Evolving Quasigroups by Genetic Algorithms*

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

[1] Department of Computer Science, FEECS, VŠB – Technical University of Ostrava
17. listopadu 15, 708 33 Ostrava – Poruba, Czech Republic
`{vaclav.snasel,jiri.dvorsky,eliska.ochodkova, pavel.kromer,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. Most implementations of quasigroups are based on look-up table of the quasigroup, on system of distinct representatives etc. Such representations are infeasible for large quasigroups. An analytic quasigroup is a recent concept that allows usage of certain quasigroups without the need of look-up table. The concept of isotopy enables consideration of many quasigroups and genetic algorithms allow efficient search for good ones. In this paper we describe analytic quasigroup and genetic algorithms for its optimization.

1 Introduction

Random and pseudorandom sequences can be used in many applications, e.g. in modeling, simulations, and of course in cryptography. Pseudorandom sequences are the core of stream ciphers. 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 [12]. With quasigroups in the hearth of advanced cryptosystems and hash functions, a need to find good quasigroups becomes hot topic.

Quasigroups and its applications in computer security were studied e. g. in [2]. A design of pseudorandom sequence generator (PRSG) based on quasigroup operation was presented in [3]. The authors have performed an extensive analysis of

* This paper was partially supported by GACR 205/09/1079 grant.

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 [3]. 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 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 [4,5,18], 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 $n \leq 11$ [14]. 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 [6]).

2.1 Isotopism of quasigroups

Definition 3. *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 $a \circ b = (a + n - b) \bmod n)$:

$\circ$	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 [7,17]. 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* [11,21].

The notion of analytic quasigroup enables efficient work with large quasigroups. Previos studies in this field used mostly quasigroups of small order [10], or just a small parts of certain quasigroup were utilized e.g. 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 candidate, hash function Edon$\mathcal{R}$ [9].

The properties of one analytic quasigroup isotopic to the quasigroup of modular subtraction were studied in [11]. 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.

2.2 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.

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 [11]. However, there are ongoing efforts to implement bit permutation instructions in hardware, which would improve the performance of the proposed algorithm significantly [8].

3 Genetic algorithms

Genetic algorithms (GAs) are generic and reusable population-based metaheuristic soft optimization method [16]. 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 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). The basic workflow of the standard generational GA is shown in algorithm 1.

Algorithm 1: A summary of genetic algorithm

```
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
```

Many variants of the standard generational GAs 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 [21]. In this study, we introduce a new fitness function based on associativity and commutativity that is used for quasigroup optimization.

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 subsection 2.2, 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 suitable for problems involving permutation optimization [19]. 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

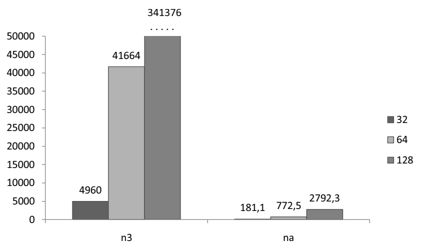
The fitness function f we propose in this work is based on commutativity and associativity in quasigroup.

$$f(n, n_a, n_c, \alpha) = \alpha \frac{n_2 - n_c}{n_2} + (1 - \alpha) \frac{n_3 - n_a}{n_3} \tag{3}$$

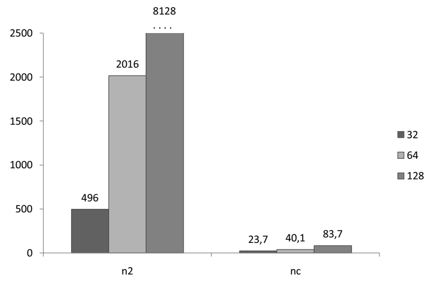
In (3), n stands for the order of the quasigroup, n_2 stands for the number of all combinations of 2 elements out of n, n_3 represents all combinations of 3 elements out of n, n_c stands for the number of element pairs that have the commutativity property and n_a represents the number of element triples that have the associativity property. The coefficient $\alpha \in [0, 1]$ is used to prioritize between commutativity and associativity. The value of fitness function is 0 for $n_c = n_2$ and $n_a = n_3$ at the same time and 1 for $n_c = 0$ and $n_a = 0$. Informally, we can say that it seeks for a quasigroup that has “low associativity“ and “low commutativity“.

5 Experimental optimization

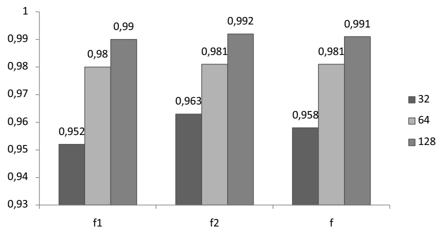
We have investigated the associativity and commutativity in randomly generated quasigroups isotopic to the quasigroup of modular subtraction of orders 32, 64 and 128 respectively. The average values of n_a, n_c and fitness in random quasigroups are shown in Figure 1.



(a) Average n_a in random quasigroups ($n3$ illustrates the number of all combinations of 3 elements out of n).



(b) Average n_c in random quasigroups($n2$ illustrates the number of all combinations of 2 elements out of n).



(c) Average fitness in random quasigroups. $f1$ corresponds to $\frac{n_2-n_c}{n_2}$, $f2$ to $\frac{n_3-n_a}{n_3}$ and $f = 0.5f1 + 0.5f2$

Fig. 1. Average values of n_a, n_c and fitness in random quasigroups of order 32, 64 and 128.

In the next step, we have performed genetic search for better (in terms of low associativity and low commutativity) quasigroups isotopic to the quasigroups of modular subtraction with the dimensions 32, 64 and 128 respectively. We have implemented genetic algorithm with permutation encoding and fitness function as defined above. The parameters of the algorithm (α, P_M, P_C etc.) were selected after initial tuning of the algorithm. The paramteres are summarized in Table 1.

Beacuse genetic algorithm is a stochastic method, every experiment was repeated 10 times and presented results are average values after 10 independent runs. The values of optimized fitness, n_c and n_a are shown in Table 2. A comparison of optimized values with n_a, n_c and fitness in random quasigroups isotopic

Table 1. The settings of genetic algorithm for quasigroup search

Parameter	value
Fitness function coefficient α	0.5
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

to the quasigroups of modular subtraction are illustrated in Figure 2. We can see that in every experiment (for every quasigroup dimension), the genetic optimization process delivered a quasigroup better than random one.

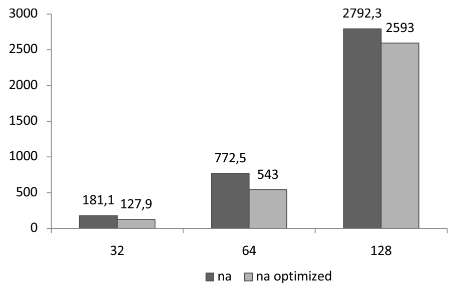
Table 2. Results for average evolved quasigroup of the dimension 32, 64 and 128 respectively.

	Property		
Dimension	n_c	n_a	fitness
32	4.3	127.9	0.983
64	5	543	0.992
128	24	2593	0.995

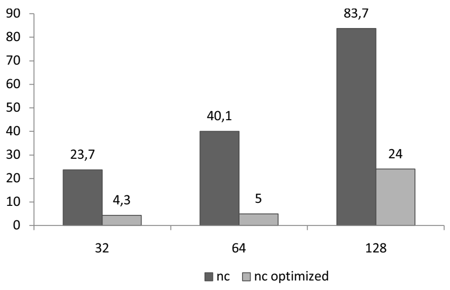
6 Conclusions

In this paper was described a genetic algorithm for optimization of an analytic quasigroup. The genetic algorithm performs a search for good bit permutations that are then 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.

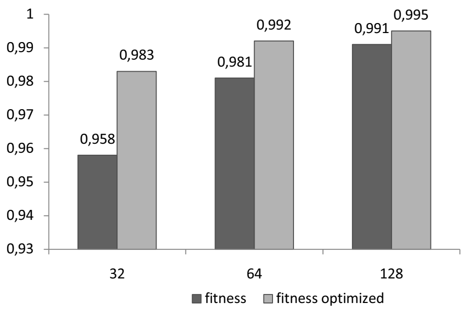
The fitness function we use in this paper is based on associativity and commutativity in quasigroups. It triggers a search for quasigroups that have "low associativity" and "low commutativity". In a numerical experiment, we have been able to find quasigroups with better properties than random ones have. The drawback of this method is at this point its computational expensiveness. In order to evaluate the fitness function, all combinations of 2 elements out of n and all combinations of 3 elements out of n (quasigroup dimension) have to be found and evaluated using quasigroup operation $\circ$. However, the main aim of this work was to verify that genetic algorithm can evolve quasigroups with above average properties.



(a) Average n_a in optimized quasigroups. Lower is better.



(b) Average n_c in optimized quasigroups. Lower is better.



(c) Average fitness in optimized quasigroups. Higher is better.

Fig. 2. Average values of n_a, n_c and fitness in optimized quasigroups of order 32, 64 and 128 compared with n_a, n_c and fitness in random quasigroups.

In our future work, we want to investigate the properties of generated quasigroups, study the fitness function and look for alternative fitness functions. We want to focus on efficient implementation of used genetic algorithms with the utilization of GPGPU. Moreover, we will investigate other successful computational intelligence methods for quasigroup optimization (e.g. differential evolution, ant colony optimization).

References

1. G. Marsaglia and W. W. Tsang, "Some Difficult-to-pass Tests of Randomness", Journal of Statistical Software, volume 7,number i03.
2. S. Markovski, "Quasigroup String Processing and Applications in Cryptography", Proceedings 1st Conference of Mathematics and Informatics for Industry, Thessaloniki, Greece, pp. 278–290, 2003.
3. V. Dimitrova, J. Markovski, "On quasigroup pseudo random sequence generator", Proc. of the 1-st Balkan Conference in Informatics, Y.Manolopoulos and P. Spirakis eds., Thessaloniki, pp. 393–401, Nov 2004.
4. Belousov, V. D. Osnovi teorii kvazigrup i lup (in Russian), Nauka, Moscow, 1967.

5. Dénes, J., Keedwell, A. Latin Squares and their Applications. Akadémiai Kiadó, Budapest; Academic Press, New York (1974)
6. Dénes, J., Keedwell, A. A new authentication scheme based on Latin squares. Discrete Mathematics (106/107) (1992) pp. 157–161
7. J. Dvorský, E. Ochodková, V. Snášel, Hash Functions Based on Large Quasigroups, Proceedings of Velikonoční kryptologie, Brno, 2002, pp. 1–8.
8. Y. Hilewitz, Z. J. Shi, Lee, and 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), pp. 1856–1863, Nov. 2004 2004.
9. D. Gligoroski, 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
10. D. Gligoroski, S. Markovski, L. Kocarev and J. Svein. The Stream Cipher Edon80. The eSTREAM Finalists, *Lecture Notes in Computer Science*, Vol. 4986. pp. 152-169 , 2008
11. V. Snášel, A. Abraham, J. Dvorský, P. Krömer, and J. Platoš, "Hash functions based on large quasigroups.," in *ICCS (1)* (G. Allen, J. Nabrzyski, E. Seidel, G. D. van Albada, J. Dongarra, and P. M. A. Sloot, eds.), vol. 5544 of *Lecture Notes in Computer Science*, pp. 521–529, Springer, 2009.
12. S. J. Knapskog, "New cryptographic primitives," in *CISIM '08: Proceedings of the 2008 7th Computer Information Systems and Industrial Management Applications*, (Washington, DC, USA), pp. 3–7, IEEE Computer Society, 2008.
13. J. Koza, "*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.
14. B. D. McKay and I. M. Wanless. On the Number of Latin Squares. *Journal Annals of Combinatorics*, Issue Vol.ume 9 (2005), No. 3, pp. 335-344.
15. R .C. Merkle, *Secrecy, authentication, and public key systems.* Stanford Ph.D. thesis 1979, pages 13-15.
http://www.merkle.com/papers/Thesis1979.pdf
16. M. Mitchell, *An Introduction to Genetic Algorithms.* Cambridge, MA: MIT Press, 1996.
17. E. Ochodková, V. Snášel, Using Quasigroups for Secure Encoding of File System, Proceedings of the International Scientific NATO PfP/PWP Conference "Security and Information Protection 2001", May 9-11, 2001, Brno, Czech Republic, pp.175–181.
18. J. D. H. Smith, An introduction to quasigroups and their representations, Chapman & Hall/CRC, 2007.
19. L. V. Snyder and M. S. Daskin, "A random-key genetic algorithm for the generalized traveling salesman problem," *European Journal of Operational Research*, vol. 174, no. 1, pp. 38–53, 2006.
20. M. Vojvoda. Cryptanalysis of One Hash Function Based on quasigroup. In *Conference Mikulášská kryptobesídka*, pp. 23-28., Praha, 2003.
21. V. Snášel, A. Abraham, J. Dvorský, E. Ochodková, J. Platoš, P. Krömer, Searching for Quasigroups for Hash Functions with Genetic Algorithms, Proceedings of the 2009 World Congress on Nature & Biologically Inspired Computing, pp. 367 - 372 IEEE Computer Society, 2009.