Improving the Efficiency of the Optimization Algorithm Angels & Mortals

R.Bielek, M.Virčíková, J. Jadlovský
Technical University of Košice/Department of Cybernetics and Artificial Intelligence, Košice, Slovakia
radoslav.bielek@tuke.sk, maria.vircikova@tuke.sk, jan.jadlovsky@tuke.sk

***Abstract* — We present an improved algorithm called Angels and Mortals mainly used for NP problem solutions. The primary improvements are several modifications of cloning and mutation process and computation of mortals' lifespan in order to faster the algorithm. We report our experience on the problems of Graph Coloring and Boolean Satisfiability. Comparing to the Classic evolutionary algorithm, the main advantages of the proposed algorithm are more setting parameters and omitting the crossing part to save execution time.**

I. Introduction

Evolutionary and search algorithms are nowadays part of the artificial intelligence used to solve many problems but mostly the *NP* (nondeterministic polynomial time) problems. Exact methods for solving these types of problems are well known, but are inapplicable in real life as the time of the solving rises exponentially with the dimension of the problem. Evolution and searching algorithms are inspired mostly by the selection processes of the fittest individual which take places in the nature, social life or anywhere else, but their aim is not to model these complex events, just to adopt their basic principles.

Because of relatively easy implementation of evolutionary and searching algorithms their popularity is rising and they are more often used for solving the real world problems [4] which solutions have to be evaluable by the so called fitness function.

II. Solved Problems

A. GCP – Graph Coloring Problem

This problem appeared with the need to distinguish the neighbouring countries on the political maps where all the bordering lands had to be distinguished by the colors.

From the mathematical point of view the graph G is created from the set of vertices V and edges E. Solving the graph contains of assigning the color to each vertex until there are no vertices connected with edge which have the same color.

The minimal number of colors needed for solving the graph is called the chromatic number and the ratio between the overall number of edges and the number of possible edges in the whole graph is called density of the graph [5].

Nowadays is the graph coloring still remarkably active field of research ([6],[7],[8]) and can be used in the scheduling problems, bandwidth allocation to radio stations or in the register allocation during the computer program execution [9].

B. SAT – Boolean Satisfiability Problem

This problem belongs to the area of propositional logic and it works with the variables $x_1, x_2, \ldots, x_3$. Consider the l as the literal which can represent x_i or its logical negation $\neg x_i$. These literals take places in the disjunction relations and form the clause ω. By the connection of these clauses with the conjunction relations we will get the *SAT* problem [2]. To get it solved we need to obtain complete proposition with true value so in each clause ω there have to be at least one true literal.

The practical use of the *SAT* problem is the design of complex electrical circuits or in the area of automated theorem proving when the proof for the mathematical theorem is obtained by computers [10].

III. Angels and Mortals Algorithm

Main principle of this searching algorithm is the complex behaviour of an ecological system [1]. The aim is the evolution of the whole system to the best solution.

As the title suggests, the system contains of two individual types: angles – the permanent inhabitants of the world and the mortals – carries of genetic information (solution). These individuals are moving randomly in the torus world mapped on the grid with square cells.

As it was mentioned the immortal angels do not carry the genetic information, so they do not directly participate on the problem solving. Their function is to produce new mortals and take care of the selection process. Each time when the mortal "touches" the angel (in the algorithm implementation it means that they are standing in the neighbouring cells), the mortal is cloned, mutated and placed in the proximity of angel. According to the modification angels can also prolong the life of "touching" mortal.

Mortals are moving randomly in the world for their whole life which lengths are computed according to the fitness of each individual. During the each cycle/generation of the algorithm the lifespan of each mortal is decreased by one and the mortal is removed from the world after its life exhaustion.

Angels and Mortals algorithm allows us to modify its basic functioning by the mutation of each mortal every generation without meeting the angel, but there is a need to recount all the mortals' lifespans. Exhausted lifespans are not forgotten so the mortal cannot life forever – algorithm AM_1.

Basic version of the Angels and Mortals algorithm adapted from [1] is presented below.

```
Angels & Mortals Algorithm():
Begin
    Set MaxGenerations,
    Initialize an n x m world with A angels and M mortals;
    Associate a random solution to each mortal;
    Assign the lifespan of each mortal accordingly to their solution fitness;
    Repeat Until (currentGeneration < MaxGenerations) Do
    Look for the best mortal;
    If (this mortal solution is the global optimum) Then
        Report solution and exit algorithm;
    endIf
    Decrease life counters;
    Reap mortals;
    Move randomly all angels and mortals to a neighbour cell;
    Clone mortals near angels;
    Mutate mortals;
    Recalculate fitnesses and assign new lifespans;
    Increment currentGeneration;
    endDo
End.
```

Fig. 1. The basic Angels & Mortals algorithm

Second modification is the omitting of the cloning process so there have to be a mutation in every generation to modify the genetic code of mortals. Meeting with an angel brings a lifespan prolongation for mortals – algorithm AM_2.

A. Cloning

According to the placement of the newly cloned mortal there can be used several types of cloning. Authors of the [1] implemented type in which the new individual was placed on the "upper" cell of the angel. If it was reserved, the nearest cell in the clockwise rotation was used.This can "pushes" the mortals' population in one direction.

Our modifications consist of random placement in the angel's area or placing new individual in the cell of the parent mortal. Last mentioned type is particularly suitable in situations when there are no new mortals cloned because of no free cells around the angels what can lead to the selection process violation.

B. Mutation

The basic mutation used was one with change of a single gene. For the *GCP* it was for the best possible value (obtained from the connected vertices) and in the case of *SAT* the value was negated.

Mutation of only one gene cannot create enough computational power for solving complex problems in the reasonable time so there were implemented a few more types. In the beginning of the mutation process, all of the genes were ordered by the number of "conflicts" they provide. In the case of *GCP* it means the number of identically colored connected vertices for each graph vertex. For the *SAT*, they are variables mostly used in the untrue clauses.

After ordering the genes, we had several options for mutation. We used mutation of exact number of genes k, random number of genes from interval *0* to k when exactly k genes were changed with probability P_k, and the last one is the special case of the previous one when modifications are present on the *0* to k genes with the same probability.

C. Lifespan assignment

Basic formula for the computation of mortals' life is

$$lifespan = fitness\ (in\ \%). \qquad (1)$$

In the cases of population's fast convergence, most of the mortals would have same lifespan. This can lead to the impractically long life of mortals (especially in the low mutation modifications). From this reason we used formula

$$lifespan = fitness\ (in\ \%) / \lambda, \qquad (2)$$

where λ is the constant number for the entire program execution.

D. Reaper

Each individual is removed from the world in two cases: if his whole lifespan is gone or to faster the selection process and to raise the selection pressure if his fitness does not reach the boundary fitness value F_B.

IV. Experiments and Results

A. Solved Graphs and SATs

There were *2* sizes of solved graphs: *100* vertices graphs (*GCP 100*) with *249* randomly generated edges what makes graph density equals to *5%* and *200* vertices graphs (*GCP 200*) with *499* edges what makes density equals approximately to *2.5%*. Chromatic number of both types was *3* what is also number of colors used in the program execution.

SAT problems were also of two dimensions: one with *50* (*SAT 50*) and second with *100* (*SAT 100*) variables. Both consist of clauses with *3* literals and *SAT 50* was formed from *218* and *SAT 100* from *430* clauses.

All of the experiments were executed *50* times with *10* different graphs or *SATs* what makes *5* tries on each problem.

B. Algorithm Settings

Because of large number of setting parameters it is not possible to try every combination of them. According to that we decided to found the best possible values of parameters by the way of searching only one parameter value, while all of the others remained constant. After that, we switch our attention to next parameter and so on.

In order to obtain as big gene variation as possible we decided to work with the population of *200* mortals.

The best settings of the AM_1 algorithm emerged from the various experiments as follows: world dimensions *29x29* (Fig. 2), *10* angels (Fig. 3), *100* initial and *200* maximum mortals with coefficient $\lambda = 12$ (Fig. 4) and maximal number of generations *10 000*.

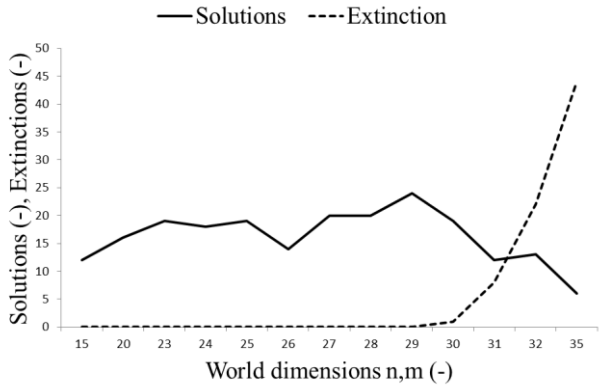

Fig. 2. Solutions and extinctions according to the world dimensions in the test of AM_1 algorithm.

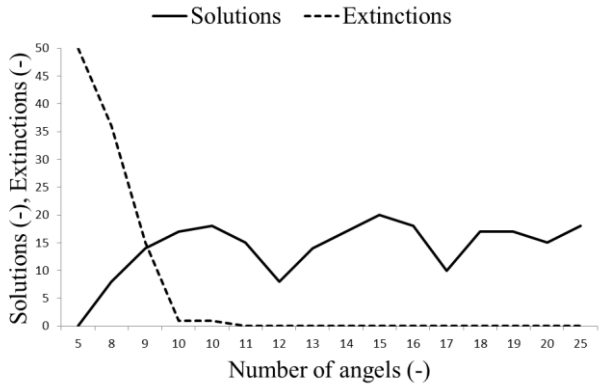

Fig. 3. Number of solutions and extinctions according to the number of angels in the test of AM_1 algorithm.

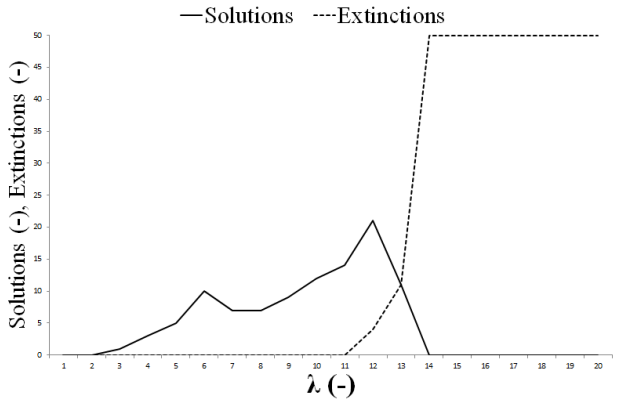

Fig. 4. Number of solutions and extinctions according to the fitness recalculation coefficient λ in the test of AM_1 algorithm.

For the cloning, it was random placement in the area of angel. The best results were obtained when using mutation containing the ordering genes and changing random number of them from the selected interval.

Testing of the mutation type brings the best values for the mutation of *0* to *k* genes with the same probability.

In the addition of mutation probability test (Fig. 5) we can see that there is the rising tendency to find the solution, but also with extension of program execution time therefore all other tests of the AM_1 works with the

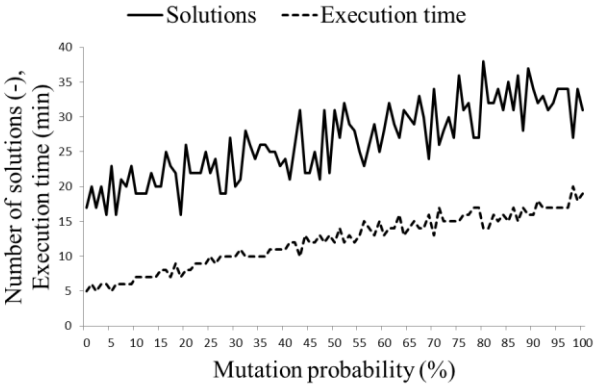

Fig. 5. Dependency of solutions and execution time according to the mutation probability during solving the *GCP 100* in the test of AM_1 algorithm.

50% mutation probability as a compromise between the number of obtained solutions and computational demands.

According to the solved problem, the boundary fitness was set as follows: $F_B = 65$ for *GCP 100*, $F_B = 64$ for *GCP 200*. In the case of *SATs* the boundaries are quite high: $F_B = 84$ for *SAT 50* and $F_B = 88$ for *SAT 100*.

Settings of AM_2 algorithm differs in the world size of dimensions *20x20* with *200* initial mortals (as there are no new mortals created). Lifespan extension was made by adding the *6* units while encounters the angel. Mutation probability was set to *100%* and the $\lambda = 1$. All of the others settings remained as in the case of AM_1.

C. Comparing Evolution Algorithm

For comparing the computational power of the Angels and Mortals algorithms we used evolutionary algorithm (*EA*) as described in [3] and [4], but with exclusion of crossing operator in order to equalize both algorithms.

This *EA* starts with initialization of genes with random values, followed by evolutionary cycle which consists of selection, offspring creation and replacement. Selection process is made of the contest between two randomly selected individuals. The one with higher fitness is moved to the set of parents. Hence, it is possible to have more than one copy of individual in the parents set.

For the offspring creation, the *EA* uses mutation of random number of ordered genes (as in the case of *AM*).

Replacement was just the exchange between offspring set and the actual individuals. Maximum number of generations was *10 000* and the population was formed by *30* individuals.

V. Experimental Results

According to the significant importance of mutation for each algorithm we present the upper boundaries of mutated genes intervals in addition of tasks (Table I).

TABLE I
BEST VALUES OF UPPER BOUNDARIES OF MUTATED GENES ACCORDING TO THE SOLVED PROBLEMS.

	$AM_{1,2}$	EA
GCP 100	9	20
GCP 200	14	25
SAT 50	9	29
SAT 100	14	31

TABLE II
RESULTS OF AM AND EA ALGORITHMS AND THEIR COMPUTATION TIMES ACCORDING TO 100 GENERATIONS IN SECONDS. TIMES OF THE AM2 ARE NOT LISTED AS THEY VARY WITH THE DESCENDING NUMBER OF MORTALS PRESENTED IN THE WORLD.

	AM_1		**AM_2**	**EA**	
	Solutions	Time	Solutions	Solutions	Time
GCP 100	36	0.19	50	45	0.30
GCP 200	5	0.43	19	27	1.21
SAT 50	50	0.54	47	50	0.15
SAT 100	41	1.15	26	45	0.47

Table II shows the comparison of three algorithms according to their success in solving and their computation time.

VI. CONCLUSION

Both *AM* modifications have markedly worse results especially in the solving *GCP 200*, but the time of the solution is on the side of *AM*. For the solving the *100* vertices graphs the AM_2 brings even more solution than *EA* (all *50* vs. *45*).

Comparing the results in the case of *SAT* problems the *EA* shows better performance not even in the founded solutions, but also in the time of computation. We could say that the main reason is the better selection process and smaller set of individuals.

If we take a better look on the mutation process (which is the computationally hardest part of the algorithms, because of the ordering genes) we can see that in the *AM* algorithm with *150* mortals as an average value and with the mutation probability equals to *50%*, there are *75* mutations in every cycle, mutation of newly cloned mortals excluded. With *30* mutations in every cycle in *EA*, we can conclude that it is much more effective than the *AM* and that the selection process with possibility of multiple copies of each individual in the parents set suits much better at least to this problem.

The main con of the *AM* algorithms is also its advantage in the high number of settings parameters. Its user should have better control of the execution, but finding the best values of each coefficient is demanding on time. On top of that several parameters have the very similar effects i.e. the number of angels and the world size. Both of them vary the selection process in the same way: change the possibility of mortal to meet and angel.

What makes the comparison of these algorithms hard is the number of generations. In the classic *EA* on cycle/generation of the program produces new individuals, but it is not true for the *AM* where one cycle/generation can bring new individuals, but also nothing can happen as far as the cloning process is conditioned by the positioning of angels and mortals. If we account the coefficient λ and its used value *12* and the average fitness of mortal as *80*, we obtain the lifespan equals to approximately *7* units. So in order to compare the *AM* and *EA* generation number we should consider this inequality in population replacement.

Interesting feature which *AM* algorithm brings to the field of searching algorithms is the extinction of the whole population. In the case of dead end population, it can save time and start the solving again.

From the experiments presented in this paper we can imagine use of *AM* algorithm in the modification without cloning AM_2. Reason is the decreasing population and their selection only on a few best individuals.

REFERENCES

[1] COMELLAS, Francesc – GALLEGOS, Ruben: Angels & Mortals: A New Combinatorial Optimization Algorithm. In: Studies in Fuzziness and Soft Computing. 2004, nr.166, p.397-405.

[2] LYNCE, Inês – OUAKNINE, Jöel: Sudoku as a SAT Problem. In: Proceedings of the Ninth International Symposium on Artificial Intelligence and Mathematics (AIMATH 2006), 2006.

[3] MACH, Marián: Enhanced Eiben's Operator For Constraint Satisfaction Problem. In: Proceeding of the Ninth International Conference on Soft Computing, Brno: Mendel, 2003. p.243-248.

[4] MACH, Marián: Evolučné algoritmy, Prvky a princípy.Košice: Elfa s.r.o., 2009. ISBN 978-80-8086-123-0.

[5] SARNOVSKÝ, Martin et al.: Farbenie vrcholov grafu [online]. [cit. 2012-11-05]. Online: http://neuron.tuke.sk/~sarnovsky/

[6] BRÉLAZ, Daniel: New methods to color the vertices of a graph, Communications of the ACM, v.22 n.4, p.251-256, April 1979.

[7] SCHNEIDER, J.: A new technique for distributed symmetry breaking. Proceedings of the Symposium on Principles of Distributed Computing, 2010.

[8] BARENBOIM, Leonid - ELKIN Michael: Distributed (δ+1)-coloring in linear (in δ) time. Proceedings of the 41st annual ACM symposium on Theory of computing, May 31-June 02, 2009, Bethesda, MD, USA.

[9] CHAITIN, G. J.: Register allocation & spilling via graph coloring, Proceedings of the 1982 SIGPLAN symposium on Compiler construction, p.98-105, June 23-25, 1982, Boston, Massachusetts, United States.

[10] DAVIS, Martin – LOGEMANN, George – LOVELAND, Donald: A machine program for theorem-proving, Communications of the ACM, v.5 n.7, p.394-397, July 1962.