Genetic Algorithms Evolving Quasigroups with Good Pseudorandom Properties

Václav Snášel[1], Jiří Dvorský[1], Eliška Ochodková[1], Pavel Krömer[1], Jan Platoš[1], and Ajith Abraham[2]

[1] Department of Computer Science
Faculty of Electrical Engineering and Computer Science
VŠB – Technical University of Ostrava
17. listopadu 15, 708 33 Ostrava – Poruba, Czech Republic
{pavel.kromer,vaclav.snasel,jan.platos}@vsb.cz
[2] Center of Excellence for Quantifiable
Quality of Service, Norwegian
University of Science and Technology
O.S. Bragstads plass 2E,
N-7491 Trondheim, Norway
ajith.abraham@ieee.org

Abstract. Quasigroups are a well-known combinatorial design equivalent to more familiar Latin squares. Because all possible elements of a quasigroup occur with equal probability, it makes it an interesting tool for the application in computer security and for production of pseudorandom sequences. Prior implementations of quasigroups were based on look-up table of the quasigroup, on system of distinct representatives etc. Such representations are infeasible for large quasigroups. In contrast, presented analytic quasigroup can be implemented easily. It allows the generation of pseudorandom sequences without storing large amount of data (look-up table). The concept of isotopy enables consideration of many quasigroups and genetic algorithms allow efficient search for good ones.

1 Introduction

The need for random and pseudorandom sequences arises in many applications, e.g. in modeling, simulations, and of course in cryptography. Pseudorandom sequences are the core of stream ciphers. They are popular due to their high encryption/decryption speed. Their simple and cheap hardware design is often preferred in real-world applications. The design goal in stream ciphers is to efficiently produce pseudorandom sequences - keystreams (i.e. sequences that possess properties common to truly random sequences and in some sense are "indistinguishable" from these sequences).

The use of quasigroups and quasigroup string transformations is a recent but successful tendency in cryptography and coding [17]. With quasigroups in the hearth of advanced cryptosystems and hash functions, a need to find good quasigroups becomes hot topic.

D. Taniar et al. (Eds.): ICCSA 2010, Part III, LNCS 6018, pp. 472–482, 2010.

Quasigroups and its applications in computer security were studied e. g. in [3]. A design of pseudorandom sequence generator (PRSG) based on quasigroup operation was presented in [4]. The authors have performed an extensive analysis of 2^{16} randomly chosen quasigroups of the orders 5, 6, 7, 8, 9 and 10 and concluded that different quasigroups produce pseudorandom sequences with different period (i.e. the number of elements after which the pseudorandom sequence starts to repeat). They have show that only a small number of quasigroups feature very large value of coefficient of period growth, a property that significantly affects the period of generated pseudorandom sequence[4]. This results encourage research of efficient methods for search for good quasigroups in the field of pseudorandom generators and cryptography.

Genetic algorithms are probably the most popular and wide spread member of the class of evolutionary algorithms (EA). EAs build a class of iterative stochastic search and optimization methods based on mimicking successful optimization strategies observed in nature [7,8,16,21]. The essence of EAs lies in the emulation of Darwinian evolution, utilizing the concepts of Mendelian inheritance for practical applications in computer science [7].

EAs operate with a population of artificial individuals (chromosomes) encoding potential problem solutions. Encoded individuals are evaluated using a carefully selected objective function which assigns a fitness value to each individual. The fitness value represents the quality (relative ranking) of each individual as a solution to given problem. Competing individuals explore in a highly parallel manner problem domain towards an optimal solution [16].

2 Quasigroups

Definition 1. *A quasigroup is a pair $(Q, \circ)$, where $\circ$ is a binary operation on (finite) set Q such that for all not necessarily distinct $a, b \in Q$, the equations*

$$a \circ x = b \text{ and } y \circ a = b.$$

have unique solutions.

The fact that the solutions are unique guarantees that no element occurs twice in any row or column of the table for $(\circ)$. However, in general, the operation $(\circ)$ is neither a commutative nor an associative operation.

Quasigroups are equivalent to more familiar Latin squares. The multiplication table of a quasigroup of order q is a Latin square of order q, and conversely, as it was indicated in [6,9,23], every Latin square of order q is the multiplication table of a quasigroup of order q.

Definition 2. *Let $A = \{a_1, a_2, \ldots, a_n\}$ be a finite alphabet, a $n \times n$ Latin square L is a matrix with entries $l_{ij} \in A$, $i, j = 1, 2, \ldots, n$, such that each row and each column consists of different elements of A.*

For $i, j, k \in A$ the ordered triple $(i, j; k)$ is used to represent the occurrence of element k in cell (i, j) of the Latin square. So a Latin square may be represented by the set $\{(i, j; k) |$ entry k occurs in cell (i, j) of the Latin square $L.\}$

All reduced Latin squares of order n are enumerated for $n \leq 11$ [19]. Let L_n be the number of Latin squares of order n, and let R_n be the number of reduced Latin squares of order n. It can be easily seen that

$$L_n = n!(n-1)!R_n.$$

Number of distinct Latin squares of a given order grows exceedingly quickly with the order and there is no known easily-computable formula for the number of distinct Latin squares. The problem of classification and exact enumeration of Latin squares of order greater than 11 probably still remains unsolved. Thus, there are more than 10^{90} quasigroups of order 16 and if we take an alphabet $L = \{0 \ldots 255\}$ (i.e. data are represented by 8 bits) there are at least $256!255! \ldots 2! > 10^{58000}$ quasigroups.

Multiplication in quasigroups has an important property; it is proven that each element occurs exactly q times among the products of two elements of Q, q^2 times among the products of three elements of Q and, generally q^{t-1} among the products of t elements of Q. Since there are q^t possible ordered products of t elements of Q, this shows that each element occurs equally often among these q^t products (see [10]).

2.1 Pseudorandom Sequence Generator Based on Quasigroups

The construction of pseudorandom sequence generator based on quasigroup operations was described in [4]. In this paper we use simplified design of PRSG to verify the proposed quasigroup optimization method.

Definition 3. *Let $(Q, \circ)$ be a quasigroup and Q^+ be a set of all nonempty words formed by the elements $q_i \in Q$, $1 \leq i \leq n$.*

For a fixed $a \in Q$ let the pseudorandom sequence $y_1, y_2, \ldots, y_n$ be defined as

$$\begin{aligned} y_1 &= a \circ a \\ y_2 &= a \circ y_1 \\ &\vdots \\ y_i &= a \circ y_{i-1} \end{aligned}$$

2.2 Isotopism of Quasigroups

Definition 4. *Let $(G, \cdot)$, $(H, \circ)$ be two quasigroups. An ordered triple (π, ρ, ω) of bijections π, ρ, ω of the set G onto set H is called an* isotopism *of $(G, \cdot)$ upon $(H, \circ)$ if $\forall u, v \in G, \pi(u) \circ \rho(v) = \omega(u \cdot v)$. Quasigroups $(G, \cdot)$, $(H, \circ)$ are said to be* isotopic.

We can imagine an isotopism of quasigroups as a permutation of rows and columns of quasigroup's multiplication table.

Example 1. Consider a multiplication table for a quasigroup isotopic to the quasigroup of modular subtraction, with operation $\circ$ defined as $a \circ b = (a + n - b) \bmod n$):

∘	0	1	2	3
0	0	3	2	1
1	2	1	0	3
2	1	0	3	2
3	3	2	1	0

The table was created from table of modular subtraction. The second and the third rows were exchanged. Permutations π, ρ were identities and $\omega = [0213]$. A multiplication in this quasigroup can be illustrated by e.g. $1 \circ 0 = \omega(1) \circ 0 = 2 \circ 0 = 2$.

Starting with the quasigroup of modular subtraction, we can explore a large class of quasigroups isotopic to the quasigroup of modular subtraction [11,22]. This allows us to utilize quasigroups with very large number of elements without the necessity of their storage in program memory. The multiplication in such isotopic quasigroup is defined as follows:

$$a \circ b = \pi^{-1}((\omega(a) + n - \rho(b)) \bmod n). \tag{1}$$

We call the quasigroup defined by its multiplication formula and three selected permutations an *analytic quasigroup*[15,27].

The notion of analytic quasigroup enables efficient work with large quasigroups. Previos studies in this field used mostly quasigroups of small order [14], or just a small parts of certain quasigroup were utilized mainly as a key for Message Authentication Code. Such small quasigroups are represented as look-up tables in main memory. Larger quasigroup of order 2^{256} is used by NIST's SHA-3 competition[1] candidate, hash function Edon$\mathcal{R}$ [13].

The properties of one analytic quasigroup isotopic to the quasigroup of modular subtraction were studied in [15]. The quasigroup was created using three static functions that divided the sequence of n elements of the quasigroup into several parts. The parts were rotated in various directions and exchanged among themselves. It was shown that the investigated quasigroup has some faults in its properties.

2.3 Constructing Quasigroups Isotopic to the Quasigroup of Modular Subtraction

Consider a quasigroup on the length n defined by multiplication $a \circ b = (a + n - b) \bmod n)$. Then three permutations π, ρ, ω must be chosen in order to implement isotopic quasigroup, whose multiplication will be defined by (1).

Obviously, there is $n!$ different permutations of n elements. Because three independent permutations are used to define any isotopic quasigroup, there are $n!n!n!$ possible choices of π, ρ and ω.

Permutations of elements cannot be sought for an analytic quasigroup directly, because its elements are not stored in memory. Instead, the permutation needs to be implemented as a function of an element of Q. One way to achieve this goal is the use of bit permutation.

[1] http://csrc.nist.gov/groups/ST/hash/sha-3/index.html

A quasigroup over a set of n elements requires $log_2(n)$ bits to express each element. Each permutation of bits in the element representation represents also a permutation of all elements of the quasigroup (if n is a power of 2). Bit permutation can be implemented easily as a function of $q \in Q$.

The bit permutation is an elegant way of implementing permutations over n elements of Q. Although it enables us to explore only a fragment ($log_2(n)!log_2(n)!$ $log_2(n)!$) of all possible permutation triples over the quasigroup of n elements, it is useful because it does not require all n elements in main memory and therefore fits into the framework of analytic quasigroups.

Bit permutations are computationaly more expensive than the static functions used to implement permutation in [15]. However, there are ongoing efforts to implement bit permutation instructions in hardware, which would improve the performance of the proposed algorithm significantly [12].

3 Genetic Algorithms

Genetic algorithms are generic and reusable population-based metaheuristic soft optimization method [5,16,21]. GAs operate with a population of chromosomes encoding potential problem solutions. Encoded individuals are evaluated using a carefully selected domain specific objective function which assigns a fitness value to each individual. The fitness value represents the quality of each candidate solution in context of the given problem. Competing individuals explore the problem domain towards an optimal solution [16]. The solutions might be encoded as binary strings, real vectors or more complex, often tree-like, hierarchical structures (subject of genetic programming [18]). The encoding selection is based on the needs of particular application area.

The emulated evolution is driven by iterative application of genetic operators. Genetic operators algoritmize principles observed in natural evolution. The crossover operator defines a strategy for the exchange of genetic information between parents (sexual reproduction of haploid organisms) while the mutation operator introduces the effect of environment and randomness (random perturbation of genetic information). Other genetic operators define e.g. parent selection strategy or the strategy to form new population from the current one. Genetic operators and algorithm termination criteria are the most influential parameters of every evolutionary algorithm. The operators are subject to domain specific modifications and tuning [21]. The basic workflow of the standard generational GA is shown in Fig. 1.

Many variants of the standard generational GA have been proposed. The differences are mostly in particular selection, crossover, mutation and replacement strategy [16].

In the next section, we present genetic algorithm for the search for good analytic quasigroups. It is an extended version of the initial GA for quasigroup evolution introduced in[27]. In this study, we present modified GA for quasigroup optimization with reengineered fitness function to find quasigroups that produce good pseudoramdom sequences.

1 Define objective (fitness) function and problem encoding;
2 Encode initial population P of possible solutions as fixed length strings;
3 Evaluate chromosomes in initial population using objective function;
4 **while** *Termination criteria not satisfied* **do**
5 Apply selection operator to select parent chromosomes for reproduction: $sel(P_i) \rightarrow parent1$, $sel(P_i) \rightarrow parent2$;
6 Apply crossover operator on parents with respect to crossover probability to produce new chromosomes: $cross(pC, parent1, parent2) \rightarrow \{offspring1, offspring2\}$;
7 Apply mutation operator on offspring chromosomes with respect to mutation probability: $mut(pM, offspring1) \rightarrow offspring1$, $mut(pM, offspring2) \rightarrow offspring2$;
8 Create new population from current population and offspring chromosomes: $migrate(offspring1, offsprig2, P_i) \rightarrow P_{i+1}$;
9 **end**

Fig. 1. A summary of genetic algorithm

4 Genetic Search for Analytic Quasigroups

The genetic algorithm for the search for analytic quasigroup is defined by encoding of the candidate solutions and fitness function to evaluate chromosomes.

4.1 Encoding

As noted in section 2.3, any analytic quasigroup isotopic to quasigroup of modular subtraction is defined by three permutations. Such permutation triple represents a problem solution and should be mapped to one GA chromosome. Permutations can be for the purpose of genetic algorithms encoded using several strategies. In this study, we use random key encoding.

Random key (RK) encoding is an encoding strategy available for problems involving permutation optimization [24]. In random key encoding, the permutation is represented as a string of real numbers (random keys), whose relative position changes after sorting corresponds to the permutation index. An example or random key encoding is shown in (2).

$$\Pi_5 = \begin{pmatrix} 0.2 & 0.3 & 0.1 & 0.5 & 0.4 \\ 2 & 3 & 1 & 5 & 4 \end{pmatrix} \tag{2}$$

To encode a quasigroup (isotopic to the quasigroup of modular subtraction) of the length $n = 2^l$, we use a vector of $3l$ real numbers $v = (v_1, \ldots, v_{l-1}, v_l, \ldots v_{2l-1}, v_{2l}, \ldots, v_{3l})$. The vector is interpreted as three concatenated RK encoded permutations of the length l.

This encoding allows us to use traditional implementations of genetic operators, such as n-point crossover and mutation. Crossover was implemented as mutual exchange of genes between selected parents and mutation was implemented as a replacement of gene with a uniform random number from the interval $[0, 1]$.

4.2 Fitness Function

Fitness function is used to rank candidate solutions among themselves. There is a number of methods to measure randomness and evaluate pseudorandom sequences. The DIEHARD battery of tests[2] and NIST test battery[3] are de-facto standard tools to detect non-randomness in pseudorandom sequences. They can be used to find sound statistical evidence that a pseudorandom sequence feature non-random patterns. Many randomness tests in both mentioned test suites are based on χ^2 test defined by[1]:

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i} \tag{3}$$

where O_i is the observed count of occurences of $i-$th number and E_i is expected number of occurences of $i - th$ number.

In this work, we have used the χ^2 statistics of the frequency of numbers produced by the generator described in def. 3 as the goodness of fit measure of the generated pseudorandom sequence. The χ^2 statistics was computed after 10000 numbers were produced by the generator. We expect the same uniform probability of every symbol $E_i = \frac{n}{no\_of\_symbols}$ in the pseudorandom sequence.

The fitness function f was defined as:

$$f = \frac{1}{\chi^2} \tag{4}$$

so that better (i.e. producing more uniform sequence of symbols) pseudorandom generator was characterised by higher fitness value.

5 Experimental Optimization

This section summarizes experimental genetic search for a quasigroups isotopic to the quasigroups of modular subtraction with the dimensions 128, 512 and 2048 respectively. We have implemented genetic algorithm with permutation encoding and fitness function as defined above. The parameters of the algorithm (probability of mutation, probability of crossover etc.) were selected after initial tuning of the algorithm. The paramteres are summarized in Table (1).

Table 1. The settings of genetic algorithm for quasigroup search

Parameter	value
Population size	20
Probability of mutation (P_M)	0.02
Probability of recombination (P_C)	0.8
Selection operator	elitist
Max number of generations	1000

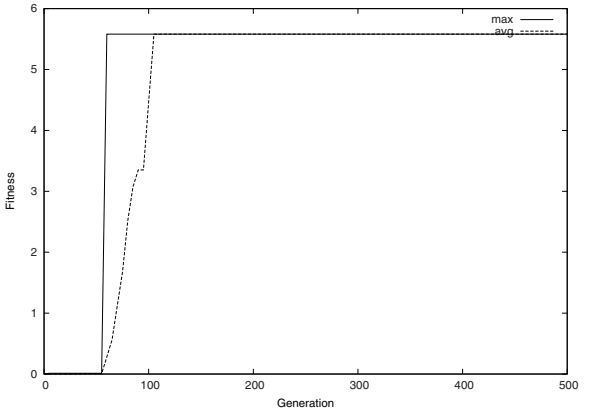

Fig. 2. Maximum and average fitness during an optimization of the quasigroup of dimension 128

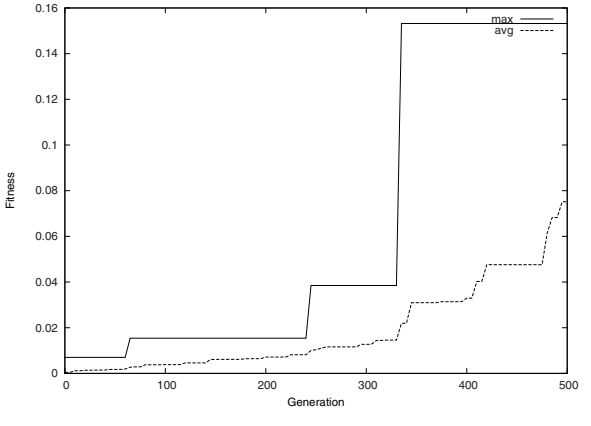

Fig. 3. Maximum and average fitness during an optimization of the quasigroup of dimension 512

The progress of the genetic optimization of analytic quasigroups of the dimensions 128, 512 and 2048 is shown in Fig. 2, Fig. 3 and Fig. 4 respectively.

The evolutionary search has found better quasigroups (in terms of used fitness function) in all experiments. For the quasigroup of length 128, it improved fitness from average 0.00011 in the randomly generated initial population to 5.58. For quasigroup of the length 512, the average initial fitness of 0.00044 was im-

[2] http://www.stat.fsu.edu/pub/diehard/
[3] csrc.nist.gov/rng/

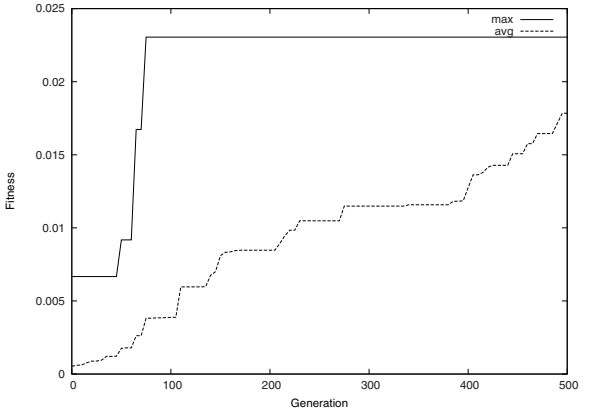

Fig. 4. Maximum and average fitness during an optimization of the quasigroup of dimension 2048

proved to 0.153 and for quasigroup of the length 2048 improved the optimization procedure fitness from 0.00053 to 0.023. In all cases was the final improvement achieved before generation 500.

The results demonstrate that genetic algorithms can improve randomly selected quasigroups (isotopic to the quasigroup of modular subtraction) towards properties defined by used fitness function. The resulting quasigroups are better PRSG generators in terms of normality of symbol frequency in the generated pseudo random sequence. Therefore, the optimized quasigroups are more suitable for applications in computer security.

6 Conclusions

In this paper was described a genetic algorithm for optimization of an analytic quasigroup. The genetic algorithm looks for good bit permutations that are used to construct analytic quasigroups with desired properties. Both, the analytic quasigroup and bit permutation, do not rely on the lookup table of the quasigroup stored in memory. Therefore, large quasigroups can be used and optimized efficiently.

A χ^2 based goodness of fit statistics evaluating the randomness of pseudorandom sequence produced by the quasigroup operation was used as a basis of the fitness function. When used as a pseudorandom number generator, the optimized quasigroups generate better pseudorandom sequences than randomly chosen quasigroups of the same length.

Conducted numerical experiments suggest that the genetic algorithm is a good tool to find optimized quasigroups. In our future work, we aim to find more appropriate fitness function that will allow us to find quasigroups suitable for cryptography and also other application areas.

Acknowledgment

This work was supported by the Czech Science Foundation under the grant no. 102/09/1494.

References

1. Knuth, D.E.: The art of computer programming. In: Seminumerical Algorithms, 3rd edn., vol. 2, Addison-Wesley/Longman Publishing Co., Inc. (1997)
2. Marsaglia, G., Tsang, W.W.: Some Difficult-to-pass Tests of Randomness. Journal of Statistical Software 7(i03)
3. Markovski, S.: Quasigroup String Processing and Applications in Cryptography. In: Proceedings 1st Conference of Mathematics and Informatics for Industry, Thessaloniki, Greece, pp. 278–290 (2003)
4. Dimitrova, V., Markovski, J.: On quasigroup pseudo random sequence generator. In: Manolopoulos, Y., Spirakis, P. (eds.) Proc. of the 1-st Balkan Conference in Informatics, Thessaloniki, November 2004, pp. 393–401 (2004)
5. Bäck, T., Hammel, U., Schwefel, H.-P.: Evolutionary computation: comments on the history and current state. IEEE Transactions on Evolutionary Computation 1, 3–17 (1997)
6. Belousov, V.D.: Osnovi teorii kvazigrup i lup, Nauka, Moscow (1967) (in Russian)
7. Bodenhofer, U.: Genetic Algorithms: Theory and Applications. Lecture Notes, Fuzzy Logic Laboratorium Linz-Hagenberg (Winter 2003/2004)
8. Dianati, M., Song, I., Treiber, M.: An introduction to genetic algorithms and evolution strategies, technical report, University of Waterloo, Ontario, N2L 3G1, Canada (July 2002)
9. Dénes, J., Keedwell, A.: Latin Squares and their Applications. In: Akadémiai Kiadó, Budapest. Academic Press, New York (1974)
10. Dénes, J., Keedwell, A.: A new authentication scheme based on Latin squares. Discrete Mathematics (106/107), 157–161 (1992)
11. Dvorský, J., Ochodková, E., Snášel, V.: Hash Functions Based on Large Quasigroups. In: Proceedings of Velikonoční kryptologie, Brno, pp. 1–8 (2002)
12. Hilewitz, Y., Shi, Z.J., Lee, R.B.: Comparing fast implementations of bit permutation instructions. In: Conference Record of the Thirty-Eighth Asilomar Conference on Signals, Systems and Computers, Pacific Grove, California, USA, November 2004, pp. 1856–1863 (2004)
13. Gligoroski, D., et al.: EdonR cryptographic hash function. Submition to NIST's SHA-3 hash function competition (2008), `http://csrc.nist.gov/groups/ST/hash/sha-3/index.html`
14. Gligoroski, D., Markovski, S., Kocarev, L., Svein, J.: The Stream Cipher Edon80. In: Robshaw, M.J.B., Billet, O. (eds.) New Stream Cipher Designs. LNCS, vol. 4986, pp. 152–169. Springer, Heidelberg (2008)
15. Snášel, V., Abraham, A., Dvorský, J., Krömer, P., Platoš, J.: Hash functions based on large quasigroups. In: Allen, G., Nabrzyski, J., Seidel, E., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds.) ICCS 2009, Part I. LNCS, vol. 5544, pp. 521–529. Springer, Heidelberg (2009)
16. Jones, G.: Genetic and evolutionary algorithms. In: von Rague, P. (ed.) Encyclopedia of Computational Chemistry. John Wiley and Sons, Chichester (1998)

17. Knapskog, S.J.: New cryptographic primitives. In: CISIM 2008: Proceedings of the 2008 7th Computer Information Systems and Industrial Management Applications, pp. 3–7. IEEE Computer Society, Washington (2008)
18. Koza, J.: Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems. Technical Report STAN-CS-90-1314, Dept. of Computer Science, Stanford University (1990)
19. McKay, B.D., Wanless, I.M.: On the Number of Latin Squares. Journal Annals of Combinatorics 9(3), 335–344 (2005)
20. Merkle, R.C.: Secrecy, authentication, and public key systems. Stanford Ph.D. thesis, pp. 13-15 (1979), http://www.merkle.com/papers/Thesis1979.pdf
21. Mitchell, M.: An Introduction to Genetic Algorithms. MIT Press, Cambridge (1996)
22. Ochodková, E., Snášel, V.: Using Quasigroups for Secure Encoding of File System. In: Proceedings of the International Scientific NATO PfP/PWP Conference Security and Information Protection 2001, Brno, Czech Republic, May 9-11, pp. 175–181 (2001)
23. Smith, J.D.H.: An introduction to quasigroups and their representations. Chapman & Hall/CRC (2007)
24. Snyder, L.V., Daskin, M.S.: A random-key genetic algorithm for the generalized traveling salesman problem. European Journal of Operational Research 174(1), 38–53 (2006)
25. Vojvoda, M.: Cryptanalysis of One Hash Function Based on quasigroup. In: Conference Mikulášská kryptobesídka, Praha, pp. 23–28 (2003)
26. TREC Web Corpus: GOV (2009), http://ir.dcs.gla.ac.uk/test_collections/govinfo.html
27. Snášel, V., Abraham, A., Dvorský, J., Ochodková, E., Platoš, J., Krömer, P.: Searching for Quasigroups for Hash Functions with Genetic Algorithms. In: Proceedings of the 2009 World Congress on Nature & Biologically Inspired Computing, pp. 367–372. IEEE Computer Society, Los Alamitos (2009)