Evolutionary Computation

Dirk Thierens

Universiteit Utrecht
The Netherlands

Course organization

- Part 1: lectures ⇒ Dirk Thierens + 2 guest lectures Peter Bosman
- Part 2: seminar ⇒ papers & presentation (student groups)
- Part 3: practical assignment ⇒ report (groups of 1 or 2 students)
  - Specific discrete benchmark functions
  - Graph Bipartitioning

Course grading

1. Written exam = 40%
2. Paper presentation = 20%
3. Practical assignment report = 40%

Evolutionary Computation

= Population-based, stochastic search algorithms inspired by mechanisms of natural evolution

- EC part of Computational Intelligence
- Evolution viewed as search algorithm
- Natural evolution only used as metaphor for designing computational problem solving systems
- No modelling of natural evolution (≠ evolutionary biology)

Evolutionary algorithm

1. `P(0) ← Generate-Random-Population()`
2. `P(0) ← Evaluate-Population(P(0))`
3. **while** `Not-Terminated?` **do**
    1. $P^s(t) \leftarrow$ `Select-Mates(P(t))`
    2. $P^o(t) \leftarrow$ `Generate-Offspring(`$P^s(t)$`)`
    3. $P^o(t) \leftarrow$ `Evaluate-Population(`$P^o(t)$`)`
    4. $P(t+1) \leftarrow$ `Select-Fittest(`$P^o(t) \cup P(t)$`)`
    5. $t \leftarrow t + 1$
4. **return** `P(t)`

Key concepts of a Darwinian system

1. Information Structures
2. Copies
3. Variation
4. Inheritance
5. Competition

Darwinian process characteristics ⇒ GA

1. Structures
   ⇒ **e.g. binary strings, real-valued vectors, programs, ...**
2. Structures are copied
   ⇒ **selection algorithm**
3. Copies partially vary from the original
   ⇒ **mutation & crossover operators**
4. Structures are competing for a limited resource
   ⇒ **selecting fixed sized parent pool**
5. Reproductive success depends on environment
   ⇒ **user defined fitness function**

- **Neo-Darwinism**

  organism
  ⇑
  ...AUUCGCCAAU...

- **Genetic Algorithm**

  f: $\Re$
  ⇑
  ...0101001111...

* user: string representation and function **f**
* GA: string manipulation
    - **selection:** copy better strings
    - **variation:** generate new strings

- **selection:** copy better strings
  - tournament selection
  - truncation selection
  - proportionate selection
- **variation:** generate new strings
  1. crossover

     2-point crossover: $\begin{cases} 1111111111 \\ 0000000000 \end{cases} \Rightarrow \begin{cases} 1111000011 \\ 0000111100 \end{cases}$

     uniform crossover: $\begin{cases} 1111111111 \\ 0000000000 \end{cases} \Rightarrow \begin{cases} 1001110101 \\ 0110001010 \end{cases}$

  2. mutation

     $\{1111111111 \Rightarrow \{1111111011$

Toy example

$$x \,\epsilon\, [0,31] \;:\; f(x) = x^2$$
binary integer representation: $x_i \,\epsilon\, \{0,1\}$
$$x = x_1 * 2^4 + x_2 * 2^3 + x_3 * 2^2 + x_4 * 2^1 + x_5 * 2^0$$

- Initial Random Population:

  10010 : $18^2 = 324$
  01100 : $12^2 = 144$
  01001 : $9^2 = 81$
  10100 : $20^2 = 400$
  01000 : $8^2 = 64$
  00111 : $7^2 = 49$

  population mean fitness $\bar{f}(0) = 177$

- Generation 1:
  tournament selection, 1-point crossover, mutation

Parents	Fitness	Offspring	Fitness
100!10	324	10100	400
101!00	400	10111	529
01!000	64	00010	4
10!010	324	10010	324
0110!0	144	11100	784
1010!0	400	10000	256

Parent population mean fitness $\bar{f}(1) = 383$

- Generation 3:

Parents	Fitness	Offspring	Fitness
1!1111	961	11110	900
1!1100	784	11011	729
110!00	576	11110	900
111!10	900	11101	841
1101!1	729	11111	961
1100!1	625	01001	81

Parent population mean fitness $\bar{f}(0) = 762$

Schema = similarity subset

$$eg.: 11\#\#0 = \{11000, 11010, 11100, 11110\}$$

gen.	1####	0####	####1	####0
0	2	4	2	4
1	5	1	1	5
2	6	0	2	4
3	6	0	3	3
4	6	0	3	3
5	5	1	4	2

Schema

- definitions:
  - o(h): schema order $o(11\#\#0) = 3$
  - δ(h): schema defining length $\delta(11\#\#0) = 4$
  - m(h,t): number of schema h instances at generation t
  - $f(h,t) = \overline{\sum_{i \in P} f_i}$: schema fitness is average fitness of individual members
- key issue: **changing number of schemata members in population**
- fit schemata increase in proportion
- mutation and recombination destructive operators !

Schema growth by selection

- Reproduction ratio $\phi(h,t)$

$$\boxed{\phi(h,t) = \frac{m(h,t^s)}{m(h,t)}}$$

- **proportionate selection**
  - probability individual i selected: $\frac{f_i}{\bar{f}(t)}$
  - Expected number of copies that are member of schema h after selection:

$$m(h,t^s) = m(h,t)\phi(h,t) = m(h,t)\frac{f(h,t)}{\bar{f}(t)}$$

- **tournament selection**
  - tournament size s: $0 \leq \phi(h,t) \leq s$

Schema disruption by mutation

- probability bit flipped: p_m
- schema h survives iff all the bit values are *not* mutated

$$p_{survival} = (1 - p_m)^{o(h)}$$

- for small values $p_m << 1$

$$(1 - p_m)^{o(h)} \approx 1 - o(h).p_m$$

- disruption factor $\epsilon(h,t)$ by mutation:

$$\boxed{\epsilon(h,t) = o(h).p_m}$$

Schema disruption by recombination

- probability crossover applied p_c
- **1-point crossover**
  - schema h survives iff cutpoint *not* within defining length δ:

$$p_{survival} = 1 - \frac{\delta(h,t)}{l-1}$$

- **uniform crossover** (bit swap probability: p_x)
  - schema h survives iff none or all bits swapped together

$$p_{survival} = p_x^{o(h)} + (1-p_x)^{o(h)}$$

- disruption factor $\epsilon(h,t)$ by recombination:

$$\boxed{\epsilon(h,t) = p_c.(1-p_{survival})}$$

Schema Theorem

- Selection, mutation, and recombination combined:

$$\boxed{m(h,t+1) \geq m(h,t)\phi(h,t)[1-\epsilon(h,t)]}$$

- net growth factor: $\gamma(h,t) = \frac{m(h,t+1)}{m(h,t)}$

$$\boxed{\gamma(h,t) \geq \phi(h,t)[1-\epsilon(h,t)]}$$

schemata with $\gamma(h,t) > 1$ increase in proportion
schemata with $\gamma(h,t) < 1$ decrease in proportion

Schema Theorem cont'd

- low order, high performance schemata receive exponentially (geometrically) increasing trials $\rightarrow$ **building blocks**
- according to the k-armed bandit analogy this strategy is near optimal (Holland, 1975)
- happens in an implicit parallel way

  $\rightarrow$ only the short, low-order schemata are processed reliably

- enough samples present for statistically reliable information
- enough samples survive the disruption of genetic operators

Building Blocks

Building block hypothesis

= *building blocks can be juxtaposed to form near optimal solutions*

Consequences

1. schema sampling is a statistical decision process: **variance considerations**
2. building blocks must be juxtaposed before convergence: **mixing analysis**
3. low order schemata might give misleading information: **deceptive problems**

Permutation problems

- **Goal**
  Design suitable representations and genetic operators for permutation or sequencing problems
- **Examples**
  - scheduling
  - vehicle routing
  - queueing
  - ...

Permutation problems

- travelling salesman
- non-binary strings
  - p1 = 1 2 3 4 5 6 7 8
  - p2 = 4 6 2 1 7 8 5 3
  - simple crossover $\Rightarrow$ illegal tours
  - c1 = $\underline{1}$ 2 $\underline{3}$ | $\underline{1}$ 7 8 5 $\underline{3}$
  - c2 = $\underline{4}$ $\underline{6}$ 2 | $\underline{4}$ 5 $\underline{6}$ 7 8
- alternative search space representation
- alternative genetic operators

Insert mutation

randomly select one element from the sequence and insert it at some other random position in the sequence

$$\begin{array}{c} A\ B\ \underline{C}\ D\ E\ F\ G\ H \\ \Downarrow \\ A\ B\ D\ E\ F\ \underline{C}\ G\ H \end{array}$$

Swap mutation

randomly select two elements from the sequence and swap their position

$$\begin{array}{c} A\ B\ \underline{C}\ D\ E\ F\ \underline{G}\ H \\ \Downarrow \\ A\ B\ \underline{G}\ D\ E\ F\ \underline{C}\ H \end{array}$$

Scramble mutation

randomly select a subsequence and scramble all elements in this subsequence

A B | C D E F | G H
⇓
A B | D F E C | G H

very destructive !

→ efficiency is problem dependent !

Mutation operator: 2-opt

⇒ randomly select two points along the sequence and invert one of the subsequences

A B | C D E F | G H
⇓
A B | F E D C | G H

2-opt can be applied to $\frac{n(n-1)}{2}$ pairs of egdes: if none of these gives an improvement a local optimum has been reached.

Mutation operators

- TSP: *adjacency* of elements in permutation is important
  → 2-opt only minimal change
- scheduling: *relative ordering* of elements in permutation is important
  → 2-opt large change
  e.g.: priority queue: line of people waiting for supply of tickets for different seats on different trains

mutation principle: “small” moves in search space should be more likely than “large” moves

Recombination operators

- ’standard’ crossover operators generate infeasible sequences

  A B C D E | F G H
  b f d h g | e a c
  ⇓
  A B C D E | e a c
  b f d h g | F G H

- different aspects
  - adjacency
  - relative order
  - absolute order

⇒ whole set of permutation crossover operators proposed !

Order crossover

p1: A B | C D E F | G H I
p2: h d | a e i c | f b g
⇓
ch: a i C D E F b g h

1. randomly select two crosspoints
2. copy subsequence between crosspoints from p1
3. starting at 2nd crosspoint: fill in missing elements retaining relative order from p2

Partially mapped crossover

p1: A B | C D E F | G H I
p2: h d | a e i c | f b g
⇓
ch: h i C D E F a b g

1. randomly select two crosspoints
2. copy p2 to child
3. copy elements between crosspoints from p1 to child while placing the replaced element from p2 at the location where the replacer is positioned

Position crossover

p1: A B C D E F G H I
p2: h d a e i c f b g
 * * * *
⇓
ch: A h C d E F b g I

1. randomly select k positions
2. copy unmarked elements from p1 to child
3. scan p2 from left to right and fill in missing elements

Maximal preservative crossover

p1: A B | C D E F | G H I
p2: h d | a e i c | f b g
⇓
ch: i a C D E F b g h

1. randomly select two crosspoints
2. copy subsequence between crosspoints from p1
3. add successively an adjacent element from p2 starting at last element in child
4. if already placed: take adjacent element from p1

Cycle crossover

p1: A B C D E F G H I
p2: f c d a e b h i g
cy: 1 1 1 1 2 1 3 3 3
⇓
ch: A B C D E F h i g

1. mark cycles
2. cross full cycles

⇒ emphasizes absolute position above adjacency or relative order

edge recombination

parent tours [ABCDEF] & [BDCAEF]

edge map:

city	*edges*
A	B F C E
B	A C D F
C	B D A
D	C E B
E	D F A
F	A E B

edge recombination algorithm:

1. choose initial city from one parent
2. remove current city from edge map
3. if current city has remaining edges
   goto step 4
   else
   goto step 5
4. choose current city edge with fewest remaining edges
5. if still remaining cities, choose one with fewest remaining cities

1. random choice ⇒ B
2. next candidates: A C D F
   choose from C D F (same edge number) ⇒ C
3. next candidates: A D
   (edgelist D < edgelist A) ⇒ D
4. next candidate: E ⇒ E
5. next candidates: A F
   tie breaking ⇒ A
6. next candidate: F ⇒ F

resulting tour: [BCDEAF]

Fitness correlation coefficients

- genetic operators should preserve useful fitness characteristics between parents and offspring
- calculate the fitness correlation coefficient to quantify this
- k-ary operator: generate n sets of k parents
- apply operator to each set to create children
- compute fitness of all individuals
- $\{f(p_{g1}), f(p_{g2}), ..., f(p_{gn}\}$
- $\{f(c_{g1}), f(c_{g2}), ..., f(c_{gn}\}$

Fitness correlation coefficients

- F_p : mean fitness of the parents
  F_c : mean fitness of the children
  $\sigma(F_p)$ = standard deviation of fitness parents
  $\sigma(F_c)$ = standard deviation of fitness children
  $cov(F_p, F_c) = \sum_{i=1}^{n} \frac{(f(p_{gi})-F_p)(f(c_{gi})-F_c)}{n}$
  covariance between fitness parents and fitness children
- operator fitness correlation coefficient ρ_{op}:

$$\rho_{op} = \frac{cov(F_p, F_c)}{\sigma(F_p)\sigma(F_c)}$$

Traveling Salesman problem: mutation operators

- various mutation operators applicable
  - 2opt mutation (*2OPT*)
  - swap mutation (*SWAP*)
  - insert mutation (*INS*)

  performance: $2OPT > INS > SWAP$
- mutation fitness correlation coefficients ρ_{mutate} :

ρ_{2OPT}	0.86
ρ_{INS}	0.80
ρ_{SWAP}	0.77

Traveling Salesman problem: crossover operators

- various crossover operators in applicable
  - cycle crossover (*CX*)
  - partially matched crossover (*PMX*)
  - order crossover (*OX*)
  - edge crossover (*EX*)

  performance: $EX > OX > PMX > CX$
- crossover correlation coefficients ρ_{cross} :

ρ_{EX}	0.90
ρ_{OX}	0.72
ρ_{PMX}	0.61
ρ_{CX}	0.57