CULTURAL ALGORITHMS: A TUTORIAL

DR. ROBERT G. REYNOLDS
WAYNE STATE UNIVERSITY
DETROIT, MICHIGAN

OUTLINE

• I. Ideational Theories of Cultural Evolution
• II. Cultural Algorithms: A Computational Framework
• III. General Features
• IV. Suitable Problems
• V. Designing Cultural Algorithms
• Embedding a weak method into the Cultural Algorithm Framework: A Genetic Algorithm Example
• IV. Example Applications
• V. Future Directions

Ideational Approaches to Cultural Evolution

• Edward B. Tylor was the first to introduce the term “Culture” in his two volume book on Primitive Culture in 1881.
• He described culture as “that complex whole which includes knowledge, belief, art, morals, customs, and any other capabilities and habits acquired by man as a member of society”.
• Early approaches to studying culture focused on classification of cultures worldwide into groups based upon “adhesions” between cultural elements.
• George Murdoch (1957) produced a “catalog” of 565 cultures based upon 30 sample characteristics.
• Research in Cybernetics and Systems Theory in 1960’s spawned new views of culture as a system that interacted with its environment. It provided regulatory mechanisms that provide positive and negative feedback that can respectively amplify and counteract behavioral deviations of individuals within a cultural group. Flannery 1968.

Ideational Approaches Continued

• In the 1960’s Cultural Ecology emerged as a discipline concerned with the nature of the interactions between the cultural system and its environment.
• In the 1970’s saw a new emphasis on how culture shaped the flow of information in a system, a generalization of the cultural ecology perspective.
• Geertz (1973) “Culture is the fabric of meaning in terms of which human beings interpret their experience and guide their actions.” Less purposeful.
• Durham(1990) “Culture is shared ideational phenomena (values, ideas, beliefs, and the like)”. Less purposeful.
CULTURAL ALGORITHMS ARE COMPUTATIONAL MODELS OF CULTURAL EVOLUTION

BASIC PSEUDOCODE FOR CULTURAL ALGORITHMS IS AS FOLLOWS:

```plaintext
Begin
  t = 0;
  Initialize Population POP(t);
  Initialize Belief Space BLF(t);
repeat
  Evaluate Population POP(t);
  Adjust(BLF(t), Accept(POP(t)));
  Adjust(BLF(t));
  Variation(POP(t) from POP(t-1));
until termination condition achieved
End
```

Cultural algorithms are computational models of cultural evolution. Basic pseudocode for cultural algorithms is as follows:

- **Begin**
  - `t = 0;`
  - Initialize Population `POP(t);`
  - Initialize Belief Space `BLF(t);`
- **repeat**
  - Evaluate Population `POP(t);`
  - Adjust `BLF(t), Accept(POP(t));`
  - Adjust `BLF(t);`
  - Variation `POP(t) from POP(t-1);`
- **until** termination condition achieved
- **End**

**General Features**

- Dual Inheritance (at population and knowledge levels)
- Knowledge are “beacons” that guide evolution of the population
- Supports hierarchical structuring of population and belief spaces.
- Domain knowledge separated from individuals (e.g. ontologies)
- Supports self adaptation at various levels
- Evolution can take place at different rates at different levels (“Culture evolves 10 times faster than the biological component”).
- Supports hybrid approaches to problem solving.
- A computational framework within which many all of the different models of cultural change can be expressed.

The cultural algorithm components consist of a belief space and a population space. The components interact through a communication protocol.
Can support the emergence of hierarchical structures in both the belief and population spaces

Suitable Problems

- Significant amount of domain knowledge (e.g. constrained optimization problems).
- Complex Systems where adaptation can take place at various levels at various rates in the population and belief space.
- Knowledge is in different forms and needs to be reasoned about in different ways.
- Hybrid systems that require a combination of search and knowledge based frameworks.
- Problem solution requires multiple populations and multiple belief spaces and their interaction.
- Hierarchically structured problem environments where hierarchically structured population and knowledge elements can emerge.

II. Designing Cultural Algorithms

1. Design of the knowledge component
   A. Ontological knowledge (shared common concepts for a domain) representation
   B. Constraint knowledge representation
   C. Solution representation
   D. Which will be modified? Update function for each modifiable component.
   E. Knowledge Maintenance

2. Design of the Population Component
   A. State variables that determine solution behavior
   B. How those variables are used to produce a problem solving strategy or behavior.
   C. How such behavior is evaluated?

Designing Cultural Algorithms: Embedding a Weak Method

- Use Genetic Algorithms as an example population model. Show how it can be embedded in the Cultural Framework for a sequence of increasingly complex problems.
- Whether you begin with the belief level or the population level depends on the problem. That is, which of the two is more constrained by the problem?
- Classification Problems Vs. Construction Problems. With former often start with the belief space, with the latter the population space. In real world situations may have both, select the most constrained of the two.
- In either case, iterate between the two adding detail as you go.
The Genetic Algorithm (Davis, 1991)

1. Initialize a population of chromosomes
2. Evaluate each chromosome in the population
3. Create new chromosomes by mating current chromosomes: apply mutation and crossover as the parent chromosomes mate.
4. Delete members of the population to make room for the new chromosomes.
5. Evaluate the new chromosomes and insert them into the population.
6. If time is up, stop and return the best chromosome; if not go to 3.

A Classification Problem

- Mastermind problem.
- Guess the set of objects that the oracle has in mind.
- Can only get information about whether a specific object is included or not.
- Card Problem.

Static Version Spaces

- Use Mitchell's candidate elimination search procedure

G set = \{ # # \}

S set = \{ \}

Cards are divided into two independent categories: suit and face.

Based upon this a possible population is

- [Suit | Face]
- Generate examples at random
- Accept all examples
- No influence (scorecard) until termination
- Update using Mitchell's Candidate Elimination Alg.
- Focus on Suit {a=##, b=##0, r=##1, s=##00, c=##10, h=##01, d=##11}
Classification Example

- Generalize on positive examples and specialize with negative examples. When the arrows overlap then a maximally specific concept is identified. The most general concept or set description that is consistent with the negative examples.
- Here factored the space into two independent subspaces. Information about guesses is used to update each space independently.
- Then select a population representation to generate the guesses.
- Suit|Card Suit = {club, spades, hearts, diamonds} Card = {2, J, Q, K, A}
- Performance function = oracle {right, or wrong}
- Acceptance function all guesses made this generation.
- Influence Function, generate only guesses consistent with the current S and G sets.
- Reproduction and modification, mutate each parent to values within the intersection of the S and G sets.

A Construction Problem

- In a construction problems the state variables are often not independent.
- This means that the lattice may not be easily factored into sub-lattices and updated in parallel. Theoretically all parameter values can be used to organize the set.
- The fan-out at a given level can be an exponential function of the problem size in the worst case.
- Can also be multiple solutions.
- Add operations in the belief space to compensate.
- E.G. Merge, and stable classes. Can prove properties about the operators (e.g. merge does not lose information Sverdlik)
**Boole Problem:**
Infer the characteristic function for a unknown boolean multiplexer.

**Example:**
Characteristic function:
\[ F_6 = A'0A'1D0 + A0A'1D1 + A'0A1D2 + A0A1D3. \]
For \( F_6 \) (2 address lines, 4 data lines).

---

**Problem Representation**

**Chromosome Description**

\[
\begin{array}{cccccc}
A1 & A0 & D0 & D1 & D2 & D3 \\
1 & 0 & 1 & 0 & 1 & 0 & 1
\end{array}
\]

**Version Space Description**

- Generating a homogeneous region with respect to the acceptance function.

---

**Schematic Description of Cultural Algorithm**

- VIP Protocol interconnects the biological and cultural components

---

**“Segmentation”**

\[ G = [ \cdots \#\#\cdots ] \]

\[ S = [0010001] \]

- Generating a homogeneous region with respect to the acceptance function.
“Merging”

- Maximally Specific Generalization

Schema can be merged to share experiences. This can produce group schema.
- By making the relations between schemata explicit one can exploit nested collections of high performance
- E.G. Clustering of successful cases in circuit design problem [Louis et al., FLAIRS 92]
- Cultural Algorithms can exploit collections of nested schemata which is necessary when dealing with complex non-linear systems.

![Diagram](image1)

**Figure 3:** A closer look at the clustered cases reveal nested schemata.

**VGA Symbiosis**

The Version Spaces approach is now feasible for large problems. Since example generation is done automatically by the GA. The Version Space guides the generation process using the VIP relation.

**Schema Theorem:**

\[ m(h_{i+1}) = m(h_i) \times \frac{m(h_i \text{ | } h \in \text{PATHS}_i(h_i))}{m(h_i \text{ | } h \in \text{PATHS}_i(h_i))} \times \frac{1}{m(h_i \text{ | } h \in \text{PATHS}_i(h_i))} + \frac{1}{m(h_i \text{ | } h \in \text{PATHS}_i(h_i))} \]

- The presence of the version space allows the GA system to retain experience outside of its own knowledge base and explore the space at a high rate, even in localized search.
- In addition, the population size needed can be reduced markedly.
- Interpretation of the results can be done at “high level”, relative to accepted hypotheses in the version space.

**Hyperschema Theorem:**

\[ m(h_{i+1} \mid h \in \text{PATHS}_i(h_{i+1})) =\]

\[ = m(h_i \mid h \in \text{PATHS}_i(h_i)) \times \frac{\text{avg}(f(h) \mid h \in \text{PATHS}_i(h_i))}{e} \]

\[ \times \frac{1}{e} \left[ e \times \frac{\text{avg}(\text{len}(h) \mid h \in \text{PATHS}_i(h_i))}{\text{len}} - 1 \right] \]

**Comparison of VGA on Boole with other systems.**

<table>
<thead>
<tr>
<th>Learning Test</th>
<th>Number of Instances seen</th>
<th>Accuracy of Test Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boole</td>
<td>15,000</td>
<td>97.1%</td>
</tr>
<tr>
<td>VGA</td>
<td>15,000</td>
<td>100%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning Test</td>
</tr>
<tr>
<td>---------------</td>
</tr>
<tr>
<td>Boole</td>
</tr>
<tr>
<td>VGA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Quinlan's C4 System (1988).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning Test</td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td>Boole</td>
</tr>
<tr>
<td>VGA</td>
</tr>
</tbody>
</table>
Performance as a function of Generic Operator Probability.

Mutation.

<table>
<thead>
<tr>
<th>Probability of Mutation</th>
<th>Average Number of Reproductions</th>
<th>Marginal Accuracy of The Test Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>15.8</td>
<td>35.6%</td>
</tr>
<tr>
<td>0.2</td>
<td>14.8</td>
<td>100.0%</td>
</tr>
<tr>
<td>0.3</td>
<td>11.2</td>
<td>98.7%</td>
</tr>
</tbody>
</table>

Crossover.

<table>
<thead>
<tr>
<th>Probability of Crossover</th>
<th>Average Number of Reproductions</th>
<th>Marginal Accuracy of The Test Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>16</td>
<td>91.9%</td>
</tr>
<tr>
<td>0.5</td>
<td>14.8</td>
<td>98.1%</td>
</tr>
<tr>
<td>0.8</td>
<td>11.2</td>
<td>90.3%</td>
</tr>
</tbody>
</table>

Experimental Results of $F_{11}$ as a function of population size.

<table>
<thead>
<tr>
<th>Initial Population Size</th>
<th>Average Number of Patterns in the Solution</th>
<th>Average Number of Patterns in the Initial Set</th>
<th>CPU Time in Seconds</th>
<th>Marginal Accuracy of The Test Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>80</td>
<td>16</td>
<td>24</td>
<td>5.4 2385.7 80.9%</td>
</tr>
<tr>
<td>44</td>
<td>48.6</td>
<td>15.8</td>
<td>24</td>
<td>9.6 3172.2 80.6%</td>
</tr>
<tr>
<td>66</td>
<td>37.0</td>
<td>15.8</td>
<td>23.8</td>
<td>12.6 789.3 75.9%</td>
</tr>
<tr>
<td>88</td>
<td>31.4</td>
<td>16</td>
<td>24</td>
<td>1.2 727.9 97.1%</td>
</tr>
<tr>
<td>110</td>
<td>26.0</td>
<td>16</td>
<td>24</td>
<td>0.0 725.5 100%</td>
</tr>
<tr>
<td>132</td>
<td>23.9</td>
<td>16</td>
<td>24</td>
<td>1.0 769.8 97.6%</td>
</tr>
<tr>
<td>154</td>
<td>22.2</td>
<td>16</td>
<td>24</td>
<td>0.2 695.5 99.9%</td>
</tr>
<tr>
<td>176</td>
<td>20.3</td>
<td>16</td>
<td>23.8</td>
<td>0.2 757.2 99.5%</td>
</tr>
<tr>
<td>198</td>
<td>19.8</td>
<td>16</td>
<td>24</td>
<td>0.0 791.1 100%</td>
</tr>
<tr>
<td>220</td>
<td>19.0</td>
<td>16</td>
<td>24</td>
<td>0.0 809.2 100%</td>
</tr>
</tbody>
</table>

Experimental Results for $F_{6}$ as a function of population size.

<table>
<thead>
<tr>
<th>Initial Population Size</th>
<th>Average Number of Patterns in the Solution</th>
<th>Average Number of Patterns in the Initial Set</th>
<th>CPU Time in Seconds</th>
<th>Marginal Accuracy of the Test Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>25.4</td>
<td>8</td>
<td>8</td>
<td>12.0 19.4 44.0%</td>
</tr>
<tr>
<td>24</td>
<td>25.0</td>
<td>8</td>
<td>8</td>
<td>5.8 1.8 72.4%</td>
</tr>
<tr>
<td>36</td>
<td>22.2</td>
<td>8</td>
<td>8</td>
<td>2.6 2.2 86.0%</td>
</tr>
<tr>
<td>46</td>
<td>18.6</td>
<td>8</td>
<td>8</td>
<td>1.6 2.5 90.6%</td>
</tr>
<tr>
<td>60</td>
<td>19.0</td>
<td>8</td>
<td>8</td>
<td>0.4 3.9 97.6%</td>
</tr>
<tr>
<td>72</td>
<td>17.2</td>
<td>8</td>
<td>8</td>
<td>0.4 3.1 97.6%</td>
</tr>
<tr>
<td>84</td>
<td>14.8</td>
<td>8</td>
<td>8</td>
<td>0.4 4.0 97.6%</td>
</tr>
<tr>
<td>96</td>
<td>15.0</td>
<td>8</td>
<td>8</td>
<td>0.4 4.2 97.6%</td>
</tr>
<tr>
<td>108</td>
<td>13.4</td>
<td>8</td>
<td>8</td>
<td>0.4 4.6 97.6%</td>
</tr>
<tr>
<td>120</td>
<td>12.6</td>
<td>8</td>
<td>8</td>
<td>0.0 5.3 100%</td>
</tr>
</tbody>
</table>

Comparison:

- The VGA performs as well as C4 but does not need to generate the 200 examples by hand.
- The VGA requires an order of magnitude fewer trials to solve the problem relative to the Classifier approach.
- The VGA is much less sensitive to genetic operator probabilities which corresponds with behavior predicted by the Hyperschema Theorem.
- Therefore the attention paid to possible symbiotic relationships among components in a hybrid learning system may result in a system capable of outperforming that of its components.
Population Component

- Genetic Algorithms
- Often population model has an inherent knowledge structure associated with it.
- Genetic Algorithms exploit schemata. The VGA model described earlier is nothing more than the explicit use of binary schemata to guide the generation of examples by the Genetic Algorithm population.
- Exploits building blocks. In hierarchical problems building blocks at one level can be exploited and combined at the next level.
- Need to allow our representation scheme to emerge based upon the level of complexity achieved in the mined building blocks.

ROYAL ROAD PROBLEM

- A SIMPLE EXAMPLE
  Parameters: $l = 2, j = 2, u = 3, a^* = 1, v = .02, m^* = 4, b = 8$

<table>
<thead>
<tr>
<th>Goal</th>
<th>score = .06</th>
</tr>
</thead>
<tbody>
<tr>
<td>Individual</td>
<td>0 0 0 0 1</td>
</tr>
<tr>
<td>Individual</td>
<td>0 0 0 0 1</td>
</tr>
<tr>
<td>Individual</td>
<td>0 0 0 0 0 0 1</td>
</tr>
</tbody>
</table>

ROYAL ROAD FUNCTION

\begin{align*}
\text{begin} \\
\text{for each target schema } i \text{ at level } 1 \\
\text{begin} \\
\text{if } (m_i < m^*) \text{ then} \\
\quad p_{i} = i \text{;} \\
\text{else if } (m_i = m^*) \text{ then} \\
\quad p_{i} = i - m_i \text{;} \\
\text{else} \\
\quad p_{i} = 0; \\
\text{end} \\
\text{for each level } j \text{ in hierarchy} \\
\text{begin} \\
\text{if } (t_j > 0) \text{ then} \\
\quad b_{i,j} = a^* + (t_j - 1)k; \\
\text{else} \\
\quad b_{i,j} = 0; \\
\text{end} \\
\text{score+0;} \\
\text{for each target schema } i \text{ at level } 1 \\
\text{score = score + } p_{i}; \\
\text{for each level } j \text{ in hierarchy} \\
\text{score = score + } b_{i,j}; \\
\text{return(score);} \\
\text{end.}
\end{align*}

PATHFINDER

- LOWEST LEVEL OF BELIEF SPACE

Once we acquire building blocks at one level we can

Re-size the version space to exploit them
Can move up and down the hierarchy of bases depending upon how well two adjacent bases do.

**REAL-VALUED SCHEMA IN THE BELIEF SPACE**

Eschelman and Schaeffer proposed interval schemata for real-valued variables.

\[ X \quad Y \]

Caep used this as belief space knowledge to guide search using an ep population to solve unconstrained real-valued function optimization problems. (Chung and Reynolds 1994)

For problems with large basins and or valleys, less information was gained from each individual during a generation. Fuzzy schemata used fuzzy intervals to direct search in these instances (Zhu and Reynolds, 1998).

\[ X \quad Y \]

**TWO BASIC TYPES OF KNOWLEDGE IN THE BELIEF SPACE:**

**Normative Knowledge: Standards of Behavior**

(E.g. \(10 > x > 2\)) Acceptable range of values for parameter \(x\) in a parameter optimization problem

**Situational Knowledge: Individual Examples of Problem Solving Success and or Failure.**

(E.g. \(f(0,1,0)\) has the best observed performance so far.)
Figure 3.8 shows a result of adjusting normative knowledge according to the adjustment rules from the population in the figure 3.8. Since the best individual has better performance value (0.9991) than that of the current exemplar, the current exemplar is replaced with the current best, (0.9991) in the population space.

Figure 3.9 An example result of Adjusting Situational Knowledge

Figure 3.10 shows a result of adjusting normative knowledge according to the adjustment rules from the population in the figure 3.8. The top 2 individuals, <0.01, 0.01> with performance score 0.0001 and <0.0, 0.1> with performance score 0.01 are used to adjust the current normative knowledge from the population.

Figure 3.10 An example result of Adjusting Normative Knowledge

The individuals in figure 2.8 are then become the parents for the next generation of the CAEP system and the process begins anew.
Influence Function for Interval Schemata

Use Cultural Algorithms as a framework in which to perform knowledge-based evolutionary learning

Replace $\sigma_i$ with empirical generalizations produced in the belief space.

Mutation

$$x_i' = x_i + \sigma_i \cdot N_i(0,1)$$

- Interval size information
- Directional knowledge

How is this done?

Adding Constraint Knowledge

- With the addition of constraint knowledge, one-dimensional interval schemata are combined to produce an n-dimensional region.
- Regional schemata result from imposing a grid system of a certain granularity on the space.
- Grid squares are sampled by scouts. They can be classified based upon the problem characteristics they exhibit: e.g. feasible, infeasible, partially feasible, etc.
- The influence function here cause individuals to migrate to or from cells as a function of their characteristics.
- New cells are broken down into subregions, explored and exploited.
- Knowledge base operations allow the fissioning and fusioning of cells.

Regional Schema: an n-dimensional region defined as a combination of intervals that circumscribe a portion of n-dimensional space

- MULTIPLE M-DIMENSIONAL REGIONAL SCHEMATA
- THE ORGANIZATION OF THESE SCHEMATA INTO A HIERARCHICAL STRUCTURE

Acceptance Function:
Here, all individuals are used to update constraint knowledge. The top 20% are used to update the normative knowledge. These 20% are called the eminent individuals.

Update:
Use inference rules to adjust the classification of active cells. E.g. feasible, infeasible, semi-feasible. Adjust the hierarchical structure based upon this information. E.g.

Fission: Split a semi-feasible cell into smaller cells when the number of individuals becomes too high.

Fusion: Merge, recombine children into the original parent. Then can decompose the parent in a different way. E.g. current decomposition is unattractive, e.g. infeasible cell becomes semi-feasible.
Evolution of Constraint Knowledge

Evolution of Normative Knowledge

Implementation and test results

To access the approaches, we used a nonlinear constrained optimization problem [Fiacchini 1990], which is given below:

**Problem Description**

\[
\begin{align*}
\text{Min} & \quad -12x-7y+y^2 \\
\text{Domain constraints:} & \quad 0 \leq x \leq 2, \quad 0 \leq y \leq 3 \\
\text{Problem constraints:} & \quad y \leq 2x^2 + 2 \\
\text{Global best point:} & \quad x^* = 0.71751, \\
& \quad y^* = 1.470 \\
\text{Global best value:} & \quad -16.73889 \\
\text{Optimization goal:} & \quad \leq -16.70
\end{align*}
\]

Cultural Algorithm
Configuration: Embedding Other Methods

- Population models used
  - Genetic Algorithms (Concept learning, optimization)
  - Genetic Programming (Evolving agent strategies)
  - Evolutionary Programming (Real valued function optimization)
  - Evolution Strategies (Robot soccer plays)
  - Memetic models (Evolution of agriculture)
  - Agent based modeling (Evolution of the state, Environmental Impact)
Knowledge Models Used

• Schemata
  • Binary valued (Maletic concept learning, Boole problem, data mining)
  • Real-valued interval schemata (Chang: unconstrained optimization)
  • Fuzzy Real-valued schemata
  • Regional Schemata (Xidong Jin: constrained optimization)
• Semantic Networks (DLMS: Rychtyckyj)
• Graphical Models (GP: Zannoni, Ostrowski)
• Logical and Rule Based models
  (HYBAL: Sverdlik), Fraud Detection (Sternberg), Lazar (Data mining)

Evolution of the State

• Evolution of Complex Social Systems
• Valley of Oaxaca, Mexico
• Implement Marcus and Flannery’s Model of State Formation and Observe the Social Networks that form as a result.
• Compare to the Archaeological data for the Valley

Future Directions

• Integrating Multiple Representations and Population Models
• Parallelization
• Belief Space Evolution
• Designing Cultural Systems
• How does a Culture’s structure and content reflect its problem solving environment (Saleem)

**Book Chapters Co-Authored:**


---

**A Selected Bibliography of Cultural Algorithms**

**Book Chapters:**


---

**Journal Articles:**


---


Papers Published in Conference Proceedings:


