Coevolving Solutions of the 3-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

Coevolutionary genetic algorithms are genetic algorithms that evolve simultaneously two or more populations with coupled fitness. In this paper we consider coevolutionary genetic algorithms for the 3-satisfiability problem.

Keywords: genetic algorithms, coevolution, satisfiability

Genetic algorithms are a class of search algorithms inspired by the biological process of evolution by natural selection. Along with other heuristics (see e.g. [1, 2]), genetic algorithms are extensively used for solution of various hardly formalizable problems (see e.g. [3] – [14]).

Usually, genetic algorithms evolve a population of candidate solutions to a given problem by iteratively applying stochastic search operators. Coevolutionary genetic algorithms are a subclass of genetic algorithms that evolve simultaneously two or more populations with coupled fitness function (see e.g. [15]).

The satisfiability problem is a core problem in contemporary computer science. Satisfiability algorithms widely used for solution of different computationally hard problems (see e.g. [16] – [29]). In particular, genetic algorithms
for the 3-satisfiability problem is extensively studied and used for solution of other problems (see e.g. [30] – [38]). In this paper we consider coevolutionary genetic algorithms for the 3-satisfiability problem.

Let \( f(z[1], \ldots, z[m]) \) 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

\[
u[i] = u[i, 1] \ldots u[i, m], u[i, j] \in \{0, 1\}, 1 \leq i \leq n, 1 \leq j \leq m.
\]

We can consider \( u[i] \) as a solution for \( f(z[1], \ldots, z[m]) \). In particular, we assume that \( z[j] = u[i, j] \). We consider only 3CNFs. So, we can assume that

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

where \( C[j](z[1], \ldots, z[m]) \) is a clause. We say that \( u[i, j] \) is a true assignment for \( C[k](z[1], \ldots, z[m]) \) if \( z[j] = u[i, j] \) evaluates \( C[k](z[1], \ldots, z[m]) \) to true. Let \( C(u[i, j]) \) be the set of clauses such that \( C[k](z[1], \ldots, z[m]) \in C(u[i, j]) \) if and only if \( u[i, j] \) is a true assignment for \( C[k](z[1], \ldots, z[m]) \).

During each successive generation, a proportion \( P \) of the existing population is selected to breed a new generation. We assume that \( P = \frac{n}{q} \). Individual chromosomes are selected by a fitness function \( F \). After this, the next step is to generate a second generation population of chromosomes. We can use two genetic operators: crossover \( C \) and mutation. In this paper, we consider only crossover. Usually, crossover defines a part of parent chromosome which used for construction of child chromosome. We use standard random crossover. In particular, if \( u[i] \) and \( u[j] \) are two parent chromosomes, then we obtain two child chromosomes:

\[
C(u[i], u[j]) = u[i, 1] \ldots u[i, C(u[i])]u[j, C(u[i])] + 1 \ldots u[j, m],
\]

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

If \( u[i[1]], \ldots, u[i[p]] \) selected to breed a new generation and \( p = 2q \), then

\[
C(u[i[1]], u[i[2]]), C(u[i[2]], u[i[1]]), \ldots,
\]

\[
C(u[i[p - 1]], u[i[p]]), C(u[i[p]], u[i[p - 1]])
\]

is a second generation population of chromosomes. If

\[
u[i[1]], \ldots, u[i[p]]
\]

selected to breed a new generation and \( p = 2q + 1 \), then

\[
u[i[1]], C(u[i[2]], u[i[3]]), C(u[i[3]], u[i[2]]), \ldots,
\]
Coevolving solutions of the 3-satisfiability problem

\[ C(u[i[p - 1]], u[i[p]]), C(u[i[p]], u[i[p - 1]]) \]

is a second generation population of chromosomes. This generational process is repeated until a termination condition \( T \) has been reached. As \( T \) we consider time function. For simple genetic algorithm (SGA), we can use

\[ \mathcal{F}(u[i]) = |\bigcup_{j=1}^{m} C(u[i, j])|. \]

Now, we consider an example. Let

\[ f = (z[1] \lor z[2] \lor z[3]) \land (z[1] \lor z[2] \lor \neg z[3]) \land (z[1] \lor \neg z[2] \lor z[4]) \land \\
(z[1] \lor z[2] \lor z[4]) \land (z[1] \lor z[3] \lor z[4]) \land (z[1] \lor \neg z[2] \lor \neg z[5]) \land \\
(z[2] \lor z[3] \lor z[4]) \land (z[2] \lor \neg z[3] \lor z[4]) \land (z[1] \lor z[2] \lor z[5]), \]


It is easy to check that

\[ \mathcal{F}(u[1]) = 8, \mathcal{F}(u[2]) = 7, \mathcal{F}(u[3]) = 8, \mathcal{F}(u[4]) = 7. \]

In case of SGA, \( u[1] \) and \( u[3] \) are selected to breed a new generation. Clearly, as a second generation population of chromosomes we can obtain only

\[ (10000, 10100, 10000, 10000) \]

or

\[ (10000, 10100, 10000, 10100). \]

So, using SGA, \( f \) can not be assigned in such a way as to make the formula evaluate to true. However, \( u[2] \) and \( u[4] \) can give us a true assignment 01010.

To prevent the loss of such solutions we consider a coevolution of solutions. In particular, let

\[ \mathcal{F}_k(u[i]) = \max_{\{p[1],...,p[k]\} \subseteq \{1,...,n\}} |\bigcup_{1 \leq j \leq m, q \in \{i,p[1],...,p[k]\}} C(u[q, j])|. \]

Using \( \mathcal{F}_k \) instead \( \mathcal{F} \), we can easily transform SGA into coevolutionary genetic algorithm CGA\([k]\). Selected experimental results are given in Table 1.

**ACKNOWLEDGEMENTS.** The work was partially supported by Analytical Departmental Program “Developing the scientific potential of high school” 8.1616.2011.
<table>
<thead>
<tr>
<th>average number of generations</th>
<th>$10^2$</th>
<th>$10^3$</th>
<th>$10^4$</th>
<th>$10^5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>average number of true clauses for SGA</td>
<td>57 %</td>
<td>68 %</td>
<td>77 %</td>
<td>81 %</td>
</tr>
<tr>
<td>average number of true clauses for CGA[1]</td>
<td>56 %</td>
<td>65 %</td>
<td>84 %</td>
<td>92 %</td>
</tr>
<tr>
<td>average number of true clauses for CGA[2]</td>
<td>55 %</td>
<td>63 %</td>
<td>91 %</td>
<td>97 %</td>
</tr>
<tr>
<td>average number of true clauses for CGA[3]</td>
<td>54 %</td>
<td>60 %</td>
<td>81 %</td>
<td>88 %</td>
</tr>
</tbody>
</table>

Table 1: Experimental results for different genetic algorithms.

References


Coevolving solutions of the 3-satisfiability problem


[33] A. Gorbenko and V. Popov, Hamiltonian Alternating Cycles with Fixed Number of Color Appearances, Applied Mathematical Sciences, 6 (2012), 6729-6731.

[34] A. Gorbenko and V. Popov, Footstep Planning for Humanoid Robots, Applied Mathematical Sciences, 6 (2012), 6567-6571.


Received: November 1, 2012