Solving the SAT problem with a genetic algorithm and a heuristic method

Teodor Simionescu

1 Abstract

There is not trivial solution to solve SAT because The Boolean satisfiability problem (SAT) is a nondeterministic polynomial-time complete (NP-complete) problem. Further more, the SAT problem will be approached using the Simulated Annealing method and a Genetic Algorithm.

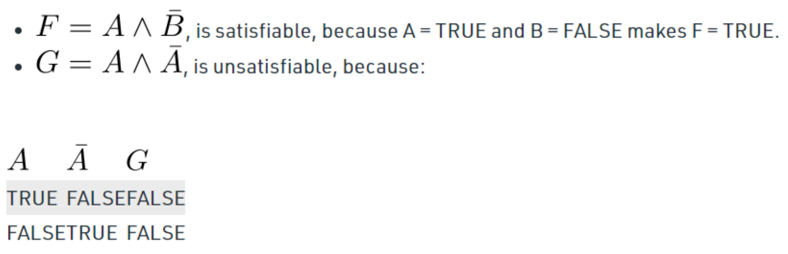
To understand better which method is more likely to work with the SAT problem, the results obtained are compared side by side. All this are closed in a conclusion that is drawn in the final part.

2 Introduction

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 class k-SAT contains all SAT instances where each clause contains upto k literals. While 2-SAT is solvable in polynomial time, k-SAT is NP-complete for k greater than 3."*[Aru].

With general belief that no time efficient local optimization search algorithm exists, heuristic SAT-algorithms are the preffered approach in finding a solution as they can compute a result in a shorter time.

SAT involves a boolean formula consisting of a set of variables. Usually, the form has a conjunction of clauses, with each clause being a disjunction of literals. This formula is satisfiable if there exists a truth assignment to the variables such that each clause is satisfiable. The goal is to determine a permutation of values to the variables such that the whole formula given is satisfiable. There are two approaches which will be used in this paper.



- $F = A \wedge \bar{B}$, is satisfiable, because A = TRUE and B = FALSE makes F = TRUE.
- $G = A \wedge \bar{A}$, is unsatisfiable, because:

A	$\bar{A}$	G
TRUE	FALSE	FALSE
FALSE	TRUE	FALSE

Fig. 1. Example of a boolean formula

3 Methods

3.1 Algorithm

– Simulated Annealing
  Simulated Annealing provides the option to choose a 'bad' neighbour (is NOT satisfying the give condition), based on a 'temperature' that gets decreased with each iteration. This method can be similar to that in metallurgy, where metals are shaped at a high temperature and detailed at a lower temperature. The temperature will be at the beginning 100, and will stop when is lower or equal of 10^{-9}. To assure the accuracy, the temperature will be multiplied by 0.98765 (is an arbitrary value). In this manner, the halting condition will be approached only after a few thousands changes. Therefore, the probability of accepting a worse solution decreases with the temperature
– GA
  A Genetic Algorithm (GA) is a metaheuristic inspired by the process of natural selection. Genetic Algorithms are a population-based approach, as they maintain and improve multiple candidate solutions, often using population characteristics to guide the search and relying on bio-inspired operators such as mutation, crossover and selection. In order to keep the survival of the test concept, to each individual we apply a fittness function to rate their quality. This fittness function must be chosen such that the value of this function is higher for the best solutions. This way, the better individuals are given a higher chance of selection over generations evolving better solutions until reaching a stopping criterion.

3.2 Implementation

The experiment started with a simple Simulated Annealing and a simple Genetic Algorithm. The classic Simulated Annealing method performed as expected, but for the classic genetic algorithm things got complicated as the space got bigger. Therefore, to optimise the results, a greedier method of improving the chromosome were added: uniform crossover and greedy mutation.

3.3 Classic GA

The classic genetic algorithm used was a simple one where the mutation was made with a fixed probability (0.01) and with a single cut point crossover. When the number of clauses is a large number, the pressure of selection is too low to be able to actually improve the population over generations. Therefore, this method is best used only when the number of clauses and variables are reduced.

3.4 Updated GA

As mentioned above, the problem appears when big instances have to be solved. The previous algorithm shows that the chromosome is not evolving fast enough. To improve that, one possible solution is to flip each bit that improved the number of clauses satisfied for a solution instead of random mutations one. Another possible solution would be to improve the chromosome until no more improvement was found. Because first option proved to be computationally expensive and the second option showed a decent convergence towards solutions, the second option was implemented. To balance the time, the number of generations was reduced. Finally, in order to assure an extended search space, a uniform crossover was implemented.

3.5 Details

Simulated Annealing

- Encoded in binary form, the number of variables = the length of the solution
- The initial temperature is 100

- The halting condition is 10^{-9}. Additional stopping condition after a number of 100 iterations without any improvement - matters when big instances are used.
- Temperature is multiplied by 0.98765 to guarantee that the search is happening for multiple times.
- Inner loop is an iteration of 10.000 steps.

Genetic Algorithm

- The population (100) is a set of fixed-length bit strings equal with the number of variables in the formula.
- The quality of the chromosome is evaluated based on the number of clauses satisfied.
- Selection proportional with the fitness value (Roulette-Wheel).
- Resulting chromosomes replace their parents. 1000 Generations
- Mutation function:
  Flip multiple bits - iterate through the chromosome and flip each bit with a certain probability. Probability if 0.1
  Greedy flip - flip every bit of a chromosome that improves the chromosome(left to right)
- Cross-over
  Single cut point crossover - change the substring of two chromosomes enclosed by a random selected pivot selected and the last bit. Probability of 0.7
  Uniform cross over - exchange each pair of bits between two chromosomes at a fixed 0.5 probability

4 Experiments

The experiments were conducted on 10 different instances ranging from 20 variables and 91 clauses up to 1534 variables and 132295 clauses.

5 Conclusion

Genetic algorithm has the following advantages: It can find good solutions even if the search space is large or the problem is difficult to solve. It is a flexible approach that can be applied to a wide range of

Table 1. Results

Instance: uf20-01.cnf				
Method	avg time	avg result	minim result	maxim result
SA	4.164615	100%	100%	100%
Updated GA	2.256473	100%	100%	100%
Instance: uf50-01.cnf				
Method	avg time	avg result	minim result	maxim result
SA	13.279344	99.9%	99%	100%
Updated GA	331.581861	96.7%	96%	97%

Table 2. Results

Instance: uf75-01.cnf				
Method	avg time	avg result	minim result	maxim result
SA	11.105896	100%	100%	100%
Updated GA	274.941877	95%	95%	95%
Instance: uf100-01.cnf				
Method	avg time	avg result	minim result	maxim result
SA	23.506873%	99.4%	99%	100%
Updated GA	467.300625	94%	94%	94%

Table 3. Results

Instance: uf125-01.cnf				
Method	avg time	avg result	minim result	maxim result
SA	63.140127	99.4%	99%	100%
Updated GA	1292.229563	93.2%	93%	94%
Instance: uf150-01.cnf				
Method	avg time	avg result	minim result	maxim result
SA	21.072992	100%	100%	100%
Updated GA	1224.688313	93%	93%	93%
Instance: uf175-01.cnf				
Method	avg time	avg result	minim result	maxim result
SA	95.446	99.1%	99%	100%
Updated GA	1391.912875	92.6%	92%	93%

Table 4. Results

Instance: uf200-01.cnf				
Method	avg time	avg result	minim result	maxim result
SA	137.6023	99%	99%	99%
Updated GA	3558.81475	92.8%	92.0%	100%
Instance: uf225-01.cnf				
Method	avg time	avg result	minim result	maxim result
SA	113.9513	99.3%	99%	100%
Updated GA	5382.166438	92%	92%	92%
Instance: uf250-01.cnf				
Method	avg time	avg result	minim result	maxim result
SA	77.0234	99.6%	99%	100%
Updated GA	2999.923	95%	92%	98%

SAT problems. It can be parallelized, which can improve the speed of the algorithm. The disadvantages:It may not find the optimal solution to the problem, only an approximate solution. It may require a large number of iterations to find a satisfactory solution, which can be time-consuming. It may require a significant amount of fine-tuning of the GA parameters in order to achieve good results.

When speaking about Simulated annealing, some of the advantages are: It can find good solutions even if the search space is large or the problem is difficult to solve. It is a flexible approach that can be applied to a wide range of SAT problems. The drawbacks:It may not find the optimal solution to the problem, only an approximate solution. It may require a large number of iterations to find a satisfactory solution, which can be time-consuming. It may require a significant amount of fine-tuning of the SA parameters in order to achieve good results.

The beauty of SAT is based on the fact that a solution is valid only when all the clauses are true. Even if both algorithms are indeed really close to satisfy all the clauses (somewhere at 90% success rate), it doesn't really matter. As long as a solution do not have all the clauses true, it is not a valid solution.

The updated version of GA includes a mutation operator which resembles a greedy improvement (mutate all bits as much as possible and then keep the best out of the chromosome) and a uniform crossover. These two processes are complementary and they allow to explore the search space and to exploit particular interesting areas in a more effective manner. Moreover, this combination of crossover and aggressive mutation ensures a sufficient diversity in the involved population. The GA has been evaluated on a set of 10 instances and has been compared with SA. Both algorithms are providing very interesting results.

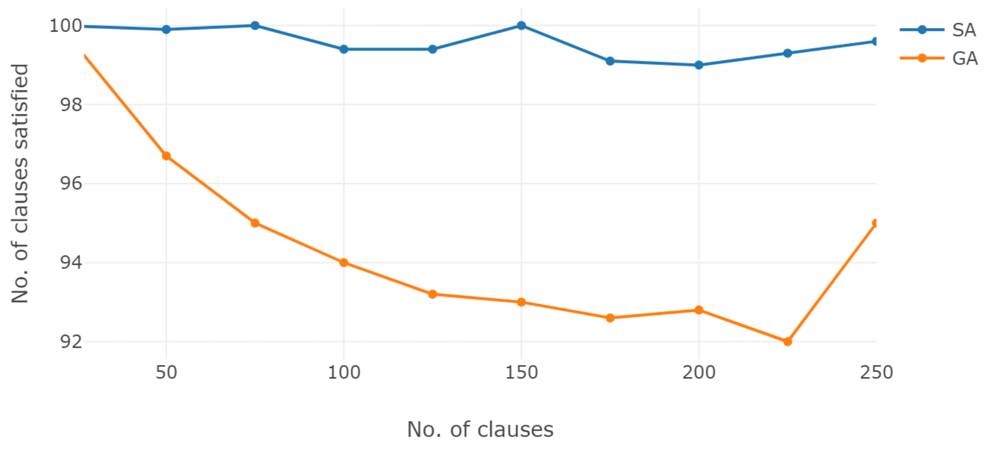


Fig. 2. Diagram of the results

No doubt that for this record SA is overall better than GA, but that can not be a general case. GA was highly limited on number of generations and size of the population. Indeed, in order to scale up the results a compromise with the time execution must have been made.

Overall, the decision to use a genetic algorithm or simulated annealing to solve a SAT problem will depend on the specific characteristics of the problem and the trade-off between the potential benefits and drawbacks of each approach.It is also possible to use hybrid approaches that combine both techniques in order to take advantage of their complementary strengths.

References

[Aru] Prabal Chauhan Arunava Bhattacharjee. *Solving the SAT problem using Genetic Algorithm*. URL: `https://www.astesj.com/publications/ASTESJ_020416.pdf`.

Course support (https://profs.info.uaic.ro/ eugennc/teaching/ga/HomeworkEn)
Text editor (https://www.overleaf.com/)
An Improved Adaptive Genetic Algorithm for Solving 3-SAT Problems Based on Effective Restart and Greedy Strategy (https://www.atlantis-press.com/journals/ijcis/25888772/view)
Diagram editor (https://www.mathcha.io/editor)