Genetic Algorithms: Introduction and Principles

Marcus Schmitz
(Petri Eles)

Outline

- Introduction
  - Origin
  - Jargon
- Basic Algorithm
- A GA Simulation by Hand
- Mathematical Foundation
- Implementation Issues
- Applications
  - Mapping
  - Traveling Salesman Problem
From Nature to Genetic Algorithms

- **Charles R. Darwin (1809-1882)**
  - The Origin of Species (1859)
    - "As natural selection works solely by and for the good of each being, all corporeal and mental endowments will tend to progress towards perfection."
    - Survival of the fittest: Organisms that most fit to their environment will tend to survive the struggle for existence. Naturally, survivors pass on their hereditary dispositions to off-springs.

- **Gregor Mendel (1822-1884)**
  - Father of modern genetics
  - Mating experiments with pea plants
  - Mendel’s Laws
    - Law of Segregation
    - Law of Independent Assortment
From Nature to Genetic Algorithms

- Reason for inheritance in organisms is the cell nucleus
- Chromosome: long, continuous piece of DNA which carries genes

From Nature to Genetic Algorithms

- Genetic Algorithms (Rechenberg 1973)
  - Mimic the principles of natural selection to solve search and optimization problems
Introduction

- The algorithm requires feedback in form of a fitness value
  - Fitness function (Cost function)
    - Some idea of the solution quality to guide search

- Multiple objective optimization

- Multiple solutions are evolved in parallel
  - “Communication” through “building blocks” of solutions

Jargon

- **Chromosome**: String of genes, representing a solution candidate
- **Population**: Set of chromosomes (possible solutions)
- **Gene**: Single entry in the chromosome, parameter of the solution set
- **Allele**: Value of a gene
- **Locus**: Gene position in the chromosome
- **Genetic operators**: Transform current chromosomes into new chromosomes
Jargon: Chromosome, Gene

- String of genes, representing a solution candidate
  - Example: HW/SW Co-Design

The Fundamental Algorithm

```
begin
    t ← 0
    initialize P(t)
    evaluate P(t)
    while (not termination)
    begin
        t ← t + 1
        P(t) ← selection(P(t-1))
        crossover P(t)
        mutation P(t)
        evaluate P(t)
    end
end
```
Initialize Population

begin
  \( t \leftarrow 0 \)
  initialize \( P(t) \)
evaluate \( P(t) \)
while (not termination)
begin
  \( t \leftarrow t + 1 \)
  \( P(t) \leftarrow \text{select}(P(t-1)) \)
crossover \( P(t) \)
mutation \( P(t) \)
evaluate \( P(t) \)
end
end

Evaluate Population

begin
  \( t \leftarrow 0 \)
  initialize \( P(t) \)
evaluate \( P(t) \)
while (not termination)
begin
  \( t \leftarrow t + 1 \)
  \( P(t) \leftarrow \text{select}(P(t-1)) \)
crossover \( P(t) \)
mutation \( P(t) \)
evaluate \( P(t) \)
end
end
Selection

begin
  t ← 0
  initialize P(t)
  evaluate P(t)
  while (not termination) begin
    t ← t + 1
    P(t) ← select(P(t-1))
    crossover P(t)
    mutation P(t)
    evaluate P(t)
  end
end

Selection is randomly performed, with a higher probability of selecting chromosomes of high fitness.

⇒ The number of individuals with high fitness increases from population to population.

Crossover

begin
  t ← 0
  initialize P(t)
  evaluate P(t)
  while (not termination) begin
    t ← t + 1
    P(t) ← select(P(t-1))
    crossover P(t)
    mutation P(t)
    evaluate P(t)
  end
end

⇒ New solutions are generated from existing ones.

Crossover point (randomly)

0 0 0 1 1
1 0 1 1 1
1 0 0 1 1

0 0 0 1 1
1 0 1 1 1
1 1 0 0 1
0.93
1.42
0.08

Copied into the next population (generation).
begin 
    \( t \leftarrow 0 \) 
    initialize \( P(t) \) 
    evaluate \( P(t) \) 
    while \( \text{(not termination)} \) do 
        \( t \leftarrow t + 1 \) 
        \( P(t) \leftarrow \text{select}(P(t-1)) \) 
        crossover \( P(t) \) 
        mutation \( P(t) \) 
        evaluate \( P(t) \) 
    end 
end

\( \rightarrow \) New individuals (points in the search space) are visited. Also solutions that would not be reached through crossover.

**Algorithm Outline**

- **Mutation**
  - Low probability
  - Mutate genes

- **Crossover**
  - High probability
  - From parents to child

- **Evaluation**
  - Assign fitness

- **Selection**
  - From parents to child

- **Populations**
  - String 1
  - String 2
  - String 3
  - String 4
  - String 5
  - String 6

- **Quality Function**
  - High
  - Low
GA Simulation by Hand

\[ f(x) = (x - 0.25)^2 + 0.5 \]

\[ f(x_m) \leq f(x), \forall x \in [0..1] \quad \text{(find minimum)} \]

Analytic solution:

\[ f'(x) = 2x - 0.5 \]
\[ f'(x_m) = 0 \]
\[ x_m = 0.25 \]

Chromosomes: Binary Encoding

- The interval [0..1] is encoded into a 8 bit string:

<table>
<thead>
<tr>
<th>Binary String</th>
<th>Decimal Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>0</td>
</tr>
<tr>
<td>00000001</td>
<td>0.0039216</td>
</tr>
<tr>
<td>00000010</td>
<td>0.0078431</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>11111111</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ \frac{1 - 0}{2^8 - 1} = 0.0039216 \]
Create Initial Population

<table>
<thead>
<tr>
<th>P(t)</th>
<th>x</th>
<th>f(x)</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000011</td>
<td>0.0117</td>
<td>0.5568</td>
<td>2</td>
</tr>
<tr>
<td>11011000</td>
<td>0.8471</td>
<td>0.8565</td>
<td>5</td>
</tr>
<tr>
<td>01111111</td>
<td>0.4980</td>
<td>0.5615</td>
<td>3</td>
</tr>
<tr>
<td>10001001</td>
<td>0.5373</td>
<td>0.5825</td>
<td>4</td>
</tr>
<tr>
<td>00010010</td>
<td>0.0706</td>
<td>0.5322</td>
<td>1</td>
</tr>
</tbody>
</table>

Fitness Function \( f(x) = (x - 0.25)^2 + 0.5 \)

## Selection

<table>
<thead>
<tr>
<th>P(t)</th>
<th>x</th>
<th>f(x)</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000011</td>
<td>0.0117</td>
<td>0.5568</td>
<td>2</td>
</tr>
<tr>
<td>11011000</td>
<td>0.8471</td>
<td>0.8565</td>
<td>5</td>
</tr>
<tr>
<td>01111111</td>
<td>0.4980</td>
<td>0.5615</td>
<td>3</td>
</tr>
<tr>
<td>10001001</td>
<td>0.5373</td>
<td>0.5825</td>
<td>4</td>
</tr>
<tr>
<td>00010010</td>
<td>0.0706</td>
<td>0.5322</td>
<td>1</td>
</tr>
</tbody>
</table>

1. RandFloat(0,1) = 0.21 \( \Rightarrow \) 1
2. RandFloat(0,1) = 0.65 \( \Rightarrow \) 3  \( \text{selected for } P(t+1) \)
3. RandFloat(0,1) = 0.98 \( \Rightarrow \) 5
### Crossover (2-point)

<table>
<thead>
<tr>
<th>$P(t+1)$</th>
<th>$x$</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>11011000</td>
<td>$0.0117$</td>
<td>$f(x) = 0.5568$</td>
</tr>
<tr>
<td>01111111</td>
<td>$0.8471$</td>
<td>$f(x) = 0.8565$</td>
</tr>
<tr>
<td>00010010</td>
<td>$0.4980$</td>
<td>$f(x) = 0.5615$</td>
</tr>
<tr>
<td>01110010</td>
<td>$0.5373$</td>
<td>$f(x) = 0.5825$</td>
</tr>
<tr>
<td>00011111</td>
<td>$0.0706$</td>
<td>$f(x) = 0.5322$</td>
</tr>
</tbody>
</table>

Parents: 01111111 00010010

Children: 01110010 00011111

X-over at random point!

### Replacement

<table>
<thead>
<tr>
<th>$P(t+1)$</th>
<th>$x$</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000011</td>
<td>$0.0117$</td>
<td>$f(x) = 0.5568$</td>
</tr>
<tr>
<td>11011000</td>
<td>$0.8471$</td>
<td>$f(x) = 0.8565$</td>
</tr>
<tr>
<td>01111111</td>
<td>$0.4980$</td>
<td>$f(x) = 0.5615$</td>
</tr>
<tr>
<td>10001001</td>
<td>$0.5373$</td>
<td>$f(x) = 0.5825$</td>
</tr>
<tr>
<td>00010010</td>
<td>$0.0706$</td>
<td>$f(x) = 0.5322$</td>
</tr>
</tbody>
</table>

Children: 01110010 00011111
### Second Iteration

<table>
<thead>
<tr>
<th>$P(t+1)$</th>
<th>$x$</th>
<th>$f(x)$</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>01110010</td>
<td>0.4471</td>
<td>0.5388</td>
<td>3</td>
</tr>
<tr>
<td>11011000</td>
<td>0.8471</td>
<td>0.8565</td>
<td>5</td>
</tr>
<tr>
<td>01111111</td>
<td>0.4980</td>
<td>0.5615</td>
<td>4</td>
</tr>
<tr>
<td>00011111</td>
<td>0.1216</td>
<td>0.5165</td>
<td>1</td>
</tr>
<tr>
<td>00010010</td>
<td>0.0706</td>
<td>0.5322</td>
<td>2</td>
</tr>
</tbody>
</table>

- Next selection for crossover: 1 and 4

### Why do GAs work?

<table>
<thead>
<tr>
<th>$f(x)$</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5568</td>
<td>3</td>
</tr>
<tr>
<td>0.8565</td>
<td>5</td>
</tr>
<tr>
<td>0.5615</td>
<td>4</td>
</tr>
<tr>
<td>0.5165</td>
<td>1</td>
</tr>
<tr>
<td>0.5322</td>
<td>2</td>
</tr>
</tbody>
</table>

- Relationship between similarities and high fitness!
  - Information to help guide the search
Similarity Templates (Schemata)

- Which information is admitted?
  - Schemata help to answer this question

  *0000* matches \{00000, 10000\}
  *111* matches \{01110, 01110, 11110, 11111\}

  * ← Don’t care symbol

  \(k^l\): alternative string \((2^5 = 32)\)
  \((k+1)^l\): schemata \((3^5 = 243)\)

Information Amount

- Number of unique schemata in population
  - Each string is a member of \(2^l\) schemata
  - Between \(2^l\) and \(n \cdot 2^l\) \((n: \text{population size})\)

- Defining length of a schema
  - Distance between last and first fixed string position
  \(d(*11*00*) = 6 – 2 = 4\)

- Order of a schema
  - Number of 0 and 1 (fixed) positions
  \(O(*11*00*) = 4\)
Usefully Processed?

- Effect of Selection (Reproduction)
  - Ever-increasing number of individuals with good similarity patterns
- Effect of Crossover
  - Schema can be disrupted or left unscathed
    Examples: 1****0 and **11*
- Effect of Mutation
  - Schema is disrupted with low frequency (low mutation rate)
- Conclusion: Highly fit schemata with short-defining-length and low order (building blocks) are propagated from generation to generation.

Algorithm Setup & Parameters

- Chromosome type (Encoding)
- Population type & size
- Selection scheme
- Crossover types (2-point, 3-point, etc.)
- Mutation strategy & probability
- Fitness function
- Termination criterion
Chromosome

- Principle of meaningful building blocks
  - "Select encoding so that short, low-order schemata are relevant to the underlying problem", i.e., short distance between related bit positions

- Principle of minimal alphabets
  - "Choose smallest alphabet that permits a natural expression of the problem"

Population Types & Size

- Generation-based GAs
  - In each generation all individuals of the population are replaced

- Steady-state GAs
  - Generational overlap: A certain fraction of the population is replaced by new individuals

- Multiple Populations with Immigration
  - Several populations evolve in parallel, individuals can immigrate between population islands (computing clusters)

- Typical Sizes 25 - 2000 chromosomes
Initial Population

- Randomly selected individuals
- Mixed population
  - A fixed amount of individual constructed through different constructive heuristic
  - In addition, random individuals

Selection Scheme

Assignment of reproduction opportunities to the individuals

- Roulette Wheel Selection
  - Fitness determines selection probability
- Ranking-based Selection
  - Ranking determines selection probability
  - Avoids problems with “super-individuals”
- Tournament Selection
  - Randomly select two individuals, the better one is chosen
Crossover Types

1-point : random

2-point:

String Encoding

- Recall: Short defining-length, low order, high fitness schemata (building blocks) recombine
- The coding decision influences the efficiency of GAs

a b c d e f

Likely to be disrupted (long defining-length)

1 * * * * 1 → highest average fitness

Reordering of genes

a f c d e b

Likely to be left undisrupted (short defining-length)

1 1 * * *
Mutation Strategies & Probability

- **Constant Mutation Rate**
  - Genes are altered permanently during optimization with fixed probability (common value <1%)

- **Decreasing Mutation Rate**
  - An initially high mutation rate decreases during optimization run

- **Stimulating Mutation**
  - If premature convergence is detected, an increasing number of individuals are mutated

Fitness function

- **Single-objective optimization**
  - Fitness depends on calculated cost

- **Multi-objective optimization**
  - Objective weighting: 
    \[
    F(x) = \sum_{i=1}^{k} w_i \cdot f_i(x)
    \]
  - Pareto ranking: Distance based
### Pareto Ranking

#### Energy vs. Timing (QoS)

- **Non-Dominated solutions**
- **Pareto front**
- **Dominated solutions**

Non-dominated solutions: at least one of the solution weights is the smallest among all other solutions!

### Termination Criterion

- A given maximal number of generations has been reached
- A certain amount of generations has not produced any further improvements
- The diversity in the population has reached a lower limit
Applicability

- Large Search Space
  - Not perfectly smooth (no gradient-based tech.)
  - Not unimodal (extreme points)
  - Not well understood
  - Noisy fitness function

- Global optimum is not essential
  - High quality solution is sufficient

Knowledge-based Techniques

- In the most general case, GAs are “blind” heuristics, i.e., no problem specific knowledge is required

- Hybrid Schemes
  - Example: GA + local search (GA finds hills, local search climbs hills)
  - Performance improvement
Evolution Programs

- Difference between GAs and EPs?
  - GAs: binary string representations
  - EPs: Complex data structures
  - GAs: Standard genetic operators
  - EPs: Specialized genetic operators

Available Implementations

- GALib (MIT, http://lancet.mit.edu/ga)
  - Includes several GA types
  - Comes with numerous crossover, replacement, mutation types
  - Easily adaptable to specific problems (new genetic operators can be created)

  - Support for multiple, simultaneously evolving populations (computing clusters)
  - Additional optimization algorithms are built-in
    - Simulated annealing
    - Tabu search
Further Readings

- Books
  - Michalewicz, “Genetic Algorithms + Data Structures = Evolution Programs”

- Conference proceedings
  - International Conference on Genetic Algorithms
  - International Conference on Evolutionary Programming

- Journals
  - IEEE Transactions on Evolutionary Computation
  - Evolutionary Computation Journal (MIT Press)

Applications

- Application Mapping in Multiprocessor Systems

- Traveling Salesman Problem
Application Mapping

Task Properties

- $t_i(C)$ is the execution time of task $\tau_i$ on component $C$
- $a_i(C)$ is the area required to accommodate task $\tau_i$ on component $C$
- $P_i(C)$ is the power dissipated by task $\tau_i$ on component $C$

Competing objectives:
- Performance
- Area
- Power consumption
Encoding: Mapping String

- Locus determines task position
- Allele determines task mapping

Fitness Function

\[
F_M = \sum_{\tau \in \mathcal{T}} E(\tau) \cdot \left(1 + \frac{D V^2}{T^*_{\text{rep}}} \right) \cdot \prod_{\pi \in \mathcal{C}} A P_\pi
\]

\[
A P_\pi = \begin{cases} 
1 & \text{if } AA_\pi \geq SA_\pi \\
k \cdot \left( \frac{UA_\pi}{AA_\pi} - 1 \right) + 1 & \text{otherwise}
\end{cases}
\]
GA Mapping Algorithm

Initial Population -> Scheduling

Insertion -> Mutation -> Mating

Selection -> Assign fitness

Timing, Energy + Area

Ranking

Iter.

Termination

Final Population

By itself a hard problem!

Experimental Setup

- Population size: 50
- Generational overlap: 20%
- Two-point crossover
- Dynamic mutation probability 5%

Initial Population

GA (Mapping)
Evolution Run

Fitness vs Generation

30 nodes
3 processors

Multi-Objective Optimization

Area overhead vs Average Power (mW)

Pareto-front
Pareto-points
Experimental Results

<table>
<thead>
<tr>
<th>No. Nodes</th>
<th>CPU time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>5.5</td>
</tr>
<tr>
<td>30</td>
<td>11</td>
</tr>
<tr>
<td>40</td>
<td>37</td>
</tr>
<tr>
<td>100</td>
<td>127</td>
</tr>
</tbody>
</table>

PentiumIII/500MHz

Optimization times include overheads due to scheduling and energy management

Traveling Salesman Problem

- “...given a finite number $n$ of "cities" along with the cost of travel between each pair of them, find the cheapest way of visiting all the cities and returning to your starting point.”

- The problem has a solution space of $(n-1)!/2$

<table>
<thead>
<tr>
<th>Cities</th>
<th>Possible routes</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>181,440</td>
</tr>
<tr>
<td>25</td>
<td>310e21</td>
</tr>
<tr>
<td>100</td>
<td>466e153</td>
</tr>
</tbody>
</table>
Recombination Problem

- Standard GA operators fail to produce meaningful chromosomes
  - Example: 1-point crossover

\[
\begin{align*}
[A & B & C & D & E & F] \\
B & D & C & A & E & F
\end{align*}
\rightarrow
\begin{align*}
[A & B & C & A & E & F] \\
B & C & D & E & F
\end{align*}
\]

- Repair algorithm to restore a valid solution is not effective
- Using appropriate operator that leads to feasible solutions

---

Edge Recombination Operator

- Similarities between tours should be preserved
  - Offspring should be constructed from “links” that exist in the parent tours

- Key to solve the problem is a meaningful recombination technique
  - For example: Edge recombination operator [1]
An Example

- Parent tours: [A B C D E F] and [B D C A E F]
- Edge map:
  - A: B F C E
  - B: A C D F
  - C: B D A
  - D: C E B
  - E: D F A
  - F: A E B


1. Initialize child tour with one of the two initial cities of the parents.

Randomly chosen B.

An Example

- Parent tours: \([A, B, C, D, E, F]\) and \([B, D, C, A, E, F]\)

- Edge map:
  - \(A: B, F, C, E\)
  - \(B: A, C, D, F\)
  - \(C: B, D, A\)
  - \(D: C, E, B\)
  - \(E: D, F, A\)
  - \(F: A, E, B\)

2. Remove all occurrences of \(B\) in the edge map.

Child tour: \([B \ ? \ ? \ ? \ ? \ ?]\)

An Example

- Parent tours: \([A, B, C, D, E, F]\) and \([B, D, C, A, E, F]\)

- Edge map:
  - \(A: B, F, C, E\)
  - \(B: A, C, D, F\)
  - \(C: B, D, A\)
  - \(D: C, E, B\)
  - \(E: D, F, A\)
  - \(F: A, E, B\)

3. Which of the cities in edge list \(B\) has the fewest cities in its own edge list? \(C, D, F\)

Randomly chosen \(C\).

Child tour: \([B, C \ ? \ ? \ ? \ ?]\)
An Example

- Parent tours: \([A \ B \ C \ D \ E \ F] \) and \([B \ D \ C \ A \ E \ F]\)

- Edge map:
  - \(A: B \ F \ C \ E\)
  - \(B: A \ C \ D \ F\)
  - \(C: B \ D \ A\)
  - \(D: C \ E \ B\)
  - \(E: D \ F \ A\)
  - \(F: A \ E \ B\)

4. Remove all occurrences of \(C\) in the edge lists.

Child tour: \([B \ C \ ? \ ? \ ? \ ?]\)

An Example

- Parent tours: \([A \ B \ C \ D \ E \ F] \) and \([B \ D \ C \ A \ E \ F]\)

- Edge map:
  - \(A: B \ F \ C \ E\)
  - \(B: A \ C \ D \ F\)
  - \(C: B \ D \ A\)
  - \(D: C \ E \ B\)
  - \(E: D \ F \ A\)
  - \(F: A \ E \ B\)

5. Which of the cities in edge list \(C\) has the fewest cities in its own edge list? \(D!\)

Chosen \(D\).

Child tour: \([B \ C \ D \ ? \ ? \ ?]\)
An Example

- Parent tours: [A B C D E F] and [B D C A E F]
- Edge map:
  
  A: B F C E  
  B: A C D F  
  C: B D A  
  D: C E B  
  E: D F A  
  F: A E B

6. Remove all occurrences of D in the edge lists.

Child tour: [B C D ? ? ?]

An Example

- Parent tours: [A B C D E F] and [B D C A E F]
- Edge map:
  
  A: B F C E  
  B: A C D F  
  C: B D A  
  D: G E B  
  E: D F A  
  F: A E B

7. Which of the cities in edge list D has the fewest cities in its own edge list? E!

Chosen E.

Child tour: [B C D E ? ?]
An Example

- Parent tours: $[A \ B \ C \ D \ E \ F]$ and $[B \ D \ C \ A \ E \ F]$

- Edge map:
  - $A: B \ F \ C \ E$
  - $B: A \ C \ D \ F$
  - $C: B \ D \ A$
  - $D: C \ E \ B$
  - $E: D \ F \ A$
  - $F: A \ E \ B$

8. Remove all occurrences of $E$ in the edge lists.

Child tour: $[B \ C \ D \ E \ ? \ ?]$

An Example

- Parent tours: $[A \ B \ C \ D \ E \ F]$ and $[B \ D \ C \ A \ E \ F]$

- Edge map:
  - $A: B \ F \ C \ E$
  - $B: A \ C \ D \ F$
  - $C: B \ D \ A$
  - $D: C \ E \ B$
  - $E: D \ F \ A$
  - $F: A \ E \ B$

9. Which of the cities in edge list $E$ has the fewest cities in its own edge list? $F$!

Randomly chosen $A$.

Child tour: $[B \ C \ D \ E \ A \ ?]$
An Example

- Parent tours: [A B C D E F] and [B D C A E F]

- Edge map:
  - A: B F C E
  - B: A C D F
  - C: B D A
  - D: C E B
  - E: D F A
  - F: A E B

All cities have been visit \( \rightarrow \) STOP

Child tour: [B C D E A F]

GA-TSP: Results

- 30 cities (optimal solution 420)
  - 4.42e30 possible tours
  - 10 sub-populations with a size of 200 each
  - 7,000 recombinations
  - 30 out of 30 runs optimal solution found

- 105 cities (optimal solution 14,383)
  - 5.14e165 possible tours
  - 10 sub-populations with a size of 1000 each
  - 200,000 recombinations
  - 15 out of 30 runs optimal solution found
  - 15 out of 30 runs with 1 percent of optimal solution
References


Conclusions

- Simple GA has been introduced
- We have examined how GAs work
- Implementation issues
  - Crossovers
  - Encoding
- Applications
  - Task mapping
  - TSP
- GAs provide a robust, easy to implement heuristic search strategy that can be applied to large number of optimization and search problems