Hypergraph-based Evolutionary Design System

E. Grabska, B. Strug, G. Ślusarczyk.
Faculty of Physics, Astronomy and Applied Computer Science, Jagiellonian University, Reymonta 4, Cracow, Poland
e-mail uigrabsk}@cyf-kr.edu.pl, barbara.strug@uj.edu.pl,, gslusarc@uj.edu.pl

Abstract

This paper deals with applying evolutionary methods in computer aided design. The design process is an iterative one consisting of several steps. It starts with a preliminary or conceptual design, which is then analyzed or tested in order to find out which of its elements must be redesigned or refined. The process of evaluation and optimization is repeated until an acceptable solution is found. Since designing can be treated in computer science as a search process, where all possible designs form a search space, it is possible to use search techniques such as evolutionary ones.

As evolutionary search consists in evaluating and refining possible solutions, it can be seen as analogous to a human design iterative process of analysis, testing and optimization. Similarly to the refinement step in human design, in evolutionary search designs to be transformed are determined according to their evaluation (fitness). The refinement step is often performed not on actual solutions (phenotypes) but on their coded equivalents (genotypes).

Since in design problems genotypes in the form of binary strings are very often insufficient we propose to use a graph-based representation of genotypes which enables us not only to express geometrical properties of an object but also its attributes (like color, material etc.) and relations between object components.

In this paper we adopt hierarchical hypergraphs as they can represent an artifact with both multi-argument relations and hierarchical dependence which are impossible to express by other structures. The greatest advantage of this representation is its ability to describe in a uniform way all types of relations and objects and to produce highly fitted individuals.

Using hypergraphs in an evolutionary search requires the adaptation of traditional evolutionary operators like cross-over and mutation. As the hypergraphs selected to be transformed by the evolutionary operators at the subsequent stage of the evolution and their structures are not known a priori the operator must be defined in a way which allows for an "online" computation of new hypergraphs. Genetic operators working on hypergraphs and the structure of an evolutionary design system is presented.

The method is illustrated by examples of floor-layouts generated by a house design system, where structures of floor-layouts are represented by hypergraphs. In our approach a cross-over operation exchanges subgraphs representing the functional

areas with different internal arrangements, while mutation affects local and global attributes as well as the graph structure (by adding or deleting subgraphs)

1. Introduction

The design process, computer aided or traditional, is an iterative one consisting of several steps [1]. It starts with a preliminary or conceptual design, which is then analyzed or tested in order to find out which of its elements must be redesigned or refined. The process of evaluation and optimization is repeated until an acceptable solution is found. Still, majority of computer aided design systems focuses on refining parameters specifying design and optimizing it. They usually work on a single design at a time. Since designing can be treated in computer science as a search process, with all possible designs forming a search space, it is possible to use search techniques used in other domains.

There is a number of search methods well established in computer science that can also be used in the space of designs. [15]. One of them is an evolutionary technique. Instead of one solution at a time a larger subset of the search space, known as a population, is considered. As evolutionary search consists in evaluating and refining possible solutions it can be seen as analogous to a human design iterative process of analysis, testing and optimization [1,3]. Similarly to the refinement step in human design, which is based on earlier analysis and testing, in evolutionary search designs to be transformed are determined according to their evaluation (so called fitness). The refinement step is often performed not on actual solutions (called phenotypes, in this paper - designs) but on their coded equivalents (called genotypes).

In design problems genotypes in the form of traditionally used binary strings [1,4,11,14] are very often insufficient as not only geometrical properties of an object has to be represented but also its attributes (like color, material etc.) as well as relations between object components.

The methods used in CAD problems like boundary rep resentations, sweep-volume representation, surface representations or CSG (constructive solid geometry) [10,12,13] allow only for the "coding" of geometry of an object being designed and do not take into account the inter-related structure of many design objects i.e. the fact that parts (or components) of an object can be related to other parts in different ways. Such a structure is usually represented as a graph.

Different types of graphs have been used in this domain, for example composition graphs [6,7]and hierarchical graphs, in which relations such as being a part of or being included in were allowed [8]. An evolutionary design system based on these types of graphs was presented earlier in [18]. These graphs proved useful in different domains of design [2], but they lack the ability to represent structures in which more than two elements are related by the same relation. Such a possibility is given by a so called hypergraph. But traditional hypergraphs are in turn unable to represent hierarchical relations. Therefore in this paper we adopt hierarchical hypergraphs to evolutionary design as they can represent an artifact with both multi-argument relations and hierarchical dependence which are impossible to express with the use of traditional graph structures.

Using hypergraphs as a representation in an evolutionary search requires the adaptation of traditional evolutionary operators like cross-over and mutation. As the hypergraphs selected to be transformed by the evolutionary operators at the subsequent stage of the evolution and their structures are not known a priori the operator must be defined in a way which allows for an "online" computation of new hypergraphs. Thus the operator has to be specified by an algorithm rather than a set of rules.

An example of the application of this method is shown and some advantages and disadvantages of this approach as well as possible future research directions are briefly discussed. The method is illustrated by examples generated by a design system based on the proposed method.

2. Representation

Hypergraphs (HyGs) are a generalization of traditional graphs. They consist of nodes and hyperedges. What makes them different from standard graphs is that hyperedges in HyGs can connect an arbitrary number of nodes. The hyperedges are used to represent both relations and geometrical objects.

A hyperedge in a hypergraph may thus represent a geometrical object or a relation between a group of objects. These hyperedges are called object hyperedges and relational hyperedges, respectively.

Moreover in a hierarchical hypergraph a hyperedge may also be used to hide certain details of a designed object that are not needed at a given stage of design or to group object having some common features (geometrical or functional). Hyperedges that do not represent actual geometric entities or relations but are used to represent a hierarchical structure are called hierarchical

An example of a hierarchical hypergraph and a corresponding floor layout and layout design diagram are depicted in fig.1c, 1a and 1b, respectively. The hyperedges depicted as rectangles are object hyperedges, while the oval ones represent relational hyperedges. Nodes are depicted as small black filled circles.

Nodes and hyperedges in hypergraphs can be labelled and attributed. Labels are assigned to nodes and hyperedges by means of node and hyperedge labelling functions respectively, and attributes - by node and hyperedge attributing functions. Attributes represent properties (for example size, position, colour or material) of a component represented by a given hyperedge.

A labelled attributed hierarchical hypergraph may represent a potentially infinite number of designs having the same structure. To represent an actual design we must determine an instance of a hypergraph. An instance of a hypergraph is a labelled attributed hierarchical hypergraph. where to each attribute a value of the attribute domain has been assigned.

As such a hypergraph defines only a structure of a design, to create a visualisation of an object an interpretation is necessary. The interpretation determines the assignments of geometrical objects to object hyperedges, correspondence between relational

hyperedges and sets of relations between objects (components of a design). The geometry of these objects may be internally represented by means of any known representation that allows for easy application of similarity transformations. Geometrical objects used depend on the domain of application, for example when designing a house the set of geometrical objects could contain some primitive objects, or some predefined domain-oriented objects like doors, windows, stairs and other parts of a house and a set of relations could consist of an adjacency and accessibility relations.

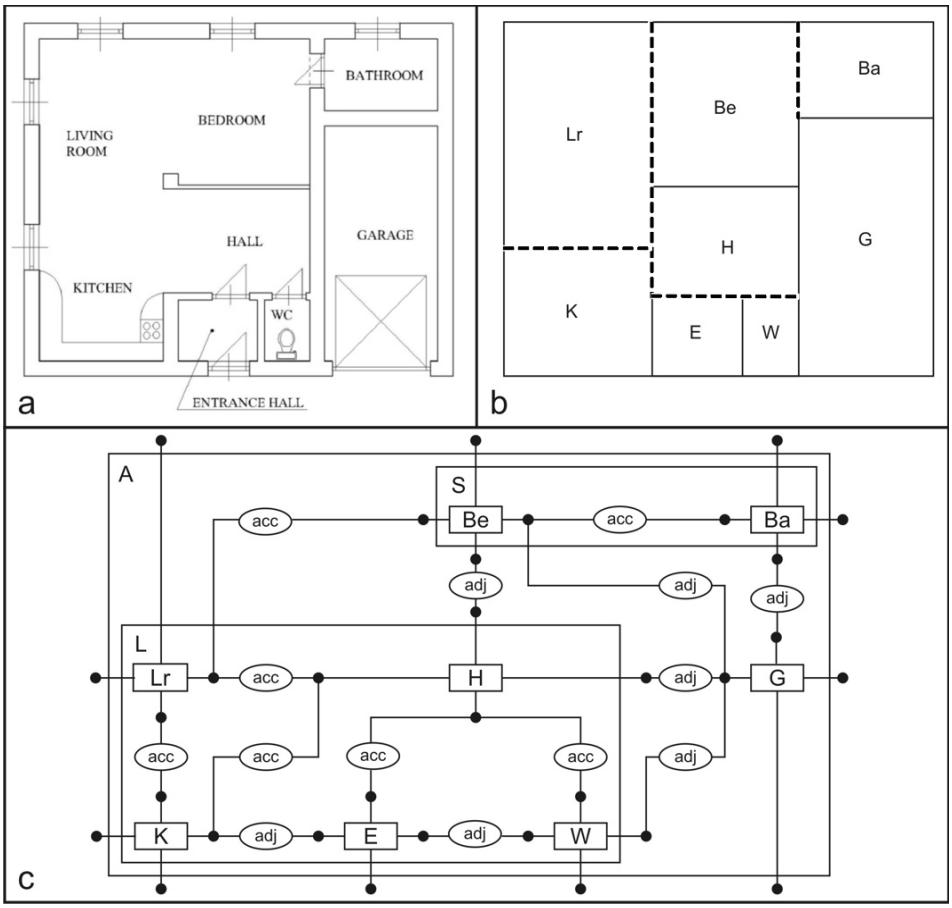

Fig. 1 A floor layout, a layout diagram and a corresponding hypergraph

A floor layout shown in fig.1a is the one of many possible designs the hypergraph from fig. 1c can represent. The layout was obtained after choosing an instance of this hypergraph and then interpreting it.

2 Evolutionary Design System

As it has been mentioned, a binary coding of design solutions is very often insufficient. This paper proposes to replace this standard coding by a hypergraph representation. To

use such a representation in an evolutionary design system a number of elements of this system must be defined.

Firstly a method of initialization must be chosen. One of the possibilities is to generate a population of random hypergraphs consisting of nodes and hyperedges from a given set. Although this method is easiest to implement in any design system it is usually very slow in producing acceptable or feasible designs as many designs are rejected. The other possible mechanism is known as a *graph grammar* and it has been successfully used in many domains to generate graphs [17]. Such a grammar describes all syntactically correct solutions, for example layouts.

It also possible to allow the user to generate an initial population of hypergraphs or to use hypergraphs generated by another program. Hypergraphs can be also generated using operations performed on hypergraphs [5].

2.1 Evolutionary graph operators

The genetic operators (usually a crossover and a mutation) constitute the next element of an evolutionary algorithm. As in this paper a nonstandard representation is used, new genetic operators have to be proposed.

The hypergraph based equivalent of a standard crossover operator requires establishing subgraphs that would be then exchanged. When a crossover is performed on two selected hypergraphs, H_1 and H_2 the subgraphs h_1 and h_2, respectively, are selected in these hypergraphs. Then each subgraph is removed from a hypergraph and inserted into the second one. As a result two new hypergraphs are generated. However there may exist hyperedges connecting nodes belonging to a chosen subgraph with nodes which do not belong to it. Such hyperedges are called embedding of a subgraph. So removing a subgraph from a graph and inserting it into another requires a method allowing for proper re-connection of these hyperedges. The underlying idea is that all hyperedges should be re-connected to nodes similar to those they were connected to in the hypergraph from which they were removed. There is probably more than one possibility of defining nodes' similarity.

In this paper a similarity-like relation is used. This relation is called {\it homology}. The name was inspired by the gene homology in biology. This relation is responsible for establishing subgraphs of selected hypergraphs that are homologous - or similar in some way- and thus can be exchanged in the crossover process. The definition of this relation is based upon the assumption that both hypergraphs selected for crossover code designs consisting of parts having similar or even identical functions (even if these parts have a different internal structure, material or/and geometrical properties).

In other words both hypergraphs are assumed to belong to the same class. The homology relation is defined on three levels that differ in terms of requirements put on hypergraphs to be in the relation. The weakest of these relations is called context free homology and it only requires two subgraphs to have the same number of object hyperedges with identical labels. It is the least restrictive of the three relations and it

allows for higher variety of new hypergraphs to arise from a crossover but at the same time it is able to produce the least meaningful hypergraphs or, in other words, the most "disturbed" ones.

On the opposite side the strongly context dependent homology is defined. It requires the top-level hyperedges in both subgraphs to have not only identical labels but also to have identically labelled ancestors up to the top-most level of the hypergraph hierarchy. Nevertheless the internal structure of a hyperedge and its attributes are not taken into account so even exchanging strongly homologous subgraphs may still produce considerably different new hypergraphs. When the context free relation is too weak, i.e., it results in too many hypergraphs being unacceptable (rejected by fitness function) and the strong homology is too restrictive or results in designs that are very similar or even identical to its parents the weakly context dependent homology may be useful. It takes into consideration direct ancestors of a given hyperedge but not any ancestors of higher levels in the graph hierarchy.

Formally, a crossover operator *cx* is defined as a 6-tuple (H_1, H_2, h_1, h_2, T, U), where H_1, H_2, h_1, h_2 are hierarchical hypergraphs and their subgraphs, respectively. The crucial elements of this operator are T and U that are called embedding transformations, i.e., they describe how hyperedges of the embedding are to be re-connected. They are sets of pairs of the form (n, n'), where n denotes a node to which a hyperedge was assigned originally and n' - the one to which it will be assigned in a new hypergraph.

It is important to notice however that the hypergraphs to be crossed over and their respective subgraphs are selected during the execution of the evolutionary algorithms so the embedding transformations can not be defined a priori (as it is in graph grammars [6,17]. Hence probably the most difficult problem is to find a method allowing us to establish these transformations. The algorithm generating these transformations requires only the subgraphs being exchanged to be homologous. For each level of homology a crossover operator is defined, thus we have three crossover operators having different levels of context dependence.

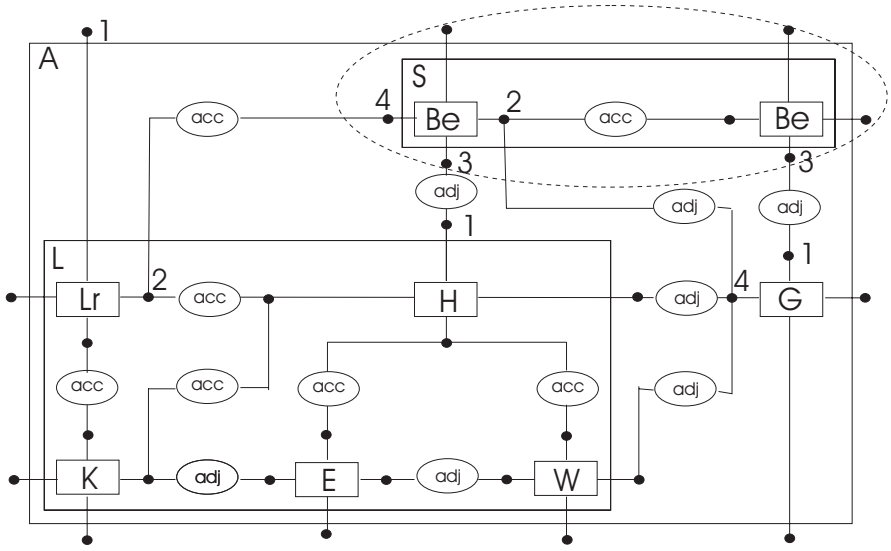

Fig. 2 A hypergraph H_1 representing a floor layout with selected subhypergraph h_1

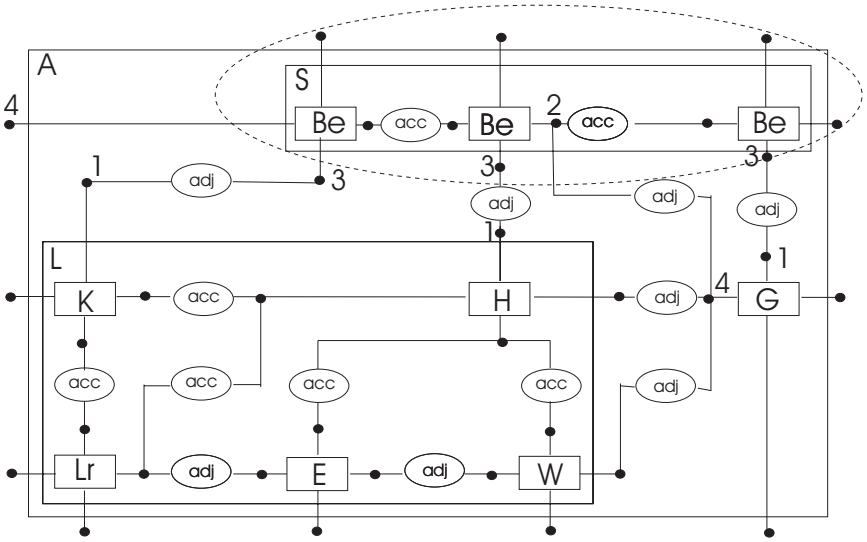

Fig. 3 A hypergraph H_2 representing a floor layout with selected subhypergraph h_2

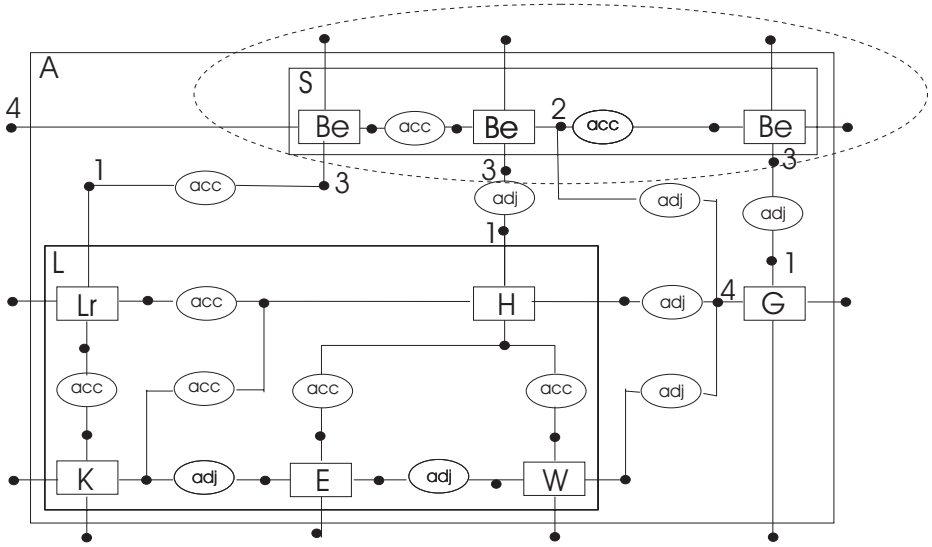

Fig. 4 A hypergraph H'$_1$ representing a floor layout resulting from cross-over operation

The idea behind the algorithm that generates automatically such an embedding transformation is to preserve the relations between the object hyperedges as much as possible i.e. to connect each hyperedge removed from one graph to a hyperedges in the second graphs that represent the same or similar object (i.e has the same label).

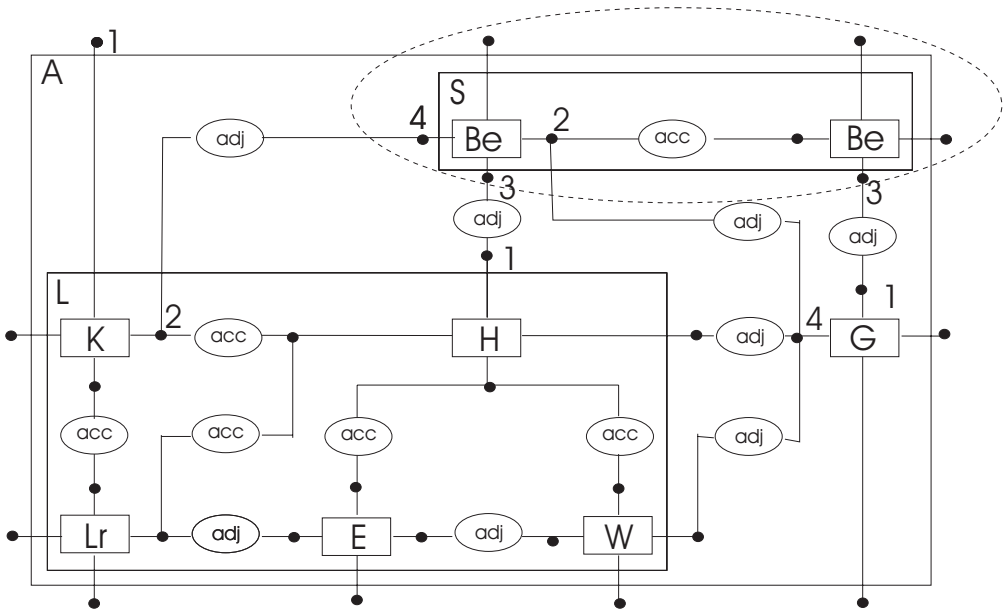

Fig. 5 A hypergraph H'$_2$ representing a floor layout resulting from cross-over operation

Example

Let us consider a house design system, where structures of floor-layouts are represented by hypergraphs. These hypergraphs are determined by a functional graph in which all required modules, represented by subgraphs, are defined. In this example the required functional modules include: a sleeping area, a living area and a garage. Other functional modules usually used in a house design include a communication area, a cooking area and sanitary one. In our example these functionalities are contained in other modules.

Applying a cross over operation we can exchange subgraphs representing the same functional areas: for example two living areas or two sleeping areas or just hyperedges representing single rooms that may have different internal arrangements. In fig. 2 and fig. 3 two hypergraphs, H_1 and H_2, representing layouts of two apartments (shown in fig. 6a and 6b, respectively) are depicted. Object hyperedges represent components of the apartment, while relational hyperedges, labelled *acc* and *adj*, represent accessibility and adjacency, respectively. The nodes connected by hyperedges are numbered and they denote walls of the rooms. For reasons of clarity in all figures numbers of nodes are shown only for nodes participating in crossover operator.

The subgraphs selected in H_1 and H_2,denoted h_1 and h_2, respectively, are surrounded by a dashed line. In this example a subgraph h1 consists of a hierarchical hyperedge labelled *S*, representing the sleeping area, and all its descendants (hyperedges and nodes assigned to them). In hypergraph H_2 a homologous subgraph h_2 was selected, that is one with the same label *S*, marked as the dashed oval in fig. 3.

The first step of crossover consists in removing selected subgraphs and their respective embeddings. The embedding of subgraph h_1 in H_1 consists of four relational hyperedges: a hyperedge labelled acc which connected node *4* of hyperedge labelled *Be* in h_1 and node *2* of hyperedge labelled *Lv* in H_1 - h_1, and three relational hyperedges labelled adj which connect nodes assigned to object hyperedges of h_1 with nodes assigned to object hyperedges of H_1 - h_1.

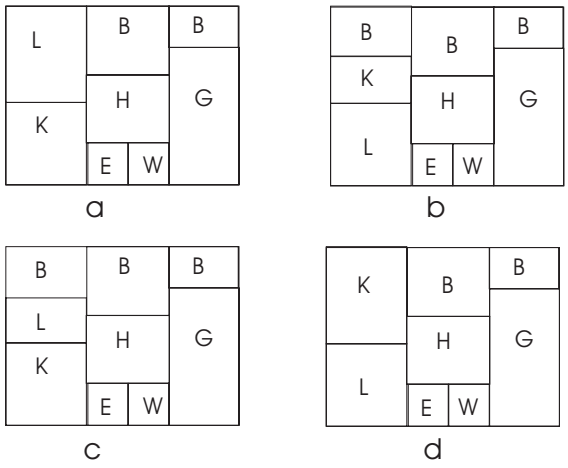

Fig. 6 a, b, c, d: Floor layout diagrams represented by hypergraphs from figs. 2,3,4 and 5.

Then the subgraph h_2 is put into the hypergraph. If an object hyperedge is connected to the object hyperedge with the same relation as in the source hypergraph the relation is preserved. Otherwise the relation is taken from the destination hypergraph. The hypergraphs resulting from crossing over the hypergraph depicted in fig.2 with the hypergraph in fig. 3 are shown in fig. 4 and fig. 5. The layouts represented by these hypergraphs are depicted in fig. 6c and 6d, respectively.

2.2 Mutation

As the second genetic operator mutation is usually used. This operator is much easier to be defined for hierarchical hypergraph-based representation.

The mutation operators may be divided into structure changing mutations and attributes changing ones. The second group can be further divided into local and global mutation operators.

The attribute changing operators change values of attributes of a selected object hyperedge (local mutation) or all object hyperedges (global mutation). As a result it changes geometrical properties of objects assigned to this hyperedge or hyperedges by the interpretation. However it is also possible to define mutation operators introducing structural changes to an artifact being designed which would not be possible using a binary representation. Such mutations could consist in adding or removing hyperedges from a hierarchical hypergraph. In the layout design system these mutations may for example result in adding or removing rooms.

So while crossover allows us to generate artifacts being combinations of previously existing designs, mutation may introduce wholly new elements into the object being designed.

3. Conclusions

Applying evolutionary methods to the design domain poses many problems. One of the main problems concerns representing designs in such a way that they can be easily modified during an evolutionary process. In the proposed approach a hierarchical hypergraph is used as a genotype and equivalents of standard genetic operators are defined on hypergraphs. Hypergraph-based operators are more complex than standard binary ones but we think that the benefits of using a hypergraph representation (possibility of coding multiple-argument relationships between components of an artifact and ability to introduce structural changes) compensate for it. The strongest point of a hypergraph-based representation is its ability to represent in a uniform way all types of relations and objects and to produce highly fitted individuals.

The use of graph grammars makes it possible to generate an initial population of graphs representing designs belonging to a desired class. Thus the graph grammar and fitness function are the only elements of the evolutionary design system that has to be changed in order to design different objects.

In this paper we evolve hypergraphs representing the structure of the whole design. In future we plan to run evolutionary process separately for each functional module. Then

the resulting solutions could be combined into one hypergraph structure, which can be farther evolved. Such an approach leads to a hierarchical evolutionary algorithm.

References

[1] P. J. Bentley, Generic Evolutionary Design of Solid Objects using a Genetic Algorithm, PhD thesis, UCL London 1), 3-38, (1997).
[2] Borkowski A., Grabska E., Nikodem P, and Strug B., Searching for Innovative Structural Layouts by Means of Graph Grammars and Evolutionary Optimization,, Proc. 2nd Int. Structural Eng. And Constr. Conf, Rome, (2003).
[3] De Jong K, Arciszewski T, and Vyas H, An Overview of Evolutionary Computation and its Applications, in. Artificial Intelligence in Engineerig,9-22, Warsaw, (1999).
[4] D.E.Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Reading, MA, Addison-Wesley, (1989).
[5] E. Grabska. A. Lachwa, G. Slusarczyk, K. Grzesiak-Kopec and J. Lembas; Hierarchical Layout Hypergraph Operations and Diagrammatic Reasoning, Machine GRAPHICS and VISION, (in print)
[6] E.Grabska,Theoretical Concepts of Graphical Modelling. Part one: Realization of CP-graphs. Machine GRAPHICS and VISION, 2(1993).
[7] Grabska, E.. Graphs and designing. Lecture Notes in Computer Science, 776 (1994).
[8] E.Grabska, W. Palacz, Hierarchical graphs in creative design. Machine GRAPHICS and VISION, 9(1/2), 115-123. (2000).
[9] Hajela P. and Lee, J, Genetic Algorith in Truss Topological OpJournal of Solids and Structures vol.32, no 22 , 3341-3357, (1995).
[10] Hoffman, C. M.,Geometric and Solid Modeling: An Introduction, Morgan Kaufmann, San Francisco, CA, (1989).
[11]Holland, J. H. Adaptation in Natural and Artificial Systems, Ann Arbor, (1975).
[12] Mantyla, M.,An Introduction To Solid Modeling, Computer Science Press, Rockville,MD,vol.87, (1988).
[13] Martin, R R and Stephenson, P C Sweeping of Three-dimensional Objects? Computer Aided Design Vol 22(4) (1990), pp. 223-234.
[14] Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag, Berlin Heidelberg New York (1996).
[15] Michalewicz, Z. Fogel, D. B.: How to Solve It: Modern Heuristics.. Springer-Verlag, Berlin Heidelberg New York (2000).
[16] P. Nikodem and B. Strug. Graph Transformations in Evolutionary Design, Lecture Notes in Computer Science,vol 3070, pp. 456-461, Springer, 2004.
[17] Rozenberg, G. Handbook of Graph Grammars and Computing by Graph. Transformations, vol.1 Fundations, World Scientific London (1997)
[18] B. Strug. Hierarchical Representation and Operators in Evolutionary Design, Parallel Processing and Applied Mathematics (PPAM 2005), LNCS vol 3911, pp. 447-454 Springer 2006.