STUDIA INFORMATICA 2013
Volume 34 Number 2A (111)

Michał KOZIELSKI
Silesian University of Technology, Institute of Electronics
Łukasz STYPKA
Future Processing Sp. z o.o.

GENE ONTOLOGY BASED GENE ANALYSIS IN GRAPH DATABASE ENVIRONMENT

Summary. The article presents evaluation of the application of Neo4j graph database to Gene Ontology graph analysis. Graph-based term similarity measures are calculated in order to assess the effectiveness of the system. Two types of common ancestor search are presented and evaluated, and parallel execution of the analysis is also evaluated.

Keywords: graph database, gene analysis, gene ontology term similarity, Gene Ontology

ANALIZA GENÓW OPISANYCH PRZEZ ONTOLOGIĘ GENE ONTOLOGY W ŚRODOWISKU GRAFOWEJ BAZY DANYCH

Streszczenie. Artykuł przedstawia ocenę zastosowania grafowej bazy danych Neo4j do analizy grafu ontologii Gene Ontology. Ocena systemu została przeprowadzona na podstawie obliczenia bazujących na analizie grafu miar podobieństwa terminów ontologii. Przedstawione i ocenione zostały dwa sposoby wyszukiwania rodziców w grafie. Analizie poddano również równoległą realizację badanych algorytmów.

Słowa kluczowe: grafowa baza danych, analiza genów, podobieństwo terminów ontologii, Gene Ontology

1. Introduction

The Gene Ontology project [5] provides a database containing an ontology of defined terms representing gene product properties. It has been developed for the last thirteen years

and it contains a certain biological knowledge continuously extended and enriched. This knowledge can be mined [1, 2, 6, 8, 9, 13, 15] and combined with other information derived from biological experiments [6, 8, 9, 15]. Formally, the ontology is a directed acyclic graph, where nodes are the ontology terms and edges are the relations of different types existing between the ontology terms. Analysis of such data can be performed in many environments, e.g., Matlab [11] has a Bioinformatics Toolbox that is, among others, dedicated to gene ontology processing. However, an interesting option is to apply the environment dedicated for graph processing to the analysis of this type.

Graph database is a system that stores data in a graph structure. Graph databases belong to the class of database management systems called *Not only SQL* (NoSQL) systems. NoSQL database systems cover, among others, such solutions as XML databases or object databases. Their distinguishable feature is a data format different than relational and, as a consequence, a query language different than SQL.

One of the popular graph database systems is Neo4j [12]. It offers, among others, graph oriented query language and Java based API. Neo4j was already presented as a valuable graph analysis tool applied to social networks [16], in this article we present issues connected with applying this database management system to gene ontology analysis.

The analysis that is considered in this article focuses on the calculation of Gene Ontology term similarity [8, 9]. Term similarity can be calculated in order to compare genes annotated to these terms and to, e.g., cluster such genes [6]. Among different measures of Gene Ontology term similarity we can identify two classes of measures that process graph structure in order to compare terms: semantic similarity measures and path based similarity measures

The goal of the article is to:

- evaluate the Neo4j environment as gene ontology analysis tool,
- evaluate Neo4j efficiency in the considered applications,
- present the approach to Gene Ontology term similarity calculation that is more efficient than direct use of native Neo4j solution,
- compare the execution of the analyzed algorithms in Neo4j and Matlab environment.

The structure of this work is as follows. Section 2 presents the similarity measures that are compared in the analysis. The database environment is presented in section 3. Section 4 presents the experiments, their results and a discussion of the obtained outcome. Conclusions of the work are presented in section 5.

2. Graph-based term similarity measures

Gene Ontology is a valuable source of expert knowledge. It can be treated as an additional source of information that can be combined with the results of the analysis of biological experiments. The example of such analysis can be gene clustering based on gene expression evaluation that can be compared with gene similarity calculated on the basis of GO representation. Gene similarity in GO representation can be calculated, e.g., on the basis of similarity of the terms annotating the analyzed genes.

The following paragraph presents Gene Ontology term similarity measures that require graph processing. Two classes of such measures that can be taken into consideration are: semantic similarity measures and shortest path based similarity measures.

Semantic term similarity measures utilise the concept of Information Content $\tau(a_i)$ of an ontology term $a_i \in A$ (where A is a set of all GO terms) given by the following formula:

$$\tau\left(a_i\right) = -\ln\left(P\left(a_i\right)\right), \tag{1}$$

where *P(a)* is a ratio of a number of gene annotations to a term *a*, to a number of analyzed genes.

The simplest semantic similarity measure was proposed by Resnik [14]. It takes into consideration only the Information Content of the most informative common ancestor $\tau_{ca}(a_i,a_j)$ of the compared terms a_i and a_j:

$$s_A^{(R)}\left(a_i, a_j\right) = \tau_{ca}\left(a_i, a_j\right). \tag{2}$$

More complex approaches were proposed by Jiang-Conrath [7] and Lin [10] and they take into consideration also information content of the terms themself. There is also an approach extending the number of common ancestors included into calculations [2].

The second approach that is considered here is to define the distance between two terms a_i and a_j as a length $l(a_i,a_j)$ of the shortest path between them. One of the widely implemented methods for shortest path calculation was introduced by Dijkstra [3] and this name is adopted as the name of the method.

Calculating shortest paths in Gene Ontology it has to be taken into consideration that the ontology graph is a directed one. Therefore, the length of a path between two ontology terms that are not connected by a parent-child relation can be set as infinity or it can be calculated as a sum of path lengths leading to the common ancestor. The latter approach was chosen in the work presented.

When the shortest paths are calculated then the similarity of the two GO terms can be defined as exponential dependency on a path length $l(a_i,a_j)$ [1]:

$$s_A^{(p)}\left(a_i, a_j\right) = e^{-l\left(a_i, a_j\right)}. \tag{3}$$

Each of the approaches presented above requires evaluation of the common ancestor node in a graph and the path based method requires calculation of the shortest path additionally. Therefore, performing the analysis within the environment dedicated to graph processing is justified and it should be profitable.

3. Graph database environment

Graph database, which representative is Neo4j [12], is a special kind of object database. Its two main components are nodes and relationships. Both the nodes and the relationships do not have a scheme imposed from above, what makes the graph database extremely flexible. Neo4j was written in Java, and it is developed on an open-source license. Neo4j satisfies the assumptions of the ACID (atomicity, consistency, isolation, durability) transaction properties. By forcing each operation that modifies the data to occur in the transaction, it ensures the consistency of data.

Graph database Neo4j is available in two editions: embedded and standalone. In the created application the embedded edition was used. Ontology database contains a relatively small number of nodes (39720) and relationships between them (70282), thus the choice of this edition allows the highest performance and reduce the time needed to communicate between application and database server.

Fig. 1 presents the main node and the relationships between the nodes used in the application calculating the gene ontology based similarity between genes.

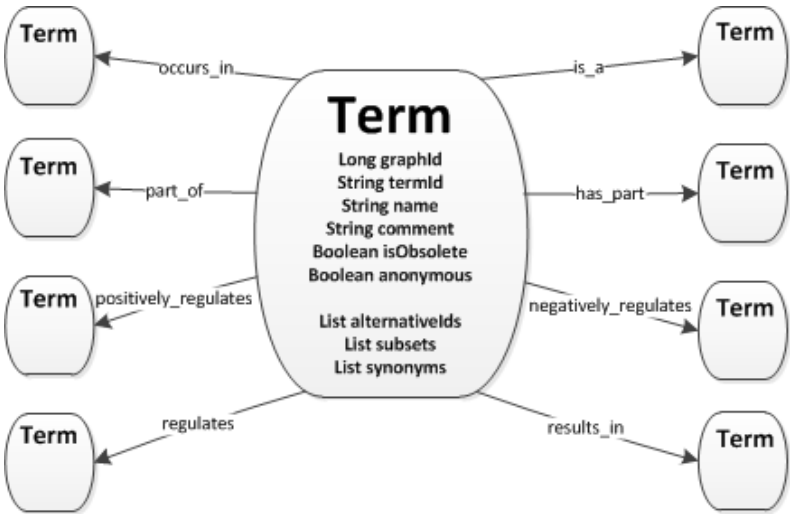

Fig. 1. The main node and the relationships between the nodes used in the presented application
Rys. 1. Główny węzeł oraz relacje pomiędzy węzłami wykorzystywane w prezentowanej aplikacji

Neo4j database has several algorithms implemented that facilitate the work with a graph (e.g. Dijkstra algorithm, shortest path algorithm, A* algorithm). A special query language,

Cypher, has been also introduced to the system. Its syntax reminiscent SQL, but it is enriched by special phrases useful when working with a graph. One of the most important Cypher expressions is the MATCH phrase, that is used to find the given pattern in the graph. The example of the MATCH phrase is presented below.

Example 1

```
START a=node({0}), b=node({1}) MATCH a-[:IS_A*]->ancestor<-[:IS_A*]-b RETURN
distinct ancestor
```

In this example, the common ancestors of two nodes *a* and *b* are searched. The search will take into consideration only the relationships of IS_A type and a number of relations is not limited.

In most of the algorithms calculating the similarity between terms in gene ontology, one of the main problems is searching the common ancestor for the given pair of terms. In this work, two distinct approaches to this problem have been used. The first one uses the internal mechanisms of the Neo4j database utilizing the pattern matching in a graph. This example is shown above and it will be further referred by a * sign.

The second approach (further referred by a ** sign), that was introduced in this work, is a separate search for all ancestors of the first term and the second term, which is presented below:

Example 2

```
START a=node({0}) MATCH a-[:IS_A*]->ancestor RETURN distinct ancestor
```

Next, the number of two received sets is checked and the elements of the larger set are inserted into hash table. Having all the elements of the set inserted, it is checked, for each element of a smaller set, whether the element is already placed in the hash table. If so, the term is the common ancestor of the two terms. If the element is not placed in the hash table, it is rejected. Assuming that in the hash table there are no conflicts, the complexity of checking which terms are common ancestors (Example 2) is of O(n+m), while the complexity of comparing two sets (Example 1) amounts to O(n*m), where n and m are the numbers of terms in the analyzed sets.

4. Experiments and results

The aim of the experiments was to compare the execution time of the methods implemented in different ways and ran in different environments. Two environments were tested: Matlab with Bioinformatics Toolbox providing methods dedicated to Gene Ontology analysis and Neo4j graph database. The created Java application executed in Neo4j environment con-

tained different approaches to common ancestor identification and in addition it was paralleled in order to allow the determination of the similarity between multiple genes simultaneously.

The experiments were conducted on a Yeast dataset [4] that consists of 274 genes annotated to 248 GO terms. This dataset contains genes expression profiles from budding yeast *S. cerevisiae* that were measured during several different DNA microarray experiments.

Each experiment consisted of evaluation of term similarity according to the methods presented in section 2 and calculation of gene similarity based on term similarity.

The execution time results obtained by the Java application based on Neo4j graph database are presented in Table 1. Each method has two implementations depending on the approaches presented in Example 1 and 2 referred by * and ** respectively. Table 1 contains the following sections:

- Total time – the time needed for calculation of gene similarity based on term similarity calculated by a given method, where genes from Yeast dataset were analyzed and terms annotating these genes were taken into consideration,
- Comparison of two terms (average) – the average time needed to compare two terms in the ontology,
- Standard deviation – the standard deviation of the time needed to compare two terms in the ontology.

Table 1

Time duration of the analysis

	No. of threads	Resnik**	Dijkstra**	Resnik*	Dijkstra*
Total time [min]	1	110.11	133.73	197.80	836.33
	2	75.33	84.21	105.69	444.65
	3	62.55	62.48	80.15	316.84
	4	59.14	52.84	74.89	266.08
Comparison of two terms (average) [ms]	1	11	14	22	95
	2	16	18	23	101
	3	20	20	26	108
	4	25	22	33	121
Standard deviation [ms]	1	12.05	25.50	75.24	556.81
	2	13.88	28.11	81.07	594.07
	3	16.09	29.79	83.95	638.11
	4	19	32.86	93.30	744.73

The chart presented in Fig. 2. shows, that the average time of comparing two terms in the case of Resnik and Dijkstra algorithm is similar in the case of a separate search for the ancestors of two terms (**). There is, however, a significant difference in execution time in case of using mechanism of path pattern matching in a graph (*). In that case execution of Dijkstra algorithm is about 5 times slower than Resnik algorithm.

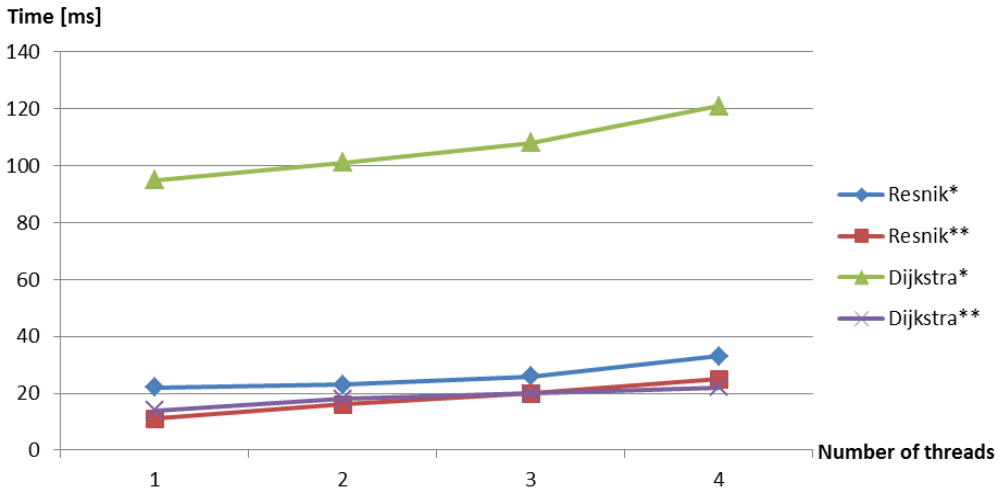

Fig. 2. Average time duration of two terms similarity calculation
Rys. 2. Średni czas wyznaczenia podobieństwa dwóch terminów

It can be also noticed that the average time of two terms similarity calculation increases with the increased number of threads involved in the calculations. It can be explained by the reduced access to the common cache memory with the increased number of threads involved in processing. Although the average execution time of individual calculation increases the parallel processing enables us to reduce the total processing time significantly.

Table 2
Ratio of execution time for a given number of threads in relation to execution time evaluated for a single thread

No. of threads	Resnik**	Dijkstra**	Resnik*	Dijkstra*
1	1.00	1.00	1.00	1.00
2	0.68	0.63	0.53	0.53
3	0.57	0.47	0.41	0.38
4	0.54	0.40	0.38	0.32

Table 3
Percentage of improvement for a given number of threads (k) comparing to k-1 number of threads

No. of threads	Resnik**	Dijkstra**	Resnik*	Dijkstra*
k=2	31.59	37.03	46.57	46.83
k=3	16.97	25.80	24.17	28.74
k=4	5.45	15.43	6.56	16.02

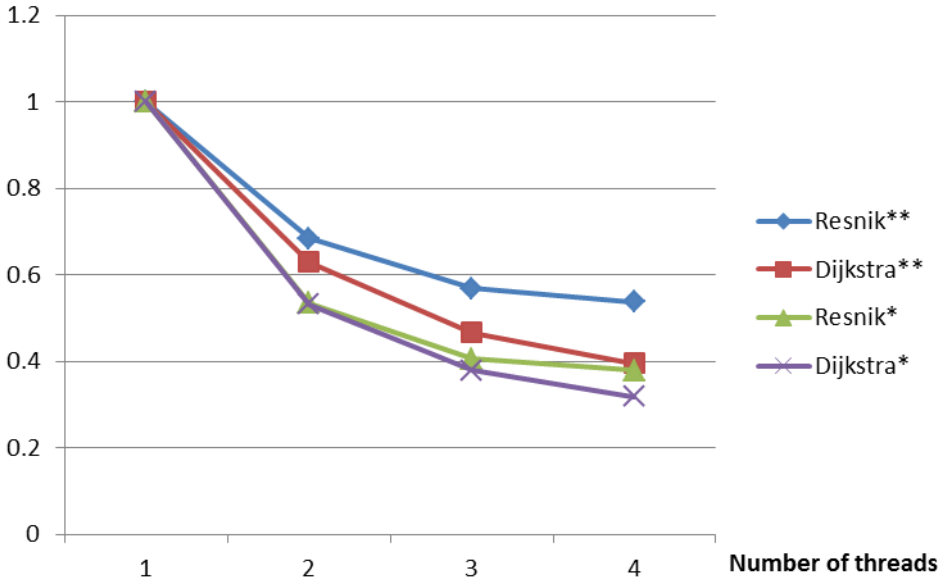

Fig. 3. Ratio of execution time for a given number of threads in relation to execution time evaluated for a single thread

Rys. 3. Stosunek czasu wykonania danej liczby wątków do czasu wykonania pojedynczego wątku

Another observation is that the improvement of the total execution time decreases very quickly with the increasing number of threads, what is presented in Table 2 and 3, and is illustrated in Fig. 3. Table 2 contains a ratio of execution time for a given number of threads in relation to execution time evaluated for a single thread. What can be seen is that increasing a number of threads to 2 gives a significant improvement in terms of execution time (up to over 46% - see Table 3) and increasing a number of threads from 3 to 4 gives much smaller improvement (up to over 16% - see Table 3).

The experiments showed also that the location of the analyzed term in a graph has a large impact on the time of the calculation executed in the Neo4j database. The nodes which have a greater number of ancestors and are located further away from the root of a graph are analyzed considerably longer, than the nodes located at the root, as it is presented in Table 4.

Table 4

Examples of time execution dependence on the number of term ancestors

First node	No. of ancestors	Second node	No. of ancestors	Common ancestor	No. of ancestors	Time [ms]
GO:0015031	7	GO:0043039	23	GO:0008150	0	< 1
GO:0006412	14	GO:0006811	4	GO:0008150	0	< 1
GO:0008152	1	GO:0043039	23	GO:0008152	1	< 1
GO:0015986	48	GO:0015986	48	GO:0015986	48	3140
GO:0006184	36	GO:0006754	36	GO:0009205	20	3479
GO:0006184	36	GO:0015986	48	GO:0009205	20	3226

The analysis of standard deviation of the average time of the two terms similarity calculation (Table 1) enables evaluation how much the analyzed methods are vulnerable to the phe-

nomenon presented above. As presented in Table 4, this time depends on the location of terms in a graph. The results enclosed in Table 1 clearly indicate that separate searching for common ancestors of the first and second term, and then creating the intersection of the sets (methods marked as **) is more stable and less dependent on the location of terms in a graph, than the use of the pattern matching within graph, which method is available in Neo4j.

The analysis of the Matlab application outcomes that are presented in Table 5 and compared in Fig. 4, shows that this environment is significantly less effective than the graph database when Gene Ontology graph is analyzed. This result was expected and it is a consequence of the processing model implemented in both environments.

Table 5

Time duration of the analysis in Matlab environment

	Resnik	Dijkstra
Total time [min]	276.88	871.48
Comparison of two terms (average) [ms]	44.53	140.46
Standard deviation [ms]	10.21	97.68

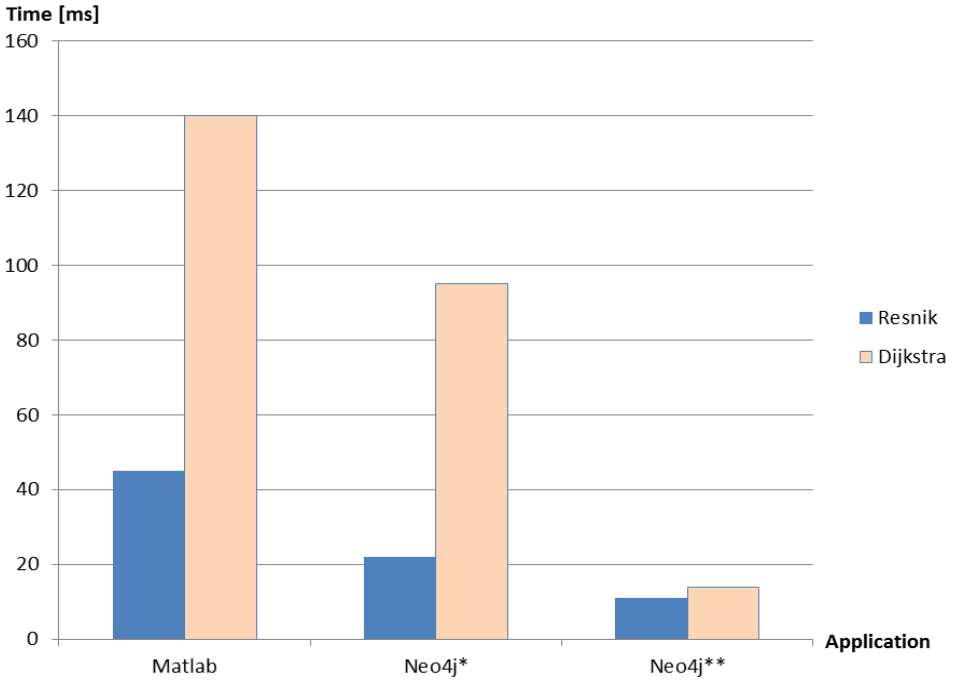

Fig. 4. Average time duration of the comparison of two terms – comparison between execution environments

Rys. 4. Średni czas porównania dwóch terminów – porównanie pomiędzy środowiskami

5. Conclusions

The article presented application of Neo4j graph database environment to analysis of Gene Ontology graph. Two different approaches to a term common ancestor search were

evaluated, where one (*) is based on a database query language only and the second (**) is based on a database query and additional processing performed within the application. The possibility of parallel search execution was also verified and additionally, the comparison of the execution time of the analysis in Matlab environment and in Neo4j environment was performed.

The results of the analysis showed that the approach (**), that was introduced in the article, is more effective and less sensitive to the term location in the ontology graph. Introduction of parallelism to the processing in Neo4j environment enabled improvement of the total execution time but showed not to be very scalable with the number of threads executed. The comparison of the execution time of the analysis in Matlab environment and in Neo4j environment confirmed the superiority of the graph database environment in the considered application.

BIBLIOGRAPHY

1. Al Mubaid H., Nagar A.: Comparison of four similarity measures based on GO annotations for Gene Clustering. IEEE Symposium on Computers and Communications, ISCC 2008, p. 531÷536.
2. Couto F. M., Silva M. J., Coutinho P. M.: Measuring semantic similarity between Gene Ontology terms. Data Knowledge Engineering, Vol. 61, 2007, p. 137÷152.
3. Dijkstra E. W.: A note on two problems in connexion with graphs. Numerische Mathematik, Vol. 1, 1959, p. 269÷271.
4. Eisen M. B., Spellman P. T., Brown P. O., Botstein D.: Cluster analysis and display of genome-wide expression patterns. Proc. Natl. Acad. Sci. USA, Vol. 95, 1998, p. 14863÷14868.
5. GO-Consortium: The Gene Ontology (GO) database and informatics resource. Nucleic Acids Research 2004, 32 (http://www.geneontology.org).
6. Gruca A., Kozielski M., Sikora M.: Fuzzy Clustering and Gene Ontology Based Decision Rules for Identification and Description of Gene Groups. AISC, Vol. 59, 2009, p. 141÷149.
7. Jiang J. J., Conrath D. W.: Semantic similarity based on corpus statistics and lexical ontology. Proc. on Int. Conference on Research in Computational Linguistics, 1997, p. 19÷33.
8. Kozielski M., Gruca A.: Evaluation of Semantic Term and Gene Similarity Measures. LNCS, Vol. 6744, Springer, 2011, p. 406÷412.

9. Kozielski M., Gruca A.: Visual comparison of clustering Gene Ontology data when different similarity measures are applied. Studia Informatica, Vol. 32. No. 2A (96), Gliwice 2011, p. 169÷180.
10. Lin D.: An information-theoretic definition of similarity. Proc. of the 15th Int'l Conference on Machine Learning, 1998, p. 296÷304.
11. http://www.mathworks.com
12. http://www.neo4j.org
13. Pesquita C., Faria D., Falca A. O., Lord P., Couto F. M.: Semantic Similarity in Biomedical Ontologies PLoS Comput. Biol. 5(7), 2009, p. 1÷12.
14. Resnik P.: Semantic Similarity in a Taxonomy: An Information-Based Measure and its Application to Problems of Ambiguity in Natural Language. J. Artif. Intell. Res. (JAIR), Vol. 11, 1999, p. 95÷130.
15. Wang H., Azuaje F., Bodenreider O., Dopazo J.: Gene expression correlation and gene ontology-based similarity: an assessment of quantitative relationships. In Proc. of the 2004 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology CIBCB '04, 2004, p. 25÷31.
16. Warchał Ł.: Using Neo4j graph database in social network analysis. Studia Informatica, Vol. 33, No. 2A, Gliwice 2012, p. 271÷279.

Wpłynęło do Redakcji 16 stycznia 2013 r.

Omówienie

Artykuł przedstawia ocenę zastosowania grafowej bazy danych Neo4j do analizy grafu ontologii Gene Ontology. Ocena systemu została przeprowadzona na podstawie wyznaczania bazujących na analizie grafu miar podobieństwa terminów ontologii. Analizowane w artykule miary podobieństwa należą do klasy miar semantycznych oraz wyznaczających najkrótsze ścieżki w grafie. W przypadku analizowanej ontologii obydwie miary wymagają wyznaczenia wspólnego rodzica porównywanych terminów. Przedstawione i ocenione zostały dwa sposoby wyszukiwania rodziców w grafie. Metody te charakteryzują się różną efektywnością i podatnością na wpływ położenia analizowanych terminów w grafie.

Analizie poddano również równoległą realizację badanych algorytmów. Wykorzystanie do 4 wątków pozwoliło na znaczące przyspieszenie realizacji obliczeń, jednakże zwiększanie liczby wątków szybko napotyka ograniczenia w poprawie efektywności realizacji aplikacji.

Przeprowadzone porównanie z czasem realizacji obliczeń w środowisku Matlab, zawierającym przeznaczony do tego typu analiz pakiet Bioinformatics Toolbox, pokazało znaczącą przewagę środowiska grafowej bazy danych.

Addresses

Michał KOZIELSKI: Politechnika Śląska, Instytut Elektroniki, ul. Akademicka 16, 44-100 Gliwice, Polska, michal.kozielski@polsl.pl
Łukasz STYPKA: Future Processing Sp. z o.o., ul. Bojkowska 37, 44-100 Gliwice, Polska, lstypka@future-processing.com