ArcGIS Services Recommendation Based on Semantic and Heuristic Optimization Algorithm

Jiaqi Zheng[1], Jin Diao[2], Zhangbing Zhou[2,3(✉)], and Yongli Xing[1]

[1] School of Science, China University of Geosciences (Beijing), Beijing 100083, China
[2] School of Information Engineering, China University of Geosciences (Beijing), Beijing 100083, China
zhangbing.zhou@gmail.com
[3] Computer Science Department, TELECOM SudParis, 91011 Evry, France

Abstract. It is a common phenomenon that GIS service, a convenient tool, helps people to solve the problem in various fields. However, single GIS service can no longer meet the diverse needs of users. To address this challenge, a GIS services composition recommendation framework based on semantic and heuristic optimization algorithms is proposed in this paper. The Normalized Google Distance (NGD), as an indicator of invoking between two services, is used to construct dynamic semantic network. In order to save processing time, we use the hierarchical structure of ArcGIS services. In addition, we use the improved heuristic optimization algorithm to find the solution with the highest semantic value quickly. Consequently, once the initial parameters set and the end parameters set are given by the user, our GIS services composition recommendation framework will find the most appropriate Directed Acyclic Graph (DAG) to the user. The result of evaluation proves that our method could give more meaningful solution, compared with others.

Keywords: Semantic GIS service composition · Normalize Google Distance (NDG) · Heuristic optimization algorithm · ArcGIS service

1 Introduction

More and more tools and method for geospatial data analysis are being developed and distributed on the web, which makes it easier for us to solve problems in our lives [1]. For example, GIS services helps us find the best location to set up a fire station easily and quickly in [2]. Beside that, GIS services are also used in agriculture, medical care, transportation and various fields. Therefore, it is a trend that composing many GIS services together to provide added values to meet the user's requirement. Automatic services composition can be of great value to the GIS users, cause it can greatly broaden the functional ability to handle users' requirement [3]. However, it is still a big challenge for service

H. Ning (Ed.): CyberDI 2019/CyberLife 2019, CCIS 1137, pp. 487–501, 2019.
https://doi.org/10.1007/978-981-15-1922-2_34

developers that making GIS service composition fulfill functional requirements [4].

The process of service discovery, selection and composition is a crucial task in web service based application development [5]. The methods of [6] is based on syntax matching, which didn't take the services semantics information into account. Later, in [11], the author proposed to optimize the service composition by considering QoS, which didn't consider the semantic information. And some scholar proposed that automate interactions between web services are important [7]. So the concept of ontology is proposed in [8–10], which is used to measure the semantic distance between services. However, it is a huge problem that how to build a comprehensive and standard ontology library of GIS.

To solve the problem mentioned above, we proposed a method that can compose and recommend GIS services in a semantic way. The main contributions of this article are summarized as follows:

- To get semantic relationships between services, we use Normalized Google Distance (NGD) to discover the actual inter-service invocation status.
- Considering the hierarchical structure of ArcGIS Services, a round of filtering is carried out before the network is built for reducing the retrieval time.
- In order to speed up the search time in the network, this paper use improved simulated annealing algorithm to get a relatively better solution.

The rest of this paper is organized as follows. Section 2 defines relevant concepts. Section 3 introduces the mechanism about how to construct the dynamic semantic model. Section 4 use an improved heuristic optimization algorithm to accelerate the processing of selecting DAG. Section 5 shows the result about our experimental evaluation and analysis the research. Section 6 introduces the related works of service recommendation. Finally, 7 concludes about this work.

2 Preliminaries

Definition 1 (User Requirement). *An user requirement is a tuple req=(IuP, OutP), where:*

- *IuP is a parameter set containing all user input parameters;*
- *OutP is a parameter set containing all user input parameters;*

A *req* is consist of input and output parameters set, given by the user.

Definition 2 (Directed Acyclic Graph). *A Directed Acyclic Graph, which can be performed to meet the user requirement, is a tuple DAG = (S, INV), where:*

- *S is the set of ArcGIS Services contained in this DAG, which can also regard as lots of vertices in this DAG;*
- *INV is the set of direct links, which represents the invocation relationships between these ArcGIS Services contained in this DAG;*

A *DAG* is used to describe the invocation relationship between services, which is generated to meet user requirements.

Definition 3 (ArcGIS Service). *An ArcGIS Service is a tuple s = (nm, dsc, IuP, OutP), where:*

- *nm is the name of ArcGIS Service;*
- *dsc is an explanation of the functionality of this ArcGIS Service;*
- *IuP is the set of input parameters contained in this ArcGIS Service;*
- *OutP is the set of output parameters contained in this ArcGIS Service;*

Each *s* has a specific function, which can be used to solve specific problem.

Definition 4 (Semantic Services Network Model). *A Dynamic Semantic Services Network is a triple SNetM = (S, INV, WGT), where:*

- *S are the services contained in this Dynamic Semantic Services Network;*
- *INV is the set of direct links between* ArcGIS Services*, which represents the ability that this ArcGIS Services may invoke others;*
- *WGT are the weights defined upon the direct links INV, which represent the specific possibility that an ArcGIS Services is invoked by the other; contained in this ArcGIS Services.*

There is an example in Fig. 5. Each vertex represents a service, each oriented edge represents the direction of service execution, and the value on the edge represents the semantic similarity between services.

3 Construction of Semantic Network Model

3.1 Hierarchical Structure of ArcGIS Services

ArcGIS offers advanced GIS functionalities geoprocessing tool to the users to solve the problem, which are organized in a tree structure [12]. Such a special structure can help us to remove off the unnecessary ArcGIS services to save the time and computing resources, which is shown in Fig. 1. For example, if the output parameter of the previous service(s) is vector data, there is no need to retrieve the cluster of ArcGIS services which could only use raster data as input parameters in the same subtree. Thus, using ArcGIS services tree structure can help us reduce the scope of the search and speed up the retrieve.

3.2 Services Semantic Calculation

(1) *Normalized Google Distance (NDG)*

Based on the principle that words with similar meanings appear more frequently in the browser web page, we use NGD to calculate the invocable between services. NGD is calculated by Eq. 1:

$$NGD(x,y) = \frac{max(logf(x),logf(y)) - logf(x,y)}{logM - min(logf(x),logf(y))} \tag{1}$$

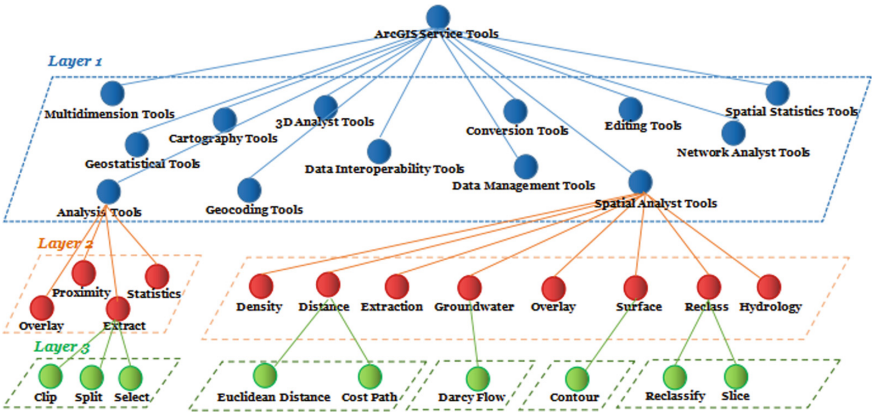


Fig. 1. ArcGIS services tree structure.

In Eq. 1, M represents the total number of pages searched by Google. $f(x)$ and $f(y)$ are the hits of the search terms x and y, respectively. $f(x, y)$ is the number of pages that appear in both x and y. If two search terms x and y never appear together on the same page, the normalized Google distance between them is infinite. Thus, the value of NDG ranges from 0 to infinity, the larger value represents the greater the distance, which meaning the greater semantic distance between two words, and vice versa.

(2) *Services Semantic Calculation*

The name of GIS services would be broken dowm into multiple words. Then use the minimum cost and maximum flow algorithm [13,14] adopted method to compute the cost between WD_{ArcNm1} and WD_{ArcNm2}. So, the names similarity can be computed by Eq. 2.

$$sim_{serNm}(ser.nm_1, ser.nm_2) = 1 - \frac{cost}{max(SizeOf(WD_{serNm1}, WD_{serNm2}))} \tag{2}$$

The text description similarity of ArcGIS Services is calculated by Eq. 3, which use *xsimilarity* [15]. In this method, words similarity in sentences (denoted as *wordSim*) and the words order (denoted as *ordSim*) are taken as parameters. The specific calculation formula is as:

$$sim_{serDsc}(ser_1.dsc_1, ser_2.dsc_2) = \xi \times wordSim + (1 - \xi \times ordSim) \tag{3}$$

The Similarity Computation between ArcGIS Services is calculated by parameters sim_{serNm} and sim_{serDsc} in Eq. 4.

$$\begin{aligned} &sim_{act}(act_1, act_2) \\ &\quad = \varrho \times sim_{serNm}(Arc_1).nm_1, ser_2.nm_2) \\ &\quad + (1 - \varrho) \times sim_{serDsc}(ser_1.dsc_1, ser_2.dsc_2) \end{aligned} \tag{4}$$

(3) *Calculating the Semantic value of Workfolw Pattern*

There are two common workflow patterns for GIS service composition: sequential workflow pattern and parallel workflow pattern, which can see in Fig. 2. The semantic value for sequential workflow pattern and parallel workflow pattern are calculated by Eqs. 5 and 6 respectively.

$$SIM_{seq} = \sum_{i=1}^{n} S_i \tag{5}$$

$$SIM_{para} = \frac{\sum_{i=1}^{n} S_i}{n} \tag{6}$$

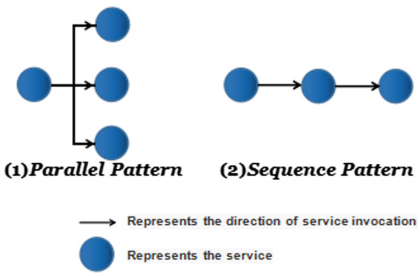


Fig. 2. Sequential workflow pattern and parallel workflow pattern.

3.3 Construction Network

(1) *Narrowing Candidate Service Set*

It would be a huge project that retrieving the entire set of ArcGIS services when we selected the candidate services. So we can use the unique tree structure of the ArcGIS services (refer to Sect. 3.1), which can help us reduce the services search space. Algorithm 1 tells about how to narrow the service candidate set.

In Algorithm 1, S can be obtained. T represents the set of all GIS services organized in a tree structure. I represents all the input parameters. S represents all candidate services which take these parameters as input parameters. First, set S copies all GIS services in T. Count the number of first level subtrees in the tree structure and assign this value to variable n (lines 1–2). By checking

Algorithm 1. Narrowing Candidate Service Algorithm

Require:
- T: all ArcGIS services set organized by tree structure.
- I: all input parameters set.

Ensure:
- S: all candidate services set.
- P: output parameters set generated by the candidate services

1: $Var\_S \leftarrow T$; $S \leftarrow \varnothing$; $P \leftarrow \varnothing$;
2: $n \leftarrow$ the number of tree categories in the first layer;
3: **for** $i = 1$: n **do**
4: **if** *I.par* $\neq$ *subtree(i).par* **then**
5: remove subtree(i) from S;
6: **end if**
7: **end for**
8: $k \leftarrow$ the number of subtree in S;
9: **for** $i = 1$: n **do**
10: **for** $j = 1$: i **do**
11: **if** subsubtree(j).par $\supseteq$ I.par **then**
12: remove subsubtree(j) from Var_S;
13: **end if**
14: **end for**
15: **end for**
16: find candidate services set S by retrieving $Var\_S(I)$
17: $S \leftarrow s(I)$;
18: $P \leftarrow S.OutP$;

the required parameter types between the subtree and I, we can remove the unmatch subtree from S. When all subtree nodes have been detected, count the number of subtrees left and assign the value to variable k (lines 3–8). For each subsubtrees in the subtree, a parameter type check is performed again. If the required parameters for the service to run in the subsubtree are more than the parameter types in I, this subsubtree is deleted (lines 9–15). And then find the services in Var_S, taking all parameters in I as input (denoted as $Var_S(I)$), and assign it to S. Finally, put the output parameters of S into P (lines 16–18).

(2) *Building Semantic Networks*

The Algorithm 2 is used to build a solution space network, from which generate the DAG and recommend it to users. Therefore, the Algorithm 2 takes user requirements req as input and the solution space network model $SNetM$ as output. First, copy the parameters in InP to P and set Var_S, INV as empty sets, where Var_S is used to store the services generated in the process and INV is used to record invocation relationships between services. Record the number of parameters in P and put them into variable n.Set parameter Var_P to null to store the generated parameters (lines 1–2). For all parameters in P, if using Algorithm 1 (denoted as $NarrSer$) finds a narrowed service set, then find the appropriate service from the narrowed service set and put it into the variable

Var_s. The output parameters of all services generated during this process are put into the variable Var_p. Record the relationship and sematic value between these services into the INV (lines 3–9). Looking for a candidate service with multiple parameters as input is similar to looking for one parameter as a candidate service (lines 10–18). Then, the number of iterations k is increased once and the parameters in the intermediate variable Var_P are copied into the P set. $NetM$ can be output if the generated parameters include the parameters required by the user or if the number of iterations is greater than the threshold. Otherwise, jump to the line 2 and continue with the above procedure (lines 19–24).

Algorithm 2. Building Semantic Networks Algorithm

Require:
- req: $req = (InP, OutP)$.

Ensure:
- $SNetM$: service network model.

1: $P \leftarrow InP$; $Var\_S \leftarrow \varnothing$; $k \leftarrow 0$; $INV \leftarrow \varnothing$;
2: $Var\_P \leftarrow \varnothing$; $n \leftarrow$ the number of parameters in P;
3: **for** $i = 1$: n **do**
4: **if** $NarrSer(P(i))$ **then**
5: $Var\_S \leftarrow$ find services in $NarrSer(P(i)).S$;
6: $Var\_P \leftarrow Var\_S.P$;
7: recording INV and $INV.SIM_{seq}$;
8: **end if**
9: **end for**
10: **for** $i = 1$: n **do**
11: **for** $j = 1$: n **do**
12: **if** $NarrSer(P(i), P(j))$ **then**
13: $Var\_S \leftarrow$ find services in $NarrSer(P(i), P(j)).S$;
14: $Var\_P \leftarrow Var\_S.P$;
15: recording INV and $INV.SIM_{para}$;
16: **end if**
17: **end for**
18: **end for**
19: $k++$; $P \leftarrow Var\_P$;
20: **if** $Var\_P \supseteq OutP \parallel k \leqslant 50$ **then**
21: $SNetM = (Var\_s, INV)$;
22: **else**
23: turn to Line 2;
24: **end if**

In this way, a dynamic semantic web is formed, which contains the DAG required by users. For instance, Fig. 5 is a $SNetM$. According to the user input and output parameters *Req.I*, Algorithms 1 and 2 are used to constructing semantic network model, which contains the DAG needed by users.

4 Recommendation System Based on Improved Simulated Annealing Algorithm

4.1 Generating New Path

To reach global optimal solution instead of local optimal solution, the simulated annealing algorithm is required to accept the new solution with a certain probability. Therefore, this section will talk about how to generate new path.

- *Dividing the Solution into Small Module:* The resulting graph solution could be divided into blocks according to workflow patterns (Fig. 2).
- *Selecting the Replacement Module:* The marked block should be replaced by the other block(s) in the SNetM. So use the random number generator to select a block, which will be replaced by other block, which is shown in Fig. 4.
- *Generating New Solution:* Replace the selected block and connect the selected block between the former block and the latter block. Consequently, a new graph result is produced, which can be seen example B in Fig. 3.

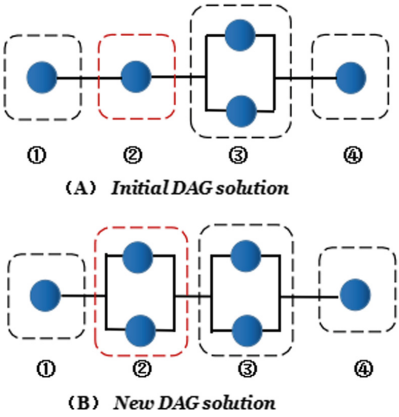


Fig. 3. Dividing into blocks.

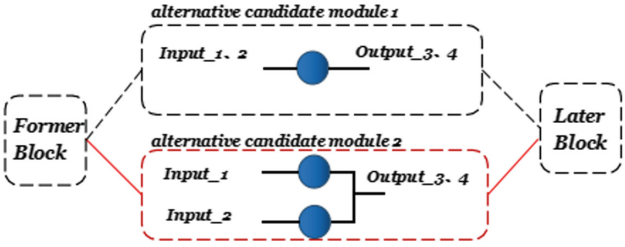


Fig. 4. Dividing into blocks.

4.2 Improved Simulated Annealing Algorithm

Algorithm 3. Building Dynamic Semantic Network Model Algorithm

Require:

- Cur_DAG: an arbitrary initial DAG.
- $coolingtable(t, \alpha, EPS, ILOOP)$: the parameters of simulated annealing algorithm were recorded
- $LIMIT$: upper limit of probability selection.
- $OLOOP$: number of external cycles.
- $Best\_DAG$: DAG recommended to user.

Ensure:

$P\_L = 0$; $P\_F = 0$;
$Best\_DAG = Cur\_DAG$; $New\_DAG = Best\_DAG$;
while 1 **do**
  for $i = 0$; $i < ILOOP$; i++ **do**
    $New\_DAG = changeSolution(Cur\_DAG)$;
    $dE = SIM_{New\_DAG}$ - SIM_{Cur_DAG};
    if $dE < 0$ **then**
      $Cur\_DAG = New\_DAG$;
      P_L=0; P_F=0;
    else
      if $exp(dE/t) > rand(0,1)$ **then**
        $Cur\_DAG = New\_DAG$;
        P_L++;
      end if
    end if
    if $P\_L > LIMIT$ **then**
      P_F++; $break$;
    end if
  end for
  if $SIM_{Cur\_DAG} < SIM_{Best\_DAG}$ **then**
    $Best\_DAG = Cur\_DAG$;
  end if
  if $P\_F > OLOOP$ || $t < EPS$ **then**
    $break$;
  end if
  $t * = \alpha$;
end while

The simulated annealing algorithm starts with the initial solution i and the control parameter t and the process is controlled by the cooling schedule, which includes the initial value of the control parameter t and its attenuation factor α, the iteration number $ILOOP$ of each t and the stop condition EPS in Algorithm 3. Cur_DAG is a result randomly found from the network that meets the user's input and output requirements. The $Best\_DAG$ represents the DAG

which can better meet the user's requirement. Coolingtable represents a set of parameters that control the progress of an algorithm.

Parameters P_L and P_F are set to record the times of receiving bad results in a certain stage of annealing process and the times of this process respectively. Temporarily set $Best\_DAG$ and New_DAG to be the same value as the Cur_DAG (lines 1–2). Use the algorithm *changeSolution()* to generate the New_DAG and calculate the semantic value difference between the two path (denoted as dE). If the semantic values of New_DAG (denoted as SIM_{New_DAG}) is higher than that of Cur_DAG (denoted as SIM_{Cur_DAG}), the New_DAG will be accepted as the Cur_DAG. Otherwise, the above operation is carried out with a certain probability to avoid falling into local optimal and increment the value of the P_L by 1. If PL is greater than LIMIT, jump out of the loop (lines 3–19). After the above process, if the SIM_{Cur_DAG} is higher than SIM_{Best_DAG}, replace the Cur_DAG with $Best\_DAG$ (lines 20–22). Then, determine whether the program is completed by judging whether P_F is greater than $OLOOP$ or the temperature t reaches the minimum value EPS. If the exit condition is not reached, use attenuation coefficient α to cool the temperature and continue the cycle (lines 20–27). As a result, the DAG is found in the semantic web in Fig. 5 and recommended it to the user.

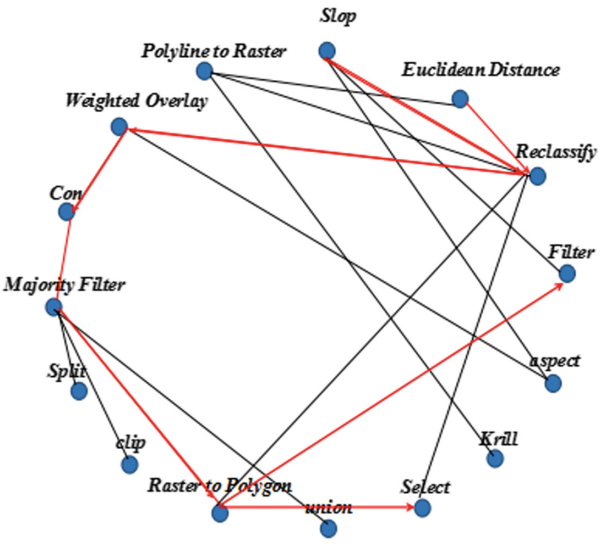


Fig. 5. The dynamic semantic network model.

5 Experiment

5.1 Dataset Description and Precision

In order to verify the effectiveness of our proposed method, we use the Java language to test the method and use MySQL database to store the data, which is conducted on a desktop with an Inter (R) Core (TM) i7-3770 CPU @ 3.40 GHz, 8.00 GB memory, and a 64-bit Windows 10 operating system.

The data uses 300 geoprocessing services organized by tree from ArcGIS Toolbox. In addition, 112 DAG rules, which represents the invocation rules between services based on different requirements, are found from numerous communities such as CSDN.

Our experimental results will be evaluated by precision and running times. The precision is computed as follows:

$$precision = \frac{DAG_P \bigcap DAG_R}{N} \tag{7}$$

In Eq. 7, DAG_P represents the DAG generated by our method and the DAG_R represent the right DAG that really meets the requirements of the user in the DAG rule set. N is the operation number contained in a DAG_P. To get a more correct value of precision, we proceed experiment with different user requirement for 112 times. The average precision is 76.4%.

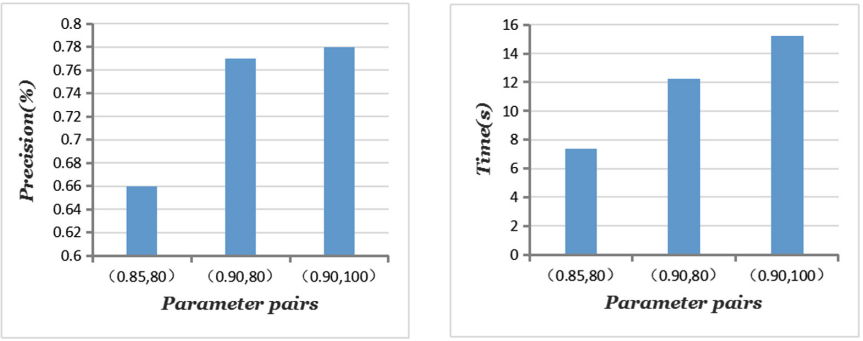
5.2 Impact of Parameters in Cooling Table

To investigate the effect of Cooling Table parameters in the proposed method in Algorithm 3. As show in Fig. 6, we set the parameters in the Cooling Table to three different sets of values and compared them.

The cooling Table contains four parameters t_0, α, EPS and $ILOOP$. Normally, the values of t_0 and α are 1000 and 0, so we only consider α and $ILOOP$, which are denoted as (α, $ILOOP$) in Fig. 6. α represents the rate of temperature decay and $ILOOP$ represents the number of temperature drops in the same stage, which are mutually dependent. Although the higher value of α represents the better ability to cool the temperature in Fig. 6(a). It will also take a lot of times. For the same reason that higher value of $ILOOP$ will cost more computing resource in Fig. 6(b), the value of parameter $ILOOP$ should not be very large. Therefore, the experimental accuracy is relatively high and the computation time consumption is relatively small, when α is 0.9 and $ILOOP$ is 80.

5.3 Compare with Other Method

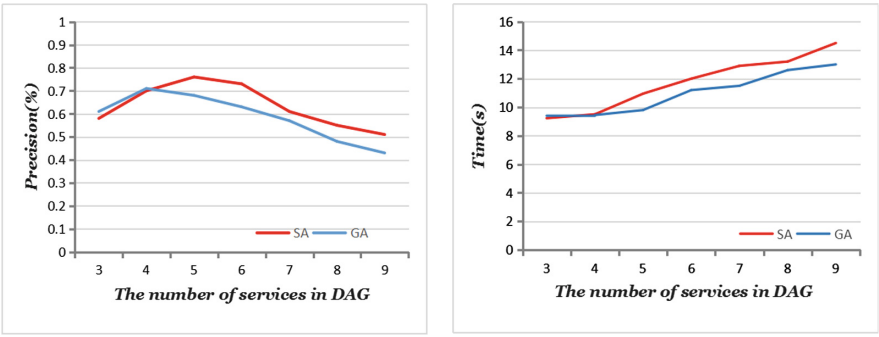
The number of services is varies from 3 to 9, so we consider the impact of the number of services on the service composition. We compare our method with the method proposed by the author in [16], which used the GA as heuristic optimization algorithm.



(a) Effect of Cooling Table on precision (b) Effect of Cooling Table on Running time

Fig. 6. Influence of parameters in cooling tables.

Figure 7 shows that the precision value reaches the peak value when the service number of DAG is from 4 to 6. The reason for this phenomenon are as follows. If a large number of services need to be found in the required DAG, but some detailed or transitional services may not be found during the actual execution, thus affecting the precision. That is the reason why precision decreases as the number of services increases. Because comparison method can only get services chain, the precision of our method is higher than compared one in Fig. 7(a). And as the number of services in the DAG increases, the service composition consumes more time. The reason why our method takes more time is that our graph structure solution is much more complex than chain structure in Fig. 7(b). But the usability of our proposal is much higher than the comparison one.



(a) Compare Precision for SA and GA (b) Compare Running time for SA and GA

Fig. 7. The precision and run time of proposed method.

6 Related Work

6.1 Service Composition Technology

Web services composition technology, aiming to provide added values by loosely coupling web services, has been used to efficiently find near-optimal composite services to satisfy users' requirements reasonably well [17]. The syntax-based service composition depends on the matching between selected keywords and Web service description [18], which takes little account of the semantics of web services. To get the concepts relationship, scholar use a certain criterion to measure the semantic distance in [19]. In [10], the authors proposed a novel Permutation-based Multifactorial Evolutionary Algorithm to solve the fully automated semantic service composition problem for diverse user segments with different QoSM preferences. And the principle of [20,21] is that using ontology as a fundamental criterion to measuring the concept distance of the user's requirement and the services. The method of using ontology is not suitable for direct application in GIS domain, cause it's a hard work to construct the ontology. It is obviously that the accuracy of web services semantic annotations will significantly improve the effectiveness of the web service discovery, recommendation and composition [22]. In [11], the author proposed an invocation-based technique to verify the QoS accuracy by using annotations.

6.2 GIS Services Composition

The GIS domain service composition can be divided into three categories: semi-automated GIS services composition, syntax-based GIS services composition, and semantic GIS services composition. In [23], the author proposed the registration-binding-lookup mechanism, which is a semi-automated approach to service composition recommendation. In order to provide services to user automatically, some authors suggest that taking services context into consider. In [24], the authors proposed an active proxy, which can regard service context and user's requirement, extract useful information and send it to the server. But this method can only used in location-based service. In [25], the authors mapped the OWS input/output message to WSRF ResourceProperties, which could bring higher efficient. But this method doesn't incorporate many useful WSRF function. Besides, high performance data transfer is a challenge in GIS service.

7 Conclusion

The enhancement of Internet technologies has improved the technology in GIS services discovery, composition and recommendation. It is becoming increasingly important to combine GIS services to help users solve a various problems. Therefore, in this paper, we discusses the related technologies of service composition in GIS and computer fields, and analyzes the principles of these technologies. We

find that effective use of semantic information between services can improve the quality of service composition, which could meet the users' requirement better. To solve this problem, by using the tree organization structure of ArcGIS service, we can quickly select the set of services that meet the requirements according to the syntax matching relationship between services. To further explore the semantic correlation between services we use the NDG to build the dynamic semantic network. Then simulated annealing algorithm is used to find the DAG with high semantic value and recommend it to the user. Experiments show that our method could recommend a meaningful DAG with higher precision.

References

1. Scheider, S., Ballatore, A., Lemmens, R., Hartmann, S.: Finding and sharing GIS methods based on the questions they answer. Int. J. Digit. Earth **12**, 594–613 (2019). https://doi.org/10.1080/17538947.2018.1470688
2. Linn, K.N.Z., Lupin, S., Linn, H.H.: Analysis of the effectiveness of fire station locations using GIS-model. In: 2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), Saint Petersburg and Moscow, pp. 1840–1843. (2019). https://doi.org/10.1109/EIConRus.2019.8657048
3. Di, L.: Distributed geospatial information services-architectures, standards, and research issues. Int. Arch. Photogramm. Remote. Sens. Spat. Inf. Sci. (Part 2) (2004). https://doi.org/10.4018/978-1-60960-192-8.ch001
4. Sadeghiram, S., Ma, H., Chen, G.: Distance-guided GA-based approach to distributed data-intensive web service composition. arXiv preprint arXiv:1901.05564. Arxiv (2019). https://doi.org/10.1145/3319619.3322015
5. Kamath, S., Ananthanarayana, V.S.: Discovering composable web services using functional semantics and service dependencies based on natural language requests. Inf. Syst. Front. **21**, 175–189 (2019). https://doi.org/10.1007/s10796-017-9738-2
6. Zhang, S., Wang, F.: GIS geoprocessing services search based on breadth-first reverse share pruning AND/OR tree algorithm. In: 2014 10th International Conference on Natural Computation (ICNC), vol. 12, pp. 850–855. IEEE (2014). https://doi.org/10.1109/ICNC.2014.6975949
7. Farzi, P., Akbari, R., Bushehrian, O.: Improving semantic web service discovery method based on QoS ontology. In: 2017 2nd Conference on Swarm Intelligence and Evolutionary Computation (CSIEC), pp. 72–76. IEEE (2017). https://doi.org/10.1109/CSIEC.2017.7940175
8. Yue, P., Di, L., Yang, W., Yu, G., Zhao, P.: Semantics-based automatic composition of geospatial web service chains. Comput. Geosci. **33**, 639–665 (2007). https://doi.org/10.1016/j.cageo.2006.09.003
9. Zaharia, R., Vasiliu, L., Hoffman, J., Klien, E.: Semantic execution meets geospatial web services: a pilot application. Trans. GIS **12**, 59–73 (2008). https://doi.org/10.1111/j.1467-9671.2008.01135.x
10. Ţucăr, L., Diac, P.: Semantic web service composition based on graph search. Procedia Comput. Sci. **126**, 116–125 (2018). https://doi.org/10.1016/j.procs.2018.07.215
11. Huang, K., Zhang, J., Tan, W., Feng, Z., Chen, S.: Optimizing semantic annotations for web service invocation. IEEE Trans. Serv. Comput. **12**, 590–603 (2016). https://doi.org/10.1109/TSC.2016.2612632

12. Kulawiak, M., Dawidowicz, A., Pacholczyk, M.E.: Analysis of server-side and client-side web-GIS data processing methods on the example of JTS and JSTS using open data from OSM and geoportal. Comput. Geosci. **129**, 26–37 (2019). https://doi.org/10.1016/j.cageo.2019.04.011
13. Stein, C., Wein, J.: Approximating the minimum-cost maximum flow is P-complete. Inf. Process. Lett. **42**, 315–319 (2019). https://doi.org/10.1016/0020-0190(92)90229-O
14. Zhou, Z., Cheng, Z., Zhang, L.-J., Gaaloul, W., Ning, K.: Scientific workflow clustering and recommendation leveraging layer hierarchical analysis. IEEE Trans. Serv. Comput. **11**, 169–183 (2018). https://doi.org/10.1109/TSC.2016.2542805
15. Zhou, Z., Cheng, Z., Ning, K., Li, W., Zhang, L.-J.: A sub-chain ranking and recommendation mechanism for facilitating geospatial web service composition. Int. J. Web Serv. Res. (IJWSR) **11**, 52–75 (2014). https://doi.org/10.4018/ijwsr.2014070103
16. Hu, B., Zhou, Z., Cheng, Z.: Web services recommendation leveraging semantic similarity computing. Procedia Comput. Sci. **129**, 35–44 (2018). https://doi.org/10.1016/j.procs.2018.03.041
17. Wang, C., Ma, H., Chen, G., Hartmann, S.: A memetic NSGA-II with EDA-based local search for fully automated multiobjective web service composition. In: Genetic and Evolutionary Computation Conference Companion, vol. 11, pp. 52–75. ResearchGate (2019). https://doi.org/10.1145/3319619.3321937
18. Cheng, B., Li, C., Zhao, S., Chen, J.: Semantics mining & indexing-based rapid web services discovery framework. IEEE Trans. Serv. Comput., 1. (2018). https://doi.org/10.1109/TSC.2018.2831678
19. Wang, C., Ma, H., Chen, G., Hartmann, S.: Evolutionary multitasking for semantic web service composition, pp. 2490–2497. arXiv preprint arXiv:1902.06370. arxiv.org (2019). https://doi.org/10.1145/2481492.2481495
20. Arul, U., Prakash, S.: A unified algorithm to automatic semantic composition using multilevel workflow orchestration. Clust. Comput. **126**, 1–22 (2018). https://doi.org/10.1007/s10586-018-2604-2
21. Fellah, A., Malki, M., Elci, A.: A similarity measure across ontologies for web services discovery. In: Web Services: Concepts, Methodologies, Tools, and Applications, pp. 859–881. IGI-Global (2019). https://doi.org/10.4018/978-1-5225-7501-6.ch047
22. Derczynski, L., Maynard, D., Aswani, N., Bontcheva, K.: Microblog-genre noise and impact on semantic annotation accuracy. In: Proceedings of the 24th ACM Conference on Hypertext and Social Media, pp. 21–30. Dl.ACM (2013). https://doi.org/10.1145/2481492.2481495
23. Wenjue, J., Jianya, G., Bin, L.: GIS integration and interoperability based on GIS service chain. In: Proceedings of the 2005 IEEE International Geoscience and Remote Sensing Symposium, IGARSS 2005, vol. 7, pp. 4962–4965. IEEE(2005). https://doi.org/10.1109/IGARSS.2005.1526788
24. Li, X., Shin, W., Li, L., Yoo, S.B.: GIS web service using context information in mobile environments. In: Gavrilova, M., et al. (eds.) ICCSA 2006. LNCS, vol. 3980, pp. 895–903. Springer, Heidelberg (2006). https://doi.org/10.1007/11751540_97
25. Gui, Z., Song, K.: Building improved GIS service based on WSRF. In: 2008 International Conference on Internet Computing in Science and Engineering, vol. 33, pp. 274–277. IEEE (2008). https://doi.org/10.1109/ICICSE.2008.12