Niching and Speciation: Fitness Sharing

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.

Different Niching Techniques

Two major categories of niching techniques are:

1. Sharing, also known as fitness sharing, and
2. Crowding

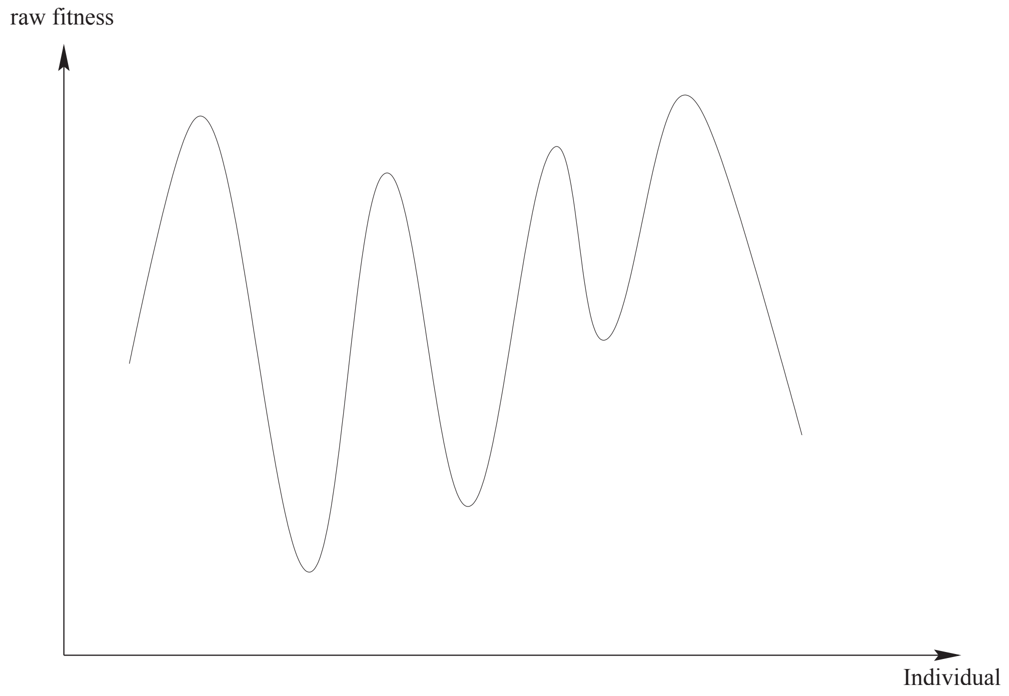
Other niching techniques include sequential niching and parallel hill-climbing.

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 will have to share the resources.

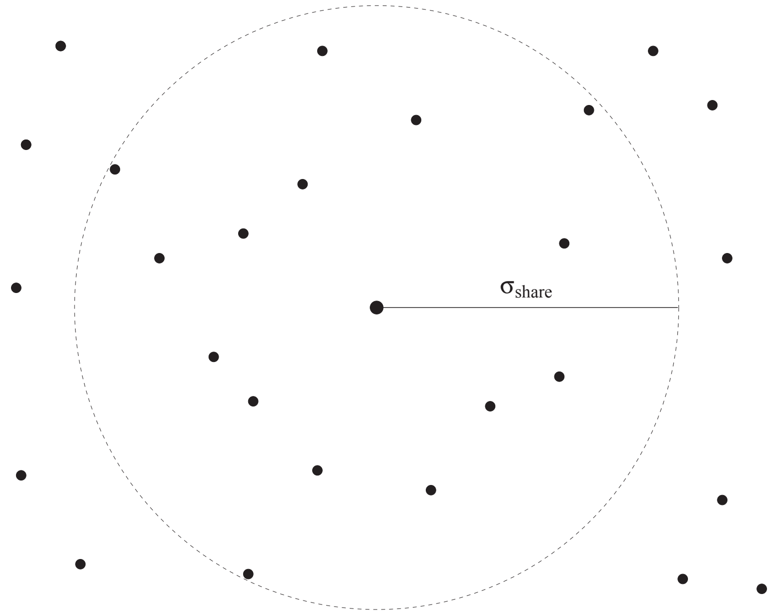
Fitness Sharing: An Example

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



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.

Sharing Function and Shared Fitness

1. The *sharing function* can be defined as

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

   where d_{ij} is the distance between individuals i and j.

2. The *shared fitness* of individual i can be defined as

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

   where μ is the population size.

Fitness Sharing: Discussions

1. Sharing can be done at genotypic or phenotypic levels. For example,
   - Genotypic level: Hamming distance
   - Phenotypic level: Euclidean distance

   The key is how to define the "distance" (i.e., similarity).
2. Sharing radius, σ_{share}, can be difficult to set. *Why?*
3. Population size is particularly important in sharing. *Why?*
4. A population may not be able to locate all peaks. It may lose peaks located. *Why?*
5. Fitness sharing needs extra computation time.
6. Fitness sharing as described above may not work well.

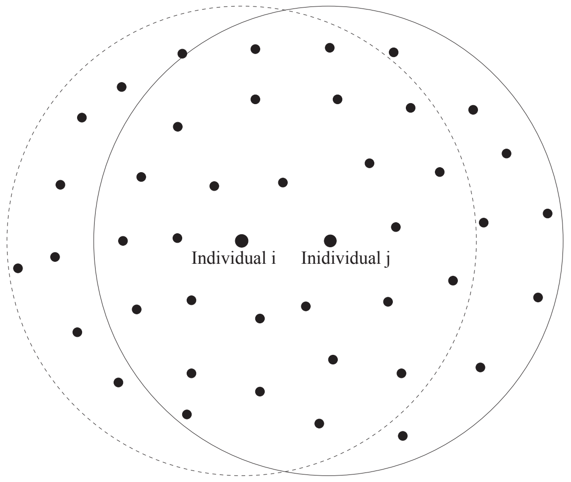
Fitness Scaling in Sharing

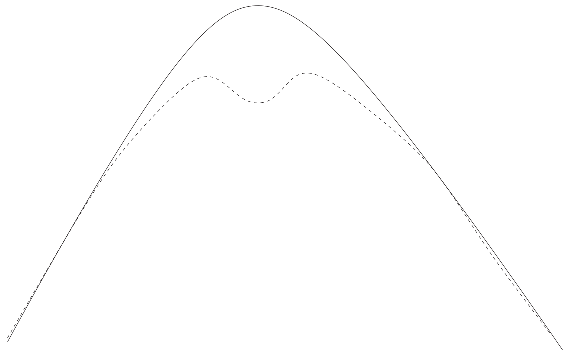
To make fitness sharing work, *raw* fitness scaling is often needed as follows:

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

where $\beta > 1$ is a scaling factor.

Why do we need it?





β needs to be sufficiently large. *Why?*

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
- anneal β, i.e., staring from $\beta = 1$ and increasing it gradually.

Implicit Fitness Sharing: Background

- The idea comes from the immune system: antibody which best matches an invading antigen 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 learning.

Implicit Fitness Sharing: Algorithm

Assume we are dealing with a machine learning problem.

For each *test case* i to be solved, do the following C times:

1. Select a sample of σ 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.*

Why?

Niching and Speciation

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

Summary

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.

References

1. T. Bäck, D. B. Fogel, and Z. Michalewicz (eds.), Handbook of Evolutionary Computation, IOP Publ. Co. & Oxford University Press, 1997. Section C6.1 and Section C6.2. (In the school library)
2. P. Darwen and X. Yao, "A dilemma for fitness sharing with a scaling function," *Proc. of 1995 IEEE Conference on Evolutionary Computation (ICEC'95)*, Dec. 1995, Perth, Australia, IEEE Press, pp.166–171.
3. P. Darwen and X. Yao, "Every niching method has its niche: fitness sharing and implicit sharing compared," *Lecture Notes in Computer Science, Vol.1141, Proc. of Parallel Problem Solving from Nature (PPSN) IV*, Springer-Verlag, Berlin, pp.398-407, 1996.