# An Evolutionary Approach to Case Adaptation

**Andrés Gómez de Silva Garza** and **Mary Lou Maher**

**Appears in:**
**Case-Based Reasoning Research and Applications. Third International Conference on Case-Based Reasoning, ICCBR-99, Monastery Seeon, Munich, Germany, July 1999, Proceedings.**

# An Evolutionary Approach to Case Adaptation

Andrés Gómez de Silva Garza and Mary Lou Maher

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

FAX: (+61-2) 9351-3031
Phone: (+61-2) 9351-2053
E-mail: {andres,mary}@arch.usyd.edu.au

**Abstract.** We present a case adaptation method that employs ideas from the field of genetic algorithms. Two types of adaptations, case combination and case mutation, are used to evolve variations on the contents of retrieved cases until a satisfactory solution is found for a new specified problem. A solution is satisfactory if it matches the specified requirements and does not violate any constraints imposed by the domain of applicability. We have implemented our ideas in a computational system called GENCAD, applied to the layout design of residences such that they conform to the principles of feng shui, the Chinese art of placement. This implementation allows us to evaluate the use of GA's for case adaptation in CBR. Experimental results show the role of representation and constraints.

## 1 Introduction

Many different methods have been proposed for performing the task of case adaptation in CBR. They have been surveyed in several publications, including [1], [2], and [3]. Different approaches may be better for different domains, different knowledge representation schemes, different reasoning tasks, or other reasons. Approaches may differ on the types of adaptation they support, the amount of change in a case they permit an adaptation to make, the number of cases they can rely on to generate solutions to new problems, and other factors. The adaptation method we present here is flexible, in that it allows for a wide variety of options along all of these dimensions. In our approach, several types of adaptation are available, cases may end up being completely transformed or just slightly tweaked, and final solutions may contain features from one or many cases.

In this paper we present a case adaptation method based on genetic algorithms. In this method, cases are adapted incrementally and in parallel, until a satisfactory solution is found for a given problem. We have employed this approach for design, though it can be used for

other reasoning tasks. Within design, we have tried it out on several domains, though in this paper we focus on just one, introduced below. The main concern of this paper is to describe our process model for case adaptation, not to discuss the quality of the designs produced by the application.

Our case adaptation method supports two broad types of adaptation: parametric and structural. Parametric adaptation of cases is achieved through mutation. Structural adaptation of cases is achieved through crossover. Depending on the specifics of a given domain and the richness of the representation chosen for it, several mutation and crossover operators, with different nuances in the effects they produce, can potentially be made available.

The method assumes that the requirements of a new problem will partially match, and therefore result in retrieving, more than one case in memory. These retrieved cases are used to seed an evolutionary process, i.e., they form its initial population. The adaptations produced by the crossover and mutation operators of the evolutionary process are evaluated, and the best ones selected to participate in the next round of genetic adaptations, until a satisfactory solution is found. Evaluation requires domain knowledge in order to recognise whether proposed solutions are acceptable for a given domain or not; crossover, mutation, and selection can operate independently of the domain.

Depending on which randomly evolved variations on the originally retrieved cases are selected to remain in the population after being evaluated, final solutions may have evolved from just one of the cases, or from all of them. They may differ greatly in structure and/or in parameter values from all of the originally retrieved cases, or may be similar to one or several of them. Thus, the method is useful in a wide variety of problem situations and domains requiring different types and degrees of adaptation.

In the following sections we discuss our evolutionary case adaptation method in more detail, we present an implementation for a specific domain and the knowledge representations we have adopted for this domain, and we give some experimental results.

## 2 Case Adaptation Method

We have developed a process model of design that combines the precedent-centered reasoning capabilities of case-based reasoning (CBR) (see for example [1]) with the incremental evolution of multiple potential solutions, an idea taken from the paradigm of genetic algorithms (GA's) (see for example [4]). The process model involves the use of CBR as the overall reasoning strategy and the use of a GA to perform the case adaptation subtask. Because a general-purpose, knowledge-independent GA is used, case adaptation is knowledge-lean. It is only in the evaluation module of the GA that domain knowledge is required so that proper decisions are made about which potential solutions generated by the GA are useful to keep in future GA cycles.

Our process model is shown in Fig. 1. In this model we assume the existence of a case memory in which descriptions of previously existing solutions are stored. Each case is represented as a set of attribute-value pairs. The cases that are retrieved from memory given a new problem specification are adapted by repeatedly combining and modifying their descriptive features. After each cycle of combination and modification, solutions are evaluated and the best are selected, to be adapted in the next cycle. Through this incremental, evolutionary process, the case adaptation method converges to a satisfactory solution to the new problem. The solution will contain features and/or modifications of features from several of the cases that were initially retrieved from memory. Thus, our process model adapts past

solutions by evolving different combinations of their features in parallel and continuously, until a satisfactory combination is found.

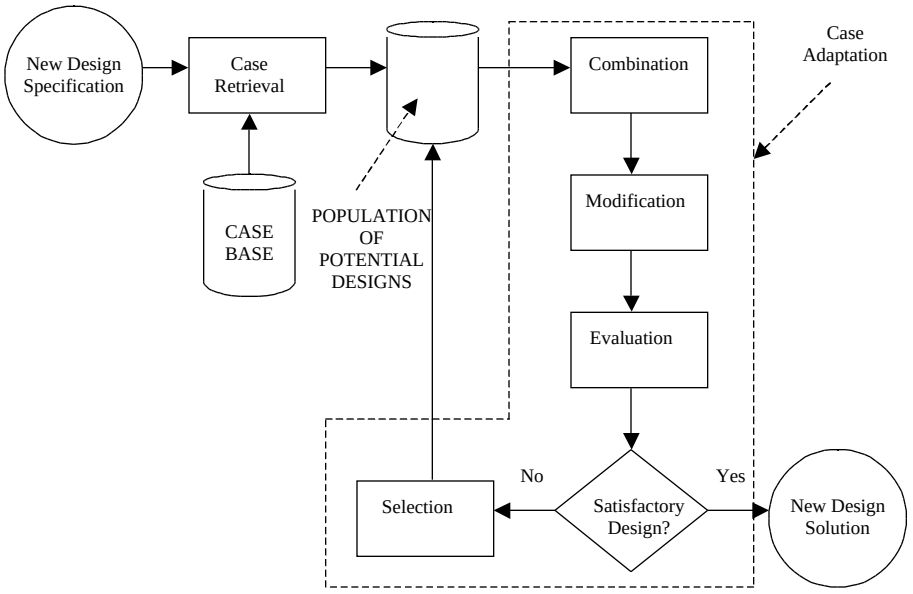


**Fig. 1.** Evolutionary case adaptation method.

The main emphasis of our process model is on proposing new solutions based on the knowledge contained in previously known solutions, i.e., it is a precedent-based approach. But a major component is the evolutionary approach to adapting the known solutions in order to generate solutions to new problems. The two strategies of CBR and GA's complement each other. The cases retrieved from memory serve as the initial population for a genetic algorithm, while the genetic algorithm adapts the cases until it finds an acceptable solution.

The combination subtask of case adaptation performs several cut-and-paste crossover operations. Each crossover is done on two randomly-chosen "parents" from the population of potential solutions, at randomly-chosen crossover points, and produces two "offspring" suggested solutions. The modification subtask performs several mutation operations. Each mutation produces a new "offspring" suggested solution by:

- randomly choosing a "parent" from the population of potential solutions,
- randomly selecting an element to mutate in the description of the parent,
- randomly choosing an attribute of that element to mutate, and
- randomly selecting a new value for that attribute.

Knowledge of which values are valid for which attributes can be used so that mutation does not suggest completely nonsensical solutions. If the process model were to be used to design buildings, for instance, it would be a waste of time for mutation to change the value of the *number-of-stories* attribute from 25 to 834 or –15, for instance.

The evaluation subtask of case adaptation analyses a suggested solution according to domain constraints. Depending on the domain, different constraints may have to be satisfied in order for a solution to be considered acceptable or satisfactory. A fitness value is assigned

during evaluation to each suggested solution. The total fitness F of a given solution, given N constraints ($C_1$ through $C_N$) and M problem requirements ($R_1$ through $R_M$), is calculated with the following equation:

$$F = \sum_{i=1}^{N} C_i + \sum_{j=1}^{M} R_j$$

where $C_i = 0$ if constraint $C_i$ is not violated by the solution or
$C_i = 1$ if constraint $C_i$ is violated by the solution, and
$R_j = 0$ if requirement $R_j$ is met by the solution or
$R_j = 1$ if requirement $R_j$ is not met by the solution.

Convergence to an acceptable solution occurs if an individual in the population has a total fitness of 0, meaning that none of the constraints has been violated and all of the problem requirements have been met.

The selection subtask of case adaptation takes all of the evaluated individuals in a population of suggested solutions, including those inherited from previous adaptive cycles and those generated in the current one, and keeps the *k* best ones to serve as the initial population of the next cycle. The value of *k*, as well as the number of offspring produced at each cycle by crossover and mutation, is chosen so that the size of the population does not change from one cycle to the next. Thus, the value of *k* depends on the number of cases initially retrieved from memory.

In this method of case adaptation, the synthesis of potential solutions is done in a task- and domain-independent fashion. The power of mutation can be enhanced by providing access to some simple domain knowledge, namely the values that are valid for the attributes that describe objects in the domain, as mentioned above. But on the whole, domain knowledge is needed only for evaluating the generated solutions to determine their quality. In other words, recognition (analytical) knowledge, rather than generative knowledge, is needed to apply our method to a given domain.

## 3 Implementation and Domain

We have implemented our ideas in a computational system named GENCAD written in Common LISP. Our method of case adaptation has been applied to the structural engineering design of high-rise buildings [5] and to the layout design of residences such that they conform to the principles of feng shui (pronounced “fong sway”), the Chinese art of placement. Here we describe the feng shui application.

Feng shui, also known as Chinese geomancy, is an ancient 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.

There are several different feng shui sects that may contradict each other or place different priorities on different aspects of residential layouts. Despite this variety, of prime importance to performing any feng shui analysis is information on the relative positions of objects. In addition, other attributes of objects are usually also taken into account, such as

their orientations, shapes, and relative sizes. In our work we have used the knowledge of feng shui presented in [6], which corresponds to the Tibetan black-hat sect of feng shui.

Feng shui analyses different aspects of a residential layout to determine its auspiciousness or lack thereof. Some classes of inauspicious layouts can be "cured" by the proper placement of an acceptable curing object. Thus, feng shui knowledge is complex, in that some potentially bad layouts can actually be acceptable if the proper cure is present. It is not just a matter of determining whether a layout is "good" or "bad," but even if it would normally be considered bad, one has to determine whether it has been cured or not before rejecting it outright.

The feng shui knowledge contained in [6] 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).

GENCAD applies its case adaptation GA to one of the three levels of description of a residence at a time. This is because there are very few feng shui constraints that relate objects belonging to different levels of description; the constraints involve relations between objects within the same level. Thus, potential solutions to the new problem at the landscape level can be evolved (and evaluated) independently from potential solutions to the same new problem at the house level, etc. For other domains, GENCAD's GA might have to operate on and evolve hierarchical solutions containing several levels of description at once. This will have implications for the speed of convergence as well as the complexity of the implementation of the crossover and mutation operators.

## 4 Knowledge Representation

*Feng shui* analysis assumes knowledge of spatial relationships among the objects at 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 has been chosen to describe the locations of objects within each of the three levels. We locate objects on each level in 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 as follows, with north assumed to be at the top of the page:

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

Objects can occupy more than one grid sector, and grid sectors can contain more than one object, 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 objects that typically need to be represented at each of the three levels, it is adequate in most cases.

### 4.1 Case Representation

GENCAD's case library currently contains 12 cases, each of which describes one of Frank Lloyd Wright's prairie houses, obtained from [7]. Note that the designs of these houses do not necessarily conform to the principles of feng shui. However, designs that are acceptable to feng shui practitioners can still be generated by evolving combinations and mutations of the features of the design cases. If the original cases did conform to feng shui practice, given a new problem, convergence to a solution acceptable to feng shui practitioners might be faster, but this is not a requirement of our case adaptation method.

Each of GENCAD's design cases is a residence described at the landscape, house, and room levels. Within each level, objects are represented using attribute-value pairs to describe features that are relevant to feng shui analysis. Some attributes such as locations and types of objects are required for all objects, whereas others such as shapes and steepness are optional, and don't even make sense for some objects. A diagrammatic example of a residence at the landscape level is shown in Fig. 2. This is followed by an abbreviated version of the symbolic case representation of the same residence.

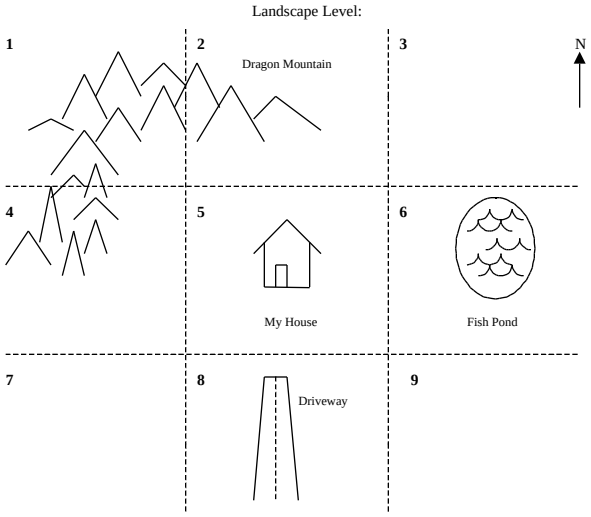


**Figure 2.** A residence and its place in the landscape.

```
(((level landscape)
  (elements (((type mountain) (name dragon-mountain)
              (location (1 2 4)) (steepness high) ...)
             ((type pond) (name fish-pond) (location (6))
              (clarity murky) ...)
             ((type house) (name my-house) (location (5)))
             ...)))
  ...)
```

When running GENCAD at the landscape level, this is the fragment of a case that would form part of the population of the GA. The fragments describing the house and room levels would be dealt with separately. The list of attribute-value pairs is modified through mutation and combined with that of other cases through crossover as the GA proceeds.

### 4.2 Representation of Feng Shui Analysis Knowledge

Feng shui analysis knowledge is used in the evaluation function of the GA. We have taken the text description of the analysis knowledge and converted it to a set of constraints; each constraint is implemented as a procedure. There are several constraints at each of the three levels of feng shui description.

An example of a feng shui constraint at the landscape level, quoted directly from [6], is:

> A house facing a hill will be bad...CURE: If a house faces a mountain and the backyard is a garden, place a spotlight in the back of the garden and shine it toward the top of the house, or install a flagpole at the rear of the garden to balance ch'i. [Page 35]

This constraint is implemented by first finding the description of all the houses and mountains/hills at the landscape level, particularly their locations and the orientations of the houses (if known). A predicate *facing* has been written that, given the location and orientation of an object, and the location of a second object (within the 3x3 grid), determines whether or not the first object faces the second (even partially). If any of the houses is located and oriented such that it faces any of the mountains/hills in the landscape, then the constraint has been violated. However, first we must check whether or not a cure is present for the constraint violation, i.e., if there is a garden behind the violating house, and if so whether there is a flagpole in it, or a spotlight oriented towards the house. A predicate *behind* has been written that, given the location of an object, and the location and orientation of a second object, determines whether or not the first object is behind the second. The pseudocode that performs this analysis, i.e., the procedural representation of the constraint, given a proposed solution at the landscape level S, is shown as follows:

```
Get the list H of all houses in S;
Get the list M of all mountains/hills in S;
Get the list C of all potential cures for this constraint
   in S;
For each house h in H or until a bad omen has been found:
   Get the location lh of h;
   Get the orientation oh of h;
   For each mountain/hill m in M or until a bad omen has
   been found:
      Get the location lm of m;
      If facing(lh,oh,lm) Then:
         Get the list G of all gardens in S;
         Set flag g-behind? to False;
         Repeat
            Get the next unprocessed garden g in G;
            Get the location lg of g;
```

```
   If behind(lg,lh,oh) Then
      Set flag g-behind? to True;
Until g-behind?=True or all gardens in G have
    Been processed;
If g-behind?=True Then
   For each potential cure c in C or until a bad
    omen has been found:
      Get the location lc of c;
      Get the type tc of c;
      If tc=spotlight Then:
         Get the orientation oc of c;
         If facing(lc,oc,lh) and subset(lc,lg)
            Then signal a bad omen situation;
      Else
         If subset(lc,lg)
            Then signal a bad omen situation;
```

## 5 Evaluation and Experimental Results

In this section we evaluate our evolutionary case adaptation method according to three issues: the coverage of the method, its efficiency, and the quality of the solutions it produces.

### 5.1 Coverage

Often, CBR is criticised because even large case bases are not guaranteed to cover the entire search space, thus making some problems unsolvable using "pure" CBR. In our framework, even small case bases can provide sufficient information on typical structures and contents of solutions to problems in the domain for the method to eventually converge to a solution. Of course, the larger the case base, the more cases are likely to be retrieved given a new set of problem requirements, and the faster the GA is likely to find a satisfactory adaptation of their features and converge.

If N cases are initially contained in the population of the GA, then after 1 cycle of the GA the proposed solutions in its population will combine features from at most 2 cases (due to crossover). Thus, after N-1 cycles some of the proposed solutions in the population can combine features from all of the N retrieved cases. The selection operator in the GA ensures that only those combinations that seem to be leading towards an acceptable solution are kept for future GA cycles, i.e., it helps to prune the search.

But even an exhaustive search of all the possible combinations of the features of all retrieved cases is not guaranteed to find satisfactory solutions to the new problem. The inclusion of a mutation operator in the GA, in addition to combination, ensures that all points in the search space can potentially be reached. Of course, whether a certain point will be reached or not depends on the particular sequence of mutations and combinations followed during a given application of the GA to the retrieved cases. The mutation operator introduces into the proposed solutions features that weren't present in any of the originally retrieved cases, or different values for those features that were present. Thus, our method can potentially cover the entire search space, even if a large case base is not available.

### 5.2 Efficiency

We have explored the efficiency of combining GA's with CBR by comparing our method with a GA that is exactly the same except for the lack of cases. In the alternative method, instead of initiating the GA search with a population consisting of cases retrieved from memory, we initiated it with randomly generated "cases" (i.e., random starting points in the search space). In this way, any differences in efficiency will be attributable to the use of CBR as the guiding framework, and we can evaluate our decision to combine the two AI paradigms of CBR and GA's.

In order to perform this efficiency experiment, GENCAD was run 20 times using 12 cases retrieved from a case base of floor plans of Frank Lloyd Wright prairie houses, and 20 times using 12 randomly-generated cases, on the same problem. The problem specification for this test problem (at the landscape level) is:

```
(((level landscape)
  (requirements ((house 1) (river 1) (trees 2)))))
```

This problem specification can be interpreted as "we want to build a house on a property in which there is a river, and we're thinking of planting two clumps of trees around the house." The problem is now to use GENCAD to generate a configuration containing these four elements, specifying their relative positions within the landscape, such that the configuration is auspicious according to the principles of feng shui.

GENCAD was given a limit of 500 GA cycles in which to find an acceptable solution, i.e., if convergence did not occur by cycle 500, the search was ended without a solution being given. Some of the cases in the randomly generated case base, as well as the Frank Lloyd Wright cases, do contain two clumps of trees, and/or a house, and/or a river in the landscape. In addition, there are configurations of these four types of element that are valid according to feng shui practice. Therefore, achieving a solution through the cyclical combination and/or mutation of the cases retrieved from either case base is theoretically possible.

In the experiment, 5 of the 20 trials using the random starting points converged. Similarly, 5 of the 20 trials using the Frank Lloyd Wright cases converged. Thus, whether cases or random starting points are used to initiate the search doesn't seem to make a difference as far as the frequency of convergence. However, a clear difference can be seen when we analyse the number of GA cycles required before convergence occurred (in those trials in which it did occur), as seen in Table 1.

**Table 1**. GA cycles required before convergence:

| Trial # | Random | Trial # | FLW cases |
|---|---|---|---|
| 1 | 114 | 25 | 54 |
| 9 | 333 | 31 | 34 |
| 11 | 357 | 36 | 32 |
| 14 | 274 | 37 | 406 |
| 17 | 160 | 39 | 90 |
| **Avg.:** | 241.6 | **Avg.:** | 123.2 |

As can be seen from the results, when cases are used to guide (i.e., provide starting points for) the search, convergence occurs on average twice as fast as when the search is initiated from random starting points. This demonstrates the efficiency of combining the ideas of CBR with those from GA's. Convergence does not always occur, as can also be seen (or does not occur within a reasonable number of iterations). Whether it will converge or not, or how rapidly it will converge, can vary greatly due to the random nature of the genetic operators of crossover and mutation. However, the process can be applied again and again to the same problem, using the same initial set of retrieved cases, and it is possible that it will converge in future attempts.

### 5.3 Quality

The use of CBR as the overall framework helps ensure that the solutions proposed by our method are of high quality. For example, a typical problem specification for a floor plan layout at the house level is that the house should have 3 bedrooms and 2 bathrooms. A residence of this size typically also has, as a minimum, a kitchen, a living room, and a dining room. These are not normally given as requirements, but it is an implicit assumption that any solution will have these additional rooms.

Now let us assume that we used the problem specification mentioned in the last paragraph to perform a GA search using randomly generated initial solutions, or to perform an exhaustive search of the solution space, for instance. Such searches would most probably eventually find a solution that has 3 bedrooms and 2 bathrooms, and that satisfies any domain constraints (such as relationships among the rooms acceptable to feng shui practitioners). But it would be likely that these would be the only components that would be present in the solution. Unless further knowledge and heuristics were used to guide the search, solutions would be minimalistic.

Instead, by using cases that include kitchens, living rooms, and dining rooms (and perhaps additional rooms that might be considered to be useful *post facto* such as pantries) to initiate the search, the solutions to which our method will converge will most likely also include these important but unspecified rooms. Thus, the quality of solutions proposed by our method is equal or greater than if CBR were not used as the guiding framework. Cases provide complete scenarios that serve to guide both the structure and contents of proposed solutions.

## 6 Discussion

We have presented a case adaptation method that is based on ideas from genetic algorithms. Variations on retrieved cases are evolved incrementally, and at each cycle their quality is verified and the best variants from amongst the initial population plus the new variants generated at the current cycle are kept. This evolutionary method of case adaptation combines the benefits of case-based reasoning and other knowledge-based approaches with those of general-purpose problem solvers such as genetic algorithms.

For instance, being able to use starting points for problem solving search based on similar past experiences, and being able to apply the process model to highly-specialised problem solving domains are two advantages of CBR. On the other hand, having a large number of operators with greatly differing effects available, and being able to apply the process model to a wide variety of problem solving domains are two advantages of GA's. Our evolutionary method of case adaptation benefits from having all of these characteristics.

Domain knowledge is required and represented in the form of constraints used for the evaluation of proposed solutions; this is recognition knowledge, not generative knowledge. This difference with other approaches is especially important in applying our method to tasks such as design. In design it is relatively easy to recognise whether a proposed design is an acceptable solution for a given problem or not, whereas it is quite difficult to come up with a set of reasoning steps or heuristics to follow that will lead to the generation of acceptable designs. The knowledge engineer's task of knowledge elicitation and knowledge acquisition is thus simplified when using our evolutionary approach to case adaptation.

This use of constraints for evaluation rather than generation is one of the differences between our work and that of others that have used constraint-satisfation techniques in the context of CBR, for instance [8], [9], [10], or [11]. In these projects, constraints with potentially complex interactions guide the generation of solutions to new problems by adapting past cases. This generation of solutions uses domain knowledge or heuristics to make what is generally an NP-complete problem tractable. In our method, the constraints are independent of each other, and they help in a cumulative fashion to eliminate bad solutions, rather than in a mutually interacting way to generate good ones.

There has been other work in the past that has combined concepts from GA's with CBR. [12] presents a GA that is initialised based on the information held in cases. However, in [12] cases contain descriptions of past executions of a GA (e.g., the values of the GA parameters, the task environment in which those parameter values were used successfully, etc.), irrespective of the type of problem being solved with the GA. Thus, cases help the GA dynamically adapt to changing problem situations; the authors use concepts from CBR in aid of GA's. In our work, on the other hand, cases contain descriptions of known solutions for the type of problem being solved, and these cases provide guidance for the search that our case adaptation GA will perform; thus, we use concepts from GA's in aid of CBR.

The research presented in [13] is more similar to ours, in that cases contain descriptions of solutions to the type of problem being solved, and a GA is used to adapt the cases to solve the problem. However, [13] is not a pure CBR approach, as only a small fraction (10%-15%) of the initial population in the GA comes from cases in memory; most of the initial population is generated at random, as in a classical GA. The authors do this for valid reasons of balancing exploration and exploitation in their GA search, but it provides a different flavour to their research. Again, their work places more of an emphasis on the GA, and on making it efficient and effective, than on contributing to CBR research. In contrast, we have examined the possibilities of using a GA for case adaptation from the perspective of CBR.

## References


1. Kolodner, J.L.: *Case-Based Reasoning*, Morgan Kaufmann Publishers (1993)
2. Leake, D.B.: *Case-Based Reasoning: Experiences, Lessons, & Future Directions*, AAAI Press/The MIT Press, Boston (1996)
3. Maher, M.L. and Pu, P. (eds.): *Issues and Applications of Case-Based Reasoning in Design*, Lawrence Erlbaum Associates, Mahwah, New Jersey (1997)
4. Mitchell, M.: *An Introduction to Genetic Algorithms (Complex Adaptive Systems Series)*, MIT Press, Boston (1998)
5. Gómez de Silva Garza, A. and Maher, M.L.: A Knowledge-Lean Structural Engineering Design Expert System, *Proceedings of the Fourth World Congress on Expert Systems*, Mexico City, Mexico (1998)
6. Rossbach, S.: *Interior Design with Feng Shui*, Rider Books, London (1987)

7. Hildebrand, G.: *The Wright Space: Pattern & Meaning in Frank Lloyd Wright's Houses*, University of Washington Press, Seattle (1991)
8. Zhang, D.M.: *A Hybrid Design Process Model Using Case-Based Reasoning*, Ph.D. dissertation, Department of Architectural and Design Science, University of Sydney, Australia (1994)
9. Hinrichs, T.R.: Plausible Design Advice Through Case-Based Reasoning, in Maher, M.L. and Pu, P. (eds.), *Issues and Applications of Case-Based Reasoning in Design*, 133-159, Lawrence Erlbaum Associates, Mahwah, New Jersey (1997)
10. Faltings, B.: Case Reuse by Model-Based Interpretation, in Maher, M.L. and Pu, P. (eds.), *Issues and Applications of Case-Based Reasoning in Design*, 39-60, Lawrence Erlbaum Associates, Mahwah, New Jersey (1997)
11. Pu, P. and Purvis, L.: Formalizing the Adaptation Process for Case-Based Design, in Maher, M.L. and Pu, P. (eds.), *Issues and Applications of Case-Based Reasoning in Design*, 221-240, Lawrence Erlbaum Associates, Mahwah, New Jersey (1997)
12. Ramsey, C.L. and Grefenstette, J.J.: Case-Based Initialization of Genetic Algorithms, *Proceedings of the Fifth International Conference on Genetic Algorithms*, 84-91, Morgan Kaufmann Publishers (1993)
13. Louis, S.J. and Johnson, J.: Robustness of Case-Initialized Genetic Algorithms, *Proceedings of FLAIRS (Florida Artificial Intelligence Conference) '99*. To appear (1999)