FACTA UNIVERSITATIS
Series: **Automatic Control and Robotics** Vol. 16, N^{o} 1, 2017, pp. 73 - 81
DOI: 10.22190/FUACR1701073M

ONE APPROACH FOR ELIMINATING COMMUTATIVE PAIRS OF ELEMENTS IN WEB APPLICATIONS

UDC 004.738.2/.12

Petar Milić, Nataša Veljković

University of Niš, Faculty of Electronic Engineering, Niš, Republic of Serbia

Abstract. *Nowadays web applications contain graphical elements which are often connected in a way that reflects relationship between them. Connections have different structure depending on needs of application. A common type of structure used to describe relationship between elements consists of four members, two of which are elements and two lines represent relationship between them. This paper presents an approach to eliminate commutative pairs in array of elements with mutually exclusive connections. The ECOMPAIR (Elimination of COMmutative PAIRs) algorithm allows using one line with associated descriptions to link two elements. Experimental results obtained from applying the ECOMPAIR algorithm on jsPlumb library for visualization show significant reduction on number of elements necessary for visualization.*

Key words: *commutative pairs, web visualisation, web application*

1. INTRODUCTION

Web application development has great popularity due to emergence of modern software development methodologies and web application frameworks. Areas of their usage span from science, health, e-commerce, research, architecture, and advertising to small home projects. In many situations client-server architecture prevails. Developers usually follow the convention that most of work that is required by the application is performed on the server side, relieving the client from unnecessary work for whom is unknown level of performance. Client side is entrusted with data visualization in a user-friendly manner.

Presenting the results processed on the server side of the application include display of data gathered from database, data gathered from APIs, data for visualization on graphs and many other [1][2]. Visualization of data in web applications requires consideration of

Received September 22, 2016
Corresponding author: Petar Milić
University of Niš, Faculty of Electronic Engineering, Aleksandra Medvedeva 14, 18000 Niš, Republic of Serbia
E-mail: milicpetar86@gmail.com

the best available options for optimization of necessary workload on clients which is needed to process data gathered from server. Data that are to be displayed on graphs or data used for any other graphical elements are mostly encoded in JSON format [3], because of its simplicity for transfer over network and ease of use. There is vast amount of client-side technologies and libraries that can process JSON and enable visual representation. We have explored the possibilities of a library named jsPlumb [4]. It relies on HTML, CSS and JavaScript and SVG to enable visual interaction of elements in web applications. jsPlumb offers advanced user interface features such as pan/zoom, data binding, various graph operations with graphical elements and more. It is integrated with contemporary JavaScript frameworks such as Angular[1], Angular 2[2], React[3] and Vue 2[4] enabling thus easy way for building web applications with visual connectivity at their core.

2. RELATED WORK

Data visualization is often preferred over tables or text elements since it allows visitors to process data faster and gain deeper understanding of data and relationship between them [5]. Interaction visualization techniques such as 2D/3D displays, icon-based displays, dense pixel displays, geometrically transformed displays, allow users to interact directly with a visualization enabling thus relation and combination of different elements in order to provide more information about elements [6]. They deal with visualization of different data types and integration of visualization tools without going deeper in essence of presenting those elements to the end user from the aspect of time consumption. Other researchers [2] utilize hidden links, mapping and unmapping to enable single-screen visualization of hyperbolic space with multiple path links, which is a display of a tree structure with nodes in which child nodes has a single primary parent connected via primary and secondary path. In this tree, structure nodes are organized along parent and child relationships, where each parent can have many children and each child can have only one parent. This approach lacks consideration of time consumption and other visualization approaches besides the trees. A movement towards a reduction of processing power needed for visualization of data was done by Kreuseler, Lopez and Schumann [7]. They proposed a framework that integrates a scalable preprocessing pipeline for organizing large unstructured high-dimensional information spaces for visualizing information structure along with information contents. Their work offers methods and techniques for visualizing different types of information such as hierarchical information, structured or unstructured multi-dimensional information spaces, without taking into account possible relation between visualized data.

Tominski, Abello and Schumann [8] in their work of interactive graph visualization system, describe the solution for interaction between nodes in graph, supporting in that manner visual exploration of such structure. Common operations on visualized graph such as: brushing, zooming and panning require proper dealing with structure of the graph maintaining at the same time relationships between nodes. As this is a general solution, it doesn't address the type of relationships between elements on the graph as well as the consideration of performance aspect and how it affects drawing of the elements and lines.

[1] https://angular.io/
[2] http://www.angular2.com/
[3] https://facebook.github.io/react/
[4] https://vuejs.org/2016/04/27/announcing-2.0/

The work of Noack [9] on clustering of graphs with nonuniform degrees deals with relating graph elements with double edges forming thus reciprocated relationships which consequently affect visibility and visual exploration of such graph, but at the same time it doesn't offer discussion about time needed to draw the graph and unnecessary lines between elements. Further, a hybrid visualization system of social networks [10] proposed by Henry, Fekete, and McGuffin gives answers on questions how graphical elements are linked and what kind of links are used, while user visually explores graph. This system focuses on visualization of graph elements without going deeper in representation of relationships between them, and how that affect time consumption necessary for drawing.

3. Problem Definition

We wanted to find the answer for the following question: can we visualize connection of two elements, A and B in way A → B and B → A via same line with associated type of connection? Mentioned situation is similar to the notion commutativity in mathematics, where changing the order of the operands does not change the result. Similarly, different data [11] and elements in web applications can be connected with "parent of" / "child of", "links from" / "links to", "derives from" / "has derivation" and other types of connection. Thus, knowing only one type of connection between the two elements, one can easily identify the other one. In that way, a visibility and readability is significantly improved. There is no need to display both connections as preview of one connection brings information on what is the opposite type of connection. Without existence in this way defined type of connections, we would have to represent connections in both directions in order to fully describe elements, which are time consuming process (see Fig.1).

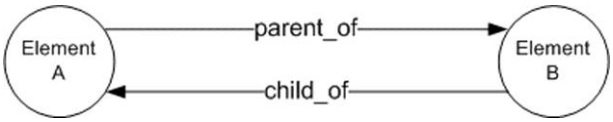

Fig. 1 Doubled connections between the elements

Described problem can be analyzed from the point of theory of graphs, by looking at the directed multigraph or multigraph with multiple arrows which connect two distinct elements in opposite direction creating loop between elements. This representation is intuitive, but, it is not guaranteed that a viewer is able to determine edge direction as quickly and unambiguously as possible; alternative representations that exhibit less visual clutter might be better suited. With excluding direction from graph, they become undirected graphs in which edges have no orientation. Noack [9] in his research on graph drawing and visualization, points out that node with double edges correspond to the reciprocated relationships which waste much area unnecessarily and especially are reflected on the clustering of large graphs and their visibility.

Visualization of graph elements (nodes) connected with multiple links (arrows) in dense graphs that contain high number of elements with adjacent elements, imposes the question of readability [10], thereby aggravating attentive visual analysis by the humans and processing by the computers. For example, visual analysis of different combinations in social networks with dense communities may be a potential problem, as it is known that social networks are

globally sparse and locally dense. Similar problem can be found in WordNet[5], where visualization of relationships between any two words who has "IS-A" type of relationship or visualization of relationships between synonyms and many others imposes drawing of two arrows in order to present all relationships. Keeping in mind that graphical representation of two words in WordNet always contains one pair of arrows, we can then relate them via only one line with relationship description at both sides creating thus clear visualization between them.

In the following sections we will give our contribution to the described problem in general manner, focusing on time necessary to represent relationships between two graphical elements with only one line and showing that speedup in drawing such visualization approach can be achieved. In Section 4. we propose solution of the problem in terms of defining a new algorithm called ECOMPAIR. Use case scenario carried out on the graphical elements of jsPlumb library [4] along with validation of ECOMPAIR algorithm is given in Section 5. In the final section we conclude work done and present ideas for the future work.

4. SOLUTION

In order to improve visibility of elements and reduce drawing on the screen, we came to idea to draw only one line between elements and give description of connection on both elements, thus minimizing necessary drawing. Mutually exclusive connections can be described by using notion commutativity. For example, in operations of addition and subtraction a sign beside operands exist, and whether we change the place of operands this does not affect the result. Consequently, it is enough to write only one expression, not both.

Fig. 2 One connection between the elements

As we can see from the Fig 2. there is only one connection, which is by our opinion enough to describe the relationship. Recall that if

$$A - B = C\,, \tag{1}$$

that is equivalent to:

$$-B + A = C\,, \tag{2}$$

In our case, if we have pairs of elements denoted as [[A, B], [C, D], [E, F]], notation [B, A], [D, C] and [F, E] will give the same result as we use previously defined types of connections which clearly describes relationships between them. Also, if we have set of pairs of elements [[A, B], [A, C], [B, A], [C, D], [D, A], [C,A]], enough for display will be set [[A, B], [A, C], [C, E], [E, A]]. Note that [B, A] and [C, A] are unnecessary.

The pseudo code of the algorithm for removing unnecessary pairs of elements is given in the Fig. 3.

[5] https://wordnet.princeton.edu/

```
Algorithm for finding commutative pairs:

Input: "A" array of n members
Output: filtered array without commutative pairs
Begin:
find all elements which are paired by parsing each element of array "A"
put element indexes which are paired in array "P"
for each element from array P find its pairs and put pairs in array "C"
copy "C" to "C1"
let "1" and "2" be index of elements in pair
for each element from "C" do
  for each element from "C1" do
  if C(current index of C)(1) == C1(current index of C1)(2) and C(current index of C)(2) == C1(current index of C1)(1)
    then
   C1(current index of C1)(1) ==  C(current index of C)(1)
   C1(current index of C1)(2) ==  C(current index of C)(2)
from "C1" remove duplicate pairs
compare "C1" pairs with pairs from "P" to get type and associate it with each pair from "C1"
copy "C1" to "C"
return C
```

Fig. 3 ECOMPAIR - Algorithm for finding commutative pairs

If we apply this algorithm on following array: [A → ([A, B, type], [A, C, type]), B → ([B, A, type]), C → ([C, A, type], [C, D, type]), D → ([D, C, type]), E, F], the result will be [[A, B, type], [C, A, type], [C, D, type], E, F]. Graphically representation is given on Fig. 4 and Fig. 5, before and after applying proposed algorithm.

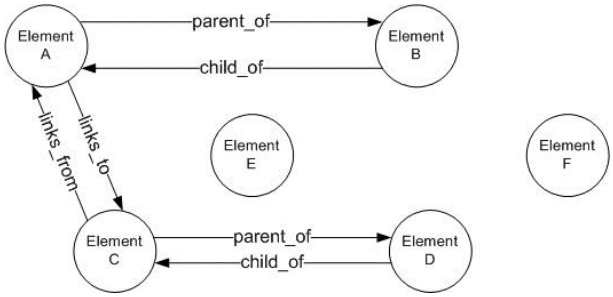

Fig. 4 Graph before applying ECOMPAIR

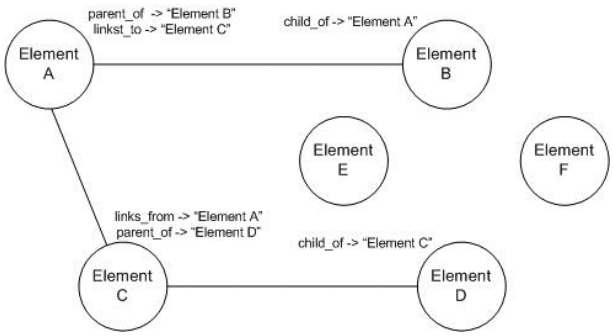

Fig. 5 Graph after applying ECOMPAIR

5. A USE CASE STUDY: THE USE IN WEB APPLICATIONS

Application of ECOMPAIR for visualization of graphical elements [6] in web applications brings savings in the time required for drawing elements and consequently speeds up the execution of application on the client computer, because it takes less processing power and memory for the job. The jsPlumb library offers variety of visualization methods, but each of them uses same coding of displayed elements and their relationships in the background. Any other library which uses format for representation of graphical elements and their relationships as nodes with mutually exclusive connections can also be used. In the Fig. 6. we depicted the process of generating elements in a web page by using jsPlumb library.

```
var example3Color = "rgba(229,219,61,0.5)";
var exampleEndpoint3 = {
endpoint:["Dot", {radius:17} ],
anchor:"BottomLeft",
paintStyle:{ fillStyle:example3Color, opacity:0.5 },
isSource:true,
connectorStyle:{
      strokeStyle:example3Color,
      lineWidth:4
},
connector : "Straight",
isTarget:true,
maxConnections: 20,
beforeDetach:function(conn) {
return confirm("Detach connection?");
},
onMaxConnections:function(info) {
alert("Cannot drop connection " + info.connection.id + " : maxConnections has been reached on Endpoint " + info.endpoint.id);
}
};
e1 = instance.addEndpoint("Element A", { anchor:[1, 0, 0, 0] }, exampleEndpoint3,{uuid:"1"});
```

Fig. 6 Element description using jsPlumb

Class instance has addEndpoint method used for adding endpoint (*Element A*), with anchor (at top right corner), predefined style (*exampleEndpoint3*) and unique identifier (*uuid*). Connections between elements are created using connect method and specifying source, target and scope of connection. For example using *instance.connect*(*source : e1; target : e2; scope : "parent of"*) two elements e1 and e2 are connected with e1 being parent of e2.

We used jsPlumb library to create the CKAN plug-in for visualization of open datasets as simple graphical elements interconnected via simple lines. Open datasets or i.e. open data represents freely redistributed data across the web that can be reused and redistributed by anyone. CKAN is a platform for publishing of such open datasets, which offers possibility of their interconnection by defining relationships [12]. Application of the jsPlumb library for visualization of graphical elements in web application with the usage of ECOMPAIR for optimization is depicted in Fig. 7. As it can be seen there is only one connection between elements and appropriate description of connection on both elements.

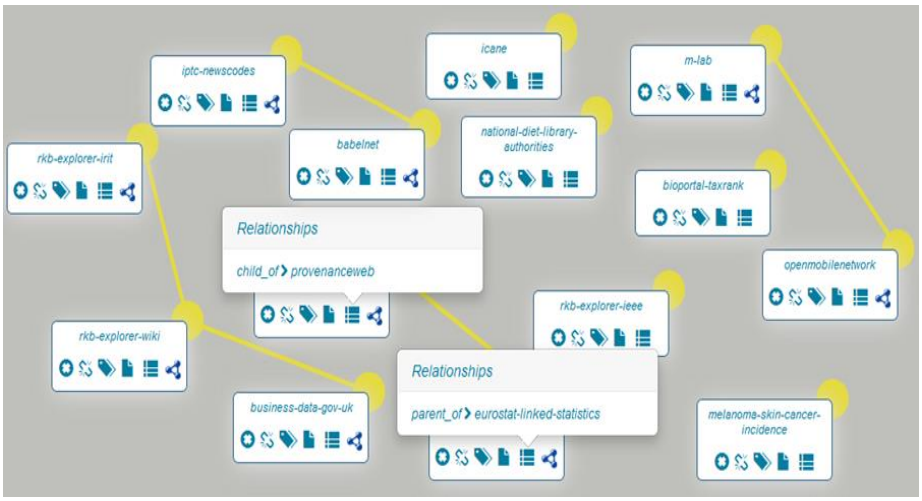

Fig. 7 Connecting elements in jsPlumb using ECOMPAIR algorithm for enhancement

5.1. Validation of algorithm

To confirm improvements and benefits of applying ECOMPAIR, we analyzed added enhancements in the aforementioned CKAN plug-in on the test application installed on local computer. This algorithm was created during the development of software solution suitable for the visual representation of the datasets published on the CKAN platform.

For validating that ECOMPAIR gives mentioned benefits, we statistically analyzed different number of elements using model-based approach as described in [13].

Table 1 Comparative view of pairs before and after apply of algorithm

Number of pairs before applying of algorithm	Number of pairs after applying of algorithm	Coefficient
10	5	
20	10	
50	25	
100	50	
150	75	0.5
200	100	
250	125	
300	150	
350	175	

Results obtained are given in Table 1. It can be noted that coefficient of reduction is 50%, as total number of connections that are needed to visualize is reduced to half. This confirms our assumption that the algorithm reduces time necessary to present the visual elements to the user.

Table 2 Analysis of creation of connections with different number of elements

NE	NCA + NE	NCB + NE	(NCA + NE) / (NCB + NE)	Factor
8	36	64	0.5625	0.4375
10	55	100	0.55	0.45
15	120	225	0.5333	0.4666
20	210	400	0.525	0.475
25	325	625	0.52	0.48
30	465	900	0.5166	0.4833
35	630	1225	0.5142	0.4857

To test worst case scenario, we tried to create maximum number of connections between different elements that can appear in web application. If there is for example n number of elements, than maximum number of connections can be computed as:

$$max = n*(n-1) \tag{3}$$

Table 2 summarizes obtained results. We used the following abbreviations: NCA - Number of Connections After applying of algorithm, NCB - Number of Connection Before applying of algorithm and NE - Number of Elements. Any graphical element or connection in jsPlumb library needs some unit of time to be drawn on the screen. Looking at Table 2 and keeping in mind previous fact, than the values from columns NCA, NCB and NE can be added. Finding relationship between units of time of number of elements and number of connections before and after applying of the algorithm can show to which extent algorithm speeds up drawing of elements on the screen.

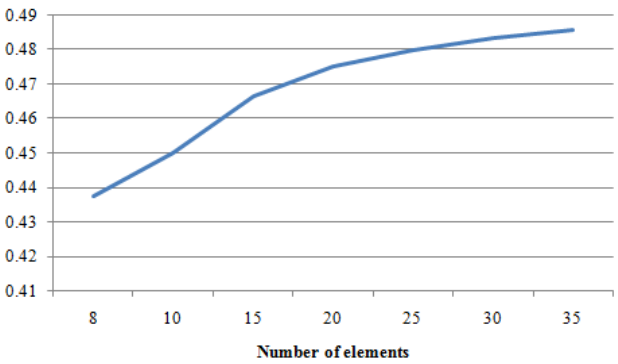

Fig. 8 Factor of improvement using ECOMPAIR

From graph on Fig. 8. it can be noticed that application of ECOMPAIR gives improvements in drawing of elements with large number of mutual connections. As number of elements increases, algorithm gives better results.

6. CONCLUSION

In this paper we described ECOMPAIR algorithm for finding and eliminating commutative pairs of elements in web applications that utilize visual representation of elements as nodes with lines as connections between them. ECOMPAIR algorithm is especially suited for cases where exists mutually exclusive connections such as "parent of"/"child of", "links from"/ "links to", "is a"/"contains" and others. Reduction in number of elements and processing time to display results is confirmed in practice. A valuable gain is also in minimizing the space necessary to present the final view, thus achieving better clarity of visualization. We showed that it is enough to display only one connection without direction, with appropriate description on both elements, which improves performance of web application in whole and lowers time consumption. As there are many libraries for visualization in web applications, we tried to make our algorithm generalized in order to enable its interoperability and applicability in different visualization systems.

Further research in this area should include empirical examination of different visualization systems that support relationships between elements in order to find which of them gives best results. Also, it would be interesting to find and examine different data structures that are used to represent graphical elements and relationships and to test their behavior. Moreover, application of ECOMPAIR in those systems and comparative analysis of their characteristics and performances would be interestingly as well as comparison with approaches mentioned in the Section 2.

REFERENCES

[1] S. Mavromoustakos, A.S. Andreou: "WAQE: a web application quality evaluation model". Internation. Journal of Web Engineering and Technology, 2, 2007, pp. 96-120.

[2] M. C. Hao, M. Hsu, U. Dayal and A. Krug: "Technique for visualizing large web-based hierarchical hyperbolic space with multi-paths". U.S. Patent No. 6,377,287., 2002.

[3] Z. U. Haq, F. K. Gul and H. Tazar: "A Comprehensive analysis of XML and JSON web technologies". New Developments in Circuits, Systems, Signal Processing, Communications and Computers, pp. 102-109.

[4] jsPlumb toolkit, https://jsplumbtoolkit.com

[5] N. Veljković, I. Antolović, and L. Stoimenov: "Visual Analysis in Reference Management Software" Applied Mechanics and Materials, 197, pp 633-637, 2012.

[6] D.A. Keim: "Visual exploration of large data sets". Communications of the ACM, 44, 8, pp. 38-44, 2001.

[7] M. Kreuseler, N. Lopez and H. Schumann: "A Scalable Framework for Information Visualization" In Proceedings of the InfoVis EEE symposium, pp. 27-36, IEEE, 2000.

[8] C. Tominski, J. Abello and H. Schumann: "An Interactive Graph Visualization System" Computers & Graphics, 33, 6, pp. 660-678, 2009.

[9] A. Noack: "Energy-Based Clustering of Graphs with Nonuniform Degrees", In P. Healy and N. S. Nikolov, editors, Proceedings of the 13th International Symposium on Graph Drawing (GD 2005), Springer-Verlag, pp. 309-320, 2005.

[10] N. Henry, J.D. Fekete, and M.J. McGuffin: "NodeTrix: A Hybrid Visualization of Social Networks" IEEE Transactions on Visualization and Computer Graphics, 13, 6, pp. 1302-1309, 2007.

[11] I. Bisevac, D. Rancic, and M. Pavlovic: "Graphical presentation of statistical data using Web technologies". 20th Telecommunications Forum (TELFOR). 2012.

[12] P. Milić, N. Veljković, and L. Stoimenov: "Linked Relations Architecture for Production and Consumption of Linksets in Open Government Data". In: Janssen M. et al. (eds) Open and Big Data Management and Innovation. Lecture Notes in Computer Science, Vol. 9373. Springer, 2015.

[13] M. Shams, D. Krishnamurthy and B. Far: "A Model-Based approach for Testing the Performance of Web Applications", In Proceedings of the 3rd SOQUA, pp. 54-61, ACM, 2006.