Evolving Algebraic Constructions for Designing Bent Boolean Functions

Stjepan Picek and Domagoj Jakobovic

KU Leuven, ESAT/COSIC and iMinds, Belgium
Faculty of Electrical Engineering and Computing, Croatia

6th July 2016
Outline

1. Introduction
   - Introduction
   - Motivation

2. Our Approach
   - Designing constructions

3. Results
   - Human competitive results
   - Perspectives

4. Conclusion
Relevance of the problem

- Boolean functions are widely used in cryptography, sequences, and coding theory.
- Obtaining new functions as well as new constructions is an interesting problem.
- Since the search space is very large ($2^{2^n}$) exhaustive search is virtually impossible for sizes larger than five.
- From the practical perspective we are interested in much larger functions (in cryptography, at least 13 inputs).
Relevance of the problem

- There are algebraic constructions, random search, and heuristics.
- Furthermore, all constructions are either primary or secondary.
- There exists only a few algebraic constructions for generating (bent) Boolean functions.
- The last such construction was introduced around 2006.
The evolutionary community has been very interested in the problem of obtaining Boolean functions of certain sizes and properties.

For smaller sizes of Boolean functions, EC is extremely competitive when compared with algebraic constructions.

However, for larger sizes the search space is too big and the usual encodings too inefficient to obtain top results.

Therefore, we aim to combine the best of the algebraic constructions and heuristics worlds.
Relevance of the problem

- Instead of evolving bent Boolean functions, we evolve algebraic constructions that result in bent functions.
- Completely novel approach.
- To that end, we use Genetic Programming technique.
- Simple fitness functions, small function set.
Comparison with other approaches

- Our technique offers extremely fast generation of a large number of bent Boolean functions.
- With our approach the problem is “easy”, i.e., it scales for any size, which is not the case with the random search and other heuristics.
- Our approach on average requires only several thousands of evaluations in order to reach good construction.
Comparison with other approaches

- To generate bent Boolean functions with 16 inputs we require less than one minute on a single computer core.
Comparison with other approaches

- To generate bent Boolean functions with 16 inputs we require less than one minute on a single computer core.
- To generate bent Boolean functions with 18 inputs we require several minutes on a single computer core.
Comparison with other approaches

- To generate bent Boolean functions with 16 inputs we require less than one minute on a single computer core.
- To generate bent Boolean functions with 18 inputs we require several minutes on a single computer core.
- PPSN 2014 paper (Humies 2014 bronze award) for 16 inputs bent Boolean functions requires on average around 600 seconds on a 40-node parallel environment.
Comparison with other approaches

- To generate bent Boolean functions with 16 inputs we require less than one minute on a single computer core.
- To generate bent Boolean functions with 18 inputs we require several minutes on a single computer core.
- PPSN 2014 paper (Humies 2014 bronze award) for 16 inputs bent Boolean functions requires on average around 600 seconds on a 40-node parallel environment.
- Furthermore, the authors state that their approach is able to find 18 inputs bent Boolean functions.
Comparison with other approaches

- To generate bent Boolean functions with 16 inputs we require less than one minute on a single computer core.
- To generate bent Boolean functions with 18 inputs we require several minutes on a single computer core.
- PPSN 2014 paper (Humies 2014 bronze award) for 16 inputs bent Boolean functions requires on average around 600 seconds on a 40-node parallel environment.
- Furthermore, the authors state that their approach is able to find 18 inputs bent Boolean functions.
- We used our approach for finding bent Boolean functions up to 30 inputs.
Comparison with other approaches

Table: Comparison among construction techniques for bent Boolean functions.

<table>
<thead>
<tr>
<th>Construction</th>
<th>Advantages</th>
<th>Disadvantages</th>
<th>Human competitiveness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human-made</td>
<td>Provably correct</td>
<td>Very hard to design</td>
<td>Truly human competitive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Limited number of solutions</td>
<td></td>
</tr>
<tr>
<td>Other metaheuristics</td>
<td>Large number of solutions</td>
<td>Inefficient for larger sizes</td>
<td>Not really (trivial for small sizes, poor for larger)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Very slow</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Affine-equivalent solutions?</td>
<td></td>
</tr>
<tr>
<td>Our approach</td>
<td>Very fast</td>
<td>Affine-equivalent solutions?</td>
<td>Overcomes disadvantages of other approaches</td>
</tr>
<tr>
<td></td>
<td>Large number of solutions</td>
<td>Scalable</td>
<td>As good as human-made</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Future Perspectives

- A tool how to obtain balanced Boolean function with high nonlinearity.
- A tool to obtain primary algebraic constructions.
- Inspiration point for completely new constructions.
We investigate relevant, real-world problem that is very difficult due to a huge search space size.

The results show that our novel technique is extremely fast and powerful.

As far as we are aware, for realistic sizes, no other heuristics is able to compete even by far.

Since we are able to obtain many constructions, we can produce many Boolean functions, which is not the case with algebraic (human-made) constructions.
Thank You for Your attention.