Problem Solving: Search, Exploration and Co-evolution

Josiah Poon

Information Environments Program
Dept. of CSEE
University of Queensland
Brisbane, QLD 4073, Australia
josiah@csee.uq.edu.au

Abstract

Genetic Algorithms have become useful problem solving tools and they are frequently applied in the area of optimization. The nature of an optimization problem is that both the search space and the objectives are well-defined in advance. However, back to the nature where the metaphor comes from, an individual species does not evolve in isolation, a species co-evolves with the other species. In this paper, the nature of problem solving is examined and two evolutionary algorithms are proposed to overcome the limitation of a Simple Genetic Algorithm (SGA). The new algorithms are Exploratory Genetic Algorithm (ExpGA) and Co-evolutionary Genetic Algorithm (CoGA) which are capable to handle ill-defined problems. It is further suggested that the SGA is a special case of the CoGA.

1 INTRODUCTION

John Holland is considered to be the father of Genetic Algorithms. Holland (1975) tackled the issue of adaptation in natural and artificial systems. His theoretical work laid the foundation for the conventional Genetic Algorithm (GA). In a conventional GA, the global optimum is searched in a solution space under a fixed fitness function.

The modern GA attracted greater attention when Goldberg (1989) introduced this adaptive search technique to a larger community. Two keywords from the title of this book – search and optimization - shifted the attention to the optimization area which further draws researchers from the engineering discipline to use GA as a problem solving tool. The characteristics of an optimization problem is that a performance evaluation function is defined in advance, and this function remains the same throughout the process when the solution is sought.

To consider GA as a tool for optimization imposes the view that GA is another implementation of a state-space search. Harvey (1992) suggests that the notion of using a search space as a metaphor is like asking the question: "Where in this whole search space is the optimum?", but this metaphor implies a space of pre-defined extent with a predefined goal. In most situations, a large part of the effort is expended to find out what the problem is and how the problem boundary should be defined. The perspective of GA as a mere state-space search restricts the potential of GA in problem solving. Problem solving is not simply to search for a solution from a given well-defined space. Problem solving is, in fact, similar to a definition offered to Artificial Life by Langton (1992),

> "… In addition to providing new ways to study the biological phenomena associated with life here on Earth, *life-as-we-know-it*, Artificial Life allows us to extend our studies to the larger domain of "bio-logic" of possible life, *life-as-it-could-be* ..."

Hence, problem solving should be viewed as the process of what the sort of solution a problem can lead to, rather than as solutions we have already known. This paper aims to investigate the appropriateness of a Simple Genetic Algorithm (SGA) for the revised view to problem solving. In the next section of this paper, the process of problem solving is studied to highlight the essential characteristics of a well-defined problem and an ill-defined problem. Three different types of evolutionary algorithms (based on GA) are presented in Section 3, namely the SGA, the Exploratory GA (ExpGA) and the Co-evolutionary Algorithm (CoGA). A formal notation is used to accompany these algorithms to highlight their similarities and differences. It is then followed by a discussion to address the relationships between these algorithms and their relations with problem solving. The paper is summarized by the conclusion in Section 4.

2 PROBLEM AND PROBLEM SOLVING

The terms "problem" and "problem solving" convey different meanings to different people. These terms are revisited in this section using three cases to reveal the nature and characteristics that can stem from these terms.

Case 1. The first case study is cryptarithmetics. A typical question for this kind of problem is

```
  SEND
+ MORE
 MONEY
```

where goal of this problem is to find out what digits in the range of 0 to 9 that each of these letters represent. A digit can only be mapped to one letter. The solution space is a clearly defined space of digits of 0 to 9. There are explicit constraints specified in advance, i.e. no repetition of digits. To solve the problem, a reasoning process can be employed. For example, the two initial numbers are 4-digit long while the sum of them has 5 digits. The number that can appear in the most significant digit by adding these two numbers can only be 1 in this scenario, hence "`M = 1`" is deduced. The reasoning process continues until the mapping for all the letters are found. The reasoning process may sometimes come to an impasse, by then, the steps are backtracked and to go for the alternatives. These problems are usually characterized by a clearly defined evaluation function together with some well-defined constraints. And the problem solving process can be characterized by using the state-space search paradigm in which the problem remains unchanged throughout the process, while the solution can be found from this pre-defined and fixed solution space. The problem solving process is the traversal of possible solutions in this space (Figure 1).

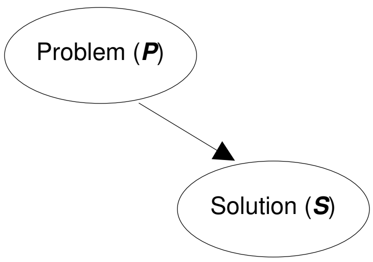

Figure 1 Problem solving as search

Case 2. Another case study to problem solving is the scientific discovery. BACON is the pioneer computational system by Langley et al (1987) in this area. The system could re-discover quantitative empirical laws by finding patterns from a given set of numerical data. The system could derive some of the useful laws in physics in an intelligent way.

On the surface, this problem seems to resemble the search paradigm of problem solving where the goal of the system is to find a relationship of the given parameters according to the observed data. However, Langley et al (1987) reported in their post-morterm analysis that the system only mimics part of the discovery process of its human counterpart. The critical decision on the choice of what input data to be analyzed were made by the researcher. A significant research time is required by human scientists to make hypotheses on the possible relationships among the parameters before the data are analyzed. In other words, given a solution space, scientists make special focus to the problem before the search goes on. The result produced from a focussed search helps the scientists to determine if the right focus has been concentrated. The scientists shift attention based on the current results for the next attempt if *the* solution is not found. Hence, when BACON started crunching the data to determine the relations among the data from a search, the researchers decided a focus of the search bound by some pre-identified parameters. In other words, BACON worked with a well-defined problem after the human researchers have untied the knots of a complex problem. The tactics of this problem solving approach is to explore the solution space by paying special focus in each search, and the search may shift to a different area in the solution space according to the feedback of the interim solution. This is a kind of problem solving by exploration (Figure 2).

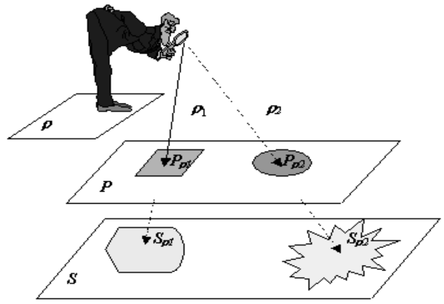

Figure 2 Problem solving as exploration

Case 3. The last case study is a floor plan layout design. In general, a client asks an architect to design a house with a preliminary requirement of room-types, sizes, budget etc. If this were a true state-space search paradigm, there should exist a floor plan (or a definite set of plans) in the search space for architects to recognize as the final solution. This same plan should be proposed by any architect working on the given set of requirements. The selected plan would be commonly agreed by all other architects to be the best design within the requirements and constraints specified by the client. As soon as this design solution is found, the design process could be called to a halt. That is to say, there is a *right* plan for a given set of requirements while the other plans are wrong!

However, this does not occur in real design projects. The reason is that the requirements are generally ambiguous, imprecise and inconsistent, and the client always has second thoughts on seeing the interim plans produced by the architect. The client revises his requirements after reflection to the extra information provided by the plan. Frequent changes to the requirements gives us a non-stationary fitness landscape in the solution space, whereas optimization is characterized by an unchanged goal and a static fitness landscape. An optimum design solution in one landscape may be a poor performer when the landscape is modified. As a result, the final solution is too ambiguous to be named as an optimum floor plan. The final plan is admitted as a "satisficing solution" (Simon, 1981).

To consider this problem as state-space search model imposes a restriction of not allowing the requirements (goal) to change. The concept of a problem being defined once-and-for-all is not generally true (Logan and Smithers, 1993). However, simply considering the process as exploration is not sufficient because this eliminates the interaction (mutual influence) between the problem space and the solution space. Hence, to generate a solution for a given initial problem is not the whole of the problem solving process, part of the process to solve an ill-defined problem is to deliver a well-defined problem and an associated well-defined solution (Figure 3). This type problem solving is classified as co-evolution.

Two or more interacting species evolve in parallel in a co-evolution. The interaction between the species may cause new genetic materials to inject to the other species. For instance, many plants in North America are adapted through co-evolution by hummingbirds. The hummingbird carries genetic material from one species to another while pollinating the flowers. By carrying the genetic material, different colors are transferred from one plant to another. Therefore, a co-evolutionary approach to problem solving allows the requirements to change during the process where the change is due to the interaction with the new knowledge from the solution space. New variables may be introduced which alter the problem space. This phenomenon applies to the solution space as well.

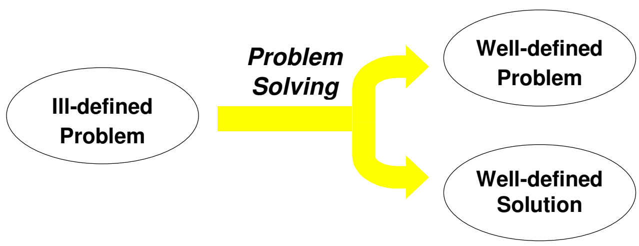

Figure 3 Outcome of solving an ill-defined problem

To conclude this section, a problem is characterized into well-defined*ness* and ill-defined*ness*. A well-defined problem maps out a fixed search space where optimum solution is always assumed while an ill-defined problem may not be able to construct a space due to the contradicting and ambiguous conditions. Problem solving can be categorized into search, exploration and co-evolution. A typical search process generates a solution as its output when the input is a well-defined problem. An exploration process takes in a problem and the solution is found by considering different foci, which lead to separate regions in the solution space to be searched. A co-evolutionary process derives a well-defined problem and the corresponding solution from an ill-defined problem. The view to problem solving as co-evolution does not only consider different regions in a solution space, the solution space is in fact changed because of the introduction of new variables.

3 MODELING PROBLEM SOLVING PROCESS

In this paper, a problem is specified as a set of requirements, which can be considered as a topological space. This is separate, and in addition, to the consideration of the solutions as another topological space. The focus of a problem solving is the way to measure the quality of the solution. For example, if the design of a floor plan focuses on maximizing the floor space, the quality of alternative solutions is measured by the amount of floor space each one provides. Therefore, the focus of a search can be considered as a metric space (Mendelson, 1963). The SGA is revisited in this section together with the introduction of two more evolutionary algorithms, i.e. ExpGA and CoGA. A notation is devised, based on the topological theory, to highlight the characteristics of these algorithms. The overview of each algorithm is followed by a brief outline of a typical implementation of the corresponding problem solving process the algorithm intends to model. There is no attempt to discuss and analyze these algorithms in details in this paper. They are referred in this section as computational models to problem solving process.

3.1 SIMPLE GENETIC ALGORITHM (SGA)

The efficiency and flexibility of biological systems is due to rules of self-repair, self-guidance and reproduction that hardly exist in most artificial systems. This metaphor from nature is modeled as a SGA and is shown below:

```
t = 0;
initialize 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);
```

Problem solving is a search when the focus of the problem does not change as the process continues. The requirement space and the solution space do not change during the process. Problem solving as search can be modeled as follows,

$$S_\rho = \text{best}_\rho\{S_1(\boldsymbol{P}), S_2(\boldsymbol{P}), S_3(\boldsymbol{P}), \ldots\}$$

where

$\boldsymbol{\rho}$	is f($\boldsymbol{P}$), a given focus of design,
$\boldsymbol{P}$	is the space of problem requirements,
$S_i(\boldsymbol{P})$	is the *i*th solution,
$\boldsymbol{S_\rho}$	is the best solution corresponding to the space of $\boldsymbol{P}$ based on the focus $\boldsymbol{\rho}$.

Example. The experiment on the structural optimization of truss design by Coello (1994) is a typical example in this class of problem. The goal of the optimization is to minimize the weight of a 10-bar truss structure (Figure 4) where the objective function is

$$f(x) = \sum_{i=1} \rho A_i L_i$$

where x is the candidate solution, A_i is the cross-sectional area of the *i*th member, L_i is the length of the *i*th member, and ρ is the weight density of the material. The truss is subject to the following set of constraints

$$\sigma_i \leq \sigma_a, \text{ for i= 1 to 10}$$
$$u_i \leq u_a$$

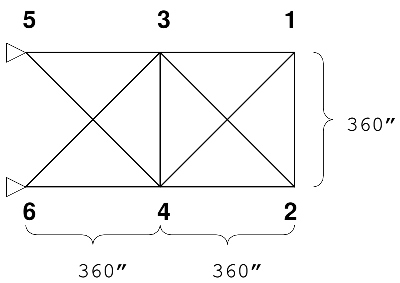

Figure 4 A 10-bar truss

where σ_i is the stress in member i, σ_a is the maximum allowable stress for all members, u_i is the displacement of each node (horizontal and vertical) and u_a is the maximum allowable displacement for all nodes. Since ρ and L are kept constant throughout the design, the values that need to be found are the cross-sectional area, which in turn is a set of 42 possible values that are defined by the American Institute of Steel Construction Manual. As a result, a chromosome of 60 bits long is constructed where each 6-bit represents the 42 available sections ($2^6 = 64$).

Although the intrinsic size of the search space is large, 42^{10} ($\approx 10^{16}$), the space is still finite and well-defined. The application of a SGA to this problem is to find the right cross-section for each of these 10 bars in the required configuration from a well bound space.

3.2 EXPLORATORY GA (ExpGA)

Exploration is a search where the focus changes as the process continues. A different focus, derived from the requirement space, can imply a different solution space. Exploration can be modeled as follows:

$$\begin{cases} S_{\text{final}} \in \{S_{\rho_1}, S_{\rho_2}, S_{\rho_3}, \ldots\} \\ S_{\rho_i} = \text{best}_{\rho_i}\{S_i\} = \text{best}\{S_{\rho_{i1}}(\boldsymbol{P}), S_{\rho_{i2}}(\boldsymbol{P}), S_{\rho_{i3}}(\boldsymbol{P}), \ldots\} \end{cases}$$

where

$\boldsymbol{P}$	is a space of problem requirements,
$\boldsymbol{\rho_i}$	is $f(P_i)$ a given focus of the problem at time i, $P_i \subseteq \boldsymbol{P}$,
$\boldsymbol{S_i}$	is the *i*th set of solution with the focus $\boldsymbol{\rho_i}$,
$S_{\rho_{ij}}(\boldsymbol{P})$	is the *j*th solution corresponding to the focus of the problem $\boldsymbol{\rho_i}$,
$\boldsymbol{S_{\rho_i}}$	is the best solution in S_i,
S_{final}	is the final selected solution.

Problem solving as exploration aims to search other solution spaces by changing the focus. Viewed from different angles and perspectives, different parts of a given solution space are considered or possibly different solution spaces become relevant. The difficulty in problem solving as exploration is to determine a way to change the focus of the problem, when the conventional evolutionary computation assumes a fixed fitness function.

Example. An example of problem solving using exploration is recorded in Maher and Poon and Boulanger (1996). The problem in this experiment is to find the appropriate configuration of a braced frame panel for a multi-story building. There are a few criteria to be considered which includes the conformity to the bay layout, the structural efficiency and structural integrity. However, instead of approaching the problem as a multi-criteria optimization, an ExpGA looks at these criteria separately and allows to change the focus of attention during the problem solving. In each step, one criterion is considered to be the most important and the other criteria have to pay the sacrifice. Since the computational resources are expended to find a solution for the current focus, the solution found at each cycle may be the best solution according to the current focus, but this same solution may be of poor quality if other criteria are considered at the same time. The ExpGA switches to another focus when the problem solving process proceeds.

3.3 CO-EVOLUTIONARY GA (CoGA)

Co-evolution allows the focus of the problem to change. The requirement space and the solution space can change through *mutual interaction*. The interaction between the two sets of spaces occurs through the focus of the search. The focus for the solution space is based on the requirements in the problem space and the focus for the problem space is based on the solutions in the solution space. The first interaction corresponds to the first downward arrow in Figure 5, from ***P*** to ***S***. If no satisfactory solution is found with the stated requirements, the solution space becomes the basis for the focus for searching the problem space. Hence, the second part of the first phase corresponds to a change in the fitness function when a solution space is given, i.e. the upward arrow from ***S*** to ***P***. After searching for relevant requirements in the problem space, another fitness function is derived from the new requirement space to be the focus of the search for a solution. The co-evolution model is described as follows:

$$
\begin{cases}
S_{final} \in \{S_{\rho_1}, S_{\rho_2}, S_{\rho_3}, \dots S_{\rho_n}\} \\
S_{\rho_i} = best_{\rho_i}\{S_i\} \\
\rho_i = f(P_i) \\
\\
P_{final} \in \{P_{\rho'_1}, P_{\rho'_2}, P_{\rho'_3}, \dots P_{\rho'_n}\} \\
P_{\rho'_i} = best_{\rho'_i}\{P_i\} \\
\rho'_i = f(S_i)
\end{cases}
$$

where

- $\boldsymbol{\rho_i}$ is a given focus for the solution at time i,
- $\boldsymbol{\rho'_i}$ is a given focus for the requirements at time i,
- ***P*** is a space of problem requirements,
- ***S*** is a space of solutions,
- $\boldsymbol{S_{\rho_i}}$ is the solution corresponding to the space of $\boldsymbol{P_i}$ with the given focus $\boldsymbol{\rho_i}$, $\boldsymbol{S_{\rho_i} \in S}$,
- $\boldsymbol{P_{\rho'_i}}$ is the problem requirements corresponding to the space of S_i with the given focus $\boldsymbol{\rho'_i}$, $\boldsymbol{P_{\rho_i} \in P}$.

The pseudo code for this co-evolutionary algorithm can be found at Figure 6.

Interactions in co-evolution may take different forms during the process. With co-evolutionary problem solving, the requirements and solutions interact with each other through two kind of interaction mechanisms: natural selection and niche construction. A set of requirements of P_{t1} is used to search for solutions S_{t1} via the mechanism of natural selection. The niche construction is founded on the mechanism of "addition". Some genetic materials from the solutions S_{t1} help generate the next requirements P_{t2}. The interaction cycles continue until the co-evolution terminates.

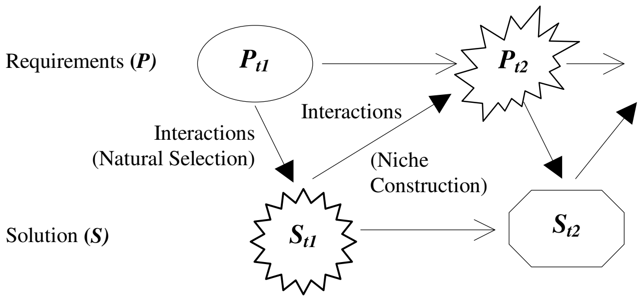

Figure 5 A model of co-evolution

```
T=0;
t=0;
initialize genotypes of requirements P_t and solutions S_t;
While termination conditions are not satisfied
{
  T=T+1;
  /* Phase 1 */
  ρ_T = f(P_t);
  Repeat
    t=t+1;
    S_t ::= select genotypes in S_t-1;
    reproduce, crossover and mutate genotypes in S_t;
    calculate fitness of phenotypes in S_t using ρ_T ;
  until convergence using DNA testing;
  check for termination

  /* Phase 2 */
  ρ_T' = f(S_t);
  Repeat
    t=t+1;
    P_t ::= select genotypes in P_t-1;
    reproduce, crossover and mutate genotypes in P_t;
    calculate fitness of phenotypes in P_t using ρ_T';
  Until convergence using DNA testing;
}
```

Figure 6 Pseudo code for a co-evolutionary algorithm

Convergence in a SGA means that the search process has led to the "best" solution in terms of the specified fitness function. Convergence is typically the criteria for termination of the evolutionary search process. Since the fitness function in co-evolutionary problem solving changes from one phase to another, the idea of convergence needs to be reconsidered. This requires a consideration of the purpose of co-evolutionary problem solving as compared to evolutionary search. The purpose of an evolutionary search is to find the best solution for a given environment, where the environment is effectively represented by the fitness function. In co-evolutionary problem solving, the purpose is to explore both the problem and solution spaces, allowing both to change in reaction to each other until a satisfactory combination of a problem statement and solution state is found. The exploratory nature of the co-evolutionary process implies that the process should continue until the potential for new ideas is reduced. Hence, in co-evolution, convergence is not related to fitness, but to the similarity of the members of the population. A population in which there is little change in the genotypes of the members when compared to the previous population indicates that the search process has converged.

Convergence is based on the comparison of the genotypes in one population to the genotypes in the previous generation. The idea of DNA testing is borrowed from biology to support this procedure. DNA testing involves the direct examination and comparison of the genetic material that a child inherited from its biological predecessors. If the DNA of the tested person does not contain sufficient genetic characteristics present in his/her predecessors, then the probability that this person is the true biological successor is calculated and reported.

In co-evolutionary problem solving, DNA refers to the genotype of a member of the population of the problem space and the solution space. Genotypes are broken down into fragments. The fragments are then separated according to the representation of the function, behaviour and structure, or some other relevant elements of representation, and pieces from the highly variable regions are "tagged". The DNA patterns that are formed as a result of the tagging provide identity profiles used in establishing the identity of the requirements and solutions.

To carry out the proposed DNA testing in co-evolution, a DNA bank is set up to store the DNA from the previous generations. Those DNA are the genetic characteristics of the functions, behavior and structures, or, for instance, the shape patterns. During the DNA testing process, the genetic characteristics of the requirements or solutions of the members of the population are compared to those stored in the DNA bank. If the genetic characteristics cannot be found in the DNA bank, those fragments are identified as new genetic characteristics and then are added to the DNA bank. Usually, these new characteristics come from the genetic changes of co-evolution, either from crossover and mutation or the interaction between the problem and solution spaces.

The link between convergence and termination in evolutionary algorithms occurs because the convergence to the "best" solution indicates that the search should be terminated. In co-evolutionary design, convergence is determined for each phase of the search, i.e. for a given focus. Following the convergence for one focus, another focus is determined and another search commences in the other space. This indicates a separation of termination and convergence. Termination is reserved to indicate when the co-evolutionary process should stop, and convergence relates to when the search in a given space for a given focus should stop.

One criterion for termination is the number of cycles of the co-evolution process. This criterion is equivalent to setting a time limit for the problem solving process. Often, the time limit is a major criterion for signaling when exploration of changes in problem and solution should stop.

Another criterion for termination is similar in concept to the DNA testing described above for convergence. A characterization of each phase in the co-evolution process is a change in focus, or fitness. With each change in phase, the fitness function is appended to a list of fitness functions, a process terminates when there are no new fitness functions added to the list. The significance of this criterion is that the algorithm is not able to identify a different focus and, therefore, new ideas have been exhausted.

Example. An example to this class of problem the crossword puzzle experiment that Maher and Wu (1998) reported. Traditional crossword puzzle is a grid-space which contains NxM cells where some of these cells are blackened out. A crossword designer creates an N×M crossword by constantly reconsidering and revision of the words and the layout. The goal of a crossword player is to fill in the blanks with some words according to the given hints.

The characteristics of this *open* crossword is that the crossword player is the crossword designer at the same time. The NxM grid-space does not contain any black squares and the goal is to enter interesting words without resorting into the N×M crossword. The problem that an *open* N×M crossword needs to resolve includes the selections of words and words layout instructions.

Most conventional search techniques are not suitable for creating crosswords problem because they usually expect the requirements of problem have been defined clearly in advance and does not change during the process. However, this problem has a very fluid boundary where the layout is allowed to change. Although it is inappropriate for SGA or ExpGA, the CoGA is a very good candidate to solving this problem.

The creation of an N×M crossword is modeled as problem with two evolving populations; words and layout instructions. The evolutions of words and layout instructions cooperated through interactions between them. A "good" N×M crossword can only be created when the selected words and the layout instructions satisfy each other.

Unlike a SGA, the genotype of a crossword employed in this experiment uses a combined gene approach which consists of two parts of gene: words part and layout instructions part (Figure 7). A phenotype of crossword is the physical characteristics of a crossword: what a crossword actually looks like.

The score of a crossword derived from a genotype may be improved because there are still a lot of unfilled spaces in the crossword. However, the problem is that the existing set of selected word may be not able to provide enough words to fill these blank cells. The interaction between the problem space and the solution space is the unique feature in a CoGA. Therefore, one way to improve the performance of a crossword with the current set of layout instructions is via the niche construction. The niche for this problem is to identify an extra set of words that can improve the actual performance of a genotype.

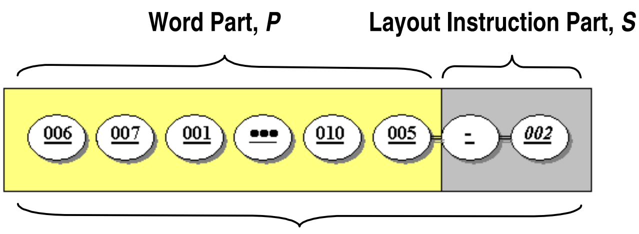

Figure 7 Genotype representation of the a NxM crossword puzzle

The termination of a co-evolutionary crosswording cannot rely on the values of actual fitness of the crosswords alone. Due to the incomplete nature of the *open* NxM crossword, the performance of a co-evolutionary crosswording has to consider the maximum fitness and actual fitness of a crossword at the same time. The maximum fitness presents the potential score that a crossword can achieve. And the actual fitness presents the behavior owing to the co-evolution. The form of *the* solution is vague but the best set of words and layout instructions to be identified should have the following characteristics

- Their actual fitness and the maximum fitness are as close as possible,
- The value of the crossword's actual fitness is the greatest.

Therefore, the termination conditions of a co-evolutionary crosswording depends on the fitness evaluation and the limitation of computation time.

3.4 DISCUSSION

Three evolutionary algorithms are introduced in this section. A SGA works on a fixed focus, while the problem space and solution space remain unchanged. There is no feedback from the solution space to influence the problem requirements. Solutions from the same space are evaluated by the same fitness function under the same focus.

The ExpGA is similar to SGA but this new algorithm recognizes the significance of shifting focus during problem solving. To consider one criteria as the focal point implies the algorithm prepares to sacrifice the quality of the other criteria for the sake of the focussed criteria. When a specific focus is chosen, the problem space, the solution space and the performance evaluation are frozen, the "best" solution under these temporary frozen conditions can be obtained using a SGA. However, once the solution under these frozen conditions is found, the information accumulated in the solution space provides feedback to the problem space which can steer the attention in the next exploration. From the above description, a SGA is subsumed in an ExpGA.

The CoGA differs from ExpGA by allowing the interactions between the problem space and the solution space. New genes may be added to the gene pool as a result of the interaction. The subsequent effect is that the problem space and solution space are altered, and the search space will be brought to a different dimension. The search that takes place after the interaction "stage" can be an ExpGA or a SGA. That is to say, SGA is a special case of ExpGA, which in turn is a subset of CoGA (Figure 8). Hence, to have a flexible problem solving environment, a GA package should include parameters to control interaction between spaces and to allow and the change of focus during the evolution (Table 1).

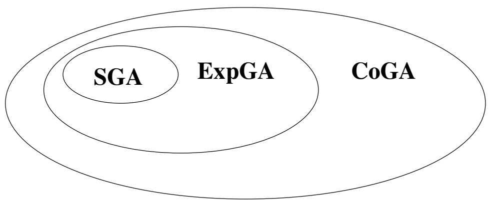

Figure 8 Relations among the evolutionary algorithms.

Table 1: Parameter settings for evolutionary algorithms

	Focus can change	Interaction allowed
SGA	no	no
ExpGA	yes	no
CoGA	yes	yes

It has been shown that the various evolutionary algorithms are useful in different kind of problem solving. The SGA is suitable to a well-defined problem while the CoGA is appropriate for ill-defined problem. To borrow what Langton (1992) has suggested for Artificial Life (refer to Section 1), the aim of using a SGA concerns the finding of a **solution-as-we-know-it,** while the goal for co-evolutionary algorithm is to find a **solution-as-it-could-be**. Using the view of co-evolution, problem solving becomes more interesting and creative.

4 CONCLUSIONS

This paper briefly reviewed the nature of problem solving process. A conventional SGA is found to be useful in well-defined problems, e.g. optimization. There are other types of problems that are beyond the limitations of a SGA, and they can only be solved using the two new algorithms (ExpGA and CoGA) introduced in this paper. The paper has also argued that a SGA is only a special case of the CoGA.

Acknowledgments

I gratefully acknowledge the useful discussion with Mary Lou Maher and Peter Wu from the Key Centre of Design Computing of the University of Sydney, Australia.

References

Coello, C. A. 1994. Discrete Optimization of Trusses using Genetic Algorithms. In Chen, J. G. and Attia, F. G and Crabtree, D. L. (eds). *EXPERSYS-94, Expert Systems Applications and Artificial Intelligence*, I.I.T.T. International Technology Transfer Series. Pages. 331-336.

Harvey, I. 1992. Species Adaptation Genetic Algorithms: A basis for a continuing SAGA. In F.J. Varela and P. Bourgine (eds.). *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, MIT Press/Bradford Books, Cambridge, MA, 1992, Pages 346-354.

Holland, J. H. 1975. *Adaptation in Natural and Artificial Systems*. University of Michigan Press.

Goldberg, D. E. 1989. Genetic Algorithms: In Search, Optimization and Machine Learning. Addison-Wesley.

Langley, P. and Simon, H. A. and Bradshaw, G. L. and Zytkow, J. M. 1987. *Scientific Discovery: Computational Explorations of the Creative Processes*. MIT Press.

Langton, C. G. 1992. Preface. In Langton, C. G., Taylor, C., Farmer, J. D., and Rasmussen, S. (editors). *Artificial Life II, Volume X of SFI Studies in the Sciences of Complexity*, Addison-Wesley, Redwood City, CA. Pages xiii-xviii.

Logan, B. and Smithers, T. 1993. Creativity and Design as Exploration. In Gero, J. S. and Maher, M. L (eds.). *Modelling Creativity and Knowledge-Based Creative Design*, Lawrence Erlbaum Associates, Pages 139-175.

Maher, M. L. and Poon, J. and Boulanger, S. 1996. Formalising Design Exploration as Co-evolution: A Combined Gene Approach. In Gero, J. S. (ed.). *Advances in Formal Design Methods for CAD*, Chapman and Hall, London. Pages 1-28.

Maher, M. L. and Wu, P. 1998. Fitness and Convergence in Coevolutionary Design. *Proceedings of the AI'98*, Australia.

Mendelson, B. 1963. *Introduction to Topology*. Blackie & Son Limited.

Simon, H. A. 1981. *The Sciences of the Artificial*, 2 edn, MIT Press.