Machine Dynamics Research
2015, Vol. 39, No 1, 103 - 113

Using Similarity of Graphs in Evaluation of Designs

Barbara Strug
Faculty of Physics, Astronomy and Applied Computer Science, Jagiellonian University,
e-mail: barbara.strug@uj.edu.pl,

Abstract

This paper deals with evaluating design on the basis of their internal structures in the form of graphs. A set containing graphs representing solutions of similar design tasks is used to search for frequently occurring subgraphs. On the basis of the results of the search the quality of new solutions is evaluated. Moreover the common subgraphs found are considered to be design patterns characterizing a given design task solutions. The paper presents the generic concept of such an approach as well as illustrates it by the small example of floor layout design.

Keywords: design evaluation, frequent subgraph mining, design similarity,

1. Introduction

The idea of design patterns was introduced by Christopher Alexander in the domain of architecture and has been later adapted for the use in many other disciplines, including computer science [Alexander, 1977]. In his concept a pattern reflects the design decisions taken by many designers in different situations over the span of many years in order to solve a given problem. Although his work was based on architecture it can be extended to other domains of design. The use or lack of particular patterns can be linked to the quality of a given design. In order to use such an approach in design evaluation there is a need for a formal representation of the design knowledge, which usually requires the ability to capture geometric, structural and numerical information. On the basis of such knowledge representation the processing and evaluation of the designs can be done.

Graphs are considered to be a proven and well defined way of representing many complex objects in different domains of computer science [Rozenberg, 1997] such as engineering, system modelling and testing, bioinformatics, chemistry and other domains of science [Borkowski et al., 2003]. This paper is based one the generic notion of graphs but there are many different types of graphs extending simple graphs, for example hypergraphs or hierarchical graphs [Habel, Kreowski, 1987], which can be used in computer-aided design context. Designing new

artefacts can be thus considered to be equivalent to generating appropriate graphs representing them. By using such approach we can use any method of graph generation. There is a number of such methods including those based on formal languages [Rozenberg, 1997], graph grammars [Borkowski et al., 2003, Grabska et al., 2004,Nikodem, Strug, 2004], and grammar systems [Csuhaj-Varjú, 2004], but also evolutionary computations that were used in different domains of design [Borkowski et al., 2003, Grabska et al., 2004,Nikodem, Strug, 2004].

Each of these generative methods produces a large number of graphs - and at the same time a large number of designs they represent. Thus, it results in a basic problem of how to automatically or at least semi-automatically evaluate the quality of these graphs (where the quality of a graph is understood as the quality of the design it represents in respect to a given design problem). In many cases the process of evaluation is based on the use of a human designer who selects best solution or gives them some numerical values. The need for the human "evaluator" limits the number of possible solutions that can be analysed as all graphs have to be interpreted and visualized, what in many design problems can be complex and time-consuming. While the total elimination of a human evaluator is not the aim of this research it would be useful to eliminate as many designs as possible on the basis of their structural or spatial characteristics or on incompatibility with the design task being solved. Such problems could be found at the level of the representation without having to visualise all the designs. One of the possible approaches of eliminating the visualization step is to use the earlier designs, and their graph representations, for which a quality in respect to a given task is already known. Such a set of previous designs can be considered as corresponding to the “knowledge" or "experience" factor used by the human designers. As it can be observed that designs getting high quality evaluations often are similar in some way, that is. they share some common elements or patterns. Thus exploring this fact by finding frequently recurring patterns in graphs is proposed in this paper. The domain of frequent pattern analysis is well established in computer science and there is a number of available algorithms that can be used.

The paper is organized in the following way. In the next section a graph based knowledge representation is presented briefly, then in section 3 the use of graph patterns is described, including both theoretical information as well as their application to design evaluation. The approach presented in section 3 is illustrated with a case study based on the layout design in section 4. Finally in section 5 some conclusions are drawn as well as possibilities for further research are indicated.

2. Graph-based representations

There is a number of methods in which a design can be represented in computer-aided design context. Many of the methods used in CAD problems, like boundary representations, sweep-volume representation, surface representations or CSG (constructive solid geometry) [Hoffman, 1989, Mantyla, 1988], allow for the

geometry of an object being designed to be coded but do not take into account the inter-related structure of many design objects i.e. the fact that parts of an object can be related to other parts in different ways. As a result we only have a geometrical description of the design, while the designer often thinks and reasons rather in terms of structural properties of the design. Moreover such a representation makes it more difficult to reuse a selected part or parts of the design. Such a reuse of parts of earlier designs is very often used by designers to shorten the time needed to develop solutions to new, but similar design tasks.

A representation taking into account such a structural properties, in addition to the geometrical ones, is usually based on some type of graphs. Graphs consist of nodes and edges (or hyperedges in case of hypergraphs), commonly denoted as atoms. In simple graphs edges always connect two nodes, in hypergraphs a hyperedge can connect several nodes. Nodes and edges in graphs can be labelled and attributed. Labels are assigned to nodes and edges by means of node and edge labelling functions, respectively, and attributes - by node and edge attributing functions. Labels usually denote the type of the entity represented by a given atom or geometrical primitive used. In case of the floor layout problem the labels can for example denote type of the area (for nodes) or the relation (for edges) Attributes, on the other hand, represent properties, including all the geometrical properties, (for example size, position, but also colour or material) of a component or relation represented by a given atom. Moreover for a graph to represent one particular design all attributes must be assigned a proper value from the attribute domain. Such a graph with added labels and attributes represents all the knowledge about a given design.

In Fig. 1a an example of the floor layout is depicted and in Fig. 1b a graph representing such a layout is shown. Here, node labels represent types of spaces and edge labels represent the relations between the; adj represents the adjacency and acc - accessibility relation (thus acc is equivalent to the presence of doors in this example).

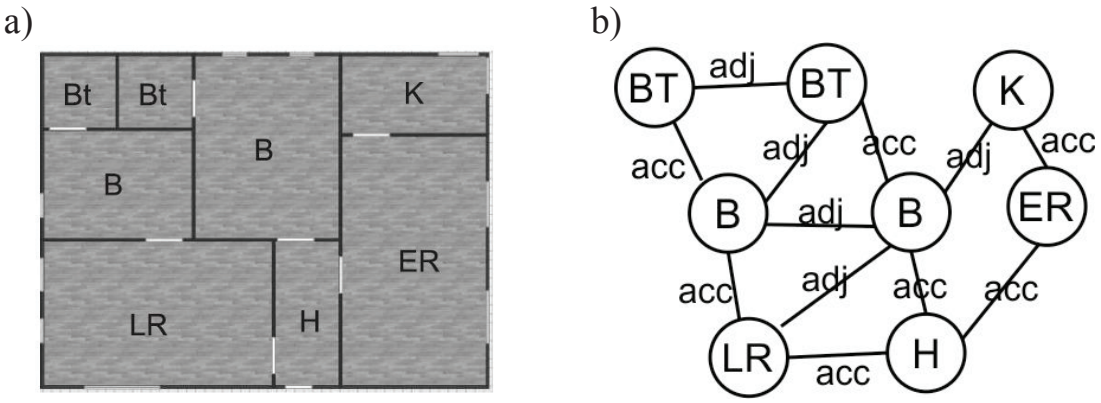

Fig.1. Example of a flat layout (a) and a graph representing this layout (b)

3. Using graph patterns

In computing frequent graph mining techniques were developed on the basis of a more general frequent pattern mining. Frequent pattern mining was first proposed by Agrawal et al. [Agrawal,1993] for market basket analysis in the form of association rule mining and later extended for other types of applications [Han et al., 2004, Inokuchi et al., 2000]. In graph mining a subgraph is considered frequent if its support, i.e. the number of graphs that contain this subgraph, is larger then some predefined threshold. The support is usually expressed as a percentage of graphs containing a given subgraph. Two of the most common algorithms are FFSM and gSpan. Both algorithms can work on undirected graphs with labelled nodes and edges [Yan et al., 2005]. They perform the analysis of graphs in a depth-first order and can only find connected subgraphs.

Let $D = \{G_1,...,G_m\}$ be a set of graphs representing designs (depicted on the right side of Fig.2), $P = \{p_1, ...,p_N\}$ be a set of frequent patterns found by the pattern mining algorithm (depicted in the middle in Fig. 2) and N be the number of these patterns (subgraphs). Thus it is possible to evaluate a new design, (for example represented by a graph depicted on the left in Fig. 2), is to calculate the number of frequent patterns contained in G. Let $m_1(G)$ be the a similarity value for the graph G, calculated as the proportion of frequent patterns that are subgraphs of G to the total number of frequent patterns. Let $P(G) = \{p_i: p_i \in G\}$ be the subset of P consisting of frequent patterns found in G. Thus

$$m_1(G) = \frac{|P(G)|}{N} \tag{1}$$

Thus the more frequent patterns a design contains the better it is considered. In many cases such approach gives good results as it corresponds to the fact that in many design tasks we actually construct new designs using "building blocks" tested and verified in many earlier designs.

Yet, such approach may not always be well suited to the type of designs analyzed as it only calculates the number of frequent patterns in the new design without taking into account other characteristics for example how often a given pattern is present in both the new design and in the designs included in the database, how large/small a given pattern is or what was the quality value of designs in which the pattern actually occurred.

Hence in some cases a more interesting results can be obtained for example by using the average frequency of a pattern in the designs present in the database and in the graph representing the new design. Let f_G^j denote the number of times the pattern p_j is present in the graph G. Moreover let $fq(p_j)$ denote the average frequency of occurrences of p_j in graphs of the database D. Using such a frequency the difference between the number of occurrences of a given pattern in a given graph and the average number of its occurrences in the database can be calculated.

The smaller the difference the more similar is the new design to the designs represented by the graphs in the database not only in terms of containing the same patterns but also similar number of the patterns. The difference is calculated according to equation (2):

$$diff(G) = \sum_{i=1}^{M} (f_G^j - fq(p_j)). \tag{2}$$

This value is higher as the difference is larger, thus to get the higher suitability score for smaller differences the score is calculated by taking the inverse of the difference, and setting a special MAX value in case of difference being equal to 0:

$$m_2(G) = \begin{cases} \dfrac{1}{diff\,(G)} & if\ \ diff\,(G) \neq 0 \\ MAX & if\ \ diff\,(G) = 0 \end{cases} \tag{3}$$

Such a way of evaluating the quality of a given design better reflects the influence of the number of occurrences of a given pattern in the new design as it takes into account a typical average number of times a given pattern may occur in the solutions of a given design task. It has to be observed that the frequency of occurrences of a given pattern can be calculated either in respect to all graphs in the database or only to those in which it actually is found thus giving different results, each of which can be useful in some situations.

The two evaluation methods above use only the number of patterns present in new design into consideration. They do not take into account any information about the quality of the design these patterns were mined from even if it is available. Yet, while the database of the designs used to mine for frequent patterns contains good, previously realized products there may still be difference in their quality so it may be useful to give preference to patterns mined from designs known to be, even slightly, better then other ones.

Thus let P(G) be a set of frequent patterns present in graph G from the database, let $r_i^j = 1$ denote the fact that pattern j belongs to graph G_i in database and $r_i^j = 0$ - that it does not and the same way let $r_G^j = 1$ denotes the fact that pattern j belongs to graph G and $r_G^j = 0$ - that it does not. Moreover let q_i be the quality of the design represented by graph G_i in the database. Thus $q(p_j)$ being the quality of pattern j is calculated according to the equation (4):

$$q(p_j) - \frac{\sum_{i=1}^{N} q_i r_i^j}{\sum_{i=1}^{N} r_i^j} \tag{4}$$

On the basis of such quality value a quality score for graph G can be calculated according to the equation (5).

$$m_3(G) = \sum_{j=1}^{M} q(p_j) r_G^j \tag{5}$$

This evaluation takes into account the quality of design solutions from which patterns were mined and gives more weight to those coming from better solutions. It can be seen as a parallel to the behaviour of the human designer who usually tends to take more "building blocks" from more successful earlier designs and less from worse.

In general the patterns are used to evaluate new design in the following way. In the first step the set of known designs/solutions to a given or similar task are encoded in the form of graphs. Then the frequent pattern searching algorithm is applied to this set of graphs. Then the set of patterns is used as the basis for the evaluation of new design(s). The process is schematically depicted in Fig. 2.

The process can be extended by using also the set of negative examples i.e. examples of bad or incorrect solutions of a given design task. In such a case a separate frequent patterns set can be generated from the bad designs producing as a result a set of negative patterns.

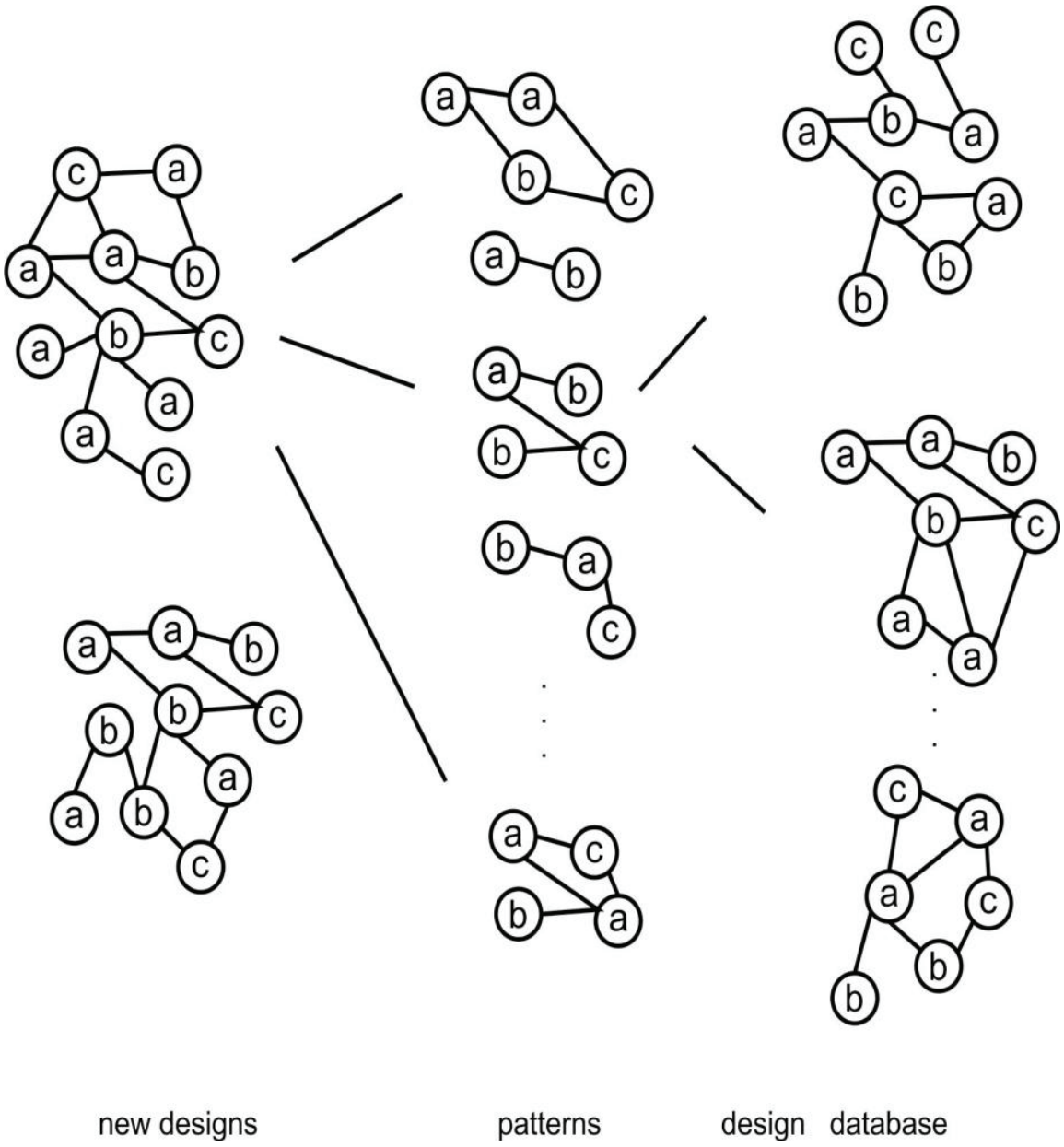

Fig. 2 Schematic example of the application of patterns to design evaluation

4. Layout design case study

The concept presented in this paper has been implemented and tested on examples of a floor layout design. The process starts by coding a database of floor layouts, consisting of a number graphs of size of 20 to 50 atoms, in GraphML format and importing to the GraphSearcher application [Tomanek, 2009]. Then the set of frequent patterns is generated and finally these patterns are used to evaluate new designs. Fig. 3 shows the application window containing one of the graphs analysed. The frequent design patterns found by the application (subgraphs) are then used as a basis for the evaluation of other graph representing new solution to the design task at hand.

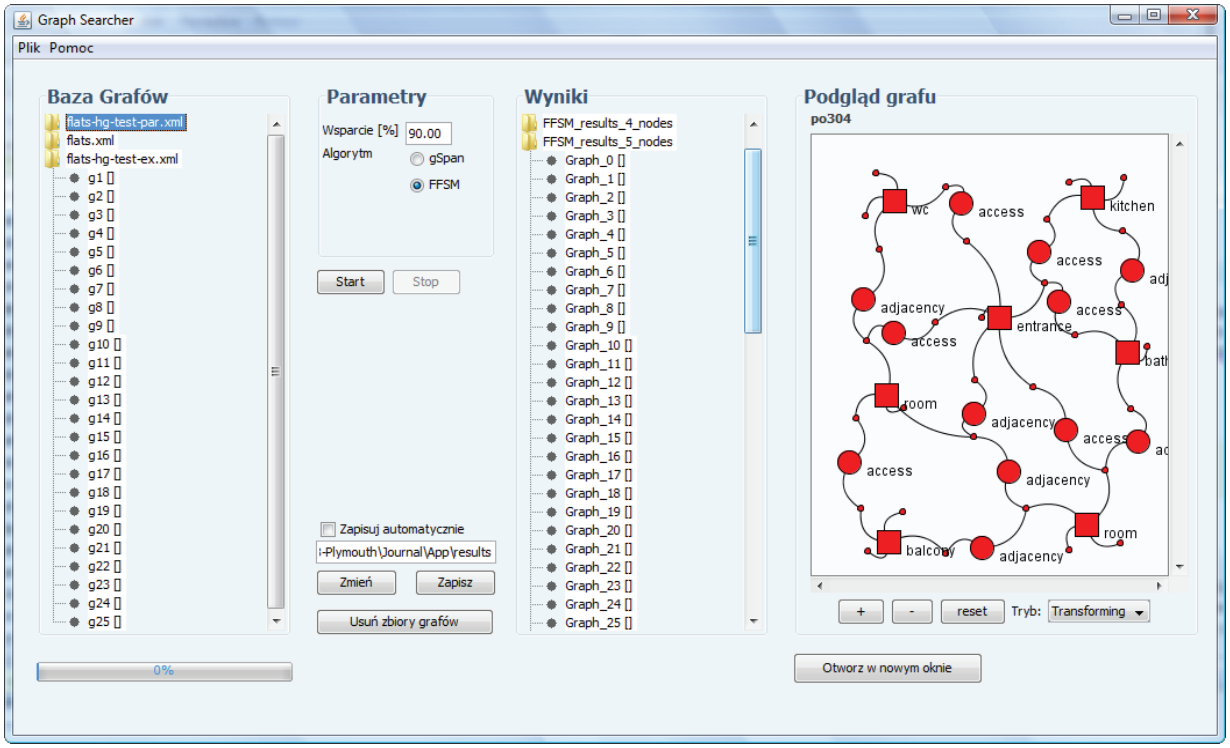

Fig. 3 GraphSearcher application used for floor layout graphs analysis

Examples of such frequent patterns are depicted in Fig. 4 a and b. They represent design requirements typical for flat's floor layout. The pattern depicted in Fig. 3a represents the existence of the access to some other space from the entrance, while the one depicted in Fig. 3b represents the existence of the access between the entrance and the kitchen.

Fig. 5 shows examples of patterns found in the set of bad examples. The one depicted in Fig. 5a represents the situation in which the bedroom is adjacent to four other spaces but is not accessible from them (i.e. there exists no door to/from this room). The pattern in Fig. 5b corresponds to the situation in which there exists a direct access between kitchen and bathroom which is usually undesirable.

On the basis of patterns found, the evaluation of six new design has been carried out. The results are gathered in Table 1. The designs G1-G3 are good designs, M1 and M2 - correct ones and B1 was a specially generated bad design.

Table 1 Evaluation results

	m_1	m_2	m_3
G1	0.92	0.43	0.32
G2	0.83	0.41	0.35
G3	0.91	0.31	0.19
M1	0.55	0.10	0.08
M2	0.62	0.08	0.10
B1	0.24	0.014	0.0012

It can be observed that the scores for the designs differ significantly for different evaluation approaches yet the ranking of them is similar: the better design obtain consistently better scores than bad ones, although small changes in ranking among good and medium designs can be observed. Thus the approach and scores obtained should be used as a ranking tool rather than treated as some absolute quality values.

It has to be noticed that as the analysis of the new graph involves determining which/how many frequent patterns it contains we have to deal with the problem of subgraph isomorphism which is known to be NP hard. As the result of using node labelling actually complexity of the problem is reduced and can be tackled in practice.

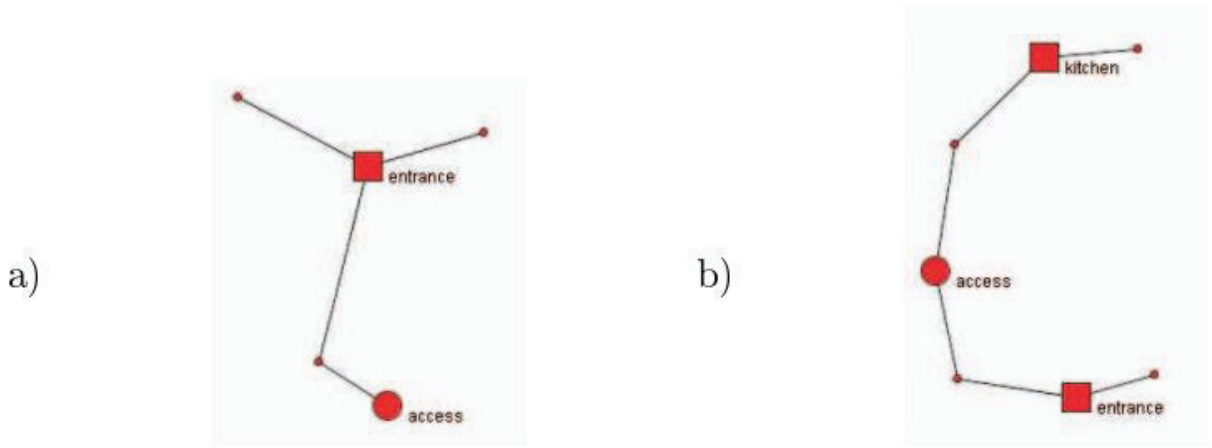

Fig. 4 Examples of frequent patterns representing design requirements

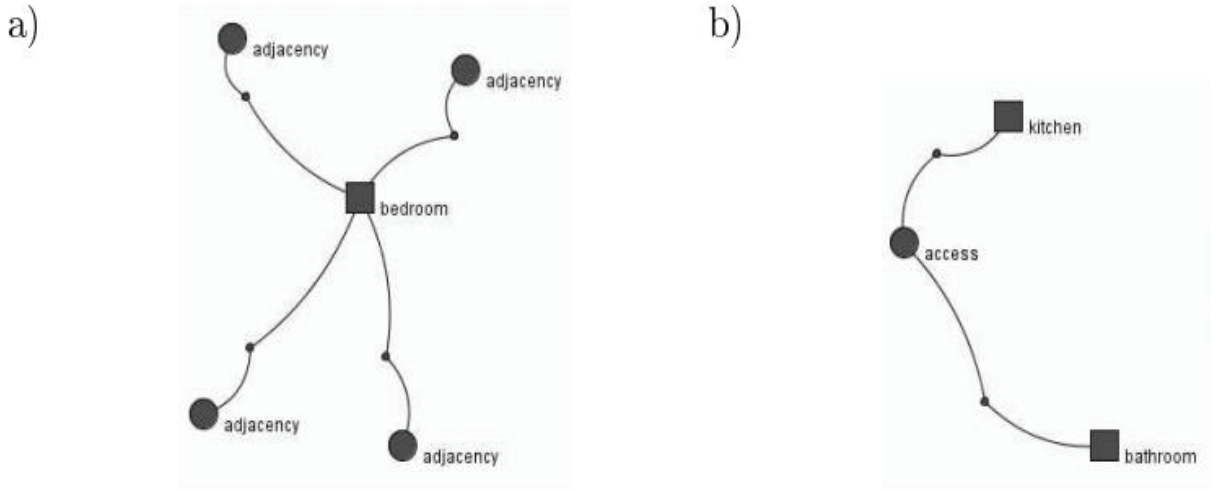

Fig. 5 Examples of negative patterns

5. Conclusions and future work

In this paper a general concept of design evaluation on the basis of design patterns is presented as well as some possible ways of calculating the quality value. This approach can be applied to many different types of design tasks in which some form of design patterns can be determined. Some results of such approach for different types of graphs were also presented in other works [Strug, Ślusarczyk, 2009, Strug, 20011a, Strug, 2011b, Strug, 2011c].

As noticed above methods of calculating the quality of design presented in equations (1) and (3) do not exhaust all possibilities. Many other factors are currently being researched. Some possible directions include considering the size of the pattern as larger patterns usually are more specific, or the co-occurrence of design patterns as some patterns can have higher impact on design quality when they co-occur in a design.

References

Agrawal R., Imielinski T., Swami A. (1993) Mining association, rules between sets of items in large databases. In: *Proc. 1993, ACM-SIGMOD* , Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26-28, 1993. pp 207--216, ACM Press 1993.

Alexander Ch, A Pattern language, Towns, buildings, constructions, *Oxford University Press,* ISBN 0-19-501919-9, 1977.

Borkowski A., Grabska E., Nikodem P, Strug B. Searching for Innovative Structural Layouts by Means of Graph Grammars and Esvolutionary Optimization, *Proc. 2nd ISEC Conf*, 2nd International Conference on Structural and Construction Engineering, Rome, Italy, Sep. 23-26, System-based Vision for strategic and Creative Design, vols 1-3, pp. 475--480. Rome 2003.

Csuhaj-Varjú E. Grammar systems: A short survey, *Proceedings of Grammar Systems Week* 2004: 141-157, Budapest, Hungary, 2004.

Grabska E., Nikodem P., Strug B. Evolutionary Methods and Graph Grammars in Design and Optimization of Skeletal Structures *11th ICE-, Proceeding of the 11th Workshop of the European Group of I Intelligent Commputing in Engineering, EG-ICE11, pp 145-155 Weimar Germany,* 2004 .

Habel A., H. J. Kreowski H. J., Some structural aspects of hypergraph languages generated by hyperedge replacement, *Lecture Notes in Computer Science* 247, pp. 207-219, *Springer*, 1987.

Han J., Pei J., Yin Y, Mao R. Mining Frequent Patterns without Candidate Generation: A Frequent-pattern Tree Approach. Data Mining and Knowledge Discovery, 8(1), pp 53-87, *Kluver*, 2004.

Hoffman C. M.,Geometric and Solid Modeling: An Introduction, *Morgan Kaufmann*, San Francisco, CA, 1989.

Inokuchi A., Washio T. Motoda H. A.,An Apriori-Based Algorithm for Mining Frequent Substructures from Graph Data, *Proc. of PKDD 2000*, Proceedings of the 4th

European Conference on Principles and Practice in Knowledge Discovery in databases, Lyon, France, Sep. 13-16, 2000, pp.87–92, 2000.

Mantyla M., An Introduction To Solid Modeling, *Computer Science Press, Rockville*, MD, vol.87, 1988.

Nikodem P., Strug B. Graph Transformations in Evolutionary Design, ICAISC 2004, *Lecture Notes in Computer Science* 3070, pp. 456--461, *Springer*, 2004.

Rozenberg G. Handbook of Graph Grammars and Computing by Graph. *Transformations, vol.1-3, World Scientific* London , 1997-99.

Strug B, Ślusarczyk G., Reasoning about designs through frequent patterns mining, *Advanced Engineering Informatics* 23, pp. 361--369, 2009.

Strug B., Using graph mining approach to automatic reasoning in design support systems, *Advances in Intelligent and Soft Computing,* vol. 95, pp. 489-498, 2011.

Strug B., Genetic Selection of Subgraphs for Automatic Reasoning in Design Systems, *Lecture Notes in Computer Science* 6678, pp. 280-287, *Springer,* 2011.

Strug B., Graph Similarity Measure in Automatic Evaluation of Designs, *Advances in Intelligent and Soft Computing*, vol 103, pp. 267-275, 2011.

Tomanek M. Searching for graph patterns and applications. *MSc Thesis, Jagiellonian University* (in Polish), 2009.

Yan X., Yu P. S., Han J., Substructure Similarity Search in Graph Databases, *SIGMOD'05*, Proceedings of the 2005 ACM SIGMODInternational Conference on Management of Data, Baltimore, MD, USA , June 13 - 17, 2005 pp 766-777, 2005. doi>10.1145/1066157.1066244