Course: Natural Computing

*5. Theory of Genetic Algorithms

J. Michael Herrmann
School of Informatics, University of Edinburgh
michael.herrmann@ed.ac.uk, +44 131 6 517177

Theory

- last week: No free lunch theorem
- **Now: Schema theorem and building blocks**
- Next week: Convergence, parameters etc.

Reminder: The Canonical Genetic Algorithm

1. Old population
2. Roulette-wheel selection
3. Intermediate population
4. Single point recombination with rate p_c (per pair of individuals)
5. Mutation with rate p_m (per position in all strings)
1. New population
(repeat until termination)

one generation

- A population is a (multi-) set of individuals
- An individual (genotype, chromosome) is encoded by a string $S \in \mathcal{A}^L$ ($\mathcal{A}$: alphabet; canonical: $\mathcal{A} = \{0, 1\}$, L fixed)
- Normalised fitness represents the objective of the problem

Search Spaces as Hypercubes

Binary encoding: solution $c \in \{0,1\}^L$

$\Rightarrow$ Each Solution is a corner of the **hypercube**.

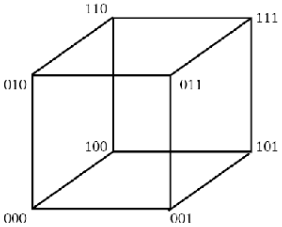
e.g. $c = (0,1,0)$ for $L = 3$ or $c = (0110)$ for $L = 4$

Sets of solutions:

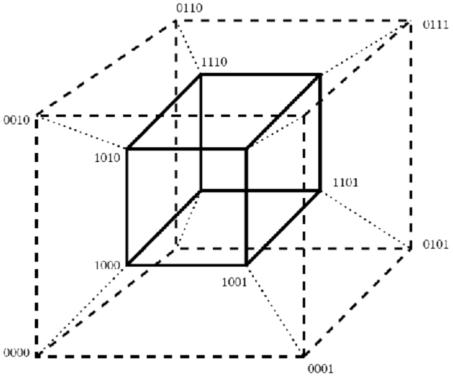
$(0,*,0)$ denotes a line

$(*,1,*)$ denotes a plane

$(1,*,*,*)$ denotes a subcube.



From a tutorial by Erik D. Goldman GECCO09



Schemata (J. Holland, 1975)

- A schema is a string that contains wildcards ("*"), but not only asterisks, i.e. schema $H \in \{0, 1, *\}^L \setminus \{*\}^L$
- A schema defines a set of solutions (which coincide at the no-wildcard symbols)
- All (inheritable) features of the phenotype are encoded by schemata
- The order of the schema is the number of bits that are actually there, e.g. **01***1 is a schema of order 3 (and length 8)
- There are $3^L - 1$ different schemata (not counting the schema of order 0: ** . . . *)
- Each solution is part of 2^L hyperplanes (or $2^L - 1$ schemata)
- Implicit parallelism: Each individual samples many hyperplanes

How do GAs work?

The schema theorem (J. Holland, 1975)

- How does selection improve fitness?
- What is the fate of the schemata in face of selection, mutation and and crossover?

Goal:

$$E\left(m\left(H,t+1\right)\right) \geq \frac{\hat{u}(H,t)}{\bar{f}(t)}\, m\left(H,t\right) \left(1 - P_c \frac{d(H)}{L-1}\right) \left(1-p_m\right)^{o(H)}$$

- H is a schema
- t counts generations
- m is the number of individuals carrying a schema in a generation
- E is the mathematical expectation
- p_c and p_m should be clear, for $\hat{u}$, $\bar{f}$, o, d, L see below

The Schema Theorem
(before starting the proof)

Consider first an individual solution $c_i \in \mathcal{A}^L$:

$f(c_i, t)$: fitness of solution c_i in generation t

$m(c_i, t)$: number of copies of c_i in the population in generation t

$\bar{f}(t)$: average fitness of the population in generation t

$$\boxed{E\left(m\left(c_i, t+1\right)\right) = \frac{f(c_i,t)}{\bar{f}(t)} m\left(c_i, t\right)}$$

$\frac{1}{n}\frac{f(c_i,t)}{\bar{f}(t)}$ the probability of selecting c_i

$\bar{f} = \frac{1}{n}\sum_{i=1}^{n} f(c_i)$
n: population size

So above-average-fitness strings get more copies in the next generation and below average fitness strings get fewer.

The Schema Theorem

Growth of fit subpopulations

Suppose c_i has above-average fitness of $(1+\delta)\,\bar{f}$ (i.e. $\delta > 0$). Then

$$\boxed{E\left(m\left(c_i, t+1\right)\right) = \frac{f\left(c_i\right)}{\bar{f}} m\left(c_i, t\right) = \frac{(1+\delta)\bar{f}}{\bar{f}} m\left(c_i, t\right) = (1+\delta) m\left(c_i, t\right)}$$

If δ is constant then $m\left(c_i, t\right) = (1+\delta)^t m(c_i, 0)$: Exponential growth

If $m\left(c_i\right)$ is small compared to the population size n then δ can indeed be considered constant $\Rightarrow$ Innovations that cause an increase in fitness spread quickly in the population.

Growth is self-limiting: The relative advantage shrinks because with more fit individuals also the average fitness increases$\Rightarrow$ Fit solution tend to dominate the population (crossover and mutation being ignored for the moment).

Analogously: Exponential decay for $\delta < 0$.

Fitness of Schemata

If the solutions c_i, c_j, c_k, ... all sample the same schema H their fitnesses define the (average) fitness of H at time t

$$\hat{u}(H,t) = \frac{1}{m(H,t)} \sum_{c_i \in H} m(c_i,t) f(c_i,t)$$

$m(H,t)$ is the number of instance of H in the population at time t

Note, that the sum is not taken over all possible $c_i \in H$ but only over those which are actually present in the population.

How many instances of H can be expected after selection?

$$E(m(H,t+1)) = \frac{\hat{u}(H,t)}{\bar{f}(t)} m(H,t)$$

Selection of Fit Schemata: Example

What happens when we select and duplicate strings based of fitness?

Suppose the solutions c_i,c_j,c_k sample the schema H, i.e. $c_i \in H$ etc.

Further suppose the average fitness in the population is $\bar{f} = 1$

Using the formula for solutions:

$$\begin{aligned} f(c_i,t) = 2.0,\ m(c_i,t) = 2 &\quad\Rightarrow\quad E(m(c_i,t+1)) = 2 \times \tfrac{2.0}{1.0} = 4 \\ f(c_j,t) = 2.5,\ m(c_j,t) = 2 &\quad\Rightarrow\quad E(m(c_j,t+1)) = 2 \times \tfrac{2.5}{1.0} = 5 \\ f(c_k,t) = 1.5,\ m(c_k,t) = 2 &\quad\Rightarrow\quad E(m(c_k,t+1)) = 2 \times \tfrac{1.5}{1.0} = 3 \end{aligned}$$

All are fitter than average, all increase in their number in the population.

For the schema H (assume sampled only byc_i,c_j,c_k): $m(H,t) = 6$,

$\hat{u}(H,t) = \frac{1}{6}(2 \times 2.0 + 2 \times 2.5 + 2 \times 1.5) = 2$

$\hat{u}(H,t+1) = \frac{1}{12}(4 \times 2.0 + 5 \times 2.5 + 3 \times 1.5) = 2,083$

Number of samples in this hyperplane is expected to increase, but...

Disruption of Schemata

Crossover and mutation are both disruptive and constructive with regards to schemata. consider only disruptive effects.

Crossover:

1 1 * * * * * *

1 * * * * * * 1

Probability of disruption by crossover?

Mutation:

1 1 0 0 1 0 0 1 1 1 0 1 * *

1 1 * * * * 0 1 * * * * * *

Many disruptive possibilities

Only 4 disruptive possibilities

Schema Jargon

Number of defined bits is the **order** $o(H)$ of the schema H

1 0 * * 1 1 0 *	order 5
* 0 * * 1 1 * *	order 3

Defining length is the distance $d(H)$ between the first and the last bit of the schema (i.e. number of potential cuts)

1 0 * * 1 1 0 *	defining length 6
* 0 * * 1 1 * *	defining length 4

i.e. bit position of last 0/1 minus bit position of first 0/1

Disruptive Effects of Crossover

- 1-point crossover with probability p_c
- $d(H)$ is the defining length of H
  $H = * * 1\ 0 * 1 * * \Rightarrow d(H) = 3$
- In a single crossover there are $L-1$ crossover points:
  1 0 1 0 0 1 0 0 7 crossover points
- Of these, $d(H)$ points will disrupt the schema

$$\Pr(\text{disruption}) = p_c \frac{d(H)}{L-1}$$

- Higher chance of survival if $d(H)$ is low

Example: Suppose $p_c = 0.8$, $d(H) = 3$, $L = 100 \Rightarrow$

$$\Pr(\text{disruption}) = 0.8 \times \tfrac{3}{100} = 0.024$$

Disruptive Effects of Mutation

- Single-point mutation with probability p_m (applied to each bit in turn)

$$
\begin{aligned}
&o\left(H\right) \text{ is the order of } H\\
&H = *\ *\ 1\ 0\ *\ 1\ *\ * \Rightarrow o\left(H\right) = 3\\
&H = 1\ 1\ 1\ 0\ *\ 1\ *\ 1 \Rightarrow o\left(H\right) = 6
\end{aligned}
$$

- Probability that a bit survives is $1 - p_m$
- Flipping a defined bit always disrupts a schema, so the probability that the schema survives is

$$\Pr\left(\text{survival}\right) = \left(1 - p_m\right)^{o\left(H\right)}$$

- Best chances for surviving crossover and mutation when $d\left(H\right)$ and $o\left(H\right)$ are both low

Towards the Schema Theorem

First Component of the Schema Theorem

$$E\left(m\left(H,t+1\right)\right)=\frac{\hat{u}\left(H,t\right)}{\bar{f}\left(t\right)}m\left(H,t\right)$$

The other parts of the Schema Theorem

$$\Pr\left(\text{surviving crossover}\right)=1-p_c\frac{d\left(H\right)}{L-1}$$

$$\Pr\left(\text{surviving mutation}\right)=\left(1-p_m\right)^{o\left(H\right)}$$

$$E\left(m\left(H,t+1\right)\right)=\frac{\hat{u}\left(H,t\right)}{\bar{f}\left(t\right)}m\left(H,t\right)\left(1-p_c\frac{d\left(H\right)}{L-1}\right)\left(1-p_m\right)^{o\left(H\right)}\quad ???$$

The Schema Theorem

Schemata are not only being destroyed, but can also be created though crossover and mutation. So we should write an inequality

Goal:

$$E\left(m\left(H,t+1\right)\right) \geq \frac{\hat{u}(H,t)}{\bar{f}(t)} m\left(H,t\right)\left(1-P_c\frac{d(H)}{L-1}\right)\left(1-p_m\right)^{o(H)}$$

Highest when

- $\hat{u}\left(H,t\right)$ is large: fit
- $d\left(H\right)$ is small: short
- $o\left(H\right)$ is small: small number of defined bits

The Schema Theorem in words:
Short, low-order, above-average schemata receive exponentially increasing trials in subsequent generation of a genetic algorithm.

Consequences of the Schema Theorem

- How do schemata arise? Constructive role of mutation and crossover
- Which genes belong to a good schema?
  The algorithm does not easily distinguish important genes from "hitchhikers"
- How well does the expectation describe the population?
- Gradual reduction of relative fitness advantage:
  Other ways to change the fitness?

The Building Block Hypothesis

During crossover, these “building blocks” become exchanged and combined

So the Schema Theorem identifies the building blocks of a good solution although it only addresses the disruptive effects of crossover (and the constructive effects of crossover are supposed to be a large part of why GA work).
How do we address the constructive effects?

Building block hypothesis: A genetic algorithm seeks optimal performance through the juxtaposition of short, low-order, high-performance schemata, called the building blocks.

Crossover combines short, low-order schemata into increasingly fit candidate solutions

- short low-order, high-fitness schemata
- “stepping stone” solutions which combine H_i and H_j to create even higher fitness schemata

The Building Block Hypothesis

Arguments against the validity of the BBH

- **Collateral convergence:** Once the population begins to converge, even slightly, it is no longer possible to estimate the static average fitness of schemata using the information present in the current population.
- **Fitness variance within schemata:** In populations of realistic size, the observed fitness of a schema may be arbitrarily far from the static average fitness, even in the initial population.
- **Compositionality:** Superposition of fit schemata does not guarantee larger schemata that are more fit and these are less likely to survive.

Adapted from John J. Grefenstette. Deception Considered Harmful. 1992

Theory of Genetic Algorithms

Course: Natural Computing (week *5)
(II) GA Variants

J. Michael Herrmann
School of Informatics, University of Edinburgh
michael.herrmann@ed.ac.uk, +44 131 6 517177

Variants of GAs

Selection

- Roulette wheel (see above)
- Non-linear distortions of the fitness function (e.g. steeper for better fitnesses)
- Tournament selection (especially for relative fitnesses, e.g. evolving a strategy for a game
    - select a pair of individual and keep two copies of the winner of the tournament
    - keep one copy of the winner plus with probability p_t a copy of the winner and with probability $1 - p_t$ a copy of the looser
- Elitism: best individuals are moved unchanged to the next generation
- 'Pocket' algorithms remember the current best
- Insertion of a few new random individuals in each generation

Variants of GAs

Crossover

- 1-point
- 2-point, . . . , *n*-point
- Cut and splice (a different cutting point in each of the parents, children of different length)
- Half-uniform crossover scheme (exactly half of the non-matching bits are swapped)
- More than two parents
- Respecting problem structure (and possibly schemata)
- Elitist crossover
- Islands: crossover mostly within groups (more generally: topology or networks)

Variants

Mutation

- Point mutation: flip or random
- Exchange two randomly chosen characters (perhaps coupled mutations)
- Inversion
- Respecting problem structure (and possibly schemata)
- Fitness-dependent (e.g. mutation rate zero for current best and maximal for worst)
- Adaptive mutation rates

Tournament selection vs. Roulette Wheel selection

- Roulette Wheel selection (see above)
  - May be used on (raw) fitness values or rank (here: rank)
  - Chance of survival in a single run (for rank i): $p = (2i)/(n2 + n)$ (at least one from n runs $P = 1 - (1 - p)n$ for the first variant)
  - Best (rank n): $p = 2/(n + 1)$, worst (rank 1): $p = 2/(n2 + n)$
  - Roulette wheel with elitism is fairly similar to tournament
- Tournament selection (n winners from n tournaments)
  - Chance of survival depends on rank: $P = (i - 1)/(n - 1)$ (rank is used for analysis and does not need to be known for the algorithm)
  - selection for tournament may also depend on rank
  - best (rank n) individual beats any other: $P = 1$
  - worst (rank 1) $P = 0$
  - Outcome of a tournament may be stochastic (add elitism)
  - Main advantage: Can be used if fitness function cannot be calculated explicitly, e.g. in the evolution of chess players
  - Better parallelisable

Making it better

- Start the GA from good initial position (seeding). If you know roughly where a solution might lie, use this information.
- Use a representation close to the problem: Does not have to be a fixed length linear binary string – avoid the Hamming Cliff[1]
- Use operators that suit the representation chosen, e.g. crossover only in specific positions
- Run on parallel machines: Island model GA (Evolve isolated subpopulations, allow to migrate at intervals)
- Reduce mutation/crosssover towards the end of run

Reading: Mitchell Chapter 4

[1]) a transition from 011111 to 100000 is small for the phenotype, but may be hard to find for the GA

Behaviour near the optimal solution

How to improve good individuals to perfect ones? ("exploitation")

Problem: [De Jong] Say range of payoff values is [1,100]. Quickly get population with fitness say in [99,100]. Selective differential between best individual and rest, e.g. 99.988 and 100 is very small. Why should GA prefer one over another?

- Dynamically scale fitness as a function of generations or fitness range (scale minimal fitness in the population to zero)
- Use rank-proportional selection to main a constant selection differential. Slows down initial convergence but increases "exploitation" in the final stages.
- Elitism. Keep best individual so far, or, selectively replace worst members of population
- Change parameters to shift balance from exploration at start to exploitation towards the end

Conclusions on GA

- In order to be successful GA algorithms need well structured problems containing building blocks that are indicative of good fitness and large populations (as quasi-models of the problem structure).
- GAs can be useful in setting the basic structure or design of a task by choosing among components that are produced by other approaches.
- GAs are far from reaching the power of natural evolution.