Adaptive Systems

Ezequiel Di Paolo
Informatics

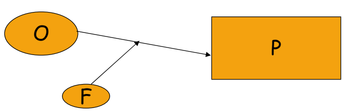
Lecture 10: Evolutionary Algorithms

Evolutionary computing

- Very loose, usually highly impoverished analogy between:
  - Data structures and genotypes,
  - Solutions and phenotypes
  - Operators and natural genetic transformation mechanisms (mutation, recombination, etc.)
  - Fitness mediated selection processes and natural selection.
- Closer to breeding than to *natural* selection
- Genetic Algorithms, Evolution Strategies, Genetic Programming, Evolutionary Programming …

Evolutionary computing

- Family of population-based stochastic direct search methods
- P[t] = O[t] x P[t-1]
- P is a population of data structures representing solutions to the problem at hand
- O is a set of transformation operators used to create new solutions F is a fitness function



Evolutionary computing

- **Is it magic?** *No.*
- **Is it universal?** *No. Very good for some problems, very bad for others.*
- **Is it easy to apply?** *Sometimes …*
- **Why should we be interested in EC?** *Can perform much better than other more standard techniques (not always). Good general framework within which to devise some powerful (problem specific) methods*
- **Uses:** Engineering Optimisation, combinatorial problems such as scheduling, Alife, Theoretical Biology
- **Article:** Genetic algorithms in optimisation and adaptation. P. Husbands (1992).

What used for?

Found to be very useful, **often in combination with other methods**, for:

- Complex multi-modal continuous variable function optimisation
- Many combinatorial optimization problems
- Mixed discrete-continuous optimisation problems
- Basics of artificial evolution
  - Design
  - Search spaces of unknown or variable dimensionality

Optimization and Search

- Classical deterministic techniques (often for problems with continuous variables)
  - Direct search methods (function evaluations only)
  - Gradient descent methods (making use of gradient information)
- Operate in a space of possible (complete or partial) solutions, jumping from one solution to the next
  - Evaluative
  - Heuristic
  - Stochastic

Direct search methods.

Used when:

- The function to be minimized is not differentiable, or is subject to random error;
- The derivatives of the function are discontinuous, or their evaluation is very expensive and/or complex;
- Insufficient time is available for more computationally costly gradient based methods;
- An approximate solution may be required at any stage of the optimization process (direct search methods work by iterative refinement of the solution).

No free lunch

- All algorithms that search for an extremum of a cost function perform exactly the same, according to any performance measure, when averaged over all possible cost functions. In particular, if algorithm A outperforms algorithm B on some cost functions, then, loosely speaking, there must exist exactly as many other functions where B outperforms A. **Number of evaluations must always be used for comparisons.**
- However, set of practically useful or interesting problems is, of course, a tiny fraction of the class of all possible problems ...

- D.H. Wolpert (1992). On the connection between in-sample testing and generalization error. Complex Systems, 6:47-94.
- D.H.Wolpert (1994). Off-training set error and a priori distinctions between learning algorithms. Tech. report, Santa Fe Institute.

Grid search

- Very simple adaptive grid search algorithm (x is an n-dimension vector, i.e. point in n-dimension space):
- a) Choose a point x1. Evaluate f(x) at x1 and all the points immediately surrounding it on a coarse n-dimensional grid.
- b) Let x2 be the point with the lowest value of f(x) from step a. If x2 = x1, reduce the grid spacing and repeat step (a), else repeat step (a) using x2 in place of x1.
- Problems: generally need very large numbers of function evaluations, you need a good idea of where minimum is.

Hill-climbing, local search

1. Generate initial solution
2. Current solution=initial solution
3. Generate entire neighbourhood of current solution
4. Find best point in neighbourhood. If best_point > current_soln,
5. Current_soln=best_point, goto 3, else STOP.

The *neighbourhood* of a point in the search space is the set of all points (solutions) one *move* away. Often infeasible to generate entire neighbourhood: Greedy local search (generate members of neighbourhood until find better soln than current), or stochastic sampling of neighbourhood.

Simulated annealing

- *Inspired by annealing (gradual cooling) of metals*
- 1) Initialize T (analogous to temperature), generate an initial solution, Sc, cost of this solution is Cc
- 2) Use an operator to randomly generate a new solution Sn from Sc, with cost of Cn
- 3) If (Cn-Cc) < 0 , i.e. better solution found, then Sc = Sn. Else if exp[-(Cn – Cc)/T] > random, then Sc = Sn, ie accept bad move with probability proportional to exp[-(Cn-Cc)/T].
- 4) If annealing schedule dictates, reduce T, eg linearly with iteration number
- 5) Unless stopping criteria met, goto step (2)

Potential strengths of EAs

- To some extent EAs attack problems from a global perspective, rather than a purely local one.
- Because they are population-based, if set up correctly, multiple areas of the search space can be explored in parallel.
- The stochastic elements in the algorithms mean that they are not necessarily forced to find the nearest local optimum (as is the case with all deterministic local search algs.)
- However, repeated random start local search can sometimes work just as well.

Hybrid algorithms

- Often best approach is to hybridize a 'global' stochastic method with a local 'classical' methods, (local search as part of evaluation process, in genetic operators, heuristics, pre-processing, etc.)
- Each time fitness is to be evaluated apply a local search algorithm to try and improve solution: take final score from this process as fitness. When new population is created, the genetic changes made by the local search algorithm are often retained (Lamarckianism).
- As above but only apply local search occasionally to fitter members of population.
- Embed the local search into the move operators -- e.g. heuristically guided search intensive mutations or Xover.

Encodings

- Direct encoding: vector of real numbers or integers $P_1\ P_2\ P_3\ P_4 \ldots\ldots P_N$
- Bit string sometimes appropriate, used to be very popular, not so much now. Gray coding sometimes used to combat uneven nature of mutations on bit strings.
- Problem specific complex encodings used including indirect mappings (genotype → phenotype).
- Mixed encodings: important to use appropriate mutation and crossover operators.
- Eg, 4 parameter options with symmetric relations, best to encode as 0, 1, 2, 3 than 00, 01, 10, 11.
- Use uniform range for real-valued genes (0,1) and map to appropriate parameter ranges after.

Crossover

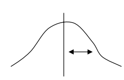
1 point

2-point

- Uniform: build child by moving left to right over parents, probability *p* that each gene comes from parent 1, *1-p* that it comes from parent 2 (p = 0.5).
- All manner of complicated problem specific Xover operators (some incorporating local search) have been used.
- Xover was once touted as the main powerhouse of GAs now clear this is often not the case. Building blocks hypothesis (fit blocks put together to build better and better individuals) also clearly not true in many cases…

Mutation

- Bit flip in binary strings
- Real mutation probability function in real-valued EAs

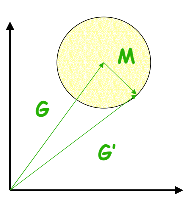


Probability of changing variable from current value

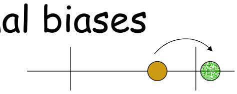
- All manner of problem specific mutations….
- Once thought of as low probability 'background' operator. Now often used as main, or sometimes only, operator with probability of operation of about one mutation per individual per generation.
- Prob of **no** mutation in offspring = $(1 - m)^{GL}$, with GL genotype length, m mutation rate *per locus*

Vector mutation

- Mutates the whole genotype. Used in real-value EAs
- Genotype **G** is a vector in an N-dimensional space.
- Mutate by adding a small vector **M** = *R* **m** in a random direction.
- Components of **m**: random numbers using a <u>Gaussian</u> distribution, then normalized. *R* is another Gaussian random number with mean zero and deviation *r* (strength of mutation). (Beer, Di Paolo)

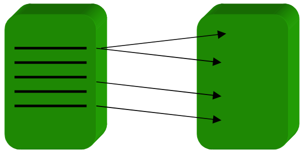


Mutational biases



- In real-valued EAs, if genes are bounded values there are many choices for mutations that fall out of bounds:
- Ignore 
- Boundary value 
- Reflection 
- Reflection is the less biased in practice <u>(try to work out why!)</u>

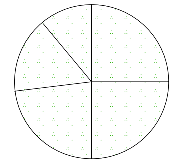
Selection: Breeding pool



- for each individual Rint[fi.N/Σfi] copies put into pool
- pick pairs at random from pool
- Rint: round to nearest integer; N: population size; fi fitness of ith individual

Selection: Roulette wheel

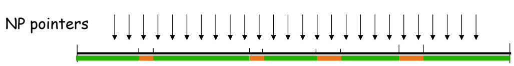
```
for(i=0;i<POPSIZE;i++)
    sum += fitness[i];

for(i=0;i<POPSIZE;i++){
        n=random(sum);    ← rand num 0-sum
        sum2=0;
        i=0;
        while(sum2<n){
          i=i+1;
          sum=sum2+fitness[i];
        }
        Select(i);
}
```



- Prob. of selection proportional to fi/Σfi. Subject to problems early loss of variabilty in population, oversampling of fittest members ...

Stochastic universal sampling

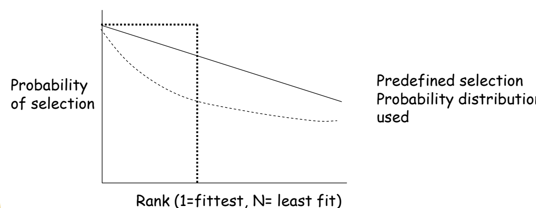
- Reduces bias and spread problems with standard roulette wheel selection.
- Individuals are mapped onto line segment [0,1]. Equally spaced pointers (1/NP apart) are placed over the line starting from a random position. NP individual selected in accordance with pointers.



First pointer is randomly positioned in range [0,1/NP]

Baker, J. E.: Reducing Bias and Inefficiency in the Selection Algorithm. in [ICGA2], pp. 14-21, 1987.

Rank based selection



Rank population according to fitness, then select following probability distribution.
Truncation is an extreme case.
Elitism -> fittest is selected with probability 1

Tournament selection

1. pick 2 members of population at random, Parent1 = fitter of these.
2. pick 2 members of population at random,
3. Parent2 = fitter of these

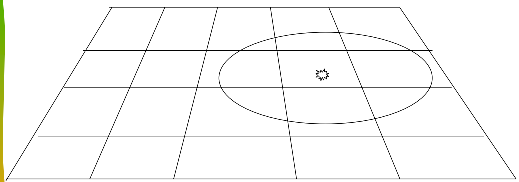
- Can have larger tournanemt sizes ...
- Microbial GA (Harvey): tournament based steady state, genetic tranference from winner to loser.

Steady state algorithms

- Population changed one at a time rather than whole generation at a time

1. Randomly generate initial population
2. Rank (order) population by fitness
3. Pick pair of parents using rank based selection
4. Breed to produce offspring
5. Insert offspring in correct position in (ordered) population (no repeats),
6. Push bottom member of population off into hell if offsping fitter
7. Goto 3 unless stopping criteria met

Geographically distributed EAs



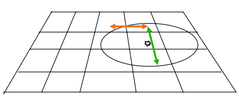
- 'Geographical' distribution of population over a 2D grid
- Local selection
- Asynchronous
- Good for parallelisation

Geographically distributed EAs

1. Create random genotypes at each cell on a 2D toroidal grid
2. Randomly pick cell on grid, C, this holds genotype Cg
3. Create a set of cells, S, in neighbourhood of C
4. Select (proportional to fitness) a genotype, m, from one of the cells in S
5. Create offspring, O, from m and Cg
6. Select (inversely proportional to fitness) a genotype, d, at one of the cells in S
7. Replace d with O.
8. Goto 2

<u>How to create neighborhood</u> (Repeat N Times, N: 5—8)

1) Choose Δx, Δy from Gaussian probability distribution, flip whether +/-direction

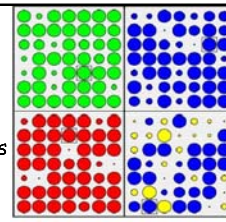


2) define sets of cells at distance 1,2,3 .. from current cell) … pick distance from Gaussian distribution, pick cell at this distance randomly

3) N random walks

4) Deterministic (e.g. 8 nearest neighbours)

Distributed EAs



Vaughan, 2003

- Fairly easy to tune.
- Robust to parameter settings
- Reliable (very low variance in solution quality)
- Find good solutions fast
- Tend to outperform simpler EAs
- Island model: Similar idea but divide grid into areas with restricted migration
- Whitley, D., Rana, S. and Heckendorn, R.B. 1999 *The Island Model Genetic Algorithm: On Reparability, Population Size and Convergence. Journal of Computing and Information Technology*, 7, 33-47.

Evolution of 3D objects using superquadric-based shape description language

- Shape description language is based on combinations (via Boolean operators) of superquadric shape primitives
- The individual primitives can also undergo such global deformations as twisting and stretching
- Shape description (genotypes) are easily genetically manipulated
- Genotypes translated to another format for polygonization and viewing
- Survival of the most interesting looking
- Husbands, Jermy et al. Two applications of genetic algorithms to component design. In *Evolutionary Computing* T. Fogarty (ed.), 50-61, Springer-Verlag, LNCS vol. 1143, 1996.

Superquadrics

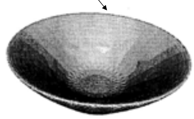
$$G(r) = \left(\left(\left(\frac{x}{a1}\right)^{\frac{2}{e2}} + \left(\frac{y}{a2}\right)^{\frac{2}{e2}} \right)^{\frac{e2}{e1}} + \left(\frac{z}{a3}\right)^{\frac{2}{e2}} \right)^{\frac{e1}{2}} - 1$$

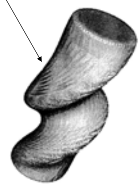
r is a point in 3D space, a1,a2,a3 are scaling parameters; e1,e2 are scaling parameters controlling how round, square or pinched the shape is. G(r) is an inside/outside function. G(r) < 0 => point inside the 3D surface, >0 => outside the surface and =0 => on the surface.

- Very wide range of shapes generated by small numbers of parameters.

Operators

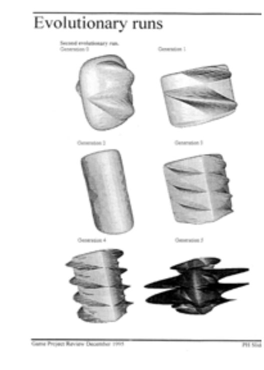
- Boolean operators: UNION, INTERSECT, DIFFERENCE
- Global deformations: translation, rotation, scaling, reflection, tapering, twisting, bending, cavity deformation

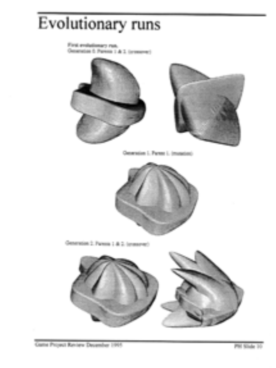




Genetic encoding

- The encoding is an array of nodes making up a directed network
- Each node has several items of information stored within it
- The directed network is translated into a shape description expression
- The network is traversed recursively, each node has a (genetically set) maximum recursive count. This allows repeated structures without infinite loops.





Other topics…

- Some to be covered in future lectures on evolutionary robotics:
- Co-evolutionary optimization
- Multi-objective problems
- Noisy evaluations
- Neutrality/evolvability