

Evolutionary Programming

Chapter 5



EP quick overview

- Developed: USA in the 1960's
- Early names: D. Fogel
- Typically applied to:
 - traditional EP: machine learning tasks by finite state machines
 - contemporary EP: (numerical) optimization
- Attributed features:
 - very open framework: any representation and mutation op's OK
 - crossbred with ES (contemporary EP)
 - consequently: hard to say what "standard" EP is
- Special:
 - no recombination
 - self-adaptation of parameters standard (contemporary EP)

EP technical summary tableau

Representation	Real-valued vectors
Recombination	None
Mutation	Gaussian perturbation
Parent selection	Deterministic
Survivor selection	Probabilistic ($\mu+\mu$)
Specialty	Self-adaptation of mutation step sizes (in meta-EP)

Historical EP perspective

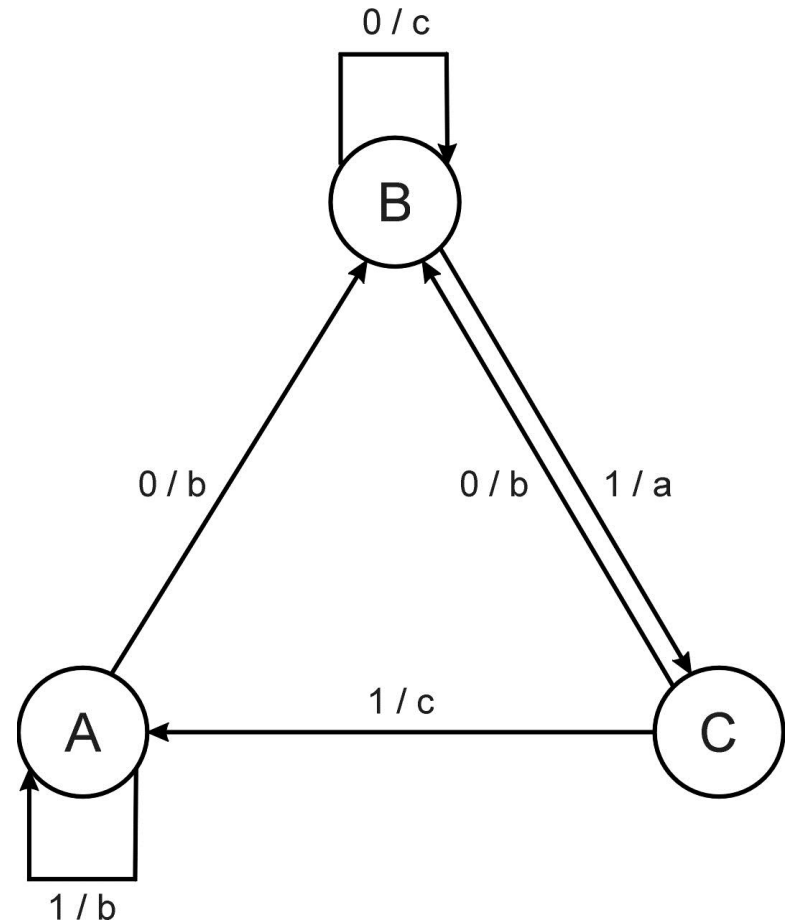
- EP aimed at achieving intelligence
- Intelligence was viewed as adaptive behaviour
- Prediction of the environment was considered a prerequisite to adaptive behaviour
- Thus: capability to predict is key to intelligence

Prediction by finite state machines

- Finite state machine (FSM):
 - States S
 - Inputs I
 - Outputs O
 - Transition function $\delta : S \times I \rightarrow S \times O$
 - Transforms input stream into output stream
- Can be used for predictions, e.g. to predict next input symbol in a sequence

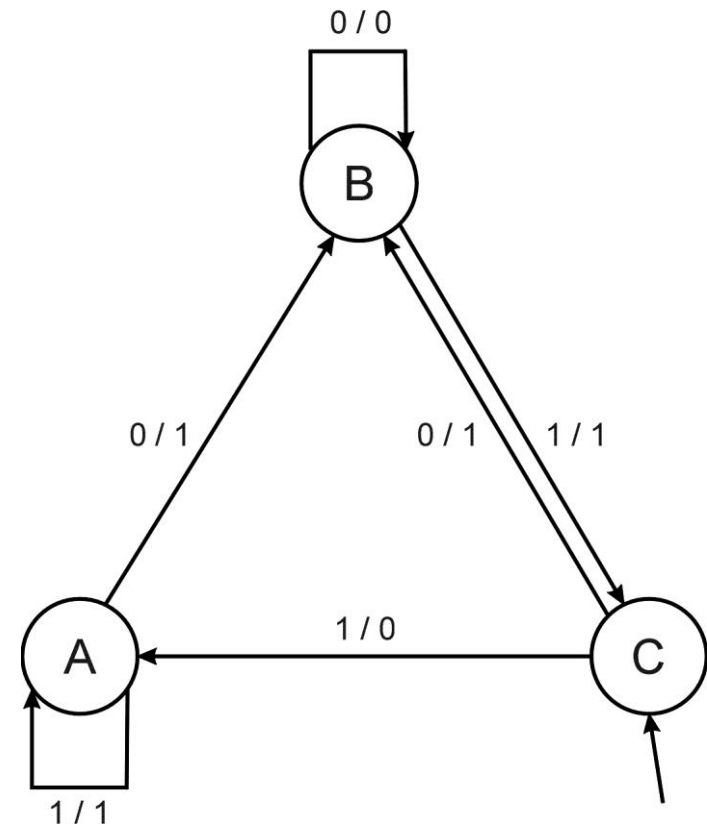
FSM example

- Consider the FSM with:
 - $S = \{A, B, C\}$
 - $I = \{0, 1\}$
 - $O = \{a, b, c\}$
 - δ given by a diagram



FSM as predictor

- Consider the following FSM
- Task: predict next input
- Quality: % of $in_{(i+1)} = out_i$
- Given initial state C
- Input sequence 011101
- Leads to output 110111
- Quality: 3 out of 5



Introductory example: evolving FSMs to predict primes

- $P(n) = 1$ if n is prime, 0 otherwise
- $I = N = \{1, 2, 3, \dots, n, \dots\}$
- $O = \{0, 1\}$
- Correct prediction: $out_i = P(in_{(i+1)})$
- Fitness function:
 - 1 point for correct prediction of next input
 - 0 point for incorrect prediction
 - Penalty for “too much” states

Introductory example: evolving FSMs to predict primes

- Parent selection: each FSM is mutated once
- Mutation operators (one selected randomly):
 - Change an output symbol
 - Change a state transition (i.e. redirect edge)
 - Add a state
 - Delete a state
 - Change the initial state
- Survivor selection: ($\mu+\mu$)
- Results: overfitting, after 202 inputs best FSM had one state and both outputs were 0, i.e., it always predicted “not prime”

Modern EP

- No predefined representation in general
- Thus: no predefined mutation (must match representation)
- Often applies self-adaptation of mutation parameters
- In the sequel we present *one EP variant*, not the canonical EP

Representation

- For continuous parameter optimisation
- Chromosomes consist of two parts:
 - Object variables: x_1, \dots, x_n
 - Mutation step sizes: $\sigma_1, \dots, \sigma_n$
- Full size: $\langle x_1, \dots, x_n, \sigma_1, \dots, \sigma_n \rangle$

Mutation

- Chromosomes: $\langle x_1, \dots, x_n, \sigma_1, \dots, \sigma_n \rangle$
- $\sigma'_i = \sigma_i \cdot (1 + \alpha \cdot N(0,1))$
- $x'_i = x_i + \sigma'_i \cdot N_i(0,1)$
- $\alpha \approx 0.2$
- boundary rule: $\sigma' < \varepsilon_0 \Rightarrow \sigma' = \varepsilon_0$
- Other variants proposed & tried:
 - Lognormal scheme as in ES
 - Using variance instead of standard deviation
 - Mutate σ -last
 - Other distributions, e.g. Cauchy instead of Gaussian

Recombination

- None
- Rationale: one point in the search space stands for a species, not for an individual and there can be no crossover between species
- Much historical debate “mutation vs. crossover”
- Pragmatic approach seems to prevail today

Parent selection

- Each individual creates one child by mutation
- Thus:
 - Deterministic
 - Not biased by fitness

Survivor selection

- $P(t)$: μ parents, $P'(t)$: μ offspring
- Pairwise competitions in round-robin format:
 - Each solution x from $P(t) \cup P'(t)$ is evaluated against q other randomly chosen solutions
 - For each comparison, a "win" is assigned if x is better than its opponent
 - The μ solutions with the greatest number of wins are retained to be parents of the next generation
- Parameter q allows tuning selection pressure
- Typically $q = 10$

Example application: the Ackley function (Bäck et al '93)

- The Ackley function (here used with $n = 30$):

$$f(x) = -20 \cdot \exp\left(-0.2 \sqrt{\frac{1}{n} \cdot \sum_{i=1}^n x_i^2}\right) - \exp\left(\frac{1}{n} \sum_{i=1}^n \cos(2\pi x_i)\right) + 20 + e$$

- Representation:
 - $-30 < x_i < 30$ (coincidence of 30's!)
 - 30 variances as step sizes
- Mutation with changing object variables first !
- Population size $\mu = 200$, selection with $q = 10$
- Termination : after 200000 fitness evaluations
- Results: average best solution is $1.4 \cdot 10^{-2}$

Example application: the Ackley function (Bäck et al '93)

- The Ackley function (here used with $n = 30$):

$$f(x) = -20 \cdot \exp\left(-0.2 \sqrt{\frac{1}{n} \cdot \sum_{i=1}^n x_i^2}\right) - \exp\left(\frac{1}{n} \sum_{i=1}^n \cos(2\pi x_i)\right) + 20 + e$$

- Representation:
 - $-30 < x_i < 30$ (coincidence of 30's!)
 - 30 variances as step sizes
- Mutation with changing object variables first !
- Population size $\mu = 200$, selection with $q = 10$
- Termination : after 200000 fitness evaluations
- Results: average best solution is $1.4 \cdot 10^{-2}$

Example application: evolving checkers players (Fogel'02)

- Neural nets for evaluating future values of moves are evolved
- NNs have fixed structure with 5046 weights, these are evolved + one weight for “kings”
- Representation:
 - vector of 5046 real numbers for object variables (weights)
 - vector of 5046 real numbers for σ 's
- Mutation:
 - Gaussian, lognormal scheme with σ -first
 - Plus special mechanism for the kings' weight
- Population size 15

Example application: evolving checkers players (Fogel'02)

- Tournament size $q = 5$
- Programs (with NN inside) play against other programs, no human trainer or hard-wired intelligence
- After 840 generation (6 months!) best strategy was tested against humans via Internet
- Program earned “expert class” ranking outperforming 99.61% of all rated players