Engineering Applications of Artificial Intelligence xx (xxxx) xxxx–xxxx

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Immunological algorithms paradigm for construction of Boolean functions with good cryptographic properties

Stjepan Picek[a,*], Dominik Sisejkovic[b], Domagoj Jakobovic[b]

[a] KU Leuven, ESAT/COSIC and iMinds, Kasteelpark Arenberg 10, bus 2452, B-3001 Leuven-Heverlee, Belgium
[b] Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia

ARTICLE INFO

Keywords:
Artificial immune systems
Evolutionary algorithms
Boolean functions
Cryptography
Comparison
Efficiency analysis

ABSTRACT

In this paper we investigate the efficiency of two immunological algorithms (CLONALG and opt-IA) in the evolution of Boolean functions suitable for use in cryptography. Although in its nature a combinatorial problem, we experiment with two representations of solutions, namely, the bitstring and the floating point based representation. The immunological algorithms are compared with two commonly used evolutionary algorithms – genetic algorithm and evolution strategy. To thoroughly investigate these algorithms and representations, we use four different fitness functions that differ in the number of parameters and difficulty. Our results indicate that for smaller dimensions immunological algorithms behave comparable with evolutionary algorithms, while for the larger dimensions their performance is somewhat worse. When considering only immunological algorithms, opt-IA outperforms CLONALG in most of the experiments. The difference in the representation for those algorithms is also clear where floating point works better with smaller problem sizes and bitstring representation works better for larger Boolean functions.

1. Introduction

Cryptography, in its core, is a science (and art) of secret writing with the goal of hiding the meaning of a message (Paar and Pelzl, 2010). As such, it plays a tremendous role in people's everyday life. To ensure the secrecy of information (but also authentication and data integrity among other relevant goals (Katz and Lindell, 2015)) we rely on strong, well-designed cryptographic algorithms – commonly referred as **ciphers**.

When the ciphers use keys, we can divide them on the basis whether all communicating parties use the same key or not (Knudsen and Robshaw, 2011). If all entities use the same key for encryption and decryption operations then we talk about **symmetric-key cryptography** or **secret-key cryptography**. Assume that two parties (commonly denoted as Alice and Bob) want to exchange some message and they want it to remain secret, i.e. that an attacker (commonly denoted as Eve) cannot read it. Alice could encrypt her message and send it over an insecure (public) channel to Bob. If Bob has the same key as Alice, he can then decrypt and read the message. Eve cannot decrypt the message if she does not know the key. Therefore, if Alice and Bob want to keep their communication private they need either to keep the key secret or the algorithm secret. However, in 19th century Kerchoff stated that a cryptosystem should be secure even if everything about the system, except the key, is publicly known (Kerckhoffs, 1883).

A classical division of symmetric-key cryptography is on **block ciphers** and **stream ciphers** (Knudsen and Robshaw, 2011). A common trait for all those ciphers is that they are designed in accordance with a number of cryptographic criteria they need to fulfill. Those criteria enable ciphers to resist various cryptanalysis attacks where to resist linear (Matsui and Yamagishi, 1993) cryptanalysis, we require that the cipher possess enough nonlinearity.

In both block and stream ciphers one common source of nonlinearity (although not the only one) are **Boolean functions**. In block ciphers, one usually uses vectorial Boolean functions or Substitution boxes (S-boxes) (Knudsen and Robshaw, 2011) where input and output dimensions are comparable (e.g. the same as in the AES cipher (Daemen and Rijmen, 2002) or similar like in the DES cipher (FIPS, 1999)). On the other hand, in stream ciphers more common are either Boolean functions or S-boxes where output dimension is strictly smaller than the input dimension (Carlet, 2010).

To build such nonlinear elements, one has three main options on his disposal: algebraic constructions, random search, and heuristics (Picek et al., 2014). Heuristic techniques are well visited with works spanning from simulated annealing (Clark and Jacob, 2000) and evolutionary algorithms (Picek et al., 2015b), to particle swarm optimization (Mariot and Leporati, 2015). All those methods have in common that they are highly successful and give results comparable to algebraic constructions.

* Corresponding author.

http://dx.doi.org/10.1016/j.engappai.2016.11.002
Received 15 January 2016; Received in revised form 17 October 2016; Accepted 3 November 2016
Available online xxxx
0952-1976/© 2016 Elsevier Ltd. All rights reserved.

Please cite this article as: Picek, S., Engineering Applications of Artificial Intelligence (2016), http://dx.doi.org/10.1016/j.engappai.2016.11.002

In the rest of this paper, we concentrate only on Boolean functions, i.e. where the output dimension of a function equals one. As the construction principle, we use heuristics, more precisely immunological inspired computation where we experiment with the clonal selection algorithm and the optimization immune algorithm. Both algorithms are a special class of immune algorithms and are inspired by the clonal selection principle of the human immune system (Burnet, 1959; Castro and Timmis, 2002). From the evolutionary algorithms paradigm, we experiment with genetic algorithms (GAs) and evolution strategies (ES). We note that this work presents the first step when investigating the effectiveness of immunological algorithms for the construction of Boolean functions with good cryptographic properties. By doing so, we explore two different representations of solutions, namely the bitstring and the floating point representation and compare the aforesaid algorithms. As a benchmark suite, we use four fitness functions and a number of Boolean function sizes.

1.1. Motivation and contributions

Historically, Boolean functions were mostly used in combination with Linear Feedback Shift Registers (LFSRs) in two models – the filter generator and the combiner generator. In a filter generator, the output is obtained by a nonlinear combination of a number of positions in one longer LFSR while in a combiner generator, several LFSRs are used in parallel and their output is the input for a Boolean function. Such Boolean functions need to fulfill a number of properties: to be balanced, with high nonlinearity, large algebraic degree, large algebraic immunity, large fast algebraic immunity, and large correlation immunity (in the case of combiner generators) (Carlet, 2010). However, obtaining large enough Boolean functions with good values of the aforesaid properties is not a trivial task. First, to illustrate the size of the problem, we give the corresponding search space sizes in Table 1.

However, in our experiments we concentrate only on two properties out of those listed above – balancedness and nonlinearity. By doing so, we are able to concentrate more on the comparison between different heuristic techniques as well as with related work. Therefore, although talking about constructing Boolean functions with good cryptographic properties, we emphasize that here we consider these problems as benchmarks where the end goal are Boolean functions possessing certain properties. Such obtained functions could, but will not in general, have all necessary properties of a sufficient quality.

In this paper, we give two main contributions and a number of smaller ones. The first contribution represents an experimental investigation on the efficiency of immunological algorithms when designing Boolean functions with good cryptographic properties. Although a simple application of a new algorithm to a well-researched problem does not necessarily constitute a significant contribution, we believe here to be relevant since the whole area of applying immunological algorithms to cryptography is a new one. Moreover, by switching the paradigm from evolutionary algorithms to immunological algorithms we should be able to give an insight whether further improvements are possible by simply changing the algorithms. Indeed, by doing so, we try to give knowledge that is not domain specific and tells us whether for performance increase is more important the choice of algorithm, representation, or fitness function. The second contribution is that we are the first, as far as we know, to experiment with the floating point representation of solutions to evolve cryptographic Boolean functions. Finally, in this work we investigate larger sizes of Boolean functions than can usually be found in the literature, where we go up to Boolean functions with 16 variables. We note that by doing so, we experiment with sizes that have practical importance since 13 inputs is considered the minimal size of a Boolean function to be useful in cryptography (Carlet, 2010). To strengthen our experimental results, we compare two immunological algorithms with two well-researched evolutionary algorithms for both bitstring and floating point representation. Naturally, in order to examine the relevance of algorithms' parameters, we conduct a detailed tuning phase. Finally, all experiments are conducted on four fitness functions where two are well-known ones from the literature, and two are modifications that provide more gradient in the search process.

Table 1
The search space size for various input size n.

n	6	8	10	12	14	16
#	2^{64}	2^{256}	$2^{1\,024}$	$2^{4\,096}$	$2^{16\,384}$	$2^{65\,536}$

1.2. Outline of the paper

This paper is divided into two main parts: the description of our experimental setup in Section 5, and the obtained results in Section 6. However, first we start with a short introduction to Boolean functions, their properties, representations, and notation we use in Section 2. Then, after we introduced the readers with basic notions and properties, we give a selection or relevant works in Section 3. Next, Section 4 introduces fitness functions we use in our experiments. Detailed explanation about the representation perspectives for heuristic algorithms is presented in Section 5.1. The parameter tuning phase is discussed in Section 6.1 and the results for the first and second fitness functions in Sections 6.2 and 6.3, respectively. In Section 6.4, we give a short discussion about the efficiency of the algorithms used as well as some potential future research directions. Finally, we conclude in Section 7.

2. Boolean functions properties and representations

Let $n, m \in \mathbb{N}$. We denote the set of all n-tuples of the elements in the field $\mathbb{F}_2$ as $\mathbb{F}_2^n$, where $\mathbb{F}_2$ is the Galois field with two elements. The set $\mathbb{F}_2^n$ represents all binary vectors of length n and it can be viewed as a $\mathbb{F}_2$ – vectorspace (Carlet, 2010). The inner product of vectors $\vec{a}$ and $\vec{b}$ is denoted as $\vec{a}\cdot\vec{b}$ and it equals $\vec{a}\cdot\vec{b} = \oplus_{i=1}^{n} a_i b_i$. Here, "$\oplus$" represents addition modulo two. The Hamming weight (HW) of a vector $\vec{a}$, where $\vec{a} \in \mathbb{F}_2^n$, is the number of non-zero positions in the vector. An (n, m) – function is any mapping F from $\mathbb{F}_2^n$ to $\mathbb{F}_2^m$ where Boolean functions represent m=1 case.

2.1. Boolean function representations

A Boolean function f on $\mathbb{F}_2^n$ can be uniquely represented by a truth table (TT), which is a vector $(f(\vec{0}),\ldots,f(\vec{1}))$ that contains the function values of f, ordered lexicographically (Carlet, 2010).

The second unique representation of a Boolean function is the Walsh–Hadamard transform W_f that measures the correlation between $f(\vec{x})$ and all linear functions $\vec{a}\cdot\vec{x}$ (Carlet, 2010; Forrié, 1990). The Walsh–Hadamard transform of a Boolean function f equals:

$$W_f(\vec{a}) = \sum_{\vec{x}\in\mathbb{F}_2^n} (-1)^{f(\vec{x})\oplus\vec{a}\cdot\vec{x}}. \quad (1)$$

There exist other unique representations of Boolean functions like the algebraic normal form or numerical normal form (Carlet, 2010). However, since we do not work with those representations nor we need properties that are usually derived from those representations, we omit the details and refer interested readers to Carlet (2010).

2.2. Boolean function properties and bounds

A Boolean function f is **balanced** if the Walsh-Hadamard spectrum of a vector $\vec{0}$ equals zero (Preneel et al., 1991):

$$W_f(\vec{0}) = 0. \quad (2)$$

Download English Version:

https://daneshyari.com/en/article/4942754

Download Persian Version:

https://daneshyari.com/article/4942754

Daneshyari.com