Fuzzy Genetic for 3-CNF-Satisfiability Problems

Isaac Teoh Yi Zhe
Institute of Computer Science
Malaysia
isaacteohyizhe@student.usm.my

Selvaraj Kanimozhi
Institute of Computer Science
Malaysia
kanimozhipraveen@student.usm.my

ABSTRACT

The research paper present a hybrid method based on fuzzy system and genetic algorithm in solving 3-CNF-Satisfiability Problems. The idea is that the fuzzy method is integrated to FlipGA as a way to solve problems particularly local optimal solution or slower population diversity that may be encountered in genetic algorithm. The algorithm implement the population diversity and evolutionary speed to modify the crossover disruption rate automatically based on fuzzy logic. The hybrid algorithm are being compared with the predecessor algorithm of FlipGA to identify whether there is improvement made. The performance evaluation are based on the number of generation the genetic algorithm in solving the 3-SAT problems and the success rate in solving the problems. It is found that the predecessor FlipGA slightly outperformed Fuzzy FlipGA by eight out of in solving ten proposed benchmark problems.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics • **Networks** → Network reliability

KEYWORDS

ACM proceedings, text tagging

*Produces the permission block, and copyright information
†The full version of the author's guide is available as acmart.pdf document
[1]It is a datatype.

WOODSTOCK'97, July 2016, El Paso, Texas USA

https://doi.org/10.1145/123_4

ACM Reference format:
G. Gubbiotti, P. Malagò, S. Fin, S. Tacchi, L. Giovannini, D. Bisero, M. Madami, and G. Carlotti. 1997. SIG Proceedings Paper in word Format. In *Proceedings of ACM Woodstock conference, El Paso, Texas USA, July 1997 (WOODSTOCK'97)*, 4 pages. https://doi.org/10.1145/123 4

1 INTRODUCTION

The Satisfiability Problem (SAT) is one of the most prevalent and systematically studied combinatorial problems. Many applications particularly hardware and software verification, logistics or scheduling have executed satisfiability problems.

A SAT instance shows formula in propositional logic form. In propositional logic, there are propositional variables that can allotted with value true or false. Literal is a variable that exist in positive or negative form. Literals are joined with "logical operators especially conjunctions ∧, the logical "and" and disjunctions ∨, the logical "or" to construct a propositional formulae".

When reviewing SAT instances, particularly with SAT solver in evaluation and development, then these instances are limited to "conjunctive normal form (CNF)". Clauses are" a formula F in CNF consists of disjunctions over literals" where it is combined using conjunctions.

The benefit of demonstrating a SAT instance as a formula in CNF is when a formula in CNF is satisfied with just one true the clause are satisfied. Since the conjunctions are combining the clauses, so all the clause need to be satisfied to satisfied the problems.[1] In this case, the 3-CNF-Satisfiability problem is used as the benchmark problem to be solved. 3-CNF-Satisfiability contain 3 literals within a clause.

There are many existing methods that can solved the 3-CNF-Satisfiability problems. It is important to study some of the methods that may solve the problem. So, in next section the literature review of the methods are study to generate some idea in solving the problems.

2 LITERATURE REVIEW

2.1 Genetic Algorithm

Genetic Algorithm (GA) are a universal "search technique based on the" procedure of "natural selection which" join the "survival of the fittest" method with some mutation and/or randomization. This method make use of the solution space characteristics, consequential in somewhat "fast convergence to a global optimum or most-fit solution". GA are good in "solving optimization problems" especially "kinetic parameters". GA has the parallel search feature, which are a proficient "method for optimization over a huge parameter space". Next, GA operates with a "group of population", each representing a "candidate solution" to a certain "optimization problem", but "not directly with the real parameters to be optimized". So GA is good in optimization problems where the fitness function is not identified for as it "do not require knowledge of the gradient of the fitness functions". GA are

shown to be outstanding in compromising between flexibility, globality and convergence based on many research on it [2].

2.3 Fuzzy system

Fuzzy systems are "universal approximators to algebraic functions" and it can estimate or understand the system behaviour in which the analytic functions or numerical relations are devoid[3].

Fuzzy systems are "based on fuzzy sets and fuzzy logic". Fuzzy system uses fuzzy sets and its operations for problem solving by using the if then rules to make deductions or decisions. "Fuzzy systems can be divided into two types namely Mamdani-type and Takagi-Sugeno type". Mamdani-type also known as linguistic system represented as:

"If x_1 is A_1 and x_2 is A_2 and … and x_n is A_n then y is B"

"Takagi-Sugeno type also known as functional type is represented as:

If x_1 is A_1 and x_2 is A_2 and … and x_n is A_n then y =f(x_1, x_2,.., x_n)"

[4].

Fuzzy systems is also a complex system that can involve with human state especially with social and economic system or biological and medical system or political systems, in which the wide array of inputs and outputs are not feasible to control or secure analytically in any predictable sense. There are many lacking of functionality in a fuzzy system. Firstly, membership functions are fully depended by Fuzzy Inference System (FIS). Fuzzy systems have been effectively used in many areas particularly automated control, data classification, expert system, pattern recognition and etc [5].

3 METHODOLOGY

3.1 Genetic Algorithm

There are many types of genetic algorithm that has been used to solve the 3-SAT Problems. In this case, FlipGA is used to solve the 3-SAT Problems.
The flowchart of FlipGA are shown as follows in Figure 1:

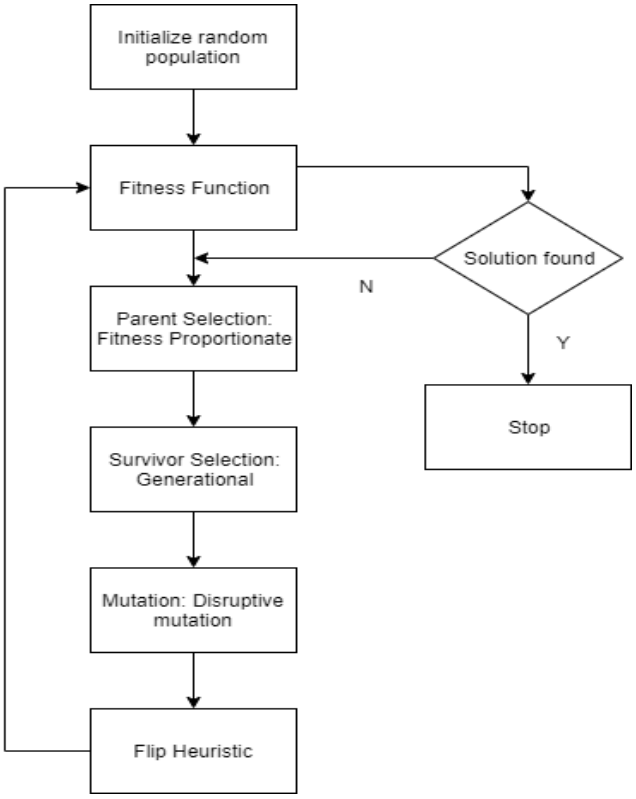

Figure 1: Flowchart of FlipGA Algorithm

The following shows the genetic scheme of FlipGA:

Algorithm 1: FlipGA

```
1: Initialize random population
2: While i <Max Generation
       Termination criteria satisfied
3:     Fitness Proportionate selection
4:     Generational replacement
5:     Disruptive mutation
6:     Flip Heuristic
7: end while
```

The core features of FlipGA are summarized as follow:

1. **Representation scheme.**

   Binary Representation (T & F)

2. **Parent population size**.

   20

3. **Offspring population size**.

   Similar as the number of population-20.

4. **Parent selection mechanism**.

   Fitness proportional. [6] Selecting new but same number of list of parents from the list of population based on if random value of probability generated is smaller than the

probability which is continuously sum of the probability of fitness value over the sum of fitness value. It is possible the same parents are selected more than once. The formula given is:

$$p(i) = \frac{f(i)}{\sum_{j=1}^{n} f(j)}$$

where each individual i of the current population a probability p(i) of being selected, proportional to its fitness f(i) [7].

5. **Survivor selection mechanism**.

   Generational: A generational replacement scheme with elitism, copying the best two individuals of the current population into the population of the next generation[6]

6. **Reproductive operators.**

   Random disruptive mutation and bit flip mutation: FlipGA using FlipHeuristic and population reference the best value based on fitness value indexing the parent selection. FlipGA mutation basically keep flipping bits from left to right which increase the fitness of that individual. Once the right end is reached, again start from the left end until no more flipping is possible[8] [9] The core of FlipGA is the flip heuristic applied to each individual after performing mutation. The heuristic scans the genes in random order: each gene is flipped, and the flip is accepted if the gain (that is, the number of clauses that are satisfied after the flip minus the number of clauses that are satisfied before the flip) is greater than or equal to zero. When all the genes have been considered, the process is repeated if the fitness of the obtained chromosome has been increased with respect to the previous scan of the genes. [6]

7. **Initialization.**

   Random value but values(T/F) generated are to provide a linear distribution of T/F values by discretize f where f(x) = x where x is number of true values within range of 0 up to variable count into uniform width partitions of k. Each partition corresponds to the value in a population the truncated mid in a partition to set the initial number of true values in a corresponding value.

   The starting variables are distributed randomly and then transpose for shuffling only the first axis. Such subpopulation diversity reduces the chance that all population prematurely converge to the same poor quality solution. This approach is equivalent to simply taking the best solution after multiple executions of the SGA on different initial populations

8. **Fitness function design.**

   Evaluate all the literals in all the clause. If found just one true variable in a clause then add 1 satisfied count. If the number of true value is equal to the full number of clause return the value. If not continue.

9. **Stopping criterion.**

   If found solution return the value and success rate while if not then evaluation of 30000 entity is done. If population selection is 20 then iterations of more than 1500 return false.

10. **Parameter control**.

    The offspring production=Population size-propagation limit must be even number. Its better the population size around 10 to 20 as if too many populations might result in longer computation time and affect the efficiency of algorithm. Highly disruptive mutation operator is used with probability 0.9 and the operator flips each gene with probability of 0.5 [8]. The disruptive mutation operator should be only around 0.8 to 1.0 but must not be too small.

3.2 Fuzzy Algorithm

The purpose of adding the fuzzy logic controller is to adjust the disruptive mutation rate dPm dynamically for avoidance of precociousness appearance in genetic algorithm as displayed in Figure 2:

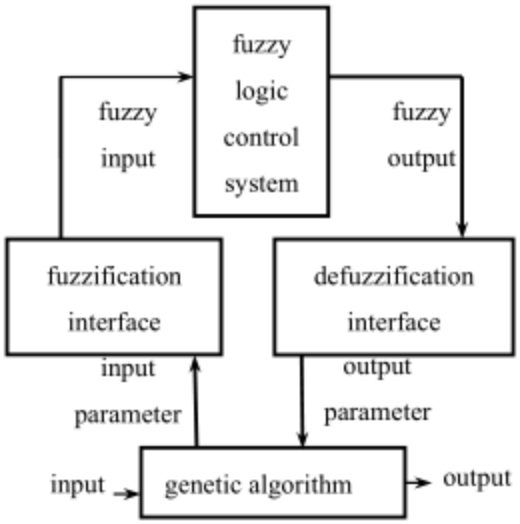

Figure 2: Structure of Fuzzy Genetic Algorithm

At the initial generation, the genetic algorithm provides the input parameters to the fuzzification interface, then the parameters are converted into fuzzy state inputs and then transmitted to the fuzzy logic control system through the interface, then the inference operation formed the fuzzy state output. The output parameter is thought as the input parameter of current generation from the defuzzification interface[10].

In FlipGA operation, it initially computes the evolutionary process value and the population diversity to acquire the evolutionary process as fuzzy variable and average distance from the fuzzification interface as fuzzy variable for deduction through the fuzzy logic controller. Through the defuzzification process, the disruptive mutation rate are obtained. Therefore, good diversity of generation and high fitness of population are maintained. The process are repeated until the termination criteria is met and the maximum generation number is reached.

In the proposed genetic algorithm, the primary factors are the evaluation of the convergence rate and population diversity and acquiring the fuzzy logic control system parameters.

3.2.1 Algorithm implementation.

The "evolutionary process" is characterised by the current population fitness ratio which surpasses the last population fitness. [10]

The equation is presented as:

$$ES(t+1) = \frac{\frac{1}{N}\sum_{i=1}^{n} FitnessValue(t+1, i) - \frac{1}{N}\sum_{i=1}^{n} FitnessValue(t, i)}{\frac{1}{N}\sum_{i=1}^{n} FitnessValue(t, i)} \quad (1)$$

where N=population size, the FitnessValue(t,i) =fitness value for individual i in generation t.

The population diversity is characterised by the average distance to the population centre as presented as:

$$AD(t) = \frac{1}{N}\sum_{i=1}^{n}\sqrt{(x_{p,1}-x_{i,1})^2 + (x_{p,2}-x_{i,2})^2 + \ldots + (x_{p,n}-x_{i,n})^2} \quad (2)$$

$$x_{p,j} = \frac{1}{N}\sum_{i=1}^{n} x_{i,j}$$

where N=population size, the $x_{p,i}$ = average of vector i, the $x_{i,j}$ = vector j of individual i. The centre of the population is a vector presented as P = ($x_{p,1}$,$x_{p,2}$,…,$x_{p,n}$) and the AD =average distance between the individuals and the vector P

The following are the fuzzy variable for mutation disruption rate, 1-F_{pm} where F_{pm} ϵ {Lower, Low, Medium, High, Higher}, the mutation disruption rate is designated not to be too high or the GA will be a random search algorithm. Based on Figure 3 for membership of mutation disruption rate, the range of Lower= (0.0, 0.05, 0.01), Low= (0.08, 0.1, 0.12), Medium= (0.11, 0.13, 0.15), High=(0.14,0.16,0.18), Higher=(0.17, 0.18, 0.2)

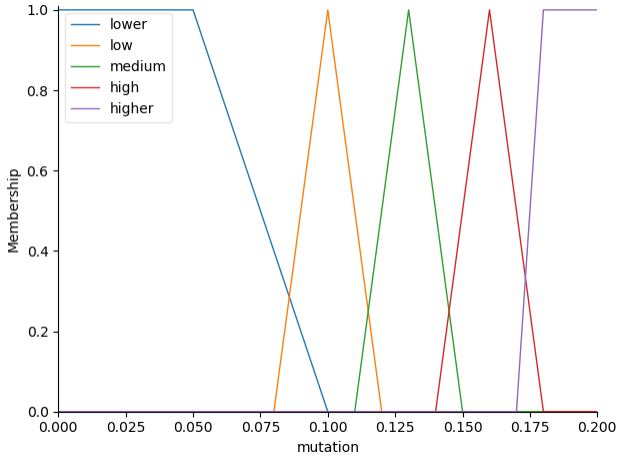

Figure 3: Membership for mutation disruption rate

The following are the fuzzy variable for evolutionary process ES where F_{ES} ϵ { Low, Medium, High, }. Based on Figure 4 for membership of ES, the range of Low= (0, 0.5, 1), Medium= (0.8,1.4, 2), High= (1.8, 2.4, 3)

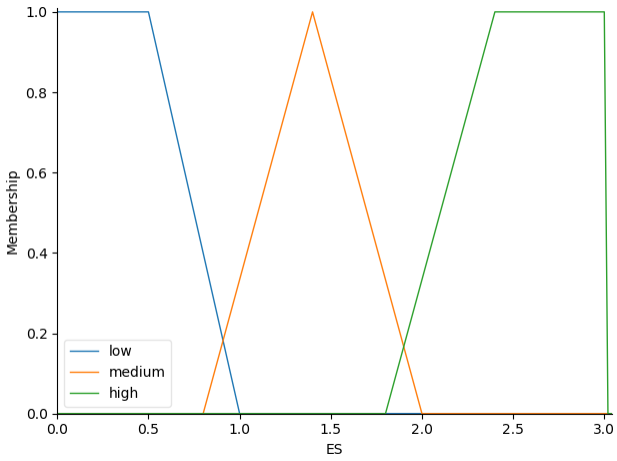

Figure 4: Membership function of ES.

The following are the fuzzy variable for average distance AD where $F_{AD} \epsilon$ { Low, Medium, High, }. Based on Figure 5 for membership of AD, the range of Low= (0, 0.1, 0.3), Medium= (0.2, 0.4, 0.6), High= (0.4, 0.7, 1)

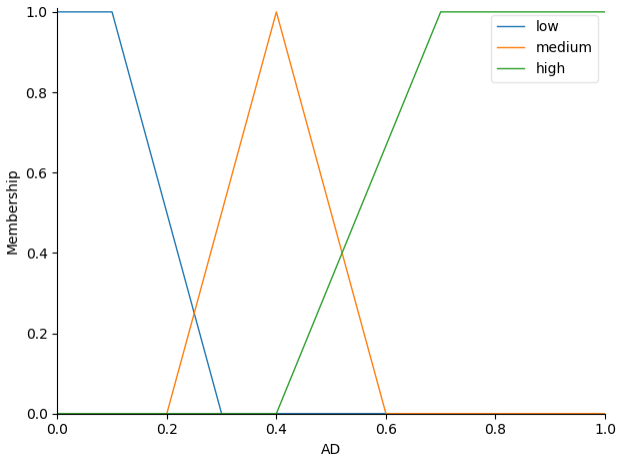

Figure 5: Membership function of AD.

Figure 6 exhibit the computation of the fuzzy system to find the mutation disruptor rate.

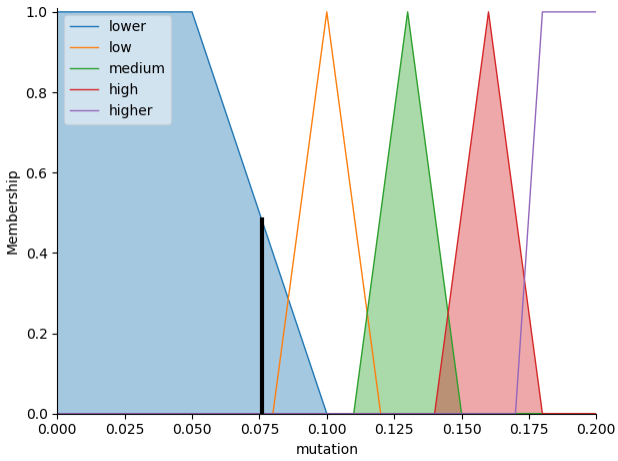

Figure 6: Computation of Mutation rate using tip.

For the research, the fuzzy rules are set as presented in Table 1 for mutation disruption rate in FlipGA.

Table 1: Fuzzy Rules for mutation disruptor rate

ES / Pm / AD	Low	Medium	High
Low	Lower	Low	High
Medium	Low	Medium	High
High	Medium	High	High

Table 1 is used for fine-tuning the mutation disruption rate. Lower evolution process would cause the population to be in the precocity state. Therefore, the disruptor mutation rate are decreased for precocious state prevention. Higher evolutionary process means the population has many superior individuals, thus mutation rate disruptor are increase to keep the superior individuals and deconcentrate the population. When there's higher population diversity, the mutation disruptor should be increased for evolutionary process validation. When cases where both evolutionary process and population diversity are low, the population need to be in precocity state, the mutation rate disruptor needs to be lower. However, when the evolutionary process is low and the population diversity is high then the mutation disruptor rate should be increased to check for probable local optimal solutions in the population [10].

4 RESULTS AND DISCUSSION

In this section, the FlipGA and fuzzy FlipGA presented in Section 3 are tested on the ten benchmark suites taken from http://www.cs.ubc.ca/~hoos/SATLIB/benchm.html. For each benchmark problems, the maximum evaluations are 30000 and

the results are averaged over 20 runs. The experiment evaluations are based on the success rate and number of generations used to solve the problems. The experiment simulation is implemented in Python with Pycharm used as an ide.

Table 2 and Table 3 display the results of the experiments based on the 10 benchmark problems for FlipGA and Fuzzy FlipGA.

Table 2: Experiments on the 10 Benchmark problems for FlipGA

N=20				N=50			N=70		
01	029	050	070	010	050	082	01	067	073
Success Rate									
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Generation									
6	15	38	2	25	95	11	28	568	13

Table 3: Experiments on the 10 Benchmark problems for Fuzzy FlipGA

N=20				N=50			N=70		
01	029	050	070	010	050	082	01	067	073
Success Rate									
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Generation									
7	17	44	3	26	71	13	33	533	15

Based on the Table 2 and Table 3, FlipGA are slightly better than the Fuzzy FlipGA on solving problems faster with a 8/10 benchmark problem solving using the least generation. Only benchmark problem of N=50-050 and N=70-67 does FlipGA lose out to Fuzzy FlipGA in solving the problem faster, It could be that after integrating the fuzzy to the FlipGA to improve the disruptive mutation rate, the IDE runs low on memory and it affect randomize value and its performance. Most of the results solved faster by FlipGA are also based on the good randomize diversity of population. Take for example in N=50-050, some of the generations FlipGA solved the problem are at the 2nd generation and it also solved the problem at the 150th or 303th generation.

While in Fuzzy FlipGA, there are 2 benchmarks where it betters FlipGA. In N=70-67 and N=50-050, the disruptor mutation rate seems to be mostly fixed at low rate of 0.07579710144927536 meaning the 3CNF satisfiability are at low/medium population diversity and at medium/low evolutionary process. It is a possibility that the benchmarks itself even when randomize it is better to have disruptor mutation rate at low rate to solve the problems faster.

5 CONCLUSION

In summary, both experimental and theoretical study of the genetic algorithm and fuzzy system are discussed in the research to explore the feasibility of implementing a hybrid system that improves the performance in solving the 3CNF-satisfiability problems. The experimental results are proved to be efficaciously implemented to study the comparison of hybrid and non-hybrid methods in solving the 3CNF-satisfiability problem. FlipGA without integrating fuzzy system still performs better than the hybrid system in terms of speed in solving the 3CNF satisfiability problems.

ACKNOWLEDGMENTS

Finally, I would like to thank the Almighty God for giving me insights and opportunity to grow throughout the research. I would like to thank Prof. Dr. Wong Li Pei and Dr. Mohd Nadhor Ab Wahab for invaluable comments and discussions to find an interesting and feasible projects.

REFERENCES

[1] F. Lonsing, “SAT A Plugin for HeuristicLab,” no. July, 2005.

[2] H. Huang and W. Su, “Application of micro-genetic algorithm for calibration of kinetic parameters in HCCI engine combustion model,” *Front. Energy Power Eng. China*, vol. 2, no. 1, pp. 86–92, 2008.

[3] T. J. Ross, *Fuzzy logic with engineering applications*. 2017.

[4] S. Kar, S. Das, and P. K. Ghosh, “Applications of neuro fuzzy systems: A brief review and future outline,” *Appl. Soft Comput. J.*, vol. 15, pp. 243–259, 2014.

[5] H. Singh *et al.*, “Real-Life Applications of Fuzzy Logic,” *Adv. Fuzzy Syst.*, vol. 2013, pp. 1–3, 2013.

[6] J. Gottlieb, S. A. P. Ag, E. Marchiori, and C. Rossi, “Evolutionary

Algorithms for the Satisfiability Problem," vol. 10, no. 1, pp. 1–16, 2002.

[7] K. Jebari, "Parent Selection Operators for Genetic Algorithms," no. November 2013, 2014.

[8] E. Marchiori and C. Rossi, "A Flipping Genetic Algorithm for Hard 3-{SAT} Problems," *Proc. of the {G}enetic and {E}volutionary {C}omputation {C}onf. {GECCO}-99*, pp. 393–400, 1999.

[9] A. Bhattacharjee and P. Chauhan, "Solving the SAT problem using Genetic Algorithm," *Adv. Sci. Technol. Eng. Syst. J.*, vol. 2, no. 4, pp. 115–120, 2017.

[10] H. X. Zhang, B. Zhang, and F. Wang, "Automatic fuzzy rules generation using fuzzy genetic algorithm," *6th Int. Conf. Fuzzy Syst. Knowl. Discov. FSKD 2009*, vol. 6, pp. 107–112, 2009.