Lecture 7 - Population diversity, Niching, Speciation & Co-Evolution

Yuhui Shi

CSE, SUSTech

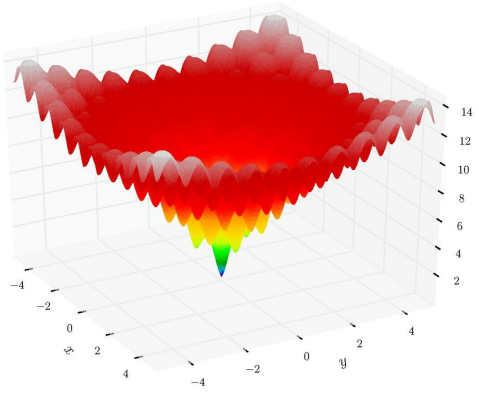
Outline of This Lecture

- Population diversity, Niching, Speciation
  - ✓Why Niching
  - ✓Different Niching Techniques
  - ✓Fitness Sharing
  - ✓Crowding and Speciation
- Co-Evolution

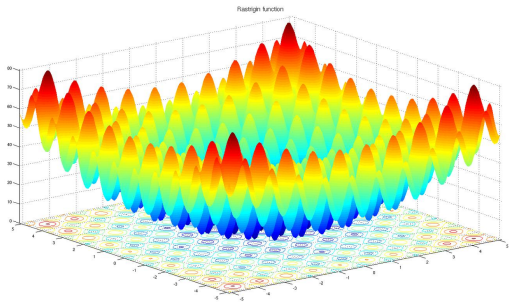
Outline of This Lecture

- **Population diversity, Niching, Speciation**
  - ✓Why Niching
  - ✓Different Niching Techniques
  - ✓Fitness Sharing
  - ✓Crowding and Speciation
- Co-Evolution

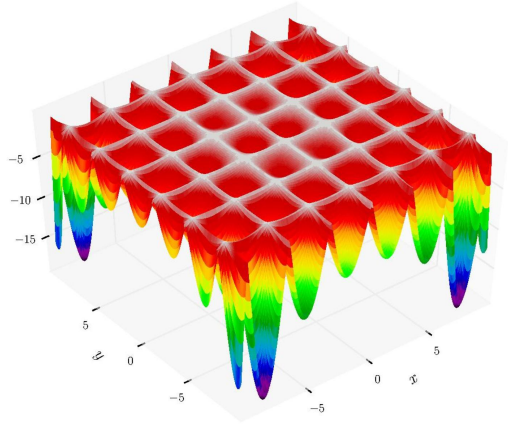
Multimodality (多峰形性)



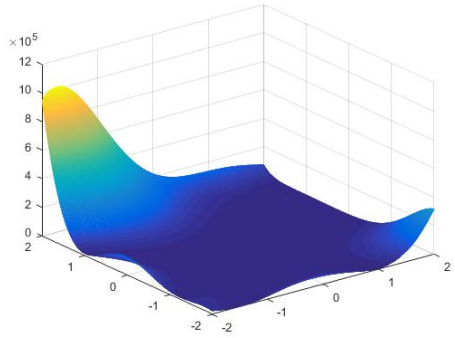
(a) Ackley Function



(b) Rastrigin Function



(c) Easom Function



(d) Goldstein-Price Function

Figure 1: Some multimodal benchmark functions.

Exploitation-Exploration Dilemma for EAs

- **Exploitation** of learned information by investigating regions containing high fitness solutions discovered.
- **Exploration** aiming to uncover new regions with high fitness.

Niching

- **What Is It?** It refers to the formation of groups of individuals in a population. Individuals within a group are similar to each other, while individuals from different groups are very different from each other.
- **Why Do We Need It?** Niching is useful in a number of cases.
  - ✓It helps to encourage and maintain population diversity, and thus explore a search space better.
  - ✓It helps to optimise multiple objectives simultaneously.
  - ✓It helps to learn an ensemble of machine learning systems that cooperate.
  - ✓It helps to simulate complex and adaptive systems, e.g., artificial ecological systems (人工生态系统).
  - ✓...

Different Niching Techniques

Major categories of niching techniques are:

- Sharing [1], also known as fitness sharing,
- Crowding, and
- Speciation.

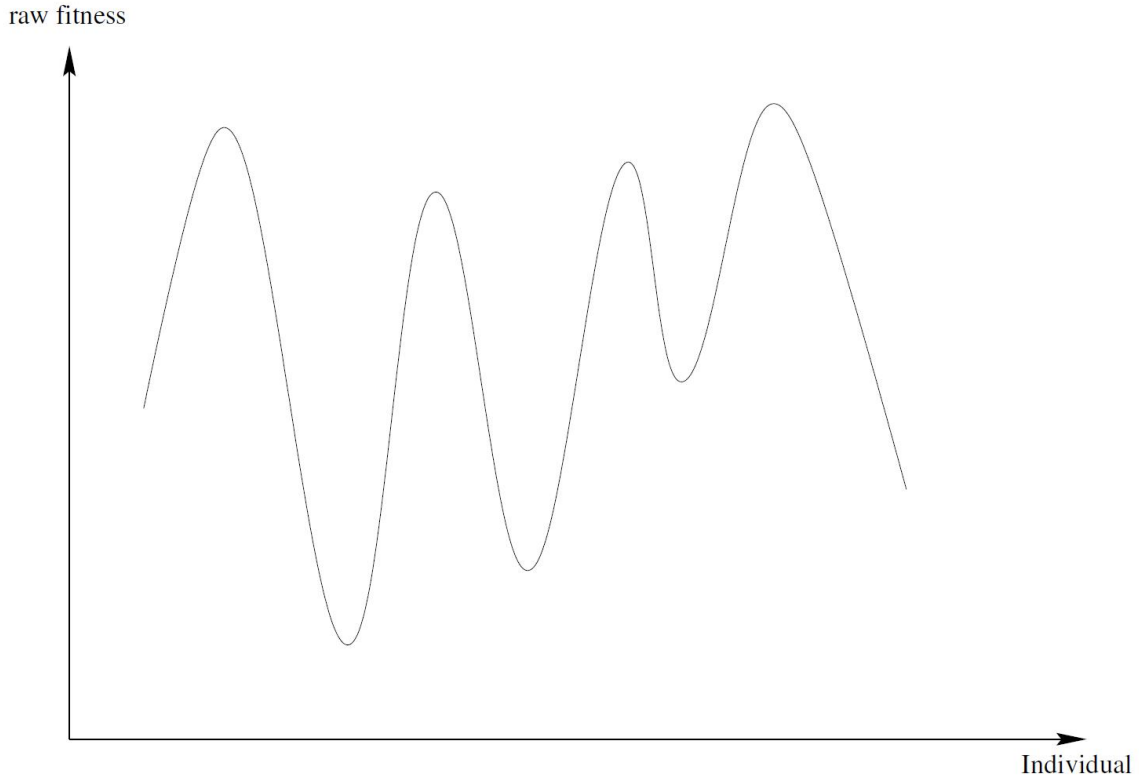
Other niching techniques include sequential niching and parallel hill-climbing.

(Explicit) Fitness Sharing: Introduction

- Fitness sharing transforms the raw fitness of an individual into the shared one (usually lower).
- The idea is that there is only limited and fixed amount of “resources” (i.e., fitness value) available at each niche. Individuals occupying the same niche (小生境) will have to share the resources.

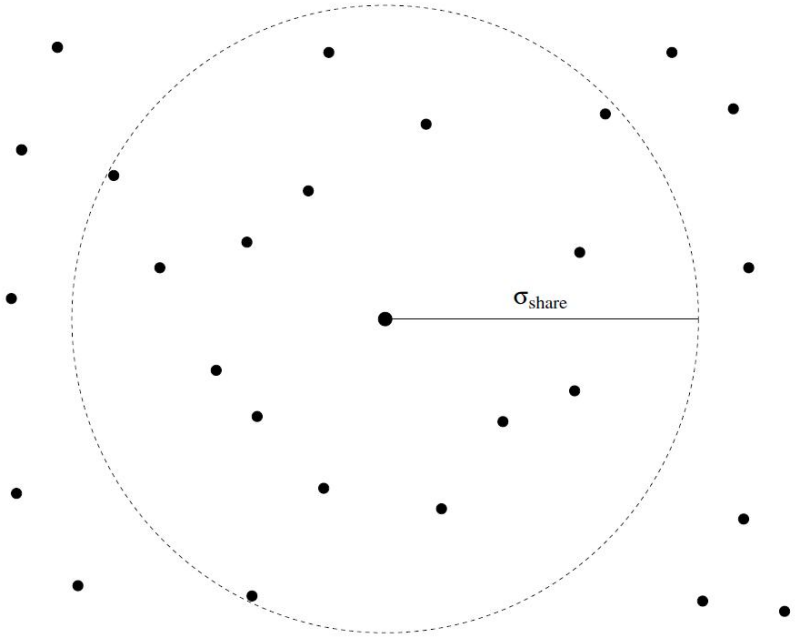
Fitness Sharing: An Example

- Sharing can be explained by a simple example of locating multiple peaks on a multimodal function in a single run.



Sharing Radius

- Sharing radius, σ_{share}, defines the niche size. Individuals within this radius will be regarded as being similar to each other and thus need to share fitness.
- The similarity between two individuals is defined by the distance between them.
  - ✓For example, the similarity between two binary strings can be defined by their Hamming distance. Thus, the "distance metric" used here is Hamming distance.



Sharing Function and Shared Fitness

- The shared fitness of individual *i* can be defined as

$$f_{share}(i) = \frac{f_{raw}(i)}{\sum_{j=1}^{\mu} sh(d_{i,j})},$$

where μ is the population size and $sh(d_{i,j})$ is defined as follows.

- The sharing function can be defined as

$$sh(d_{i,j}) = \begin{cases} 1 - \left(\frac{d_{i,j}}{\sigma_{share}}\right)^{\alpha}, & \textbf{if } d_{i,j} < \sigma_{share}, \\ 0, & \textbf{otherwise}, \end{cases}$$

niche count of individual i

where

- ✓ $d_{i,j}$ is the distance between individuals *i* and *j*.
- ✓ α determines the shape of sharing function.
  - ➢ The function is linear when $\alpha = 1$.
  - ➢ When increasing α, f_{share} reduces more rapidly with distance.
- ✓ σ_{share} is share radius, it decides
  - ➢ how many niches can be maintained in a population and
  - ➢ the granularity (粒度) with which different niches can be discriminated.

Fitness Sharing: Discussions I

1. Sharing can be done at genotypic or phenotypic levels. For example,

   Genotypic level: Hamming distance, i.e., the number of positions at which the corresponding chromosomes are different, $d_H(\mathbf{x}_1, \mathbf{x}_2) = \sum_{k=1}^{d} \mathbf{I}(x_1(k) \neq x_2(k))$**, where** $x_1(k)$ **refers to the** k**-th coordinate of** $\mathbf{x}_1$**. E.g.,** $d_H(1001, 0110) = 4$**.**

   Phenotypic level: Euclidean distance, $d_H(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{\sum_{k=1}^{d} (x_1(k) - x_2(k))^2}$**. E.g.,** $d_E(1001, 0111) = 2$**.**

   The key is how to define the “distance” (i.e., similarity).

2. Fitness sharing needs extra computation time.

Fitness Sharing: Discussions II

3. Sharing radius, σ_{share}, can be difficult to set. Why?
4. Population size is particularly important in sharing. Why?
5. A population may not be able to locate all peaks. It may lose peaks located during evolution. Why?
6. Fitness sharing as described above may not work well.Why?

Fitness Scaling in Sharing
-A Solution to Question 6

To make fitness sharing work, raw fitness scaling is often needed as follows:

$$f_{share}(i) = \frac{(f_{raw}(i))^{\beta}}{\sum_{j=1}^{\mu} sh(d_{i,j})},$$

where β > 1 is a scaling factor.

A Dilemma

- With a low scaling factor: Individuals won't converge to the real optima because they are not attractive.
- With a high scaling factor: “Super individuals” in initial populations may dominate the population quickly. The evolution may not be able to locate all peaks.

Solutions?

- Large population.
- Soft selection (note: fitness sharing is utilized to affect selection)
- Anneal β, i.e., staring from β = 1 and increasing it gradually.

  ← Dynamic β

Implicit Fitness Sharing: Background

- The idea comes from the immune system (免疫系统): antibody (抗体) which best matches an invading antigen (入侵性抗原) receives the payoff for that antigen.
- Similar situation appears in games: a strategy that scores the best against a test case receives payoff.
- Implicit fitness sharing has often been used in machine learning.

Implicit Fitness Sharing: Algorithm

- Assume we are dealing with a machine learning problem.
- For each case i to be solved, do the following C times:
  1. Select a sample of λ individuals from the population.
  2. Find the individual in the sample that achieves the best performance for solving test case i.
  3. This best individual (and this one only) receives the payoff. Ties are broken evenly, i.e., payoff will be shared evenly among all best individuals if they have the same best performance.

Implicit Fitness Sharing: Discussion

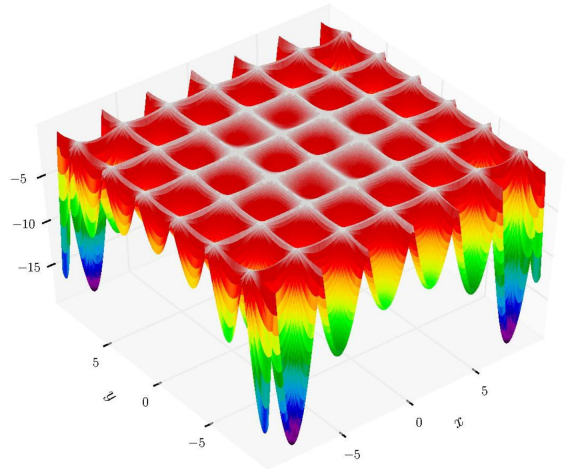
1. It has been shown that implicit fitness sharing and (explicit) fitness sharing have the same theoretical basis.
2. A larger C often leads to better sharing performance, but more time-consuming.

Comparison Between the Two Sharing Methods

1. Implicit fitness sharing covers optima more comprehensively even when those optima have small basins of attraction, if the population size is large enough for a species to form at each optimum.
2. Explicit fitness sharing can find the optima with larger basins of attraction and ignore the ones with smaller bases, if the population size is not large enough to cover all optima.

Question: Why?

Remark: The above can guide you which technique to use, given a population size.



Niching and Speciation

- We will use the two terms interchangeably although some people distinguish between them.
- Difference:
  - ✓Niching is concerned more with locating peaks (locating basins of attraction),
  - ✓while speciation is more focused on converging to the actual peaks.

What Is Crowding

- Fitness sharing encourages individual grouping, while crowding techniques strive to maintain the preexisting diversity of a population.
- Crowding techniques insert new individuals into the population by replacing similar individuals.
- Crowding techniques do not modify fitness.

Deterministic Crowding

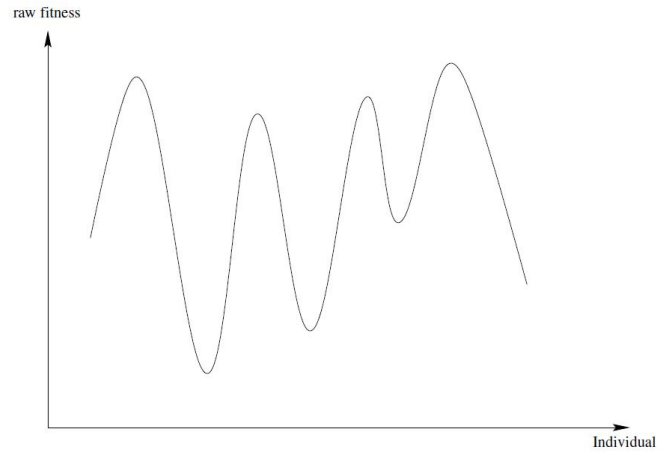
$P(0) \longleftarrow initialise()$;
FOR $t \longleftarrow 1$ **TO** g **DO**
    $P(t) \longleftarrow shuffle(P(t-1))$;
    FOR $i \longleftarrow 0$ **TO** $\mu/2 - 1$ **DO**
        $p_1 \longleftarrow a_{2i+1}(t)$; ***%select a parent***
        $p_2 \longleftarrow a_{2i+2}(t)$; ***%select a parent***
        $\{c_1, c_2\} \longleftarrow crossover(p_1, p_2)$;
        $c_1' \longleftarrow mutate(c_1)$;
        $c_2' \longleftarrow mutate(c_2)$;
        IF $[d(p_1, c_1') + d(p_2, c_2')] \leq [d(p_1, c_2') + d(p_2, c_1')]$ **THEN**
            IF $f(c_1') > f(p_1)$ **THEN** $a_{2i+1}(t) \longleftarrow c_1'$ **FI;**
            IF $f(c_2') > f(p_2)$ **THEN** $a_{2i+2}(t) \longleftarrow c_2'$ **FI;**
        ELSE
            IF $f(c_2') > f(p_1)$ **THEN** $a_{2i+1}(t) \longleftarrow c_2'$ **FI;**
            IF $f(c_1') > f(p_2)$ **THEN** $a_{2i+2}(t) \longleftarrow c_1'$ **FI;**

Discussions

- Capable of niching, i.e., locating and maintaining peaks.
- Minimal replacement error.
- Few parameters to tune.
- Fast because of no distance calculations over whole population.
- Population size must be large enough.
- Should use full crossover, i.e., crossover rate = 1.0.

Crowding Methods in General

- Unlike sharing methods, crowding methods do not allocate individuals proportional to peak fitness.
- Instead, the number of individuals congregating around a peak is largely determined by the size of that peak's basin of attraction under crossover.
- Similarity can be measured at either genotypic or phenotypic levels.



Speciation in a Narrow Sense

Many biologists believe that individuals in a sexually reproductive species can be created and maintained by allowing restrictive mating only among individuals from the same species

Speciation in a narrow sense focuses search within a peak. Speciation aims at finding the exact peak not just its basin.

- A speciation method restricts mating to similar individuals and discourages mating of individuals from different species.
- In order to apply such a speciation method, individuals representing each species must be found first. The speciation method cannot be used independently.

in niching, the main emphasis is devoted to distributing the population members across different peaks.

- Niching and speciation are complementary.
- Similarity can be measured at either genotypic or phenotypic levels.

A speciation method used in evolutionary computation studies, on the other hand, restricts mating to that among like solutions and discourages mating among solutions of different peaks.

Mating Restriction: Use Tags

Each individual consists of a tag and a functional string.

# 1 # 0	10010	1010 101
template	tag	functional string

- Tags participate in crossover and mutation, but not fitness evaluation.
- Templates can also be used.
- This method has been shown to be effective for multi-modal function optimisation.
- Only individuals with the same tag are allowed to mate.

Mating Restriction: Use Distance

- Define a threshold parameter, σ_{mate}.
- Two individuals are allowed to mate only when their distance is smaller than σ_{mate}.
- EAs with niching and mating restriction were found to distribute the population across the peaks better than those with sharing alone.

Mating restriction is always applied during crossover.

In order to choose a mate for an individual, their distance (in phenotypic mating restriction the Euclidean distance and in genotypic mating restriction the Hamming distance) is computed. If the distance is closer than a parameter σ_{mate}, they participate in the crossover operation; otherwise another individual is chosen at random and their distance is computed. This process is continued until a suitable mate is found or all population members are exhausted, in which case a random individual is chosen as a mate.

Fitness Sharing by Speciation

- Use tags to identify species (peaks).
- For a given problem, let k be the number of different tags. Let {S_0, S_1, . . . , S_{k-1}} be k species of individuals and ||.|| be the cardinality of the set. Then,

$$f_{share}(i) = \frac{f_{raw}(i)}{\|S_j\|}, \quad i \in S_j, \quad j = 0, 1, \cdots, k-1$$

- Recombination occurs only among individuals with the same tag.
- A tag can be mutated.
- No distance is used here.
- This is actually sharing plus mating restriction.

Summary So Far (I)

1. Niching techniques enable us to find multiple peaks simultaneously in evolution.
2. Every niching technique has its own “niche.” There is no single best niching method for all problems.
3. All fitness sharing techniques transform fitness values.
4. Population size becomes an important parameter when fitness sharing is used.

Summary So Far (II)

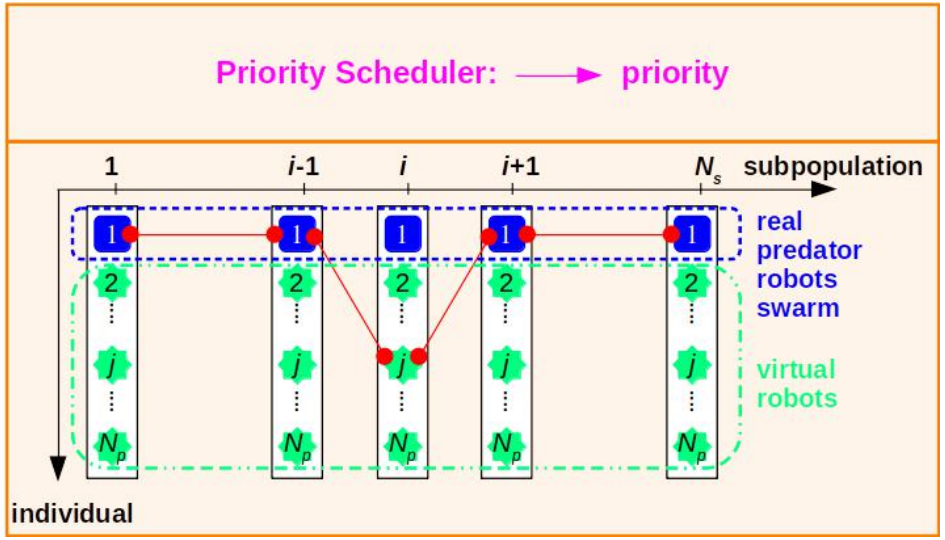
- Fitness Sharing modifies fitness (occurs right before selection operations).
  - ✓(explicit) fitness sharing
  - ✓implicit fitness sharing
  - ✓fitness sharing with mating restriction
- Crowding is about replacement strategies.
  - ✓deterministic crowding
- Speciation in a narrow sense occurs during recombination. It is all about mating restriction, which only affect crossover operations, not selction operations.
  - ✓by tags
  - ✓by distances

Outline of This Lecture

- Population diversity, Niching, Speciation
  - ✓Why Niching
  - ✓Different Niching Techniques
  - ✓Fitness Sharing
  - ✓Crowding and Speciation
- **Co-Evolution**

What Is Co-Evolution

- The fitness of one individual depends on other individuals.
  - ✓The fitness landscape is not fixed, but coupled.
  - ✓The same individual may have different fitness in different populations.
- Co-Evolution can be regarded as a kind of fitness landscape coupling where adaptive moves by one individual will deform landscapes of others, e.g., in prey-predator problems.



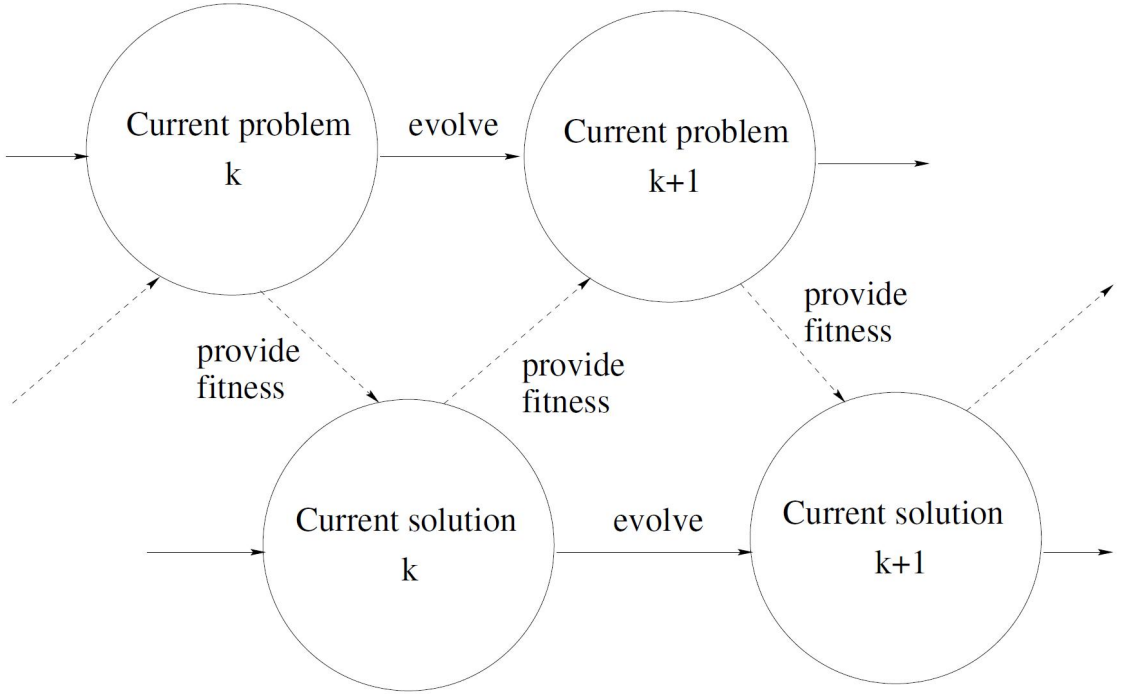
Different Types of Co-Evolution

- Based on the number of populations involved:
  - ✓Inter-population Co-Evolution has two or more populations.
  - ✓Intra-population Co-Evolution has only one population.
- Based on the relationship among individuals:
  - ✓Competitive Co-Evolution: Individuals are competing against each other.
  - ✓Co-operative Co-Evolution: Individuals cooperate with each other in performing a task.

Co-Evolution in Design

- Inter-population Co-Evolution

  Many design problems do not have a fixed specification because people change minds and design environments change.

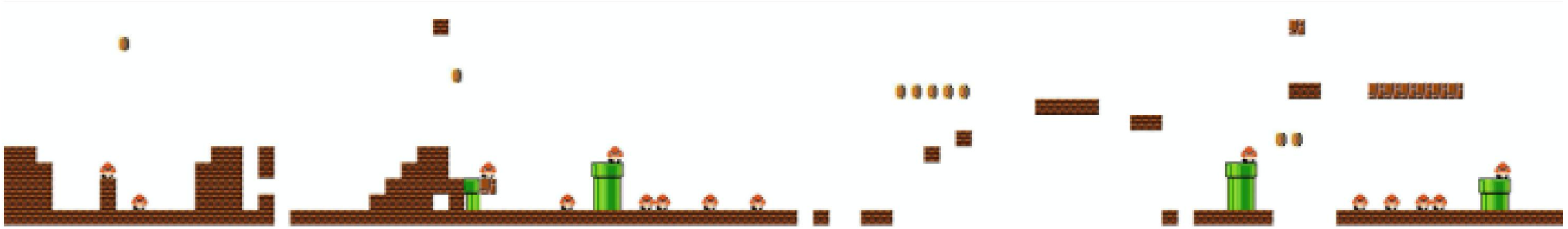


Co-evolving Programs and Test Cases

1. Given a software specification, one can co-evolve programs along with test cases/data.
2. Furthermore, one can use a similar idea to fix software bugs automatically using Co-Evolution.
3. The key issue is fitness evaluation.

Co-evolving Game and Agent

1. Given a game, one can co-evolve game level along with game-playing agent/human player.
2. The key issue is fitness evaluation.



Co-evolving Agent and Opponent Agent in 2P Games

- Considering a 2P adversarial game.
- Evolve two populations for both players at the same time.
- In each population, each individual is a game-playing agent.
- The fitness evaluation is often time consuming.

Intra-population Co-Evolution

- There is only one population involved.
- Famous examples: TD-Gammon and AlphaGo.
  - ✓A grand master level player.
  - ✓Learns by self-playing (i.e., Co-Evolution)

What make the player so strong: good machine learning algorithm or self-playing?

Figure 2: Backgammon. Image from https://tse2-mm.cn.bing.net/th/id/OIP.4P1l9g54tl2xwFAlsHpnTAHaE8?pid=ImgDet&rs=1

A Simple Experiment

Evolve a neural network that plays backgammon.

1. **Let the initial NN be NN_0, $k = 0$;**
2. **Generate a mutant challenger of NN_k:**

$$w'_{ij} = w_{ij} + Gaussian(0, \sigma).$$

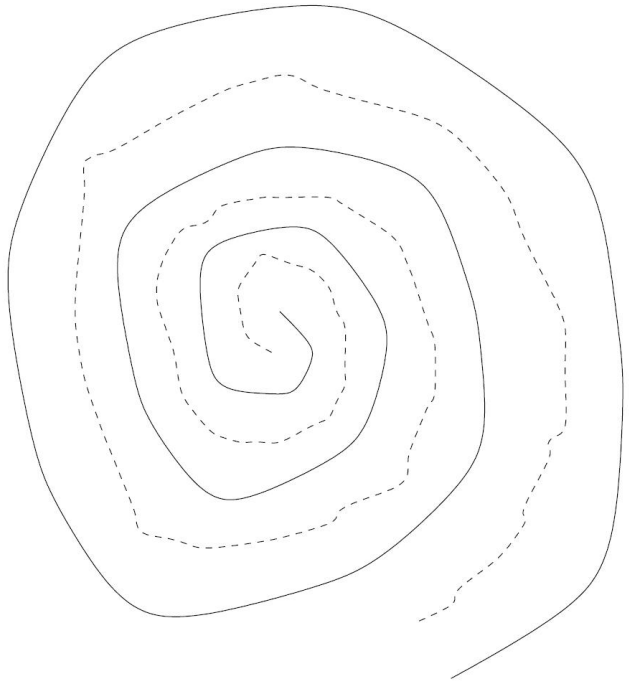
3. **If NN'_k is beaten by NN_k,**
   - **then $NN_{k+1} = NN_k$,**
   - **else $NN_{k+1} = 0.95 \times NN_k + 0.05 \times NN'_k$.**
4. **$k = k + 1$, go to Step 2 unless the halting criteria are satisfied.**

Comments

- Use 197-20-1 fully-connected feed-forward NN. Initial weights were 0's.
- No training for NNs!
- The EA had population size 1!
- There was only one simple mutation — adding Gaussian noise to weights.
- No recombination at all.

Performance: 40% winning against PUBEVAL after 100,000 generations. PUBEVAL is a strong programme trained by experts.

Separating Two Interwinded Spirals



Task: Given 194 training points, learn to separate two spirals.

The Two Spirals Problem

1. Very tough problem because of its highly nonlinear decision boundaries, especially for MLPs and decision trees.
2. Competitive Co-Evolution based on covering really helps in this case.
   - If no Co-Evolution is used, fitness of an NN is usually determined by the classification accuracy.
   - In Co-Evolution, fitness is evaluated based on pair-wise competition. It depends only on the number of training cases correctly classified by this individual but not by its opponent.

Iterated Prisoner's Dilemma Games

"non-cooperative" indicates that no preplay communication is permitted between the players

- Non-zero sum, non-cooperative games.

"non-zerosum" indicates that the benefits obtained by a player are not necessarily the same as the penalties received by another player

	Cooperate	Defect
Cooperate	R R	T S
Defect	S T	P P

T>R>P>S and R>(S+T)/2

Player A \ Player B	C	D
C	3 3	5 0
D	0 5	1 1

- How can an EA learn to play the game optimally starting from a population of random strategies?

Representation of Strategy [7]

To be more specific, consider the set of strategies that are deterministic and use the outcomes of the three (two, or one) previous moves to make a choice in the current move. Since there are four possible outcomes for each move, there are 4x4x4 (4x4 or 4) = 64 (16 or 4) different histories of the three (two or one) moves. Therefore, to determine its choice of cooperation or defection, a strategy would only need to determine what to do in each of the situation that could arise. This could be specified by a list of sixty- four (sixteen or 4) C's and D's.

For example a strategy with one previous move will be $a_1 a_2 a_3 a_4$.

CC	a_1
CD	a_2
DC	a_3
DD	a_4

Evolutionary Approach to IPD Games

- Use Co-Evolution: The fitness of an individual is determined by the payoff it obtains by playing against all other individuals in the population.
- Representation of strategies: Each strategy can be encoded as a binary string easily. One could also use NNs or payoff matrices.
- Evolutionary Algorithm: Initialised by random strategies.
- Results: Cooperation (the optimal strategy in this case) can be evolved easily.

Co-evolutionary Particle Swarm Optimization to solve constrained optimization problems (converting to min-max problems first) [9]

The constrained optimization problem is expressed as

$$\min_{x \in \Re^n} f(x)$$

subject to

$$g_i(x) \le 0, \quad i = 1,..., m$$
$$h_i(x) = 0, \quad i = 1,..., l$$

where the vector x consists of n variables:

$$x = [x_1, x_2, ..., x_n]^T \in \Re^n$$

The set $S \subseteq \Re^n$ designates the search space, which is defined by the lower and upper bounds of the variables: $x_{i\,l} \le x_i \le x_{i\,u}$ with $i = 1,...,n$.

Forming a max problem

By introducing the Lagrangian formulation, the dual problem associated with the constrained optimization problem can be written as

$$\max_{\lambda,\mu} L(x,\mu,\lambda)$$

subject to

$$\mu_i \geq 0, \qquad \text{for } i = 1,\ldots,m$$

with

$$L(x,\mu,\lambda) = f(x) + \mu^T g(x) + \lambda^T h(x)$$

and μ is a mx1 multiplier vector for the inequality constraints and λ is a lx1 multiplier for the equality constraints.

Forming min-max Problem
-*Part 1*

If the constrained optimization problem satisfies the convexity conditions over S, then the solution of the constrained optimization problem is the vector $x^*$of the saddle-point $\{x^*,\lambda^*,\mu^*\}$ of $L(x^*,\lambda^*,\mu^*)$ so that:

$$L(x^*,\mu,\lambda) \le L(x^*,\mu^*,\lambda^*) \le L(x,\mu^*,\lambda^*)$$

Solving the min-max problem

$$\min_{x} \max_{\lambda,\mu} L(x,\mu,\lambda)$$

provides the minimizer x^* as well as the multiplier λ^*,μ^* .

Forming min-max Problem
-Part 2

For non-convex problems, however, the solution of the dual problem does not coincide with that of the constrained optimization problem. In this case, the Lagrangian function is added a penalty function.

$$P(\boldsymbol{x}) = \frac{P}{2}\left(\sum_{i=1}^{m}(g^{2}{}_{i}^{+}(\boldsymbol{x}) + \sum_{i=1}^{l}(h_{i}^{2}(\boldsymbol{x}))\right)$$

where *P* is a positive penalty parameter, and

$$g_i^+(\boldsymbol{x}) = \max\left(0, g_i(\boldsymbol{x})\right), \quad \text{for} \quad i = 1,\ldots,m$$

The augmented Lagrangian is written then as

$$L_a(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\lambda}, p) = f(\boldsymbol{x}) + \boldsymbol{\mu}^T(\boldsymbol{x}) + \boldsymbol{\lambda}^T h(\boldsymbol{x}) + P(\boldsymbol{x})$$

Co-evolutionary PSO
-*Part 1*

- Two populations of PSOs are involved in the co-evolutionary PSO to solve the min-max problem.
- The first PSO focuses on evolving the variable vector x with “frozen” μ and λ.
- The second PSO focuses on evolving Lagrangian multiplier vectors μ and λ with “frozen” x.
- The two PSOs interact with each other through the fitness evaluation.

Co-evolutionary PSO
-*Part 2*

$$\uparrow L(x^*,\mu,\lambda) \le L(x^*,\mu^*,\lambda^*) \le L(x,\mu^*,\lambda^*) \downarrow$$

- For the first PSO, the problem is a minimum problem and the fitness value of each individual x is evaluated according to

$$f(x) = \max_{\mu,\lambda \in P_2} L(x,\mu,\lambda)$$

- For the second problem, the problem is a maximum problem and the fitness value of each individual μ and λ is evaluated according to

$$g(\mu,\lambda) = \min_{x \in P_1} L(x,\mu,\lambda)$$

When and Why Co-Evolution

1. We don't know the fitness function.
2. There are too many cases to test in order to obtain a fitness value. Co-Evolution can be used to focus search on the difficult part.
3. The fitness is inherently changing in time.
4. Increase and maintain diversity.
5. Improved robustness and fault-tolerance.

More Examples of Co-Evolution

- Computer-aided learning
- Robot morphology and controller
- Character recognition
- Checker, chess, Go, etc.
- many others.

Summary of Co-Evolution

1. Co-Evolution is everywhere. It can occur in a single or multiple populations. It can be used in many areas.
2. An individual's fitness is not fixed in Co-Evolution.
3. Co-Evolution is not well studied yet in evolutionary computation.
4. Artificial Co-Evolution and biological evolution.

Reading List for This Lecture

1. Bäck, Thomas, David B. Fogel, and Zbigniew Michalewicz (eds.), Handbook of evolutionary computation, 1997, IOP Publ. Co. & Oxford University Press, 1997. Section C6.1 and Section C6.2.
2. Darwen, Paul, and Xin Yao. “A dilemma for fitness sharing with a scaling function.” Proceedings of 1995 IEEE International Conference on Evolutionary Computation. Vol. 1. IEEE, 1995.
3. Darwen, Paul, and Xin Yao. “Every niching method has its niche: Fitness sharing and implicit sharing compared.” International Conference on Parallel Problem Solving from Nature. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996.
4. Pollack, Jordan B., Alan D. Blair, and Mark Land. "Coevolution of a backgammon player." Artificial Life V: Proc. of the Fifth Int. Workshop on the Synthesis and Simulation of Living Systems. Cambridge, MA: The MIT Press, 1997.
5. Juillé, Hugues, and Pollack, Jordan B. "Co-evolving intertwined spirals." Proc. of the 5th Annual Conference on Evolutionary Programming. Vol. 461. 1996.
6. Arcuri, Andrea, and Xin Yao. "Coevolving programs and unit tests from their specification." Proceedings of the 22nd IEEE/ACM International Conference on Automated Software Engineering. 2007.
7. Axelrod, Robert. “The evolution of strategies in the iterated prisoner's dilemma.” The dynamics of norms 1.1 (1987).
8. Sun, Lijun, Chao Lyu, and Yuhui Shi. "Cooperative coevolution of real predator robots and virtual robots in the pursuit domain." Applied Soft Computing 89 (2020): 106098.
9. Shi, Yuhui, and Renato A. Krohling. "Co-evolutionary particle swarm optimization to solve min-max problems." Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600). Vol. 2. IEEE, 2002.

Reading List for Next Lecture

1. X. Yao and P. J. Darwen, “An experimental study of n-person iterated prisoner's dilemma games,” in Progress in evolutionary computation, Springer, 1993, pp. 90–108.
2. J. H. Davis, P. R. Laughlin, and S. S. Komorita, “The social psychology of small groups: Cooperative and mixed-motive interaction,” Annual review of Psychology, vol. 27, no. 1, pp. 501–541, 1976.
3. A. M. Colman, Game theory and experimental games: The study of strategic interaction. Elsevier, 2016.
4. N. S. Glance and B. A. Huberman, “The dynamics of social dilemmas,” Scientific American, vol. 270, no. 3, pp. 76–81, 1994.
5. ——, ”The outbreak of cooperation,” Journal of Mathematical sociology, vol. 17, no. 4, pp. 281–302, 1993.
6. P. J. Darwen and X. Yao, “On evolving robust strategies for iterated prisoner's dilemma,” in Progress in Evolutionary Computation, Springer, 1993, pp. 276–292.
7. B. V. Gnedenko, Theory of probability. Routledge, 2017.
8. H. I. Jacobson, “The maximum variance of restricted unimodal distributions,” The annals of mathematical statistics, vol. 40, no. 5, pp. 1746–1752, 1969.
9. S. Y. Chong, P. Tino, and X. Yao, “Measuring generalization performance in coevolutionary learning,” IEEE Transactions on Evolutionary Computation, vol. 12, no. 4, pp. 479–505, 2008.