Improve the Evolutionary Algorithm Search Efficiency with SAT Problem

Rasha Abdelkawy, Dr. Walid Gomaa, Dr. Soheir foad

Abstract— Since the 1990s, the use of incomplete algorithm for solving the SAT problem has grown quickly. Even though the incomplete algorithm is unable to prove unsatisfiability, but it may find solutions for a satisfying problem quickly. In this paper, the improvement of GA Performance in solving the 3-SAT Problem was our main objective and it is shown that the GA can be more efficient if SAT problem-knowledge is oriented in the GA encoding phase, and the GA operators is tuned according to the encoding phase gained knowledge. In this aspect, a novel evolutionary local search algorithm is developed. In this paper the first Part of our research will be presented, that Part of improve the EAs search Performance with the SAT Problem, that was integrated later in evolutionary local search algorithm. The EAs Enhancement was assessed using a set of well-known benchmarks that includes instances with different sizes, and compared with blind EAs algorithm.

Index Terms— Genetic Algorithms, Preprocessing algorithm, Genetics encoding, SAT Problem, Incomplete Solver.

1 INTRODUCTION

Given a conjunctive normal form (CNF) propositional formula where the formula is defined as a conjunction (AND, ∧) of clauses, where each clause is a disjunction (OR, ∨) of literals, and each literal is either a variable or its negation (NOT, ¬). The Boolean Satisfiability Problem (SAT) is defined as the following: Given a CNF formula F, does F have a satisfying assignment? Interest in Satisfiability is expanding for a variety of reasons, not in the least because nowadays more problems are being solved faster by SAT solvers than other means [14].This is probably because Satisfiability stands at the crossroads of logic, graph theory, computer science, computer engineering, and operations research. Thus, many problems originating in one of these fields typically have multiple translations to Satisfiability and there exist many mathematical tools available to the SAT solver to assist in solving them with improved performance [15]. Techniques have two-sided error, they cannot determine that a formula is unsatisfiable, and may not find a solution when the formula is satisfiable.

One of the key motivations for studying incomplete techniques was the finding that complete algorithms perform quite poorly on certain randomly generated formulas. Also, most of the incomplete methods can work as a solution approach for the problem of maximum satisfiability, or MAX-SAT problem by providing the "best found" truth assignment upon termination.

Incomplete methods are based on heuristic algorithms, such as algorithms based on translation to Integer Programming, Finite Learning Automata, local search, simulated annealing, tabu search, and evolutionary algorithms [1,3,17]. A more comprehensive description of Boolean Satisfiability and basic algorithms to solve is provided at [13]. There have also been attempts at hybrid approaches that explore combining ideas from two or more heuristic algorithms, such as Evolution Algorithms and local search techniques [4, 5, 6, 7, and 8].

The rest of this paper is organized as follows: Section 2, presents a historical resume of handling SAT problem with Evolutionary Algorithms and Local search. In section 3, a detailed description of the GA representation issue for the satisfiability problem is presented plus the proposed algorithm and its implementation details. In section 4, the conclusions of this work are discussed. Finally, in last section, the experiment study designed and results are presented.

Currently there are two categories to classify the SAT Solution techniques. Complete and Incomplete techniques, each technique has his own advantages and disadvantages in solving different SAT instances. A complete solution technique is the one that provides the guarantee that it will eventually either report a satisfying assignment or declare that the given formula is unsatisfiable. Despite the worst-case exponential run time of all known algorithms for these category techniques, they have the advantage of always providing proofs of unsatisfiability. An incomplete solution technique is the one that typically run with a pre-set resource limit, (e.g. Max Iteration Number), after which they either produce a valid solution or report failure. Despite the quickly efficient solutions that can be found, the incomplete

- *Rasha Abdelkawy is currently pursuing PhD degree in Department of computer Science, faculty of Engineering, Alexandria University, Egypt, E-mail: rabdelkawy@informatica-me.com*
- *Dr. Walid Gomaa: Department of Computer Science and Engineering, Japan University of Science and Technology Alexandria – Egypt "Currently on leave from the Faculty of Engineering, Alexandria Universi"*
- *Dr. soheir foad. Proof in Department of computer Science, faculty of Engineering, Alexandria University, Egypt*

2 HANDLE SAT PROBLEMS WITH EVOLUTIONARY ALGORITHMS AND LOCAL SEARCH.

Evolutionary algorithms (EAs) are heuristic algorithms that have been applied to SAT and many other NP-complete problems. Some negative results question the basic ability of EAs to solve SAT. De Jong and Spears [1] proposed a classical genetic algorithm (GA) for SAT and observed that the GA may not outperform highly tuned, problem-specific algorithms. This result was confirmed experimentally by Fleurent and Ferland[2], who reported scarce performance of pure GAs when compared to local search. Recent results showed that EAs can nevertheless yield good results for SAT if equipped with additional techniques to overcome the weaknesses of classical GAs. A local search method is incorporated into the evolutionary algorithm to improve individuals [18]. While GAs operators help the Search process to explore the SAT solution domain without stuck with local optimum points.
There have been many attempts to solve SAT problem using hybrid approaches that combining ideas from Evolutionary Algorithms and local search techniques [2,4,5,7,8,18]. They are all relying on pure random initial populations. While the approaches SAWEA [18] and RFEA [4] are based on adaptive fitness functions, other EAs [7, 2] are based on a problem-specific variation operators and a tabu search stage. While recent technique are based on local optimization [5, 8], our research focus on the EAs capability to handle the SAT problem. We investigate the possibility of increase the EAs search efficiency by encoding the problem and use this encoding technique to tune the EAs operators (parent selection, mutation, crossover, and replacement scheme).

3 SOLVING SAT PROBLEMS WITH GENETIC ALGORITHMS

The most obvious way to represent a solution candidates for SAT are binary strings, each of which has a sequence of 0 or 1, of length n, where every variable is associated to one bit. The position of variable in the chromosome depend on it's appear in the SAT Problem clauses. The suggested encoding would transforms the original SAT problem by rearranging the variables to satisfy the condition that the most related ones are in closer positions inside the chromosome, [20,21] was the first research to suggested that, and reported that their proposed approach outperforms blind GAs in all test accomplished. That encourages us to believe that the GA encoding (representation) is a primary aspect of enhancing the GAs performance in SAT problem. Also, The schema theory [23,25] implicitly lists Perquisites features that a representation should exhibit in order to utilize a GA search, namely that with an above average probability, short, low-order schemata will combine and form a higher-order co-adapted schemata. Both of previous researches intuitive use to encode the SAT problem and use this encode to building blocks that enhance the search Process.
GA like many natural systems, assume a certain holistic structure, a structure where the whole different from the sum of its parts. So knowing the value of the parts does not necessarily enable the calculation of their effect together. In GA epistasis is used to indicate the extent of nonlinearity and interdependency among the element composing the representation. Epistatsis, is a form of interaction between nonallelic genes in which one combination of such genes has a dominant effect over other combinations [24].
In [20, 21] work an Epistasis Reducer Algorithm (ERA) is presented. It is used to preprocess SAT instances for reducing the epistasis of its representation and in this way to improve the performance of a simple genetic algorithm (using classical crossover) when used to solve SAT formulae. First, the SAT problem represented by a weighted hypergraphs, where each clause was represented by a hyperedge and each variable is represented by a node, and the weights between two nodes represent how many times the corresponding variables is related (the number of clauses both variables appear in) . Second, the bandwidth minimization problem for graphs (BMPG) is used to find a labeling for the vertices of a graph, where the maximum absolute difference between labels of each pair of connected vertices is minimum. After variables remap according to the ERA, a simple genetic algorithm which uses classical genetic operators applied on the recoded SAT problem.

Our approach is based on the intuitive that a strong epistatic relation may exist in shape of genes groups within a genotype, rather than between genes and each other. That would condense brief view of such interactions rather than a detailed epistatic relation like the previous research. We suggest using a cluster algorithm to connect related genes into groups. Where high intra-cluster similarities express a strong epistatic relation between each cluster members and low inter-clusters similarity express weak relations between the genes in different clusters. In computation complexity wise, the bandwidth minimization problem is an NP-complete problem which will increase the overall SAT problem complexity, while the suggested cluster method, to encode the SAT problem, has O (n2) complexity [26]. Further, we intend to use each instance cluster-result as a guide to tune the GA Operators.

Next the main implementation details of the cluster Algorithm and the GA are described.

3.1 The Clustering Algorithms

A satisfiability problem is expressed as a Boolean formula over a set of variable. Suppose that we have satisfiability problem that is expressed as a Boolean formula with m clauses, over a set of n variable. The objective is assigning the SAT-set variables into groups so that the variables in the same cluster are more similar to each other than to those in other clusters. We are interested in developing enhanced k-mediods cluster technique [26, 27] that should work fast and efficient for large sparse data sets. Assume the appropriate cluster number is K; the proposed algorithm is composed of the following steps:

Step 1: Build Similarity Matrix

The similarity matrix should reflect the relation between every pair of all instance variables, for simplicity, we choose to represent the similarity between two-variable as the number of clauses both variables appears in. this calculation does not recognize the different of related variables or related variables' negation.

Step 2: Select initial medoids

After calculating the similarity matrix we will order the variables descending according to the most related to other variables, we would not take the sum of similarity between that variable and the others as a reference, rather we would use the count of variables related to that variable with a similarity value larger than zero. That means that first k-element would contain the most centered "popular" K-variables to others, and those would be considered the k-mediods for our algorithm.

Step 3: Build Clusters-relation Matrix

For each variable have similarity value larger than zero with one of the clusters mediods, the cluster-relation between that variable and any cluster is calculated as the similarity value between that variable and that cluster mediods. By normalize variable clusters-relation vector, so that the sum of the relations between the variable and all clusters equal one, we express the degree of the adherence of the variable and each cluster.
Other variables that do not have direct relation with the cluster-mediods, the clusters-relation value would be represented as the sum of the similarity value between that variable and every element in the cluster (multiplied by normalized variable clusters-relation vector).

Step 4: Build Clusters members

To generate the final clusters, assign each variable to the closest cluster, the cluster that has the largest clusters-relation with that variable.

That was a greedy enhancement for the k-mediods technique; it may not be the best cluster technique to use, but it works fast and more efficient in sparse large SAT instances than the normal iterative k-mediods technique. The most sensitive setting parameter is the cluster number, which depend on each dataset characteristics.

3.2 Label the variables

The As a preliminary step toward implement the GAs algorithms on the SAT instance we need to rearrange the instance variables according to the cluster result to satisfy the condition that most related ones are in a closer position inside the chromosome.
First we would order the variables belong to the same cluster with the following rule, For each cluster, the mediod variable is labeled as the center position of that cluster and other variables would be rearranged according to the distance from variable to the medoid. So that variables with higher similarity value with the mediods would be labeled to a closer position to the center of the cluster.
Finally, the intra-cluster distance between generated clusters should be used to rearrange the clusters' members into the chromosome. As center cluster, the cluster has the lowest intra-cluster distance with other clusters, its member would labeled with the center positions in the chromosome, and other clusters would be rearranged according to the intra-cluster distance between that cluster and center cluster.
By the above step we guarantee that the most related cluster to other generated-clusters is in a middle position in the chromosome, so that its variables members would be in the middle positions for other SAT instance variables. The same technique was used to re-arrange variable inside the same cluster, most related "popular" variable, cluster mediod, is in a middle positions for other variables.

3.3 Apply GA for solving SAT problems

In brief, the evolution theory portrays the potential solutions (after encode the internal problem variables) to satisfy the SAT formula as individuals, each solution is evaluated, by external function, to check how close that solution to the optimal solution. Best fitted individuals are then selected, Parent selection step, to generate new population through making crossover between individuals and applying mutation to the offspring to avoid premature convergence. In order to demonstrate the benefits to preprocess SAT instances using cluster technique, the knowledge gain used to tune genetic operators as the following:

Chromosome Definition: The solution candidate for SAT is represented as a bit string of length n, where

every variable is associated to one bit, according to the position defined by the cluster technique.

Fitness Function: the simplest fitness function used is the count of the clauses that are satisfied by the candidate solution. Another fitness function used is "Cluster fitness" where each cluster has a score of Number of satisfied clauses by its members current assignment divided on the cluster size.

Crossover operator: would happen between different clusters not between Individual, by successively replace certain cluster members in one parent with the counterpart cluster members of another Parent. Single point crossover was implemented using point position proportional to the cluster size. For example: if the single point was 0.25 and the cluster size was 20 gene then the single point of crossover would be gene position number 5 inside the cluster while if the cluster size was 100 gene, the crossover single point would be in gene position number 25 .

Mutation Operator: As the mutation rate is used to maintain the diversity of the entire population [30], we suggest use tow static mutation probabilities as the following:

1. Cluster Mutation Probability

The mutation rate is set independently for every cluster, proportionally to the Cluster Size. We experimentally notice that the clusters in the chromosome center, which indicate higher influence of their members values over the whole chromosome, have larger size than the clusters in the chromosome borders. So we use the following equation to calculate the mutation rate for each cluster:

Cluster Mutation Rate = 1 / (N – Cluster Size)

Where the N is the chromosome Length, No of variables, and Cluster Size is the number of genes belong to this cluster.

2. Gene Mutation Probability

A standard normal distribution variable would be used to choose the mutation bit position inside each cluster, that distribution would make the rate of mutation for the genes in the cluster center high (most related variable to others in cluster), and the genes positioned on the cluster border (less related variable) have a lower mutation rate.

Parent Selection: Parent selection process depend of choosing the best fitted solutions to generate the next generation. The tournament parent selection technique is used [11]. For each GA generation Parents, the individuals "solutions" would be divided into 2 groups, "highly fitted" group, that has the best fitted solutions of the current generation and "Less fitted" group that has the least the best fitted solutions of the current generation. In crossover Operator, one Parent would be chosen randomly from the "highly fitted" Individuals while the other would be chosen randomly from the "Less fitted" Individuals. Experimentally, that action slows the unmatured convergence of GA and decrees the possibility of getting similar solutions with each generation.

Replacement scheme: A small group of best ever solutions through all the Parents generations, namely elitism group, we be kept and updated through each new Parents generation. As well as, a small group of best "cluster fitness" solutions, namely minority group, their size depends on the dataset size. After the parent selection stage finished, the least fitted solutions of "highly fitted" parents group would be replaced by the elitism group. And the least fitted solutions of "Less fitted" parents group would be replaced by the minority group solutions. A check step is needed before the current parents are replaced with the elitism or minority groups' members. If current parent solutions have a similar solutions to these groups' members, then the similar solutions would be replaced not the least fitted solutions. This action was taken to keep best ever found solution for the SAT problem "elitism group" as it may be lost during crossover and mutation operators, plus keeping the individuals with best scored parts. As best fitted score of the solutions' parts does not necessarily enable the best whole fitted score solutions.

4 Experimental Study

4.1 Design of Experiments

All the Experiments were performed on several instances from the DIMACS Benchmark set [29]. The instances used are hard instances with the number of clauses (m) and variables (n) ratio (m/n) equals 4.26 its descriptions is given in Table 1. In order to obtain statistically significant results, several runs are required for each benchmark instance under consideration, for computation simplicity, 10 independent runs were executed for each instance with each technique, and the average value is used.

The experiments ran into an Intel(R) Core(TM) 2 Due CPU 3.00 with 2.00 GB of RAM, and implemented with C# programming language.

The techniques quality measure should express the efficiency of using each GA algorithm with the SAT instances, where both, the maximum number of clauses satisfied found by each technique, and the computational cost required, are taken into consideration.

For the first measure MCNR (Maximum clause number rate), which represents the portion of solution found at each run. It measures the percentage of the maximum number of satisfied clauses found to the total number of clauses in each

instance. While the second measure TR (Time required) quantifies the computational cost of each technique by measure the average time required to run the technique.

A classic GA was implemented where its internal representation is based on binary strings of length N in which the i^{th} bit represents the truth value of the i^{th} Boolean variable of the problem. A Classic genetic operators defined by Holland [25], are internally implemented, the recombination is done in the standard way using single point Crossover operator with middle gene position parameter, a static mutation probability (equals 1/n) was used. While the Parent selection operation was the tournament Algorithm.

While the exact method to solve the BMPG problem take O (n K) where k is the bandwidth searched for that graph. An Approximate algorithm that does not guarantee to find the optimal bandwidth of the graph but, get best bandwidth minimization with the given limitation is used [20]. The ERA algorithm is sensitive to its parameters, e.g. the temperature, the maximum number of accepted moves at each temperature, and the cooling rate. The parameters of the ERA algorithm, implemented in the following section, were chosen experimentally taking into account its authors experiments reported results in [21], and some related work reported in [31, 32].

TABLE 1. TEST INSTANCES

Seq	Instance name	n	m	Satisfiable?
1	unif-k3-r4.26-v360-c1533-S1217224084-10.SAT.shuffled.cnf	360	1533	Yes
2	unif-k3-r4.26-v360-c1533-S1373941726-12.UNSAT.shuffled.cnf	360	1533	No
3	unif-k3-r4.26-v500-c2130-S539876024-03.SAT.shuffled.cnf	500	2130	Yes
4	unif-k3-r4.26-v500-c2130-S1161940794-17.UNSAT.shuffled.cnf	500	2130	No
5	unif-k3-r4.261-v650-c2769-S1089058690-02.SAT.shuffled.cnf	650	2769	Yes
6	unif-k3-r4.261-v650-c2769-S440198403-10.UNSAT.shuffled.cnf	650	2769	No
7	unif-k3-r4.26-v800-c3408-S141590207-13.SAT.shuffled.cnf	800	3408	Yes
8	unif-k3-r4.26-v800-c3408-S1013535775-14.UNSAT.shuffled.cnf	800	3408	No
9	unif-k3-r4.26-v1000-c4260-S1141835011-09.SAT.shuffled.cnf	1000	4260	Yes
10	unif-k3-r4.26-v1000-c4260-S2083146463-03.UNSAT.shuffled.cnf	1000	4260	No

4.2 Epistatsis Reducer Effect Experimental Results and Analysis

In this section, experiment measures the efficiency of reducing GA encoding epistatsis using the proposed algorithm compared to bandwidth minimization problem for graph (BMPG) algorithm.
As the objective of the encoding algorithm is to re-label the most related variables of the problem and place them in closer positions inside the chromosome. Then we expect that the absolute difference between labels of different variable related into one clause in the SAT instances will be reduced. A new measure βA will be defined and used to quantify the SAT variables epistatsis generated due to their different positions with each technique.
The classic GA algorithm will measure the original SAT instance variable labeling difference, while both the ERA and cluster techniques will be ran on original SAT instance first, then βA will be measured using the new re-labeled SAT instance.
For each clause in the SAT Problem, the difference between the 3 variables label will be calculated. By subtract the absolute value of each variable label in the clause and the absolute value of neighbors' variables. Then sum is carried out for all the absolute clauses' values of the SAT instance, and divided by (3 * clauses number) to get the average βA. For Example, in the following clause (x1, -x8, x13), the difference between the 3 variables label will be equals |1-8| + |1-13| + |8-13|=24, and as we have the summation of 3 different variable labels' epistatsis, the average βA will be calculated as 24/3= 6 for this clause. By measure the average for all SAT instance clauses, with different encoding algorithms, we may get good indicator of the efficient performance of GA based on its variables epistatsis values.
Table 2 presents the comparison between the proposed algorithm in this research, namely CLU, and the ERA Algorithm. Data included are: Seq. of the formula; average variables epistatsis βA obtained with the original SAT problem and different encoding algorithms; the CPU time in seconds used by the encoding algorithms is Tβ .

TABLE 2 AVERAGE VARIABLES EPISTATSIS AND CPU TIME IN SECONDS MEASUREMENTS FOR ERA && CLU

Seq.	**Initial SAT βA**	**ERA βA**	**ERA Tβ**	**CLU βA**	**CLU Tβ.**

1	100	60	220	64	80
2	102	63	260	69	80
3	134	90	320	98	96
4	138	95	380	103	96
5	184	142	520	148	110
6	188	148	540	152	110
7	226	192	760	196	130
8	220	196	790	203	130
9	292	248	920	252	150
10	295	254	920	260	150

The results of experimentation presented showed that the ERA Algorithm outperforms the CLU algorithm in reducing the average encoding epistatsis, but with insignificant amount; while in computing time, CLU algorithm outperforms the ERA with tremendous amount especially when increasing the problem size, number of SAT variables.

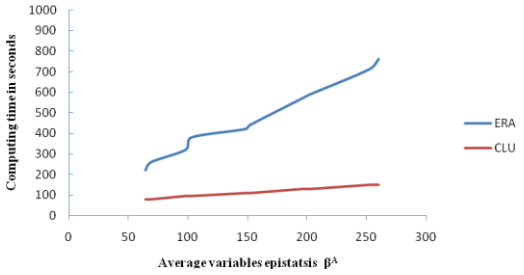

Fig 1: Average variables epiststsis versus computation time

Figure 1 showed results, the X- axis represents the Average variables epistatsis βA while the Y-axis represents the computing time in seconds required by the two algorithms.
Additionally, we notice that the unsatisfiable problems have higher average encoding epistatsis with the original problem, as well as, for both algorithms used to re-encode the problem.
We notice also that CLU algorithm running time does not affected; if the original problem is satisfiable, or not, while ERA affected negatively if the problem is unsatisfiable

4.3 Epistatsis Reducer Effect Experimental Results and Analysis

In this section, experiment measures the search efficiency of the proposed CLU+GA technique in solving SAT problem. The proposed algorithm will be tested compared to ERA+GA algorithm, as well as, the classic genetic algorithm.
Data included are: Seq. of the formula; the number of max satisfied clauses rate, for each approach, and the CPU time in seconds, for each approach. It is important to remark that the times presented in the next table for the combined approach in the column TR measure takes into account the CPU time used for the preprocessing step in case ERA+GA, or CLU+GA.

TABLE 3 AVERAGE MAX CLAUSES SATISFIED AND CPU TIME IN SECONDS MEASUREMENTS FOR ERA && CLU && GA

Seq.	GA βA	GA TS	ERA βA	ERA TS	CLU βA	CLU TS.
1	0.556	230	0.576	320	0.769	240
2	0.558	460	0.570	660	0.766	510
3	0.421	280	0.526	488	0.754	260
4	0.418	780	0.522	980	0.742	820
5	0.394	540	0.512	690	0.718	500
6	0.411	1220	0.509	1780	0.715	1280
7	0.381	840	0.506	960	0.646	620
8	0.397	2000	0.501	2440	0.645	2120
9	0.366	1020	0.502	1250	0.62	850
10	0.388	2560	0.503	3000	0.617	2630

As the unsatisfied SAT problems make the three algorithms reach their run-time end-condition limitation without find a suitable answer, the batch of unsatisfied instances reported separately than the satisfied instances.

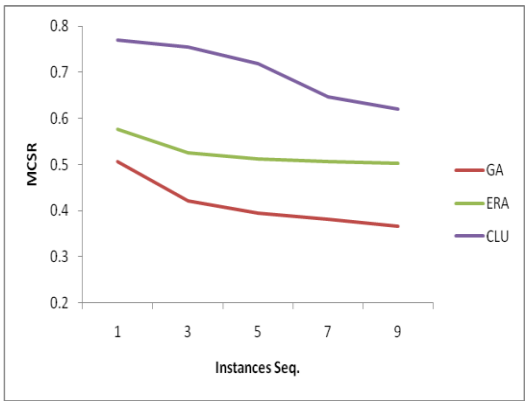

Fig. 2 Satisfied instance MCSR measure

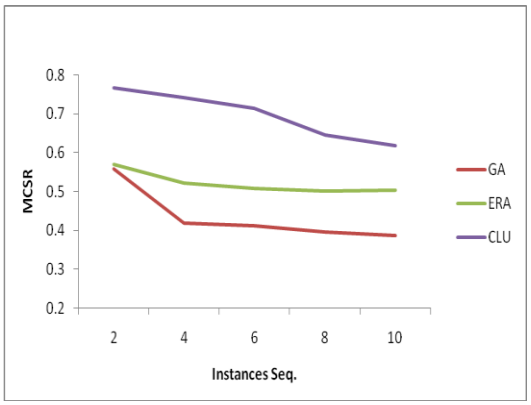

Figure 3 Unsatisfied instance MCSR measure

Figure 2, 3 represent the Max clause rate Satisfied by each technique. The X-axis represents the instance seq. used from table 1, while the Y-axis represents the portion of clauses satisfied by each algorithm, compare to the total clauses number. It is noticed that the natural of the problem does not affect the three algorithms performance, as similar size SAT problem, satisfied or not, almost have similar MCSR measurements. Satisfied Problems usually have higher MCSR. While The problem size, affect the three algorithms in negative way as the three algorithms have a descending performance, increase in computation time, with increasing the number of variables, while CLU has less affected with the problem size, the classic genetic algorithms has a serious impacted by the problem size which increase our intuitive that merge the CLU+GA with Local search will increase the efficiency of using GA as a heuristic algorithm to solve SAT problem.

Figure 4, 5 represent the computation time by each technique. The X-axis represents the instance seq. used from table 1, while the Y-axis represents the CPU time consumed by each algorithm. The natural of the problem, satisfied or not, affect differently the computation time consumed by each algorithm. While CLU+GA run faster than the classic GA in the satisfied instances, it take more time in case of the unsatisfied, we believe that the CLU+GA can solve the SAT problem faster than the GA, while in the unsatisfied instances it bear the computation time of the pre-processing step, cluster algorithm, plus running the GA with the tuned operators. The Preprocess ERA algorithm consume a lot of computation time especially when increase the SAT problem size, that make the ERA+GA the highest time consuming algorithm.

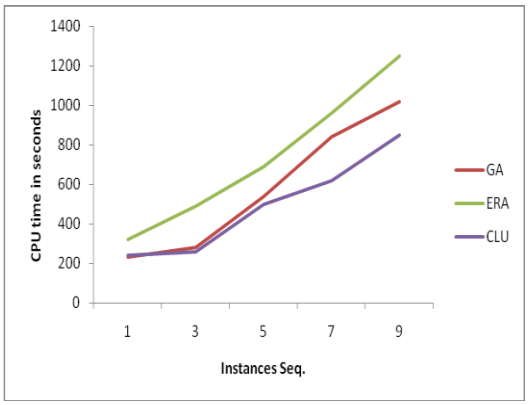

Figure 4 Satisfied instance CPU time measure

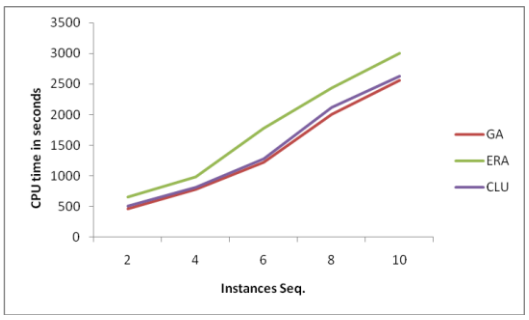

Figure 5 Unsatisfied instance CPU time measure

To conclude, the results of experimentation presented showed that the combined approach CLU+GA outperforms the ERA+GA algorithm, as well as, simple GA in all the tests accomplished, not only in solution quality but also in computing time.

5 CONCLUSIONS

This Paper is a part of an extended research, where the paper presents a first part of an incomplete SAT solver, that explore a hybrid approach which combining ideas from Evolution Algorithms and local search techniques.
In this paper, we investigate the possibility of increase the EAs search efficiency by encoding the problem and use this encoding technique to tune the EAs operators (parent selection, mutation, crossover, and replacement scheme). This approach has been compared versus a simple GA without preprocessing and versus ERA+GA techniques using a set of SAT instances. Also it was encourage to complete the research of hybrid the new GA techniques with other techniques, e.g. local search, to improve the Incomplete SAT Solver efficiency.

6 REFERENCES

[1] K.A. DeJong, W.M. Spears, Using genetic algorithm to solve NP-complete problems, in: Proceedings of the 3rd International Conference on Genetic Algorithms, Virginia, USA, 1989, pp. 124–132.

[2] F. Lardeux, F. Saubion, J.K. Hao, GASAT: A genetic local search algorithm for

the satisfiability problem, Evolutionary Computation 14 (2) (2006) 223–253.

[3] J. Gottlieb, E. Marchiori, C. Rossi, Evolutionary algorithms for the satisfiability problem, Evolutionary Computation 10 (1) (2002) 35–50.

[4] Levent Aksoy, E.O.Gunes, An Evolutionary Local Search Algorithm for the Satisfiability Problem, TAINN 2005, LNAI 3949, pp. 185 – 193, 2006.© Springer-Verlag Berlin Heidelberg 2006

[5] Oscar Pérez Cruz, Alfredo Cruz , Evolutionary SAT Solver, Ninth LACCEI Latin American and Caribbean Conference (LACCEI'2011), Engineering for a Smart Planet, Innovation Information Technology and Computational Tools for Sustainable Development, August 3-5, 2011, Medellín, Colombia.

[6] Alex.S.Fukunaga. Evolutionary Computation Journal, MIT Press Cambridge, MA, USA, Volume 16 Issue 1, Spring 2008

[7] Yousef Kilani: Improving GASAT by replacing tabu search by DLM and enhancing the best members. Artif. Intell. Rev. 33(1-2): 41-59 (2010)

[8] Yusef Kilani, Comparing the performance of the genetic and local search algorithms for solving the satisfiability problems, Soft Computing Volume 10, Issue 1, January 2010, Pages 198-207

[9] A. E. Eiben , Evolutionary Algorithms and Constraint Satisfaction: Definitions, Survey, Methodology, and Research Directions, Theoretical Aspects of Evolutionary Computing, 2001

[10] Omas Bäck, Agoston E. Eiben, Marco E. Vink , A superior Evolutionary Algorithm for 3 SAT, Proceedings of the 7th Annual Conference on Evolutionary Programming, number 1477 in +LNCS, 1998

[11] T. Blickle and L. Thiele. "A mathematical analysis of tournament selection". In Proceedings of the Sixth ICGA, pages 9—16. Morgan Kaufmann Publishers, San Francisco, Ca., 1995.

[12] Carla P. Gomes, Henry Kautz, Ashish Sabharwal, Bart Selman , Handbook of Knowledge Representation © 2008 Elsevier B.V., Chapter 2:Satisfiability Solvers, Pages 89-134.

[13] Henry Kautz, Ashish Sabharwal, and Bart Selman, Incomplete Algorithms, Handbook of Satisfiability, IOS Press, 2008, chapter 6

[14] J. Marques-Silva, Practical applications of Boolean Satisfiability, 9th International Workshop on In Discrete Event Systems, 2008. pp. 74-80.

[15] John Franco and John Martin, A History of Satisfiability, Handbook of Satisfiability, IOS Press, 2009, Chapter 1.

[16] John Franco and John Martin, Handbook of Satisfiability, IOS Press, 2009

[17] M S Hasan , B P Amavasai1 and J R Travis1, A Distributed Genetic Algorithm Solution to the Boolean Satisfiability Problem, International Symposium on Innovations in Intelligent Systems and Applications, INISTA 2005

[18] Eiben, A.E.; van der Hauw, J.K. Solving 3-SAT by GAs adapting constraint weights Evolutionary Computation. IEEE International Conference on 1997

[19] Rodriguez-Tello, E., Torres-Jimenez, J., SAT solving using an epistasis reducer algorithm plus a GA. Computational Intelligence and Multimedia Applications Fifth International Conference, 2003.

[20] Eduardo Rodriguez-tello , Jose Torres-jimenez. ERA: An algorithm for reducing the epistasis of sat problems. In Genetic and Evolutionary Computation - GECCO 2003

[21] Eduardo Rodriguez-tello , Jose Torres-jimenez. Improving the Performance of a Genetic Algorithm Using a Variable-Reordering Algorithm. Genetic and Evolutionary Computation – GECCO 2004. Volume 3103/2004, 102-113, DOI: 10.1007/978-3-540-24855-2_10.

[22] Cyril fonlupt, Denis Robilliard, Philippe Preux. A Bit-Wise Epistasis Measure for Binary Search Spaces. PPSN V Proceedings of the 5th International Conference on Parallel Problem Solving from Nature. Springer-Verlag London, UK ,1998.

[23]

[24] Davidor Y, Epistasis Variance: A Viewpoint of GA-Hardness. In foundations of Genetic Algorithms. Morgan Kaufmann, page 23-35, 1991

[25] Davidor Y, Epistasis Variance: Suitability of a Representation to Genetic Algorithms. Complex system publication Inc, edition 4 , 1990

[26] J.H. Hollend , Adaptation in Natural and Artificial systems, University of Michigen Press , Ann Arbor , 1975

[27] L Kaufman, P J Rousseeuw „Finding Groups in Data: An Introduction to Cluster Analysis . New York, John Wiley & Sons.1990.

[28] Hae-Sang Park, Jong-Seok Lee and Chi-Hyuck Jun. A simple and fast algorithm for K-medoids clustering. Expert Systems with Applications: An International Journal archive, Volume 36 Issue 2, March, 2009

[29] http://www.satlib.org/benchm.html

[30] ftp://dimacs.rutgers.edu/pub/challenge/satisfiability/contributed/UCSC/instances

[31] Korejo, I., Yang, S. and Li, C., A comparative study of adaptive mutation operators for metaheuristics, 8th Metaheuristic International Conference (MIC 2009).

[32] J. Torres-Jimenez and E. Rodriguez-Tello, "A New Measure for the Bandwidth Minimization Problem", Proceedings of the IBERAMIA-SBIA 2000, Number 1952 in LNAI (Antibaia SP, Brazil), Springer-Verlag, November 2000, pp. 477—486.

[33] W. M. Spears, "Simulated Annealing for Hard Satisfiability Problems", Tech. Report AIC-93- 015, AI Center, Naval Research Laboratory, Washington, DC 20375, 1993.