Aghiles Gasselin

RoyalRoads

2023/09/13

Paper introduction

Melanie Mitchell
Santa Fe Institute
1660 Old Pecos Trail, Suite A
Santa Fe, NM 87501

Computer scientist - analogical reasoning, complex systems

John H. Holland
Dept. of Psychology
University of Michigan
Ann Arbor, MI 48109

Computer scientist & elec engineering- 1975 wrote book on “Adaptation in Natural and Artificial Systems”

Stephanie Forrest
Dept. of Computer Science
University of New Mexico
Albuquerque, NM 87131

Computer scientist – specialized in biocomputation

1. History of the work

Reminder of the Building Block Hypothesis : A genetic algorithm (will be referred as GA in the next slides) seeks

optimal performance through the juxtaposition of short, loworder, high-performance schemata, called the building blocks.

- No detailed description on how combination occurs

  -> Design of fitness landscapes : Royal Blocks functions

History of the work

Aghiles Gasselin

- What Makes a Problem Hard for a Genetic Algorithm? Some Anomalous Results and Their Explanation – 1992 by Michel and Forrest.
- The Royal Road for Genetic Algorithms: Fitness Landscapes and GA Performance – 1993 Mitchell, Forrest and Holland.
- When will a Genetic Algorithm Outperform Hill Climbing – 1994 by Michel, Forrest and Holland

- Mitchel Royal Roads, An Introduction to GAs, The MIT Press, 1996

2. Major idea in the paper

- Suggestion of two features of fitness landscapes :

  - the presence of short, low-order, highly fit schemas

  - the presence of intermediate“stepping stones”—intermediate-order higher-fitness schemas

3. Model provided in the paper

- Simple Royal Road function :

$$R_1(x) = \sum_i c_i \delta_i(x), \text{ where } \delta_i(x) = \begin{cases} 1 & \text{if } x \in s_i \\ 0 & \text{otherwise.} \end{cases}$$

s_1 = 11111111**; $c_1 = 8$
s_2 = ********11111111**; $c_2 = 8$
s_3 = ****************11111111**; $c_3 = 8$
s_4 = ************************11111111********************************; $c_4 = 8$
s_5 = ********************************11111111************************; $c_5 = 8$
s_6 = **11111111****************; $c_6 = 8$
s_7 = **11111111********; $c_7 = 8$
s_8 = **11111111; $c_8 = 8$
s_{opt}=1111111111111111111111111111111111111111111111111111111111111111

- Then, what is R1(111……1) = ?
- With this method, GA should outperform simple hill-climbing schemes, no ?

Testing of GA algo

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

Steepest-ascent hill climbing (SAHC)

- 1. Choose a string at random. Call this string current-hilltop.
- 2. Going from left to right, systematically flip each bit in the string, one at a time, recording the fitnesses of the resulting one-bit mutants.
- 3. If any of the resulting one-bit mutants give a fitness increase, then set current-hilltop to the one-bit mutant giving the highest fitness increase (Ties are decided at random.)
- 4. If there is no fitness increase, then save current-hilltop and go to step 1. Otherwise, go to step 2 with the new current-hilltop.
- 5. When a set number of function evaluations has been performed (here, each bit flip in step 2 is followed by a function evaluation), return the highest hilltop that was found.

Next-ascent hill climbing (NAHC)

- 1. Choose a string at random. Call this string current-hilltop.
- 2. For i from 1 to l (where I is the length of the string), flip bit i: if this results in a fitness increase, keep the new string, otherwise flip bit i back. As soon as a fitness increase is found, set current-hilltop to that increasedfitness string without evaluating any more bit flips of the original string. Go to step 2 with the new current-hilltop, but continue mutating the new string starting immediately after the bit position at which the previous fitness increase was found.
- 3. If no increases in fitness were found, save current-hilltop and go step 1.
- 4. When a set number of function evaluations has been performed, return the highest hilltop that was found.

Random-mutation hill climbing (RMHC)

- 1. Choose a string at random. Call this string best-evaluated.
- 2. Choose a locus at random to flip. If the flip leads to an equal or higher fitness, then set best-evaluated to the resulting string.
- 3. Go to step 2 until an optimum string has been found or until a maximum number of evaluations have been performed.
- 4. Return the current value of best-evaluated.

Results of GA against HC algorithms

Table 4.1 Mean and median number of function evaluations to find the optimum string over 200 runs of the GA and of various hill-climbing algorithms on R_1. The standard error ($\sigma/\sqrt{\text{number of runs}}$) is given in parentheses.

200 runs	GA	SAHC	NAHC	RMHC
Mean	61,334 (2304)	> 256,000 (0)	> 256,000 (0)	6179 (186)
Median	54,208	> 256,000	> 256,000	5775

Aghiles Gasselin

Under what conditions will a GA outperform other search algorithms, such as hill climbing?

- Why is RMHC better ?
- Hitchhiking :
    - Happens When :
        - an instance of a higher-order schema is discovered
        - Implicates -> its high fitness allows the schema to spread quickly in the population, with zeros in other positions in the string
        - Result -> slow discovery of schema in other positions
- RMHC doesn’t lose progress -> GA can with crossover and mutations

Aghiles Gasselin

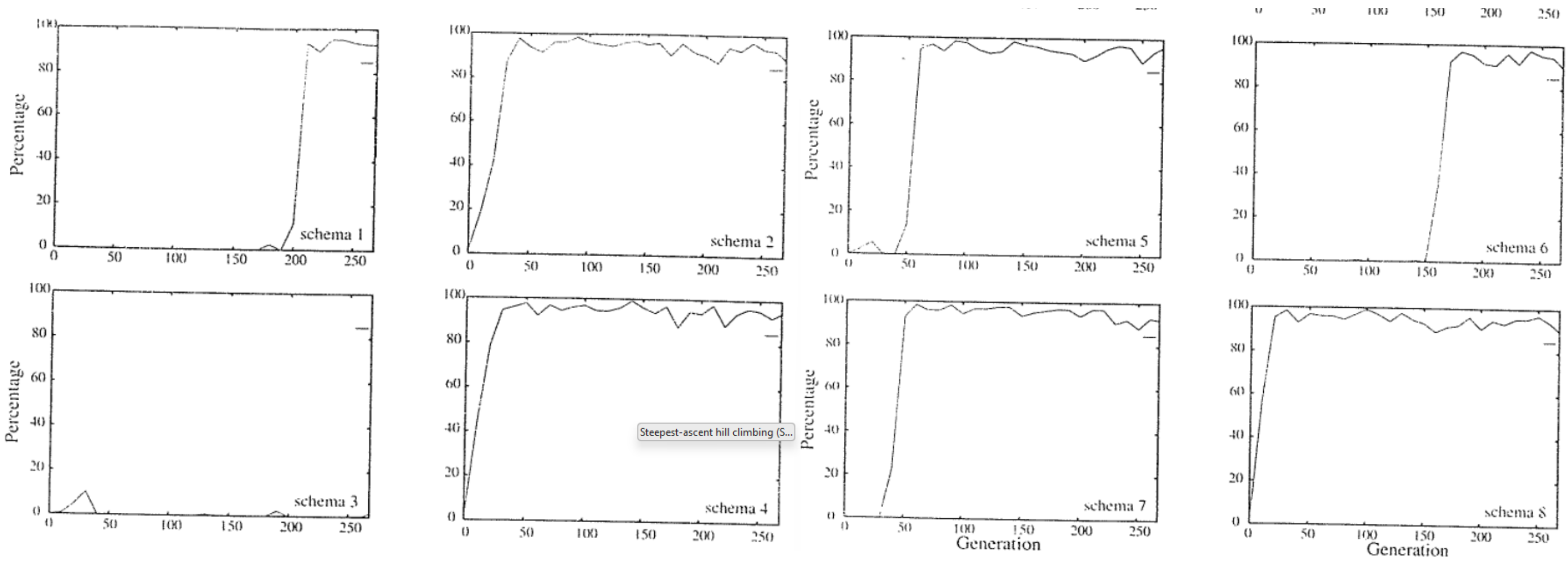

Figure 2 : Mean and median number of function evaluations to find the optimum string over 200 runs of the GA and of various hill-climbing algorithms on &. The standard error (o /v/number of runs) is given in parentheses.

How could we have an idealized GA

- New string random
- If good shema :
    - Keeping good building blocks
- Or:
    - Continue to take new strings
- If new good string found -> crossover with preserved string

- Result : IGA
- But what is the problem of this ? Why IGA can still be useful ?

Aghiles Gasselin

Expected time to find perfect schema with IGA

- H random schema
- p proba of finding H (p = 1/(2^k))
- q proba of not finding -> q = 1 – p
- P(t) proba finding H in time t
- -> P(t) = 1 – q^t
- Now for more than 1 schema to find: PN(t) = (1 – q^t)^N (N number of schema to find)
- Then:

$$P_N(t) = \mathcal{P}_N(t) - \mathcal{P}_N(t-1)$$
$$= (1-q^t)^N - (1-q^{t-1})^N.$$

Via binomial theorem :

$$\mathcal{E}_N = \sum_{t=1}^{\infty} t \; P_N(t)$$
$$= \sum_{t=1}^{\infty} t \; \left((1-q^t)^N - (1-q^{t-1})^N\right).$$

=

$$\left[\binom{N}{1}\left(\frac{1}{q}-1\right)q^t\right] - \left[\binom{N}{2}\left(\frac{1}{q^2}-1\right)q^{2t}\right]$$
$$+ \left[\binom{N}{3}\left(\frac{1}{q^3}-1\right)q^{3t}\right] - \cdots - \left[\binom{N}{N}\left(\frac{1}{q^N}-1\right)q^{Nt}\right].$$

Transformation of our previous result with math

$$\binom{N}{1}\left(\frac{1}{q}-1\right)\sum_{t=1}^{\infty} t q^{t}$$
$$=\binom{N}{1}\left(\frac{1}{q}-1\right)\left(q+2q^{2}+3q^{3}+\cdots\right)$$
$$=\binom{N}{1}\left(\frac{1}{q}-1\right)q\left(1+2q+3q^{2}+\cdots\right)$$
$$=\binom{N}{1}\left(\frac{1}{q}-1\right)q\frac{d}{dq}\left(q+q^{2}+q^{3}+\cdots\right)$$
$$=\binom{N}{1}\left(\frac{1}{q}-1\right)q\frac{d}{dq}\left(\frac{q}{1-q}\right) \quad \text{(using a well-known identity for } 0 \le q < 1\text{)}$$
$$=\binom{N}{1}\left(\frac{1}{q}-1\right)q\left(\frac{1}{1-q}\right)^{2}$$
$$=\binom{N}{1}\frac{1}{1-q}.$$

$$\mathcal{E}_N \approx \frac{1}{p}\left[\frac{\binom{N}{1}}{1}-\frac{\binom{N}{2}}{2}+\frac{\binom{N}{3}}{3}-\cdots-\frac{\binom{N}{N}}{N}\right].$$

K = 8 and N = 8 like example :
Time is 696 -> exact result found in
paper of 1994 (What Makes a Problem Hard for a Genetic Algorithm? Some Anomalous Results and Their Explanation)

Aghiles Gasselin

IGA compared to RMHC and why it's interesting

Can evaluate the previous operation with the following expression : ->

$$\mathcal{E}_N \approx -\frac{1}{p}\sum_{n=1}^{N}\binom{N}{n}\frac{(-1)^n}{n}$$

$$= \frac{1}{p}\sum_{n=1}^{N}\frac{1}{n}$$

$$\approx \frac{1}{p}(\ln N + \gamma)$$

$$= 2^K(\ln N + \gamma).$$

Conclusion:
Order of IGA is 2^K * lnN
Order of RMHC is 2^K * NlnN (calculated in same way, details in paper)

What conclusions can we take from that ?

4. Major result

- Royal Roads
- What stops a GA from being efficient -> hitchhiking
- Understand how and when the GA will outperform hill-climbing with comparison with IGA
- Goal is to have GA approximate as much as possible IGA
- How ? By taking features of the IGA:
  - Independent samples
  - Sequestering desired schemas
  - Instantaneous crossover
  - Speedup over RMHC
- All of this compatible ? No… everything has to be balanced ! (see The Royal Road for Genetic Algorithms: Fitness Landscapes and GA Performance)