An Evolutionary Framework for 3-SAT Problems

István Borgulya

University of Pécs, Hungary

In this paper we present a new evolutionary framework for 3-SAT. This method can be divided into three stages, where each stage is an evolutionary algorithm. The first stage improves the quality of the initial population. The second stage improves the speed of the algorithm periodically generating new solutions. The 3rd stage is a hybrid evolutionary algorithm, which improves the solutions with a local search. The key points of our algorithm are the evolutionary framework and the mutation operation that form a concatenated, complex neighborhood structure, "a variable neighborhood descent". We tested our algorithm on some benchmark problems. Comparing the results with other heuristic methods, we can conclude that our algorithm belongs to the best methods of this problem scope.

Keywords: evolutionary algorithm, 3-SAT.

1. Introduction

A Boolean satisfiability problem (SAT) involves a Boolean formula F consisting of a set of Boolean variables $x_1, x_2, \ldots, x_n$. The formula F is in conjunctive normal form and it is a conjunction of m clauses $c_1, c_2, \ldots, c_m$. Each clause c_i is a disjunction of one or more literals, where a literal is a variable x_j or its negation. A formula F is satisfiable if there is a truth assignment to its variables satisfying every clause of the formula, otherwise the formula is unsatisfiable. The goal is to determine a variable x assignment satisfying all clauses.

The class k-SAT contains all SAT instances where each clause contains exactly k distinct literals. While 2-SAT is solvable in polynomial time, k-SAT is NP-complete for $k \geq 3$ [8]. The SATs have many practical applications (e.g. in planning, in circuit design, in spin–glass model, in molecular biology [6], [4], [11]) and especially many applications and research on the 3-SAT is reported. Many exact and heuristic algorithms have been introduced.

Exact algorithms are able to determine if a problem is satisfiable or unsatisfiable, but they have an exponential worst-case complexity. Such algorithms are e.g. the splitting algorithms, which reduce the problem for the input formula F to the problem for polynomial many formulas $F_1, F_2, \ldots, F_p$, and make a recursive call for each or one of F_i's (detailed in [5])

Heuristic algorithms can find the solutions quickly, but they are not guaranteed to give a definite answer to all problems. The two most common heuristics are the stochastic local search (SLS) and evolutionary algorithms (EAs). The generic procedure of an SLS algorithm for SAT initializes the search at some truth assignment and then iteratively flips some variables truth value, where the selection of the variables depends on the input formula and the current assignment. If after a given maximum of flips no solution is found, the algorithm restarts (possibly more than once) and tries to find a solution. The main difference between SLS algorithm-variants for SAT is implementation of the selection of the variable to be flipped. The most known SLS algorithm families are the GSAT [17] and the WalkSAT [16]. Their known variants are, for example, the GSAT versions, namely the GWSAT, GSAT/TABU and the HSAT – or the WalkSAT versions, namely WalkSAT/TABU, Novelty, R-Novelty (detailed in [12]).

The EAs and its variants (e.g. genetic algorithm (GA)) have successful applications, too. However, only those algorithms have good re-

sults that consider the characteristics of the SAT problem and apply additional techniques. These techniques include adaptive fitness functions, problem-specific mutation operators and/ or local optimization procedures (as hybrid EA) [10]. So, for example, the SAWEA [1], the RFEA and its versions RFEA2, RFEA2+ are based on adaptive fitness functions [9] and use problem-specific mutation operators. The FlipGA [13] and ASAP [15] use the MAXSAT fitness function and a local search procedure (The MAXSAT fitness value is equivalent to the number of satisfied clauses).

In this paper, we present a new heuristic method to solve the 3-SAT. This heuristic is an EA framework (EF) that consists of 3 consequent phases to reach the convergence quickly. Each phase is an EA using concatenated, complex neighborhood structures for the mutations. The complex neighborhood structure is the variable neighborhood descent (VDN) [14], which enables us to concatenate varying, different neighborhoods. The efficacy of the method was studied on 3-SAT benchmarks: problems of 20 – 100 dimensions were all successfully solved. Comparing the results to those of a few EAs or SLS algorithms, we can state that the results of our new method are similar or even better.

In section 2, we describe the EF in general, and report the implementation details of our EAs. In Section 3 we present the new algorithm, and in Section 4 we demonstrate our computational experience and compare our results with other heuristic's results. Section 5 contains concluding remarks.

2. The Evolutionary Framework

Hybrid EAs are frequently used for solving combinatorial problems. These methods improve the quality of the descendant solution, for example, by the application of a local search procedure, SA, or TS. Constitution of these systems corresponds to the extension of an EA: for instance, a local search procedure is applied at every step of the EA cycle.

In two of our previous works we also developed algorithms using hybrid EAs, one for solving the quadratic assignment problem (QAP) and one for the single machine total weighted tardiness-scheduling problem (SMTWTP) [3], [2]. We applied the same EF in both cases that we want to use in solving the 3-SAT now.

Unlike former hybrid EAs based on a single stage, our EF uses a 3-stage algorithm structure, in order to speed up convergence and to produce higher quality results. The first stage is a quick "preparatory" stage that is designated to improve the quality of the initial population. The second, "quick search" stage is usually a hybrid EA that continuously enlarges the population while searching. The size of the population is extended in order to speed up the convergence. Finally, the third stage, the "slow search", is a hybrid EA that continues the search for better solutions, leaving the size of the population constant.

Therefore the EF comprises three steady-state EAs. Let us discuss the 3 EAs (stages) in more detail:

1. The first stage EF forms some solutions at random and then tries to improve them by randomly generated descendents. The descendant may replace the most similar one in the former solutions.

2. In the second stage the algorithm uses a VDN structure as mutation. Quality of the solutions is improved with a local search procedure. In selecting the parents (solutions), priority is given to the best, highest objective/fitness function value: the algorithm selects the fittest solution with 0.5 probability and another solution with $0.5/t$ probability (where t is the number of solutions).
   At certain tasks this neighborhood structure is not always enough for the complete run: the algorithm might get "stuck" at one of the local optima. To help the solution the algorithm generates new solutions. A new solution is generated randomly from an already existing one and the new variants can improve the capability and the speed of the algorithm to find the global optimum. Thus, in the second phase new solutions are periodically generated in the neighbor hood of the existing ones until a maximum number of solutions are completed.

3. In the third stage the solutions are further improved by the local search procedures employed in the second stage. Meanwhile, it generates new solutions, which are to replace the weakest solutions. Solutions with the smallest objective function values are considered to be the weakest ones and a given percentage of them will be periodically deleted.

In case of the QAP and the SMTWTP we took specific nature of the problems into consideration, so we used different recombination, mutation (neighborhood structure) operations, and different parameter values (e.g. population size).

To solve the 3-SAT, we adapted the EF operations to the problem, too. So we chose bit string representation, where each variable is associated to one bit. We used the MAXSAT fitness function, three different recombination operations, and a VDN structure as mutation. The recombination operation was discrete or single-point recombination with the probability of 0.8, or otherwise simple copy-making. We applied a varying number of *bit-flips*, a new *bit-flip-flop*, and a new *bit-repair* transformation. The applied transformations are the following:

bit-flip: a single bit (variable) is flipped randomly in the descendant.

bit-flip-flop: if there are two randomly chosen bits, the i*th* and j*th* variables in the descendant having different values, the bits i and j will be flipped.

bit-repair: this transformation sequentially selects all the clauses that are not satisfied yet and flips a single uniformly chosen variable in the selected clause. (It is possible that a new bit-flip spoils the previous correction).

We form the VDN structure

some bit-flip-flop + some bit-flip +bit-repair

of the various transformations, where the *bit-flip-flop* transformation is executed only with the probability of 0.5, and the *bit-flip* and *bit-flip-flop* transformations are executed varying times.

The local search procedure is a *1-opt* local search: the *1-opt* neighbor solution is reached by flipping a single bit in a current solution x. Therefore, the hamming distance $d^H(x, x') = 1$. The local search that examines the 1-opt neighborhood for the 3-SAT is called the *1-opt-local-search*. This search sequentially flips all variables that can improve the solution. The search is rerun until it is possible to improve the quality of the solution.

Finally, we added a check to the EF. In order to keep the diversity of the population, we selected only the best of the solutions close to each other, the other ones were deleted (x and x' are close to each other if the $d^H(x, x')$ is less than a predefined value, e.g. $n/4$). This filtering speeds up the convergence, too.

The new algorithm constructed this way, named EF_3SAT (Evolutionary Framework for the 3-SAT) may be used both for small and medium scale 3-SATs. Its structure and function are different from the former methods used to solve the 3-SAT, as it uses the EF structure, and it applies a special VDN transformation for the mutations.

3. The New Algorithm

Let us introduce the following notations:

- Let us denote the 3 stages as EA1, EA2 and EA3.
- Let the population of the it*th* generation be denoted by P(it), and let $x_1, x_2, \ldots, x_t$ be the individuals of the P(it) population.
- Let us denote the procedure responsible for the generation of a new solution by *Newind(q)*. The procedure generates a new q solution (applied selection, recombination and mutation operators) if $t<tmax$.
- Let us denote the complex neighborhood transformation (*bit-flip-flop+ bit-flip+ bit-repair)* of the q descendant by *Nhs(q)*. The first two transformations are executed randomly, maximum $n/2$ times. The probability of applying the first transformation is 0.5.
- Let us denote the procedure which swaps the worst *ddp* percent of the solutions with new ones by *SortDel*.

- The measure of the similarity of the two solutions x and z is given by $H(x, z) = 1/(1 + d^H(x, z))$ where $d^H(x, z)$ is the Hamming distance of the solutions.
- Let us denote the procedure, which filters and deletes the close and poorer solutions ($d^H(x, x') < n/4$) by *Filter.*
- Let us designate the random number generator as Rnd (uniform distribution on (0,1)).

Parameters

6 parameters affect the run of the algorithm:

- *tmax* – the maximal size of the population.
- t – the size of the population in the first stage.
- *itt* – a parameter of the second stage. If the number of iterations (it) reaches *itt*, the second stage begins.
- *kn* – a parameter which determines the timing of checks. Finding the best solution, swapping the weakest solution with new ones occur only at each *kn*th iterations. The size of the population is expanded only at each *kn*th iteration too.
- *ddp* - a parameter of the third stage: the worst *ddp* percent of the solutions will be swapped with the new ones.
- *maxbitflip* - a parameter for the stopping condition. The procedure is finished if the number of bit-flips (by mutation and local search) is greater than *maxbitflip*.

Variables:

it – the number of the iterations, *glob* – index of the most accurate solution.

Procedure EF_3SAT(*tmax, t, itt, kn, ddp, maxbitflip*, opt, optp)
* EA1 **********
$it := 0,\ glob := 1.$
Let $x_i \in \{0, 1\}^n (i=1, \ldots, t)$, $\mathrm{P}(it) \leftarrow \{x_1, \ldots, x_t\}$.
Compute $f_{\mathrm{MAXSAT}}(x_1), \ldots, f_{\mathrm{MAXSAT}}(x_t)$.
Do *itt* **times**
  Selection of a $q \in S$.
  Compute $f_{\mathrm{MAXSAT}}(q)$.
  Let $H(q, x_z) := max_j H(q, x_j); j, z \in \{1, 2, \ldots, t\}$
  If $f_{\mathrm{MAXSAT}}(q) > f_{\mathrm{MAXSAT}}(x_z)$ **then** $x_z := q$ **fi**.
  $it := it + 1, \mathrm{P}(it) \leftarrow \mathrm{P}(it - 1)$.
od.
*EA2 **********
Repeat
  Do *kn* **times**
    Two parents x_i, x_j selection
    Generation of the q descendant.
    Nhs(q).
    Compute $f_{\mathrm{MAXSAT}}(q)$.
    If $f_{\mathrm{MAXSAT}}(q) > f_{\mathrm{MAXSAT}}(x_i)$ **then** $x_i := q$ **fi**.
    $it := it+1, \mathrm{P}(it) \leftarrow \mathrm{P}(it\text{-}1)$.
  od.
  Filter, Newind.
  Let $f_{\mathrm{MAXSAT}}(x_i) := max_j f_{\mathrm{MAXSAT}}(x_j)$
  $i, j \in \{1, 2, \ldots, t\},\ glob := i,\ opt := f_{\mathrm{MAXSAT}}(glob),$
  $optp := x_{glob}$
  If "the number of bit-flips" > *maxbitflip* **then**
    exit fi. /* Stopping condition
until $t < tmax$.
* EA3 **********
Repeat
  Do *kn* **times**
    Two parents x_i, x_j selection.
    Generation of the q descendant.
    $Nhs(q)$, Local search
    Compute $f_{\mathrm{MAXSAT}}(q)$.
    If $f_{\mathrm{MAXSAT}}(q) > f_{\mathrm{MAXSAT}}(x_i)$ **then** $x_i := q$ **fi**.
    $it := it + 1, \mathrm{P}(it) \leftarrow \mathrm{P}(it - 1)$.
  od.
  Filter, SortDel
  Let $f_{\mathrm{MAXSAT}}(x_i) = max_j f_{\mathrm{MAXSAT}}(x_j)$
  $i, j \in \{1, 2, \ldots, t\},\ glob := i,\ opt := f_{\mathrm{MAXSAT}}(glob),$
  $optp := x_{glob}$
until "the number of bit-flips" > *maxbitflip*.
exit
end

4. Computational Experience

Test problems

We have tested the EF_3SAT with a benchmark set. This set consists of random 3-SAT instances and it is available on the Internet (`http://www.in.tu-lausthal.de/~gottlieb/benchmarks/`). The benchmark set has 3 parts (suite 1, 2 and 3). In suite 1 there are 3-3 instances, in suite 2 50-50 instances and in suite 3 100-100 instances per problem size. This collection of benchmark set was used in the publications of e.g. [9], [10]).

Parameter selection

To achieve a quick and accurate solution we need appropriate parameter values. Studying problems of suite 1 of the benchmark set we analyzed how the parameter values were affecting the convergence and the finding of the global optimum. In this analysis we could make use of the experiences of solving the QAP and the SMTWTP. Although the parameter values became very different, the analysis leading to them was very similar.

Summarizing the results of the analysis we found the following parameter values appropriate:

- the size of the population may be the same for all the problems ($n \leq 100$) and t=10, $tmax$=30.
- the frequency of the checks (*kn* parameter) and percentage of the elements deleted in stage 3 (*dpp* parameter) may be the same for any number of dimensions. The kn=10 and dpp=0.05 values are appropriate.
- the number of iterations in the first stage is, in turn, task-dependent: up to 50 dimensions itt=50 is appropriate, in case of 50-100 dimensions itt=300 is appropriate. (Better results can be achieved for more difficult problems by setting itt=900.)
- The value of *maxbitflip* is also task-dependent. Here, we accepted the testing practice of allowing equal number of *flips* for any test problems *(maxbitflip=300 000).*

Solutions to the test problems

We tested the EF_3SAT to all test problems ($n \leq 100$) of the benchmark set. All problems (tasks) were run 10 times, and the results present mean values calculated from the runs. The achieved quality was measured by the *success rate* (SR), which represents the portion of runs where a solution was found. The computation cost of the EF-3SAT could be estimated using the *average number of flips to solution* (AFS).

We present the results of EF-3SAT in comparison with other methods. As for comparison itself, we chose different EAs and SLS methods. As SLS we chose the WSAT, and as EA the SAWEA, RFEA2, RFEA2+, FlipGA and ASAP algorithms. The comparison was carried out by means of SR and AFS measures. We used the results of Gottlieb et al. (2002), because they used the SR and AFS measures in their comparison as well. We also adopted the *maxbitflip* value that we set it to 300 000, too.

We compared the average results. The Tables 1, 2 and 3 show comparative results for problems of the suites 1, 2 and 3. In the tables we give the algorithm name, the SR and AFS values by dimensions, averaged over the tasks.

Comparing the SR values we can conclude that the EF_3SAT solves the problems of 20-60 dimensions successfully, like other algorithms do. As regards the problems of 75-80-100 dimensions, we experienced bigger differences. As we increased the number of dimensions, the results of the EF_3SAT were getting superior.

	n=30		n=40		n=50		n=100	
	SR	AFS	SR	AFS	SR	AFS	SR	AFS
WSAT	1.00	1631	1.00	3742	1.00	15384	0.80	19680
SAWEA	1.00	34015	0.93	53289	1.00	60743	0.72	86631
RFEA2	1.00	3535	1.00	3231	1.00	8506	0.99	26501
RFEA2+	1.00	2481	1.00	3081	1.00	7822	0.97	34780
FlipGA	1.00	25490	1.00	17693	1.00	127900	0.87	116653
ASAP	1.00	9550	1.00	8760	1.00	68483	1.00	52276
EF_3SAT	1.00	12506	1.00	18314	1.00	38713	0.80	100060

Table 1. Results for benchmark suite 1.

	n=50		n=75		n=100	
	SR	AFS	SR	AFS	SR	AFS
WSAT	1.00	16603	0.84	33722	0.66	23853
RFEA2	1.00	12053	0.95	41478	0.77	71907
RFEA2+	1.00	11350	0.96	39396	0.81	80282
FlipGA	1.00	103800	0.82	29818	0.57	20675
ASAP	1.00	61186	0.87	39659	0.59	43601
EF_3SAT	1.00	17300	0.92	47382	0.93	57437

Table 2. Results for benchmark suite 2.

	n=20		n=40		n=60		n=80		n=100	
	SR	AFS	SR	AFS	SR	AFS	SR	AFS	SR	AFS
WSAT	1.00	334	1.00	5472	0.94	20999	0.72	30168	0.63	21331
SAWEA	1.00	12634	0.89	35988	0.73	47131	0.52	62859	0.51	69657
RFEA2	1.00	356	1.00	3015	0.99	18857	0.92	50199	0.72	68053
RFEA2+	1.00	365	1.00	2951	0.99	19957	0.95	49312	0.79	74459
FlipGA	1.00	1073	1.00	14321	1.00	127520	0.73	29957	0.62	20319
ASAP	1.00	648	1.00	16644	1.00	184419	0.72	45942	0.61	34548
EF_3SAT	1.00	2260	1.00	5587	1.00	10165	0.99	10863	1.00	35516

Table 3. Results for benchmark suite 3.

This tendency was only broken by the problems of suite 1 of 100 dimensions, where our algorithm yielded less good solutions – their quality was equally good as those of the WSAT. In case of the 100 dimension problems of suite 2 and suite 3 it was definitely the EF_3SAT that solved them most successfully. Analyzing the AFS values we can state that the EF_3SAT is not among the quickest algorithms. The AFS values of different algorithms vary substantially, and the values of the EF_3SAT are among the medium ones.

We can conclude that our algorithm is similarly successful in solving individual problems as other algorithms. It reaches its best results in some problems of 100 dimensions, where results are much better than in the ones of other algorithms. So we can state that our algorithm is among the best proposed heuristic algorithms for the problem addressed.

5. Summary

In this paper we presented a new heuristic algorithm, named EF_3SAT for solving the 3-SAT problem. The structure of the new algorithm is an evolutionary framework consisting of 3 stages, which is the structure we have already successfully applied in solving other combinatorial problems. For the solution to 3-SAT problems we applied problem-specific operators and a local search procedure. So we used a complex neighborhood structure for the mutations, where we could concatenate different neighborhoods.

We can conclude that the EF_3SAT was successfully tested with different kinds of 3-SAT. Comparing the results with those obtained with other heuristic methods (EAs, SLS), we can conclude that the EF_3SAT belongs to the best methods for solving this scope of problems.

6. Acknowledgements

The study was supported by the Hungarian Research Foundation OTKA T 042448.

References

[1] BÄCK T., EIBEN E., VINK M.E., A Superior Evolutionary Algorithm for 3_SAT. In Porto V.-W, Saravanan N., Waagen D., Eiben E., (eds), *Proc. of the 7th Annual Conference on Evolutionary Programming*, Lecture Notes in Computer Science, Springer, Berlin, (1998) 1447, pp. 125–136

[2] BORGULYA, I., A Cluster-based Evolutionary Algorithm for the Single Machine Total Weighted Tardiness-scheduling Problem, *Journal of Computing and Information Technology* **10**. 3. (2002) pp. 211–217.

[3] BORGULYA, I., A Heuristic Method for the Quadratic Assignment Problem, *Central European Journal of Operations Research* **11**. 1. (2003) pp. 3–16.

[4] CRISANTI A., LEUZZI L., PARISI G., The 3-SAT problem with large number of clauses in the ∞-replica symmetry breaking scheme, *J. Phys. A: Math. Gen* 35 (2002) pp. 481–497.

[5] DANTSIN E., HIRSCH E.A., IVANOV I., CSEMIRNOV M., *Algorithms for SAT and Upper Bounds on Their Complexity*, Electronic Colloquium on Computational Complexity, Report No. 12 (2001).

[6] DU D.,GU J., PARDALOS P., (eds), *Satisfiability Problem: Theory and Applications*, (1997) Vol. 35. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, AMS, Providence, Rhode Island.

[7] EIBEN E., VAN DER HAUW J., Solving 3-SAT with Adaptive Genetic Algorithms, in *Proc. of the 4th IEEE International Conference on Evolutionary Computation* (1997) pp. 81–86.

[8] GAREY M., JOHNSON D., *Computers and Intractability: a Guide to the Theory of NP-completeness*, Freeman, San Francisco, CA (1997).

[9] GOTTLIEB J., VOSS N., Adaptive fitness functions for the satisfiability problem, in Schoenauer M. (ed), *Proc. of the 6th International Conference on Parallel Problem Solving from Nature*, Lecture Notes in Computer Science, Springer, Berlin,(2000) 1917, pp. 621–630.

[10] GOTTLIEB J., MARCHIORI E., ROSSI C., Evolutionary Algorithms for the Satisfiability Problem, *Evolutionary Computing* **10** (1) (2002) pp. 35–50.

[11] HAGIYA M., ROSE J.A., KOMIYA K., SAKAMOTO K., Complexity analysis of the SAT engine: DNA algorithms as probabilistic algorithms, *Theoretical Computer Science* 287 (2002) pp. 59–71.

[12] HOOS H.H., STÜTZLE T., Local Search Algorithms for SAT: An Empirical Evaluation, *Journal of Automated Reasoning*, (2000) 24. pp. 421–481.

[13] MARCHIORI E., ROSSI C., A Flipping Genetic Algorithm for Hard 3-SAT Problems, in *Proc. of the Genetic and Evolutionary Computation Conference*, (1999), Vol. 1., pp. 393–400.

[14] REEVES C.R., Landscapes, operators and heuristic search, *Annals of Operations Research*, (1999) 86. pp. 473–490.

[15] ROSSI C., MARCHIORI E., KOK N.J., An Adaptive Evolutionary Algorithm for the Satisfiability Problem, in Caroll J. et al. (eds), *Proc. of the ACM Symposium on Applied Computing*, ACM, New York, (2000) pp. 463–469.

[16] SELMAN B., KAUTZ H., COHEN B., Noise strategies for improving local search, in *Proc. of the 12th National Conference on Artificial Intelligence* (1994) AAAI Press, CA, pp. 337–343.

[17] SELMAN B., LEVESQUE H., MITCHELL D., A new method for solving hard satisfiability problems, in *Proc. of the 10th National Conference on Artificial Intelligence* (1992) AAAI Press, CA, pp. 440–446.

Received: June, 2003
Accepted: September, 2003

Contact address:
István Borgulya
University of Pécs
Rákóczi ut 80
7621 Pécs, Hungary
e-mail: borgulya@ktk.pte.hu

ISTVÁN BORGULYA is an associate professor at the Department of Business Informatics, University of Pécs, Hungary. He received the Ph.D. degree in computer and law from the University of Pécs too. His research interests include modeling in law with the methods of artificial intelligence, fuzzy methods in the decision support, and optimization with evolutionary algorithms.