genetic algorithms
Combinatorial improvement

- Specific heuristics require specific research
- Heuristics need to be tuned for the instance classes they are to be applied to

- Genetic improvement saves research and coding time
- Genetic improvement may treat "easy" problems inefficiently
- Must outperform brute force
- Consistently produce acceptable results

Settle for a generic improvement
A *genetic algorithm* is a probabilistic search algorithm that iteratively transforms a set (called a *population*) of mathematical objects (typically fixed-length binary character strings), each with an associated *fitness* value, into a new population of offspring objects using the Darwinian principle of natural selection and using operations that are patterned after naturally occurring genetic operations, such as crossover (sexual recombination) and mutation.
quick overview

• developed: USA in the 1970’s
• early names: J. Holland, K. DeJong, D. Goldberg
• typically applied to:
  - discrete optimization
• attributed features:
  - not too fast
  - good heuristic for combinatorial problems
• special features:
  - traditionally emphasizes combining information from good parents (crossover)
  - many variants, e.g., reproduction models, operators
how do genetic algorithms work?

- the structure is relatively simple to comprehend, but the dynamic behavior is complex
how do genetic algorithms work?

• the structure is relatively simple to comprehend, but the dynamic behavior is complex

  - organisms produce offspring similar to themselves, but can have variations
    • random changes (mutation)

• combinations of features from each parent (crossover)
how do genetic algorithms work?

• the structure is relatively simple to comprehend, but the dynamic behavior is complex
  - some offspring does survive and some do not
  • the better they adapt to their environment, the higher are the chances

• over time generations become more and more adapted, because the fittest survive
how do genetic algorithms work?

• the structure is relatively simple to comprehend, but the dynamic behavior is complex

• what are the theoretical foundations if any?

• “genetic algorithms work by discovering, emphasizing, and recombining good ‘building blocks’ of solutions in a highly parallel fashion.”
  • Melanie Mitchell, paraphrasing John Holland

• using formalism
  - notion of a building block is formalized as a schema
  - schemata are propagated or destroyed according to the laws of probability
basic cycle of simple genetic algorithms

1. select parents for the mating pool proportional to their fitness, e.g. by roulette-wheel selection (size of mating pool = population size)
2. shuffle the mating pool randomly
3. for each consecutive pair apply crossover with probability $p_c$, else copy parents
4. for each offspring apply mutation (bit-flip with probability $p_m$ independently for each bit)
5. replace the population with the resulting offspring
standard representation and selection

- **genotype space**: \( \{0, 1\}^L \)
- **phenotype space**: encoding (representation) → decoding (inverse representation)

---

**Selection**
- Main idea: better individuals get higher chance
  - Chances proportional to fitness
  - Implementation:
    - Roulette wheel technique
      - Assign to each individual a part of the roulette wheel
      - Pin the wheel \( n \) times to select \( n \) individuals

- Fitnesses:
  - \( \text{fitness}(A) = 3 \)
  - \( \text{fitness}(B) = 1 \)
  - \( \text{fitness}(C) = 2 \)
example: 0-1 knapsack problem

given \( n \) "objects" each with a value and resource claim, select a subfamily of objects with maximal value within a resource constraint

\[
\begin{align*}
\text{max} & \quad \sum_{j=1}^{n} v_j x_j \quad \text{with} \quad x_j \in \{0,1\} \\
\text{subject to} & \quad \sum_{j=1}^{n} w_j x_j \leq b
\end{align*}
\]

0-1 knapsack problem

representation: binary string

fitness: \( \sum_{j=1}^{n} v_j x_j \) if \( \sum_{j=1}^{n} w_j x_j \leq b \) else 0
operators

one-point crossover:

- choose a random point on the two parents
- split parents at this crossover point
- create children by exchanging tails
- $p_c$ typically in range (0.6, 0.9)

offspring:

parents:

[diagram showing binary strings being crossed at a point]

1010001110 0011010010

randomly chosen position

1010010010 0011001110

gene-wise mutation:

- alter each gene independently with a probability $p_m$
- $p_m$ is called the mutation rate
  - typically between 1/pop_size and 1/chromosome_length

parent

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

child

0 1 0 0 1 0 1 1 0 0 0 1 0 1 1 0 0 1
an example

• simple problem: max $x^2$ over $\{0,1,\ldots,31\}$
• approach:
  - representation: binary code, e.g. $01101 \leftrightarrow 13$
  - population size: 4
  - 1-point xover, bitwise mutation
  - roulette wheel selection
  - random initialisation
• we show one generational cycle done by hand
### $x^2$ example: selection

<table>
<thead>
<tr>
<th>String no.</th>
<th>Initial population</th>
<th>$x$ Value</th>
<th>Fitness $f(x) = x^2$</th>
<th>$Prob_i$</th>
<th>Expected count</th>
<th>Actual count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0 1 1 0 1</td>
<td>13</td>
<td>169</td>
<td>0.14</td>
<td>0.58</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1 1 0 0 0</td>
<td>24</td>
<td>576</td>
<td>0.49</td>
<td>1.97</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>0 1 0 0 0</td>
<td>8</td>
<td>64</td>
<td>0.06</td>
<td>0.22</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1 0 0 1 1</td>
<td>19</td>
<td>361</td>
<td>0.31</td>
<td>1.23</td>
<td>1</td>
</tr>
<tr>
<td><strong>Sum</strong></td>
<td></td>
<td><strong>1170</strong></td>
<td></td>
<td><strong>1.00</strong></td>
<td><strong>4.00</strong></td>
<td><strong>4</strong></td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td></td>
<td><strong>293</strong></td>
<td></td>
<td><strong>0.25</strong></td>
<td><strong>1.00</strong></td>
<td><strong>1</strong></td>
</tr>
<tr>
<td><strong>Max</strong></td>
<td></td>
<td><strong>576</strong></td>
<td></td>
<td><strong>0.49</strong></td>
<td><strong>1.97</strong></td>
<td><strong>2</strong></td>
</tr>
</tbody>
</table>
### $x^2$ example: crossover

<table>
<thead>
<tr>
<th>String no.</th>
<th>Mating pool</th>
<th>Crossover point</th>
<th>Offspring after xover</th>
<th>$x$ Value</th>
<th>Fitness $f(x) = x^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0 1 1 0</td>
<td>4</td>
<td>0 1 1 0 0</td>
<td>12</td>
<td>144</td>
</tr>
<tr>
<td>2</td>
<td>1 1 0 0</td>
<td>4</td>
<td>1 1 0 0 1</td>
<td>25</td>
<td>625</td>
</tr>
<tr>
<td>2</td>
<td>1 1</td>
<td>2</td>
<td>1 1 0 1 1</td>
<td>27</td>
<td>729</td>
</tr>
<tr>
<td>4</td>
<td>1 0</td>
<td>2</td>
<td>1 0 0 0 0</td>
<td>16</td>
<td>256</td>
</tr>
<tr>
<td><strong>Sum</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>1754</strong></td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>439</strong></td>
</tr>
<tr>
<td><strong>Max</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>729</strong></td>
</tr>
</tbody>
</table>
$x^2$ example: mutation

<table>
<thead>
<tr>
<th>String no.</th>
<th>Offspring after xover</th>
<th>Offspring after mutation</th>
<th>$x$ Value</th>
<th>Fitness $f(x) = x^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>01100</td>
<td>11100</td>
<td>26</td>
<td>676</td>
</tr>
<tr>
<td>2</td>
<td>11001</td>
<td>11001</td>
<td>25</td>
<td>625</td>
</tr>
<tr>
<td>2</td>
<td>11011</td>
<td>11011</td>
<td>27</td>
<td>729</td>
</tr>
<tr>
<td>4</td>
<td>10000</td>
<td>10100</td>
<td>18</td>
<td>324</td>
</tr>
<tr>
<td>Sum</td>
<td></td>
<td></td>
<td></td>
<td>2354</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td></td>
<td></td>
<td>588.5</td>
</tr>
<tr>
<td>Max</td>
<td></td>
<td></td>
<td></td>
<td>729</td>
</tr>
</tbody>
</table>
schema theorem

- objective: support for the effectiveness of the search process by providing a model for the expectation of the survival of a group of individuals (called a schema)


- it applies to the case of a simple genetic algorithm:
  - binary alphabet;
  - fixed length individuals of equal length, \( l \);
  - fitness proportional selection;
  - single point crossover;
  - gene wise mutation.
schema

• a schema is a subset of the space of all possible individuals for which all the genes match the template for schema $H$.

• a template, much like a regular expression, or a mask, describing a set of strings
  - the set of strings represented by a given schema characterizes a set of candidate solutions sharing a property

• notation: 0 or 1 represents a fixed bit, asterisk represents a “don’t care” ("wild card")

• (defining) length
  - the distance between the first and the last fixed bit (difference between their positions)

• order
  - the number of fixed bits in a schema
examples

• for a binary individual with the gene sequence \(0 1 1 0 0 0\), one (of many) matching schema has the form, \(* 1 1 * 0 * *\)

• the schema \(H = [0 1 * 1 *]\) identifies the chromosome set,
  
  \[
  \begin{align*}
  0 & 1 & 0 & 1 & 0 \\
  0 & 1 & 0 & 1 & 1 \\
  0 & 1 & 1 & 1 & 0 \\
  0 & 1 & 1 & 1 & 1 \\
  
  \end{align*}
  \]

• \(11****00\) is the set encoded in 8 bits,
  - beginning with two ones and ending with two \textit{zeros}
    - length =7
    - order=4

• \(1*01\), beginning with \(1\) and ending with \(01\)
  - length = 3
  - order = 3

• \(0*101*\)
  - length=4
  - order =4
approximating schema dynamics

• let $H$ be a schema with at least one instance present in generation $k$
• let $e(H, k)$ be the number of instances of $H$ in $P(k)$
• let $f(H,k)$ be the average fitness of instances of $H$

$$f(H,k) = \frac{\sum_{x \in H \cap P(k)} f(x)}{e(H,k)}$$

$$F(k) = \frac{1}{N} \sum_{x \in P(k)} f(x)$$

• let $m(H, k)$ be the number of instances of $H$ in the mating pool of generation $k$
• then expected value of $m(H,k)$ is

$$\left( \frac{f(H,k)}{F(k)} \right) e(H,k)$$

• number of offspring of $x$ is $f(x)/f(pop)$
  (roulette-wheel or fitness proportionate selection)
approximating schema dynamics

• schemata with fitness greater (lower) than the average population fitness are likely to account for proportionally more (less) of the population at the next generation
• strictly speaking, for accurate estimates of expectation the population size should be infinite
• note that the average fitness of a schema is never explicitly calculates, but schema proliferation depends on its value

then expected value of $m(H,k)$ is

$$
\left( \frac{f(H,k)}{F(k)} \right) e(H,k)
$$
approximating schema dynamics

- consider the following individual, h, two matching schema, $H_1$, $H_2$ and crossover point between 3rd and 4th gene:
  
  $h = 1 \ 0 \ 1 \ 1 \ 1 \ 0 \ 0$
  $H_1 = \ast \ 0 \ 1 \ \ast \ \ast \ \ast \ 0$
  $H_2 = \ast \ 0 \ 1 \ \ast \ \ast \ \ast \ \ast$

- observations,
  - schema $H_1$ will be broken by the location of the crossover operator unless the second parent is able to 'repair' the disrupted gene.
  - schema $H_2$ emerges unaffected and is therefore independent of the second parent.
  - with $P_{\text{diff}}(H, k)$ is the probability that the second parent from generation $k$ does not match schema $H$

- under single point crossover, the (lower bound) probability of schema $H$ surviving at generation $k$ is, $P(H \text{ survives}) = 1 - P(H \text{ dies}) = 1 - p_C \frac{d(H)}{L-1} P_{\text{diff}}(H, k) \leq 1 - p_C \frac{d(H)}{L-1}$
approximating schema dynamics

- mutation is applied gene by gene.
- in order for schema H to survive, all non * genes in the schema must remain unchanged
- probability of not changing a gene is \((1 - p_m)\)
- require that all \(o(H)\) non * genes survive, or \((1 - p_m)^{o(H)}\)
- typically the probability of applying the mutation operator, \(p_m,<< 1\), thus
  \[
  (1 - p_m)^{o(H)} \approx 1 - o(H)p_m
  \]
- under gene wise mutation, the surviving probability of an order \(o(H)\) schema H at generation k is,

\[(1 - p_m)^{o(H)} \approx 1 - o(H)p_m\]
the schema theorem

• lemma 1: the expected number of instance of H in the mating pool is

\[
\left( \frac{f(H,k)}{F(k)} \right) e(H,k)
\]

• lemma 2: the probability that an instance of H in the mating pool is chosen (pc) and neither of its offspring is in H is

\[
p_c \frac{l(H)}{L-1} p_{\text{diff}}(H,k)
\]

• lemma 3: the probability that an instance of H in the mating pool remains in H after the mutation operator is

\[
(1-p_m)^{o(H)}
\]

the expected number of chromosomes in \( P(k+1) \) that matches schema H is:

\[
\left( 1-p_c \frac{d(H)}{L-1} p_{\text{diff}}(H,k) \right) (1-p_m)^{o(H)} \left( \frac{f(H,k)}{F(k)} \right) e(H,k)
\]
the schema theorem

- **lemma 1**: the expected number of instance of $H$ in the mating pool is
  \[
  \left( \frac{f(H,k)}{F(k)} \right) e(H,k)
  \]

- **lemma 2**: the probability that an instance of $H$ in the mating pool is chosen ($p_c$) and neither of its offspring is in $H$ is less than
  \[
  p_c \frac{d(H)}{L - 1}
  \]

- **lemma 3**: the probability that an instance of $H$ in the mating pool remains in $H$ after the mutation operator is approximately
  \[
  (1 - o(H)p_m)
  \]

the expected number of chromosomes in $P(k+1)$ that matches schema $H$ is at least:

\[
\left( 1 - p_c \frac{d(H)}{L - 1} - o(H)p_m \right) \left( \frac{f(H,k)}{F(k)} \right) e(H,k)
\]
the schema theorem

the theorem was a milestone in the development of genetic algorithms, but it has undesirable assumptions:

• only the worst-case scenario is considered, while ignoring positive effects of the search operators (this has lead to the development of “exact schema theorems”)
• the theorem concentrates on the number of schema surviving not which schema survive (such considerations have been addressed by the utilization of markov chains to provide models of behavior associated with specific individuals in the population)
• claims of “exponential increases” in fit schema are (unfortunately ) misleading!
  Goldberg popularized the following “result”:

  \[ e(H,k + 1) \geq (1 + c)e(H,k) \]

where \( c \) is the constant by which fit schema are always fitter than the population average.
the simple genetic algorithm

• has been subject of many (early) studies
  - still often used as benchmark for novel algorithms
• shows many shortcomings, e.g.
  - representation is too restrictive
  - mutation & crossovers only applicable
    for bit-string & integer representations
  - selection mechanism sensitive for converging populations
    with close fitness values
  - generational population model can be "improved"
    with explicit survivor selection
• therefore,
  - other crossover operators
  - more flexible crossover/mutation
  - other representations
  - other selection strategy
termination

This generational process is repeated until a termination condition has been reached. Common terminating conditions are:

- a solution is found that satisfies minimum criteria
- fixed number of generations reached
- allocated budget (computation time/money) reached
- the highest ranking solution's fitness is reaching or has reached a plateau such that successive iterations no longer produce better results
- manual inspection
- combinations of the above