represented as vectors and incorporate number of data points, linear sum of data points and square sum of data points. CF tree is a height-balanced tree with two vital parameters: branching factor and threshold. Each non-leaf node contains at most passages as CF and a child node. So a non-leaf node represents a cluster which is a composition of all the sub-clusters represented by its entries. A entry can be embedded into a CF tree by utilizing a proficient algorithm. The tree distinguishes the suitable leaf in recursively dropping the CF tree by selecting the nearest child as indicated by the selected distance metric. At the point when the algorithm achieves a leaf, it tests whether the leaf can store the entries without disregarding the threshold condition. If the condition is fulfilled the CF vector for the leaf node is redesigned to demonstrate the result. If there is space on the leaf for the new entry, the node should be splitted which is done by selecting the farthest pair of entries as seeds and then redistributing the remaining entries based on the nearest criteria. Then the route to the leaf is adjusted by updating the CF data for each non-leaf entry on the path. Then the refinements are converged to enhance the dissemination of the entries.

In phase 1, the principle work is to scan all the data and construct an underlying in-memory CF tree utilizing the given measure of memory and reusing space on disk. This CF tree tries to reflect the clustering data of the dataset as fine as could be expected under the memory limit. With crowded data points assembled as fine sub-clusters, and sparse data points eliminated as outliers, this stage makes an in-memory outline of the data. Then the computations are done fast, accurate and less order sensitive.

Phase 2 is for the mostly optional. The global clustering in the phase 3 regularly has distinctive input sizes inside which they perform well regarding both rate and quality. So conceivably there is a gap between the size of Phase 1 results and the input range of Phase 3. Phase 2 serves as a pad and connects this gap.

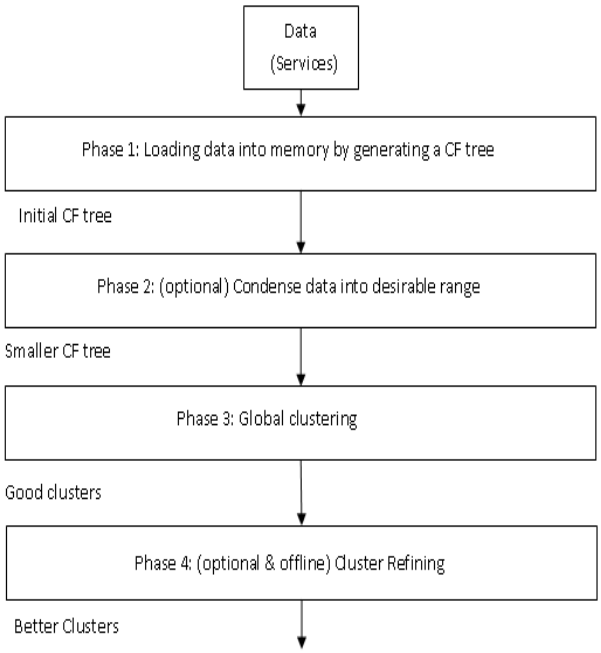

Figure 2. BIRCH clustering process

In phase 3, the unwanted consequence of the skewed input order, and partitioning invoked by page size causes to be disloyal to the real clustering patterns in the data are clarified by using a global or semi-global algorithm to cluster all leaf entries. The BIRCH is also a hierarchical clustering approach and hence the algorithm is straightforwardly applied to the sub-clusters because the data in their CF vectors is typically adequate, for calculating the distance and quality metrics. Completion of phase 3 provides a set of service clusters.

Phase 4 is also optional and it can be operated even in offline mode. Phase 4 is usually to purify the obtained set of clusters and reorganize the data points to its neighboring seed to acquire a set of new clusters. Thus the set of service clusters can be obtained.

Optimizing service clusters using PSO

The set of initial clusters obtained are used to set up the initial particle. Based on special management of memory the PSO optimized the objective function which iteratively improving a swarm of solution vectors named as particles. Each particle is updated by considering to the memory of individual and swarm's best data. Due to the collective intelligence of these particles, the swarm is able to frequently enhance its best observed solution and converges to an optimal service cluster.

C. Service discovery and ranking phase

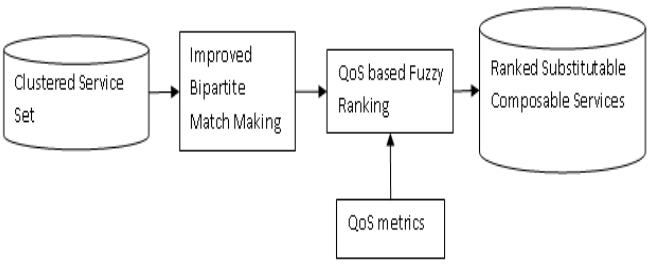

Figure 3. Service discovery and matching phase

After the clustering process, the web services are needed to be discovered to satisfy the customer requests and to rank the services in the discovered clusters based on the QoS parameters. As the Bipartite approach does not provide effective scaling and relatively inefficient due to the restrictions on the inputs, an improved bipartite graph matching is proposed which reduces the drawbacks of bipartite graph and provides better matching by producing maximal bicliques. Then the service ranking is performed by QoS-based fuzzy Ranking algorithm. Figure 3 shows the overall processing of the service discovery and ranking phase.

Improved Bipartite Graph

Integrating the service set requires the representation of the relationships between the pairs of disparate services with the interpretation of these relationships achieved through the enumeration of maximal bicliques. The efficiency is achieved through the improved bipartite graph without I/O restrictions. The improved bipartite graph utilizes Maximal Bicliques Enumeration (MBE) algorithm for combining backtracking and the branch-and-bound techniques to prune the non-maximal bicliques regions. The search space of MBE is limited to the vertices as in bicliques, the vertices in one set determines the vertices in other sets. The tree pruning and candidate selection using improved bipartite graph is shown in figure 4.

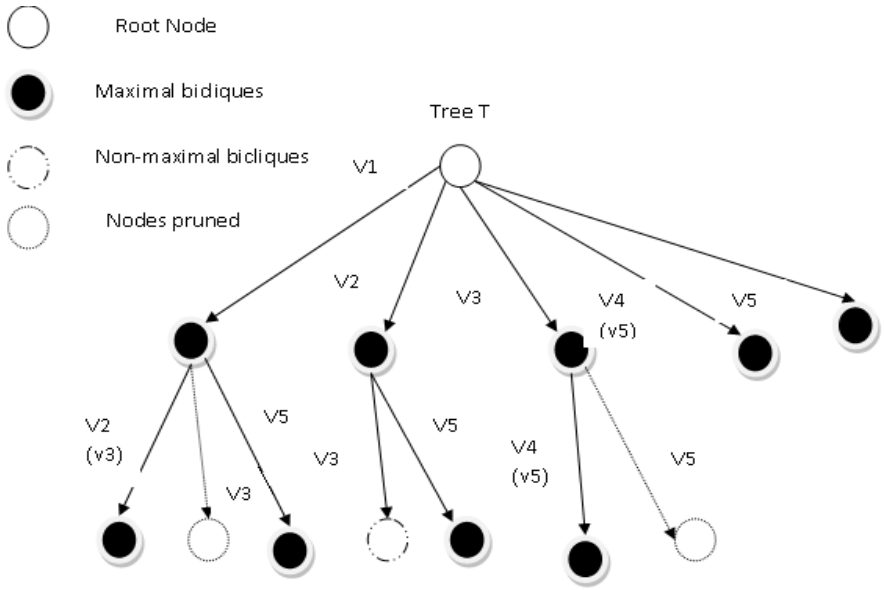

Figure 4. Improved Bipartite graph matching

Let G=(U ∪V,E) be a bipartite graph and it is understood that |U|≥|V|. MBE activates on the potentially smaller set V, by utilizing four dynamically varying sets of vertices: they are 1) R, a subset of V, 2) L, a subset of U including all the common neighbors of R, 3) P, a subset of V containing candidate vertices that may be included to R, and 4) Q, a subset of V consisting former candidates, vertices that were considered for R. The sets R, L, P and Q are utilized in a depth-first traversal of a recursion tree to form maximal bicliques. R and L are used to form such a bicliques, where R determines L. P is employed for bicliques addition. Q is used to decide maximal. P, Q and R are necessary to satisfy two significant conditions:

- P,Q,R are pair wise disjoint

  $(P \cap Q) \cup (P \cap R) \cup (Q \cap R) = \emptyset$

- P and Q must enclose every vertex in V but not R that is adjacent to at least one vertex in L.

  $P \cup Q = \{v | v \in V \setminus R, \exists u \in L, (u,v) \in E\}$

In MBE algorithm, a depth-first search tree traversal is achieved recursively using the core function biclique_find(). At first, all vertices are bicliques candidates (P = V, L = U), while the bicliques and previous candidate sets are vacant (R = Q = ∅). As the computation proceeds, R grows but L and P contracts. At each node of the search tree, biclique_find takes as input a 4-tuple ⟨L,R,P,Q⟩ and selects a candidate x from P. An additional step enhances R with x to form R', and forms L' from L by eliminating all vertices that are not linked to x. This makes L' a set of common neighbors for R'. P' and Q' are

then formed by eliminating vertices not connected to L'. P' also drops vertices linked to all of L. These are included to R. If there is no vertex in Q' is connected to L', then maximal bicliques (L', R) are detected. A recursive call is made with ⟨L',R',P',Q'⟩. x is eliminated from P and included to Q. The approach ends when either P is vacant or a vertex in Q is linked to all of L.

The deployment of MBE algorithm assures the construction of maximal bicliques. The results of MBE shows that the sub-graph induced by (L,R) is a biclique and (L, R) is maximal if no vertex in V\R is adjacent to every vertex in L. If Q includes a vertex that is neighboring to all vertices in L, then not only (L, R) is not maximal, yet there also can be no S for which (L, R ∪ S) is maximal. The MBE algorithm discovers all the maximal bicliques in a bipartite graph. It searches the entire search space of all subsets of one vertex set and discovers all the bicliques from which the maximal bicliques are found. It eliminates the bicliques which do not lead to maximality.

For termination of non-maximal bicliques and selection of maximal bicliques, the tree pruning and candidate selection approaches are utilized in the MBE algorithms. The addition of candidates whose neighborhoods contain that of x, upon recursive return MBE cares for vertices in S similar it does to other vertices in the candidate set which proves every vertex in S is chosen even if it is non-maximal subset. The generation of these branches can be avoided if S is sub-divided into two subsets. Vertices of the second group can thus be moved directly to Q upon recursive return, because any biclique that excludes x but includes v is a sub-graph of a biclique including both x and v. Then in candidate selection, the candidates v are generated the sub-branches more in the right most branches and are searched longer as the selected candidates have the same number of connections to L.

QoS-based Fuzzy Ranking Algorithm

QoS attributes are categorized into domain independent and domain dependent attributes. Attributes such as availability and successful execution rate are domain independent attributes. A fuzzy ranking approach is utilized by the service ranker to class the services in the QoS ranked list based on the QoS criteria (cost, reputation, average response time, frequency, and success rate) and the QoS constraints (QoS weights q specified by the user). Ranked services list (RSLi) includes the services for task i.

$$[\text{FinalRank}] = [\text{CostRank}]*q_1 + [\text{SuccRateRank}*q2 + [\text{Freq Rank}] *q_3 + [\text{RespRank}] *q_4 + [\text{RepRank}] *q_5 + [\text{AvailRank}] *q_6$$

where i denotes the location of web service in QoS ranked list and i varies from 1 to size of the QoS ranked list. $q1$ to $q6$ denote the QoS weight. After calculating the final rank, the services are sorted based on the final rank and stored in a ranked services list.

D. Planning, Verification and execution phase

In Planning, Verification and execution phase, the composition plans are generated by receiving the inputs from the discovery phase and then the plans are verified for its correctness. Then the optimal plan is selected based on the evaluation of multiple QoS parameters and it is executed for efficient automatic service composition. After the generation of plan, the verification method is performed using the Petri net based algebra which is used to model the control flows as a essential constituent of the dependable web service composition. For efficient automatic web service composition the optimal plan is executed for the requested customers.

EXPERIMENTAL RESULTS

In this section, the performance of the proposed method and existing method is evaluated in terms of accuracy, runtime, recall and precision. The performance results are compared with that of the web service composition framework using Hierarchical agglomerative clustering to determine the efficiency of the proposed framework. A total of 250 number of web services are considered for the evaluation with service requests ranging from 100 to 400. In this evaluation results the comparisons are made with a minimum number of 100 service requests.

A. Runtime

The evaluation values of the methods in terms of runtime are given in Table I.

Table I. Runtime (Ms) Comparison

Number of web services	AWSC-HAC (ms)	AWSC-PSO-BIRCH (ms)
50	10	8
100	14	12
150	19	15
200	29	23
250	36	30

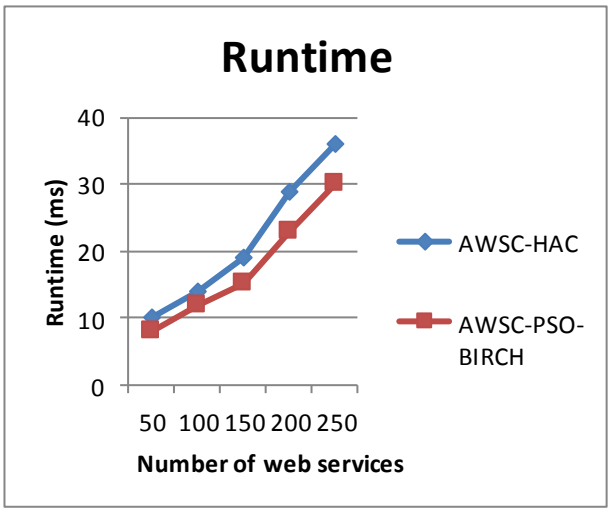

Figure 5. Comparison of Runtime

From the table, it can be found that when implemented, the time taken for the web services to respond in AWSC-HAC is more than it takes in AWSC-PSO-BIRCH, thus proving the efficiency of AWSC-PSO-BIRCH.

Figure 5 shows the comparison of AWSC-HAC and the proposed AWSC-PSO-BIRCH in terms of runtime (ms). From the figure it is clear that the proposed AWSC-PSO-BIRCH has reduced runtime and is efficient than the AWSC-HAC framework.

B. Accuracy

The evaluation values of the methods in terms of accuracy are given in Table II.

Table II. Accuracy Comparison

Number of web services	AWSC-HAC (%)	AWSC-PSO-BIRCH (%)
50	68	81
100	74	84
150	80	87
200	83	90
250	87	93

From the table, it can be concluded that the accuracy of AWSC-HAC is comparatively lesser than that of AWSC-PSO-BIRCH, thus proving that AWSC-PSO-BIRCH is more efficient.

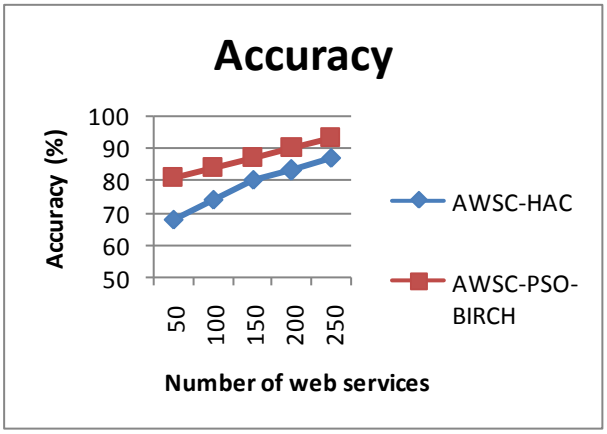

Figure 6. Accuracy

Figure 6 shows the comparison of AWSC-HAC and the proposed AWSC-PSO-BIRCH in terms of accuracy. From the figure it is clear that the proposed AWSC-PSO-BIRCH has improved accuracy and efficient than the AWSC-HAC framework.

C. Precision

The evaluation values of the methods in terms of precision are given in Table III.

Table III. Precision Comparison

Number of web services	AWSC-HAC	AWSC-PSO-BIRCH
50	0.65	0.74
100	0.72	0.78
150	0.76	0.83
200	0.81	0.85
250	0.84	0.90

From the table, it can be concluded that the precision of AWSC-HAC is comparatively lesser than that of AWSC-PSO-BIRCH, thus proving that AWSC-PSO-BIRCH is more efficient in terms of precision.

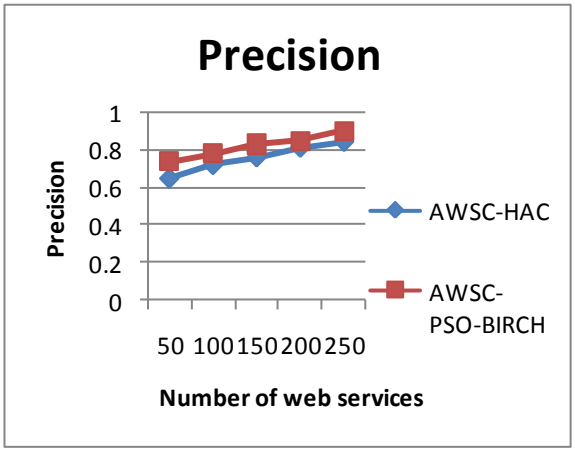

Figure 7. Precision

Figure 7 shows the comparison of AWSC-HAC and the proposed AWSC-PSO-BIRCH in terms of precision. From the figure it is clear that the proposed AWSC-PSO-BIRCH has improved the precision rate and efficient than the AWSC-HAC framework.

D. Recall

The evaluation values of the methods in terms of recall are given in Table IV.

Table IV. Recall Comparison

Number of web services	AWSC-HAC	AWSC-PSO-BIRCH
50	0.62	0.76
100	0.68	0.79
150	0.73	0.85
200	0.77	0.89
250	0.85	0.92

From the table, it can be concluded that the recall of AWSC-HAC is comparatively lesser than that of AWSC-PSO-BIRCH, thus proving that AWSC-PSO-BIRCH is more efficient in terms of recall.

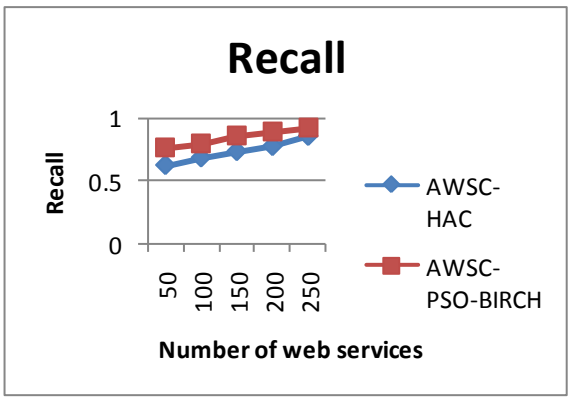

Figure 8. Recall

Figure 8 shows the comparison of AWSC-HAC and the proposed AWSC-PSO-BIRCH in terms of recall. From the figure it is clear that the proposed AWSC-PSO-BIRCH has improved its recall value and efficient than the AWSC-HAC framework.

CONCLUSION

In this paper, the novel concept of automatic web service composition is considered and an efficient framework is proposed. The proposed framework are the domain-ontology-based PSO-inspired BIRCH clustering and is developed for pre-processing phase to reduce the clustering process time by incrementally and dynamically clusters the web services in hierarchical manner along with optimization of the clustering results. Improved Bipartite graph is developed for service discovery process for efficient optimal matching of the service clusters. Then ranking is done based on the QoS parameters and the third phase of Planning, Verification and execution is carried out for better automatic web service composition. Experimental results also show that the proposed framework provides better service composition in terms of runtime, accuracy, precision and recall.

REFERENCES

[1] Alfredo Cuzzocrea, and Marco Fisichella. "Discovering semantic Web services via advanced graph-based matching." In 2011 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 608-615.

[2] Deivamani Mallayya, Baskaran Ramachandran, and Suganya Viswanathan. "An Automatic Web Service Composition Framework Using QoS-Based Web Service Ranking Algorithm." The Scientific World Journal, 2015.

[3] Dimitrios Skoutas, Dimitris Sacharidis, Alkis Simitsis, and Timos Sellis. "Ranking and clustering web services using multi-criteria dominance relationships." IEEE Transactions on Services Computing, 3 (3) pp. 163-177.

[4] Gopal N. Rai, G. R. Gangadharan, and Vineet Padmanabhan. "Algebraic Modeling and Verification of Web Service Composition." Procedia Computer Science vol 52, pp. 675-679.

[5] Guobing Zou, Yanglan Gan, Yixin Chen, and Bofeng Zhang. "Dynamic composition of Web services using efficient planners in large-scale service repository." Knowledge-Based Systems vol.62 pp. 98-112.

[6] Hai Zhou D, YongBin L. "An improved BIRCH clustering algorithm and application in thermal power." In 2010 International Conference on Web Information Systems and Mining (WISM), IEEE Vol. 1, pp. 53-56.

[7] Junming Zhang, Jinglin Li, Shangguang Wang, and Jiali Bian. "A neural network based schema matching method for Web service matching." In 2014 IEEE International Conference on Services Computing (SCC), pp. 448-455.

[8] Pablo Rodriguez-Mier, Manuel Mucientes, and Manuel Lama. "Automatic web service composition with a heuristic-based search algorithm." In 2011 IEEE International Conference on Web Services (ICWS), pp. 81-88.

[9] Prasanth, S. H. "A backtracking approach for semantic matchmaking algorithm in Web service discovery." In 2013 International Conference on Circuits, Power and Computing Technologies (ICCPCT), pp. 1065-1070.

[10] Qiang Yu, Ling Chen, and Bin Li. "Ant colony optimization applied to web service compositions in cloud computing." Computers & Electrical Engineering vol.41, pp.18-27.

[11] Tao Wen, Guojun Sheng, Yingqiu Li, and Quan Guo. "Research on Web service discovery with semantics and clustering." In Information Technology and Artificial Intelligence Conference (ITAIC), 2011 6th IEEE Joint International, vol. 1, pp. 62-67.

[12] Zhang Y, Phillips CA, Rogers GL, Baker EJ, Chesler EJ, Langston MA. "On finding bicliques in bipartite graphs: a novel algorithm and its application to the integration of diverse biological data types." BMC bioinformatics. Vol.15 (1), pp: 110.