EA overview

Based on Baeck- Schwefel 1993

- General outlines
- Evolution strategies
- Evolutionary programming
- Genetic algorithms
- Simulated annealing
- Road map of EC
- Why and when to use EC?

Components of an Evolutionary Algorithm

- $f : \mathbb{R}^n \to \mathbb{R}$ objective function to be optimized
- $\bar{x} \in \mathbb{R}^n$ an object variable vector
- I the space of individuals
- $a \in I$ an individual
- $\Phi : I \to \mathbb{R}$ the fitness function
- $\mu \geq 1$ size of the (parent) population
- $\lambda \geq 1$ number of offspring created in one cycle
- $P(t) = \{\bar{a}_1(1), \dots, \bar{a}_\mu(t)\}$ population at generation t
- $r_{\Theta_r} : I^\mu \to I^\lambda$ recombination operator
- $m_{\Theta_m} : I^\lambda \to I^\lambda$ mutation operator
- $s_{\Theta_s} : (I^\lambda \cup I^{\mu+\lambda}) \to I^\mu$ selection operator
- $\iota : I^\mu \to \{true, false\}$ termination criterion

Θ_r, Θ_m and Θ_s are control parameters of r, m and s respectively

Outline of an Evolutionary Algorithm

```
t := 0;
initialize   P(0) := {ā_1(0), … , ā_μ(0)} ∈ I^μ;
evaluate     P(0) :  {Φ(ā_1(0)), … , Φ(ā_μ(0))};

while (ι(P(t)) ≠ true)
{
        recombine:  P'(t)     := r_Θr(P(t));
        mutate:     P''(t)    := m_Θm(P'(t));
        evaluate    P''(t)    :  Φ(ā_1''(0)), … , Φ(ā_μ''(0))};
        select:     P(t + 1)  := s_Θs(P''(t) ∪ Q);
                    t := t + 1;
}
where Q ∈ {0, P(t)}
```

Evolution Strategies (1)

Representation (most general case)

$\bar{a} = (\bar{x}, \bar{\sigma}, \bar{\alpha})$, where

- $x_i, i \in \{1, \dots, n\}$, are object variables
- $\sigma_i, i \in \{1, \dots, n\}$, are the mutation stepsizes, that is the standard deviations $\sigma_i^2 = c_{ii}$
- $\alpha_i, j \in \{1, \dots, \frac{n \cdot (n-1)}{2}\}$, are rotation angles that is the covariances $\alpha_j \in \{c_{km} \mid k \in \{1, \dots, n-1\}, m \in \{k+1, \dots, n\}\}$

where c_{km} (k, m ∈ {1, … , n}) are the elements of the covariance matrix belonging to the generalized n-dimensional normal distribution with expectation vector 0

Evolution Strategies (2)

Fitness function: $\Phi(\bar{a}) = f(\bar{x})$

Mutation (most general case)

$$\sigma_i' = \sigma_i \cdot \exp(\tau \cdot N(0,1) + \tau \cdot N_i(0,1))$$
$$\alpha_j' = \alpha_j + \beta \cdot N_j(0,1)$$
$$\bar{x}' = \bar{x} + \bar{N}(\bar{0}, \bar{\sigma}', \bar{\alpha}')$$

Recombination (most general case)

$$x_i' = \begin{cases} x_{S,i} & \text{without recombination} \\ x_{S,i} \text{ or } x_{T,i} & \text{discrete recombination} \\ x_{S,i} + \chi \cdot (x_{T,i} - x_{S,i}) & \text{intermediate recombination} \\ x_{S_i,i} \text{ or } x_{T_i,i} & \text{global*, discrete} \\ x_{S_i,i} + \chi_i \cdot (x_{T_i,i} - x_{S_i,i}) & \text{global*, intermediate} \end{cases}$$

* S, T ∈ {1, … , μ} are redrawn for each i anew.

For σ's and α's the same mechanism

Evolution Strategies (3)

Selection:

Deterministic, selecting the μ best (1 ≤ μ < λ) out of

- the set of λ offspring individuals : (μ , λ)-selection
- the union of parents and offspring: (μ + λ)-selection

Evolution Strategies (4)

Outline of an Evolutionary Strategy:

```
t := 0;
initialize      $P(0) := P(0) := \{\bar{a}_1(0), \dots, \bar{a}_\mu(0)\} \in I^\mu$;
    where       $I = \mathbb{R}^{n+w}$
    and         $\bar{a}_k = (x_i, \sigma_i, \alpha_j)$
                $\forall i \in \{1, \dots, n\}, \forall j \in \{1, \dots, n \cdot (1 - n) / 2\}$;
evaluate        $P(0)$: $\{\Phi(\bar{a}_1(0)), \dots, \Phi(\bar{a}_\mu(0))\}$;
    where       $\Phi(\bar{a}_k(0)) = f(\bar{x}_k(0))$;
while ($\iota(P(t)) \neq$ true)
{
    recombine:  $\bar{a}_k'(t) := r'(P(t))$                        $\forall k \in \{1, \dots, \lambda\}$;
    mutate:     $\bar{a}_k''(t) := m_{\{\tau, \tau', \beta\}}'(\bar{a}_k'(t))$          $\forall k \in \{1, \dots, \lambda\}$;
    evaluate:   $P''(t) := \{\bar{a}_1''(t), \dots, \bar{a}_\lambda''(t)\}$ :   $\{\Phi(\bar{a}_1''(t)), \dots, \Phi(\bar{a}_\lambda''(t))\}$
        where   $\Phi(\bar{a}_k''(0)) = f(\bar{x}_k''(0))$;
    select:     $P(t + 1) :=$ if $(\mu, \lambda)$-selection     then $s_{(\mu, \lambda)}(P''(t))$;
                                                    else $(\mu + \lambda)(P(t) \cup P''(t))$;
    t := t + 1;
}
```

Evolutionary Programming (1)

Representation

- Standard EP: $\bar{a} = \bar{x}$
- Meta-EP: $\bar{a} = (\bar{x}, \bar{\sigma})$

Fitness function

scaling objective function values $f(\bar{x})$ to positive values and possibly imposing some random alteration

$\Phi(\bar{a}) = \delta(f(\bar{x}), \kappa)$

where

- $\alpha_i : \mathbb{R} \times S \to \mathbb{R}^+$ denotes the scaling function and S is an additional set of parameters,
- κ is some random alteration factor

Evolutionary Programming (2)

Mutation

- Standard EP: most common way is $x_i' = x_i + \sqrt{\Phi(\bar{x})} \cdot N_i(0,1)$
- Meta-EP: $x_i' = x_i + \sigma_i \cdot N_i(0,1)$
  $\sigma_i' = \sigma_i + \alpha \cdot \sigma_i \cdot N_i(0,1)$

where the parameter α ensures that v_i tends to remain positive

Recombination: None

Selection

stochastic q-tournament (μ + μ) style selection, sorting individuals by their score w, where

$$\boxed{w_i = \sum_{j=1}^{q} \begin{cases} 1 & \text{if } \Phi(\bar{a}_i) \le \Phi(\bar{a}_{\chi_j}) \\ 0 & \text{otherwise} \end{cases}}$$

where $\chi_j \in \{1, \dots, 2\mu\}$ is a uniform integer random variable, sampled anew for each comparison

Evolutionary Programming (3)

Outline of an Evolutionary Strategy:

```
t := 0;
initialize      $P(0) := P(0) := \{\bar{a}_1(0), \dots, \bar{a}_\mu(0)\} \in I^\mu$;
    where       $I = \mathbb{R}^n \times \mathbb{R}^{+n}$
    and         $\bar{a}_k = (x_i, v_i)$                         $\forall i \in \{1, \dots, n\}$;
evaluate        $P(0)$: $\{\Phi(\bar{a}_1(0)), \dots, \Phi(\bar{a}_\mu(0))\}$
    where       $\Phi(\bar{a}_k(0)) = \delta(f(\bar{x}_k(0)), \kappa_k)$;

while ($\iota(P(t)) \neq$ true)
{
    recombine:  $\bar{a}_k'(t) := m_{\{\alpha\}}'(\bar{a}_k(t))$               $\forall k \in \{1, \dots, \mu\}$;
    evaluate:   $P''(t) := \{\bar{a}_1'(t), \dots, \bar{a}_\mu'(t)\}$ :   $\{\Phi(\bar{a}_1'(t)), \dots, \Phi(\bar{a}_\mu'(t))\}$
        where   $\Phi(\bar{a}_k'(t)) = \delta(f(\bar{x}_k'(t)), \kappa_k)$
    select:     $P(t + 1) := s_{\{q\}}(P(t) \cup P'(t))$;

    t := t + 1;
}
```

Genetic Algorithms (1)

Representation

Bit strings of fixed length ℓ, i.e. $I = \{0, 1\}^\ell$.

For a continuous objective function:

$$f : \prod_{i=1}^{n} [u_i, v_i] \to \mathbb{R}, \text{ with } u_i < v_i$$

- $\ell = n \cdot \ell_x$,
- $\bar{a} = (a_{11} \dots a_{n\ell_x}) \in \{0,1\}^{n \cdot \ell_x} = I$ and
- a typical decoding is $\Gamma = \Gamma^1 \times \dots \times \Gamma^n$, where

$$\Gamma^i(a_{i1} \dots a_{i\ell_x}) = u_i + \frac{v_i - u_i}{2^{\ell_x}} \left(\sum_{j=1}^{\ell_x} a_{ij} 2^{j-1} \right)$$

Fitness function

$\Phi(\bar{a}) = \delta(f(\Gamma(\bar{a})))$,

Where δ is a scaling function assuring positive fitness values and best individual ~ largest fitness.

E.g. linear scaling with a scaling window of w generations.

$$\delta(f(\Gamma(\bar{a})), P(t - w)) = \max\{f(\Gamma(\bar{a}_j)) \mid \bar{a}_j \in P(t - w)\} - f(\Gamma(\bar{a}))$$

Genetic Algorithms (2)

Mutation

Bit flips with probability p_m per bit, that is

$$s_i' = \begin{cases} s_i, \chi_i > p_m \\ 1 - s_i, \chi_i \le p_m \end{cases}$$

where p_m is an external parameter, the mutation rate, χ_i is drawn from a uniform distribution

Recombination

One-point crossover: if $\bar{s} = (s_1, \dots, s_\ell), \bar{v} = (v_1, \dots, v_\ell)$ are selected as would-be parents, crossover takes place with probability p_c, called the crossover rate.

$$\bar{s}' = (s_1 \dots s_{\chi-1}, s_\chi, v_{\chi+1}, \dots, v_\ell)$$
$$\bar{v}' = (v_1 \dots v_{\chi-1}, v_\chi, s_{\chi+1}, \dots, s_\ell)$$

Where $\chi \in \{1, \dots, \ell - 1\}$ denotes a uniform random variable, the crossover point.

Genetic Algorithms (3)

Selection

Fitness proportional selection: probability of being selected is

$$p_s(\bar{a}_i) = \frac{\Phi(\bar{a}_i)}{\sum_{j=1}^{\mu} \Phi(\bar{a}_j)}$$

Genetic Algorithms (4)

Outline of an Genetic Algorithm:

```
t := 0;
initialize    P(0) := P(0) := {ā_1(0), … , ā_μ(0)} ∈ I^μ;
   where      I = {0,1}^ℓ
evaluate      P(0): {Φ(ā_1(0)), … , Φ(ā_μ(0))}
   where      Φ(ā_k(0)) = δ(f(Γ(ā_k(0))), P(0));

while (ι(P(t)) ≠ true)
{
   recombine:    ā_k'(t) := r_{pc}'(P(t))          ∀k ∈ {1, … , μ};
   mutate:       ā_k''(t) := m_{pm}'( ā_k'(t))     ∀k ∈ {1, … , μ};
   evaluate:     P''(t) := {ā_1''(t), … , ā_μ''(t)} :   {Φ(ā_1'(t)), … , Φ(ā_μ''(t))}
         where   Φ(ā_k''(t)) = δ(f(Γ(ā_k''(0))), P(t - w));
   select:       P (t + 1) := s(P''(t));
         where   p_s(ā_k''(t)) = Φ(ā_k''(t)) / Σ_{j=1}^{μ} Φ(ā_j''(t))

   t := t + 1;
}
```

Comparative overview of EA'S

	ES	EP	GA
Repr.	Real	Real	Binary
Self-adapt.	Standard dev's and covariances	Variances (in meta-EP)	None
Fitness	Objective function	Scaled obj. function	Scaled obj. function
Mutation	Main op.	Only op.	Background
Recomb.	Different, important for self-adaptation	None	Main op.
Select.	Deterministic, extinctive	Probab., extinctive	Probab., preservative

Simulated Annealing (1)

Outline of a Local Search procedure

```
procedure local-search:
{
   initialize(i_start);
   repeat
      generate(j ∈ S_i);
      if (f(j) < f(i)) then i := j;
   until (f(j) ≥ f(i)) for all j ∈ S_i;
}
```

Simulated Annealing (2)

Main drawback: gets easily stuck in local optima Possible cures:

1) several restarts with new i_{start}
2) sophisticated neighborhood structure
3) accept j even if it is worse than i

Option 3) by analogy from condensed matter physics, where state transitions lead to minimal energy level

$$P[\text{accept } j] = \exp\left(\frac{E_i - E_j}{K_b \cdot T}\right)$$

This is called the Metropolis criterion.

Simulated Annealing (3)

Outline of simulated annealing:

```
procedure simulated-annealing:
{
   initialize(i_start; c_0; L_0);
   k := 0; i := i_start;
   repeat
         for ℓ := 1 to L_k do {
               generate (j ∈ S_i);
               if (f(j) < f(i)) then i := j;
               else if $\exp\left(\frac{f(i) - f(j)}{c_k}\right) > random[0,1)$ then i := j;
         }
         k := k + 1;
         calculate-length(L_k);
         calculate-control(c_k);
   until stopcriterion;
}
```

Simulated Annealing (4)

Remarks:

1. The ‘temperature’ c_k is decreased over time.

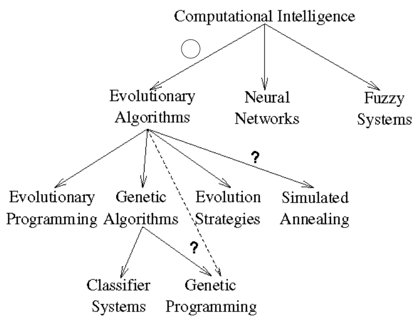
2. As c_k decreases, selection becomes more and more elitist.

3. The neighborhood S_i is mostly defined ‘operationally’, i.e. as the set of points generated with one modification (mutation) operator from i.

Simulated Annealing can be seen as an EA

- using a (1+1) selection strategy together with a
- a specific, time dependent selection regime
- where the representation and the mutation operator are not specified (left as problem dependent component)

Roadmap of EC



Why & when use evolutionary computing?

- **No in-depth mathematical understanding** of the problems needed
- Can **solve “out of range ” problems** (that cannot be solved by analytical mathematical techniques), for instance: many variables, many local optima, moving goal posts
- They are **extremely robust**; they cope well with noisy, inaccurate and incomplete data
- They are relatively **cheap and quick to implement**
- They are **easily hybridised**; they combine very productively with other techniques such as greedy methods, heuristics, simulated annealing and neural networks

Why & when use evolutionary computing?

- Extremely **adaptable**; changed priorities can be incorporated simply by changing weightings in the fitness function
- They are **modular and therefore portable**; because the evolutionary mechanism is separate from the problem representation they can be transferred from problem to problem
- They provide an extremely open and flexible approach to design, **allowing** arbitrary **constraints**, simultaneous **multiple objectives** and the **mixing of continuous and discrete parameters**
- Unlike many other methods, when evolutionary algorithms are **implemented on parallel computers** they make very efficient use of the available power