FORMALISING DESIGN EXPLORATION AS CO-EVOLUTION

A Combined Gene Approach

MARY LOU MAHER, JOSIAH POON AND SYLVIE BOULANGER
University of Sydney, Australia

Abstract. This paper introduces a model for design exploration based on notions of evolution and demonstrates computational co-evolution using a modified genetic algorithm (GA). Evolution is extended to consider co-evolution where two systems evolve in response to each other. Co-evolution in design exploration supports the change, over time, of the design solution *and* the design requirements. The basic GA, which does not support our exploration model, evaluates individuals from a population of design solutions with an unchanged fitness function. This approach to evaluation implements search with a prefixed goal. Modifications to the basic GA are required to support exploration. Two approaches to implement a co-evolving GA are: a combined gene approach and a separate spaces approach. The combined gene approach includes the representation of the requirements and the solution within the genotype. The separate spaces approach models the requirements and the solutions as separately evolving interacting populations of genotypes. The combined gene approach is developed further in this paper and used to demonstrate design exploration in the domain of braced frame design for buildings. The issues related to the coding of the genotype, mapping to a phenotype, and evaluation of the phenotype are addressed. Preliminary results of co-evolution are presented that show how exploration differs from search.

1. Introduction

Most computer-based design tools assume designers work with a well-defined problem. The traditional treatment of design as two discrete phases: problem formulation and solution synthesis, is challenged by recent research. Though the view on discrete phases may be applicable to a simple and trivial design task, current research (Logan and Smithers, 1993; Corne, Smithers and Ross, 1994; Gero, 1994; Jonas, 1993; Navinchandran, 1991) has shown that design is an ill-structured problem and the discrete phases view is not a good (or correct) description of design.

Design is an iterative interplay to “fix” a problem from the problem space and to “search” plausible solutions from the corresponding solution space. The features and constraints in the current solution can become new criteria that lead

to a redefined problem space, which in turn helps to generate a new design space. We call this phenomenon *exploration*. This is a phenomenon generally observed during conceptual design rather than detailed design. The present design tools from different domains, CAD or CASE, assume the designer has a clear picture of the problem and solution. These tools become a burden rather than help when the designer tries to alter the design. It is because these tools are not designed to cope with a change of requirements, change of design goals, change of assumptions, etc. Hence, if this phenomenon is not fully understood, it becomes difficult to know what kind of assistance could be provided to the designer during conceptual design.

The difference between search and exploration can be characterised by the input and output as illustrated in Figure 1. A typical search process generates a solution as its output with a well-defined problem as its input. However, an exploration process derives a problem and the corresponding solution from an ill-defined problem. It is not only because of the ill-defined nature of the problem that requires us to explore during design, the solution space also creates a need for this exploration.

Figure 1. Input and output of search and exploration.

Genetic Algorithms (GAs) (Holland, 1962) provide an alternative to traditional search techniques by simulating mechanisms found in genetics. Three notions are borrowed from biological systems:

- the *phenotype*, which can be a living organism for biological systems or a design solution for design systems;
- the *genotype*, which is a way of representing or encoding the information which is used to produce the phenotype; and
- the *survival of the fittest*, which determines whether a genotype survives to reproduce.

In GA systems the genotype is usually represented as a binary string whose length varies with each application. For example, a genotype may look like: 001001101. GAs manipulate a representation (genotype) that differs from its

expression (phenotype) in order to perform changes that couldn't be possible at the phenotype level. The genotype representation allows combination or mutation to occur in order to construct better strings. Some measure of fitness is applied to each string after combination and mutation to determine which strings participate in generating the next generation of the population.

A simple genetic algorithm considers a population of n strings and applies the operators: reproduction (or selection), crossover, and mutation in order to create the next generation. Reproduction is a process in which strings are copied according to their fitness function. Crossover is a process in which the newly reproduced strings are mated at random and each pair of strings partially exchanges information. Mutation is the occasional random alteration of the value of one of the bits in a string. Algorithms used to implement these processes are described in detail in Goldberg (1989).

In this paper a design process based on a genetic algorithm is presented which can model characteristics of explorative design: the search for problem definition as well as the search for solution. The use of an evolutionary system in which the genotypes represent alternative problem definitions and alternative solutions provides the basis for the co-evolution of problem space and solution space.

1.1. EXPLORATION IN DESIGN

Since design has been categorised as a problem solving activity (Simon, 1969), design is treated as a search of the solution space for a result. This idea has dominated the direction of artificial intelligence in design for some time. However, the validity of this hypothesis has been queried by recent work. For example, Corne, Smithers and Ross (1994) suggest that it is inappropriate to consider design as a search problem because a search problem requires a well defined problem space whereas a design problem is usually ill-structured. They propose design as "exploration" as follows:

.. involves the construction and incremental extension of problem statements and associated solutions ..

Logan and Smithers (1993) further elaborate this definition that

.. the formulation of the problem at any stage is not final ... As the design progresses, the designer learns more about possible problem and solution structures as new aspects of the situation become apparent and the inconsistencies inherent in the formulation of the problem are revealed. As a result, .. the problem and the solution are redefined...

Navinchandra (1991) defined exploration in the program CYCLOPS as

.. Exploration is the process of generating and evaluating design alternatives that normally would not be considered..

He focuses on alternatives and this is achieved through criteria relaxation and criteria emergence. The relaxation is not constraints relaxation but a

relaxation of the threshold value. This changing of threshold values causes a part of the solution space which is originally inside bound of the pareto curve to be explored. Solutions in this inside bound solution space can be examined as alternatives. Emergence described in CYCLOPS is a recognition activity. Criteria from precedent cases may be recognised to be relevant and interesting enough to apply to the current situation. The introduction of new criteria adds a new dimension for the designer to consider. The new criteria will be included to be part of the evaluation of design solution.

Another definition of “exploration” is provided by Gero (1994) that:

.. Exploration in design can be characterised as a process which creates new design state spaces or modifies existing design state spaces...

This definition extends the “state space” concept of search (Simon, 1969), so that the state space is changed during exploration. This definition implies that the solutions in the given or predefined state space are insufficient for exploration. Gero continues to suggest that

.. exploration precedes search and it, effectively, converts one formulation of the design problem into another .. Part of designing involves determining what to design for (function or teleology), determining how to measure satisfaction (behaviour), and determining what can be used in the final artefact (structure) ..

The definition relates exploration to search, indicating that exploration precedes search, and at the same time differentiates exploration from search.

Maher (1994) provides a definition which also relates search and exploration:

.. search becomes exploration where the focus of the search changes as the process continues ..

This definition identifies search as a part of exploration, but not the same as exploration and also characterises the two as distinct, i.e. search has a definite goal while exploration doesn't. An approach to adaptive design, with the capability for exploration guided by the human designer, is described in Maher and Kundu (1994). Search as part of exploration cannot guarantee convergence because the design requirements change with the design solutions at the same time. However, convergence criteria could be externally defined and separate to the design requirements, recognising the fact that design usually completes when time has run out or factors external to the concerned problem.

For the remainder of this paper, we present a formal model of exploration. The model is illustrated in Figure 2 as the interaction of problem space and solution space. The problem space (or the functional requirements) is represented by P , and the solution space is represented by S . *Exploration* is defined as a phenomenon in design where P interacts and evolves with S over time.

Figure 2. Problem-design exploration model.

The phenomenon of exploration as illustrated in Figure 2 has the following characteristics:

1. There are two distinct search spaces: Problem Space and Design Space.
2. These state spaces interact over a time spectrum.
3. Horizontal movement is an evolutionary process such that:
 - (a) Problem space $P(t)$ evolves to $P(t+1)$, $P(t+2)$, and so on;
 - (b) Solution space $S(t)$ evolves to $S(t+1)$, $S(t+2)$, and so on.
4. Diagonal movement is a search process where goals lead to solution. This can be *Problem leads to Solution* (downward arrow) or *Solution refocusses the Problem* (upward arrow).

The problem space $P(t)$ is the design goal at time t and $S(t)$ is the solution space which defines the current search space for design solutions. The solution space $S(t)$ provides not only a state space where a design solution can be found, but it also prompts new requirements for $P(t+1)$ which were not in the original problem space, $P(t)$. This is represented by the dashed upward arrow from design space $S(t)$ to problem space $P(t+1)$. The upward arrow is an inverse operation where $S(t)$ becomes the goal and a *search* is carried out in the problem space, $P(t+1)$, for a *solution*. This iterative relationship between problem space and design space evolves over time.

This model of exploration depicts an evolutionary system, or in fact, two evolutionary systems. The evolutionary systems are the problem space and the solution space. The evolution of each space is guided by the most recent population in the other space. This model is called co-evolution and provides the basis for a computational model of design exploration. The basis for co-evolution is the simple genetic algorithm where special consideration is given to the representation and application of the fitness function so that the problem definition can change in response to the current solution space.

1.2. RELATED RESEARCH IN GENETIC ALGORITHMS

Genetic algorithms provide the basis for modelling evolutionary systems. The application of GAs to design include the solution to the truss design problem. The ten-member truss problem (Goldberg and Samtani, 1986) aims to find the optimal weight of each member for a given pre-determined configuration, such that the whole structure is stable and has a minimum weight. The configuration and the fitness function remains unchanged throughout the GA process. This represents a basic application of genetic algorithms to a design optimisation problem.

Watabe and Okino (1993) further study this problem by searching for structural shape as part of the problem. This is achieved by the introduction of new genetic operator called T-mutation. There are two types of T-mutation. The first one, T1-mutation, adds one new node to a randomly selected bar. The second one, T2-mutation, changes the topological structure without changing the number of nodes. The effect after the application of the T-mutation results in a new species which consists of individuals with different structural configurations. The changed configuration opens up a new solution space to search. This application of a genetic algorithm shows how the representation of the genotype determines the level at which the search occurs, in this case the search included a search for a configuration. However, the goal as defined by the fitness function remains to be the minimum weight configuration.

SAGA (Harvey, 1992), Species Adaptation Genetic Algorithms, allows the genotype to change in length as well as content so that species can emerge. He suggests that the notion of a search space is a metaphor when the question of "Where in this whole search space is the optimum?" is asked. However, this metaphor implies a space of pre-defined extent with a predefined goal. If a structure is to be evolved with potentially unrestricted capabilities, the simple GA, which has fixed length genotypes, is not an appropriate tool. The capability to represent a variable-length genotype is important to evolution. As the length increases, the population evolves as a species rather than global search. However, his model does not show how individuals from the solution space can affect the problem space.

Koza (1992) recognises the importance of co-evolution and suggests the term in biology is sometimes used to reflect the fact that all species are simultaneously co-evolving in a given physical environment. He uses Game Playing Strategy to elaborate on co-evolution, where two players in a game are represented as two populations of individuals. The fitness of a strategy of a player is measured by its performance against all strategies deployed by the other player. The fitness is, thus, a relative score. The performance of the two players continue to evolve with respect to the strategies by the opposing player. The mutual interactions and implicit relationships between players in a game are extended to a general conclusion as follows:

In co-evolution, there are two (or more) populations of individuals. The environment for the first population consists of the second population. And conversely, the environment for the second population consists of the first population ... Co-evolution is a self-organising, mutually bootstrapping process that is driven only by relative fitness.

This provides a model for co-evolution where two solution spaces evolve in competition to each other, yet the goal remains the same. Co-evolution is affected by each search space defining the threshold for survival in the other search space. We present a co-evolutionary system in which the two spaces are not in competition with each other, yet they evolve in response to each other. Three differences between the co-evolution of Game Playing Strategy and the co-evolution of Problem-Design Space are:

1. The two populations in a game are opponents with the aim to beat each other, whereas in our co-evolution model, the aim is to explore the Problem Space and Design Space and to help each other to acquire better fitness values.
2. The purpose of co-evolution in the Game Playing Strategy is to measure how good an individual strategy can stand when played against various strategies by the opponent, while our co-evolution model aims to measure how good an individual from a population can satisfy (adapt) the expectations of individuals from another population.
3. The same fitness function is used for both spaces in the Game Playing Strategy, only the threshold for reproduction is changed in co-evolution, while our co-evolution model applies a potentially different fitness function to each space.

2. A Co-Evolutionary Process for Explorative Design

A simple GA, as shown below, is the basis for developing an evolutionary process model for explorative design.

When we apply the simple GA to the design process, we assume the process begins with an initial population of design genes that provide the information needed to generate a design solution. The evaluation determines which genotypes survive. The evaluation is performed by evaluating a fitness function and operates on the phenotype, which in the design process is the design solution. The processes of selection, crossover, mutation, and evaluation are the basis of the search for a design solution.

```
t = 0;
initialise genotypes in Population(t);
evaluate phenotypes in Population(t) for fitness;
while termination condition not satisfied do
```

```

t = t + 1;
select Population(t) from Population(t-1);
crossover genotypes in Population(t);
mutation of genotypes in Population(t);
evaluate phenotypes in Population(t);

```

Selection is a process in which individuals are copied according to their fitness function. This means that an individual with a higher value has a higher probability of contributing one or more offspring in the next generation. This operator is an artificial version of natural selection, a Darwinian survival of the fittest among individuals. In a natural population, fitness is determined by an individual's ability to survive. In the context of design, a fitness function representing the design requirements determines whether a design is suitable or not. Once an individual has been selected for reproduction, an exact replica of the individual is made. This individual is then entered into a mating place for further genetic operator action.

Crossover is a process in which the newly reproduced individuals are mated at random and each pair of individuals partially exchange information using a cross site chosen at random. For example if we consider the individuals $A1 = 0110 \mid 1$ and $A2 = 1100 \mid 0$ (where the separator symbol is indicated by $\mid$), the resulting crossover yields the following two new individuals $A1' = 01100$ and $A2' = 11001$. Crossover in a design process occurs when two design concepts are partially combined to form a new design concept.

Mutation is the occasional random alteration of the value of one of the bits in an individual. When used sparingly with reproduction and crossover, mutation is an insurance policy against loss of notions. In fact mutation plays a secondary role in the operation of GAs because the frequency of mutation to obtain good results in empirical GAs studies is on the order of one mutation per thousand bit transfers (Goldberg, 1989). Mutation has the potential to make small changes to a design concept, rather than a crossover process that makes large changes. We do not employ mutation in our co-evolutionary model of design.

Evaluation is a process of determining if a genotype continues in the next round of crossover. The termination condition is usually related to the evaluation, that is, when the evaluation of the population yields a suitable design, the process is terminated. Evaluation in the design process occurs by testing the performance of the design against relevant criteria. In the GA model of design, the fitness function is the basis for evaluation. The fitness function as a representation of design requirements can be predefined for the entire search process or it can be allowed to change as the genotype population changes. By changing the fitness function in response to the current population, the process models the ability of designers to change their focus when an interesting solution is found. This can be modelled as co-evolution of the design space and

the performance space, where each space then becomes the population of genotypes for its own evolution and the fitness function for the other space.

The co-evolution of the design genes (solution space) and the fitness function (problem space) provides a model for design as exploration. Two approaches to representing coevolution are:

- **CoGA1:** A single composite genotype is formed by the combination of a problem requirements and a design solution. The fitness function is defined locally for each design solution.
- **CoGA2:** The two spaces are modelled as two sets of genotypes and phenotypes: one for modelling problem requirements and one for modelling design solutions. The current population of each space provides the fitness measurements for the other.

2.1. COGA1: COMBINED GENE APPROACH

This first co-evolving algorithm has two modifications to the basic GA:

1. The fitness function (problem part, P) and design solution (solution part, S) are put into one genotype.
2. There are two phases of crossover-evaluation operations in each generation instead of the convention of one phase.

The algorithm, CoGA1, is shown below.

CoGA1

```

t = 0;
initialise genotypes in Population(t);
evaluate phenotypes in Population(t) for fitness;
while termination condition not satisfied do
    t = t + 1;
    select Population(t) from Population(t-1);
    /* Phase 1: from S to P */
        crossover genotypes in Population(t) at Performance_space;
        evaluate phenotypes in Population(t);
    /* Phase 2: from P to S */
        crossover genotypes in Population(t) at Design_space;
        evaluate phenotypes in Population(t);

```

Inside the repeating loop, there are two phases of GA operations for each generation. If no satisfactory solution is found in previous operations with the stated problem, the problem is revised to give new dimensions for the solution space. Hence, the first phase corresponds to the shift of attention of fitness function when a solution space is given, i.e. the upward arrow from *S* to *P* in our model of problem-design exploration. In phase 1, crossover occurs in the problem part of the genotype, as illustrated in Figure 3(a). For example, the crossover point to the parent genotypes cut the problem part to P11 and P12,

and P21 and P22. For the same solution carried forward from the previous phase, the fitness is evaluated using a different fitness function, i.e. the same S1, which is evaluated by P11 and P12 in parent genotype, is evaluated by P11 and P22 in the new recombined child genotype. The fitness value for each design solution represents a local fitness.

After the problem is revised, the second phase relates to the search for a solution with the reformulated fitness function from Phase 1. This corresponds to the downward arrow from *P* to *S* in the model, as shown in Figure 2. Crossover occurs on the design solution part of genotypes: the S11, S12, S21 and S22 in Figure 3(b). The fitness of a design solution is not evaluated by a common global fitness function, but by the fitness function defined as the problem part in the same genotype. In other words, the fitness score of each genotype is again a local fitness value. In our example, the offspring which has solution part composed of S11 and S22 is evaluated by P1; while the other offspring, which has S21 and S12 in its solution part, is evaluated by P2.

Figure 3. Crossover operation for CoGA1.

After phase 2 the solutions are used to check the termination condition to determine whether another generation is necessary. Currently, the termination condition is defined as a fixed number of generations, meaning that the exploration stops when a predetermined amount of time has passed. However, the termination condition can be any globally defined condition that does not evolve in response to the alternatives found in the solution space.

2.2. COGA2: TWO INTERACTING POPULATIONS

A second approach to co-evolution is to maintain separate spaces of genotypes for problem requirements and design solutions. As illustrated in Figure 4,

CoGA2 uses the current selection from each population to be the fitness function for evaluating the individuals in the other population. The problem requirements is modelled as a collection of criterion, where each criterion is represented as a genotype in the Problem Space.

Every problem criterion genotype has a label and a weighting (i.e. the genotype has a length of 2). A problem is, thus, a combination of individual genotypes with their current weights. If we allow the crossover operator to cut and paste a different weight to a criterion, followed by selecting a random number of genotypes, these problem criteria will collectively define a problem which has a different perspective and emphasis to be solved. The fitness of a solution is defined by the current collection of criterion. In the other direction, the fitness of a criterion is defined by the number of times that criterion is satisfied in the current collection of individuals in the Solution Space.

The CoGA2 starts with initialising the two populations which represent problem and solution. An initial evaluation of individuals from the Solution Space is performed using the initial design requirements as defined by the user. The initial evaluation of the Problem Space is performed based on the selected individuals from the Solution Space. The termination condition is checked and the pattern of “phases” appear in CoGA2 as well. Each phase in CoGA2 corresponds to a different evaluation function, rather than to a different crossover operation as in CoGA1. The CoGA2 algorithm is shown below.

CoGA2:

```

t = 0;
initialise genotypes in Problem_space(t);
initialise genotypes in Solution_space(t);
initial-evaluate phenotypes in Solution_space(t) for fitness according
    to user's initial requirements;
initial-evaluate phenotypes in Problem_space(t) for fitness according
    to user's initial requirements;
while termination condition not satisfied do
    t = t + 1;
    /* Phase 1: from S to P */
        select Problem_space(t) from Problem_space(t-1);
        crossover genotypes in Problem_space(t);
        evaluate phenotypes in Problem_space(t) for fitness
            according to selected individuals from
                Solution_space(t-1);
    /* Phase 2: from P to S */
        select Solution_space(t) from Solution_space(t-1);
        crossover genotypes in Solution_space(t);
        evaluate phenotypes in Solution_space(t) for fitness
            according to selected sample of individuals from
                Problem_space(t-1);

```

Figure 4. Fitness evaluations in CoGA2.

3. The Combined Gene Approach for Braced Frame Design

The design domain selected to demonstrate the combined gene approach to co-evolution is steel braced frames. The domain provides a range of design solutions that can be described geometrically and more than one design focus. The design of braced frames for buildings is done by a structural engineer with some constraints/requirements imposed by the architect's layout of the building. Two alternatives for the design focus considered here are: design for architectural compatibility and design for structural resistance. These alternatives are represented as more specific design performance criteria, as listed below.

1. Architectural compatibility
 - integration of frame into bay layout
2. Structural resistance
 - efficiency of the frame
 - integrity of the frame

Architectural compatibility is evaluated according to how well the braced frame fits the bay layout as predefined by an architect in building design. Structural efficiency is evaluated in one of two ways: as a measure of the efficiency of the use of material in the braced frame or as a measure of the integrity or ductility of the braced frame. The result of this formulation of the design problem is that there are three possible ways to evaluate the performance of a braced frame: how well the frame fits the bay layout; the structural

efficiency of the frame; and the integrity of the frame. The implication of this formulation is that one focus may lead to a different set of designs to the set that would result from another focus.

In addition to the design focus, the design of braced frames is defined by the geometric requirements imposed by the architectural design of the building. The problem definition is formalised through the values of a set of parameters, as shown below.

<i>parameter</i>	<i>range</i>
number of storeys (n_storey)	5-20 storeys
storey height (h_storey)	3.0 - 4.5 m
bay width (w_bay)	5.0 - 20.0 m
design focus (criteria)	compatibility, efficiency, or integrity

The braced frame solution description is represented by seven features, where each feature may take on one of a range of values :

<i>feature</i>	<i>range</i>
type of panel (type)	cross, single diagonal, chevron, diamond
flip within panel (flip)	none, horizontal, vertical
mirror above panel (mirror)	no, yes
eccentricity in panel	
—at top beam level (eccent_1)	none, 1/8 or 1/4 width of frame
—at bottom beam level (eccent_2)	none, 1/8 or 1/4 width of frame
multi-storey panel (m_storeys)	1, 2 or 3
ratio of bay size to panel width(c_panel)	0.5 to 2.0

A solution is generated from the above features by taking the type of the panel (i.e. cross, single diagonal, chevron, or diamond) and applying the transformations defined by the values of the remaining features. Figure 5 illustrates these transformations on a panel. Representing the alternative design solutions as a set of transformation features allows for a very rich generation of alternatives by concentrating on the geometric manipulation of the panel. These features provide the basis for the genotype representation.

Additional parameters, not coded directly in the genotype, are derived from the above features and contribute to the description and the evaluation of the frame. These parameters are shown below. The combination of the features in the genotype and the derived parameters below provide the basis for a phenotype description. This distinction is important because the genotype is the basis for producing the next generation, the phenotype is the basis for evaluation and the survival of the fittest.

PANEL OF BRACED FRAME	FEATURES
cross single diagonal chevron diamond →	type = chevron <i>This feature selects one of the four types of panel listed to the left.</i>
→	flip = horizontal <i>This feature takes the panel to the left and flips it about a horizontal axis.</i>
→	mirror = yes <i>This feature takes the panel to the left and mirrors the geometry onto the panel above the top beam.</i>
→	eccent_1 = w4 (w_panel/4) <i>This feature takes the panel to the left and introduces an eccentricity at the diagonal intersection with the top beam.</i>
→	eccent_2 = w8 (w_panel/8) <i>This feature takes the panel to the left and introduces an eccentricity at the diagonal intersection with the bottom beam.</i>
→	m_storeys = 2 <i>This features takes the panel to the left and extends it vertically to two storeys.</i>
→	c_panel = 1.25 <i>This feature takes the panel to the left and extends it or compresses it horizontally.</i>

Figure 5. Example of braced frame features and their use in defining an alternative.

parameter	equation
width of panel	$w_panel = c_panel \times w_bay$
width of frame	$W_frame = w_panel$
height of panel	$h_panel = m_storeys \times h_storey$
height of frame	$H_frame = n_storey \times h_storey$
aspect ratio of panel	h_panel / w_panel
aspect ratio of frame	H_frame / W_frame
zone of feasibility	z_feasib
coefficient of ductility	$c_ductility$

There is an important distinction between three entities: the frame, the bay and the panel. A frame is an assembly of adjacent panels covering the full height and full or partial width of the building. A bay relates to the layout of beams in the building within which the frame is integrated and represents the distance between two columns (vertical elements). A panel is a unit of the frame which may be repeated vertically and/or horizontally.

In this example we define four possible fitness functions: one evaluates the difference between the initial problem parameters and current parameters; the other three evaluate the fitness of the current features for a design focus. The first evaluation function is defined in order to follow the variation of the population from the initial requirements, remembering that during exploration the design requirements may change. The other evaluation functions are defined to measure the performance of the phenotype according to a selected design focus, remembering that the combined gene approach to co-evolution includes the definition of performance criteria within the genotype.

The evaluation functions return a value between 0 and 1. The functions themselves are based on heuristic evaluations of the alternatives, that is, no mathematical modelling or analysis of the alternatives is performed. When a function returns a value of 0.0, the alternative will not survive to reproduce in the next generation. The lower the value, the less probability of its use for the next generation. The details of the evaluation functions for braced frame design are given in the Appendix.

3.1 REPRESENTATION OF BRACED FRAME AS GENOTYPE AND PHENOTYPE

Figure 6 illustrates a drawing of a frame and its associated formal description for this demonstration. Each frame is considered in the context in which it was designed; this is a consequence of the combined gene approach where the genotype includes both the problem description and the design solution. Consider a designer who needs to find a suitable frame for an 8-storey building with 3.7 m storey height using 12 m bays, and the designer chooses to focus on the structural efficiency of the frame. By letting the problem parameters and the solution features vary, it is possible to explore a wide range of potential solutions. One suitable solution, as shown in Figure 6, is a cross-braced panel, over two storeys, which is 7.8 m wide. It is interesting to note that the initial bay width was 12 m. This solution satisfies structural efficiency very well with a value of 0.95.

The genotype of a braced frame comprises the initial problem parameters, including the design focus, and the features of the design solution. Each genotype is represented as a fixed length binary string. The bits in a genotype are grouped into chunks, where each chunk is a contiguous block of bits. The position of each chunk uniquely identifies a parameter or feature of the frame and the binary string in the chunk maps onto the value of that feature/parameter.

In this example, each chunk occupies 5 bits, i.e. each chunk represents 32 possible values.

Figure 6. Example of a complete braced frame description.

In our combined gene approach, a genotype consists of two parts: the problem part and the solution part. The values found in the chunks among the problem part represent the design requirements and focus while the solution part provides values to describe the geometry of the solution. The template of a genotype is shown in Figure 7, the positions before the double vertical line are for problem parameters, and those which are found after the lines stand for solution features.

criteria	ln_storey	h_storey	w_bay	type	flip	mirror	eccent_1	eccent_2	m_storeys	c_panel
----------	-----------	----------	-------	------	------	--------	----------	----------	-----------	---------

Figure 7. Genotype template of a braced frame.

One genotype in a population is represented as a binary string as shown in Figure 8. The figure has three parts: the top part shows the genotype as a binary string, the second part shows the mapping from binary string to attribute-value pairs and the third part shows the distribution of values in a 5 bit chunk,

Since the features/parameters do not have 32 possible values, a mapping is defined to identify the value to be used. In mapping some features, we used a biased coding scheme such that certain values of an attribute are more preferred than the other options. For example, there are three possible values for the feature “eccentricity in panel”: none, w/8, and w/4. The value none is preferred over the introduction of an eccentricity. In the mapping we have given the value none a probability of 0.5, and w/8 and w/4 probabilities of 0.25. The biased mapping for “eccentricity in panel” is shown in the “chunk” in Figure 8, where the first 16 values for the parameter are mapped onto none, etc.

Figure 8. A genotype representation of a braced frame.

The phenotype is the representation of the braced frame that is evaluated. The phenotype representation is defined through a mapping from the genotype representation followed by a derivation of additional parameters. Basically, the phenotype is represented as a set of attribute-value pairs. The mapping process involves three steps:

1. Divide the genotype into chunks.
2. Convert the binary string in each chunk to a decimal value.
3. Map this value to the appropriate symbolic value.

The first two steps are completed by a fixed mapping procedure, while the last step is different for each attribute. The mapping takes into consideration the biased coding of the feature. Hence, the feature “mirror” has two options, no or

yes, and there is no bias. Therefore, the chunk number of the mirror feature is determined using the following rules:

```

if   chunk_number < 16
then mirror = no
if   chunk_number >= 16
then mirror = yes

```

The mapping is defined in such a way as to be consistent in either direction. Once the mapping is completed, values for additional parameters can be derived from these mapped values. For example, the aspect ratio of the panel (ar_panel) is defined to be h_panel/w_panel . The value of h_panel is defined to be $m_storeys \times h_storey$, etc.

The phenotype is given a fitness value according to one of the four evaluation functions introduced above. The evaluation functions are:

```

f0: distance from initial requirements
f2: compatibility with bay layout
f3: structural efficiency
f4: structural integrity

```

The details of how the braced frame phenotype is assigned a value for each of these functions is given in the Appendix. These evaluation functions provide the value used to determine the probability of the genotype being used in the next generation. The initial population is evaluated with a special evaluation function in order to start with a population that is relevant to the initial requirements. This function ensures that the designer's focus is considered before the exploration process begins. The initial fitness function is defined below, where f_{init} is the value of the evaluation function that matches the designer's original focus.

F_i : initial fitness function = $f_0 + f_{init}$

where $f_{init} = f_2, f_3$ or f_4

In addition to these evaluation functions, another evaluation function has been defined to follow the performance of the population. A global fitness function, F_g , is defined to be the average of the values of the four evaluation functions. This value is not used in the selection process since CoGA1 is guided by a local fitness function.

3.2. RESULTS OF CO-EVOLUTION

This section presents the results of using CoGA1 on the braced frame problem formulation. We decided to generate the initial population of genotypes randomly and to run the co-evolution with a genotype population of 100. The

algorithm uses roulette selection, i.e. fitness proportionate selection. Each new population is generated with a probability of crossover being 1.0. No mutation is applied at this stage.

In run A, the design focus is defined to be structural integrity for a building that is 20 storeys high, each storey has a nominal height of 3.8 m, and the suggested bay width is 12 m. In summary, the design requirements are:

Criteria (i_criteria)	= structural integrity
No. of storey (i_n_storey)	= 20
Height of storey (i_h_storey)	= 3.8 m
Bay width (i_w_bay)	= 12 m

The best solution proposed after 23 generations is a very reasonable design solution (Figure 9).

Figure 9. Intermediate and final solutions for run A.

The chevron with an eccentricity of $w/4$ at the top beam level satisfies the initial focus (integrity) very well and does not deviate significantly from the initial problem parameters. Although the final best fit is different than the designer's solution, it remains compatible with the concept of an eccentrically braced frame chevron. Additionally, an interesting best fit emerges at generation 18, with an eccentricity introduced in what appears to be a diamond. In reality, this solution resulted from both flip and mirror features acting on a single chevron panel. Finally, the best fit of generation 23 was first proposed at generation 9.

This indicates that convergence was not conclusive at generation 9 and more exploration was required. The pursuit of a moving target (or change in focus) resulted in a variety of alternatives before the final best fit was reached. Hence, the evolution of alternatives is also reflective of the exploratory approach expressed by the interplay between problem and solution spaces.

The algorithm converged after 23 generations with a design focus of structural integrity. The chart shown in Figure 10 is a summary of the percentage of the population that had each evaluation function as its local fitness (design focus) across all 23 generations until convergence. The chart shows the variation in focus for the braced frame solutions. This demonstrates that our goal of exploration as a change in focus during the design process occurred through the rise and fall of the proportion of the population that used each design focus.

The graph in Figure 11 shows the overall performance of each generation as defined by the value of F_g . F_g did not influence the fitness of the phenotypes, it is only shown here to measure overall performance in a computational method that allows design exploration. It is interesting to note the fluctuation of the value of the best fit solution vs the average value of the entire population. This implies that the best fit influences the fitness of the entire population, regardless of its local fitness function. In the graph below the population at $t=0$ represents the fitness of the randomly generated genotypes, the population at $t=1$ and greater represent phenotypes that have been evaluated for selection. Note that the global fitness after $t=1$ increases almost monotonically.

Figure 10. Evolution of design focus for run A.

Figure 11. Overall performance of population during co-evolution for run A.

The chart shown in Figure 12 is a summary of the results of a different set of initial requirements, run B. It is shown here to illustrate a different evolution of the design focus through a different distribution of phenotypes with a specific design focus in each generation. This distribution is remarkably different to Figure 10. In Figure 12 one design focus dominates at convergence, where in Figure 10 two different foci are present at convergence. In fact, as shown in Figure 12, design focus f3 starts to dominate at generation 18 and at convergence the dominate design focus is f4. This further demonstrates design exploration through a change in design focus.

Figure 12. Evolution of design focus for run B.

4. Discussion

The development and implementation of a model of exploration as co-evolving spaces has enabled a computational approach to changes in design focus. The implementation as a modified genetic algorithm raises a number of issues.

The quality of the initial population has important effects on how the evolution progresses. If generation 0 is not filtered by F_i , the evolution converges quickly to the focus functions held by strong individuals of the initial population. This suggests that the random number selected for a particular run can be very influential on which functions dominate at convergence. If generation 0 is filtered by F_i , this tendency is present but not as strong. For instance, applying the initial F_i may reduce the overall F_g value from generation 0 to generation 1, but at least, f_{i1} has a better probability of being favoured although its value at generation 1 is low. This may influence the number of generations until convergence but not the pattern of exploration.

The quality of the evaluation functions in terms of how they relate to each other and not just how adequately they attribute a value within one evaluation function is important. A heterogeneous quality of functions may have an immediate influence on which focus function takes over for convergence. In one case, applying a weight < 1.0 to f_4 made the next run shift from f_4 to f_3 as a final focus. Although this is an important parameter to desensitise, at the same time, a shift does not necessarily lead to low values for other functions.

The relative ease of satisfying a particular design focus changes the way exploration works in this approach. In the first several runs we included an additional design focus for architectural compatibility which checks whether the frame accommodates openings as specified by the architect. We called this fitness function f_1 . For the current formulation it was much easier to satisfy f_1 than the other fitness functions. As a result, all runs were similar - f_1 dominated and the best fit solution was always one that satisfied f_1 . We removed f_1 for consideration until we could better calibrate its evaluation to be similar to the other functions for design focus. This implies that the coding of the design focus functions need to be considered in terms of their ease of satisfaction relative to the other functions.

Pool size does not appear to have marked influences on the final results. Rather, a higher number of individuals in the population seems to reduce the number of iterations required for convergence. However, similar patterns of exploration emerge.

The mapping approach used has a non-conclusive effect on the final results. Scaling possible values of one feature from 7 values to 32 values, while retaining the same minimum and maximum values, has caused an important shift favouring the evaluation function which uses that modified feature. However, no clear pattern has surfaced yet.

The parameters which need to be analysed more carefully are: the sensitivity of threshold values used on F_i to influence the initial population; the absence or presence of f_0 as a problem evaluation function which co-exists with solution evaluation functions such as f_2 , f_3 and f_4 ; the interdependence of evaluation functions in terms of their maximum and obtainable values; the ease with which designs can attain high evaluation values; the random number applied to a run; the bias and granularity applied to map a parameter's symbolic values unto a scale of 32.

5. Conclusion

The co-evolution of problem space and solution space shows promising results on how changes in focus provide a framework for explorative design. In this paper we present a model of exploration, a method of exploration that uses co-evolution, and an implementation of this method for the design of braced frames. Our preliminary results show that design exploration can be modelled as computational methods. Our preliminary results also show that the representation of the design domain can result in an unstable method - that is, one that produces a different result for the same problem in two different runs. This instability may be a desirable feature since it provides a computational model that is as unpredictable as human designers. However, we need to test the model further to understand why there is instability and how the genotype, phenotype, and evaluation functions influence the nature of the exploration.

In addition to further testing CoGA1, the model of exploration needs to be taken further to develop CoGA2, the separate spaces formulation, and to consider formulations that allow a design focus to emerge that has not been identified before the generation of alternatives.

Acknowledgments

This work is supported by the Australian Research Council and an Australian Postgraduate Research Award.

References

- Chien, E.: 1988, *Multi-Storey Steel Building Design Aid*, Canadian Institute of Steel Construction, Toronto, Canada.
- Corne, D., Smithers, T. and Ross, P.: 1994, Solving design problems by computational exploration, in J. S. Gero and E. Tyugu (eds), *Formal Design Methods for Computer-Aided Design*, North-Holland, Amsterdam, pp. 293-313.
- Gero, J. S.: 1994, Towards a model of exploration in computer-aided design, in J. S. Gero and E. Tyugu (eds), *Formal Design Methods for Computer-Aided Design*, North-Holland, Amsterdam, pp. 315-336.

- Gero, J. S., Louis, S. J. and Kundu, S.: 1994, Evolutionary learning of novel grammars for design improvement, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, **8**, 83–94.
- Goldberg, D. E.: 1989, *Genetic Algorithms: In Search, Optimization and Machine Learning*, Addison-Wesley, Reading, MA.
- Goldberg, D. E. and Samtani, M. P.: 1986, Engineering optimization via genetic algorithm, *Proceedings of the Ninth Conference on Electronic Computation*, pp. 471–482.
- Harvey, I.: 1992, Species adaptation genetic algorithms: A basis for a continuing SAGA, in F. J. Varela and P. Bourguine (eds), *Toward a Practice of Autonomous Systems, Proceedings of First European Conference on Artificial Life*, MIT Press/Bradford Books, Cambridge, Mass.
- Holland, J. H.: 1962, Concerning efficient adaptative systems, in M. C. Yovits, G. T. Jacobi, and G. D. Goldstein (eds), *Self-organizing Systems*, Spartan Books, pp. 215–230.
- Jonas, W.: 1993, Design as problem-solving? or: Here is the solution - What was the problem? *Design Studies*, **14**(2), 157–170.
- Koza, J. R.: 1992, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, Cambridge, Mass.
- Logan, B. and Smithers, T.: 1993, Creativity and design as exploration, in J. S. Gero and M. L. Maher (eds), *Modelling Creativity and Knowledge-Based Creative Design*, Lawrence Erlbaum Associates, Hillsdale, NJ, pp. 139–175.
- Maher, M. L.: 1994, Creative design using a genetic algorithm, *Computing in Civil Engineering*, ASCE, pp. 2014–2021.
- Maher, M. L. and Kundu, S.: 1994, Adaptive design using a genetic algorithm, in J. S. Gero and E. Tyugu (eds), *Formal Design Methods for Computer-Aided Design*, North-Holland, Amsterdam, pp. 245–263.
- Michalewics, Z.: 1992, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, Berlin.
- Navinchandra, D.: 1991, *Exploration and Innovation in Design*, Springer-Verlag, New York.
- Simon, H. A.: 1969, *The Sciences of the Artificial*, MIT Press., Cambridge, Mass.
- Watabe, H. and Okino, N.: 1993, Structural shape optimization by multi-species genetic algorithm, in C. Rowles, H. Liu and N. Foo (eds), *Proceedings of the 6th Australian Joint Conference on Artificial Intelligence (AI'93)*, Melbourne, Australia, pp. 109–116.

Appendix: Evaluation functions for braced frame design

There are four evaluation functions used in this example for evaluating the performance of a braced frame:

- f0: measure of closeness to initial requirements,
- f2: measure of conformance to bay layout
- f3: measure of structural efficiency
- f4: measure of structural integrity

The first function, f0, uses a combination of three functions, each concentrating on measuring the distance between the current phenotype and the one problem parameter in the initial requirements. The value of f0 is the average value of these three functions: f01, for the parameter n_storey, f02, for the parameter h_storey, and f03, for the parameter w_bay. The value of each function is determined using a normal distribution shape applied to n_storey, h_storey and w_bay, as shown in Figure A.1. The vertical axis shows the value of f0 and the horizontal axis shows the value d_parameter which corresponds to the normalised difference between the phenotype and the parameter. For each parameter, an acceptable deviation was assumed. The calculation of the value of f0 is shown below.

```

/* acceptable deviations from the initial problem parameters */
bldg->sig_n_storey = 2      /* storeys */
bldg->sig_h_storey = 0.2    /* metres */
bldg->sig_w_bay = 1.5      /* metres */

/* normal distribution shape function used to assess the difference */
bldg->f01 = exp(-pow((bldg->n_storey - bldg->i_n_storey),2)/
(2 * pow(bldg->sig_n_storey,2)))
bldg->f02 = exp(-pow((bldg->h_storey - bldg->i_h_storey),2)/
(2 * pow(bldg->sig_h_storey,2)))
bldg->f03 = exp(-pow((bldg->w_bay - bldg->i_w_bay),2)/
(2 * pow(bldg->sig_w_bay,2)))

/* sum of the squares method for obtaining function f0 */
bldg->sum_square = pow(bldg->f01, 2) + pow(bldg->f02, 2) + pow(bldg->f03, 2)
bldg->f0 = pow(bldg->sum_square, 0.5)

```

The evaluation function, f2, is concerned with evaluating how well the phenotype satisfies the design focus of architectural compatibility. This function measures how well the panel configuration merges with the bay size as defined by the architect. Designers are often faced with fitting a frame into a proposed layout. In order not to restrict the geometry immediately to the proposed bay width, some variation is permitted. This variation is represented by the panel coefficient which ranges from 0.5 to 2.0. The panel width is obtained by

multiplying the bay width by the panel coefficient. This can simulate the interaction between owner/architect/engineer during the preliminary phases. An architect proposes a layout, the engineer proposes a frame, the emphasis on architectural compatibility and structural efficiency might be discussed, resulting in a layout change, or a frame geometry change. If the layout change is an alternative, it is interesting not to limit the program to satisfy the bay size but to give it a better rating when it does. The function f_2 described in Figure A.1 gives the best rating of 1.0 to a perfect fit when $c_{\text{panel}} = 1.0$. A rating of 0.0 is given when the resulting frame is less than 5 or greater than 15 metres.

Figure A.1. Evaluation functions f_0 , f_2 , f_3 and f_4 .

The evaluation function, f_3 , measures structural efficiency by considering load resistance through axial forces as more efficient than through bending forces. Designers initially aim to obtain reasonable proportions of a frame so that member selection and detailed verification does not initiate drastic iterations of their design. To obtain reasonable proportions, aspect ratio combinations are used. They are based on one designer's experience and data from existing building frames (Chien, 1988). The aspect ratios of the panel and

of the frame provide a good indication of its performance. The performance is divided into zones of feasibility ranging from 0 to 5, as illustrated in Figure A.2.

Figure A.2. Zones of feasibility for f3 as a function of aspect ratios

The function f3 described in Figure A.1 gives a better rating for a lower zone. A lower (more feasible) zone is obtained when the frame in question fits aspect ratio guidelines proposed by Chien (1988). The highest (least feasible) zone gets a rating of 0.0, indicating that the phenotype is not very well suited for its design focus. The code below shows the calculation of the value of f3 when the panel type is chevron.

```

if((bldg->type == TY_CHEVRON) &&
    (bldg->mirror == NO) &&
    (((bldg->eccent_1 == EC_NONE) &&
      ((bldg->eccent_2 == EC_W8) || (bldg->eccent_2 == EC_W4))) ||
      ((bldg->eccent_1 == EC_W8) || (bldg->eccent_1 == EC_W4)) &&
      (bldg->eccent_2 == EC_NONE))) &&
    (bldg->m_storeys == 1) &&
    (((bldg->ar_panel > 0.3) && (bldg->ar_panel < 0.4)) &&
      ((bldg->ar_frame > 4) && (bldg->ar_frame < 10))))
    bldg->z_feasib = 1;

```

The evaluation function, f4, assesses structural integrity through a measure of redundancy and plasticity. Designers need to consider the integrity of the frame, in particular for frames subjected to seismic loads, and for large surface structures such as arenas and exposition halls. The configuration of the panel

and the eccentricities introduced at the member intersections, provided that adequate measures are taken at the member design level, ensure that multiple load paths and planned plastification will occur instead of hinge formations and member instability. A ductility coefficient is given according to these configuration factors, varying from 0.5 to 1.3, as shown in Figure A.2. A better frame gets a 0.7 coefficient. A less ductile frame gets a coefficient of 1.0 . A coefficient of 0.5 means the frame is too ductile. Such a frame gets a rating of 0.0.