# SEMBIOSPHERE: A SEMANTIC WEB APPROACH TO RECOMMENDING MICROARRAY CLUSTERING SERVICES

KEVIN Y. YIP[1], PEISHEN QI[1], MARTIN SCHULTZ[1], DAVID W. CHEUNG[5] AND KEI-HOI CHEUNG[1,2,3,4]

[1] *Computer Science,* [2] *Center for Medical Informatics,* [3] *Anesthesiology,* [4] *Genetics, Yale University, New Haven, Connecticut, USA,*
[5] *Computer Science, University of Hong Kong, Hong Kong*

Clustering is a popular method for analyzing microarray data. Given the large number of clustering algorithms being available, it is difficult to identify the most suitable ones for a particular task. It is also difficult to locate, download, install and run the algorithms. This paper describes a matchmaking system, SemBiosphere, which solves both problems. It recommends clustering algorithms based on some minimal user requirement inputs and the data properties. An ontology was developed in OWL, an expressive ontological language, for describing what the algorithms are and how they perform, in addition to how they can be invoked. This allows machines to "understand" the algorithms and make the recommendations. The algorithm can be implemented by different groups and in different languages, and run on different platforms at geographically distributed sites. Through the use of XML-based web services, they can all be invoked in the same standard way. The current clustering services were transformed from the non-semantic web services of the Biosphere system, which includes a variety of algorithms that have been applied to microarray gene expression data analysis. New algorithms can be incorporated into the system without too much effort. The SemBiosphere system and the complete clustering ontology can be accessed at `http://yeasthub2.gersteinlab.org/sembiosphere/`.

## 1. Introduction

As modern life sciences research involves high throughput bio-technologies (e.g., sequencing, DNA microarray, and mass spectrometry), a vast quantity of data is being generated, which needs to be stored, analyzed, visualized, and interpreted. As a result, a plethora of analyzing tools have been developed and made accessible via the Internet. Subsequently, it has become difficult to locate the tools that are relevant to the research questions at hand. The current web technologies rely heavily on the use of keyword-based searches in locating web resources, which suffers from the problem of sensitivity and specificity. In addition, even if the relevant software tools are found, users may still experience problems in

downloading, installing, and running the programs. It is also difficult for users to keep track of the updates, such as bug fixes and the addition of new features.

### 1.1. *Microarray Cluster Analysis*

For microarray data, clustering is a popular data analysis method. A large collection of clustering algorithms have been developed, published, and made available in different forms through many web sites. Some of them are available as downloadable software that can be installed on the client computer. For example, Eisen's cluster program[18] can be downloaded from the Eisen Lab website and run on a Windows PC. Some algorithms are available as web server applications that allow users to submit their microarray data to the server on which the cluster analysis is performed (e.g. EPCLUST [3]). More recently, some cluster analysis algorithms have been implemented as web services using SOAP[10], which allow direct machine invocation over the Internet. Biosphere[17] is a representative example. The advantage of publishing the algorithms as web services is that all programs are invoked through the same XML-based interface, regardless of what languages they are written in, and what platforms they are running on.

The availability of a large number of algorithms provides more options to choose from, but at the same time also makes it difficult for one to determine the most suitable algorithm for the current task. This problem is especially true for many microarray experimentalists who may not know much about the technical details of the algorithms. Although reports[16] have been published on the evaluation and comparison of different algorithms, it relies on the human users to study many such reports carefully to figure out which algorithms perform better under different circumstances.

To address this problem, we employ the latest semantic web technologies to transform the basic clustering web services of Biosphere into semantic web services. The semantic layer describes what the algorithms are and how they perform in addition to how they can be invoked. It is thus possible for machines to recommend which algorithms to use based on the user requirement inputs and data properties. In the following we describe SemBiosphere, a matchmaking system that provides such recommendations.

## 2. SemBiosphere

Figure 1 shows the overall system architecture of SemBiosphere. First, there are algorithm providers who expose their available algorithms as web services. As discussed, the providers have the flexibility to choose the programming language and the running platform, as long as they follow the XML standards to describe their programs. The descriptions involve two parts: a lower part based on

WSDL[11] for specifying the input/output types, and an upper part for specifying semantic descriptions. We developed an ontology for clustering algorithms using OWL[8], and described the algorithms using the ontology. The two types of descriptions are combined by the latest OWL-S language[7], and are stored in a central RDF[9] repository for querying. The recommendation system is a web application that can be accessed using web browsers. It accepts both the data and the matching requirements from users through an HTML interface. It then performs filtering and ranking to produce a sorted list of algorithms according to the matching scores. The users may pick any number of them to use. The system provides a form for inputting the parameters. It then executes the algorithms, and sends an email notification to the users when the results are ready.

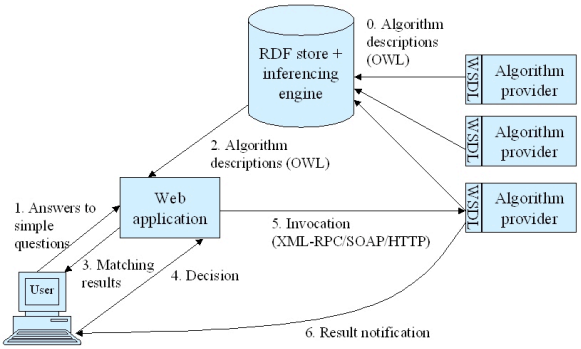



Figure 1. SemBiosphere system architecture.

Compared with other popular microarray software and systems such as Bioconductor [1] and EPCLUST [3], the SemBiosphere system is more extensible as it can integrate programs written in different languages and by different groups dynamically. The potentially larger variety of algorithms allows experimentalists to try out newer methods that may suit their specific needs more. Compared with some keyword-based searching systems, the semantic web approach allows for much more focused and structured queries and the possibility to answer questions based on logical inference rather than text associations.

There are some large-scale projects aiming at facilitating complex *in silico* experiments that adopt semantic web technologies, including myGrid [5] and BioMOBY [2]. The current study is related to, but not covered by, these projects. It is an interesting future work to see how the ideas of the current project can be applied to their general frameworks.

Below we describe in more detail some of the core components of the system.

### 2.1. *Ontology for Describing Clustering Algorithms*

Figure 2 shows the class hierarchy of our clustering algorithm ontology. It covers three aspects of clustering algorithms: algorithm types, the kinds of data that can be handled, and time complexities.

Algorithms are classified into common categories such as hierarchical, partitional and density-based. The ontology is extensible so that more categories can be added. The categorization is useful for making some quick decisions on which clustering algorithms to use. For example, if a user wants to have a dendrogram as part of the clustering output (such as those produced by TreeView[18]), hierarchical algorithms are by default more suitable.

Each algorithm is allowed to belong to multiple classes. For example, the algorithm ORCLUS[14] involves both a hierarchical phase and a partitional phase in each iteration. Some classes also have subclasses. For example, hierarchical clustering algorithms are further subdivided into agglomerative and divisive algorithms. Besides these main categories, some categories were also defined for special classes of clustering algorithms, such as projected clustering[22] and biclustering algorithms[15]. These categories are orthogonal to the main categories. Each algorithm belongs to at least one main category and any number of these special categories. For example, PROCLUS[13] is both a partitional clustering algorithm, and a projected clustering algorithm. These special categories are useful for matching some special requirements. For example, when the number of experiments (columns) in a dataset is large in compared to the number of genes (rows), algorithms that find clusters in subspaces (e.g. projected clustering algorithms) may perform better[22].

Another important classification of the algorithms is the kind of data attributes that they can handle. For example, some microarray data are discretized, such that the attributes are categorical. We selected a relevant subset of the attribute types in Kaufman and Rousseeuw[21] to define a hierarchy of attribute types. Some Biosphere implementations of the algorithms are modified versions that can handle some attributes that the original algorithms cannot handle. The extra capability is provided by defining special similarity functions. For example, if an algorithm relies on the distance between different expression values in the clustering process, by default they cannot handle categorical attributes in which the distance between different values is not defined. However, if a similarity function can also define the distance between different categorical values, then the algorithm can also work on categorical attributes. In view of this, we defined different similarity functions, each associated with a list of attribute types that it can handle. Each clustering algorithm then picks the similarity functions that it can use, from which the types of attributes that it

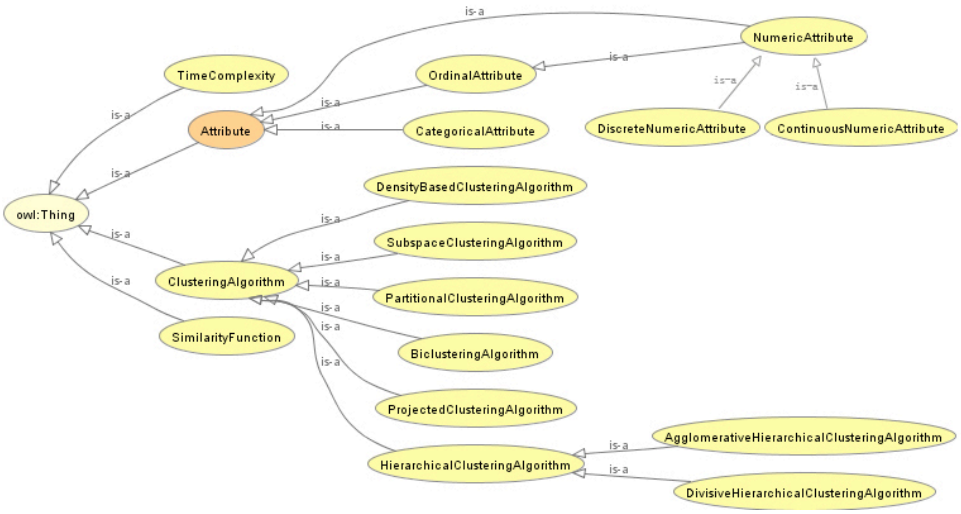



Figure 2. Clustering algorithm ontology.

can handle are defined.

Finally we defined the time complexity of each algorithm. It turned out to be very difficult to define such a category, as time complexities are expressed in functions, and may involve many variables specific to each algorithm. To make things simple, we defined several coarse complexity classes such as constant time, $O(n)$ time and $O(nlogn)$ time, and requires each algorithm to pick the tightest ones with respect to the number of experiments and genes in a dataset. This is certainly not an ideal way to specify the time complexities. Also, time complexity alone does not directly imply the actual running time of the algorithms. However, the current categories are already very useful in some basic matching. For example, if a dataset contains more than 10000 genes and the user is only willing to wait for seconds, an algorithm with cubic time complexity with respect to the number of genes is unlikely to be a good choice.

### 2.2. *Semantic Clustering Web Services*

We use the KPrototype algorithm[20] as an example to explain what have to be specified for a semantic web service. Figure 3 shows its descriptions in OWL-S, which contain the following parts:

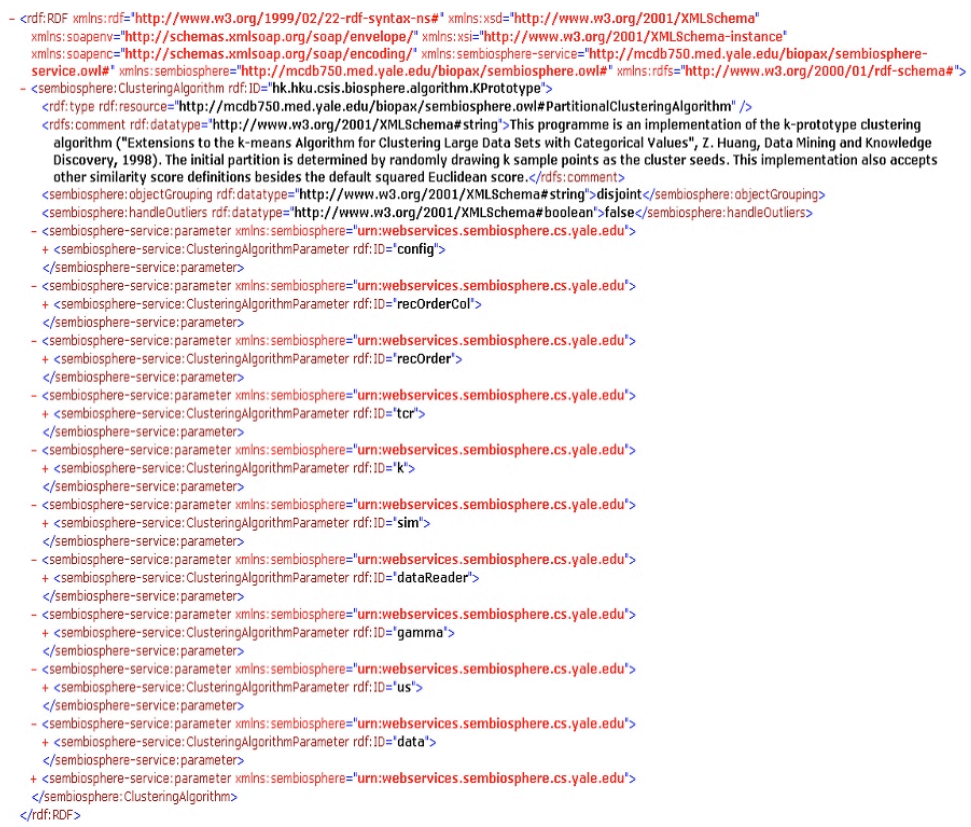
- The *service* part organizes all other parts of a web service description. The "KPrototypeClusteringService" is described by "KPrototypeClusteringProfile", presents "KPrototypeClusteringProcess" and supports

"KPrototypeClusteringGrounding". The service can be thought of as an API declaration for an entry point that a service provider wants to make accessible.

- The *profile* part tells us "what the service does"; that is, it gives the type of information needed by a service-seeking agent to determine whether the service meets its needs. It also provides the contact information of the service provider. Here, we defined "KPrototypeClusteringProfile" as an instance of "ClusteringService", which is linked to our clustering algorithm ontology by its "algorithm" property. An agent searching for clustering services can first determine the appropriate algorithms it desires by examining the ontology, and then get to the services via the "algorithm" relationships.
- The *process model* tells "how the service works". Our clustering service processes are composed of two alternative atomic processes: *getAlgorithmMetaData* and *execute*. *getAlgorithmMetaData* requires no inputs and returns the metadata description, such as the type and parameters, of the algorithm. *execute* requires three inputs: dataset URL, user email and the set of parameter values. It responds with a task ID and a security token for the retrieval of clustering results when the service is successfully executed.
- *Service grounding* specifies the details of how an agent can access a service. Here we use SOAP RPC as our communication protocol and ground our services on WSDL to specify the port used in contacting the services. Each atomic process is grounded to a WSDL operation and its inputs and outputs to the message parts of that operation. When an atomic process is invoked, the corresponding operation is called in the remote server. Inputs and outputs are transformed between semantic OWL documents and SOAP messages using the XSLT[12] stylesheet embedded in the grounding part of the OWL-S file.

### 2.3. *Matchmaker*

The matchmaking subsystem takes as input a microarray dataset and some other user requirements for a clustering task (Figure 4). In order not to request users to know too many technical details about the algorithms, the requirements are entered through answering several very simple questions. The system performs the matching and returns a list of algorithms in descending order of their matching scores (Figure 5). Matching details are provided at the bottom of the page for explaining how the matching was performed. From the list, the users can choose one or more algorithms to cluster the data. The request will be sent to



```
- <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:sembiosphere-service="http://mcdb750.med.yale.edu/biopax/sembiosphere-
    service.owl#" xmlns:sembiosphere="http://mcdb750.med.yale.edu/biopax/sembiosphere.owl#" xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
  - <sembiosphere:ClusteringAlgorithm rdf:ID="hk.hku.csis.biosphere.algorithm.KPrototype">
      <rdf:type rdf:resource="http://mcdb750.med.yale.edu/biopax/sembiosphere.owl#PartitionalClusteringAlgorithm" />
      <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">This programme is an implementation of the k-prototype clustering
        algorithm ("Extensions to the k-means Algorithm for Clustering Large Data Sets with Categorical Values", Z. Huang, Data Mining and Knowledge
        Discovery, 1998). The initial partition is determined by randomly drawing k sample points as the cluster seeds. This implementation also accepts
        other similarity score definitions besides the default squared Euclidean score.</rdfs:comment>
      <sembiosphere:objectGrouping rdf:datatype="http://www.w3.org/2001/XMLSchema#string">disjoint</sembiosphere:objectGrouping>
      <sembiosphere:handleOutliers rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">false</sembiosphere:handleOutliers>
    - <sembiosphere-service:parameter xmlns:sembiosphere="urn:webservices.sembiosphere.cs.yale.edu">
      + <sembiosphere-service:ClusteringAlgorithmParameter rdf:ID="config">
      </sembiosphere-service:parameter>
    - <sembiosphere-service:parameter xmlns:sembiosphere="urn:webservices.sembiosphere.cs.yale.edu">
      + <sembiosphere-service:ClusteringAlgorithmParameter rdf:ID="recOrderCol">
      </sembiosphere-service:parameter>
    - <sembiosphere-service:parameter xmlns:sembiosphere="urn:webservices.sembiosphere.cs.yale.edu">
      + <sembiosphere-service:ClusteringAlgorithmParameter rdf:ID="recOrder">
      </sembiosphere-service:parameter>
    - <sembiosphere-service:parameter xmlns:sembiosphere="urn:webservices.sembiosphere.cs.yale.edu">
      + <sembiosphere-service:ClusteringAlgorithmParameter rdf:ID="tcr">
      </sembiosphere-service:parameter>
    - <sembiosphere-service:parameter xmlns:sembiosphere="urn:webservices.sembiosphere.cs.yale.edu">
      + <sembiosphere-service:ClusteringAlgorithmParameter rdf:ID="k">
      </sembiosphere-service:parameter>
    - <sembiosphere-service:parameter xmlns:sembiosphere="urn:webservices.sembiosphere.cs.yale.edu">
      + <sembiosphere-service:ClusteringAlgorithmParameter rdf:ID="sim">
      </sembiosphere-service:parameter>
    - <sembiosphere-service:parameter xmlns:sembiosphere="urn:webservices.sembiosphere.cs.yale.edu">
      + <sembiosphere-service:ClusteringAlgorithmParameter rdf:ID="dataReader">
      </sembiosphere-service:parameter>
    - <sembiosphere-service:parameter xmlns:sembiosphere="urn:webservices.sembiosphere.cs.yale.edu">
      + <sembiosphere-service:ClusteringAlgorithmParameter rdf:ID="gamma">
      </sembiosphere-service:parameter>
    - <sembiosphere-service:parameter xmlns:sembiosphere="urn:webservices.sembiosphere.cs.yale.edu">
      + <sembiosphere-service:ClusteringAlgorithmParameter rdf:ID="us">
      </sembiosphere-service:parameter>
    - <sembiosphere-service:parameter xmlns:sembiosphere="urn:webservices.sembiosphere.cs.yale.edu">
      + <sembiosphere-service:ClusteringAlgorithmParameter rdf:ID="data">
      </sembiosphere-service:parameter>
    + <sembiosphere-service:parameter xmlns:sembiosphere="urn:webservices.sembiosphere.cs.yale.edu">
    </sembiosphere:ClusteringAlgorithm>
</rdf:RDF>
```


Figure 3. OWL-S representation of the KPrototype clustering web service.

the chosen clustering web services, and the users will be notified by email when the results are ready, which can be viewed on a result page with a standard colored graph for visualizing the clusters (Figure 6).

The current matching system adopts a rule-based approach. Matching is performed based on two types of predefined rules: filtering rules and scoring rules. Filtering rules define which algorithms cannot be used in the current clustering task. For example, if the dataset contains a certain kind of attributes that an algorithm cannot handle, the algorithm will be filtered. In general, a set of preconditions can be specified for each algorithm so that it can be used for a clustering task only if all the preconditions are satisfied.

The algorithms that remain are evaluated by the scoring rules. Each rule takes into account an evaluation criterion and gives a score for each algorithm. For example, one rule evaluates the speed performance of the algorithms in terms

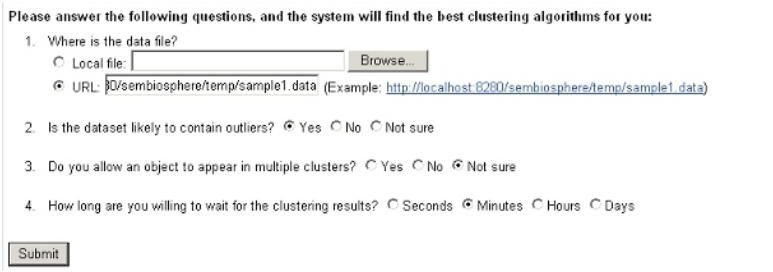



Figure 4. User input.

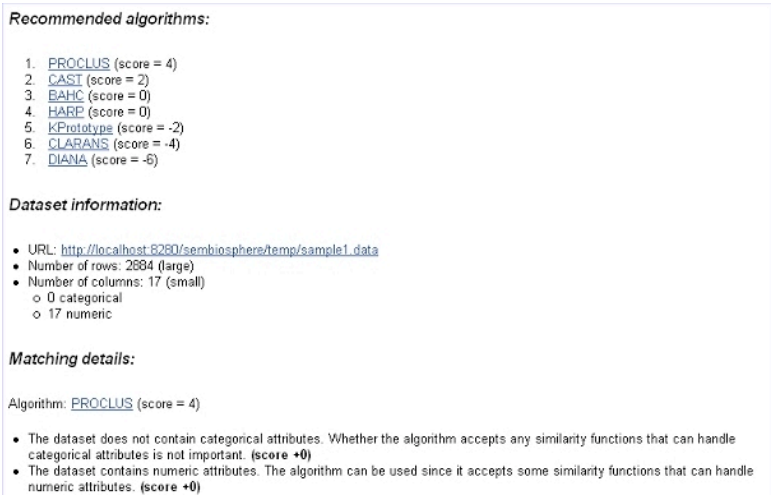



Figure 5. Matched algorithms and match description.

of the number of genes. Basically, algorithms with higher time complexities will receive lower scores. However, the importance of the rule depends on the size of the dataset as well as the time that the user is willing to wait. If the dataset is large, an algorithm with $O(n^2)$ time complexity may run significantly faster than one with $O(n^3)$ time complexity. Yet if the dataset is small, the difference may be negligible and the scores for the two algorithms will not have a large difference. Similarly, if the user wants the clustering result be ready within seconds, the execution time is very important and the rule will give a large score difference between algorithms with different time complexities. But if the user is willing to wait for days, the rule will give similar scores to the algorithms.

The final score for an algorithm is a weighted sum of the individual scores. The weights depend on the dataset, the algorithm, and also the user require-

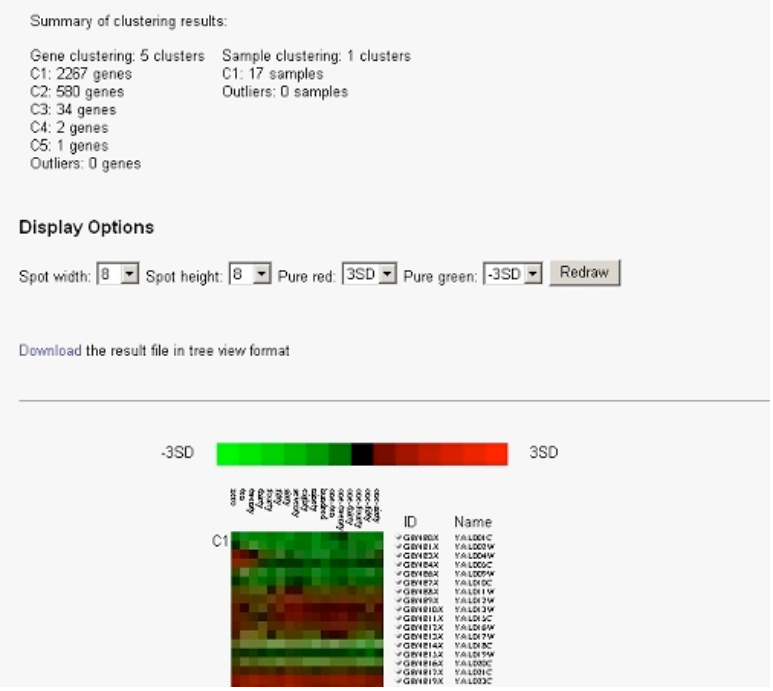



Figure 6. Visualization of clustering results.

ments. Currently the weights are set by fixed rules, but it is also possible to learn the personalized weights by collecting user feedbacks after actually running the algorithms. A difficulty is that for a reasonably large set of rules, the number of training examples required is huge. The learning may therefore need to be based on the feedbacks of all the users, but with a special emphasis on the part of the particular user. We leave this learning capability as a future research direction.

## 3. Discussion

While we considered only clustering algorithms in this study, many concepts are brought from the more general data analysis or machine learning domain, such as the hierarchy of attribute types. It is a good idea to have the ontology general enough to be able to merge with other machine learning ontology. However, over-generalization may make things complicated while not gaining much for the current goal. Also, if the logic inferencing capability is to be heavily used, a complicated ontology is more prone to errors and may incur a large computational overhead. Our suggestion is to keep the ontology minimal and

flexible. For example, we did not attempt to include all the possible clustering algorithm types in the ontology, but we kept in mind that in future more types might be defined, so the codes were written in ways such that minimal changes are required when new types are added. This flexibility is very important in a fast growing field such as bioinformatics, since new concepts evolve rapidly. For instance, the concept of finding the corresponding subspace of each cluster has become popular only in recent years due to the production of extremely high dimensional datasets from sources such as microarray experiments. Rather than being static, the ontology needs to grow dynamically with time.

The above discussion is also related to another question: by whom the algorithm descriptions should be defined or changed? One possibility is to have the classes, properties and the relationships involved defined in a commonly agreed ontology. Each group then defines the property values of their own algorithms accordingly. The problem is that since the values directly affect the chance that the corresponding services being chosen, there has to be an objective and fair way to determine the values, which is not easy if this is done by each group individually. On the other hand, it may not be possible to have the values completely determined by a central committee since the performance of the services may be dependent on some information owned only by the service provider, such as the CPU load of their servers. Some monitoring and trust mechanisms are required in such situations. Feedbacks from the user could also help fine-tune the matching mechanism to penalize imprecise descriptions.

The current ontology is mainly based on the computational issues of the algorithms. Since the system is targeted for biologists, there has to be some means to translate their high-level, domain-specific requirements into these low-level technicalities. The current implementation performs this by providing a simple user interface with several non-technically deep questions. For the system to be widely used by biologists, it has to incorporate more biological knowledge. Our proposal is to keep a low-level ontology for computational issues, but at the same time add a high-level ontology for specifying domain-specific requirements. It would then be possible to define more formal rules for the matching between algorithm characteristics and user requirements. While there are existing bio-ontologies related to microarray experiments (e.g. MGED [4]), they are mainly for describing data properties and experiments. Such ontologies are necessary, but not sufficient for specifying the user requirements. The high-level ontology also needs to include ways for describing analysis tasks. Current efforts in this area (e.g. PMML [6]) are not tailored for biological sciences. A biologists-friendly version would be needed for capturing the specific goals and common practices of the field.

We have also observed some fundamental issues at the web service layer. One

of them is the need to transfer large amount of data through the network. In our case, the dataset needs to be transmitted between the client and both the recommendation system and the clustering web services. Caching can alleviate the problem if different algorithms are applied to the same dataset, but cannot completely solve the problem. One solution is to run the algorithms at a machine close to the data, so that large data moves are localized to a small region, preferably a local area network. To achieve this, either the services have to be mirrored at different geographically areas, or there has to be some ways to "download" the service to run locally. The concept is similar to Java applets that are downloaded to run at client browsers, but here the concept is generalized in that the programs may also be run on an intermediate powerful server. In order to make this approach possible, the service programs have to be divided into two halves so that only the downloaded part works directly on the data, and only the server part connects directly to server-side resources such as backend databases. In our case, this approach seems feasible since the clustering algorithms are Java classes that can run on any machines with a Java virtual machine, and they do not need to access any server side resource directly. At the same time, server side activities such as authentication and task queuing do not need to deal with the data directly. Whether the approach is generally applicable in other situations is yet to be determined.

## 4. Conclusion

In this paper, we described how the semantic web approach could be used to address the problem of choosing among the many clustering algorithms available for analyzing microarray data. We demonstrated the importance of semantic web service descriptions (through the use of ontology), and how the semantic web technologies can be used to build a matchmaking system that can make recommendations to users on which clustering web services to use according to their requirements. While our results are preliminary and there are yet issues to be addressed, we believe this kind of web-services-based distributed and collaborative systems will become the trend of the near future. This project can be viewed as a small showcase for exploring the impact of semantic web in modern life sciences research.

## 5. Acknowledgement

We would like to thank Mark Gerstein, Andrew Smith and other members of the Gerstein Lab for their comments on a preliminary version of the system. This work was supported in part by NIH grant K25 HG02378 from the National Human Genome Research Institute, and NSF grant BDI-0135442.

## References


1. Bioconductor. http://www.bioconductor.org/.
2. BioMOBY. http://www.mygrid.org.uk/.
3. EPCLUST. http://ep.ebi.ac.uk/EP/EPCLUST/.
4. The MGED ontology. http://mged.sourceforge.net/ontologies/MGEDontology.php.
5. myGrid. http://www.mygrid.org.uk/.
6. The predictive model markup language (PMML). http://www.dmg.org/pmml-v3-0.html.
7. W3C OWL-S: Semantic markup for web services. http://www.w3.org/Submission/OWL-S/.
8. W3C OWL web ontology language overview. http://www.w3.org/TR/owl-features/.
9. W3C RDF primer. http://www.w3.org/TR/rdf-primer/.
10. W3C SOAP version 1.2 part 0: Primer. http://www.w3.org/TR/2003/REC-soap12-part0-20030624/.
11. W3C web services description language (WSDL) version 2.0 part 0: Primer. http://www.w3.org/TR/2004/WD-wsdl20-primer-20041221/.
12. W3C XSL transformation (XSLT) version 1.0. http://www.w3.org/TR/xslt.
13. C. C. Aggarwal, C. Procopiuc, J. L. Wolf, P. S. Yu, and J. S. Park. Fast algorithms for projected clustering. In *ACM SIGMOD International Conference on Management of Data*, 1999.
14. C. C. Aggarwal and P. S. Yu. Finding generalized projected clusters in high dimensional spaces. In *ACM SIGMOD International Conference on Management of Data*, 2000.
15. Y. Cheng and G. M. Church. Biclustering of expression data. In *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology*, 2000.
16. S. Datta and S. Datta. Comparisons and validation of statistical clustering techniques for microarray gene expression data. *BioInformatics*, 19(4):459–466, 2003.
17. R. de Knikker, Y. Guo, J. long Li, A. K. H. Kwan, K. Y. Yip, D. W. Cheung, and K.-H. Cheung. A web services choreography scenario for interoperating bioinformatics applications. *BMC Bioinformatics*, 5(25), 2004.
18. M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proc. Natl. Acad. Sci. USA*, 95:14863–14868, 1998.
19. J. A. Hartigan and M. A. Wong. A K-means clustering algorithm. *Applied Statistics*, 28, 1979.
20. Z. Huang. Clustering large data sets with mixed numeric and categorical values. In *The First Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 1997.
21. L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley Inter-Science, 1990.
22. K. Y. Yip, D. W. Cheung, M. K. Ng, and K.-H. Cheung. Identifying projected clusters from gene expression profiles. *Journal of Biomedical Informatics (JBI)*, 37(5):345–357, 2004.