COMBINING AN EVOLUTIONARY ALGORITHM WITH THE MULTILEVEL PARADIGM FOR THE SIMULATION OF COMPLEX SYSTEMS

Noureddine Bouhmala, Karina Hjelmervik, Kjell Ivar Øvergård
Department of Maritime Technology and Innovation
Vestfold University College
Norway
Email: {noureddine.bouhmala,karina.hjelmervik,koe}@hive.no

KEYWORDS

Maximum satisfiability problem, Multilevel paradigm.

ABSTRACT

Evolutionary Algorithms have become an efficient tool to simulate large and complex systems that require a huge amount of computational resources. Nevertheless, evolutionary algorithms may still suffer from either slow or premature convergence preventing the search to visit more promising areas, and thus leading to solutions of poor quality. In this work, the multilevel paradigm is used in order to enhance the evolutionary algorithm's performance for simulating large industrial instances. The multilevel paradigm refers to the process of dividing large and difficult problems into smaller ones, which are hopefully much easier to solve, and then work backward towards the solution of the original problem, using a solution from a previous level as a starting solution at the next level. Experimental results comparing the multilevel evolutionary algorithm against its single-level variant are presented.

INTRODUCTION

The MAX-SAT problem still deserves much research attention from a wider community of researchers due to its theoretical importance. MAX-SAT is a widely used modeling framework for simulating complex systems that turn out to be of combinatorial nature which can be conveniently formulated to SAT or MAX-SAT in an elegant way (Hoos 2004). Examples include model-checking (Biere et al. 1990) of finite state systems, design debugging (Smith et al. 2005), AI planning (Rintanen et al. 2006) to name just a few. Large and complex systems are hard to solve and require a huge amount of computational resources. Optimization search techniques tend to spend most of the time exploring a restricted area of the search space preventing the search to visit more promising areas, and thus leading to solutions of poor quality. A better approach would be to reduce these complex systems into simplified models so that the optimization algorithms used during the simulation process would be far more efficient. In this paper a multilevel evolutionary algorithm is introduced for MAX-SAT. The key feature of this algorithm involves looking at the simulation as a process that transforms the original problem into a hierarchy of increasingly smaller problems that are much easier to solve. The optimization starts at the coarsest level and works backwards to the finest level where the solution obtained at a child level is fed as an input solution to the parent level.

THE MAXIMUM SATISFIABILITY PROBLEM

Generally, a SAT problem is defined as follows. A propositional formula $\Phi = \bigwedge_{j=1}^{m} C_j$ with m clauses and n Boolean variables is given. Each Boolean variable, $x_i, i \in \{1, \ldots, n\}$, takes one of the two values, True or False. A clause, in turn, is a disjunction of literals and a literal is a variable or its negation. Each clause C_j has the form:

$$C_j = \left(\bigvee_{k \in I_j} x_k \right) \vee \left(\bigvee_{l \in \bar{I}_j} \bar{x}_l \right),$$

where $I_j, \bar{I}_j \subseteq \{1, ...n\}$, $I \cap \bar{I}_j = \emptyset$, and $\bar{x}_i$ denotes the negation of x_i. The task is to determine whether there exists an assignment of values to the variables under which Φ evaluates to True. Such an assignment, if it exists, is called a satisfying assignment for Φ, and Φ is called satisfiable. Otherwise, Φ is said to be unsatisfiable. Most SAT solvers use a Conjunctive Normal Form (CNF) representation of the formula Φ. In CNF, the formula is represented as a conjunction of clauses, with each clause being a disjunction of literals. The maximum satisfiability problem is the optimization variant of SAT. The goal is to minimize the number of unsatisfied clauses. The focus in this work is

Proceedings 27th European Conference on Modelling and Simulation ©ECMS Webjørn Rekdalsbakken, Robin T. Bye, Houxiang Zhang (Editors)
ISBN: 978-0-9564944-6-7 / ISBN: 978-0-9564944-7-4 (CD)

restricted to formulas in which all the weights are equal to 1 (i.e.unweighted MAX-SAT).

THE MULTILEVEL EVOLUTIONARY ALGORITHM (MLVMA)

The proposed multilevel evolutionary goes through four different phases following the general multilevel pattern described in (Bouhmala and Granmo 2011)(Karypis and Kumar 1998)(Walshaw 2001). The algorithm starts creating a hierarchy of increasingly smaller and coarser versions of the original problem (Phase 1). The coarsening phase works by grouping the literals representing the problem into clusters. The coarsening is computed using a randomized algorithm. The literals are visited in a random order. If a literal l_i has not been matched yet, then we randomly select one randomly unmatched literal l_j, and a new literal l_k (a cluster) consisting of the two literals l_i and l_j is created. Unmerged literals are simply copied to the next level. The new formed literals are used to define a new and smaller problem and recursively iterate the coarsening process until the size of the problem reaches some desired threshold. Initialization is then trivial and consists of generating an initial solution for the population located at the coarsest level using a random procedure. The clusters of every individual in the population are assigned the value of true or false in a random manner (Phase 2). The population at the coarsest level P_{m+1} is subject to an improvement phase using an evolutionary memetic algorithm (MA)(Norman and Moscato 1991) (Phase 3). Individuals are combined using the two-points cross-over. The task of the cross-over operator is to reach regions of the search space with better average quality. Combining pairs of individuals can be viewed as a matching process. The individuals are visited in random order. An unmatched individual i_k is matched randomly with an unmatched individual i_l. Thereafter, the two-point crossover operator is applied using a cross-over probability to each matched pair of individuals. The two-point crossover selects two randomly points within a chromosome and then interchanges the two parent chromosomes between these points to generate two new offsprings. The work conducted in (Spears 1995) shows that the two-point crossover is more effective when the problem at hand is difficult to solve. The new individuals resulted from the two-point cross-over are then enhanced with a local search and evaluated. This simple local search seeks for the new variable-value assignment which best decreases the numbers of unsatisfied clauses is identified. Even though the population at P_{m+1} is at a local minimum, the projected population may not be at a local optimum with respect to P_m. The projected population is already a good solution and contains individuals with better fitness value making the evolutionary algorithm to converge quicker within a few generation to a better assignment. During each level, the evolutionary algorithm is assumed to reach convergence when no further improvement of the fittest individual has not been made during 10 consecutive generations. The uncoarsening phase (Phase 4) refers to the inverse process followed during the coarsening phase. Having optimized the assignment on P_{m+1}, the assignment must be extended on its parent level P_m. The extension algorithm is simple; if a cluster $c_i \in P_{m+1}$ is assigned the value of true then the merged pair of clusters that it represents, c_j, $c_k \in P_m$ are also assigned the true value.

EXPERIMENTAL RESULTS

Test Suite & Parameter Settings

The tests were carried out on Unix server with 6 cpu cores, 2.80GHz and 50GB of memory. The code was written in C and compiled with the GNU C compiler version 4.6. All parameters were fixed experimentally using industrial instances that arise in the simulation of faults in model checking and large graph coloring problems from the SAT library benchmark.

instances	VAR	CLS
case1:bmc-ibm-1.cnf	9 658	55 870
case2:bmc-ibm-2.cnf	3 628	14 468
case3:bmc-ibm-3.cnf	14 930	72 106
case4:bmc-ibm-5.cnf	9 396	41 207
case5:bmc-ibm-7.cnf	8 710	39 774
case6:bmc-galileo-8.cnf	58 074	294 821
case7:bmc-galileo-9.cnf	63 624	326 999
case8:bmc-ibm-10.cnf	61 088	334 861
case9:bmc-ibm-11.cnf	32 109	150 027
case10:bmc-ibm-12.cnf	39 598	194 778
case11:bmc-ibm-13.cnf	13 215	65 278
case12:g125.18.cnf	2 250	70 163
case13:g125.17.cnf	2 125	66 272
case14:g250.15.cnf	3 750	233 965
case15:g250.29.cnf	7 250	454 622

TABLE I: Benchmark Instances

The set used in the experiments is taken from [1] and shown at Table I. The second and third column denote the number of variables and clauses respectively. IBM SPSS Statistics version 19 was used for statistical analysis. Due to the randomization nature of the algorithms, each problem instance was run 100 times with a cut–off parameter (max–time) set to 15 minutes. The 100 runs where chosen because pilot runs had shown the size of the difference to be so large that 100 runs where enough for an acceptable statistical power ($power > .95$), this is in accordance with the suggestions given in a recent report on statistical testing of randomized algorithms (Arcuri and Briand 2011) We also performed single runs with MLVMA and MA with a cut–off parameter set to 60 minutes. The intention was to see whether the solutions created by MLVMA and MA converged to identical solutions.

The selected parameters are listed below:

- Crossover probability = 0.85
- Mutation probability = 0.1
- Population size = 50

[1]SATLIB website (http://www.informatik.tu-darmstadt.de/AI/SATLIB)

#Case	Mean (SD)	Range
1	3872.8 (233.7)	3632-4830
2	157.8(8.1)	137-185
3	3320.9 (279.1)	2911-4187
4	1125.8(138.1)	976-1675
5	1475.9 (101.3)	1167-1668
6	12232.9(2464)	9809-23178
7	14412.6(2877.1)	11318-22705
8	2206.1(1117.6)	20621-25275
9	13730.9 (547.5)	13045-15413
10	17477.0 (410.7)	17079-18908
11	3875.7(273.4)	3529-5055
12	37.8 (3,5)	29-48
13	40,7 (3.5)	30-50
14	193.7 (30.9)	173-425
15	214.8(64.3)	177-637

TABLE II: MLVMA mean, standard deviation and range of unsolved clauses.

#Case	Mean (SD)	Range
1	5700.5 (427.6)	5184-7773
2	234.6(16.1)	190-273
3	10049.3 (356.8)	9547-11559
4	3282.5 (276.0)	3024-4651
5	3107.1(152.3)	2870-4044
6	51062.3 (814.1)	49375-53517
7	57301.2(877.1)	55292-59723
8	63720.2 (475.3)	62575-64626
9	26207.8(353.7)	25461-27311
10	35289.7(440.3)	34363-36871
11	8233.4(384.2)	7886-10080
12	323.9 (47.4)	225-481
13	154.2 (20.7)	92-222
14	29565.3 (1068.7)	27587-33040
15	85410.6(1759.2)	80704-89626

TABLE III: MA: mean, standard deviation and range of unsolved clauses.

	BIS: Test^{a,b}	
#Case	MD [99%CI]	Cohen's d
1	1895.2[1827.6, 1962.8]	5,3
2	76.8[71.9,81,3]	6,02
3	6728.4[6610.1,6844.2]	21,01
4	2156.7[2082.4,2238.7]	9,88
5	1631.2[1585.7,1682.1]	12,61
6	38829.4[38119.8, 39450.1]	21,16
7	42888,6[42100.6,43645.6]	20.17
8	41713,8[41396.1, 42020.2]	48,57
9	12476.9[12314.7, 12614.4]	27,07
10	17812,7[17656.6, 17963.]	41,83
11	4446,7[4329.1, 4571.1]	13,33
12	286,1[274.0. 298.4]	8,51
13	113,6[108.3,119.0]	7,66
14	29371.6[29097.1, 29650.6]	38,85
15	85195.8[84753.5, 85638.3	68,44

TABLE IV: MLVMA Vs MA: mean difference and the 99% confidence interval.

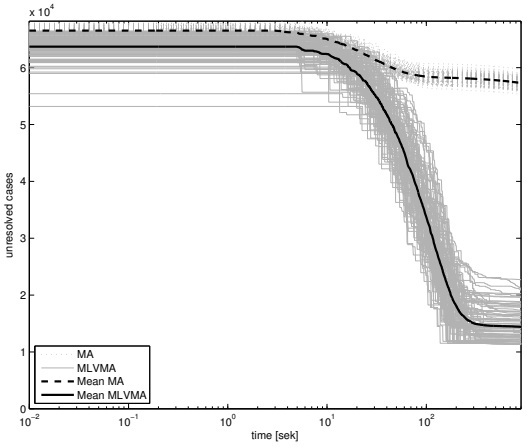

Fig. 1. Log-Log plot: MLVMA Vs MA: bmc-ibm-9.cnf: $|V|$ = 63624 , $|C|$= 326999 - Time development for 100 runs in 15 minutes

• Stopping criteria for the reduction phase: The reduction process stops as soon as the size of the coarsest problem reaches 100 variables (clusters). At this level, MA generates an initial population.
• Convergence during the refinement phase: If there is no observable improvement of the fitness function of the best individual during 10 consecutive generations, MA is assumed to have reached convergence and moves to a higher level.
• Time constraint for each run is 900 seconds and one single run 60 minutes.

Analysis of Results

The comparison of the MA and the MLVMA algorithms for each instance is shown in Tables II-IV. Tables II-III show the mean, standard deviation and the range of unsolved clauses for MLVMA and MA for each instance after a 900 seconds run time. Table IV shows the mean difference and the 99 percent confidence interval for the difference between MA and MLVMA. Bootstrapping was chosen due to the lack of knowledge regarding the statistical distribution of the underlying population. The domination of MLVMA versus MA is strengthened by the fact that none of the confidence intervals for the mean difference between MLVMA and MA contains zero (0)[2]. The standardized effect size measure Cohen's d (Cohen 1988) is very high [3] and in-

[2]we used mean-based statistics to evaluate the difference between the two algorithms because there was no overlap between the compared data sets. This is not in accordance with recent recommendations (Arcuri and Briand 2011) but was found to be necessary because non-parametric rank-based tests of significance such as Mann-Whitney U-test would not entail extra information as non-overlapping data sets would give identical solutions for different instances

[3]The mean-based effects size measure Cohen's delta (Cohen 1988) was used because we used mean-based t-tests to evaluate the difference between the data sets. We also included Cohen's delta because the common language effect size measure $\hat{A}_{12}$ (Vargha and Delaney 2000) would entail no extra information as

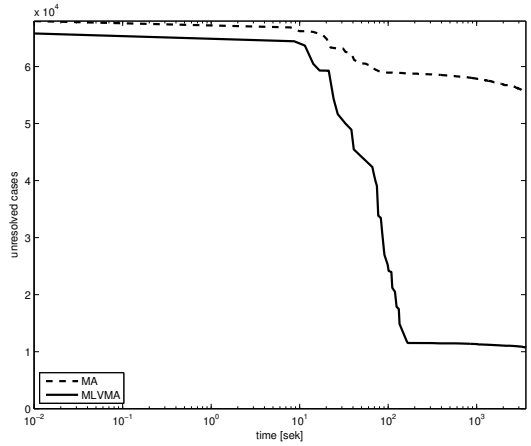

Fig. 2. Log-Log plot: MLVMA Vs MA: bmc-ibm-9.cnf: $|V|$ = 63624 , $|C|$= 326999 - Time development for one run in 60 minutes

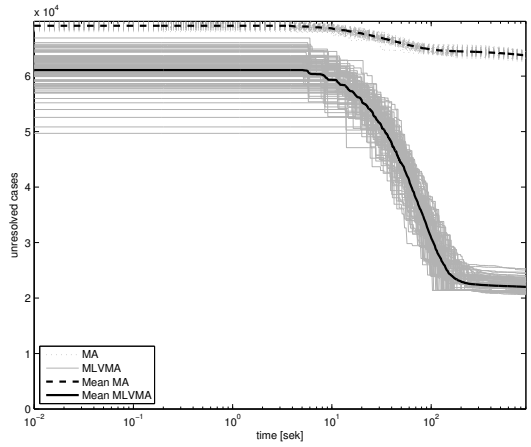

Fig. 3. Log-Log plot: MLVMA Vs MA: bmc-ibm-10.cnf: $|V|$ = 61088 , $|C|$= 334861 - Time development for 100 runs in 15 minutes

dicates that the effect of adding the multilevel paradigm to the memetic algorithm leads to a great improvement of the solutions after a 900 seconds run time.

The plots in Figures 1–4 show the general tendency of the solution quality reached as a function of time for two instances. Single runs with a cut-off parameter of (1 hour) for both algorithms was performed to see whether any cross-over occurred for the two algorithms. Figures 2 and 4 show the 100 runs of MLVMA and MA with a cut-off at 15 min as well as the mean of these runs, while Figures 1and 3 show single runs of the MLVMA and MA with a cut-off parameter of 60 minutes. The plots suggest that problem solving with MLVMA happens in two phases. In the first phase which corresponds to the early part of the search, MLVMA behaves as a hill-climbing method. The best assignment improves rapidly at first, and then flattens off as we mount the plateau, marking the start of the second phase.

The plateau spans a region in the search space where

one model would always be better, e.g. $P_{(MLVMA>MA)} = 1$ for all instances

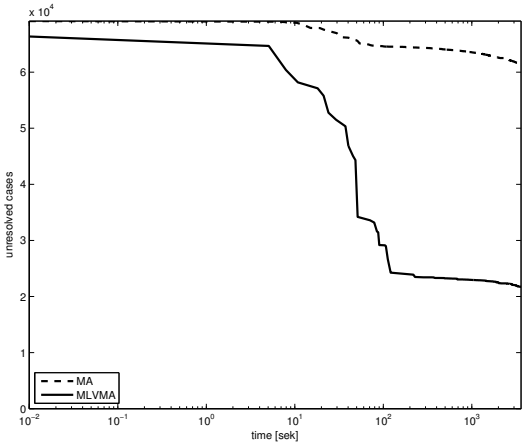

Fig. 4. Log-Log plot: MLVMA Vs MA: bmc-ibm-10.cnf: $|V|$ = 61088 , $|C|$= 334861 - Time development for one run in 60 minutes

flips typically leave the best assignment unchanged, and occurs more specifically once the refinement reaches the finest level. It is clear from the plots that the multilevel variant offers a clear advantage over its single version. We can see from the results that MLVMA have a better asymptotic convergence (to around 89.73%- 99.96% in excess of the optimal solution) as compared to MA which only reach around (80.28%- 99.86%). The instances where both algorithms reach approximately the same solution quality (with MLVMA being marginally better). MLVMA offers a cost effective solution strategy considering the amount of time required.

In our view, the efficiency of MLVMA relies on coupling the refinement process across different levels. This paradigm offers two main advantages which enables MA to become much more powerful in the multilevel context. During the refinement phase MA applies a local transformation (i.e. a move) within the neighborhood (i.e. the set of solutions that can be reached from the current one) of the current solution to generate a new one. The coarsening process offers a better mechanism for performing diversification (i.e. the ability to visit many and different regions of the search space) and intensification (i.e. the ability to obtain high quality solutions within those regions). By allowing MA to view a cluster of variables as a single entity, the search becomes guided and restricted to only those configurations in the solution space in which the variables grouped within a cluster are assigned the same value. As the size of the clusters varies from one level to another, the size of the neighborhood becomes adaptive and allows the possibility of exploring different regions in the search space while intensifying the search by exploiting the solutions from previous levels in order to reach better solutions.

CONCLUSION

A new apperoach has been outlined for solving the satisfiability problem based on a combining a memetic algorithm with the multilevel paradigm. A set of large benchmark instances were used to get a comprehen-

sive picture of the performance of the new approach. The multilevel paradigm greatly improves the MA and always returns a better solution for the equivalent runtime. The tests where MLVMA and MA offer similar asymptotic convergence, MLVMA being slightly better and considerably faster. The broad conclusions that we draw from the presented results is that the multilevel paradigm can enhance the convergence behavior of the memetic algorithm. Our future work aims at investigating other coarsening schemes together with various cross-over operators for the the possible enhancement of the multilevel memetic algorithm.

References

[1] Arcuri, A. and L. Briand. 2011. "A Hitchhiker's Guide to Statistical Tests for Assessing Randomized Algorithms in Software Engineering." Technical report, simula research laboratory, number 13.

[2] Biere, A.; A. Cimatti; E. Clarke, and Y. Zhu. 1999. "Symbolic model cheking without BDDs ". In Tools and Algorithms for the Construction and Analysis of Systems, 193-207.

[3] Bouhmala, N. and O.C. Granmo. 2011. "GSAT Enhanced with Learning Automata and Multilevel Paradigm." International Journal of Computer Science Issues, Vol. 8, Issue 3, 38-54.

[4] Cohen, J. 1988. "Statistical Power Analysis for the Behavioral Sciences." 2nd ed. Lawrence Erlbaum, (1988).

[5] Hoos,T and T. *Stützle*. 2004. "Stochastic Local Search: Foundations and Applications." Morgan Kaufmann.

[6] Karypis, G. and V. Kumar. 1998. "A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs.". SIAM J. Sci. Comput., 20(1), 351-392.

[7] Norman, M.G. and P. Moscato. 1989."A competitive and cooperative approach to complex combinatorial search ". Technical Report Caltech Concurrent Com- putation Program, Report. 790, California Institute of Technology, Pasadena, California, USA. expanded version published at the Proceedings of the 20th Informatics and Operations Research Meeting, Buenos Aires (20th JAIIO).

[8] Rintanen, J.; K. Heljanko, and I.Niemelä. 2006. "Planning as Satisfiability: Paralel plans and algorithms for plan search ". Artificial Intelligence, vol. 170, no, 12-13, 1031-1080.

[9] Smith,A.; A.G. Veneris; M.F.Ali, and A. Viglas. 2005."Fault diagnosis and logic debugging using Boolean satisfiability". IEEE Transactions on Computer-Aided Design, Vol. 24, no.10, 1606-1621.

[10] Vargha, A. and H.D. Delaney. 2000."A critique and improvement of the CL Common Language Effect Size statistics of McGraw and Wong". Journal of Educational and Behavioral Statistics, 25(2), 101-132.

[11] Spears, W. 1995."Adapting Crossover in Evolutionary Algorithms". Proc of the Fourth Annual Conference on Evolutionary Programming, MIT Press, 367-384.

[12] Walshaw, C. 2004."Multilevel Refinement for Combinatorial Optimization Problems". Annals of Operations Research 131, 325-372.