Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2000), *14,* 17–29. Printed in the USA.
Copyright © 2000 Cambridge University Press 0890-0604/00 $12.50

Case-based evolutionary design

MIKE ROSENMAN
Key Centre of Design Computing, Department of Architectural and Design Science, University of Sydney, NSW 2006 Australia

(Received December 17, 1998; Accepted August 25, 1999)

Abstract

The paper provides a solution to the current problem of adaptation in case-based design. The aim of this project is to develop a case-based model of design, using an evolutionary approach, for the adaptation of previously stored design solutions. It is argued that such a knowledge-lean methodology is more general in its applicability than conventional case-based design. A prototypical example in the context of 2-D spatial design of houses is used to test the efficacy and efficiency of this approach.

Keywords: Evolutionary Design; Case-Based Design; Evolutionary Algorithms; Spatial Layout Design

1. INTRODUCTION

Case-based design, or CBD (Flemming, 1994; Oxman & Oxman, 1994; Maher et al., 1995; Maher & Pu, 1997), is an important area in the computability of design. While there has been much work in the indexation and retrieval of cases (Williams, 1995), many issues remain to be resolved. In particular, the adaptation process has, to date, stymied the utility of the approach, as each problem area has demanded problem-specific adaptation knowledge which has proven to be very difficult to formulate.

An evolutionary approach to the process of adaptation provides a general problem-independent methodology to the process of case-based adaptation. A general representation of design cases based on a genotypic representation of designs can be used with the general evolutionary process of reproduction and stochastic selection serving as the adaptation process. Moreover, by working with the entire population of existing design solutions, it makes the indexation and selection of relevant cases redundant. In addition, it has the capability to provide a number of potentially satisfactory solutions, rather than a single solution, from which a designer can make a final selection.

A model of evolutionary case-based design, or ECBD, will be exemplified through a computer program using a set of house designs as test cases. These designs will be represented using a genotypic representation of hierarchical form growth developed previously (Rosenman, 1996*a*, 1996*b*, 1997*a*; Rosenman & Gero, 1999). The outcome of such tests is the generation of new and diverse satisfactory (house) design solutions, under new design situations.

Reprint requests to: Mike Rosenman, Key Centre of Design Computing, Department of Architectural and Design Science, University of Sydney, NSW 2006 Australia. E-mail: mike@arch.usyd.edu.au

2. CONVENTIONAL CASE-BASED DESIGN—CCBD

There are two general computational models for the design process: design using general or compiled knowledge and design using specific or episodic or case knowledge (Rosenman et al., 1992). Compiled knowledge is knowledge which is generalized from a number of experiences without the specific details of the individual experiences. Specific or case knowledge is knowledge in which the actual experience, episode or solution is stored including all relevant details regarding the situation. When a design problem is given, a relevant experience is retrieved and adapted to the given problem. The main arguments for this latter approach are (1) that the general knowledge approach "re-invents the wheel" every time a problem is faced without taking into account previous experience; (2) the problem of acquiring the compiled knowledge; and (3) in many cases, the critical information associated with an individual is that which forms the exception to the "rule" rather than the commonalities. This information is lost in compiled knowledge.

CCBD (Flemming, 1994; Oxman & Oxman, 1994; Maher et al., 1995; Maher & Pu, 1997) is derived from the general method of case-based reasoning, or CBR (Schank, 1982; Riesbeck & Schank, 1989). According to Schank (1982), the most relevant experience (solution) is selected and "tweaked" accordingly to produce the required solution to the given problem. The main problems faced by CCBD are:

1. *The representation of design cases*: It is still a matter of discussion as to what exactly should be in a case, what its contents, structure, and representation should be (Rosenman et al., 1992). It has been argued that a design case should (1) just contain only the description of the design solution, (2) that it should contain the description of the problem, that is, goals that were used to produce the solution, or (3) that it should also contain the decisions that were made to select values to achieve the goals. The most common way to represent a case is in terms of attribute-value pairs. These represent a description of a design, in what in evolutionary terms would be called a phenotype description. Both flat and hierarchical structures have been used where, in the latter, a case may be composed of case snippets in a hierarchical structure. In all instances, the description of a case is predetermined according to a particular view of how to describe that design. All subsequent views of designs must follow that particular view.
2. *Indexing*: The ability of accessing information in a case base depends on the indexing schema used which depends on the nature of the case contents. Thus, only the information which is explicitly represented can be used for case indexing and hence accessed. This prescribes the nature of the queries and hence type of problems that can be addressed.
3. *Matching*: The matching of existing cases to the new problem is done in a number of ways such as counting the number of attribute entries in the case which match those of the required problem. Since in most instance there is little or no description of the structure properties of the required design, this involves looking only at the problem description, that is, goals. Sometimes both attributes and their values are taken into consideration and some matching done on the closeness of the values to those required. Attribute matching can be done by direct text match or through some ontological mapping.
4. *Retrieval and selection*: Two strategies are (1) to find only a "best-match" case and return it, in which instance there is no further selection; and (2) to select several "close-match" cases and return these to the user for final selection. In both instances, there needs to be an evaluation for finding "best-match" or "close-match".
5. *Adaptation*: While much work has been applied to the first four problems and various techniques exist (Maher et al., 1995; Maher & Pu, 1997), the problem of adaptation still remains a major obstacle to the general application of the approach. There is no general knowledge on how to "tweak" especially in the design domain and each design application and problem needs its specific adaptation knowledge. In addition, there is no guarantee that a so-called "best-match" or "close-match" will be amenable to adaptation in an efficient way. The selected case may be so tightly coupled that any modification leads to disintegration of the design. To date there has been little if any research on the potentiality of adaptation when selecting a case for adaptation.

So, in general the limitations of CCBD are that while design problems are situated, the fixed representation of case bases, in terms of the attributes described, prescribe the type of problems that can be addressed. In addition, the adaptation process is still largely a problem area.

3. EVOLUTIONARY ALGORITHMS

Evolutionary algorithms (Holland, 1975; Goldberg, 1989; Koza, 1992; Schwefel, 1995) cover several approaches but in general have the following features:

1. They are based on a genotypic representation.
2. They use a population of members.
3. Members evolve by "adapting" to their environment, represented by a fitness function, over a number of generations.
4. There is a reproduction mechanism for generating new members using genetic operators such as crossover and mutation.
5. Selection of members for propagation is based on the "survival of the fittest" principle using stochastic methods such as roulette wheel or tournament selection based on a member's "fitness". This fitness is a measure of the member's performance, with respect to the fitness function, relative to other members' performance.
6. Different fitness functions produce different solutions.

In summary, the evolutionary approach is a general knowledge-lean methodology aimed at producing solutions to difficult and complex problems where the relationships between the solution and the requirements is not well known. Within its methodology is the mechanism for adapting members to a particular given problem.

4. AN EVOLUTIONARY APPROACH TO DESIGN

Evolutionary techniques have lately been used in design research (Louis & Rawlins, 1991; Woodbury, 1993; Michalewicz et al., 1996), mainly for optimization of parameters but also for the generation of designs (Bentley & Wakefield, 1995, 1997; Rosenman, 1996*a*, 1996*b*, 1997*a*; Koza et al., 1996; Gero et al., 1997; Dasgupta & Michalewicz, 1997; Parmee, 1997; Parmee & Beck, 1997; Bentley, 1999; Rosenman & Gero, 1999). Recently, there has been some work on evolutionary methods in case-based design

(Gero & Schnier, 1995; Maher & Gomez de Silva Garza, 1996; Louis, 1997; Rosenman, 1997*b*; Gomez de Silva Garza & Maher, 1999; Louis & Johnson, 1999).

4.1. A hierarchical model of evolutionary synthesis of form

Simon (1969) points out that complex objects cannot be generated as a whole but only through a hierarchical decomposition into more manageable components. Current work by Rosenman in evolutionary design generation (Rosenman, 1996*a*, 1996*b*, 1997*a*; Rosenman & Gero, 1999) uses a hierarchical genotype representation for the generation of form. In a multilevel evolutionary approach, at each level, a component is generated from a combination of components from the level immediately below. At the element level, an element is generated from a combination of fundamental cells. Thus, the building blocks at each level are the components generated at the level immediately below and at each level the genotype contains genes which are assemblies of lower level genes. Through this approach, only those factors relevant to the design of a particular component or assembly need be considered for that component or assembly. These building blocks or components are labeled by a single symbol replacing the string containing the lower level genes in much the same way as automatically defined functions (ADF) are represented in genetic programming (GP) (Koza, 1994). A difference between this representation and that of ADFs as used in GP is that the representation used here is much more strictly hierarchical in nature. An ADF can occur at any level, whereas in this representation there is a strict component/assembly hierarchical order. In addition, each component is evaluated according to its own particular requirements rather than by an overall single evaluation function acting over the whole structure.

4.1.1. *Genotype and phenotype*

A design grammar, based on the method for conjoining counteractive vectors in the construction of polygonal shapes and counteractive faces in the construction of polyhedral shapes, is used (Rosenman, 1995, 1999). A polygon (polymino for orthogonal geometries) may be described by its sequence of edge vectors which is the phenotype, whereas a polyhedron may be described by a sequence of faces and vectors. A gene consists of two components (shapes) and their respective counteractive vectors or faces. The genotype, for a complex shape, is the sequence of such genes.

The following will give a simple description in the domain of polyminoes. For more complex and detailed descriptions including those for polyhedra, see Rosenman (1995, 1999). The phenotype representation of a (unit) square is (W1, N1, E1, S1,) where the symbols W, N, E, S represent the vector types, and the subscript being the instance of that vector type. The counteractive edge vectors are N, S and E, W. A polymino can be grown by adding one unit square in turn to the current shape at any edge. Two polyminoes can be joined through the conjoining of any pair of counteractive edges. There are several joinings possible in both growing a polymino and in joining two polyminoes. Figure 1 shows (a) a unit square; (b) a domino with its

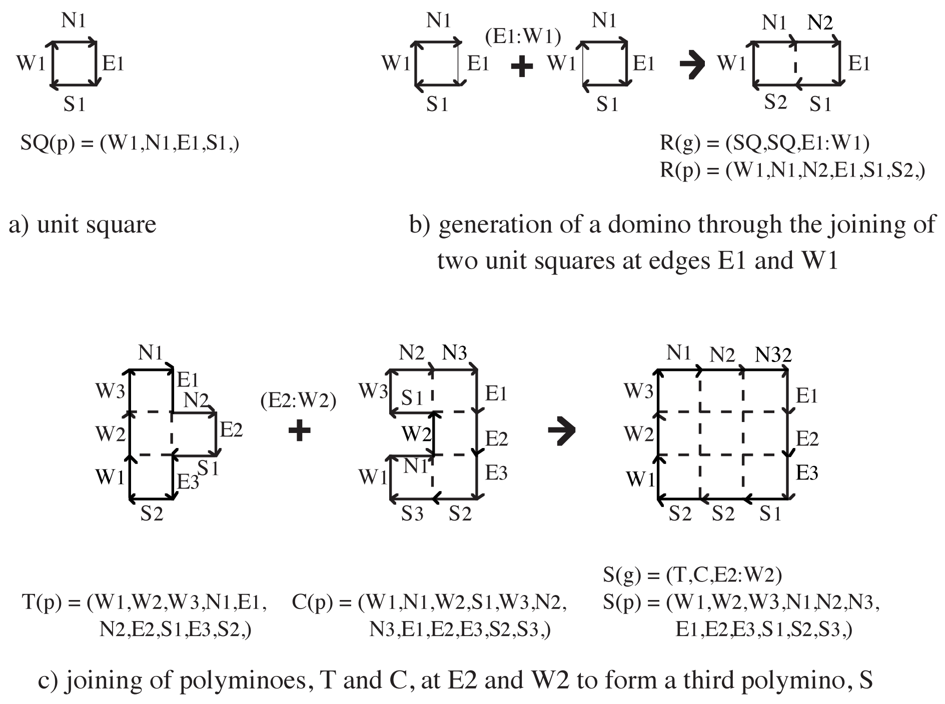

SQ(p) = (W1,N1,E1,S1,)
a) unit square
(E1:W1)
R(g) = (SQ,SQ,E1:W1)
R(p) = (W1,N1,N2,E1,S1,S2,)
b) generation of a domino through the joining of two unit squares at edges E1 and W1
(E2:W2)
T(p) = (W1,W2,W3,N1,E1, N2,E2,S1,E3,S2,)
C(p) = (W1,N1,W2,S1,W3,N2, N3,E1,E2,E3,S2,S3,)
S(g) = (T,C,E2:W2)
S(p) = (W1,W2,W3,N1,N2,N3, E1,E2,E3,S1,S2,S3,)
c) joining of polyminoes, T and C, at E2 and W2 to form a third polymino, S

Fig. 1. Growing and joining polyminoes.

genotype and phenotype descriptions; and (c) the result of joining two polyminoes through the conjoining of the counteractive edges, E2 and W2. The same result could be achieved by joining counteractive edges E1 and W3 or N2 and S1 or S1 and N1 or E3 and W1.

Note that, while in this work the phenotype interpretation of the genotype is a sequence of edge vectors representing a description of a shape, for different problems the phenotype interpretation could differ. For example, it may be a description of the type of unit, for example, its material, and its location either globally or relative to the composition in which it is located.

4.1.2. *Hierarchical evolution in the domain of house plans*

The domain of house plans will be used to describe the hierarchical nature of the representation, although the representation and concepts are general in nature. A house can be composed of zones where zones are composed of rooms and rooms, in turn, composed of number of unit space units, unit squares, depending on the area of the room required. A house is generated by firstly generating the rooms required, then generating the zones as a composition of rooms and finally by generating the house as a composition of zones. In a hierarchical evolutionary process, a room is generated by first randomly generating a number of shapes of the given number of unit squares and then evolving the shapes over a number of generations. A number of satisfactory shapes can be selected for each room. Each zone is then generated by initially generating a population of zone solutions composed of one of each required room type randomly chosen from each room type set generated and through randomly selected edge joinings. The zone population is then evolved to arrive at a number of satisfactory zone solutions. The house is generated and evolved in a manner similar to that of the zones, by randomly selecting and joining zones and evolving the population. Figure 2 shows the house hierarchy, and Figure 3 shows the general and hierarchical evolutionary process.

Figure 3 is diagrammatic only as the hierarchy may occur over a number of component/assembly levels and evolution occurs over a number of generations. The final composition of components and objects is thus a mix of several elements from those contained at the initial generation even though it appears as directly composed from the initial random generation.

Examples of genotypes and phenotypes for each level of the house hierarchy are shown in Figure 4, where the phenotypic information is the shape of the component. In the genotype description of rooms, the unit square has been omitted for simplicity and the genotype represented as a list of edge vector conjoinings. In the description of the phenotype, multiple instances of the same edge type are indicated as G1,...,*n*, where *G* indicates the edge type (N, E, S, or W) and *n* the last instance of that type in the sequence, so that W1,...,3, indicates W1, W2, W3.

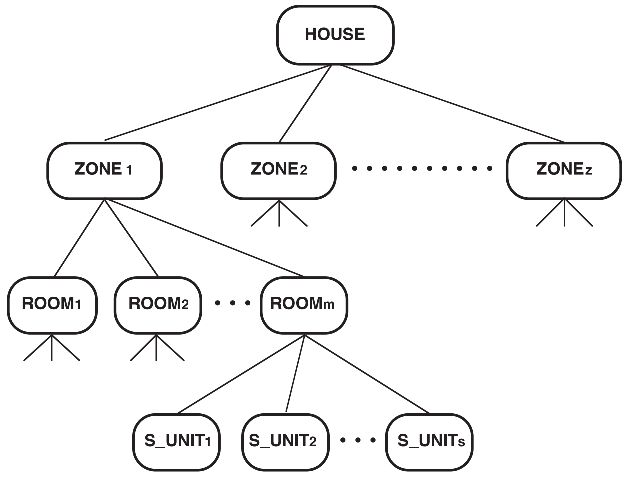


Fig. 2. House hierarchy.

Simple crossover is used for the propagation of members during the evolution process. A biased random roulette wheel method is used for selection of members for "mating" and for selection of members for new generations. Although the examples use two-dimensional (2-D) spatial forms for houses, using a square cell, the methodology is more general and the grammar can be applied to any polygonal configurations. The grammar has been extended to three-dimensional (3-D) polyhedra (Rosenman, 1999).

5. EVOLUTIONARY CASE-BASED DESIGN—ECBD

CCBD was based on Schank's model of CBR (Schank, 1982), whereas evolutionary case-based design (ECBD), in contrast, uses Calvin's model (Calvin, 1987, 1996, 1998). In Schank's model a particular closest match experience is selected and "tweaked" (adapted), whereas in Calvin's model thinking tasks are achieved by Darwinian competition to organize complex patterns.

Existing house designs can be formulated using the above hierarchical representation resulting in a case base of house designs represented by their genotype which encodes how they are to be synthesized. Given a particular set of requirements formulated as a fitness function, this case base is used in its entirety as the initial population for an evolutionary process which will evolve new and satisfactory solutions to the given problem. Starting from a population of existing house designs rather than generating a solution from the beginning has the advantage that higher level components, rooms and zones, already exist. Thus the efficiency of the process is greatly increased. An example of this is given in Figure 5 where two house solutions, H1 and H2, with low perimeter-to-area ratios, are selected as parents to produce, after crossover, the solutions H3 and H4. The solutions found are satisfactory solutions to a design problem using a fit-

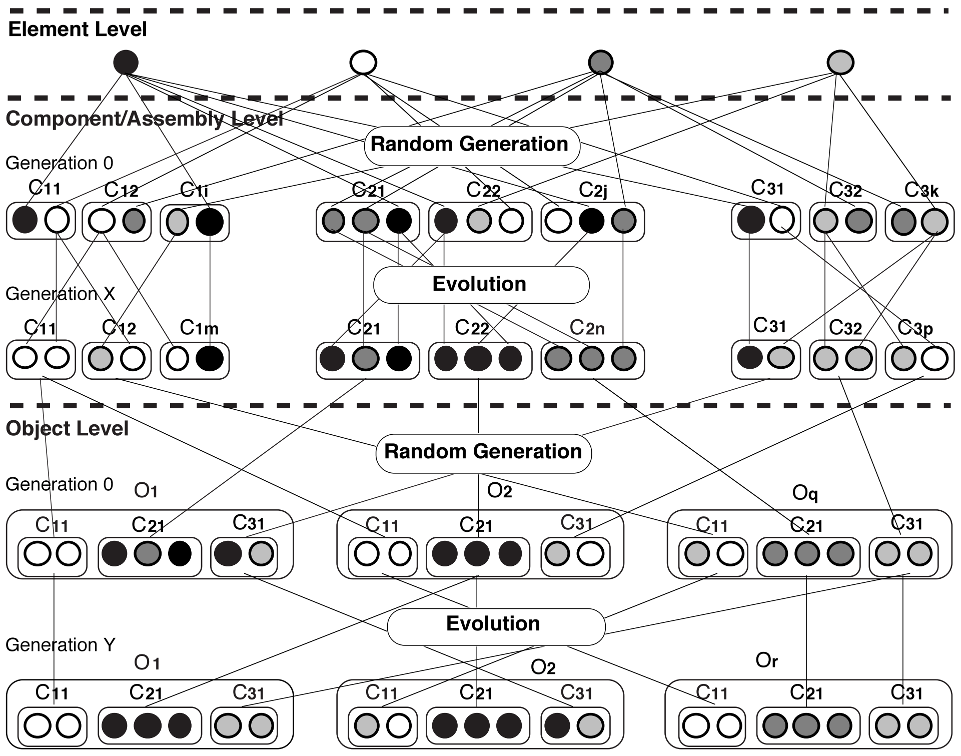


Fig. 3. General hierarchical evolutionary process.

ness function of high perimeter-to-area ratio which reflects the need for solutions for cross ventilation in a hot humid climate.

One of the shortcomings in conventional case-based design, CCBD, is the concentration on selecting the best-matched case in the assumption that this will lead to simple adaptation. This assumption breaks down when the selected case is so tightly constrained that any "tweaking" causes major perturbances. It may well be that a solution not so closely matched is actually easier to adapt. In an evolutionary case-based design approach, ECBD, it is not necessary to preselect members for adaptation as the general methodology of the approach will progressively remove less fit members from future populations. Moreover, since direct relationships between the structure (form) and function (performance) are not always known, seemingly poor performing solutions may have elements which are useful in combination with other elements of other members. Fig-

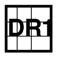
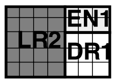
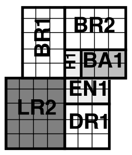
Element	Shape	Genotype	Phenotype
Room DR1		(E1:W1,N2:S1,N1:S1,W2:E1, S1:N1,W3:E1,N3:S1,E3,W1)	(N1,E1,E2,E3,S1,S2, S3,W1,W2,W3,N2,N3,)
Zone LZ1		((LR2,DR1,E4:W2),EN1,N3:S1)	(N1,2,3,E1,...,5,S1,2..7,W1,...5,N4,...7,)
House H1		(LZ1, BZ1, N7:S5)	(W1,...5, N1,...7, E1,..5,S1,E6,...,10, S2,...,8,W6,...10,)

Fig. 4. Example of room, zone, and house genotype and phenotype.

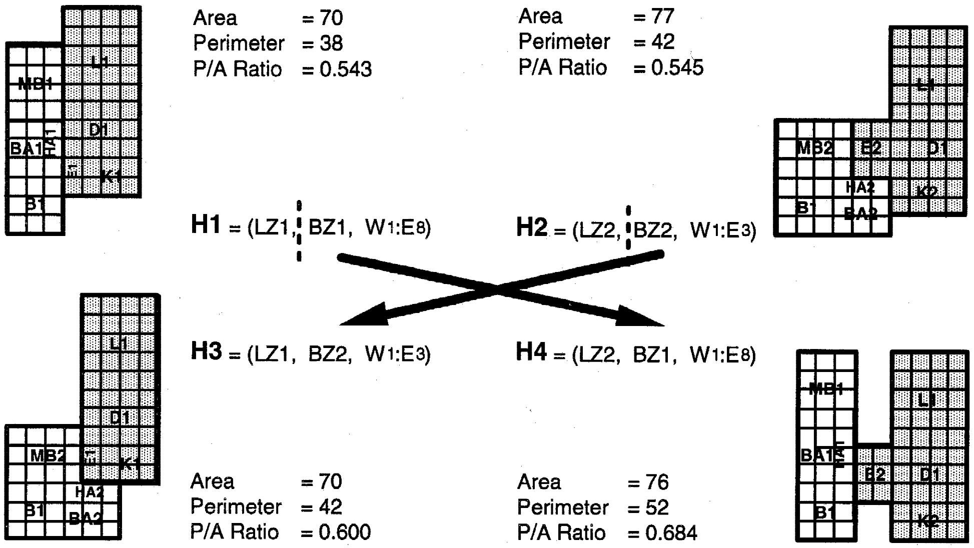


Fig. 5. Evolution of house plans resulting from a new fitness function.

ure 6 shows a diagrammatic comparison between the CCBD and ECBD approaches.

Figure 6 shows the different modules corresponding to the different functions required, the necessity for CCBD to have a knowledge base for adaptation knowledge, containing both domain specific and general knowledge and the fact that only one solution is produced with CCBD as against several solutions in ECBD. The domain-specific knowledge, together with the case representation, prescribes the type of problems that can be solved. All possible situations must be accounted for. In contrast, the evolutionary process has a general adaptation methodology. Domain-specific knowledge is built into the fitness function and hence into the interpretations of the genotype and evaluation of the phenotypes.

6. RELATION TO EXISTING WORK ON EVOLUTIONARY CASE-BASED DESIGN

There is existing work in the area of evolutionary case-based design by Gero and Schnier (1995), Maher and Gomez da Silva Garza (1996), and Louis (1997). Gero and Schnier use more direct learning in a genetic engineering approach to first evolve useful building blocks (complex genes) and then reuse them in combination with the basic genes in a manner similar to GP. The learning is carried out from a set of examples containing both good and bad solutions. The work presented here uses a more structured representation where the components are meaningful entities subject to their own requirements rather than just substrings. Moreover, there is no requirement of the case base containing any good so-

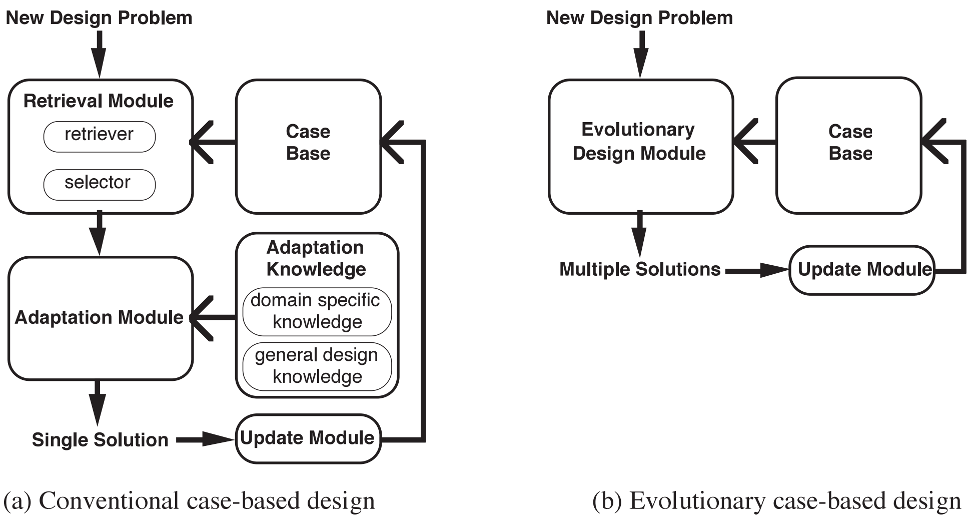


Fig. 6. Diagrammatic comparison between conventional and evolutionary case-based design.

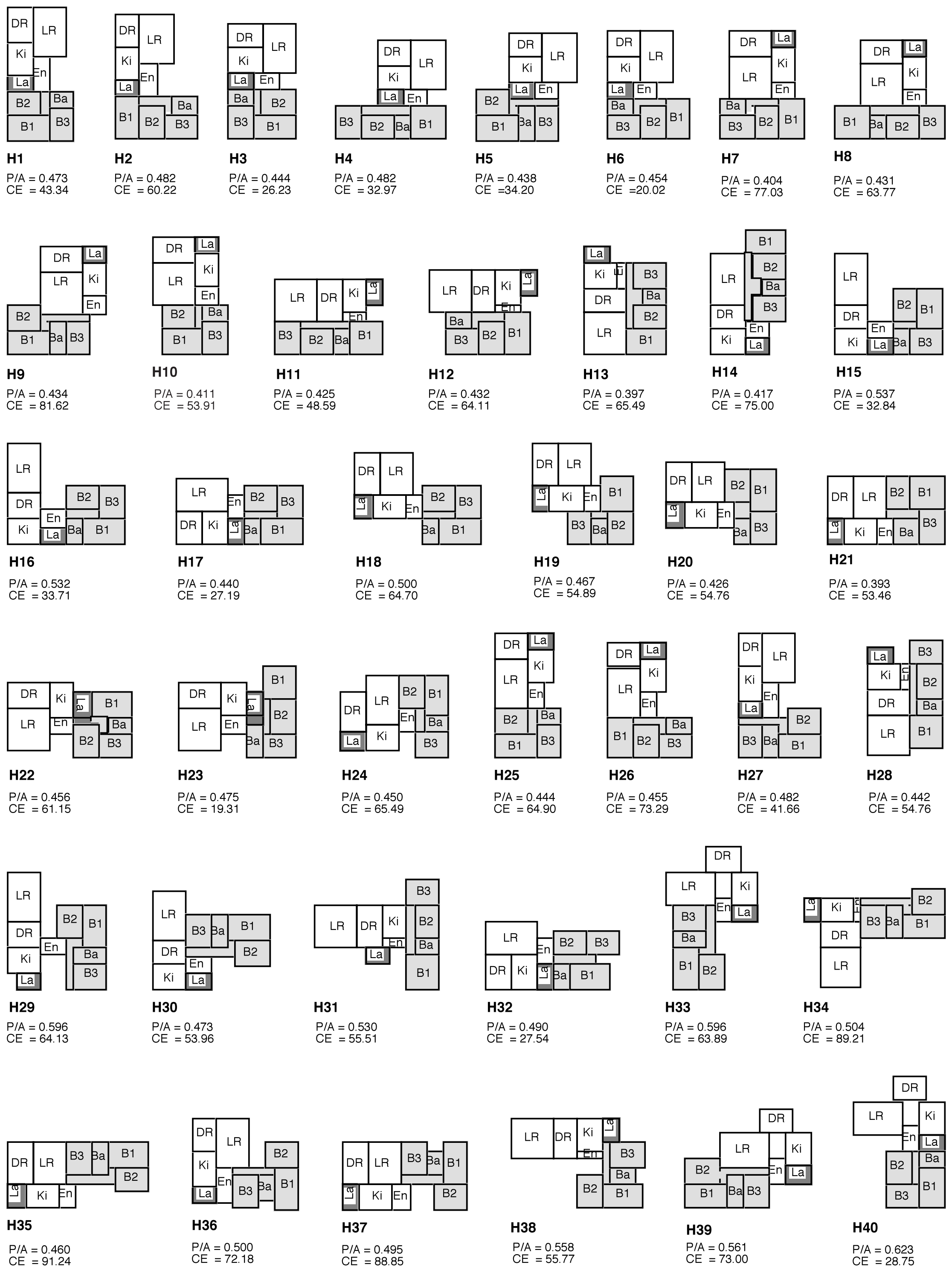

H1 P/A = 0.473 CE = 43.34
H2 P/A = 0.482 CE = 60.22
H3 P/A = 0.444 CE = 26.23
H4 P/A = 0.482 CE = 32.97
H5 P/A = 0.438 CE =34.20
H6 P/A = 0.454 CE =20.02
H7 P/A = 0.404 CE = 77.03
H8 P/A = 0.431 CE = 63.77
H9 P/A = 0.434 CE = 81.62
H10 P/A = 0.411 CE = 53.91
H11 P/A = 0.425 CE = 48.59
H12 P/A = 0.432 CE = 64.11
H13 P/A = 0.397 CE = 65.49
H14 P/A = 0.417 CE = 75.00
H15 P/A = 0.537 CE = 32.84
H16 P/A = 0.532 CE = 33.71
H17 P/A = 0.440 CE = 27.19
H18 P/A = 0.500 CE = 64.70
H19 P/A = 0.467 CE = 54.89
H20 P/A = 0.426 CE = 54.76
H21 P/A = 0.393 CE = 53.46
H22 P/A = 0.456 CE = 61.15
H23 P/A = 0.475 CE = 19.31
H24 P/A = 0.450 CE = 65.49
H25 P/A = 0.444 CE = 64.90
H26 P/A = 0.455 CE = 73.29
H27 P/A = 0.482 CE = 41.66
H28 P/A = 0.442 CE = 54.76
H29 P/A = 0.596 CE = 64.13
H30 P/A = 0.473 CE = 53.96
H31 P/A = 0.530 CE = 55.51
H32 P/A = 0.490 CE = 27.54
H33 P/A = 0.596 CE = 63.89
H34 P/A = 0.504 CE = 89.21
H35 P/A = 0.460 CE = 91.24
H36 P/A = 0.500 CE = 72.18
H37 P/A = 0.495 CE = 88.85
H38 P/A = 0.558 CE = 55.77
H39 P/A = 0.561 CE = 73.00
H40 P/A = 0.623 CE = 28.75

Fig. 7. Test case base of 40 houses.

lutions although obviously there must be sufficient possibility to evolve to good solutions. While both Louis and Maher and Gomez, have similar conceptual approaches, both carry out preselections on the case base before carrying out evolution. In the case of Maher and Gomez, this results in a very small population for the evolution process. Louis mentions that a problem exists in deciding how to select the population so as not to lose some potentially critical information. Moreover, Louis's example is in the domain of scheduling rather than design synthesis. Maher and Gomez use a flat genotype representation where genes directly represent attributes of buildings. Thus the representation in this work, which stems from the concept of hierarchical aggregation of atomic cells into more complex elements, forms a major difference between this work and other work.

7. IMPLEMENTATION

7.1. Case base

A test case base of 40 houses was formulated (Figure 7) using the genotypic representation previously described. Each house consists of three zones, namely the Living Zone, Utility Zone, and Bed Zone. Each zone type can be rotated 90, 180, or 270°, (R90, R180, or R270) or mirrored along the X or Y axes (MX or MY) or mirrored and rotated (MXR90, MYR90). Each house genotype is 3 + 2 (zones + edge joinings) long. The genotypes of each zone and room is not given here but as an example, the following zone genotypes are:

Element	Genotype
LZ1	{[(LR1, DR1, W6, E1), Ki1, S7, N1], En1, E7, W4}
UZ1	La1
BZ2	({[(MB2, Ha2, N5, S1), B1, N1, S4], Ba1, E1, W2}, B1(R90), E3, W2)

The genotype of house, H1, can be given in a tree form as

(({[(LR1, DR1, W6, E1), Ki1, S7, N1], En1, E7 W4},
[La1(R90)], S1, N3),
({[(MB2, Ha2, N5, S1), B1, N1, S4], Ba1, E1, W2},
B1(R90), E3, W2), S3, N5)

as shown in Figure 8. The room genotypes as generations of unit squares are not given here, for simplicity, but could be similarly expanded and added to the tree representation.

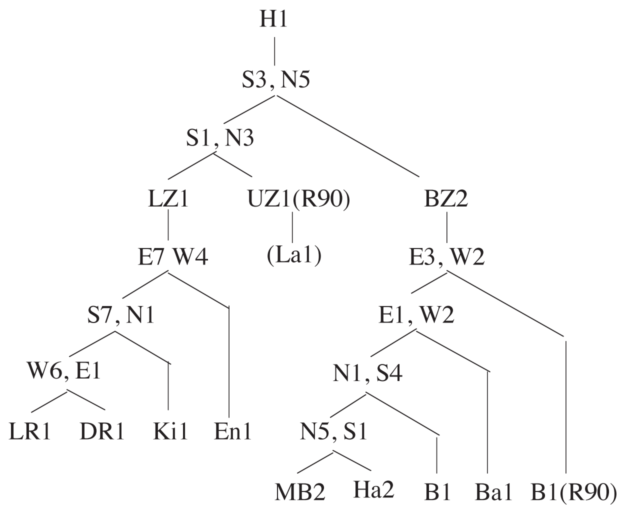


Fig. 8. Tree representation of house H1 genotype.

7.2. New problem formulation

As a first test to investigate possible issues, a design problem was formulated where the aim is to design houses suitable for hot humid climates. Such houses require the maximum cross-ventilation possible. As an approximation, this was formulated as maximizing the perimeter-to-area ratio, P/A. In addition, it was required that the circulation efficiency of the house remained reasonably efficient. The problem could have been formulated as a Pareto optimization problem with two criteria, namely maximizing the P/A ratio and minimizing a weighted sum of distances between rooms with given closeness requirements. However, it was felt that houses with very bad circulation efficiencies (CE) might very well produce large P/A ratios but be totally unsuitable. Hence, the fitness function was formulated to include a penalty function to penalize solutions for poor circulation efficiency. The fitness function was then formulated as:

$$\text{Max}F = (P/A * 100) * \text{P},$$

where P is a penalty and

$P = 1$	if $CE \leqq 100$
$P = 0.75$	if $100 < CE \leqq 150$
$P = 0.5$	if $150 < CE \leqq 250$
$P = 0.25$	if $250 < CE \leqq 500$
$P = 0.1$	if $500 < CE$

The penalty values were obtained from inspection of house solutions with acceptable and near acceptable circulation efficiencies.

In Figure 7, the fitness of each house is given in terms of its perimeter-to-area (P/A) ratio as well as its circulation efficiency (CE). The cases are divided into two groups, cases 1–26 which are in general very compact houses and cases 27–40 which include some less compact houses, the aim being to test what information is necessary for a case base to produce satisfactory results. For houses 1–26, the lowest P/A is 0.393 (house H21), the highest is 0.537 (house H15), and the average P/A is 0.450. For houses 27–40, the lowest P/A is 0.442 (house H28), the highest is 0.624 (house H40), and the average P/A is 0.523, giving an average P/A of 0.475 for the 40 cases.

7.3. Tests

A total of five tests were carried out. Each test was carried out over a number of runs with each run over a number of

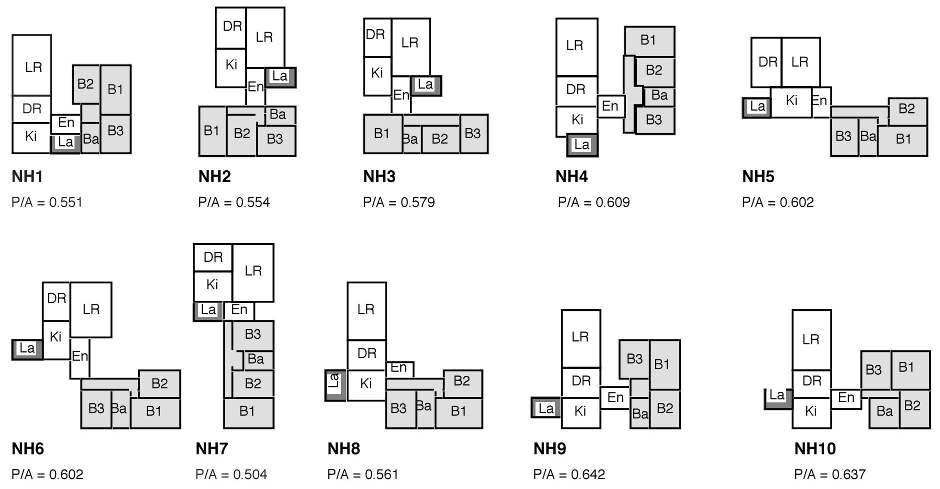


Fig. 9. Ten houses resulting from Test 1—26 cases.

generations until 10 solutions deemed satisfactory were found. The first four tests are case-based tests while the fifth test is a generation of houses using the growth process to be used as a comparison against the case-based results. This growth process involves the generation of rooms, zones, and houses. The tests are as follows:

- Test 1: crossover limited to the house level, that is, only between zones and the joining edges. Only the top part of the tree in Figure 8 was allowed for cut sites, so that only the zones with the orientation and the edge joinings as existing in the cases could be used in new combinations—26 cases.
- Test 2: crossover limited to the house level as for Test 1 but allowing random edge joinings between zones—26 cases.
- Test 3: as for Test 1—40 cases.
- Test 4: as for Test 2—40 cases.
- Test 5: growth process.

For the case base of 26 cases, there are 10 Living Zone types, 12 Bed Zone types, and 2 Utility Zone types. The augmentation to 40 cases resulted in 11 Living Zone types, 14 Bed Zone types, and 2 Utility Zone types. For Test 5, 16 Living Zone types, 2 Utility Zone types, and 12 Bed Zone Types were generated.

7.4. Results

Figures 9–13 show the 10 house results found for Tests 1–5, respectively.

Table 1 shows some of the statistics from the five tests. The number of runs and generations for Test 5 are the total for all rooms, zones, and the houses. The number of runs for the rooms and zones was 36 and the number of generations for the rooms and zones was 567. Thus, for Test 5 the number of generations for the houses was 290.

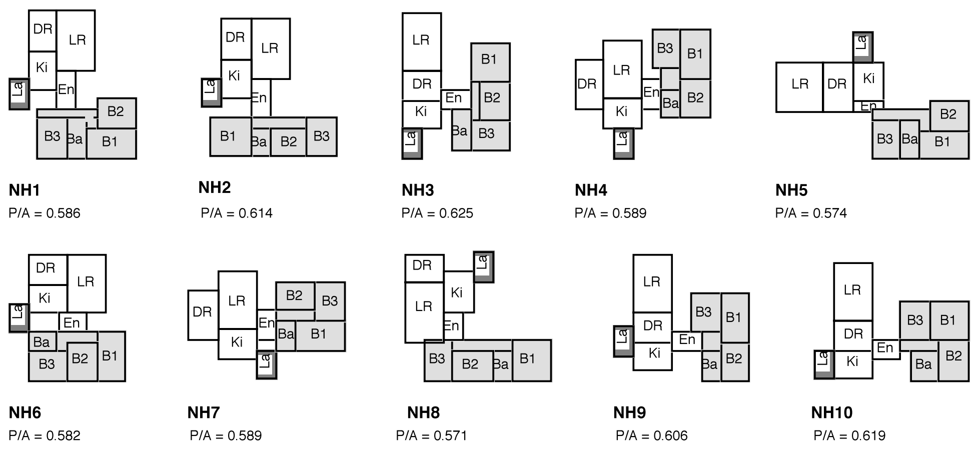


Fig. 10. Ten houses resulting from Test 2—26 cases.

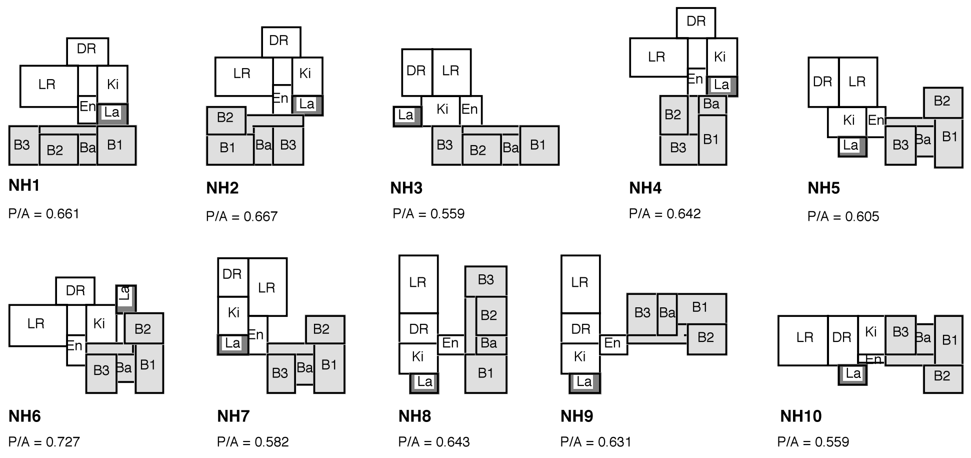


Fig. 11. Ten houses resulting from Test 3—40 cases.

7.5. Analysis of results

The two tests using the 26 very compact houses, that is, Tests 1 and 2 obtained fairly similar results. Although the maximum P/A result of 0.642 for Test 1 (solution 1-NH9) was higher than the maximum of 0.625 for Test 3 (solution 2-NH3), the average P/A result for Test 2 of 0.596 was higher than the average of 0.584 of Test 1. A similar result is obtained when comparing Tests 3 and 4. The maximum P/A of 0.727 for Test 3 (solution 3-NH6) is higher than the maximum of 0.684 for Test 4 (solutions 4-NH7 and 4-NH9), whereas the average of 0.640 for Test 4 was higher than the average of 0.628 for Test 3. This shows that allowing the zones to be randomly joined does not necessarily result in obtaining a solution that is better than using the joinings given in the case base but does result in getting a better set of solutions from which a final choice can be made. Since the aim in design is not necessarily to find a solution with the "best" based on a narrow definition of "best" given by some fitness function but may be based also on other factors not expressed in the fitness function, getting a range of potentially good solutions is important.

On the other hand, there is a marked difference between the results of the tests using 26 cases (Tests 1 and 2) and the results of the test using 40 cases (Test 3 and 4). It is quite clear that expanding the case base to 40 cases improves both the maximum P/A and average P/A results and that the number of generations required to obtain the results drops. This shows that the efficiency of the case-based approach depends strongly on the information available in the case base. If the genotypic information is poor with respect to its capability of producing fit solutions, then unless mutation can change this, the results will be impoverished. When there exists sufficient genotypic information then the case-based approach is very efficient.

There is not a great difference in the quality of results obtained from the case-based approach using 40 cases and the results obtained by generating the houses anew. How-

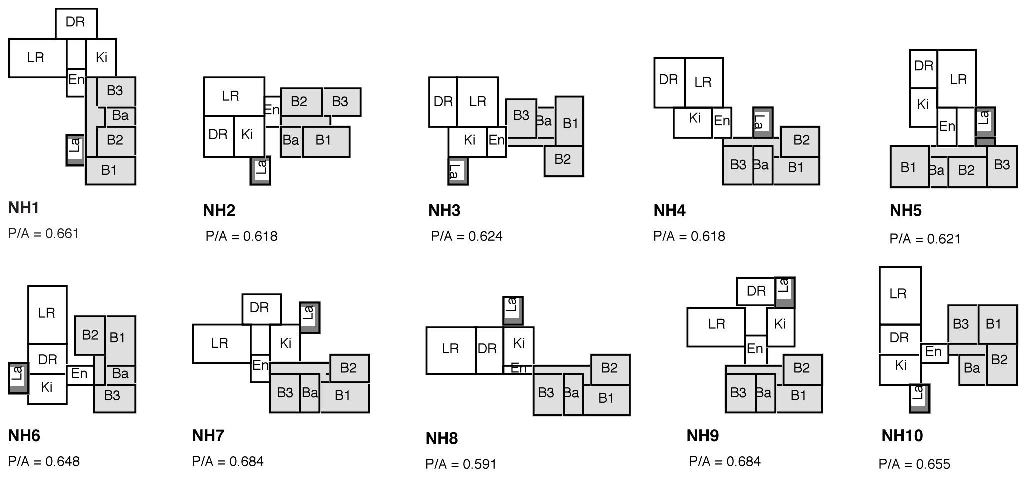


Fig. 12. Ten houses resulting from Test 4—40 cases.

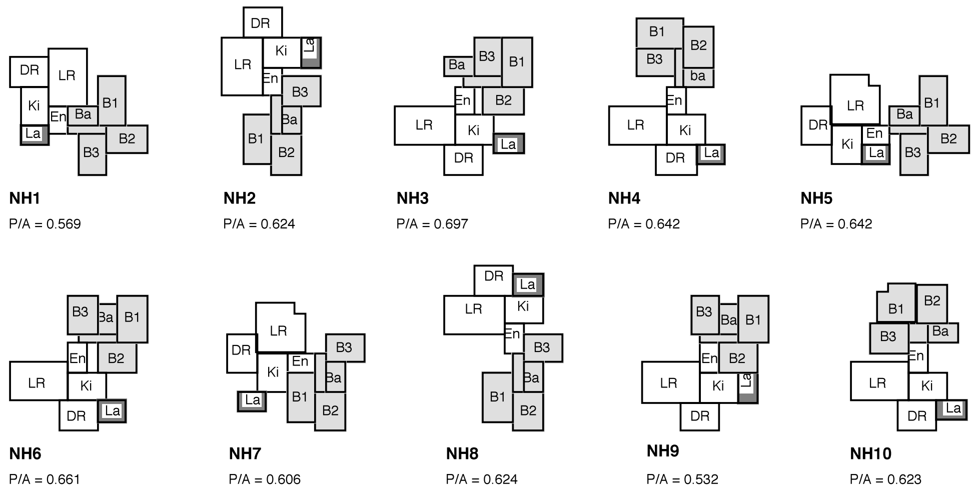


Fig. 13. Ten houses resulting from Test 5—growth process.

ever, the effort expended by the case-based approach is substantially less. For Test 2, a total of 70 generations were run whereas for Test 5 a total of 857 generations were run (counting the generations of rooms, zones, and houses). This shows that the information contained in the case base regarding the rooms, composition of zones, and the joining of zones is useful.

A feature which appeared in the generation of houses anew (Test 5) and when allowing random joining of zones (Tests 2 and 4) was that the initial random population generated was very uneven. Only very few (1 in most cases) solutions had values which were close to being satisfactory, the rest had very low values because of the poor circulation efficiency values resulting from poor joinings of zones. So that while one or two solutions may have a score of approximately 40 ($P/A \times$ Penalty) many solutions had score of less than 10. This meant that one or two solutions clearly dominated the rest of the population and the run quickly converged to those solutions. In some cases, where all the initially generated solutions were very low, the run converged without any satisfactory solutions found. This explains why the number of generations with Tests 2 and 4 is fairly high. In comparison, the initial generation for Tests 1 and 3 was the case base with a fairly even spread of solutions.

Table 1. *Statistics*

	No. of Cases	Runs	Total No. of Generations	Max. No. of Generations in a Run	Best P/A	Average P/A
Test 1	26	15	217	30	0.642	0.584
Test 2	26	19	162	27	0.625	0.596
Test 3	40	7	70	13	0.727	0.628
Test 4	40	21	170	17	0.684	0.640
Test 5	—	63	857	60	0.697	0.622

The implementations using the case-based approach did not find the solution shown in Figure 14 which has a P/A of 0.818, although solution 3-NH6 ($P/A = 0.727$) could possibly be used to suggest it.

An observation made was that while the fitness function evaluates the overall P/A of the house there is no assessment made of the potential of zones to produce the required results. This is left to the general methodology of the evolutionary method to sort out. An example is given in Figure 15. The two squares (3×3 units) have P/A ratios of 1.33, so that a selection based on the fitness of the squares would not be able to prefer one over the other. However, the shapes A and B composing the left-hand square can be composed to form the shape shown with a P/A of 2.22 as against the composition of C and D, shown on the right-hand side, with a P/A of 1.78. This shows that preselecting based on the performance of given solutions could discard solutions containing elements which have the potential to contribute to good solutions.

8. SUMMARY

The evolutionary case-based design process promises a general, domain-independent, method for case-based design with solutions to the problems of case-base adaptation, case representation, and indexing. The experimentation has shown that even at the highest building block level, it is possible to achieve good results. In addition, not only is it unnecessary to preselect "good" cases or classify the cases into "good" and "bad" solutions but this may be detrimental. This contradicts the conclusions reached by Louis and Johnson (1999) and Maher and Gomez de Silva Garza (1996). That is, all the information that is available in the design experience, represented by the case base, should be used since it is not possible to determine what components of any cases may or may not be useful by simply evaluating the performance of

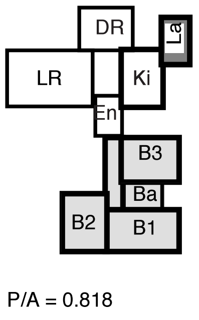


Fig. 14. "Best" solution potentially available from case base.

the overall design. However, what is critical is the information contained in the case base. This is not specific to the evolutionary approach but to case-based design in general. A case base must contain sufficient information with the potential to generate suitable solutions from its cases to be effectual.

9. FUTURE WORK

Future work is needed to resolve several issues, such as:

- How large and how varied does the case-base need be to be useful? If the variability of the case base is limited, can new components be created to overcome this and how will this then be different from just generating solutions without recourse to a case base?
- The level at which the genetic operations, for example, crossover, have to be applied in a hierarchical representation. Can they be restricted to the highest level of description for the house, that is, using zones as building blocks or will it be necessary to descend to lower levels, for example, to create new zones or even new rooms? This is allowing cut sites at different levels in the tree structure. Since there is a strict component/assembly hierarchy, where an assembly (symbol) replaces a combination of components, any new combination of components creates a new assembly (symbol). How is this accomplished in a case-based environment, how is it recognized, and how is it incorporated into the process? Unlike the general genetic programming process, the crossover in the tree representation has to be controlled between similar parts, or subtrees, of the tree to maintain consistency of structure.
- Related to the previous point, what information can be learned from a case base and reapplied and what new information needs be created? Should only the component information be used or both the component and configuration (joinings) of component information be used? For example, it may be that only the zone (and room) solutions can be reused, without the need for new solutions, whereas the rotation and edge joinings have to be generated anew.
- Since the efficiency of a hierarchical case base is related to the reuse of high-level components, if new lower level components are to be generated, a strategy is required to bias the process towards reuse rather than regeneration.

Further, the use of evolutionary case-based design brings into focus the interpretation of the genotype into phenotype descriptions depending on the particular problem. Since in the test example the fitness function was associated with perimeter values, an interpretation based on edge vectors was suitable. However, other problems may require other phenotypic interpretations. This addresses the issue that although the process for creating the solution is given by the genotype description, its description is not fixed *a priori* but determined depending on the problem.

ACKNOWLEDGMENTS

This work is supported by Australian Research Council Large Grants, A89700539.

REFERENCES

Bentley, P.J. (Ed.) (1999). *Evolutionary Design by Computers*. Morgan Kaufmann, San Francisco, California.
Bentley, P.J. & Wakefield, J.P. (1995). The table: An illustration of evolutionary design using genetic algorithms. *Genetic Algorithms in Engineering Systems: Innovations and Applications*, GALESIA '95, pp. 412–418.
Bentley, P.J. & Wakefield, J.P. (1997). Generic evolutionary design. In *Soft Computing in Engineering Design and Manufacturing*, (Chawdry, P.K., Roy, R., & Pant, R.K., Eds.), pp. 289–298. Springer Verlag, London.
Calvin, W.H. (1987). The brain as a Darwin machine. *Nature 330*, 33–34.
Calvin, W.H. (1996). *How Brains Think: Evolving Intelligence, Then and Now*. Basic Books, New York.

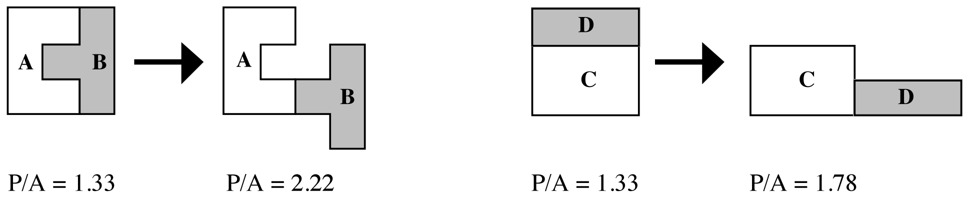


Fig. 15. Potential of different configurations.

Calvin, W.H. (1998). Competing for consciousness. *Journal of Consciousness Studies 5(4)*, 388–404.

Dasgupta, D. & Michalewicz, Z. (Eds.). (1997). *Evolutionary Algorithms in Engineering Applications*. Springer-Verlag, Southhampton and Berlin.

Flemming, U. (1994). Case-based design in the SEED system. In *Knowledge-Based Computer-Aided Architectural Design*, (Carrara, G. & Kalay, Y., Eds.), pp. 69–91. Elsevier, Amsterdam, Holland.

Gero, J.S., Kazakov, V., & Schnier, T. (1997). Genetic engineering and design problems. In *Evolutionary Algorithms in Engineering Applications*, (Dasgupta, D. & Michalewicz, Z., Eds.), pp. 47–68. Springer Verlag, Southhampton and Berlin.

Gero, J.S. & Schnier, T. (1995). Evolving representations of design cases and their use in creative design. In *Preprints Computational Models of Creative Design*, (Gero, J.S., Maher, M.L., & Sudweeks, F., Eds.), pp. 343–368. Key Centre of Design Computing, University of Sydney, Australia.

Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, Massachusetts.

Gomez de Silva Garza, A. & Maher, M.L. (1999). An evolutionary approach to case adaptation, *Case-Based Reasoning Research and Applications. Proc. Third Int. Conf. on Case-Based Reasoning, ICCBR-99*, Monastery Seeon, Munich, Germany.

Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, Michigan.

Koza, J. (1992). *Genetic Programming: On Programming Computers by Means of Natural Selection*. MIT Press, Cambridge, Massachusetts.

Koza, J. (1994). *Genetic Programming II: Automatic Discovery of Reusable Programs*. MIT Press, Cambridge, Massachusetts.

Koza, J., Bennett, III, F.H., Andre, D., & Keane, M.A. (1996). Automated designs of both the topology and sizing of analog electrical circuits using genetic programming. In *Artificial Intelligence '96*, (Gero, J.S. & Sudweeks, F., Eds.), pp. 151–170. Kluwer Academic, Dordrecht, Germany.

Louis, S. (1997). Working from blueprints: Evolutionary learning for design. *Artificial Intelligence in Engineering 11*, 335–341.

Louis, S. & Johnson, J. (1999). Robustness of case-initialized genetic algorithms. *Proceedings of FLAIRS-99*, FLAIRS, pp. 129–133.

Louis, S.J. & Rawlins, G.J. (1991). Designer genetic algorithms: Genetic algorithms in structure design. In *Proc. Fourth Int. Conf. on Genetic Algorithms*, (Belew, R.K. & Booker, L.B., Eds.), pp. 53–60. Morgan Kaufmann, San Mateo, California.

Maher, M.L., Balachandran, M., & Zhang, D.M. (1995). *Case-Based Reasoning in Design*. Lawrence Erlbaum, Hillsdale, New Jersey.

Maher, M.L. & Gomez de Silva Garza, A. (1996). The adaptation of structural system design using genetic algorithms. *Proc. Int. Conf. on Information Technology in Civil and Structural Engineering—Taking Stock and Future Directions*. Glasgow, Scotland.

Maher, M.L. & Pu, P. (Eds.). (1997). *Issues and Applications of Case-Based Reasoning in Design*. Lawrence Erlbaum, Hillsdale, New Jersey.

Michalewicz, Z., Dasgupta, D., Le Riche, R.G., & Schoenauer, M. (1996). Evolutionary algorithms for constrained engineering problems. *Computers and Industrial Engineering Journal* (special issue on genetic algorithms and industrial engineering) *30(4)*, 851–870.

Oxman, R. & Oxman, R. (1994). Case-based design: Cognitive models for case libraries. In *Knowledge-Based Computer-Aided Architectural Design*, (Carrara, G. & Kalay, Y., Eds.), pp. 45–68. Elsevier, Amsterdam, Holland.

Parmee, I.C. (1997). Strategies for the integration of evolutionary/adaptive search with the engineering design process. In *Evolutionary Algorithms in Engineering Applications*, (Dasgupta, D. & Michalewicz, Z., Eds.), pp. 453–478. Springer-Verlag, Southhampton and Berlin.

Parmee, I.C. & Beck, M.A. (1997). An evolutionary, agent-assisted strategy for conceptual design space decomposition. *Proceedings of AISB Workshop on Evolutionary Computing, Springer Lecture Notes in Computer Science, No. 1305*, Springer-Verlag, pp. 275–286.

Riesbeck, C.K. & Schank, R.C. (1989). *Inside Case-Based Reasoning*. Lawrence Erlbaum, Hillsdale, New Jersey.

Rosenman, M.A. (1995). An edge vector representation for the construction of two-dimensional shapes. *Environment and Planning B: Planning and Design 22*, 191–212.

Rosenman, M.A. (1996*a*). A growth model for form generation using a hierarchical evolutionary approach. *Microcomputers in Civil Engineering* (special issue on evolutionary systems in design) *11*, 161–172.

Rosenman, M.A. (1996*b*). The generation of form using an evolutionary approach. In *Artificial Intelligence '96*, (Gero, J.S. & Sudweeks, F., Eds.), pp. 643–662. Kluwer Academic, Dordrecht, Germany.

Rosenman, M.A. (1997*a*). The generation of form using an evolutionary approach. In *Evolutionary Algorithms in Engineering Applications*, (Dasgupta, D. & Michalewicz, Z., Eds.), pp. 69–85. Springer-Verlag, Southhampton and Berlin.

Rosenman, M.A. (1997*b*). An exploration into evolutionary models for non-routine design. *Artificial Intelligence in Engineering 11*, 287–293.

Rosenman, M.A. (1999). A face vector representation for the construction of polyhedra, *Environment and Planning B: Planning and Design 26*, 265–280.

Rosenman, M.A. & Gero, J.S. (1999). Evolving designs by generating useful complex gene structures. In *Evolutionary Design by Computers*, (Bentley, P.J., Ed.), Morgan Kaufmann, San Francisco, California.

Rosenman, M.A., Gero, J.S., & Oxman, R.E. (1992). What's in a case: The use of case bases, knowledge bases and databases in design. In *CAAD Futures '91*, (Schmitt, G.N., Ed.), pp. 285–300. Viewig, Wiesbaden, Germany.

Schank, R.C. (1982). *Dynamic Memory: A Theory of Reminding and Learning in Computers and People*. Cambridge University Press, Cambridge, UK.

Schwefel, H-P. (1995). *Evolution and Optimum Seeking*. John Wiley and Sons, New York.

Simon, H.A. (1969). *The Sciences of the Artificial*. MIT Press, Cambridge, Massachusetts.

Williams, P. (1995). *Dynamic memory for design*. *PhD Thesis*, (unpublished), Department of Architectural and Design Science, University of Sydney, Sydney, Australia.

Woodbury, R.F. (1993). A genetic approach to creative design. In *Modeling Creativity and Knowledge-Based Creative Design*, (Gero, J.S. & Maher, M.L., Eds.), pp. 211–232. Lawrence Erlbaum, Hillsdale, New Jersey.

Michael Rosenman is a Senior Research Fellow at the Key Centre of Design Computing and Cognition, Department of Architectural and Design Science, University of Sydney. He has been active in the field of design optimization and knowledge-based design for over two decades and in the field of evolutionary design for the last five years. His research in evolutionary design is directed towards complex hierarchical genotypic representations. He is the author of over 70 publications including co-author of the book *Knowledge-Based Design Systems* and has made presentations at numerous international conferences.