Natural Computing

Lecture 5

Michael Herrmann
mherrman@inf.ed.ac.uk
phone: 0131 6 517177
Informatics Forum 1.42

04/10/2011

The Building Block Hypothesis and GA Variants

The Schema Theorem
From last lecture

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

Highest when

- schema fitness $\hat{u}\left(H,t\right) = \frac{1}{m(H,t)} \sum_{c_i \in H} m\left(c_i,t\right) f\left(c_i,t\right)$ is large (fit)
- defining length $d\left(H\right)$ is small (short)
- order $o\left(H\right)$ 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.

The Building Block Hypothesis

During crossover, “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 (but 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 (BBH): 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 is a hypothesis – so we can do an experiment to test it.

Experiment: Use a problem which contains explicit building blocks and observe the population. Do the building blocks combine to give a good solution in the way the BBH predicts?

Mitchel, Forrest, Holland set up such a problem, using Royal Road (RR) functions. Details: Mitchel, Chapter 4, pp 127-133.

Define fitness in terms of particular schemata:
Substrings that, if present in a population ought to be combinable into the optimal solution.
They should lay out a "Royal Road" to the global optimum.

The first RR function R_1 is defined using a list of schemata s_i. Each s_i has a fitness coefficient c_i. The fitness $R_1(x)$ of a bit string x is given by: $R_1(x) = \sum_i c_i \delta_i$, $\delta_i(x) = \begin{cases} 1 & \text{if } x \in s_i \\ 0 & \text{otherwise} \end{cases}$

Royal Road Functions

Simple example using 16 bits. Suppose:

$s_1 = 1\,1\,1\,1 * * * * * * * * * * * *$

$s_2 = * * * * 1\,1\,1\,1 * * * * * * * *$

$s_3 = * * * * * * * * 1\,1\,1\,1 * * * *$

$s_4 = * * * * * * * * * * * * 1\,1\,1\,1$

and suppose $c_1 = c_2 = c_3 = c_4 = 4$ and

$S_{\text{opt}} = 1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1$

Then $R_1\left(S_{\text{opt}}\right) = \sum_{i=1}^{4} c_i \delta_i \left(S_{\text{opt}}\right) = 16$

Take the string $1\,1\,1\,1\,0\,1\,0\,0\,1\,0\,0\,1\,1\,1\,1\,1$. It samples matches s_1 and s_4. So $\delta_1(x) = \delta_4(x) = 1$ and $\delta_2(x) = \delta_3(x) = 0$.

And $R_1(1\,1\,1\,1\,0\,1\,0\,0\,1\,0\,0\,1\,1\,1\,1\,1) = 8$

Colors red and blue are used here only for visibility of the blocks.

Royal Road Functions

Several Royal Road functions defined in terms of different combinations of schemata with building blocks at different levels, e.g. 4 contiguous 1s, 8 contiguous 1s, 16 contiguous 1s, etc.

Try to evolve the string with all 1s and compare performance of GA against a number of hill-climbing schemes

- Steepest-ascent hill climbing (SAHC)
- Next-ascent hill climbing (NAHC)
- Random mutation hill climbing (RMHC)

Will the GA do better?

Steepest-ascent hill climbing (SAHC)

1. Let current-best be a random string
2. From left to right flip each bit in the string. Record fitness of each one-bit mutant and flip the bit back to its previous state.
3. If any mutant is fitter than current-best, set current-best to fittest mutant and goto 2.
4. If no fitness increase, save current-best and goto 1.
5. After N evaluations return fittest current-best

Next-ascent hill climbing (NAHC)

1. Let current-best be a random string
2. From left to right, flip each bit in the string. If no fitness increase, flip it back. If fitness increases, set current-best to new string and continue mutation new string form one bit after the bit at which the fitness increase was found.
3. If no fitness increase, save current-best and goto 1.
4. After N evaluations return fittest current-best

Random-mutation hill climbing (RMHC)

1. Let current-best be a random string
2. Flip a random bit in the current-best. If no fitness decrease, set current-best to mutated string
3. Repeat 2. until optimal string found or N evaluations completed
4. Return fittest current-best

See Mitchel p. 129 for these algorithms.

Hill-Climbing vs. GA: Results

Number of evaluations to find optimal string (max: 256,000)

200 runs	GA	SAHC	NAHC	RMHC
Mean	61,334	>max	>max	6,179
Median	54,208	>max	>max	5,775

Why did the GA do worse than RMHC? When do GAs perform well?

- By Markov chain analysis, RMHC's expected time is $\approx$ 6549 evaluations. OK.
- What's going wrong with the GA? Larger combinations of the schemata s_i in the GA get broken up by crossover and disrupted by mutation.
- GA suffers from "hitch-hiking": Once an instance of a high-fitness schema is discovered, the "unfit" material, especially that just next to the fit part, spreads along with the fit material. Slows discovery of good schemata in those positions.

Suppose:

Individual X_1: '1 1 1 1 0 1 0 0 1 1 1 1 1 0 0 1': fitness $R(X_1) = 8$

Individual X_2: '0 1 0 0 1 1 1 1 0 0 0 1 0 0 1 1': fitness $R(X_2) = 4$

- Fitness of individual X_1 will be reduced due to crossover with probability $\frac{11}{15} p_c$
- A single mutation may reduce fitness
- Suppose X_1 has above-average fitness and X_2 below-average fitness. Then X_2 might be extinct before successfully crossed with X_1. The population will have to rediscover the second schema. Before rediscovery the "hitch-hiking" substring '0 1 0 0' of X_1 survives because of the fitness which is due to its neighboring schemata.
- Near the global optimum progress becomes more difficult.
- Sampling of the different regions is not independent.

See Mitchell Fig. 4.2, p. 133, for hitch-hiking effect.

Analysis

- Easy problem, no-local minima (so hill-climbing works, RMHS explores systematically across flat regions)
- GA will out-perform HC on parallel machines (why?)
- GA will no sample evenly. The statement of the schema theorem becomes questionable. If partitions were sampled independently, schema theorem would make meaningful predictions.

Idealised GA

Mitchell proposes an idealised GA (IGA)

- Sample a new string X_i uniform-randomly
- If X_i contains a new desired schema, keep it and cross it over with previous best string to incorporate new schema into the solution
- IGA aims to sample each partition independently and tends to keep best schemata in each partition – static Building Block Hypothesis
- It works, and it's N times faster than HC
- IGA is unusable in practice (why?) but gives us a lower bound on the time GA needs to find optimal string.
- In IGA each new string is an independent sample, whereas in RMHC each new sample differs from the previous by only one bit – so RMHC takes longer to construct building blocks

So we have some clues as to when GAs will do well. (Reading: Mitchell Ch. 4)

When do GAs do better than Hill-Climbing

To act *like* an ideal GA and outperform hill-climbing (at least in this sort of landscape) we need

- Independent samples: Big enough population, slow enough selection, high enough mutations rate, so that no bit-positions are fixed at the same value in every chromosome
- Keeping desired schemata: Strong enough selection to keep desired schemata but slow enough selection to avoid hitch-hiking. It is possible to protect bits (by lower p_m, p_c) that were responsible for a strong fitness increase.
- We want crossover to cross over good schemata quickly when they are found to make better chromosomes (but we don't want crossover to disrupt solutions)
- Large N, long string so that speed-up over RMHC is worth it

Not possible to satisfy all constraints at once – tailor to your problem

Where now?

- Schema theorem starts to give us an idea of how GAs work but is flawed $\rightarrow$ need better mathematical models of GA convergence ...
- ... but these better models do not make our GA go faster. Can we fix it empirically? Fix what, exactly?
  1. Standard GA finds good areas, but lacks "killer instinct" to find the globally best solution
  2. Standard crossover often disrupts good solutions late in the run
  3. Binary representations of non-binary problems ofter slow the GA down rather than allowing it to sample more freely. (The "Hamming Cliff")

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

- 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)

- 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

- Change the crossover probability towards the end of run
- 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
- 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)

Reading: Mitchell Chapter 4

Behaviour near the optimal solution

Want to get from good to best individuals. ("killer instinct" or "exploitation")

[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
- 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

Aim is to shift balance from exploration at start to exploitation at end

Towards Memetic Algorithms

- Hill-climbing local neighborhood search is a fast single solution methods which quickly gets stuck in local optima (Cf. SAHC, NAHC)
- Genetic algorithms are a multi-solution technique which find good approximate solution which non-local optima
- Hence: Try applying local search (LS) to each member of a population after crossover/mutation has been applied. We might find locally better solutions, and if near the end of run find the best/optimal solution.
- GH +LS = Memetic Algorithm

Memetic Algorithms

- 1st generation: Hybrid algorithms
  - evolutionary algorithm + local refinement (development and learning)
- 2nd generation: Hyper-heuristic MA (Lamarckian)
  - includes evolution of the learning algorithm(s) by selection of memes
- 3rd generation: Co-evolution, self-generating MA
  - co-adaptation of the representation of memes including discovery of new memes

Outlook

- Biological background
- Hybrid algorithms
- Practical aspects
- Genetic programming
- Continuous evolutionary algorithms
- ACO, PSO