EA Theory Notes

Daniel Tauritz, PhD

November 16, 2020

Terminology

- Length of bitstrings is m

Terminology

- Length of bitstrings is m
- Individual's chromosome: $\{0, 1\}^m$

Terminology

- Length of bitstrings is m
- Individual's chromosome: $\{0, 1\}^m$
- Schema - a template allowing exploration of similarities among individuals (binary strings)

Terminology

- Length of bitstrings is m
- Individual's chromosome: $\{0, 1\}^m$
- Schema - a template allowing exploration of similarities among individuals (binary strings)
- A schema consists of 0's, 1's and *'s (don't care symbol)

Terminology

- Length of bitstrings is m
- Individual's chromosome: $\{0, 1\}^m$
- Schema - a template allowing exploration of similarities among individuals (binary strings)
- A schema consists of 0's, 1's and *'s (don't care symbol)
- One particular schema represents all strings (a hyperplane or subset of the search space) which match it on all positions other than '*'

Terminology

- Length of bitstrings is m
- Individual's chromosome: $\{0, 1\}^m$
- Schema - a template allowing exploration of similarities among individuals (binary strings)
- A schema consists of 0's, 1's and *'s (don't care symbol)
- One particular schema represents all strings (a hyperplane or subset of the search space) which match it on all positions other than '*'
- Every schema matches exactly 2^r strings, where r is the number of *'s

Terminology

- Length of bitstrings is m
- Individual's chromosome: $\{0,1\}^m$
- Schema - a template allowing exploration of similarities among individuals (binary strings)
- A schema consists of 0's, 1's and *'s (don't care symbol)
- One particular schema represents all strings (a hyperplane or subset of the search space) which match it on all positions other than '*'
- Every schema matches exactly 2^r strings, where r is the number of *'s
- Each string of length m is matched by 2^m schemata

Terminology

- Length of bitstrings is m
- Individual's chromosome: $\{0, 1\}^m$
- Schema - a template allowing exploration of similarities among individuals (binary strings)
- A schema consists of 0's, 1's and *'s (don't care symbol)
- One particular schema represents all strings (a hyperplane or subset of the search space) which match it on all positions other than '*'
- Every schema matches exactly 2^r strings, where r is the number of *'s
- Each string of length m is matched by 2^m schemata
- For length m there are 3^m possible schemata

Terminology

- Length of bitstrings is m
- Individual's chromosome: $\{0,1\}^m$
- Schema - a template allowing exploration of similarities among individuals (binary strings)
- A schema consists of 0's, 1's and *'s (don't care symbol)
- One particular schema represents all strings (a hyperplane or subset of the search space) which match it on all positions other than '*'
- Every schema matches exactly 2^r strings, where r is the number of *'s
- Each string of length m is matched by 2^m schemata
- For length m there are 3^m possible schemata
- The *order* of schema S (denoted by $o(S)$) is the number of *fixed* positions (non-*don't care* positions) in S $(= m - r)$

Terminology

- Length of bitstrings is m
- Individual's chromosome: $\{0,1\}^m$
- Schema - a template allowing exploration of similarities among individuals (binary strings)
- A schema consists of 0's, 1's and *'s (don't care symbol)
- One particular schema represents all strings (a hyperplane or subset of the search space) which match it on all positions other than '*'
- Every schema matches exactly 2^r strings, where r is the number of *'s
- Each string of length m is matched by 2^m schemata
- For length m there are 3^m possible schemata
- The *order* of schema S (denoted by $o(S)$) is the number of *fixed* positions (non-$don't\ care$ positions) in S $(= m - r)$
- The *defining length* of schema S (denoted by $\delta(S)$) is the distance between the first and the last fixed string positions (i.e., the number of crossover points); it defines the compactness of information contained in a schema

- The number of strings in a population at time t matched by schema S is denoted by $\xi(S,t)$

- The number of strings in a population at time t matched by schema S is denoted by $\xi(S, t)$
- Fitness of individual v_i: $eval(v_i)$

- The number of strings in a population at time t matched by schema S is denoted by $\xi(S, t)$
- Fitness of individual v_i: $eval(v_i)$
- The fitness of a schema at time t, $eval(S, t)$, is defined as the average fitness of all strings in the population matched by the schema S

- The number of strings in a population at time t matched by schema S is denoted by $\xi(S, t)$
- Fitness of individual v_i: $eval(v_i)$
- The fitness of a schema at time t, $eval(S, t)$, is defined as the average fitness of all strings in the population matched by the schema S
- Population consists of strings $\{v_1, \cdots, v_{popsize}\}$

- The number of strings in a population at time t matched by schema S is denoted by $\xi(S, t)$
- Fitness of individual v_i: $eval(v_i)$
- The fitness of a schema at time t, $eval(S, t)$, is defined as the average fitness of all strings in the population matched by the schema S
- Population consists of strings $\{v_1, \cdots, v_{popsize}\}$
- Given p strings $\{v_{i_1}, \cdots, v_{i_p}\}$ in population matched by schema S_i, then:

$$eval(S_i, t) = \sum_{j=1}^{p} eval(v_{i_j})/p \tag{1}$$

- The number of strings in a population at time t matched by schema S is denoted by $\xi(S, t)$
- Fitness of individual v_i: $eval(v_i)$
- The fitness of a schema at time t, $eval(S, t)$, is defined as the average fitness of all strings in the population matched by the schema S
- Population consists of strings $\{v_1, \cdots, v_{popsize}\}$
- Given p strings $\{v_{i_1}, \cdots, v_{i_p}\}$ in population matched by schema S_i, then:

$$eval(S_i, t) = \sum_{j=1}^{p} eval(v_{i_j})/p \tag{1}$$

- Total fitness of population $F(t) = \sum_{i=1}^{popsize} eval(v_i)$

- The number of strings in a population at time t matched by schema S is denoted by $\xi(S, t)$
- Fitness of individual v_i: $eval(v_i)$
- The fitness of a schema at time t, $eval(S, t)$, is defined as the average fitness of all strings in the population matched by the schema S
- Population consists of strings $\{v_1, \cdots, v_{popsize}\}$
- Given p strings $\{v_{i_1}, \cdots, v_{i_p}\}$ in population matched by schema S_i, then:

$$eval(S_i, t) = \sum_{j=1}^{p} eval(v_{i_j})/p \tag{1}$$

- Total fitness of population $F(t) = \sum_{i=1}^{popsize} eval(v_i)$
- Assume generational model with fitness proportional (roulette wheel) selection

- The number of strings in a population at time t matched by schema S is denoted by $\xi(S, t)$
- Fitness of individual v_i: $eval(v_i)$
- The fitness of a schema at time t, $eval(S, t)$, is defined as the average fitness of all strings in the population matched by the schema S
- Population consists of strings $\{v_1, \cdots, v_{popsize}\}$
- Given p strings $\{v_{i_1}, \cdots, v_{i_p}\}$ in population matched by schema S_i, then:

$$eval(S_i, t) = \sum_{j=1}^{p} eval(v_{i_j})/p \qquad (1)$$

- Total fitness of population $F(t) = \sum_{i=1}^{popsize} eval(v_i)$
- Assume generational model with fitness proportional (roulette wheel) selection
- Single string selection chance at time t: $eval(v_i)/F(t)$

- The number of strings in a population at time t matched by schema S is denoted by $\xi(S, t)$
- Fitness of individual v_i: $eval(v_i)$
- The fitness of a schema at time t, $eval(S, t)$, is defined as the average fitness of all strings in the population matched by the schema S
- Population consists of strings $\{v_1, \cdots, v_{popsize}\}$
- Given p strings $\{v_{i_1}, \cdots, v_{i_p}\}$ in population matched by schema S_i, then:

$$eval(S_i, t) = \sum_{j=1}^{p} eval(v_{i_j})/p \tag{1}$$

- Total fitness of population $F(t) = \sum_{i=1}^{popsize} eval(v_i)$
- Assume generational model with fitness proportional (roulette wheel) selection
- Single string selection chance at time t: $eval(v_i)/F(t)$
- Selection chance for average string matched by schema S: $eval(S, t)/F(t)$

- Combining the above we get:

$$E[\xi(S, t+1)] = \xi(S, t) \cdot popsize \cdot eval(S, t)/F(t) \qquad (2)$$

- Combining the above we get:

$$E[\xi(S, t+1)] = \xi(S, t) \cdot popsize \cdot eval(S, t)/F(t) \qquad (2)$$

- Average population fitness $\overline{F(t)} = F(t)/popsize$

- Combining the above we get:

$$E[\xi(S, t+1)] = \xi(S, t) \cdot popsize \cdot eval(S, t)/F(t) \qquad (2)$$

- Average population fitness $\overline{F(t)} = F(t)/popsize$
- Reproductive schema growth equation:

$$E[\xi(S, t+1)] = \xi(S, t) \cdot eval(S, t)/\overline{F(t)} \qquad (3)$$

- Combining the above we get:

$$E[\xi(S,t+1)] = \xi(S,t) \cdot popsize \cdot eval(S,t)/F(t) \tag{2}$$

- Average population fitness $\overline{F(t)} = F(t)/popsize$
- Reproductive schema growth equation:

$$E[\xi(S,t+1)] = \xi(S,t) \cdot eval(S,t)/\overline{F(t)} \tag{3}$$

- If schema S remains above average by $\epsilon\%$, in other words $eval(S,t) = (1+\epsilon) \cdot \overline{F(t)}$, then we can make the following derivation of the geometric progression equation:

$$\begin{aligned} E[\xi(S,t+1)] &= \xi(S,t) \cdot (1+\epsilon) \cdot \overline{F(t)}/\overline{F(t)} \\ &\quad \xi(S,t) \cdot (1+\epsilon) \\ &\quad \xi(S,t-1) \cdot (1+\epsilon)^2 \\ &\quad \xi(S,t-i) \cdot (1+\epsilon)^{i+1} \\ &\quad \xi(S,0) \cdot (1+\epsilon)^{t+1} \end{aligned} \tag{4}$$

- Now assume 1-point crossover with crossover chance p_c; a crossover point is selected uniformly among $m - 1$ possible locations

- Now assume 1-point crossover with crossover chance p_c; a crossover point is selected uniformly among $m-1$ possible locations
- Probability of schema destruction:

$$p_d(S) \leq p_c \cdot \frac{\delta(S)}{m-1} \tag{5}$$

- Now assume 1-point crossover with crossover chance p_c; a crossover point is selected uniformly among $m - 1$ possible locations
- Probability of schema destruction:

$$p_d(S) \leq p_c \cdot \frac{\delta(S)}{m-1} \tag{5}$$

- Consequently, probability of schema survival:

$$p_s(S) \geq 1 - p_c \cdot \frac{\delta(S)}{m-1} \tag{6}$$

- Now assume 1-point crossover with crossover chance p_c; a crossover point is selected uniformly among $m-1$ possible locations
- Probability of schema destruction:

$$p_d(S) \leq p_c \cdot \frac{\delta(S)}{m-1} \tag{5}$$

- Consequently, probability of schema survival:

$$p_s(S) \geq 1 - p_c \cdot \frac{\delta(S)}{m-1} \tag{6}$$

- New reproductive schema growth equation:

$$E[\xi(S,t+1)] \geq \xi(S,t) \cdot \frac{eval(S,t)}{\overline{F(t)}} \left[1 - p_c \cdot \frac{\delta(S)}{m-1}\right] \tag{7}$$

- Now assume 1-point crossover with crossover chance p_c; a crossover point is selected uniformly among $m-1$ possible locations
- Probability of schema destruction:

$$p_d(S) \leq p_c \cdot \frac{\delta(S)}{m-1} \tag{5}$$

- Consequently, probability of schema survival:

$$p_s(S) \geq 1 - p_c \cdot \frac{\delta(S)}{m-1} \tag{6}$$

- New reproductive schema growth equation:

$$E[\xi(S,t+1)] \geq \xi(S,t) \cdot \frac{eval(S,t)}{\overline{F(t)}} \left[1 - p_c \cdot \frac{\delta(S)}{m-1}\right] \tag{7}$$

- Finally, add mutation with bit mutation chance p_m; single bit survival is $1 - p_m$

- Now assume 1-point crossover with crossover chance p_c; a crossover point is selected uniformly among $m-1$ possible locations
- Probability of schema destruction:

$$p_d(S) \leq p_c \cdot \frac{\delta(S)}{m-1} \tag{5}$$

- Consequently, probability of schema survival:

$$p_s(S) \geq 1 - p_c \cdot \frac{\delta(S)}{m-1} \tag{6}$$

- New reproductive schema growth equation:

$$E[\xi(S,t+1)] \geq \xi(S,t) \cdot \frac{eval(S,t)}{\overline{F(t)}} \left[1 - p_c \cdot \frac{\delta(S)}{m-1}\right] \tag{7}$$

- Finally, add mutation with bit mutation chance p_m; single bit survival is $1 - p_m$
- Schema survival $p_s(S) = (1-p_m)^{o(S)}$

- Since $p_m \ll 1$, schema survival can be approximated as $p_s(S) \approx 1 - o(S) \cdot p_m$

- Since $p_m \ll 1$, schema survival can be approximated as $p_s(S) \approx 1 - o(S) \cdot p_m$
- Combined reproductive schema growth equation:

$$E[\xi(S, t+1)] \geq$$

- Since $p_m \ll 1$, schema survival can be approximated as $p_s(S) \approx 1 - o(S) \cdot p_m$
- Combined reproductive schema growth equation:

$$E[\xi(S, t+1)] \geq$$

$$\xi(S, t) \cdot \frac{eval(S, t)}{\overline{F(t)}} \left[1 - p_c \cdot \frac{\delta(S)}{m-1} - o(S) \cdot p_m\right] \tag{8}$$

- Since $p_m \ll 1$, schema survival can be approximated as $p_s(S) \approx 1 - o(S) \cdot p_m$
- Combined reproductive schema growth equation:

$$E[\xi(S, t+1)] \geq$$

$$\xi(S,t) \cdot \frac{eval(S,t)}{\overline{F(t)}} \left[1 - p_c \cdot \frac{\delta(S)}{m-1} - o(S) \cdot p_m\right] \tag{8}$$

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

- Since $p_m \ll 1$, schema survival can be approximated as $p_s(S) \approx 1 - o(S) \cdot p_m$
- Combined reproductive schema growth equation:

$$E[\xi(S, t+1)] \geq$$

$$\xi(S, t) \cdot \frac{eval(S, t)}{\overline{F(t)}} \left[1 - p_c \cdot \frac{\delta(S)}{m-1} - o(S) \cdot p_m\right] \tag{8}$$

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

- Building Block Hypothesis: A genetic algorithm seeks near-optimal performance through the juxtaposition of short, low-order, high-performance schemata, called the building blocks

- Building Block Hypothesis: A genetic algorithm seeks near-optimal performance through the juxtaposition of short, low-order, high-performance schemata, called the building blocks
- Consequence: the manner in which we encode a problem is critical for the performance of a GA - it should satisfy the idea of short building blocks

Example Exam Question

Given the following bit strings v_1 through v_5 and scheme S

$v_1 = (01101110101001)$ fitness(v1) $= 0.8$

$v_2 = (10110010011001)$ fitness(v2) $= 0.1$

$v_3 = (00001010011010)$ fitness(v3) $= 1.0$

$v_4 = (01001110111001)$ fitness(v4) $= 1.2$

$v_5 = (01001011100011)$ fitness(v5) $= 1.9$

$S = (01{*}{*}11101{*}100{*})$

Example Exam Question

Given the following bit strings v_1 through v_5 and scheme S

$v_1 = (01101110101001)$ fitness(v1) $= 0.8$

$v_2 = (10110010011001)$ fitness(v2) $= 0.1$

$v_3 = (00001010011010)$ fitness(v3) $= 1.0$

$v_4 = (01001110111001)$ fitness(v4) $= 1.2$

$v_5 = (01001011100011)$ fitness(v5) $= 1.9$

$S = (01{*}{*}11101{*}100{*})$

- Compute the order of S:

Example Exam Question

Given the following bit strings v_1 through v_5 and scheme S

$v_1 = (01101110101001)$ fitness(v1) $= 0.8$

$v_2 = (10110010011001)$ fitness(v2) $= 0.1$

$v_3 = (00001010011010)$ fitness(v3) $= 1.0$

$v_4 = (01001110111001)$ fitness(v4) $= 1.2$

$v_5 = (01001011100011)$ fitness(v5) $= 1.9$

$S = (01{*}{*}11101{*}100{*})$

- Compute the order of S: 10

Example Exam Question

Given the following bit strings v_1 through v_5 and scheme S

$v_1 = (01101110101001)$ fitness(v1) $= 0.8$

$v_2 = (10110010011001)$ fitness(v2) $= 0.1$

$v_3 = (00001010011010)$ fitness(v3) $= 1.0$

$v_4 = (01001110111001)$ fitness(v4) $= 1.2$

$v_5 = (01001011100011)$ fitness(v5) $= 1.9$

$S = (01{*}{*}11101{*}100{*})$

- Compute the order of S: 10
- Compute the *defining length* of S:

Example Exam Question

Given the following bit strings v_1 through v_5 and scheme S

$v_1 = (01101110101001)$ fitness(v1) $= 0.8$

$v_2 = (10110010011001)$ fitness(v2) $= 0.1$

$v_3 = (00001010011010)$ fitness(v3) $= 1.0$

$v_4 = (01001110111001)$ fitness(v4) $= 1.2$

$v_5 = (01001011100011)$ fitness(v5) $= 1.9$

$S = (01**11101*100*)$

- Compute the order of S: 10
- Compute the *defining length* of S: 13-1=12

Example Exam Question

Given the following bit strings v_1 through v_5 and scheme S

$v_1 = (01101110101001)$ fitness(v1) $= 0.8$

$v_2 = (10110010011001)$ fitness(v2) $= 0.1$

$v_3 = (00001010011010)$ fitness(v3) $= 1.0$

$v_4 = (01001110111001)$ fitness(v4) $= 1.2$

$v_5 = (01001011100011)$ fitness(v5) $= 1.9$

$S = (01{*}{*}11101{*}100{*})$

- Compute the order of S: 10
- Compute the *defining length* of S: 13-1=12
- Compute the fitness of S:

Example Exam Question

Given the following bit strings v_1 through v_5 and scheme S

$v_1 = (01101110101001)$ fitness(v1) $= 0.8$

$v_2 = (10110010011001)$ fitness(v2) $= 0.1$

$v_3 = (00001010011010)$ fitness(v3) $= 1.0$

$v_4 = (01001110111001)$ fitness(v4) $= 1.2$

$v_5 = (01001011100011)$ fitness(v5) $= 1.9$

$S = (01{*}{*}11101{*}100{*})$

- Compute the order of S: 10
- Compute the *defining length* of S: 13-1=12
- Compute the fitness of S: $\frac{0.8+1.2}{2} = 1.0$

Example Exam Question

Given the following bit strings v_1 through v_5 and scheme S

$v_1 = (01101110101001)$ fitness(v1) $= 0.8$

$v_2 = (10110010011001)$ fitness(v2) $= 0.1$

$v_3 = (00001010011010)$ fitness(v3) $= 1.0$

$v_4 = (01001110111001)$ fitness(v4) $= 1.2$

$v_5 = (01001011100011)$ fitness(v5) $= 1.9$

$S = (01{*}{*}11101{*}100{*})$

- Compute the order of S: 10
- Compute the *defining length* of S: 13-1=12
- Compute the fitness of S: $\frac{0.8+1.2}{2} = 1.0$
- Do you expect the number of strings matching S to increase or decrease in subsequent generations?

Example Exam Question

Given the following bit strings v_1 through v_5 and scheme S

$v_1 = (01101110101001)$ fitness(v1) $= 0.8$

$v_2 = (10110010011001)$ fitness(v2) $= 0.1$

$v_3 = (00001010011010)$ fitness(v3) $= 1.0$

$v_4 = (01001110111001)$ fitness(v4) $= 1.2$

$v_5 = (01001011100011)$ fitness(v5) $= 1.9$

$S = (01{*}{*}11101{*}100{*})$

- Compute the order of S: 10
- Compute the *defining length* of S: 13-1=12
- Compute the fitness of S: $\frac{0.8+1.2}{2} = 1.0$
- Do you expect the number of strings matching S to increase or decrease in subsequent generations?
  Average pop fitness: $\frac{0.8+0.1+1.0+1.2+1.9}{5} = 1.0$

Example Exam Question

Given the following bit strings v_1 through v_5 and scheme S

$v_1 = (01101110101001)$ fitness(v1) $= 0.8$
$v_2 = (10110010011001)$ fitness(v2) $= 0.1$
$v_3 = (00001010011010)$ fitness(v3) $= 1.0$
$v_4 = (01001110111001)$ fitness(v4) $= 1.2$
$v_5 = (01001011100011)$ fitness(v5) $= 1.9$
$S = (01{*}{*}11101{*}100{*})$

- Compute the order of S: 10
- Compute the *defining length* of S: 13-1=12
- Compute the fitness of S: $\frac{0.8+1.2}{2} = 1.0$
- Do you expect the number of strings matching S to increase or decrease in subsequent generations?
  Average pop fitness: $\frac{0.8+0.1+1.0+1.2+1.9}{5} = 1.0$
  Decrease, because fitness of S is equal to the average pop fitness and S has a high-order and defining length so large destruction chance.