A Genetic Algorithm
with Expansion and Exploration Operators
for the Maximum Satisfiability Problem

Anna Gorbenko

Department of Intelligent Systems and Robotics
Ural Federal University
620083 Ekaterinburg, Russia
gorbenko.ann@gmail.com

Vladimir Popov

Department of Intelligent Systems and Robotics
Ural Federal University
620083 Ekaterinburg, Russia
Vladimir.Popov@usu.ru

Abstract

There are many problems that standard genetic algorithms fail to solve. Refinements of standard genetic algorithms that can be used to solve hard problems has caused considerable interest. In this paper, we consider genetic algorithms with expansion and exploration operators for the maximum satisfiability problem.

Keywords: genetic algorithms, expansion operator, exploration operator, maximum satisfiability

The maximum satisfiability problem (MAX SAT) is the problem of determining the maximum number of clauses of a given Boolean formula in conjunctive normal form (CNF) that can be satisfied by some assignment. The problem MAX SAT remains NP-hard even if all expressions are written in conjunctive normal form with $k$ variables per clause ($k$-CNF), for any $k \geq 2$ (see e.g. [1]). The problem MAX $k$-SAT is the problem of determining the maximum number of clauses of a given $k$-CNF that can be satisfied by some assignment. It should be noted that investigation of algorithms for MAX
SAT received a lot of attention (see e.g. [2]). In this paper, we consider genetic algorithms with expansion and exploration operators to try to solve the maximum satisfiability problem.

Genetic algorithms are used to solve many different hard problems. In particular, we can mention such problems as the set of parameterized $k$-covers problem [3], different problems of sensor placement (see e.g. [4] – [9]), the binary paint shop problem [10], various problems of bioinformatics (see e.g. [11] – [20]), technical vision (see e.g. [21] – [25]), robot self-awareness (see e.g. [26] – [30]), robot anticipation (see e.g. [31] – [34]), different planning problems (see e.g. [35] – [39]), graph problems (see e.g. [40] – [43]), visual landmarks problems (see e.g. [44] – [47]), etc. But, it is well known that there are many problems that standard genetic algorithms fail to solve.

One of a number of ideas of refinement of standard genetic algorithms was proposed in [48]. The idea is to preserve good building blocks found by the genetic algorithm. For this purpose, we can use some constraints on the choice of recombination. In particular, an expansion operator can be used [48]. Also, we can use exploration operator as a refinement of mutation [48].

Expansion and exploration operators were proposed in [48] for the shortest common superstring problem. In particular, an expansion operator is an addition of another block to increase the individual’s genome length by one block. The added block is selected by expansion operator in such a way that the genome will still be a subsequence of at least one solution. Exploration operator acts as a mutative force within the building blocks population. This operator injects noise and thus promotes exploration of the huge building blocks search space. An individual may be removed and re-initialized as a new individual.

Let

$$f(z[1], z[2], \ldots, z[m])$$

be a Boolean function. Let

$$W = \{ u[1], u[2], \ldots, u[n] \},$$

$$u[i] \in \{0, 1\}^+,$$

$$1 \leq i \leq n,$$

be a population of chromosomes. We assume that

$$u[i] = u[i, 1]u[i, 2] \ldots u[i, m],$$

$$u[i, j] \in \{0, 1\},$$

$$1 \leq i \leq n,$$

$$1 \leq j \leq m.$$
A genetic algorithm for the maximum satisfiability problem

We can consider $u[i]$ as a solution for

$$f(z[1], z[2], \ldots, z[m]).$$

In particular, we assume that $z[j] = u[i, j]$. We consider only CNFs. So,

$$f = \wedge_{j=1}^{k} C[j](z[1], \ldots, z[m])$$

where

$$C[j](z[1], z[2], \ldots, z[m])$$

is a clause. We say that $u[i, j]$ is a true assignment for

$$C[k](z[1], z[2], \ldots, z[m])$$

if $u[i, j] \in \{0, 1\}$ and $z[j] = u[i, j]$ evaluates

$$C[k](z[1], z[2], \ldots, z[m])$$

to true. Let $C(u[i, j])$ be the set of clauses such that

$$C[k](z[1], z[2], \ldots, z[m]) \in C(u[i, j])$$

if and only if $u[i, j]$ is a true assignment for

$$C[k](z[1], z[2], \ldots, z[m]).$$

Now, we consider a standard genetic algorithm (SGA) for the satisfiability problem. A proportion $\mathcal{P}$ of the existing population is selected to breed a new generation during each successive generation. We assume that

$$\mathcal{P} = \lceil \frac{n}{2} \rceil.$$ 

Individual chromosomes are selected by a fitness function $\mathcal{F}$. To generate a second generation population of chromosomes, we can use two genetic operators: crossover $\mathcal{C}$ and mutation $\mathcal{M}$.

Usually, crossover defines a part of parent chromosome which used for construction of child chromosome. As $\mathcal{C}$ we use standard random operator. If $u[i]$ and $u[j]$ are two parent chromosomes, then we obtain two child chromosomes:

$$\mathcal{C}(u[i], u[j]) = u[i, 1] \ldots u[i, \mathcal{C}(u[i])]u[j, \mathcal{C}(u[i]) + 1] \ldots u[j, m],$$

$$\mathcal{C}(u[j], u[i]) = u[j, 1] \ldots u[j, \mathcal{C}(u[j])]u[i, \mathcal{C}(u[j]) + 1] \ldots u[i, m].$$

For

$$u[i[1]], u[i[2]], \ldots, u[i[p]],$$
we consider
\[ C(u[i[1]], u[i[2]]), C(u[i[2]], u[i[1]]), \ldots, C(u[i[p - 1]], u[i[p]]), C(u[i[p]], u[i[p - 1]]) \]
if \( p = 2q \),
\[ u[i[1]], C(u[i[2]], u[i[3]]), C(u[i[3]], u[i[2]]), \ldots, C(u[i[p - 1]], u[i[p]]), C(u[i[p]], u[i[p - 1]]) \]
if \( p = 2q + 1 \). This generational process is repeated until a termination condition \( T \) has been reached.

As \( T \) we consider time function. We assume that
\[ F(u[i]) = |\bigcup_{j=1}^m C(u[i, j])|. \]

Let \( M \) be a random function which with small probability changes values of \( u[i, j] \).

Now, we consider a genetic algorithm with expansion and exploration operators (GAEE). We use the following expansion operator instead \( C \). Let
\[ E_1(u[i], u[j]) = v[1] \ldots v[m] \]
where
\[ v[k] = \begin{cases} u[i, k], |C(u[i, k])| > 0, \\ u[j, k], |C(u[i, k])| < 1. \end{cases} \]

Also, we use exploration operator \( E_2 \) instead \( M \). If \( r \) is a number of generation
\[ W_r = \{u_r[1], u_r[2], \ldots, u_r[n]\}, \]
then
\[ F_r = \sum_{i=1}^n F(u_r[i]). \]

For any \( r > 1 \) and \( i < n \), we assume that
\[ F(u_r[i]) \geq F(u_r[i + 1]). \]

Let \( D_H(X,Y) \) be the Hamming distance between strings \( X \) and \( Y \). Let
\[ h_r = \sum_{i=1}^n D_H(u_r[i], u_{r-1}[i]), \]
\[ H_r = \sum_{1 \leq i < j \leq n} D_H(u_r[i], u_r[j]). \]

Operator \( E_2 \) used only if
\[ \alpha(F_r, h_r, H_r) < 1 \]
where \( \alpha(F_r, h_r, H_r) \) is a polynomial function which is predicted by a genetic algorithm. Operator \( \mathcal{E}_2 \) removes chromosome \( u_r[i] \) only if \( u_r[i] \) has high fitness value and there is \( u_{r-1}[j] \) such that

\[
D_H(u_r[i], u_{r-1}[j]) < m^\frac{1}{4}.
\]

Operator \( \mathcal{E}_2 \) replaces \( u_r[i] \) by new random chromosome \( v \) such that

\[
D_H(u_r[i], v) > \frac{m}{2}.
\]

Operator \( \mathcal{E}_2 \) replaces no more than \( n^\frac{2}{3} \) chromosomes.

Selected experimental results are given in Table 1.

<table>
<thead>
<tr>
<th>average number of generations</th>
<th>(10^3)</th>
<th>(10^4)</th>
<th>(10^5)</th>
<th>(10^6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>average number of true clauses for SGA</td>
<td>55 %</td>
<td>59 %</td>
<td>64 %</td>
<td>68 %</td>
</tr>
<tr>
<td>average number of true clauses for GAEE</td>
<td>53 %</td>
<td>61 %</td>
<td>75 %</td>
<td>82 %</td>
</tr>
</tbody>
</table>

Table 1: Experimental results for SGA and GAEE.

ACKNOWLEDGEMENTS. The work was partially supported by Analytical Departmental Program “Developing the scientific potential of high school” 8.1616.2011.

References


Received: December 3, 2012