Redundancy and Computational Efficiency in Cartesian Genetic Programming

Hossein Ajerlool

Computer Engineering Department
Sharif University of Technology, Tehran, Iran

Prof.: 
Dr. Beigy
Outline

1. What I’m Going to Talk About
2. Introduction
3. CGP
   - CGP
   - Evolutionary Algorithm
4. Results
   - Experimental Details
   - Results
5. Conclusion
Redundancy in representation of Cartesian genetic programming

The results presented demonstrate the role of mutation and genotype length in the evolvability of the representation.

It is found that the most evolvable representations occur when the genotype is extremely large and in which over 95% of the genes are inactive.
Redundancy in representation of Cartesian genetic programming
The results presented demonstrate the role of mutation and genotype length in the evolvability of the representation.
It is found that the most evolvable representations occur when the genotype is extremely large and in which over 95% of the genes are inactive.
What I’m Going to Talk About

- **Redundancy** in representation of Cartesian genetic programming
- The results presented demonstrate the role of mutation and genotype length in the *evolvability* of the representation.
- It is found that the most evolvable representations occur when the genotype is extremely large and in which over 95% of the genes are inactive.
What I’m Going to Talk About

Introduction
CGP
Results
Conclusion

Introduction?

What’s a CGP

- The **graph-based genetic programming** system is called Cartesian genetic programming (CGP).
- This is a representation of programs in which functions are linked by connections, and the genotype is represented in the form of a list of integers.

Main Proposition

I’ll investigate and discuss the characteristics and properties of the genotype representation used in CGP and, in particular, the role of redundancy and its utility in evolutionary search.
## Introduction?

### What’s a CGP

- The **graph-based genetic programming** system is called Cartesian genetic programming (CGP).
- This is a representation of programs in which **functions are linked by connections**, and the genotype is represented in the form of a list of integers.

### Main Proposition

I’ll investigate and discuss the characteristics and properties of the genotype representation used in CGP and, in particular, **the role of redundancy and its utility in evolutionary search**.
Outline

1. What I’m Going to Talk About
2. Introduction
3. CGP
   - CGP
   - Evolutionary Algorithm
4. Results
   - Experimental Details
   - Results
5. Conclusion
CGP Usage

- CGP was developed from a representation that was used for the evolution of digital circuits.
- It can represent neural networks, programs, circuits, and many other computational structures.
- An important feature is the ease with which it is able to handle problems involving multiple outputs.
CGP Usage

- CGP was developed from a representation that was used for the **evolution of digital circuits**.
- It can represent **neural networks, programs, circuits, and many other computational structures**.
- An important feature is the ease with which it is able to handle problems involving multiple outputs.
CGP Usage

- CGP was developed from a representation that was used for the evolution of digital circuits.
- It can represent neural networks, programs, circuits, and many other computational structures.
- An important feature is the ease with which it is able to handle problems involving multiple outputs.
The Cartesian genotype is a string of integers
\[ C_0, f_0; C_1, f_1; \cdots; C_{cr-1}, f_{cr-1}; O_1, O_2, \cdots, O_m \]
The Cartesian genotype is a string of integers

$$C_0, f_0; C_1, f_1; \ldots; C_{cr-1}, f_{cr-1}; O_1, O_2, \ldots, O_m$$
A single evolutionary operator is used, point mutation. A percentage of genes in the genotype are chosen randomly, and alleles are altered to another randomly chosen value (including itself), provided it conforms to the specified restrictions.
A single evolutionary operator is used, point mutation.

A percentage of genes in the genotype are chosen randomly, and alleles are altered to another randomly chosen value (including itself), provided it conforms to the specified restrictions.
Even four-parity circuit

- The outputs of nodes 8, 11, and 13 are not used.
- These genes are inactive (or temporarily junk) and have a neutral effect on genotype fitness.
- Mutation may cause them to be activated and code for something in the phenotype.
- Noncoding genes are referred to as introns and coding genes as exons.
An Example
Even four–parity circuit

- The outputs of nodes 8, 11, and 13 are not used.
- These genes are inactive (or temporarily junk) and have a neutral effect on genotype fitness.
- Mutation may cause them to be activated and code for something in the phenotype.
- Noncoding genes are referred to as introns and coding genes as exons.
An Example
Even four–parity circuit

- The outputs of nodes 8, 11, and 13 are not used.
- These genes are inactive (or temporarily junk) and have a neutral effect on genotype fitness.
- Mutation may cause them to be activated and code for something in the phenotype.
- Noncoding genes are referred to as introns and coding genes as exons.
The outputs of nodes 8, 11, and 13 are not used.

These genes are inactive (or temporarily junk) and have a neutral effect on genotype fitness.

Mutation may cause them to be activated and code for something in the phenotype.

Noncoding genes are referred to as introns and coding genes as exons.
An Example

Even four–parity circuit

- The outputs of nodes 8, 11, and 13 are not used.
- These genes are inactive (or temporarily junk) and have a neutral effect on genotype fitness.
- Mutation may cause them to be activated and code for something in the phenotype.
- Noncoding genes are referred to as introns and coding genes as exons.
Outline

1. What I’m Going to Talk About
2. Introduction
3. CGP
   - CGP
   - Evolutionary Algorithm
4. Results
   - Experimental Details
   - Results
5. Conclusion
Evolutionary Algorithm

- $1 + \lambda$ evolutionary strategy where $\lambda = 4$, i.e., one parent with four offspring (population size five).
- $g_P$ denotes the parent genotype which is mutated $\lambda$ times to generate $\lambda$ offspring $g_i$.
- $g'_P$ is the new parent that is selected from the population $\{g_P, g_i\}$.
- Selection procedure:
  
  $$
g'_P = \begin{cases} 
g_i, & f(g_i) > f(g_j), \forall i \neq j 
g_i, & f(g_i) = f(g_P), \text{ choose lowest } i 
g_P, & f(g_P) > f(g_i), \forall i
\end{cases}
$$

- The second condition in the above algorithm is extremely important. It is responsible for genetic drift.
Evolutionary Algorithm

- $1 + \lambda$ evolutionary strategy where $\lambda = 4$, i.e., one parent with four offspring (population size five).
- $g_P$ denotes the parent genotype which is mutated $\lambda$ times to generate $\lambda$ offspring $g_i$.
- $g'_P$ is the new parent that is selected from the population $\{g_P, g_i\}$.
- Selection procedure

$$g'_P = \begin{cases} 
  g_i, & f(g_i) > f(g_j), \ \forall i \neq j \\
  g_i, & f(g_i) = f(g_P), \ \text{choose lowest } i \\
  g_P, & f(g_P) > f(g_i), \ \forall i
\end{cases}$$

- The second condition in the above algorithm is extremely important. It is responsible for genetic drift.
Evolutionary Algorithm

- $1 + \lambda$ evolutionary strategy where $\lambda = 4$, i.e., one parent with four offspring (population size five).
- $g_P$ denotes the parent genotype which is mutated $\lambda$ times to generate $\lambda$ offspring $g_i$.
- $g'_P$ is the new parent that is selected from the population $\{g_P, g_i\}$.

Selection procedure

$$g'_P = \begin{cases} 
  g_i, & f(g_i) > f(g_j), \forall i \neq j \\
  g_i, & f(g_i) = f(g_P), \text{ choose lowest } i \\
  g_P, & f(g_P) > f(g_i), \forall i 
\end{cases}$$

- The second condition in the above algorithm is extremely important. It is responsible for genetic drift.
Evolutionary Algorithm

- $1 + \lambda$ evolutionary strategy where $\lambda = 4$, i.e., one parent with four offspring (population size five).
- $g_P$ denotes the parent genotype which is mutated $\lambda$ times to generate $\lambda$ offspring $g_i$.
- $g'_P$ is the new parent that is selected from the population \{\text{g}_P, \text{g}_i\}.

Selection procedure

$$g'_P = \begin{cases} 
\text{g}_i, & f(\text{g}_i) > f(\text{g}_j), \ \forall i \neq j \\
\text{g}_j, & f(\text{g}_j) = f(\text{g}_P), \ \text{choose lowest} \ i \\
\text{g}_P, & f(\text{g}_P) > f(\text{g}_i), \ \forall i 
\end{cases}$$

- The second condition in the above algorithm is extremely important. It is responsible for genetic drift.
Evolutionary Algorithm

- $1 + \lambda$ evolutionary strategy where $\lambda = 4$, i.e., one parent with four offspring (population size five).
- $g_P$ denotes the parent genotype which is mutated $\lambda$ times to generate $\lambda$ offspring $g_i$.
- $g'_P$ is the new parent that is selected from the population $\{g_P, g_i\}$.
- Selection procedure:
  
  $$g'_P = \begin{cases} 
  g_i, & f(g_i) > f(g_j), \ \forall i \neq j \\
  g_i, & f(g_i) = f(g_P), \ \text{choose lowest } i \\
  g_P, & f(g_P) > f(g_i), \ \forall i 
  \end{cases}$$

- The second condition in the above algorithm is extremely important. It is responsible for genetic drift.
Outline

1. What I’m Going to Talk About
2. Introduction
3. CGP
   - CGP
   - Evolutionary Algorithm
4. Results
   - Experimental Details
   - Results
5. Conclusion
Definition of Evolvability

Number of genotypes that have to be evaluated in order to obtain a particular level of fitness

Two problems were considered:

- **Even-three parity:**
  - The primitive function set used with even-three parity was \{AND, OR, NAND, NOR\}.
  - Evolving a correct even-three parity function is difficult using this function set.

- **Two-bit multiplier:**
  - The primitive function set used for the two-bit multiplier was \{AND, OR, NAND, NOR\}.
Definition of Evolvability

Number of genotypes that have to be evaluated in order to obtain a particular level of fitness

Two problems were considered:

- **Even-three parity:**
  - The primitive function set used with even-three parity was \{AND, OR, NAND, NOR\}.
  - Evolving a correct even-three parity function is difficult using this function set.

- **Two-bit multiplier:**
  - The primitive function set used for the two-bit multiplier was \{AND, OR, NAND, NOR\}.
Experimental Details

Definition of Evolvability

Number of genotypes that have to be evaluated in order to obtain a particular level of fitness

Two problems were considered:

- **Even-three parity:**
  - The primitive function set used with even-three parity was \{AND, OR, NAND, NOR\}.
  - Evolving a correct even-three parity function is difficult using this function set.

- **Two-bit multiplier:**
  - The primitive function set used for the two-bit multiplier was \{AND, OR, NAND, NOR\}.
Outline

1. What I’m Going to Talk About
2. Introduction
3. CGP
   - CGP
   - Evolutionary Algorithm
4. Results
   - Experimental Details
   - Results
5. Conclusion
### Minimum Computational Efforts
#### Even-three Parity Problem

<table>
<thead>
<tr>
<th>Genotype Length</th>
<th>Mutation Probability 0.01</th>
<th>Mutation Probability 0.02</th>
<th>Mutation Probability 0.03</th>
<th>Mutation Probability 0.04</th>
<th>Mutation Probability 0.05</th>
<th>Mutation Probability 0.06</th>
<th>Mutation Probability 0.08</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>115,204</td>
<td>39,052</td>
<td>25,224</td>
<td>28,208</td>
<td>24,008</td>
<td>12,904</td>
<td>19,504</td>
</tr>
<tr>
<td>100</td>
<td>48,008</td>
<td>23,604</td>
<td>18,008</td>
<td>16,404</td>
<td>11,604</td>
<td>13,604</td>
<td>15,404</td>
</tr>
<tr>
<td>200</td>
<td>30,020</td>
<td>14,408</td>
<td>12,808</td>
<td>13,608</td>
<td>12,808</td>
<td>7,804</td>
<td>13,004</td>
</tr>
<tr>
<td>300</td>
<td>23,208</td>
<td>14,004</td>
<td>9,612</td>
<td>8,808</td>
<td>9,004</td>
<td>12,408</td>
<td>9,604</td>
</tr>
<tr>
<td>400</td>
<td>20,408</td>
<td>10,004</td>
<td>8,004</td>
<td>9,612</td>
<td>10,004</td>
<td>8,604</td>
<td>9,004</td>
</tr>
<tr>
<td>500</td>
<td>13,804</td>
<td>11,412</td>
<td>7,216</td>
<td>9,608</td>
<td>8,008</td>
<td>10,812</td>
<td>10,404</td>
</tr>
<tr>
<td>600</td>
<td>12,612</td>
<td>9,728</td>
<td>7,752</td>
<td>8,016</td>
<td>8,164</td>
<td>7,928</td>
<td>12,848</td>
</tr>
<tr>
<td>800</td>
<td>12,628</td>
<td>7,688</td>
<td>5,764</td>
<td>5,904</td>
<td>7,024</td>
<td>7,564</td>
<td>14,232</td>
</tr>
<tr>
<td>1000</td>
<td>14,412</td>
<td>7,824</td>
<td>5,604</td>
<td>4,804</td>
<td>6,904</td>
<td>7,564</td>
<td>15,128</td>
</tr>
<tr>
<td>2000</td>
<td>7,040</td>
<td>6,816</td>
<td>5,008</td>
<td>7,208</td>
<td>6,664</td>
<td>11,216</td>
<td>12,624</td>
</tr>
<tr>
<td>3000</td>
<td>8,852</td>
<td>5,784</td>
<td>6,852</td>
<td>5,304</td>
<td>5,824</td>
<td>7,984</td>
<td>19,092</td>
</tr>
<tr>
<td>4000</td>
<td>6,144</td>
<td>3,484</td>
<td>4,644</td>
<td>5,244</td>
<td>7,376</td>
<td>12,208</td>
<td>16,608</td>
</tr>
</tbody>
</table>
# Minimum Computational Efforts

## Two–Bit Multiplier Problem

<table>
<thead>
<tr>
<th>Genotype length</th>
<th>Mutation probability</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.01</td>
</tr>
<tr>
<td>50</td>
<td>-</td>
</tr>
<tr>
<td>60</td>
<td>234,744</td>
</tr>
<tr>
<td>70</td>
<td>103,704</td>
</tr>
<tr>
<td>80</td>
<td>103,684</td>
</tr>
<tr>
<td>90</td>
<td>146,904</td>
</tr>
<tr>
<td>100</td>
<td>77,308</td>
</tr>
<tr>
<td>200</td>
<td>69,152</td>
</tr>
<tr>
<td>300</td>
<td>57,624</td>
</tr>
<tr>
<td>400</td>
<td>37,452</td>
</tr>
<tr>
<td>500</td>
<td>42,256</td>
</tr>
<tr>
<td>600</td>
<td>38,888</td>
</tr>
<tr>
<td>700</td>
<td>39,376</td>
</tr>
<tr>
<td>800</td>
<td>34,568</td>
</tr>
<tr>
<td>1000</td>
<td>34,576</td>
</tr>
<tr>
<td>2000</td>
<td>29,288</td>
</tr>
<tr>
<td>3000</td>
<td>22,568</td>
</tr>
<tr>
<td>4000</td>
<td>25,448*</td>
</tr>
</tbody>
</table>
Minimum Computational Efforts

Even-three Parity Problem

Hossein Ajourloo

Redundancy and Computational Efficiency in CGP
Minimum Computational Efforts
Two–Bit Multiplier Problem

Computational effort is at a minimum for the largest genotypes and small mutation probabilities.
Minimum Computational Efforts
Two–Bit Multiplier Problem

Computational effort is at a minimum for the largest genotypes and small mutation probabilities.
Minimum Computational Efforts Over All Mutation Rates
Even-three Parity Problem
Minimum Computational Efforts Over All Mutation Rates
Two–Bit Multiplier Problem
Average Final Phenotype Length for All Mutation Probabilities

![Graph showing the relationship between Average phenotype length \( L_p \) (at end of run) and Genotype length \( L_g \) (nodes).]
## Average Phenotype Length

4000 generations (with mutation probability 0.01)

<table>
<thead>
<tr>
<th>$L_{g}$</th>
<th>average $L_{p}$ of initial population</th>
<th>average increase of $L_{p}$ (in %) at end of run</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>15.41</td>
<td>18.4*</td>
</tr>
<tr>
<td>60</td>
<td>17.67</td>
<td>11.4*</td>
</tr>
<tr>
<td>70</td>
<td>19.74</td>
<td>13.1*</td>
</tr>
<tr>
<td>80</td>
<td>21.55</td>
<td>10.2*</td>
</tr>
<tr>
<td>90</td>
<td>23.55</td>
<td>6.1*</td>
</tr>
<tr>
<td>100</td>
<td>25.06</td>
<td>3.2</td>
</tr>
<tr>
<td>200</td>
<td>36.84</td>
<td>5.1</td>
</tr>
<tr>
<td>300</td>
<td>46.27</td>
<td>7.5</td>
</tr>
<tr>
<td>400</td>
<td>54.24</td>
<td>0.8</td>
</tr>
<tr>
<td>500</td>
<td>61.51</td>
<td>4.4</td>
</tr>
<tr>
<td>600</td>
<td>68.52</td>
<td>3.9</td>
</tr>
<tr>
<td>700</td>
<td>72.89</td>
<td>-0.7</td>
</tr>
<tr>
<td>800</td>
<td>77.65</td>
<td>6.7</td>
</tr>
<tr>
<td>1000</td>
<td>81.89</td>
<td>6.1</td>
</tr>
<tr>
<td>2000</td>
<td>86.48</td>
<td>2.6</td>
</tr>
<tr>
<td>3000</td>
<td>124.75</td>
<td>7.3</td>
</tr>
<tr>
<td>4000</td>
<td>156.96</td>
<td>3.8</td>
</tr>
</tbody>
</table>
The evolutionary algorithm performs best when almost all the genotype, is on average, inactive.
The evolutionary algorithm performs best when almost all the genotype, is on average, inactive.
Unexpected benefits of CGP redundancy to evolutionary search.
Julian F. Miller and Stephen L. Smith, Redundancy and Computational Efficiency in Cartesian Genetic Programming.

*IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, VOL. 10, NO. 2, APRIL 2006*