

Evolvability: What Is It and How Do We Get It?

4pm, Wed March 22, Th 391

Biological organisms exhibit spectacular adaptation to their environments. However, another marvel of biology lurks behind the adaptive traits that organisms exhibit over the course of their lifespans: it is hypothesized that biological organisms also exhibit adaptation to the evolutionary process itself. That is, biological organisms are thought to possess traits that facilitate evolution. The term evolvability was coined to describe this type of adaptation. The question of evolvability has special practical relevance to computer science researchers engaged in long-standing efforts to harness evolution as an algorithm for automated design. It is hoped that a more nuanced understanding of biological evolution will translate to more powerful digital evolution techniques. This talk will present a theoretical overview of evolvability, illustrated with examples from biology and evolutionary computing, and discuss computational experiments probing the relationship between environmental influence on the phenotype and evolvability.

An Honors Thesis Presentation ■ Light Refreshments Served



```
# print "Z", list(arz[i])
individuals.append(ind_init(self.parents[i] + self.sigmas[i] *
numpy.dot(self.A[i], arz[i])))
individuals[-1]._ps = "o", i

parents producing an offspring and chosen at random for the next
generation
else:
    ndom = tools.sortLogNondominated(self.parents, len(self.parents),
first_front_only=True)
    for i in range(self.lambda_):
        j = numpy.random.randint(0, len(ndom))
        _, p_idx = ndom[j]._ps
        individuals.append(ind_init(self.parents[p_idx] +
self.sigmas[p_idx] * numpy.dot(self.A[p_idx], arz[i])))
        individuals[-1]._ps = "o", p_idx

return individuals

select(self, candidates):
    if len(candidates) <= self.mu:
        return candidates, []

pareto_fronts = tools.sortLogNondominated(candidates, len(candidates))

chosen = list()
id_front = None
not_chosen = list()

Fill the next population (chosen) with the fronts until there is no
enough space
When an entire front does not fit in the space left we rely on the
hypervolume
for this front
The remaining fronts are explicitly not chosen
"""

for gen in xrange(ngen):
    population = toolbox.generate
    evaluate(population)
    toolbox.update(population)

.. [Collette2010] Collette, Y., M.
R. Le Riche (2010). On Object-
Examples in Scilab. In P. Brei
Multidisciplinary Design Optim
Wiley, pp. 527-565;

logbook = tools.Logbook()
logbook.header = ['gen', 'nevals']

for gen in xrange(ngen):
    # Generate a new population
    population = toolbox.generate
    # Evaluate the individuals
    fitnesses = toolbox.map(toolb
    for ind, fit in zip(population
    ind.fitness.values = fit

if halloffame is not None:
    halloffame.update(populat

# Update the strategy with th
toolbox.update(population)

record = stats.compile(popula
logbook.record(gen=gen, neval
if verbose:
    print logbook.stream

return population, logbook
```

Matthew Moreno

<https://mmore500.github.io>