mathematics

MDPI

Article

Automated Generation of EQ-Algebras through Genetic Algorithms

Hashim Habiballa *, Eva Volna and Martin Kotyrba

Department of Informatics and Computers, University of Ostrava, 30. dubna 22, 70200 Ostrava, Czech Republic; eva.volna@osu.cz (E.V.); martin.kotyrba@osu.cz (M.K.)
* Correspondence: hashim.habiballa@osu.cz

Abstract: This article introduces an approach to the automated generation of special algebras through genetic algorithms. These algorithms can be also used for a broader variety of applications in mathematics. We describe the results of research aiming at automated production of such algebras with the help of evolutionary techniques. Standard approach is not relevant due to the time complexity of the task, which is superexponential. Our research concerning the usage of genetic algorithms enabled the problem to be solvable in reasonable time and we were able to produce finite algebras with special properties called EQ-algebras. EQ-algebras form an alternate truth–value structure for new fuzzy logics. We present the algorithms and special versions of genetic operators suitable for this task. Then we performed experiments with application EQ-Creator are discussed with proper statistical analysis through ANOVA. The genetic approach enables to automatically generate algebras of sufficient extent without superexponential complexity. Our main results include: that elitism is necessary at least for several parent members, a high mutation ratio must be set, optional axioms fulfilment increases computing time significantly, optional properties negatively affect convergence, and colorfulness was defined to prevent trivial solutions (evolution tends to the simplest way of achieving results).

Keywords: EQ-algebra; genetic algorithm; superexponential problem; finite algebra

MSC: 68T20; 68T35

Citation: Habiballa, H.; Volna E.; Kotyrba, M. Automated Generation of EQ-Algebras through Genetic Algorithms. *Mathematics* **2021**, *9*, 861. https://doi.org/10.3390/math9080861

Academic Editor: Theodore E. Simos

Received: 3 May 2021
Accepted: 12 April 2021
Published: 14 April 2021

1. Introduction

Genetic algorithms (GA) are among the basic methods used to solve optimization problems. Particularly the GA are suitable for tasks with uncertain knowledge of the domain or for optimization applications. The basic idea of GA is based on the theory of the adaptation of natural selection in biological systems, according to which only individuals with the best qualities survive. By crossing individuals with appropriate characteristics, descendants are likely to be adapted to successful survival.

This paper presents the results of the use of genetic algorithms to search for the structure of truth values in a new class of algebras called EQ-algebras. A standard solution to this problem with a classical hard-computing approach, for its complexity, appears to be insoluble, even for small algebraic structures. Even if we limit the problem to the finite structures, we will still be facing a problem with an exponential class of complexity. Some of these tasks are even problems with a super-exponential class of complexity. A natural candidate for this purpose can be seen in evolutionary techniques [1]. There is also suitable criterion directly convertible into fitness function based on the fraction of axioms and expressions fulfilling the desired properties of an algebra. Our research, development and experiments concerning the usage of these optimized evolutionary techniques are discussed in the article.

The computer application working through the methods described in the article (software package EQCreator), which produces EQ-algebras (equality based structures

usable as truth values algebras) in a feasible time. The article further describes experimental evaluation on special algebras and their operators with special axioms, which could be useful for various types of fuzzy logics.

This article shows a formulation of the problem at first and also a description of EQ-algebras basic principle. We also summarize former work of several authors using GA for finite algebra structures creation. Additionally, we describe in detail our application of evolutionary principles, especially Genetic Algorithms and the implementation of the presented methods in the form of computer application EQCreator [2]. Finally, we present experiments demonstrating the time efficiency of the presented evolutionary approach as well as the main issues with an impact on the efficiency of the method.

2. Finite Algebras Automated Production

By a *finite algebra* we mean finite set of **algebra elements**, a finite number of algebra **operations (operators)** of various amount of parameters (n-ary operators) and a set of **axioms**, to which the algebra must conform to be well constructed [3].

The specificity of a particular algebra is especially predefined by the set of axioms (constraints) by which the creator (mathematician) is trying to model some system (real-life or artificial). In this article we demonstrate our methods on the specific algebras called EQ-algebras. These ones are especially useful for fuzzy logicians to model alternative fuzzy logic truth value structures based on the notion of fuzzy equality.

The research main motivation was to provide group of prof. Novak from Institute for Research and Applications of Fuzzy Modelling (Univ. of Ostrava) tool for generation suitable EQ-algebras (not only plain EQ-algebras but especially special algebras with properties suitable as truth structures of new fuzzy logics called EQ-logics), e.g., involutive EQ-algebras. These EQ-logics were presented for example in [4].

The problem is the following task. We would like to design and create **finite algebras with specific properties:**

- n—number of algebra elements (finite);
- Operators of an algebra;
- Compulsory axioms for elements and operators;
- Optional properties of operations (axioms);
- Produce algebraic structures following the defined axioms.

There is a possibility to create such an algebra ad hoc with the help of properties automated check. Such a check is not a hard task, we have to implement only simple combinatorial procedure to prove axioms on all possible variations of the candidate algebra. By a candidate algebra we mean a member consisting of n elements with defined operation tables.

The simplest possible method consists in a "brute force" (combinatorial) approach which generates whole state space (the need to generate every candidate algebra and its check against preferred properties). Such an approach is not suitable. Consider the following example.

- Example: n elements, k binary operations, l axioms (m elements dependence):
  - $N_c = (n)^{k*n*n}$ possible candidates;
  - l axioms check - expression computations $N_{ev} = l * (n^m)$ for every candidate;
  - number of axioms' operations performed $N_t = N_c * N_{ev}$;
  - axioms' operation uses tens of simple (CPU level) instructions;
  - current common computer processes about 10^9–10^{10} instructions per second.
- Fix $k = 3, l = 10, m = 3$ and observe the raw number of candidate algebras:
  - $\mathbf{n = 4}$, $\{0, a, b, 1\}$, $N_c \doteq 7.9 * 10^{28}$, $N_t \doteq \mathbf{5.1 * 10^{31}}$
  - $\mathbf{n = 5}$, $\{0, a, b, c, 1\}$, $N_c \doteq 2.6 * 10^{52}$, $N_t \doteq \mathbf{3.3 * 10^{55}}$
  - $\mathbf{n = 6}$, $\{0, a, b, c, d, 1\}$, $N_c \doteq 1.0 * 10^{84}$, $N_t \doteq \mathbf{2.4 * 10^{87}}$
  - $\mathbf{n = 7}$, $\{0, a, b, c, d, e, 1\}$, $N_c \doteq 1.6 * 10^{124}$, $N_t \doteq \mathbf{5.8 * 10^{127}}$

– ...

It is clear that even for minimal extent of the algebraic structures (limited number of elements in algebra) the range of search state space is huge and leading to high number of options (Figure 1). The superexponential complexity makes hard-computing algorithms for exploration of state space unusable and there is natural possibility to employ evolutionary techniques [5].

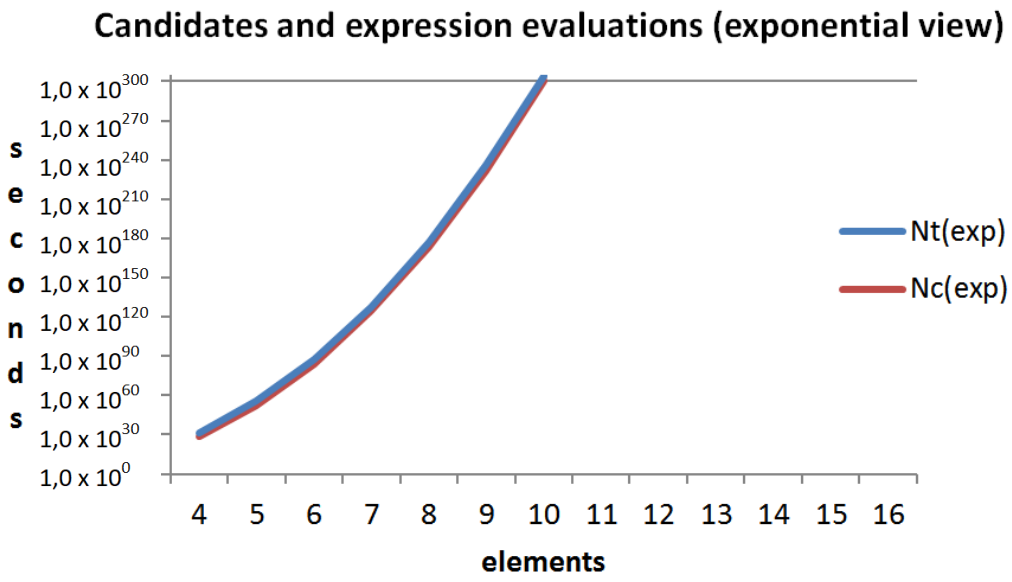

Figure 1. Superexponentiality of the problem (time—seconds vs. elements).

3. Genetic Algorithms

Genetic algorithms (GA) provide usual procedures for automated design of optimized structures based on evolution-inspired methods [1,6]. Even though GA are relatively simple to implement, it is very sensitive to suitable settings of basic parameters. It is also important to choose best fitting types of crossover and mutation for usage of evolutionary approach [7]. The paper [8,9] is an example of former research in this field. The algorithm in the paper was based on the following steps:

- The fitness is defined as sum of all error rates across all fitness states for particular operation (in some cases scaled).
- Method of generation for new members was tree-based GP technique (ECJ implementation).
- Special mutation called fair-mutation—using PTC1 and PTC2 algorithms (mutants in new subtree is guaranteed to not differ by more than predefined percentage).
- In some cases alternative selection methods were used, e.g., ECJ parsimony based tournament [8].

The paper describes the very high level of mutation (in some cases over 0.5), which shows problems with convergence of the process. We followed another approach using straightforward GA and we will also try to compare our results with [8]. However, the reader should consider our approach and objectives different with some similar impacts.

Main characteristics of Genetic Algorithms [2], Figure 2:

- Population member (candidate solution), fitness criterion (computes suitability based on axioms);
- Population—member set, default population (randomly generated);
- Derived generation is based on previous generation by selection, crossover and mutation operators;
- Generate new populations until the stop condition is fulfilled (fix number of iterations—populations, chosen fitness value set as optimal, etc.).

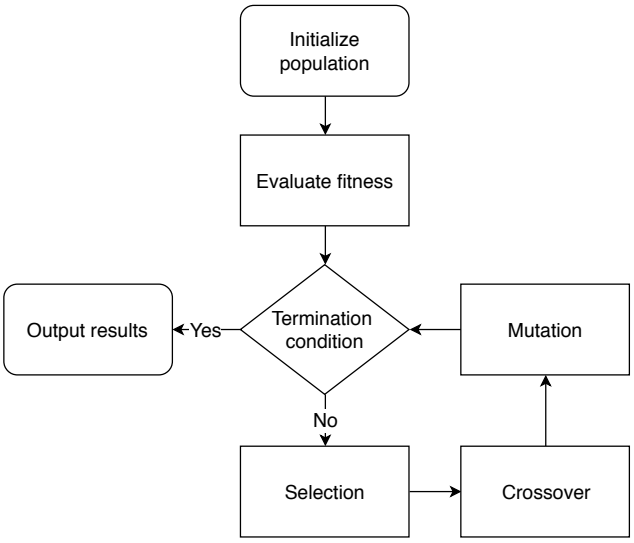

Figure 2. Genetic algorithm flowchart.

Population of Members (GA)

- Candidate solution p (Population Member / PM) coded according to its properties (usually stored in "chromosomes"—bit array, integer array etc.);
- Value of fitness function for candidate member f, $f(x) \in \langle 0, 1 \rangle$, x is PM—the most complex computation with respect to time efficiency of the whole task (parallelism should be considered as an optimization option);
- Population—fix or variable number of PM: Population member (possible algebra structure), evaluated by the fitness function (algebra axioms fulfilment), Population—sets of PMs, best PM, worst PM, median PM, Generation—particular populations in sequence are called generations $G_0, \ldots, G_r$, where $G_i = \{p_{i,j} | i, j \in N\}$, i is generation index, j describes index of member in population,
- Starting Generation G_0—it is generated as partial random population.

Genetic operators (GA)

- Selection—provides tool for choosing parent members for further processing:
  - Elitist—best m PM from G_i are used in next generation G_{i+1},
  - Crossover selection (CS)—selected PMs (parents) from G_i are set as source elements for generation of descendants (new generation) for G_{i+1},
  - CS should conform the selection criterion $prob_{CS}(p)$ for PM p monotone with respect to fitness function: $f(p_1) \geq f(p_2) \Rightarrow prob_{CS}(p_1) \geq prob_{CS}(p_2)$.
- Crossover—combination of several PMs to produce new descendant PMs for new generation:
  - Ordinary type—two parent PMs p_{old1}, p_{old2} lead to two descendants, where the first portion of chromosome is from p_{old1} and the second from p_{old2} and contrary,
  - Exponential type—if we can distinguish several portions of a chromosome, we can produce higher number of descendants than parents number (combinatorial production of all variations).
- Mutation—new PMs have predefined probability to be mutated:
  - Mutation rate—probability of selection PM for mutation;
  - Point—single element of chromosome is altered;
  - Interval—multiple parts of chromosome elements are mutated;
  - Overall—whole chromosome is altered.

4. EQ-Algebras

The task is to produce specific algebras—**EQ-algebras**. Research of EQ-Algebras is aiming mainly in its potential to serve as truth value structure for new EQ-logics [10,11]. Instead of implication, the key operation is Fuzzy Equality. EQ-algebra has three essential operations in total: Infimum $\wedge$, Multiplication $\otimes$, Fuzzy Equality $\sim$ and dependent operations (derivable)—Implication $\rightarrow$, Negation $\neg$, and relational operator LessThanOrEqual $\leq$.

EQ-algebra $\mathcal{E}$ is algebra of type (2, 2, 2, 0), i.e.,

$$\mathcal{E} = \langle E, \wedge, \otimes, \sim, \mathbf{1} \rangle \tag{1}$$

(E1) $\langle E, \wedge, \mathbf{1} \rangle$ is a commutative idempotent monoid (i.e., $\wedge$-semilattice with top element $\mathbf{1}$). We put $a \leq b$ iff $a \wedge b = a$, as usual.

(E2) $\langle E, \otimes, \mathbf{1} \rangle$ is forms a monoid and $\otimes$ is isotonic w.r.t. $\leq$.

(E3) $a \sim a = \mathbf{1}$ (reflexivity axiom)

(E4) $((a \wedge b) \sim c) \otimes (d \sim a) \leq c \sim (d \wedge b)$ (substitution axiom)

(E5) $(a \sim b) \otimes (c \sim d) \leq (a \sim c) \sim (b \sim d)$ (congruence axiom)

(E6) $(a \wedge b \wedge c) \sim a \leq (a \wedge b) \sim a$ (monotonicity of fuzzy equality axiom)

(E7) $a \otimes b \leq a \sim b$ (axiom of boundedness)

Special EQ-algebras Let $\mathcal{E}$ be an EQ-algebra and $a, b, c, d \in E$. We say that $\mathcal{E}$ is:

- *Separated* if for all $a \in E$, $a \sim b = \mathbf{1}$ implies $a = b$.
- *Good* if $a \sim \mathbf{1} = a$.
- *Residuated* if for all $a, b, c \in E$, $(a \otimes b) \wedge c = a \otimes b$ iff $a \wedge ((b \wedge c) \sim b) = a$.
- *Involutive* if for all $a \in E$, $\neg\neg a = a$.
- *Prelinear* if for all $a, b \in E$, $\sup\{a \rightarrow b, b \rightarrow a\} = \mathbf{1}$.
- *Lattice EQ-algebra* (ℓEQ-algebra) if it is a lattice and $((a \vee b) \sim c) \otimes (d \sim a) \leq (d \vee b) \sim c$.
- *Linear* if for all $a, b \in E$ $((a \wedge b) = a)$ or $((a \wedge b) = b)$.

Example EQ-algebra of the size 6 (generated by EQCreator)

$\wedge$	$\mathbf{0}$	a	b	c	d	$\mathbf{1}$
$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$
a	$\mathbf{0}$	a	b	c	d	a
b	$\mathbf{0}$	b	b	c	b	b
c	$\mathbf{0}$	c	c	c	c	c
d	$\mathbf{0}$	d	b	c	d	d
$\mathbf{1}$	$\mathbf{0}$	a	b	c	d	$\mathbf{1}$

$\otimes$	$\mathbf{0}$	a	b	c	d	$\mathbf{1}$
$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$
a	$\mathbf{0}$	a	b	c	d	a
b	$\mathbf{0}$	b	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	b
c	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	c
d	$\mathbf{0}$	d	$\mathbf{0}$	$\mathbf{0}$	d	d
$\mathbf{1}$	$\mathbf{0}$	a	b	c	d	$\mathbf{1}$

$\sim$	$\mathbf{0}$	a	b	c	d	$\mathbf{1}$
$\mathbf{0}$	$\mathbf{1}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$
a	$\mathbf{0}$	$\mathbf{1}$	$\mathbf{1}$	$\mathbf{1}$	$\mathbf{1}$	$\mathbf{1}$
b	$\mathbf{0}$	$\mathbf{1}$	$\mathbf{1}$	$\mathbf{1}$	$\mathbf{1}$	$\mathbf{1}$
c	$\mathbf{0}$	$\mathbf{1}$	$\mathbf{1}$	$\mathbf{1}$	$\mathbf{1}$	$\mathbf{1}$
d	$\mathbf{0}$	$\mathbf{1}$	$\mathbf{1}$	$\mathbf{1}$	$\mathbf{1}$	$\mathbf{1}$
$\mathbf{1}$	$\mathbf{0}$	a	a	a	a	$\mathbf{1}$

5. EQ-Algebras Design by Specific Genetic Algorithms

During exploration of potential EQ-algebras automated generation for further research, we used GA with specific options. The software application EQCreator is implemented using OOP design (EQ-algebras as GA Population Members). GA Population is implemented as *list* of PMs. The fitness function is based on relative fulfilment of mandatory and optional axioms. Automatically generated EQ-algebras designed following all criteria (axioms) are called Winners and they are stored as a result. The important element of the procedure is detection of previously produced (identical) population members (removal).

Random (starting) population is partially built to fulfil simple properties (e.g., infimum is commutative).

For every EQ-algebra operation (operator) we have to set the operation table (two-dimensional array). It depends on the type of the algebra how many operators are generated. General EQ-algebras have three basic operations (infimum, multiplication and

equality), but delta operator is further needed for the special type algebras (ΔEQ-algebras). Derived operators (implication, ordering, negation etc.) are not needed to be generated since they are computed from three basic operators.

Fitness evaluation has two phases:

- Mandatory properties computation (axioms like boundedness property—$a \otimes b \leq a \sim b$).
- Optional properties computation (e.g., Involutive—for all $a \in E$, $\neg\neg a = a$).

In every generation, PMs are sorted in population according to fitness value. There is also a need to determine stopping condition:

- Predefined number of steps performed.
- Predefined number of EQ-algebras with required properties.
- Manual (user) termination.

Algorithms are implemented in the form of software tool EQCreator—MS Windows 32-bit application with user interface [12]. The Graphical User Interface with GA and algebra settings, algebra preview and generation utilities is demonstrated on Figure 3 with sample general EQ-algebra of the size 5. Its main purpose is as follows:

- Predefine essential properties for algebras generation.
- Algebras production with EQ-algebras fulfilling specific properties.
- Properties (axioms) computation additionally required for generated structures.
- Saving of resulting optimal solutions for further processing.

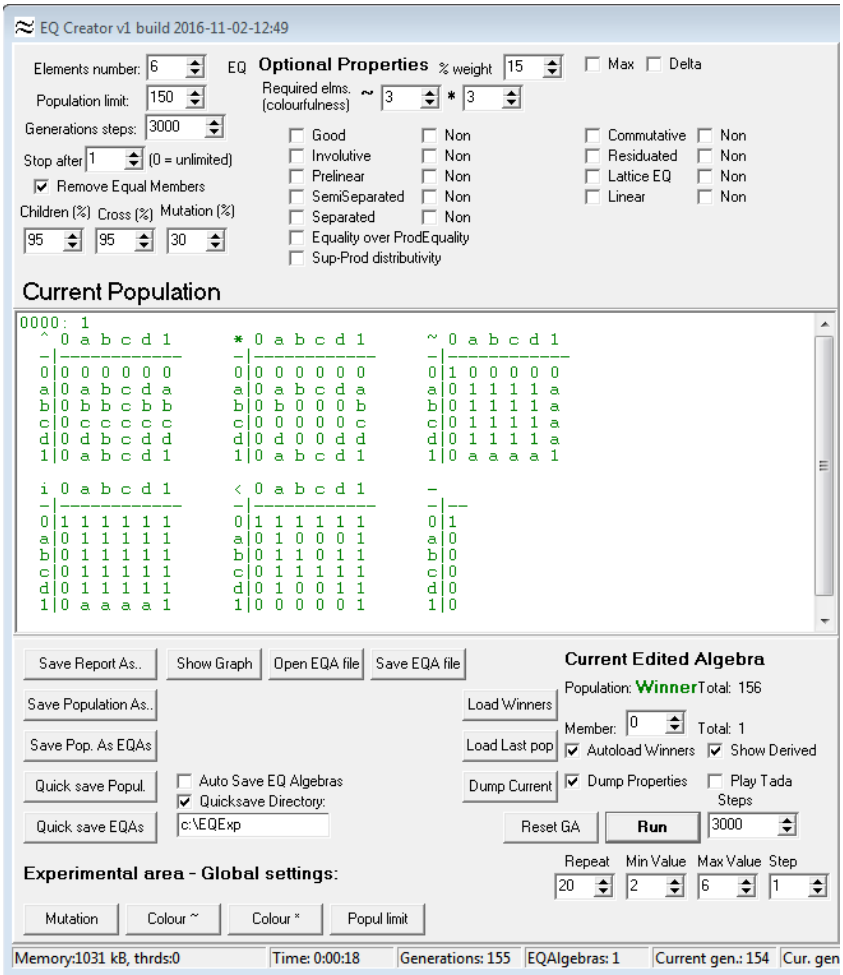

Figure 3. EQCreator showing the generated EQ-algebra.

The software is working according to basic settings—Algebra extent allow define size from 2 to 28 elements, Population limit—max. number of algebras in the population, Generation steps—max. number of generations permitted (or it can be terminated by user) (0—unlimited) and Termination after desired number of EQ-algebras is found. The application allows to set essential properties of genetic operators:

- Children ratio (0–100%)—crossover resulting descendant relative count (how large portion of the new population to be new descendants, others are old members copied from the previous generation);
- Cross ratio (0–100%)—portion of BEST members to have a possibility to be used for crossover (it is not a crossover probability!);
- Mutation ratio (0–100%)—descendant population members probability to be altered by mutation;
- Selection for crossover probability is set arbitrary (fixed)—in a descending order (by fitness) population of size N, we set probability of member i $p_i = \frac{N-i}{\frac{N*(N+1)}{2}}$ for $i = 0, ..., N-1$, where $f(i) \geq f(i+1)$ (fitness for members) e.g., for 5 members: $p_0 = \frac{5}{15}, p_1 = \frac{4}{15}, ..., p_4 = \frac{1}{15}$;
- Selection of members for crossover is done by roulette type (fitness proportionate selection) with above defined probability. The example scheme for 10 members m[0]..m[9] sorted by fitness function from best to worst is demonstrated on Figure 4;
- Weight of optional properties—relative weight of special EQ-algebras requirements (e.g., linear EQA, involutive EQA)—should be significantly lower than essential axioms weight (from experiments its best setting is 15%);
- Notion of colorfulness—required distinct elements incidence in non-trivial positions as operator function values (some combinations are trivial, e.g., $a \wedge 0 = 0$ in every EQA);
- **Colorfulness** assures non-trivial EQ-algebras to be generated, e.g., for fuzzy equality when 3 out of 5 required—at least 3 different elements occur as functional values in non-determined cases;
- colorfulness experimentally needed for Multiplication ($\otimes$) and Fuzzy Equality ($\sim$)—higher means computationally harder!

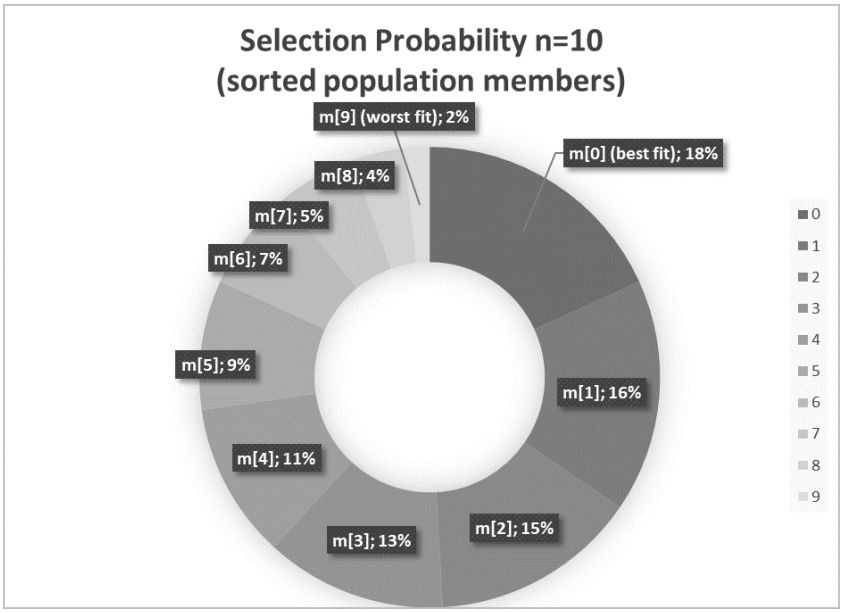

Figure 4. Roulette type selection for sorted population members by fitness (n = 10) .

EQ-creator also enables to inspect properties of both candidate algebras and special EQ-algebras when the user needs to adjust GA properties (Figure 5).

Since genetic algorithms have many types of crossover and mutation mechanisms, we have experimented with many of them (e.g.simple crossover, exponential crossover, point mutation, interval mutation). Our presented results were attained with a suitable combination of the exponential type of crossover with point mutation. Figure 6 illustrates our method, which depends on the production of $2k$ new population members by recombination of every algebra operator separately (where k is the number of operators).

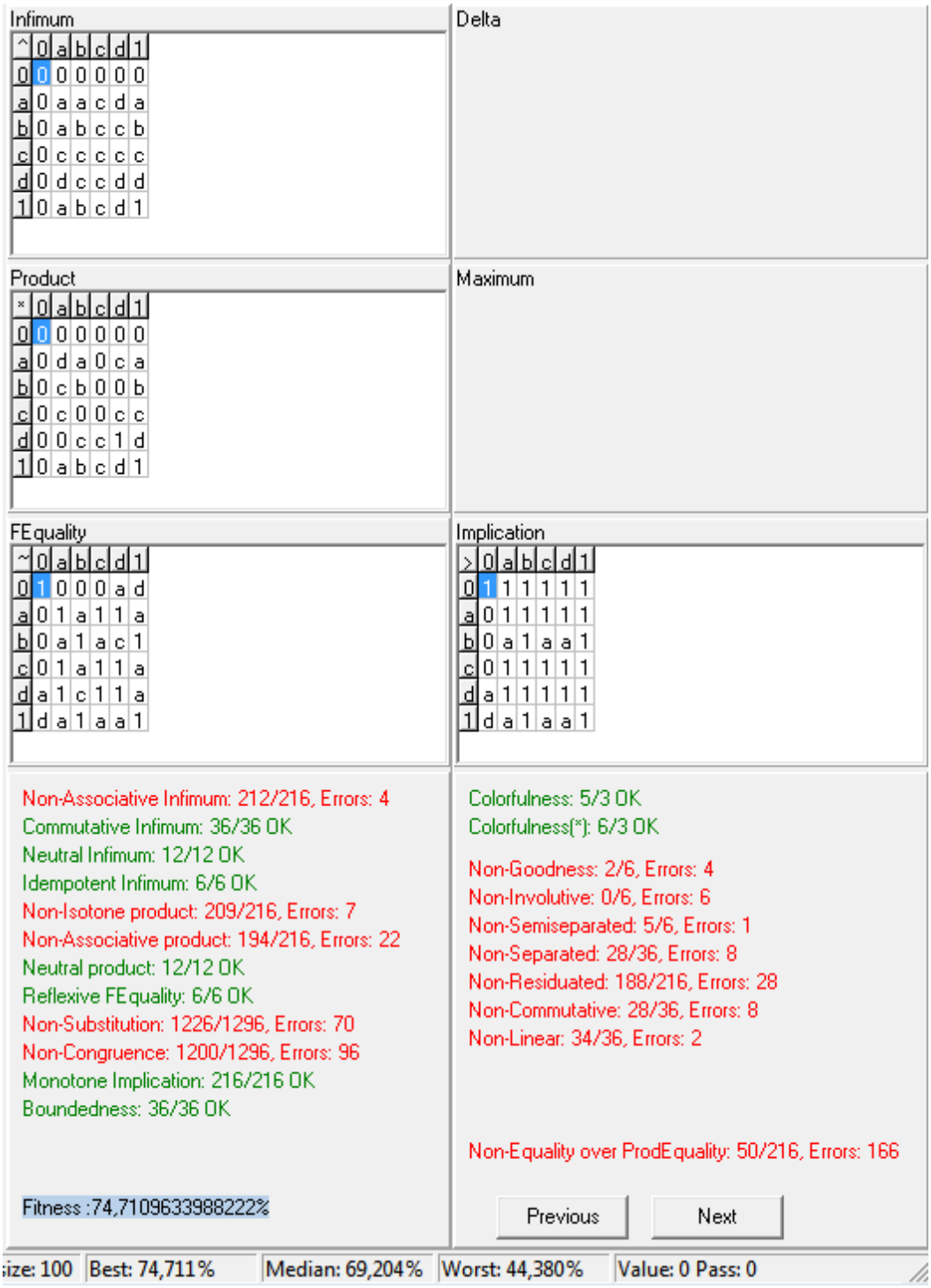

Figure 5. EQCreator and properties of an EQ-algebra.

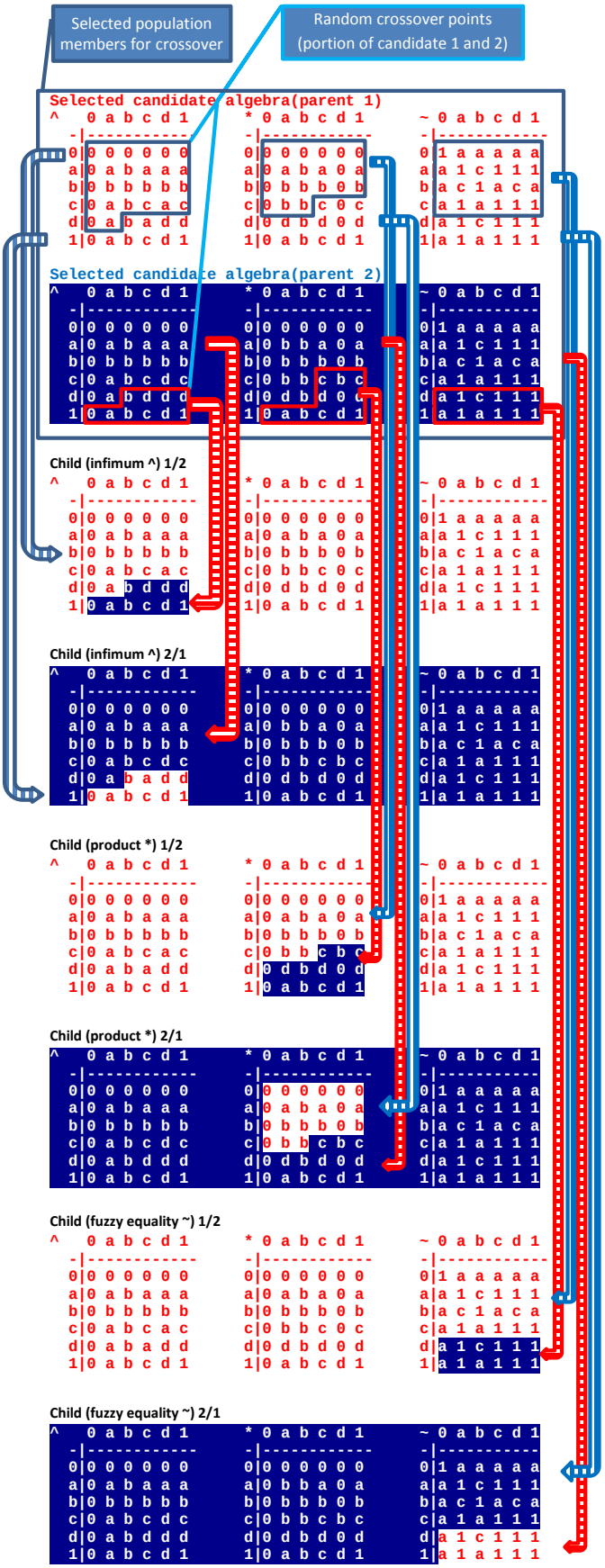

Figure 6. Specific exponential crossover by algebra operators.

As it can be seen from the experimental usage of EQCreator application (Figure 7), the time efficiency is far better than using the classical algorithmical approach. Superexponential classical approach has been reduced to something we can assume to be only exponential time complexity, which is, for our extent of algebras (about $n < 20$), an acceptable time complexity. We are now working on the possibility of paralelization of GA, in order to produce larger EQ-algebras in a shorter time.

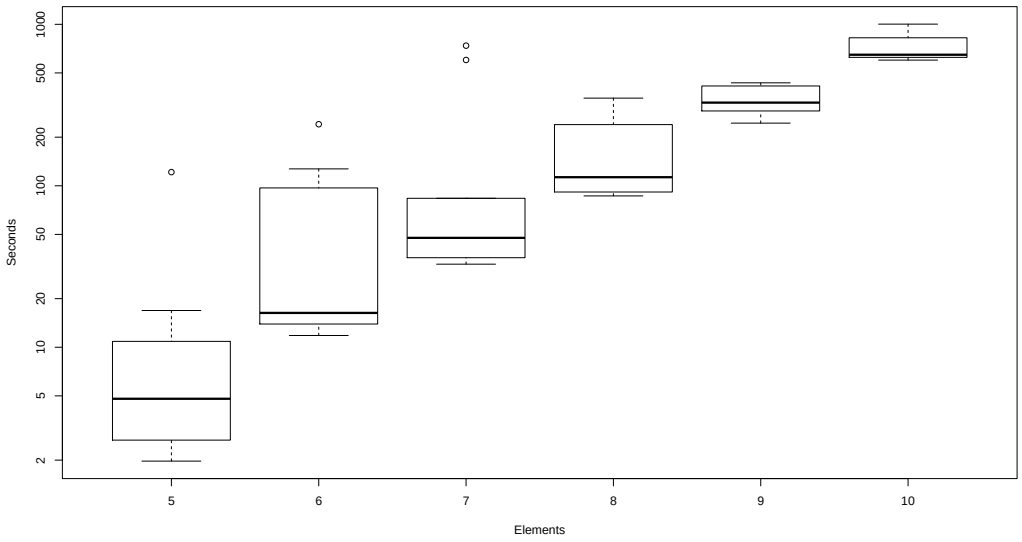

Figure 7. GA efficiency (time versus number of elements). Tested on Pentium 4—2.8 GHz. Improvement—*(no superexponentiality)*.

6. Experimental Evaluation and Optimization

Although the above described approach enabled the process of finite algebras generation to be feasible, it is also very important to tune the appropriate settings of the genetic algorithm. The main problem discovered already in [8] is connected with the level of mutation ratio—probability (the probability of a new population member to be mutated). The high mutation ratio has been observed in the mentioned article (even over 0.5) and it is also our experience during preliminary experiments. Since the mutation ratio affects convergence of the GA, it has a significant impact on computational efficiency. Since mutation probability is the most important part for successful genetic algorithms applications together with suitable crossover implementations, we have been thoroughly experimenting with this parameter.

We have created special version of EQCreator software for our experiments, capable of automatic generation of statistically appropriate samples to test the measurement results with statistical software NCSS.

6.1. Experiment No. 1

In this experiment we used EQCreator with parameters shown in Table 1 to determine the relationship between runtime and mutation rate. The mutation rate was set from 10% to 45% with 5% step.

Table 1. Experiment no. 1.

Algebra Settings	
Elements number	n = 6
Colourfulness ($\otimes$)	3
Colourfulness ($\sim$)	3
No special algebra constraints	
Analysis of variance	
F-Ratio	1.25
Prob Level	0.279402

Analysis of variance for time vs. mutation rate shows no statistically significant differences (Table 1), but the plot of the means shows a decrease in the measured time with the best results in 30–35% and then the results show increase in computation time related to the increase in mutation rate (Figures 8 and 9).

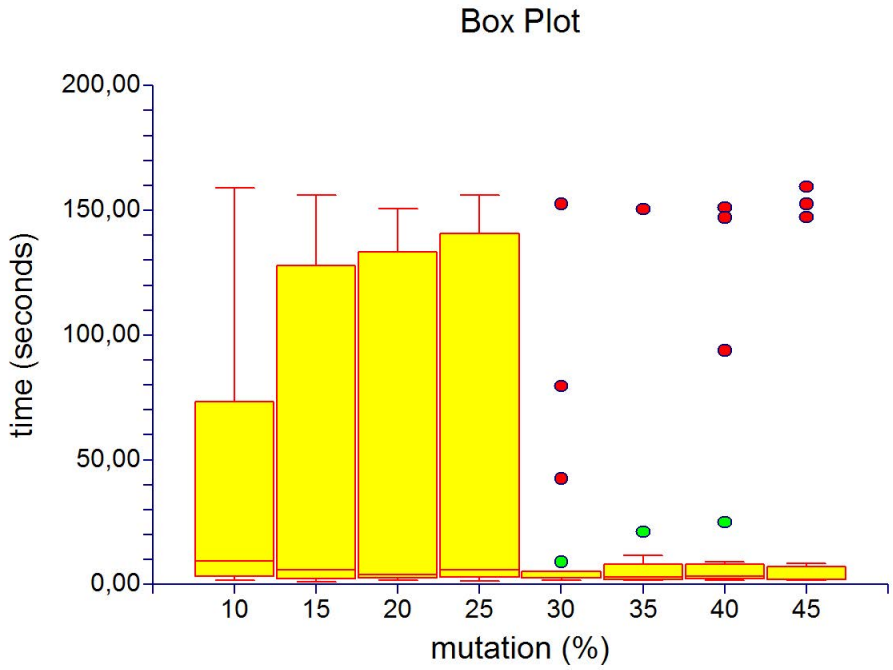

Figure 8. Exp. no. 1—Box Plot.

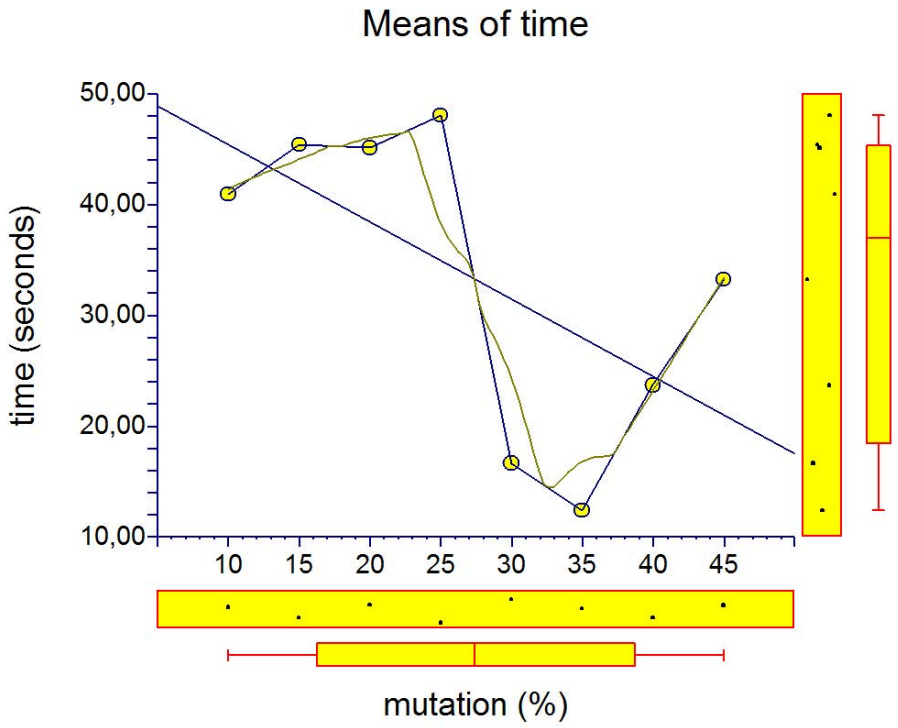

Figure 9. Exp. 1—Computing time means.

Fisher's LSD Multiple-Comparison Test shows statistically significant (0.05) difference between level 35 percent and 25 percent.

6.2. Experiment No. 2

An analysis of variance for time vs. fuzzy equality colorfulness shows no statistically significant differences. There is a natural increase in the computation time with respect to higher requirements on colorfulness of fuzzy equality (Figures 10 and 11). The last value shows a slight decrease, but not statistically significant (Table 2).

Table 2. Experiment no. 2.

Algebra Settings	
Elements number	n = 6
Colourfulness (∼)	from 2 to 5
Mutation	30%
No special algebra constraints	
Analysis of variance	
F-Ratio	1.92
Prob Level	0.134016

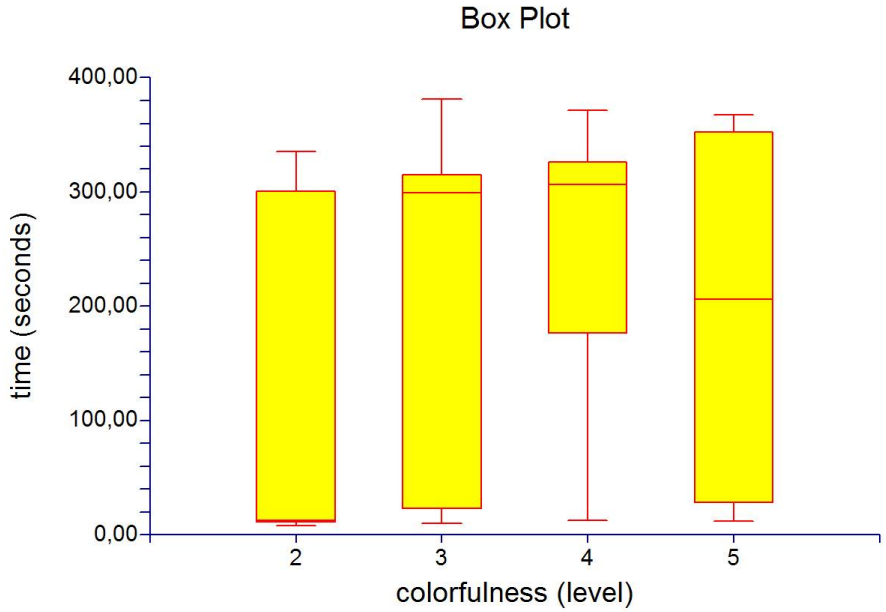

Figure 10. Exp. no. 2—Box Plot.

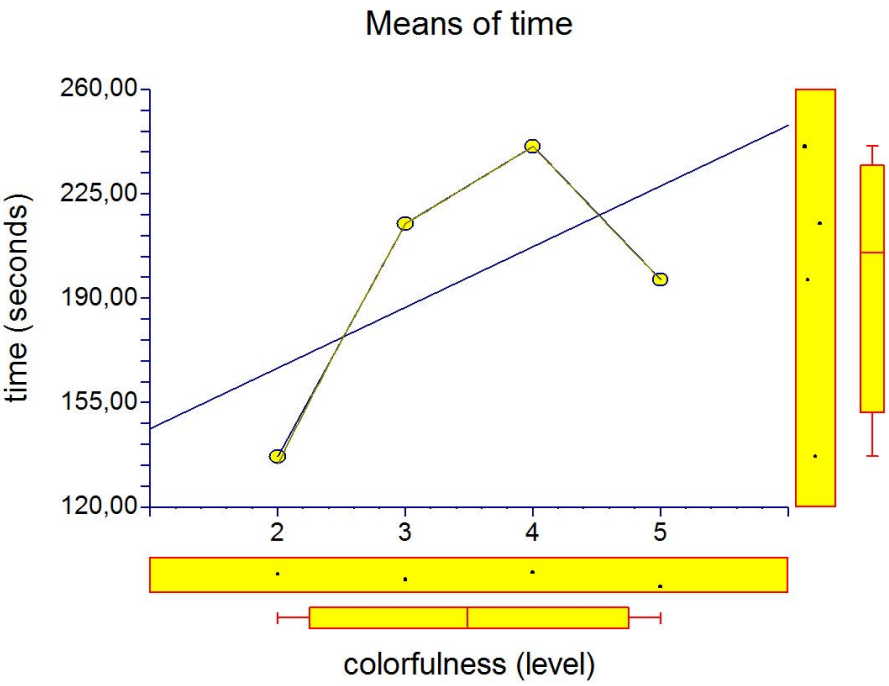

Figure 11. Exp. 2—Computing time means.

6.3. Experiment No. 3

An analysis of variance for time vs. population limit shows statistically significant differences (Table 3). It seem to be more efficient to work with smaller populations up to 250 members to attain optimal performance (Figures 12 and 13).

Table 3. Experiment no. 3.

Algebra Settings	
Elements number	n = 6
Initial population limit	100
Final population limit	500
Population step	50
Colourfulness ($\otimes$)	3
Colourfulness ($\sim$)	3
Mutation	30%
No special algebra constraints	
Analysis of variance	
F-Ratio	2.14
Prob Level	0.034365

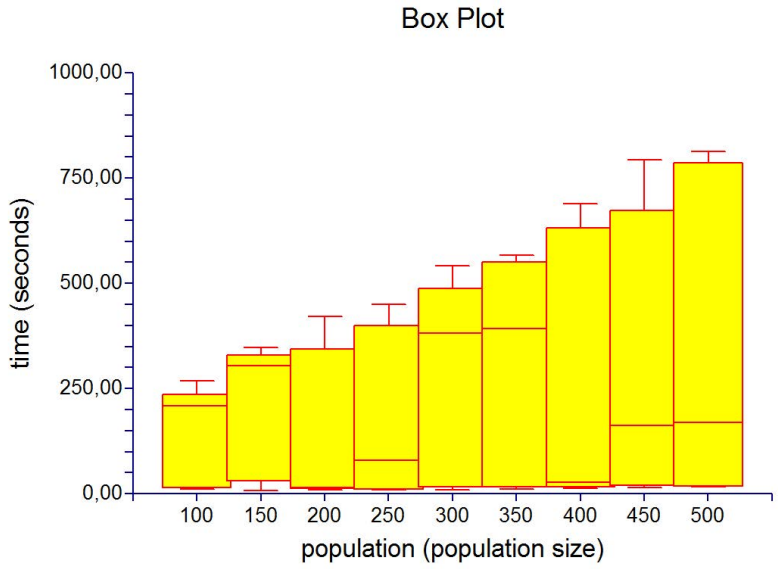

Figure 12. Exp. no. 3—Box Plot.

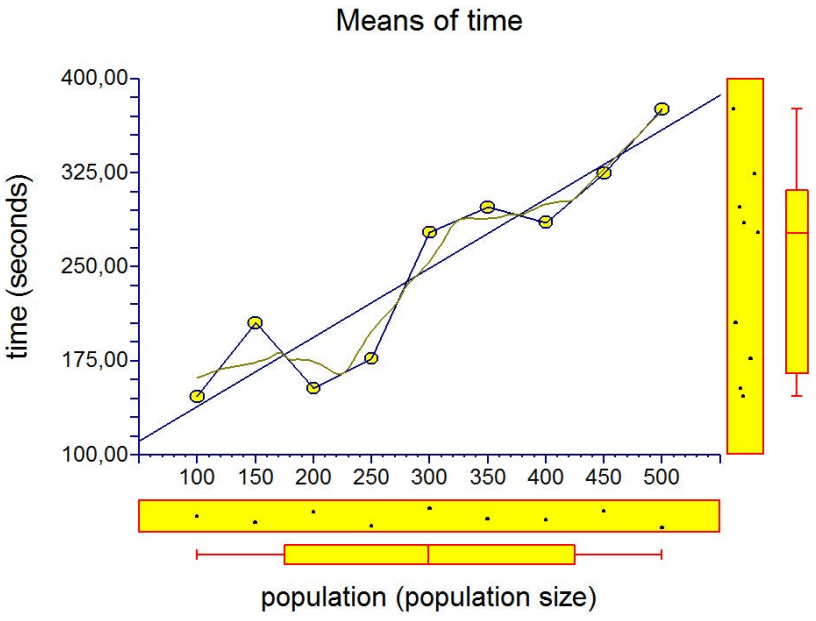

Figure 13. Exp. 3—Computing time means.

6.4. Experiment No. 4

The analysis of variance for time vs. multiplication colorfulness shows statistically significant differences. The results show that the best efficiency is for smaller colorfulness of multiplication in contrast to fuzzy equality results. (Table 4). Computing times are presented in Figures 14 and 15.

Our experiments with mutation rate showed the need to adjust the rate to relatively high values. Nevertheless ANOVA results proved that around 30% there is a typical area of best mutation rate settings for our GA optimization method.

Another important finding relates to colorfulness and its impact to computing time. Medium values of colorfulness for fuzzy equality and multiplication are harder to compute necessary EQ-algebras.

Table 4. Experiment no. 4.

Algebra Settings	
Elements number	n = 6
Colourfulness (⊗)	from 2 to 5
Mutation	30%
No special algebra constraints	
Analysis of variance	
F-Ratio	8.20
Prob Level	0.0997917

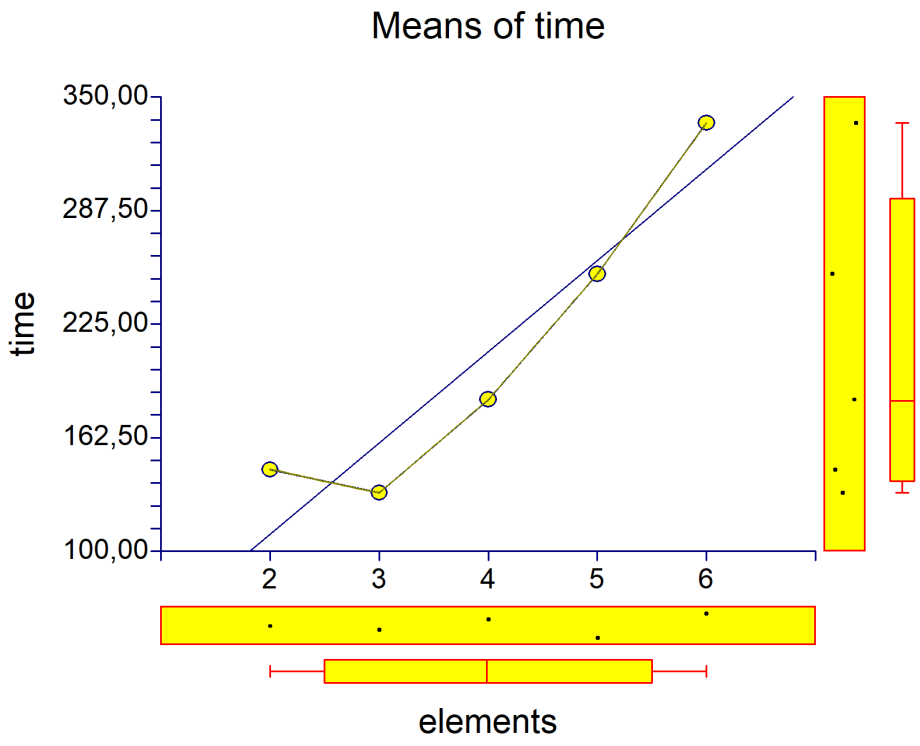

Figure 14. Exp. no. 4—Box Plot.

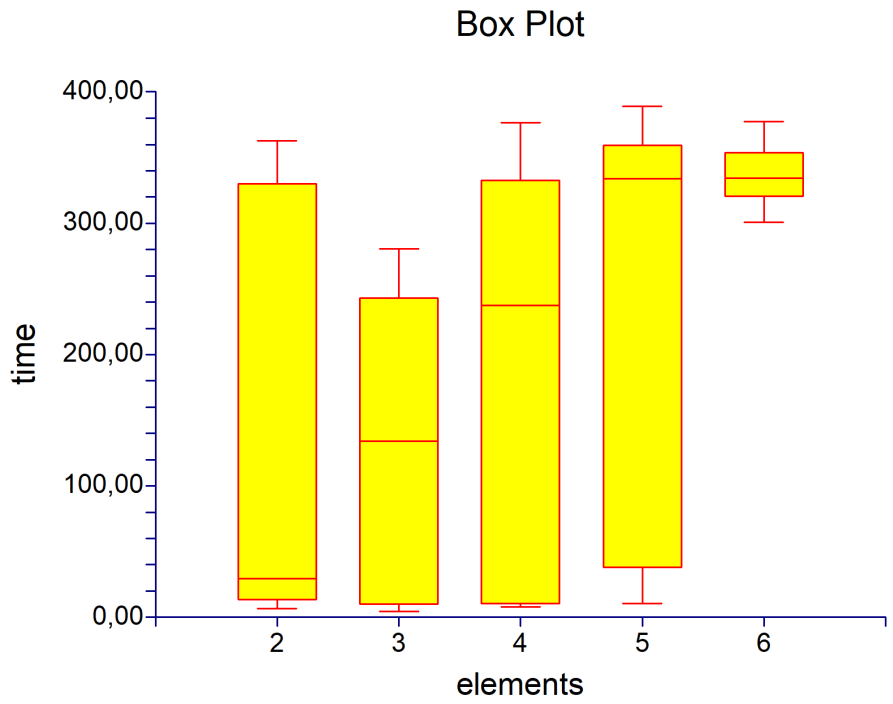

Figure 15. Exp. 4—Computing time means.

7. Examples and Application

The presented algorithm and its produced EQ-algebras were used as potential truth valued structures for EQ-logics by team of researchers of Institute for Research and Applications of Fuzzy Modeling, University of Ostrava [10]. We present some interesting examples of special algebras produced for use as a truth value structures of EQ-logics.

7.1. Good EQ-Algebra—$a \sim \mathbf{1} = a$

Good EQ-algebras have special requirement for fuzzy equality: $a \sim \mathbf{1} = a$. Table 5 presents EQCreator settings for such an algebra and the resulting sample algebra follows. Figure 16 presents GA iteration progress for this particular case.

Table 5. EQCreator settings for good algebra.

Algebra Settings	
Elements number	n = 6
Colourfulness ($\otimes$)	3
Colourfulness ($\sim$)	3
Mutation	30%
Special requirements	Good EQ-algebra

$$
\begin{array}{c|cccccc}
\wedge & \mathbf{0} & a & b & c & d & \mathbf{1} \\
\hline
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
a & \mathbf{0} & a & b & c & a & a \\
b & \mathbf{0} & b & b & b & b & b \\
c & \mathbf{0} & c & b & c & c & c \\
d & \mathbf{0} & a & b & c & d & d \\
\mathbf{1} & \mathbf{0} & a & b & c & d & \mathbf{1}
\end{array}
\qquad
\begin{array}{c|cccccc}
\otimes & \mathbf{0} & a & b & c & d & \mathbf{1} \\
\hline
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
a & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & a \\
b & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & b \\
c & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & c \\
d & \mathbf{0} & a & \mathbf{0} & c & d & d \\
\mathbf{1} & \mathbf{0} & a & b & c & d & \mathbf{1}
\end{array}
\qquad
\begin{array}{c|cccccc}
\sim & \mathbf{0} & a & b & c & d & \mathbf{1} \\
\hline
\mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
a & \mathbf{0} & \mathbf{1} & b & c & a & a \\
b & \mathbf{0} & b & \mathbf{1} & c & b & b \\
c & \mathbf{0} & c & c & \mathbf{1} & c & c \\
d & \mathbf{0} & a & b & c & \mathbf{1} & d \\
\mathbf{1} & \mathbf{0} & a & b & c & d & \mathbf{1}
\end{array}
$$

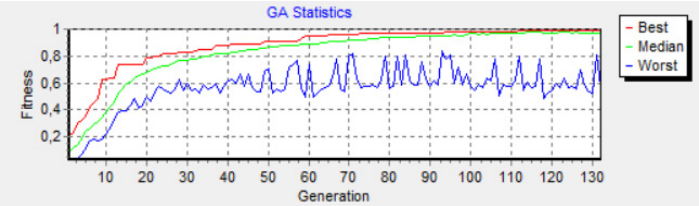

Figure 16. EQCreator computation for special algebra—good.

7.2. Involutive EQ-Algebra—For All $a \in E$, $\neg\neg a = a$

Involutive EQ-algebras have special requirement for negation, which is derivable from implication operation (derived operation defined as $a \to b = (a \wedge b) \sim a$): $a \sim \mathbf{1} = a$). Then, negation is defined as $\neg a = a \sim \mathbf{0}, a \in E$. Involutive algebra must conform to for all $a \in E$, $\neg\neg a = a$. Table 6 presents EQCreator settings for such an algebra and the resulting sample algebra follows. Figure 17 presents GA iteration progress for this particular case.

Table 6. EQCreator settings for involutive algebra.

Algebra Settings	
Elements number	n = 6
Colourfulness ($\otimes$)	3
Colourfulness ($\sim$)	3
Mutation	30%
Special requirements	Involutive EQ-algebra

$$
\begin{array}{c|cccccc}
\wedge & \mathbf{0} & a & b & c & d & \mathbf{1} \\
\hline
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
a & \mathbf{0} & a & a & a & d & a \\
b & \mathbf{0} & a & b & b & d & b \\
c & \mathbf{0} & a & b & c & d & c \\
d & \mathbf{0} & d & d & d & d & d \\
\mathbf{1} & \mathbf{0} & a & b & c & d & \mathbf{1}
\end{array}
\qquad
\begin{array}{c|cccccc}
\otimes & \mathbf{0} & a & b & c & d & \mathbf{1} \\
\hline
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
a & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & a \\
b & \mathbf{0} & \mathbf{0} & d & a & \mathbf{0} & b \\
c & \mathbf{0} & \mathbf{0} & d & a & \mathbf{0} & c \\
d & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & d \\
\mathbf{1} & \mathbf{0} & a & b & c & d & \mathbf{1}
\end{array}
\qquad
\begin{array}{c|cccccc}
\sim & \mathbf{0} & a & b & c & d & \mathbf{1} \\
\hline
\mathbf{0} & \mathbf{1} & b & a & d & c & \mathbf{0} \\
a & b & \mathbf{1} & b & a & b & a \\
b & a & b & \mathbf{1} & b & a & b \\
c & d & a & b & \mathbf{1} & d & c \\
d & c & b & a & d & \mathbf{1} & d \\
\mathbf{1} & \mathbf{0} & a & b & c & d & \mathbf{1}
\end{array}
$$

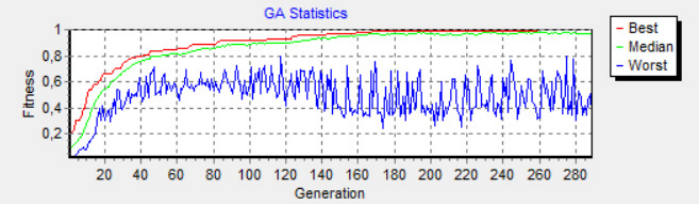

Figure 17. EQCreator computation for special algebra—involutive.

7.3. Non-Commutative Good EQ-Algebra—Multiplication $\otimes$ Is Non-Commutative Operation

Non-commutative EQ-algebras have the multiplication operation which is not commutative operation. They would be also interesting for EQ-logic creators (e.g., [10]). Table 7 presents EQCreator settings for such an algebra and the resulting sample algebra follows. Figure 18 presents GA iteration progress for this particular case.

Table 7. EQCreator settings for non-commutative multiplication algebra

Algebra Settings	
Elements number	n = 6
Colourfulness ($\otimes$)	3
Colourfulness ($\sim$)	3
Mutation	30%
Special requirements	Non-commutative ($\otimes$) and good EQ-algebra

$\wedge$	**0**	a	b	c	d	**1**
0	**0**	**0**	**0**	**0**	**0**	**0**
a	**0**	a	b	c	d	a
b	**0**	b	b	c	b	b
c	**0**	c	c	c	c	c
d	**0**	d	b	c	d	d
1	**0**	a	b	c	d	**1**

$\otimes$	**0**	a	b	c	d	**1**
0	**0**	**0**	**0**	**0**	**0**	**0**
a	**0**	a	b	**0**	**0**	a
b	**0**	**0**	**0**	**0**	**0**	b
c	**0**	**0**	**0**	**0**	**0**	c
d	**0**	**0**	**0**	**0**	**0**	d
1	**0**	a	b	c	d	**1**

$\sim$	**0**	a	b	c	d	**1**
0	**1**	**0**	d	d	b	**0**
a	**0**	**1**	b	c	d	a
b	d	b	**1**	d	d	b
c	d	c	d	**1**	b	c
d	b	d	d	b	**1**	d
1	**0**	a	b	c	d	**1**

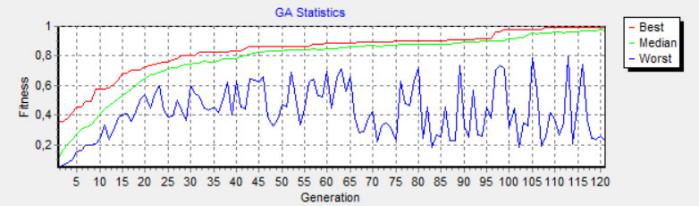

Figure 18. EQCreator computation for special algebra—non-commutative.

8. Conclusions

The main aim of this paper is to introduce an optimized solution for producing pure EQ-algebras. In the introductory section was proved that use of hard-computing approach is not suitable due to its superexponential time complexity. We utilized powerful Genetic Algorithms for production of pure EQ-algebras with optimized properties. The resulting structures were used by colleagues from Institute for Research and Application of Fuzzy Modeling for design of new equality based fuzzy logic. We can summarize the observed specific GA properties:

- Elitism is necessary at least for several parent members (5% was acceptable—despite it may negatively the convergence of the process for high elitism ratio).
- High mutation ratio must be set in contrast with traditional use of GA (best results with 25–35%).
- Optional axioms fulfilment increases computing time significantly (from experiments 15% is optimal setting).
- Optional properties negatively affect convergence.
- Colorfulness was defined to prevent trivial solutions (evolution tends to the simplest way of achieving results).

We implemented above mentioned methods into computer software EQ Creator—application for EQ-algebras only. The actual version of EQCreator can be downloaded from pages of Institute for Research and Application of Fuzzy Modeling (IRAFM). This approach is not only suitable for specific algebras, but we are currently working on an idea of general generator of algebras with user defined axioms (properties). We would also like to work especially on constrained EQ-algebras since the evolution of these algebras is even more a computationally hard task.

Author Contributions: Conceptualization, H.H. and M.K.; methodology, E.V.; software, H.H.; validation, H.H., M.K.; formal analysis, E.V.; writing—original draft preparation, H.H.; writing—review and editing, E.V., M.K.; All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by University of Ostrava specific research budget.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Hingston, P.; Barone, L.; Michalewicz, Z. *Design by Evolution: Advances in Evolutionary Design*; Springer: Berlin/Heidelberg, Germany, 2008.
2. Habiballa, H.; Schenk, J.; Hires, M.; Jendryscik, R. Automated Design and Optimization of Specific Algebras by Genetic Algorithms. In *CSOC 2016—Artificial Intelligence Perspectives in Intelligent Systems, AISC*; Springer: Cham, Switzerland, 2016; Volume 464, pp. 359–369.
3. Hobby, D.; McKanzie, R. *Structure of Finite Algebras*; American Mathematical Society: Providence, RI, USA, 1988.
4. Dyba, M.; Novák, V. EQ-logics with delta connective. *Iran. J. Fuzzy Syst.* **2015**, *12*, 41–61.
5. Langdon, W.B.; Poli, R. *Foundations of Genetic Programming*; Springer: Berlin/Heidelberg, Germany, 2002.
6. Sekanina, L. Evolvable hardware. In *Handbook of Natural Computing*; Springer: Berlin, Germany, 2012; pp. 1657–1705. ISBN 978-3-540-92909-3.
7. Kalyanmoy, D. Multi-Objective Evolutionary Algorithms. In *Springer Handbook of Computational Intelligence*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 995–1015.
8. Spector, L.; Clark, D.M.; Lindsay, I.; Barr, B.; Klein, J. Genetic programming for finite algebras. In Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation (GECCO '08), New York, NY, USA, 12–16 July 2008; ACM: New York, NY, USA, 2008; pp. 1291–1298.
9. Spector, L.; McPhee, N.F.; Spector, L.; Clark, D.M.; Lindsay, I.; Barr, B.; Klein, J. Push: Genetic programming for finite algebras. In Proceedings of the Genetic and Evolutionary Computation Conference Companion, Prague, Czech Republic, 13–17 July 2019; pp. 1175–1196.
10. Dyba, M.; Novák, V. EQ-logics—Non-commutative fuzzy logics based on fuzzy equality. *Fuzzy Sets Syst.* **2011**, *172*, 13–32. [CrossRef]
11. Novák, V.; De Baets, B. EQ-algebras. *Fuzzy Sets Syst.* **2009**, *160*, 2956–2978. [CrossRef]
12. Habiballa, H.; Hires, M.; Jendryscik, R. Properties of Genetic Algorithms for Automated Algebras Generation. In *Artificial Intelligence Trends in Intelligent Systems, CSOC Conference 2017*; Springer: Cham, Switzerland, 2017; Volume 1, pp. 424–433.