Using Evolutionary Methods for Design Case Adaptation

Andres Gomez de Silva Garza
Instituto Tecnológico Autónomo de México, Mexico

Mary Lou Maher
University of Sydney, Australia

Abstract
Case-based reasoning (CBR) provides a methodology for directly using previous designs in the development of a new design. An aspect of CBR that is not well developed for designing is the combination and adaptation of previous designs. The difficulty with this aspect of case-based design is partly due to the extensive amounts of specialised knowledge needed to select the appropriate features of a previous design to include in the new design and the adaptation of these features to fit the context of a new design problem. In this paper we present a design process model that combines ideas from CBR and genetic algorithms (GA's). The CBR paradigm provides a method for the overall process of case selection and adaptation. The GA paradigm provides a method for adapting design cases by combining and mutating their features until a set of new design requirements and constraints are satisfied. We have implemented the process model and illustrate the model for residential floor plan layout. We use a set of Frank Lloyd Wright prairie house layouts as the case base. The constraints used to determine whether new designs proposed by the process model are acceptable are taken from *feng shui*, the Chinese art of placement. This illustration not only clarifies how our process model for design through the evolutionary adaptation of cases works, but it also shows how knowledge sources with distinct origins can be used within the same design framework.

Keywords
Evolutionary Design, Case-Based Reasoning, Floor Plan Layout

1 Introduction

Case-based reasoning provides a computational model of analogical reasoning that relies on a set of previous solutions. The literature on case-based reasoning (for example, Kolodner, 1993 and Leake, 1996) highlights the reasoning process involved in making analogies but generally does not have a generic way of treating case adaptation. For designing, case-based reasoning (CBR) provides a methodology for directly using previous designs and has potential for addressing the combination and adaptation of previous designs (Maher, Balachandran, and Zhang, 1995). .Some case studies explaining how CBR has been applied to design problems can be found in (Maher and Pu 1997). The difficulty with the adaptation of previous designs is partly due to the extensive amounts of specialised knowledge needed to select the appropriate features of a previous design to include in the new design and the adaptation of these features to fit the context of a new design problem (Leake 1996). Our model overcomes this difficulty by using a genetic algorithm (see Mitchell 1998) for the task of case adaptation.

A genetic algorithm requires knowledge for evaluating the new solutions (in this case, new designs) it produces, but it doesn't require any knowledge in order to select and adapt features; this selection and generation of adaptations is done at random. Many adaptations are produced in parallel, some of them ultimately determined to be worthless. After a cycle of random generation of design adaptations, the new designs produced are evaluated. If one or more designs of sufficient quality are found to be present in the population of designs, the process can end. If not, the best of the designs can be kept, the rest discarded, and a new cycle of adaptations can begin. Thus an evolutionary design case adaptation algorithm monotonically increases the quality of the designs in its population in a cyclical fashion, and ends when it has found a sufficiently good adaptation of the original designs.

In this paper we present a design process model called GENCAD (GENetic Case ADaptation) that combines CBR and genetic algorithms (GA's). The CBR paradigm provides a method for the overall process of case selection and adaptation. The GA paradigm provides a method for adapting design cases by combining and mutating their features until a set of new design requirements and constraints are satisfied. The requirements of a new design problem are specified at the outset of the process. Design constraints provide additional knowledge for evaluating the quality of the solutions proposed through the adaptation of previously known designs by the process.

We have implemented the GENCAD process model computationally. The type of design problem used to illustrate the model is residential floor plan layout. We use a set of Frank Lloyd Wright prairie house layouts as the case base. The constraints used to determine whether new designs proposed by the CBR process are acceptable are taken from *feng shui*, the Chinese art of placement. This illustration not only clarifies how our process model for design through the evolutionary adaptation of cases works, but it also shows how knowledge sources with distinct origins can be used within the same design framework. We do not suggest that residential floor plan layout should use a combination of Frank Lloyd Wright prairie houses evaluated by feng shui constraints. The illustration was selected to make the point that the sources of knowledge for the precedents and the evaluation can be completely different in this computational model, and that the evaluation knowledge need not be numerical optimisation.

In section 2 of the paper we present GENCAD, our process model for the evolutionary adaptation of design cases. In section 3 we discuss the representation issues in residential layout design and present a representation that can be evaluated for *feng shui* constraints. In section 4 we present our implementation of the GENCAD process model and give the results of an hypothetical design problem. Section 5 closes the paper with a summary and discussion of the features of this process model.

2 GENCAD: A Process Model for the Evolutionary Adaptation of Design Cases

GENCAD's process model emphasises solving problems using information derived from precedents. The precedents that are retrieved from a

case memory serve as starting points for proposing solutions to new design problems. Multiple random combinations and modifications (together referred to as adaptations) of the retrieved cases are then generated and evolved incrementally, until a satisfactory solution to the new design problem is found. Figure 1 shows GENCAD's process model for the evolutionary adaptation of design cases.

There has been other work on case-based architectural design that has focused on case adaptation. The SEED project (Flemming et al. 1997; Fenves, Rivard, and Gomez, 2000) views case adaptation as the addition, removal, modification, or manual editing of functional units of a design case. CADRE (Faltings 1997) uses case-specific constraints to perform case adaptation. CADSYN (Zhang 1994) uses general domain constraints for the case adaptation subtask. In FABEL (Vob et al. 1997) a multitude of different case adaptation methods is provided for use in different scenarios. Rosenman (2000) shows how a genetic algorithm can be used to adapt hierarchical representations of floor plans.

2.1 Overview of the Process Model

Given the specification of the requirements of a new design problem, the first task in the GENCAD process model is to determine which precedents contain information that might be useful in solving the new problem. This is done by consulting a case base and comparing the description of the new problem with the descriptions of the precedents stored in the case base. Those cases for which any similarity is found with the new problem are retrieved from the case base. They are taken to be first approximations towards a solution to the new problem, and are put together into a population of potential designs.

The task of case adaptation is performed by an evolutionary method in GENCAD. Two types of adaptation are provided in the evolutionary method: combination and modification. These types of adaptation are performed on the designs in the population through the genetic operators of crossover and mutation, respectively. Crossover produces two offspring designs, each of which combines features from each of two parent designs. Mutation produces one offspring design which is an altered version of one parent design. Both crossover and mutation insert new designs into the population.

The two types of adaptation can result in generating offspring designs that are better (i.e., closer to being a solution to the new design problem)

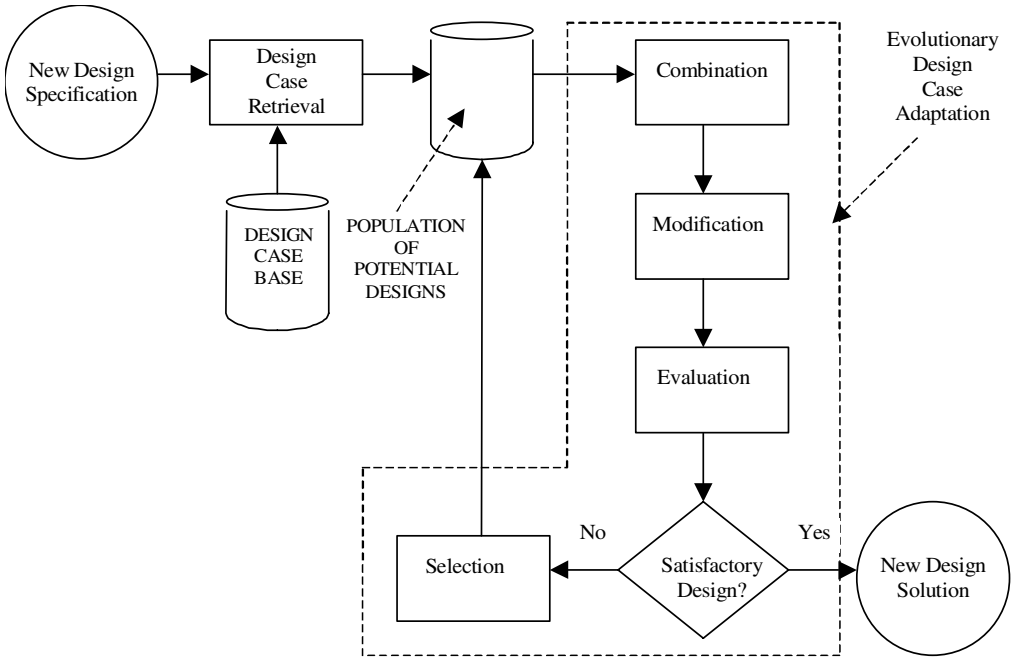

Figure 1. A process model for case-based design with evolutionary case adaptation.

than some or all of the old parent designs, or worse. In order to determine this, the potential solutions have to be evaluated and their relative worth compared. In the process of evaluating the solutions, one (or more) might be found that are good enough to satisfy the requirements of the new design problem. If this happens, the process model can stop at this point; if not, the evolutionary adaptation process continues, using the best of the designs as the initial population for the next cycle of adaptations. The best designs are selected from the augmented population containing both old (relative to the current evolutionary cycle) and newly generated potential designs.

Evaluation and selection are not directly responsible for manipulating old designs to produce new ones, unlike combination and modification. However, they are considered part of the evolutionary case adaptation process because the results they give define the paths examined during the search for a solution to the new problem. Evaluation and selection also define which designs will be available for adaptation in future cycles; thus, they guide and influence case adaptation.

2.2 Contents of a Case

An important consideration when working on a case-based reasoning project is to decide on the types of information that will be held in the cases in memory. The case representation depends on the CBR system's task, application domain, and the role that cases play in the system. In general, every case contains a description of the solution to a previously encountered problem and in some projects the cases in addition might store a description of:

- the problem that was solved by the solution,
- the problem-solving steps that were taken to reach the solution,
- annotations, explanations or justifications of aspects of the solution, and/or
- annotations, explanations or justifications of the problem-solving steps that were taken in generating the solution.

In addition, if CBR is being used for design, the description of the solution to a previously encountered problem can include not just a description of the designed artefact, but also some contextual or support information. For instance, the description of the solution might also include a description of the environment in which the designed artefact is meant to operate in or other related information that might be useful when reusing the design and that might influence some design decisions.

Not all such knowledge that can be stored together with a case might be needed in the case adaptation process. For instance, it could be that only the description of the solution itself, without any added contextual information, is required during case adaptation (while the supporting information may have been useful for performing case retrieval or for communicating the contents of the cases to the user in different ways).

Implicit in GENCAD's process model, therefore, is the possibility that some preparation will be required of the cases retrieved from memory before they can become the initial population of the evolutionary case adaptation method. Figure 2 shows this expanded view of the case retrieval task from the process model.

3 Residential Layout Design Subject to *Feng Shui* Constraints

The representation of a floor plan layout can take many forms. The representation in a CAD system is based on the way in which the floor plan is drawn and is limited to geometric and possibly materials information of the components of a floor plan. The representation for any computational system will depend on the information manipulated by the system as well as the needs for visualisation of the results. In our example we are evaluating the designs according to *feng shui*. This influences our representation of the floor plan towards information about the layout of the landscape as well as the interior of the residence. If we were to use the GENCAD model to evolve floor

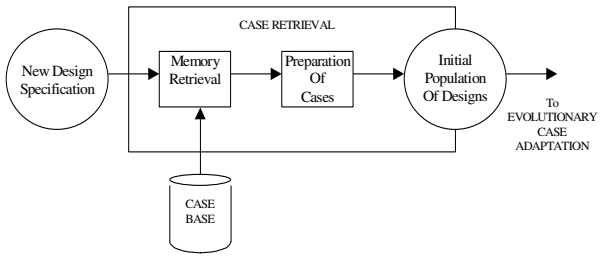

Figure 2. An expanded view of case retrieval.

plans to comply with a particular building code, the representation would be customised for the information needed to check compliance.

In this section we discuss the representation issues for a computational model of the design of residential layouts that satisfy the principles of *feng shui*. The use of *feng shui* constraints highlights the characteristics of a model in which the constraints are not primarily numerical or geometric.

3.1 Characteristics of *feng shui*

Feng shui, also known as Chinese geomancy, is an ancient Chinese technique that, among other things, determines the quality of proposed or existing layouts of residences according to several rules of thumb. Some of these heuristics seem to have a basis in common sense, or in a psychological or sociological appreciation of the human beings that inhabit (or intend to inhabit) the residence. Other heuristics seem to be of a more superstitious nature. Of prime importance to performing a *feng shui* analysis according to the principles of any of the existing sects is information on the relative positions of objects. In addition to their positions, other attributes of objects are usually also taken into account, such as their orientations, shapes, and relative sizes. In this paper we have used the knowledge of *feng shui* presented in (Rossbach 1987), which corresponds to the Tibetan black-hat sect of *feng shui*.

The *feng shui* knowledge contained in (Rossbach 1987) applies to three different levels of description of a residence:

- the landscape level (the location of a residence with respect to other objects in its environment such as mountains, rivers, roads, etc.),
- the house level (the relative placement of the rooms and functional spaces within a residence, such as bedrooms and bathrooms, as well as the connections between them, such as doors and windows), and
- the room level (the location of furniture, decorations, and other objects within each room or functional space in a residence).

Feng shui analysis assumes knowledge of spatial relationships among the objects at each of the different levels. Absolute locations and exact measures of distances and other geometric quantities are not as important. Because of this, a qualitative spatial representation of location seems to be sufficient.

3.2 Spatial representation of floor plans

The representational framework we use to describe the locations of objects within each of the three levels is a low-resolution, qualitative one. We locate objects (which we will refer to as elements) at each level on a 3x3 spatial grid, with each sector within the grid assigned a unique number between 1 and 9 to identify it. The grid is shown in Figure 3, with north assumed to be at the top of the grid.

Elements can occupy more than one grid sector, and grid sectors can contain more than one element, making the representation flexible. The resolution of this representation is not high, but considering the qualitative nature of a typical *feng shui* analysis and the number of elements that typically need to be represented at each of the three levels, it is adequate in most cases. It allows us to determine the relative positions of objects, which is the most important capability needed for an adequate *feng shui* analysis.

Figure 3 shows a typical residence viewed at the three levels of description that a *feng shui* analysis would look at, with the 3x3 spatial grid superimposed on each level. The figure is immediately followed by an equivalent symbolic representation of the residence shown in it.

```
(((level landscape)
   (elements (((type hill) (name dragon-
mountain)
              (location (1 2 4)) (steep-
ness high) ...)
             ((type house) (name my-house)
(location (5)))
```

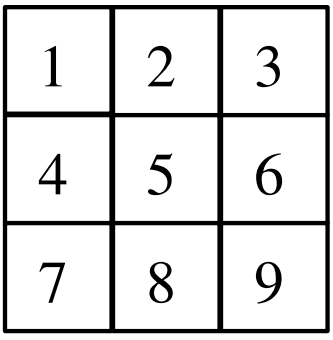

Figure 3. Qualitative spatial grid for elements.

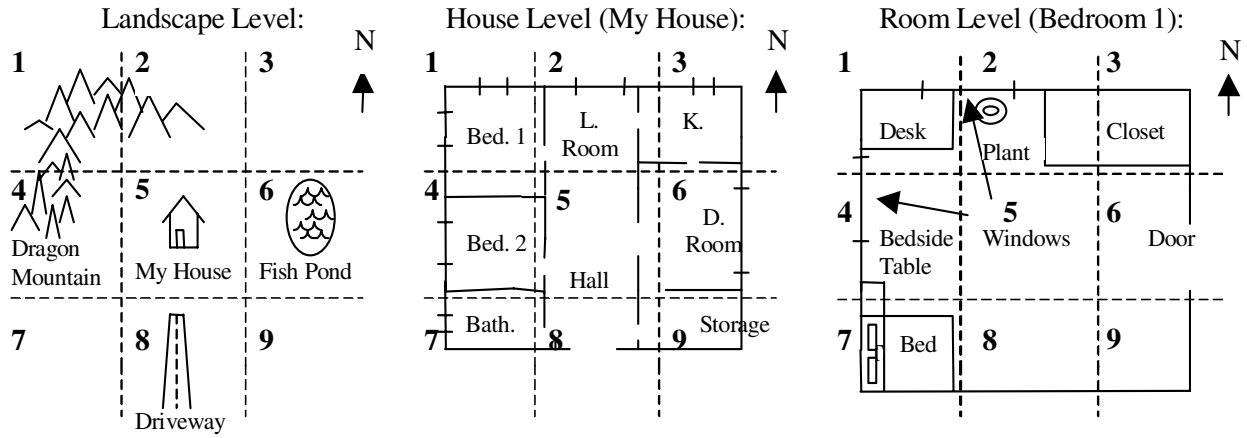

Figure 4. Example of a residence viewed at the three levels.

```
            ...)))
 ((level house)
  (elements (((type bedroom) (name bed-
room-1) (location (1))
             (shape square))
           ((type bedroom) (name bedroom-
2) (location (4)))
            ((type hallway) (name hall-
1) (location (5 8)))
            ...))
  (connectors (((type internal-door) (name
b2-hall) (location (5))
               (side-a bedroom-2) (side-
b hall-1)
               (direction ew))
                ((type window) (name b1-
window-1) (location (1))
               (side-a bedroom-1) (side-
b outside)
               (direction ew))
              ...)))
 ((level room)
  (name bed-1)
  (elements (((type bed) (name b-1) (loca-
tion (7)))
             ((type desk) (name d-1) (lo-
cation (1)))
           ((type window) (name w-1) (lo-
cation (1 2)))
            ...))))
```

4 Implementation and Experiment

We have implemented the GENCAD model for structural system design and floor plan layout. In this paper we present only the floor plan layout implementation and refer to this as GENCAD-FS. Further, we only consider the house level of description of residences, since it offers more implementation challenges and representational intricacies than the landscape or room levels of description.

4.1 Representing design problems in GENCAD-FS

For our example, the specification of a new layout problem is the types of rooms required. The design task is to decide how to place and connect the specified spaces in a configuration that is acceptable according to a set of constraints, in our illustration, *feng shui*. We represent the new design problem as a list of structural element types associated with a quantity, where this quantity represents the number of structural elements of the given type that are required. For instance:

```
((bedroom 3) (bathroom 2) (music-
                room 1))
```

4.2 Design cases and case memory in GENCAD-FS

GENCAD-FS's case library currently contains 12 cases, each of which describes one of Frank Lloyd Wright's prairie houses. These cases were obtained from (Hildebrand 1991). The cases that have been implemented are: Hickox House, Bradley House, Willits House (lower floor), Heurtley House (lower floor), Heurtley House (upper floor), Cheney House, Avery Coonley House, Robie House (main floor), Robie House (upper

floor), Hardy House (lower floor), Glasner House, and Roberts House.

A common theme in Wright's prairie houses is that they are usually designed to occupy two or even three stories, so many floor plans include staircases. Our case library represents a wide variety of designs: some are second- or third-story floors, with many windows and terraces overlooking the landscape; others are ground-level (or semi-subterranean) floors, which in Frank Lloyd Wright's designs often do not have many windows or are hidden from view from observers outside the house. Each floor of a multi-storied residence is represented as a separate case. Therefore some cases do not have a kitchen, or lack some other feature that would normally be present in a complete residence. This variety does not diminish GENCAD-FS's usefulness; in fact, it increases its capacity for creativity.

4.3 Representing genotypes in GENCAD-FS

In a representation of residential layout the important features are not just the elements that exist (such as bathrooms and bedrooms) and their properties, but also the connections between elements (such as doors and windows) and from elements to the outside. Some of the features of the connectors, such as their orientation and type, are also important. A suitable genotype representation would have to include not just information about the existence and descriptions of elements, but also about their interconnections.

In order to capture this kind of information in GENCAD-FS we use a matrix representation, based on the concept of adjacency matrices (e.g., see (Chachra et al. 1979)). A different type of matrix genotype, dealing with design elements at a much finer level of detail, has been used in (Kane and Schoenauer 1996). Kane and Schoenauer also introduce different ways of performing crossover on such matrix genotypes. In this paper we will discuss only one possible matrix-based genotype representation that is useful for our application, and one way of performing crossover on genotypes that are matrices.

Adjacency matrices represent the connectivity between different nodes in a graph. A phenotype at the house level can be visualised using a graph representation in which the nodes can be the elements (rooms and functional spaces) and the arcs can be the connectors between elements (connections between rooms). Each node and each arc carries information associated with it describing the features of the corresponding element or connector. The graph representation of the phenotype is converted into its equivalent adjacency matrix genotype before performing crossover and mutation.

A basic adjacency matrix consists of K rows and K columns, where K is the number of nodes. The rows and columns of a given adjacency matrix are labeled in the same order, i.e., row I and column I both describe the same node. The item stored at location [I, J] in the matrix (where I<>J) is a number indicating how many arcs there are between node I and node J. This means that adjacency matrices are symmetric across their diagonal and have zeroes as the elements of their diagonals.

A matrix-based genotype expands on the concept of the adjacency matrix by:

- Including name vectors as part of the genotype. These name vectors label the rows and columns of the matrix. The name vectors contain the names of the rooms corresponding to each node. The names in the name vectors are used as pointers to the other attribute-value pairs describing the corresponding room when such additional information about a node has to be accessed.
- Including an extra node in the graph (and therefore an extra row/column in the matrix) to represent the outside (any space not considered to belong to the house).
- Including a list of which connectors connect node I and J at location [I, J] in the matrix (which indirectly gives us information about how many connectors there are), rather than just a number indicating how many connectors there are. The connectors named in the list point to the attribute-value pairs describing the connectors in more detail. Because a room or functional space cannot be connected to itself, each item on the diagonal of a matrix-based genotype will be the empty list.

Figure 5 shows a simple phenotype viewed as a floor plan (without superimposing the 3x3 grid on the plan, for simplicity), with its equivalent matrix-based genotype next to it:

Within a matrix genotype, the order in which genes appear does not matter (for example, B could have been placed in the second position

instead of the first in Figure 6, and the genotype would have represented the same house layout), as long as the same order is followed in the rows and in the columns of the matrix. Phenotype-to-genotype and genotype-to-phenotype transformations at the house level involve converting from the symbolic representation to an equivalent matrix-based one, and *vice versa*.

4.4 Crossover in GENCAD-FS

It is important to combine aspects from both the elements and the connectors of a residence so that each offspring genotype inherits characteristics of both the elements and the connectors of both parents. Thus, crossover should operate not just on the adjacency matrix, but also on the name vectors describing the elements that are used to label the rows and columns of the adjacency matrix. Crossover in GENCAD-FS implies the exchange of sub-matrices of the same dimensions. The diagonal of each sub-matrix has to correspond to part of the diagonal of the matrix genotype it is extracted from to ensure that the matrices resulting from crossover maintain their diagonal symmetry.

Crossover has been implemented in GENCAD-FS so that first the size of the sub-matrices that will be exchanged is randomly chosen (taking into account that their maximum possible size is the size of the smallest of the two parent matrices). Then, separately for each sub-matrix to be extracted, the index of its upper-left hand item is chosen at random relative to the parent matrix it is to be extracted from (taking into account the size of the sub-matrix, which places limits on the possible indices). Finally, the sub-matrix extraction and exchange is performed. In parallel with this operation, the same type of extract and exchange is performed on subparts of the name vectors associated to the parent matrices using the corresponding indices chosen at random for the sub-matrices.

Figure 6 illustrates what a typical crossover operation on a matrix genotype looks like in GENCAD-FS (including the vectors labeling the rows and columns of the matrices, which are considered to be part of the genotypes). The fragments of the genotypes that are cut and exchanged during crossover are shown enclosed in thick lines (which represent the crossover "points"). One of the parent genotypes is shown plain, while the other is darkly shaded. The figure shows how parts from both parent genotypes combine to create the offspring genotypes after crossover is performed at the indicated crossover points. The lightly shaded areas indicate parts of the offspring genotypes that were not exchanged during crossover, but that may have to be revised if the introduction of new adjacencies can lead to conflicts in some of the features encoded in the genotypes.

The offspring resulting from such a crossover operation might introduce inconsistencies in the

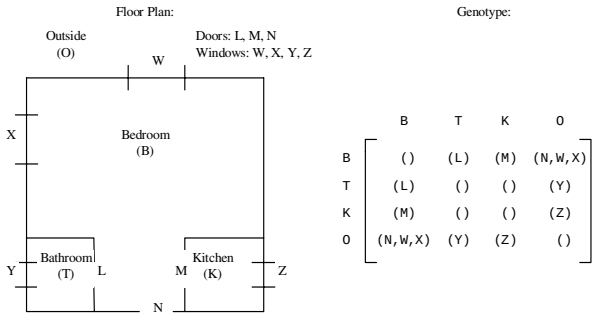

Figure 5. Graphical view of a phenotype and its equivalent matrix genotype.

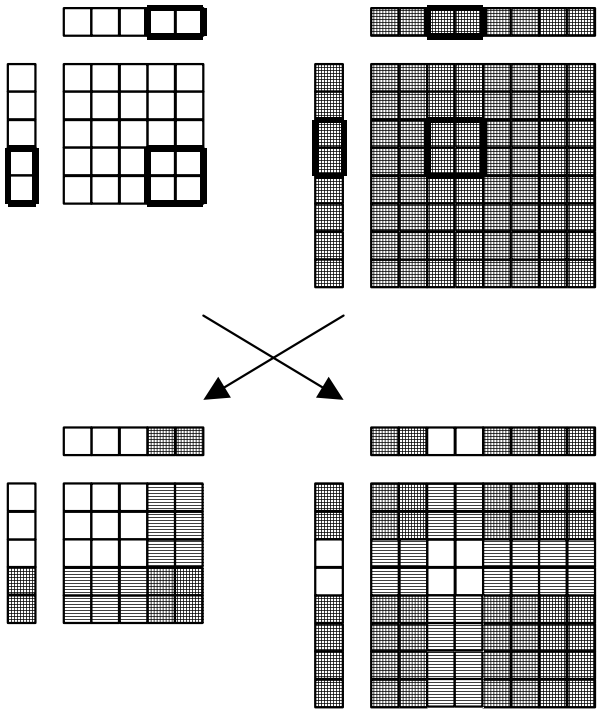

Figure 6. A typical crossover operation performed on two matrix genotypes.

adjacencies to the rooms corresponding to some rows/columns of the parent genotypes, as shown in Figure 6, and it may be necessary to reconcile these inconsistencies. This reconciliation is performed in the implementation during the transformation of the offspring genotypes into their equivalent phenotype representations by modifying the value of the *location* attribute of some of the inconsistent rooms. If a full reconciliation of the locations of all of the rooms in an offspring genotype cannot be achieved, the offspring is discarded from the population, and more crossover operations are performed as needed to produce the required number of offspring at that evolutionary cycle.

4.5 Mutation in GENCAD-FS

Performing mutation on a matrix genotype produces an offspring genotype that has the exact number of house-level elements of its parent, but one of the features of one of its elements and/or one of its connectors is changed. This achieves adaptation in the form of genotype modification. Figure 7 illustrates how a typical mutation operation in GENCAD-FS would modify a matrix genotype. The shaded squares indicate a gene that has been altered in the genotype during mutation. There are two of them because one of the triangles of an adjacency matrix is redundant (adjacency matrices are symmetric around their diagonal).

4.6 Evaluation in GENCAD-FS

The constraints used for evaluation in GENCAD-FS have been acquired from textual descriptions of *feng shui* principles presented in (Rossbach 1987). Eight *feng shui* domain constraints and eight common-sense constraints have been implemented at the house level. Common-sense constraints were included in order to filter out bad designs that don't violate particular *feng shui* principles but would nonetheless not be feasible as proper designs of residences—for instance, designs in which there is a bedroom that is not connected to any of the other rooms in the residence.

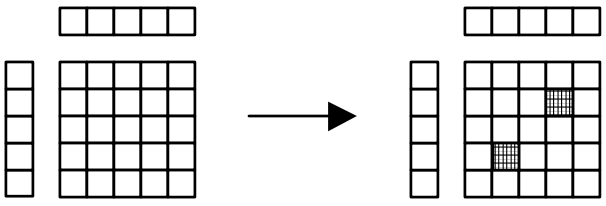

Figure 7. A typical mutation operation performed on a matrix genotype.

Each constraint is implemented as a LISP procedure which returns a value of T if it is violated or a value of NIL if it is satisfied. In situations in which the *feng shui* principle associated with the constraint has a cure, verifying whether the cure is present or not is implemented as a separate LISP function. The cure functions return T if a cure is present for the corresponding constraint, and NIL if no cure is present. Cure checking only occurs if the associated constraint appears to be violated, just before making the final decision on whether the constraint function should return T or NIL.

An example of a constraint at the house level is given by the following quote from (Rossbach 1987):

Traditionally, the Chinese avoid three or more doors or windows in a row...this...funnels ch'i [positive energy] too quickly...[CURE:]...to stop ch'i from flowing too quickly, hang a wind chime or crystal ball... (page 89)

This constraint is implemented by first finding the description of all the connectors at the house level, particularly their locations and directions. If at least three connectors are aligned such that their locations are in consecutive (or the same) grid sectors *and* they all have the same direction (e.g., north-south), then the constraint has been violated. However, before determining this we must check whether or not there are any wind chimes or crystal balls in the house that are positioned in line with the violating doors/windows. The pseudocode that represents this constraint procedurally is the following (given a phenotype P):

```
Get the list C of all connectors in
P;

Get the list Q of all potential
cures in P for this constr.;

For each connector c in C, or until
a bad omen has been found:
   Get the location l of c;
   Get the direction d of c;
```

```
    Set the list of connectors LU
lined up with c to NIL;
   Get the list Reduced of all el-
ements in C except c;
   For each connector r in Reduced:
       If the direction of r is d
And
          The location of r lines
up with l along direction d,
         Then add r to LU;
      End-If;
   End-For;
   If there are two or more connec-
tors in LU And
      no potential cure in Q lines
up with r,
      Then signal a bad omen situ-
ation;
   End-If;
End-For;
```

Similar procedural representations have been implemented for the other constraints.

4.7 Selection in GENCAD-FS

We have defined selection so that the population is constant in size. Selection consists of keeping the N best phenotypes in the augmented population. The population is augmented in each cycle through crossover and mutation of N genotypes. The N best cases selected can include none, some, or all of the potential solutions generated during one evolutionary cycle, and none, some, or all of the potential solutions known at the beginning of the same evolutionary cycle. The exact makeup will vary greatly, depending on the relative quality of the newly generated offspring solutions, as determined during evaluation, and this will depend on whether the random decisions made during crossover and mutation are "lucky" (in that they result in improved solutions) or not.

4.8 Experimental result

We have performed several experiments with GENCAD-FS; here we will only briefly describe one of them. The following implementation strategies were followed when performing the experiment:

- The population size, N, was kept constant across evolutionary cycles.
- M new individuals were produced at each evolutionary cycle, of which 80% were created through crossover and 20% through mutation. We have set the value of M to equal the value of N (the number of initially retrieved cases).
- No biasing of the population according to the fitness values of the individuals in it (a process sometimes confusingly termed "selection" or "reproduction") was performed in our GA. All retrieved cases (or all phenotypes present in a population at any given time) may contribute a significant feature to the final solution, so all are treated equally.

The purpose of the experiment was to analyse the quality of the solutions found by GENCAD-FS by comparing their features to those stylistic characteristics normally present in the Frank Lloyd Wright cases that were used as a basis for constructing the solutions. For the experiment whose results we report here, the following house-level problem specifications were given:

```
((bedroom 3) (bathroom 2) (fire-
place 1) (music-room 1))
```

That is, we asked GENCAD-FS to propose the floor plan of a residence with (at least) three bedrooms, two bathrooms, a fireplace, and a music room, such that their relative positions satisfy the principles of *feng shui*. The solution to this problem that was found by GENCAD-FS is shown in Figure 8.

We have implemented in GENCAD-FS some statistics-gathering code that allows us to trace the ancestry of each offspring design created in the process of solving a given problem. This allows us to make some observations about the way in which each design ever considered came to exist. Some characteristics of the process that GENCAD-FS underwent to produce the solution in Figure 8, as well as of the solution itself, are the following:

- All 12 Frank Lloyd Wright cases were retrieved from memory in order to participate in adaptation, as they all matched one or more of the problem requirements.
- GENCAD-FS converged after 61 evolutionary cycles to the solution shown.

- A total of 28 crossover operations and 3 mutations took place in order to generate this solution (not counting the crossover and mutation operations whose results eventually caused a dead end during the search).
- The solution combines features inherited from three of the twelve cases that were originally retrieved: Avery Coonley House, Robie House (main floor), and Willits House (lower floor). Features from the other nine retrieved cases were probably still present in the population at the 61st generation, but not in the final solution (and were also never part of any of its ancestors).

If we analyse the floor plan shown in Figure 8, and taking into account the observations on this solution made above, we can arrive at the following conclusions about the inheritance of individual features and the creativity of the resulting design:

- The fact that the solution is a floor plan that contains many hallways is a feature that is inherited from Avery Coonley House, which has 4 hallways. Neither of the other two ancestor cases has more than two hallways.
- The protuberance from the dining room is a feature that is inherited from Robie House, which has two such similar protuberances (one from the dining room and one from the living room), or from Willits House which has one (from the dining room).
- The room that is skewed compared to the north-south grid (Bedroom 44) is a feature that is inherited from Willits House, in which the rooms are all similarly skewed.
- The bedrooms in the solution are all in very different parts of the house, something that does not normally occur in a residence. In a family house, at least, the bedrooms are usually all near each other. However, if the residence were intended for several unrelated individuals sharing a house, the solution would be a creative and interesting design that would allow the occupants to have greater privacy than the design of a typical family house.

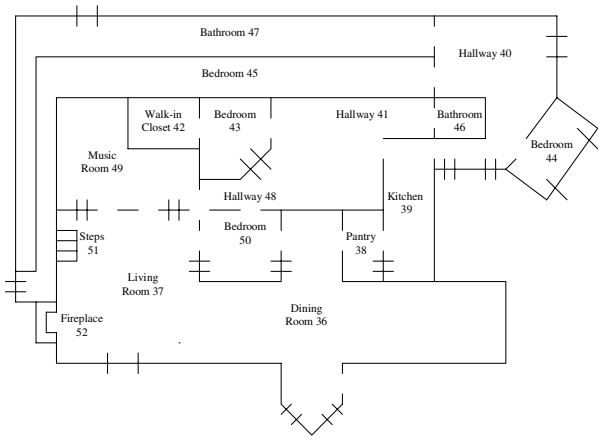

Figure 8. Solution found by GENCAD-FS.

In other words, the solution combines (in a creative way) features from several known previous solutions. The solution also has some of the features considered to be general aspects of Frank Lloyd Wright's style, such as having large dining and living rooms that merge into each other, with no partitions between them, and having a fireplace. The fireplace was a problem requirement, so any solution had to have one to be a valid solution; its location in the living room was inherited from one of the three ancestor cases (all three had this feature).

5 Summary and Discussion

We have presented GENCAD, a process model for design that combines aspects of case-based reasoning and evolutionary algorithms. From the point of view of case-based reasoning, the evolutionary algorithm in GENCAD performs the task of case adaptation. In order to do so it requires domain knowledge with which to recognise good solutions, but does not require domain knowledge with which to generate potential adaptations. This simplifies the implementation of the case adaptation task compared to "traditional" case-based reasoning systems. From the point of view of evolutionary algorithms, the cases retrieved by GENCAD given a new problem specification serve as an initial population of potential solutions for the algorithm to operate on. This gives the evolutionary algorithm an initial bias/help that "traditional" evolutionary algorithms, which are normally initialised with a population of randomly-generated potential solutions, don't have.

Our application of GENCAD to floor plan layout illustrates the process model. One of the things this illustration showed is that GENCAD supports the direct use of different knowledge sources. The use of Frank Lloyd Wright prairie houses and feng shui constraints may not be a likely commercial application of this process model, but it demonstrates that the adaptation and evaluation of design precedents can be based on knowledge sources that were not considered in the original design process.

References

Chachra, V., Ghare, P.M., and Moore, J.M. (1979). *Applications of Graph Theory Algorithms*. New York: North-Holland.

Faltings, B. (1997). Case Reuse by Model-Based Interpretation. In *Issues and Applications of Case-Based Reasoning in Design*, eds. M.L. Maher and P. Pu, 39-60. Mahwah, New Jersey: Lawrence Erlbaum Associates.

Fenves, S.J., Rivard, H. and Gomez, N. (2000). SEED-Config: a tool for conceptual structural design in a collaborative building design environment, *Artificial Intelligence in Engineering*, **14**(3):233-248.

Flemming, U., Aygen, Z., Coyne, R., and Snyder, J. (1997). Case-Based Design in a Software Environment that Supports the Early Phases in Building Design. In *Issues and Applications of Case-Based Reasoning in Design*, eds. M.L. Maher and P. Pu, 61-85. Mahwah, New Jersey: Lawrence Erlbaum Associates.

Hildebrand, G. (1991). *The Wright Space: Pattern & Meaning in Frank Lloyd Wright's Houses*. Seattle: University of Washington Press.

Kane, C., and Schoenauer, M. (1996). Topological Optimum Design Using Genetic Algorithms. *Control and Cybernetics*, Vol. 25, No. 5.

Kolodner, J. (1993). *Case-Based Reasoning*, San Mateo, CA:Morgan Kaufmann.

Leake, D.B. (1996). *Case-Based Reasoning: Experiences, Lessons, & Future Directions*. Menlo Park, California, and Cambridge, Massachusetts: AAAI Press and MIT Press.

Maher, M.L., Balachandran, M.B., and Zhang, D.M. (1995). *Case-Based Reasoning in Design*, Mahwah, NJ:Lawrence Erlbaum Associates.

Maher, M.L., and Pu, P. (eds). (1997). *Issues and Applications of Case-Based Reasoning in Design*. Mahwah, New Jersey: Lawrence Erlbaum Associates.

Mitchell, M. (1998). *An Introduction to Genetic Algorithms (Complex Adaptive Systems Series)*. Cambridge, Massachusetts: MIT Press.

Rosenman, M. (2000). Case-Based Evolutionary Design, *AIEDAM*, **14**(1):17-30.

Rossbach, S. (1987). *Interior Design with Feng Shui*. London: Rider Books.

Vob, A. (1997). Case Design Specialists in Fabel. In *Issues and Applications of Case-Based Reasoning in Design*, eds. M.L. Maher and P. Pu, 301-338. Mahwah, New Jersey: Lawrence Erlbaum Associates.

Zhang, D.M. (1994). *A Hybrid Design Process Model Using Case-Based Reasoning*. Sydney, Australia: Ph.D. Dissertation, Department of Architectural and Design Science, University of Sydney.