We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

Chapter 11

A Multilevel Evolutionary Algorithm Applied to the Maximum Satisfiability Problems

Noureddine Bouhmala, Kjell Ivar Øvergård and Karina Hjelmervik

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.72843

Abstract

The maximum satisfiability problem that is known to be nondeterministic polynomial (NP) complete plays a central role problem in many applications in the fields of very large-scale integration (VLSI) computer-aided design, computing theory, artificial intelligence, and defense. Given a set of m clauses and n Boolean variables, the maximum satisfiability problem refers to the task of finding an assignment of values to the variables that maximizes the number of satisfied clauses (or minimizes the number of unsatisfied clauses) In this chapter, a multilevel evolutionary algorithm is proposed for the maximum satisfiability problem. The multilevel process works by grouping the variables defining the problem to form clusters, uses the clusters to define a new problem, and is repeated until the problem size falls below some threshold. The coarsest problem is then given an initial assignment of values to variables and the assignment is successively refined on all the problems starting with the coarsest and ending with the original.

Keywords: maximum satisfiability problem, genetic algorithm, multilevel paradigm, discrete optimization, effect size

1. Introduction

Combinatorial optimization is a lively field of applied mathematics, combining techniques from combinatorics, linear programming, and the theory of algorithms, to solve optimization problems over discrete structures. Utilizing classical methods of operations research often fails due to the exponentially growing computational effort. It is commonly accepted that these methods might be heavily penalized by the nondeterministic polynomial (NP)-hard nature of the problems and consequently will then be unable to solve large-size instances of a problem.

Therefore, in practice meta-heuristics are commonly used even if they are unable to guarantee an optimal solution. The driving force behind the high performance of meta-heuristics is their ability to find an appropriate balance between intensively exploiting areas with high-quality solutions (the neighborhood of elite solutions) and moving to unexplored areas when necessary. The evolution of meta-heuristics has taken an explosive upturn. The recent trends in computational optimization move away from the traditional methods to contemporary nature-inspired meta-heuristic algorithms though traditional methods can still be an important part of the solution techniques for small-size problems. As many real-world optimization problems become increasingly complex and hard to solve, better optimization algorithms are always needed. Nature-inspired algorithms such as genetic algorithms (GAs) are regarded as highly successful methods when applied to a broad range of discrete as well as continuous optimization problems. This chapter introduces the multilevel paradigm combined with genetic algorithm for solving the maximum satisfiability problem. Over the past few years, an increasing interest has arisen in solving hard optimization problems using genetic algorithms. These techniques offer the advantage of being flexible. They can be applied to any problem (discrete or continuous) whenever there is a possibility for encoding a candidate solution to the problem, and a mean of computing the quality of any candidate solution through the so-called objective function. Nevertheless, GAs may still suffer from premature convergence. The performance of GAs deteriorates very rapidly mostly due to two reasons. First, the complexity of the problem usually increases with its size, and second, the solution space of the problem increases exponentially with the problem size. Because of these two issues, 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. Designing efficient optimization search techniques requires a tactical interplay between diversification and intensification [1, 2]. The former refers to the ability to explore many different regions of the search space, whereas the latter refers to the ability to obtain high-quality solutions within those regions.

In this chapter, a genetic algorithm is used in a multilevel context as a means to improve its performance. This chapter is organized as follows. Section 2 describes the maximum satisfiability problem. Section 3 explains the hierarchical evolutionary algorithm. In Section 4, we report the experimental results. Finally, Section 5 discusses the main conclusions and provides some guidelines for future work.

2. The maximum satisfiability problem

Given a set of n Boolean variables and a conjunctive normal form (CNF) of a set of m disjunctive clauses of literals, where each literal is a variable or its negation which takes one of the two values ***True*** or ***False***, the task is to determine whether there exists an assignment of truth values of the variables that satisfy the maximum number k of clauses. Multilevel approaches are special techniques which aim at producing smaller and smaller problems that are easier to solve than the original one. These techniques were applied to different combinatorial optimization problems. Examples include graph-partitioning problem [3–7], the traveling salesman problem [8, 9],

graph coloring and graph drawing [10, 11], feature selection problem in biomedical data [12], and maximum satisfiability problem [13–16]. A recent survey over multilevel techniques can be found in [1, 17, 18].

3. The multilevel evolutionary algorithm

3.1. Main idea

The multilevel paradigm works by merging the variables defining the problem to form clusters, uses the clusters to define a new problem, and the process is repeated until the problem size reaches some threshold. A random initial assignment is injected to the coarsest problem and the assignment is successively refined on all the problems starting with the coarsest and ending with the original. The multilevel evolutionary algorithm is described in Algorithm 1.

Algorithm 1. The multilevel evolutionary algorithm

input : Problem P_0

output: Solution $S_{final}(P_0)$

1 begin

2 level := 0 ;

3 **while** *Not reached the desired number of levels* **do**

4 $P_{level+1}$:=Reduce (P_{level}) ;

5 level := level + 1 ;

6 /* Proceed with Memetic algorithm */ ;

7 $S_{start}(P_{level})$ = Initial-Assignment (P_{level}) ;

8 $S_{final}(P_{level})$ = Refinement (P_{level}) ;

9 **while** $(level > 0)$ **do**

10 $S_{start}(P_{level-1})$:=Project $\left(S_{final}(P_{level})\right)$;

11 $S_{final}(P_{level-1})$:=Refinement ($S_{start}(P_{level-1})$) ;

12 level := level – 1

13 end

3.2. Reduction phase

This process (lines 3–5 of Algorithm 1) is graphically illustrated in **Figure 1** using an example with 10 variables. The coarsening phase uses two levels to coarsen the problem down to three

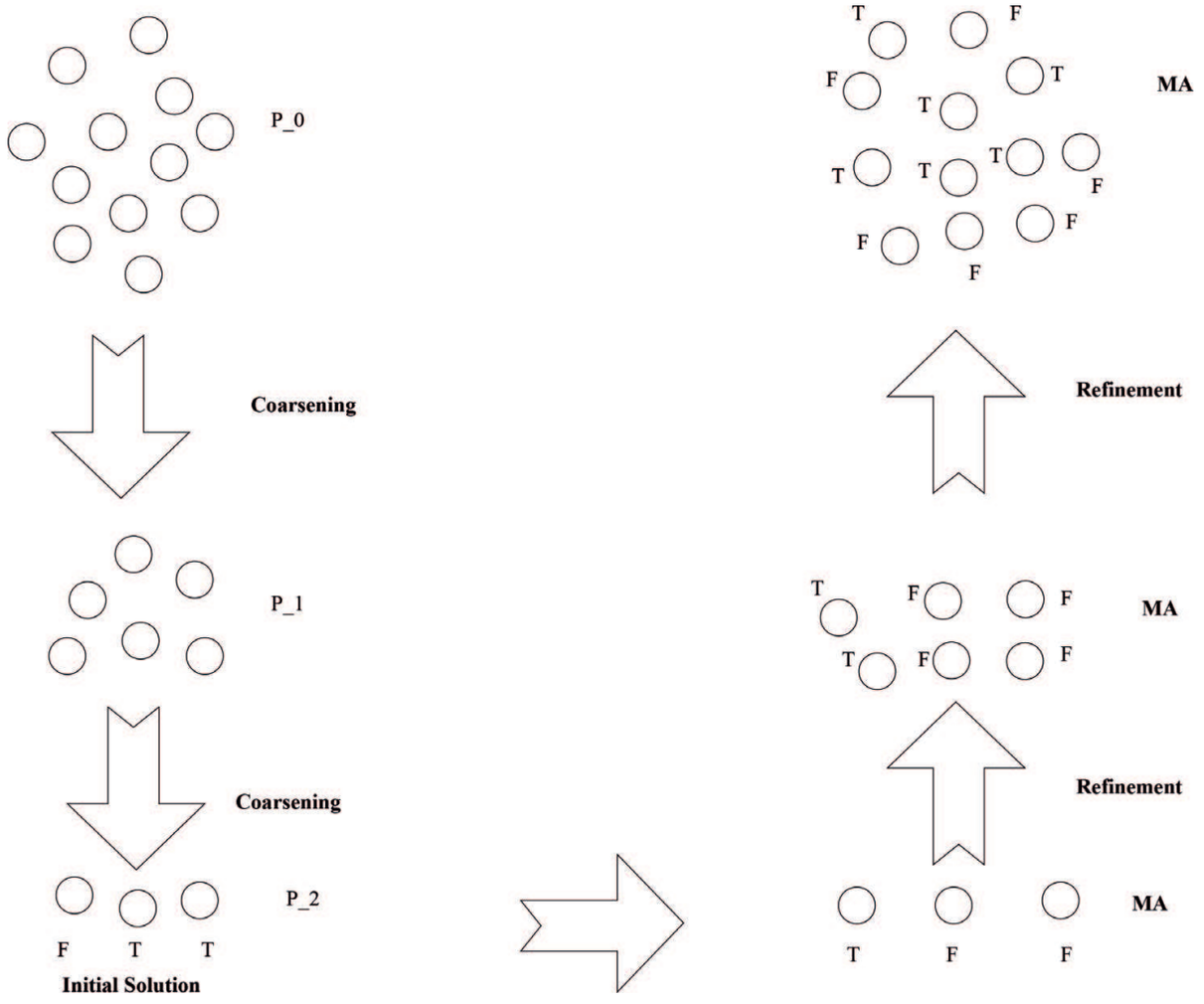

Figure 1. The various phases of the multilevel evolutionary algorithm.

clusters. P_0 corresponds to the original problem. The random-coarsening procedure is used to randomly merge the literals in pairs leading to a coarser problem (level) with five clusters. This process is repeated leading to the coarsest problem (P_3) with three clusters. An initial population is generated where the clusters are randomly assigned the value of true or false. The figure shows an initial solution where one cluster is assigned the value of true and the remaining two clusters are assigned the value false. Thereafter, the computed initial solution is then improved with the evolutionary algorithm referred to as MA. As soon as the convergence criteria are reached at P_2, the uncoarsening phase takes the whole population from that level and then extends it so that it serves as an initial population for the parent level P_1 and then proceeds with a new round of MA. This iteration process ends when MA reaches the stop criteria that is met at P_0.

3.3. Initial solution

The coarsening phase stops when the problem size reaches a threshold. A random procedure is used to generate an initial solution at the coarsest level. The clusters of every individual in the population are assigned the value of true or false in a random manner (line 7 of Algorithm 1).

3.4. Projection and refinement phases

The projection phase is the opposite process followed during the coarsening phase. The assignment reached at $level_{m+1}$ is now to be extended on is parent $level_m$. The extension algorithm is simple; if a cluster which belongs to $level_{m+1}$ is assigned the value of true, then the grouped pair of clusters that it represents, which belong to $level_m$, are also assigned the true value (line 10 of Algorithm 1). The evolutionary algorithm explained in the next section is used to improve the assignment during each level. The population reached at $level_{m+1}$ will serve as the initial population for $level_m$. The projected population already contains individuals with high fitness value leading MA to converge quicker within a few generations to a better assignment (lines 8 and 11 of Algorithm 1).

3.5. Evolutionary algorithm (MA)

The evolutionary algorithm proposed in this chapter and described in Algorithm 2 combines a genetic algorithms and local search. The algorithm maintains a population of solutions for the problem at hand (i.e., a pool having several solutions simultaneously). Each of these solutions is called an individual. Each generation consists of updating a population of individuals, hopefully leading to better solutions. The individuals from the set of solutions, which is called population, will evolve from generation to generation by repeated application of genetic operators and a local search scheme. Over many generations, the population becomes uniform and converges to optimal or near-optimal solutions.

Algorithm 2. Evolutionary algorithm

---

begin

Generate initial population ;

Evaluate the fitness of each individual in the population ;

while *(Not Convergence reached)* **do**

Select individuals according to a scheme to reproduce ;

Breed if necessary each selected pairs of individuals through crossover;

Apply mutation if necessary to each offspring ;

Apply local search to each chromosome ;

Evaluate the fitness of the intermediate population ;

Replace the parent population with a new generation

end

- **Fitness function:** it is a numerical value that expresses the performance of an individual (solution) so that different individuals can be compared. The fitness function is defined as the number of unsatisfied clauses.

- **Initial population:** the initial population consists of individuals generated randomly in which each gene's allele is assigned randomly the value 0 (false) or 1 (true).
- **Crossover:** new solutions are produced by matching pairs of individuals in the population and then applying a crossover operator to each chosen pair. An unmatched individual i_k is matched randomly with an unmatched individual i_l. Thereafter, the two-point crossover operator is applied using a crossover probability to each matched pair of individuals. The two-point crossover draws two random points within a chromosome and then interchanges the two parent chromosomes between these points to produce two new offspring. The work presented in [19] shows that the results produced by the two-point crossover are excellent especially when the problem is hard to solve.
- **Mutation:** let $C = c_1, c_2, \ldots, c_m$ be a chromosome represented by a binary chain where each of whose gene c_i is either 0 or 1. Each gene c_i is mutated through flipping this gene's allele from 0 to 1 or from 1 to 0 if the probability test is passed. The mutation probability guarantees that, theoretically, every part of the region of the search space is explored. The mutation operator adds diversity to the population while increasing the likelihood of generating individuals with better fitness values.
- **Selection:** based on each individual quality, the roulette method is used to determine the next population. The selection is stochastic and biased toward the best individuals. The first step is to calculate the cumulative fitness of the whole population through the sum of the fitness of all individuals. After that, the probability of selection is calculated for each individual as being $P_{Selection_i} = f_i / \sum_1^N f_i$.
- **Local search:** the last part of the algorithm is the use of a local search. A fast and simple heuristic is applied for each offspring during which it seeks for the new variable-value assignment which best decreases the number of unsatisfied clauses being identified.

4. Experimental results

4.1. Benchmark instances

We evaluated the performance of the multilevel evolutionary algorithm (MLVMA) against its single variant (MA) using a set of instances taken from SATLIB. (http://www.informatik.tu-darmstadt.de/AI/SATLIB). **Table 1** shows the instances used in the experiment. 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 cutoff parameter (max time) set to 15 min. The 100 runs were adopted because pilot runs had shown the size of the difference to be so large that 100 runs were 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 [20].

The tests were carried out on a DELL machine with 800 MHz CPU and 2 GB of memory. The code was written in C and compiled with the GNU C compiler version 4.6. The list of parameters used in the experiments are as follows:

Instance	Number of variables	Number of clauses
$2bitadd_{10}$.cnf	590	1422
$2bitadd_{11}$.cnf	649	1562
$2bitadd_{12}$.cnf	708	1702
$2bitcomp_{5}$.cnf	125	310
$2bitmax_{6}$.cnf	252	766
$2bitadd_{31}$.cnf	8432	31,310
$2bitadd_{32}$.cnf	8704	32,316
3block.cnf	283	9690
4blocks.cnf	758	47,820
4blocksb.cnf	410	24,758
e0ddr2-10-by-5-1.cnf	19,500	103,887
e0ddr2-10-by-5-4.cnf	1728	104,527
enddr2-10-by-5-1.cnf	20,700	111,567
enddr2-10-by-5-8.cnf	21,000	113,729
ewddr2-10-by-5-1.cnf	21,800	118,607
ewddr2-10-by-5-8.cnf	22,500	123,329

Table 1. Benchmark set of the SAT competition Beijing.

- Crossover probability = 0.85
- Mutation probability = 0.1
- Population size = 50
- Stopping criteria for the coarsening phase: the coarsening stops as soon as the size of the smallest problem reaches 100 variables (clusters). At this level, MA generates an initial population.
- Convergence: if the fitness of the best individual does not improve during 10 consecutive generations, MA is assumed to have reached convergence and moves to a higher level.

5. Results

5.1. Observed trends

The time development of the multilevel evolutionary algorithm against its single variant in solving the instances is shown in **Figures 2–8**. The plots show the 100 runs of both algorithms with a cutoff at 15 min as well as the mean of these runs. The search occurs in two phases. In the first phase, the best solution improves rapidly at first, and then flattens off as the search reaches the plateau region, marking the start of the second phase. The plateau region corresponds to a region in the search space where moves does not alter the best assignment, and

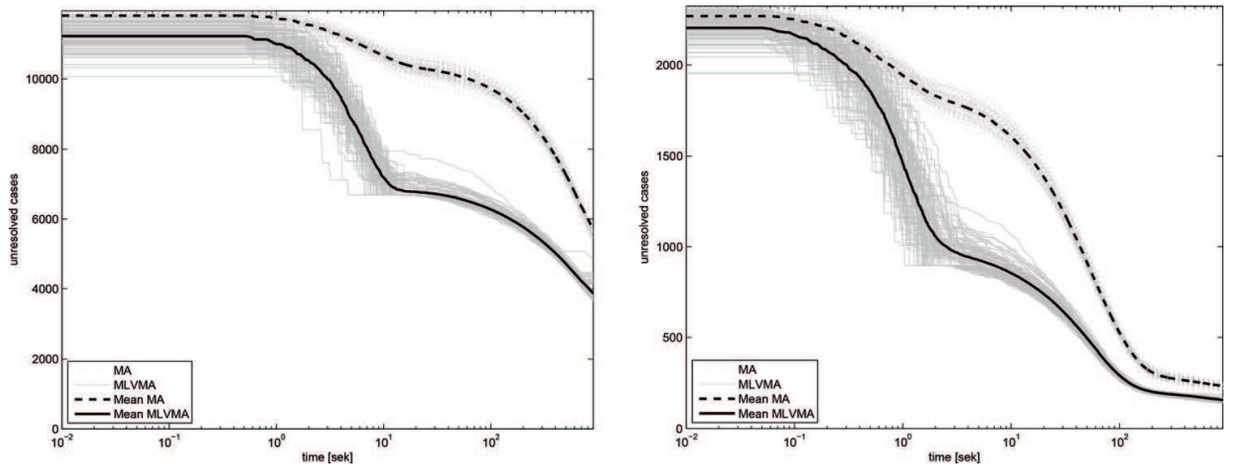

Figure 2. MLVMA versus MA: (left) 2bitadd$_{10}$.cnf, (right) 2bitadd$_{11}$.cnf—time development for 100 runs in 15 min.

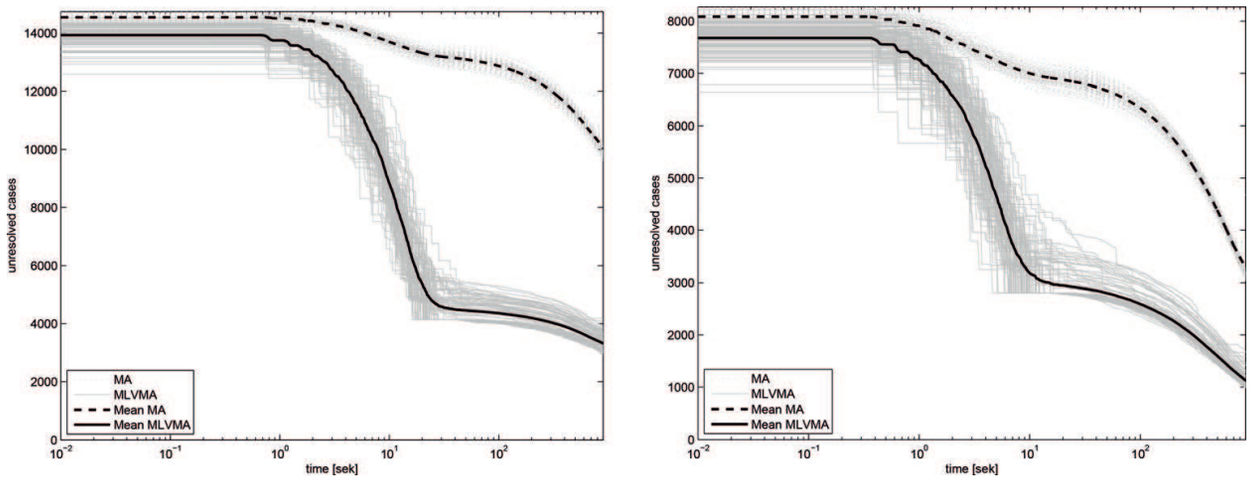

Figure 3. MLVMA versus MA: (left) 2bitadd$_{12}$.cnf, (right) 2bitcomp$_{5}$.cnf—time development for 100 runs in 15 min.

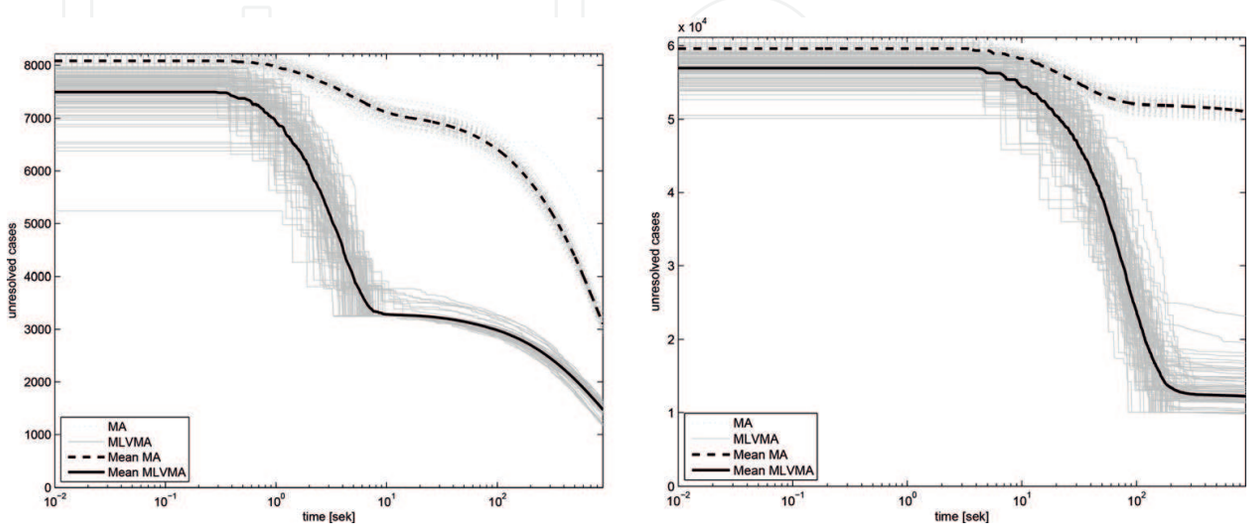

Figure 4. MLVMA versus MA: (left) 2bitmax$_{6}$.cnf, (right) 3bitadd$_{31}$.cnf—time development for 100 runs in 15 min.

http://dx.doi.org/10.5772/intechopen.72843

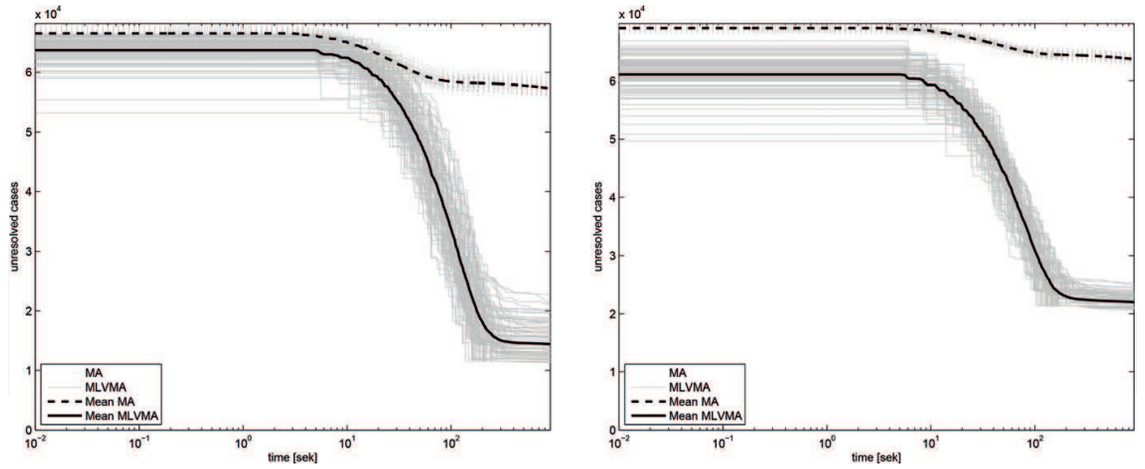

Figure 5. MLVMA versus MA: (left) 3bitadd$_{32}$.cnf, (right) 3block.cnf—time development for 100 runs in 15 min.

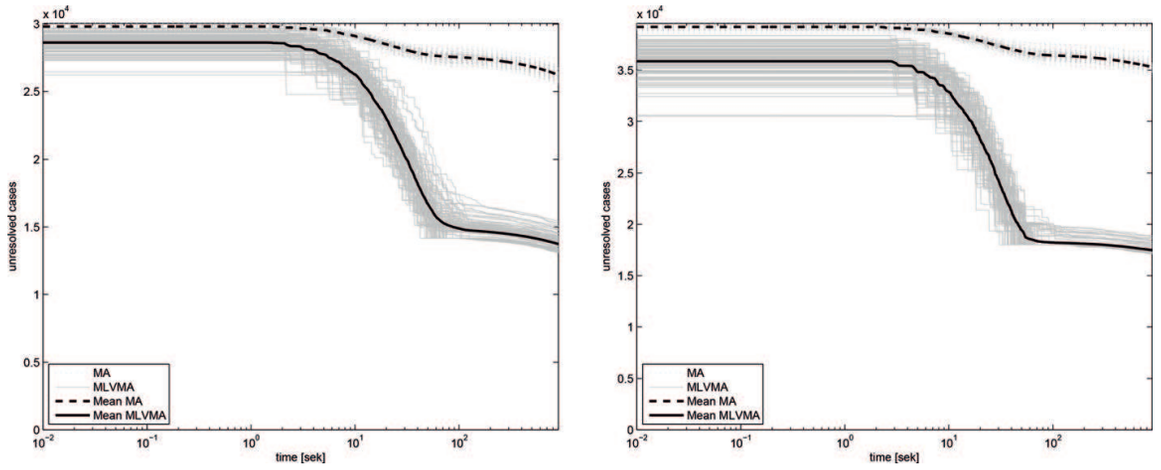

Figure 6. MLVMA versus MA: (left) 4blocks.cnf, (right) 4blocksb.cnf—time development for 100 runs in 15 min.

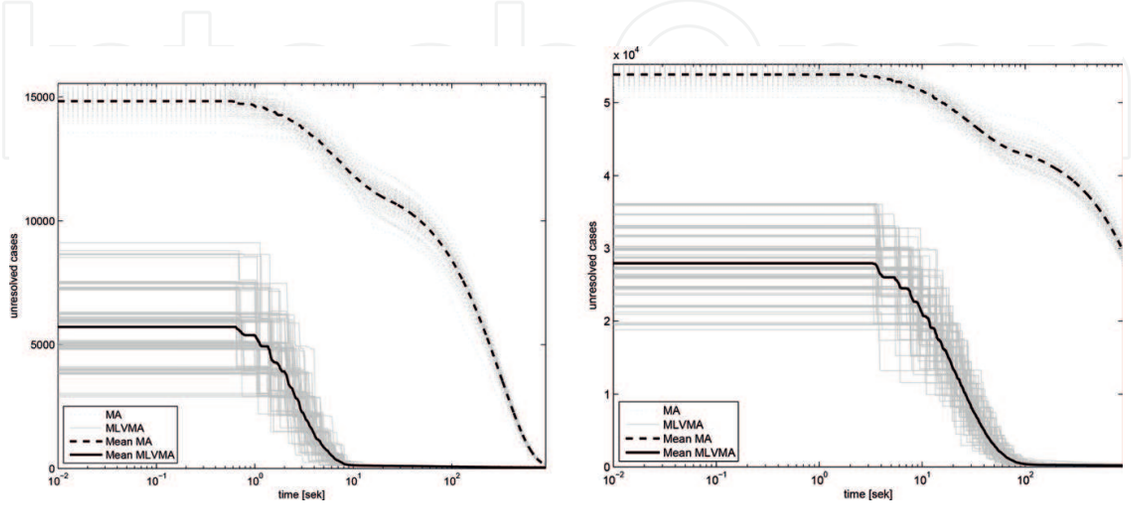

Figure 7. MLVMA versus MA: (left) e0ddr2-10-by-5-1.cnf, (right) e0ddr2-10-by-5-4.cnf—time development for 100 runs in 15 min.

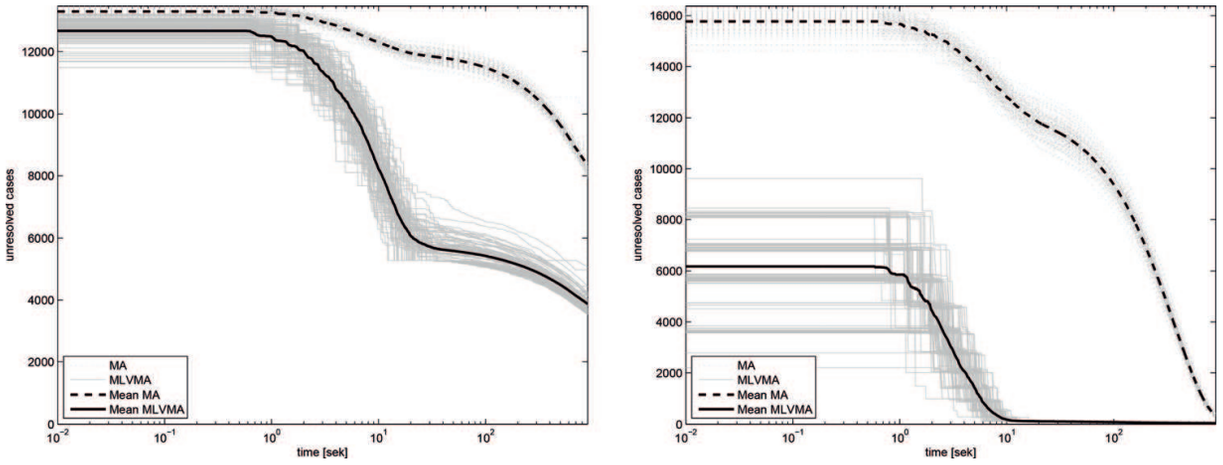

Figure 8. MLVMA versus MA: (left) enddr2-10-by-5-1.cnf, (right) enddr2-10-by-5-8.cnf—time development for 100 runs in 15 min.

occurs more specifically once the refinement reaches the finest level. The plots show that MLVMA offers a better asymptotic convergence compared to MA especially for large instances. The test cases where both algorithms reach approximately the same solution quality (with MLVMA being marginally better), the multilevel paradigm offers a cost-effective solution strategy considering the amount of time required (**Figure 9**).

This multilevel paradigm has two main advantages which enables the evolutionary algorithm to become much efficient. The coarsening process offers a better mechanism for performing diversification (i.e., searching different parts of the search space) and intensification (i.e., reaching better solutions within those regions). The coarsening allows the gene of each individual to represent a cluster of variables, leading the search to become guided and restricted to only those solutions 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 crossover and mutation operators are able to explore different regions in the search space

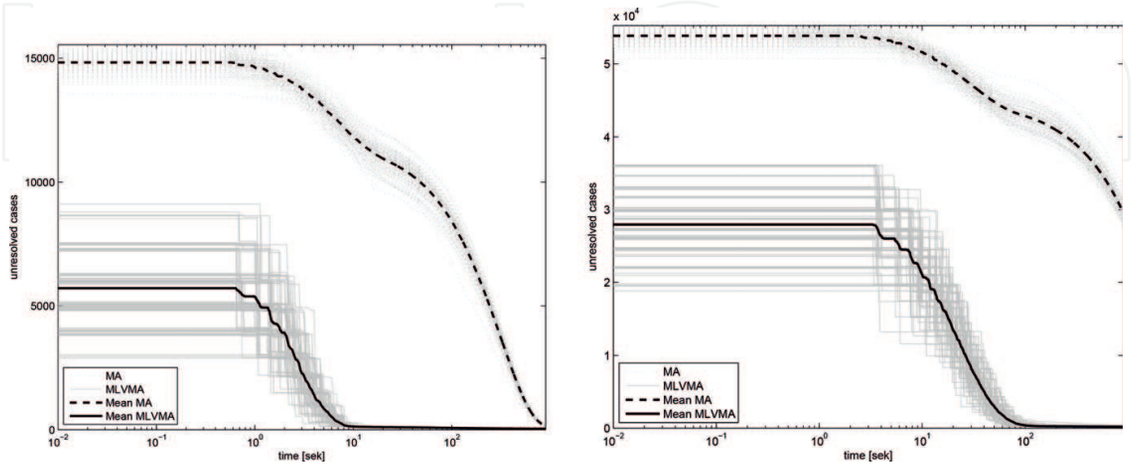

Figure 9. MLVMA versus MA: (left) ewddr2-10-by-5-1.cnf, (right) ewddr2-10-by-5-8.cnf—time development for 100 runs in 15 min.

while intensifying the search by exploiting the solutions from previous levels in order to reach better solutions.

5.2. Statistical analysis

Tables 2 and **3** summarize the results. M and SD represent the mean standard deviation of unsolved clauses for the MLVMA and MA algorithms. The range of solutions from each algorithm is also shown in order to analyze the overlap between solution spaces for any given instance. Statistical inferential analysis was done with an independent samples *t*-test which compares the difference in means between the two groups. Comparison using the non-parametric Mann-Whitney *U*-test gave identical results. The non-parametric effect size measure $\widehat{A}_{12}$ [21] was used to evaluate the relative dominance of one algorithm over the other. The $\widehat{A}_{12}$ effect size measure is calculated using the rank sum which is a common component in any non-parametric analysis such as the Mann-Whitney *U*-test [20]. Calculating $\widehat{A}_{12}$ is done according to the following formula:

$$\widehat{A}_{12} = (R_1/m - (m+1)/2)/n. \tag{1}$$

where R_1 is the rank sum of algorithm MLVMA, m is the number of observations in the first

#Case	MLVMA		MA	
	***M* (SD)**	**Range**	***M* (SD)**	**Range**
$2bitadd_{10}$.cnf	2.0 (.7)	[1–3]	16.3 (2.3)	[11–25]
$2bitadd_{11}$.cnf	1.7 (.7)	[1–4]	16.3 (3.2)	[8–24]
$2bitadd_{12}$.cnf	1.5 (.7)	[1–3]	1.6 (.7)	[1–4]
$2bitcomp_5$.cnf	1.0 (0)	[1–2]	1.0 (0.1)	[1–2]
$2bitmax_6$.cnf	1.0 (.2)	[1–2]	1.0 (0.1)	[1–2]
$2bitadd_{31}$.cnf	132.6 (10.9)	[122–216]	1106.2 (142.1)	[923–2620]
$3bitadd_{32}$.cnf	135.7 (11.9)	[123–186]	1366.9 (179.1)	[1125–1974]
3blocks	4.0 (1.8)	[2–9]	7.2 (1.0)	[4–9]
3blocks	8.2 (3.1)	[2–14]	13.0 (1.0)	[11–18]
4blocksb	5.2 (1.8)	[2–8]	7.3 (0.7)	[5–8]
e0ddr2-10-by-5-1	343.4 (119.0)	[261–697]	10871.1 (324.5)	[9895–11,527]
e0ddr2-10-by-5-4	320.6 (80.8)	[271–718]	10969.1 (360.1)	[10,190–11,784]
enddr2-10-by-5-1	371.9 (144.0)	[281–1021]	12042.9 (378.1)	[111,64–12,897]
enddr2-10-by-5-8	358.9 (136.1)	[278–967]	12241.3 (400.0)	[11,169–13,446]
ewddr2-10-by-5-1	399.8 (166.9)	[289–1124]	12939.7 (407.9)	[11,960–13,835]
ewddr2-10-by-5-8	354 (107.0)	[293–710]	13537.5 (423.8)	[12,393–14,736]

Table 2. Statistical comparisons.

#Case	Difference		Estimates of effect size	
	M diff. [95% CI of M diff.]	p	Obs. $\widehat{A}_{12}$	$\widehat{A}_{12}$[95% CI of $\widehat{A}_{12}$]
$2bitadd_{10}.cnf$	14.4 [13.8,14.9]	***	1	c
$2bitadd_{11}.cnf$	14.5 [13.9,15.2]	***	1	c
$2bitadd_{12}.cnf$	0.1 [−0.1,0.3]	.247	.548	.547 [.476,.622]
$2bitcomp_5$			.5	
$2bitmax_6$	0.0 [−0.1,0.3]	.653	.459	.459 [.475,.515]
$2bitadd_{31}$	973.6 [945.5,1001.7]	***	1	c
$3bitadd_{32}$	1231.1 [1195.8,1266.6]	***	1	c
3bloks	3.2 [2.8,3.6]	***	.918	.920 [.877,.958]
4blocks	4.8 [4.1,5.4]	***	.916	.917 [.878,.953]
4blocksb	2.1 [1.7,2.4]	***	1	c
e0ddr2-1-by-5-1	10527.7 [10459.5,10595.8]	***	1	c
e0ddr2-1-by-5-4	10648.5 [10575.8,10721.3]	***	1	c
enddr2-10-by-5-1	10671.0 [11591.2,11750.8]	***	1	c
enddr2-10-by-5-8	11882.4 [11799.1,11965.7]	***	1	c
ewddr2-10-by-5-1	12539 [12453.0,112626.8]	***	1	c
ewddr2-10-by-5-8	13182.9 [13096.7,13269.1]	***	1	c

*** means $p < 0.0001$.

Table 3. Comparing effect sizes.

data sample, and n is the number of observations in the second data sample. Calculating $\widehat{A}_{12}$ results in a number between 0 and 1 which represent the probability that MLVMA will yield a better solution than MA. If the two algorithms are equivalent, then $\widehat{A}_{12} = .5$, while a complete dominance of algorithm MLVMA over MA would entail $\widehat{A}_{12} = 1$.

$\widehat{A}_{12}$ is more easily interpreted than the more common parametric Cohen's d [22] which represents the mean difference between two groups in standard deviations for several reasons. First, Cohen's d assumes that the observed samples are normally distributed [20]. Second, when dealing with solutions to optimization problems, a researcher or a practitioner would only be interested in the single best solution given a sample of different solutions from one or more algorithms. Hence, using an effect size measure that indicates the probability that one algorithm would lead to a better solution than another (given the same amount of time) would be more informative and more easily interpretable for an optimization practitioner. The 95% confidence intervals of $\widehat{A}_{12}$ shown in **Table 3** (where applicable) are calculated using a bootstrapping procedure [23] which is used to estimate the 95% confidence interval of $\widehat{A}_{12}$. The procedure uses a computer-intensive step-by-step process that consists of the following three steps:

1. Random resampling with replacement from the original observations to create new data sets.
2. Calculation of the rank sum of MLVMA for each new data set.
3. Using the rank sum to calculate $\widehat{A}_{12}$ with Eq. (1). The three steps are then repeated 1000 times and the resulting statistic $\widehat{A}_{12}$ is saved to create a sampling distribution of the statistic $\widehat{hat}_{12}$.

The results show how MLVMA outperforms MA in 10 out of the 16 instances. MLVMA dominates MA in three instances (the 3blocks, 4blocks, and 4blocksb-instances, $\widehat{A}_{12}$, is .918, .916, and .847, respectively). For the remaining three problems ($2bitadd_{10}$, $2bitadd_{11}$, and $2bitadd_{12}$), there is no statistically identifiable difference between the two algorithms. However, when inspecting the time series for these instances it is clear that MLVMA reaches a solution much faster than MA. To test possible causes for the difference in solution quality, the relationship between the number of clauses and the quality of solutions provided by the two algorithms was analyzed. The relationship between the mean percentage of unsolved clauses and the number of clauses in each instance was estimated using a linear regression. The relationship between the mean percentage of unsolved clauses and the number of clauses for the MLVMA was much lower ($t(15) = 3.059$, = 2.041–8, 95% CI [1.163–8, 2.714–8], $p = .008$, $r = .633$) than for the MA ($t(15) = 10.067$, = 9.341–7, 95% CI [8.232–7, 1.04–6], $p < .001$, $r = .937$) indicating that the hierarchical paradigm is less affected by the size of the problem than the standard single-level evolutionary algorithm.

6. Conclusion

In this chapter, a multilevel evolutionary algorithm for solving the maximum satisfiability problem is presented. During the coarsening phase, a sequence of smaller problems, each with fewer variables, is constructed. Each child level is constructed from its parent level by collapsing pairs of variables. The new formed variables 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. An evolutionary algorithm is applied through several optimization levels, where the converged population at a child level will serve as the starting population for a parent level. A set of instances were used to compare the performance of the new approach. The results obtained assert the superiority of the evolutionary algorithm when combined with the multilevel paradigm and always return a better solution for the equivalent run-time compared to MA.

Author details

Noureddine Bouhmala*, Kjell Ivar Øvergård and Karina Hjelmervik

*Address all correspondence to: noureddine.bouhmaa@usn.no

Department of Maritime Technology and Innovation, SouthEast University, Norway

References

[1] Blum C, Puchinger J, Raidl GR, Roli A. Hybrid metaheuristics in combinatorial optimization: A survey. Applied Soft Computing. 2011;**11**:4135-4151

[2] Blum C, Roli A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Computing Surveys. September 2003;**35**(3):268–308

[3] Barnard ST, Simon HD. A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. Concurrency: Practice and Experience. 1994;**6**(2): 101-117

[4] Hadany R, Harel D. A multi-scale algorithm for drawing graphs nicely. Discrete Applied Mathematics. 2001;**113**(1):3-21

[5] Karypis G, Kumar V. A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM Journal on Scientific Computing. 1998;**20**(1):359-392

[6] Karypis G, Kumar V. Multilevel k-way partitioning scheme for irregular graphs. Journal of Parallel and Distributed Computing. 1998;**48**(1):96-129

[7] Walshaw C, Cross M. Mesh partitioning: A multilevel balancing and refinement algorithm. SIAM Journal of Scientific Computing. USA. 2000;**22**(1):63-80

[8] Walshaw C. A multilevel approach to the traveling salesman problem. Operational Research. 2002;**50**(5):862-877

[9] Walshaw C. A multilevel Lin-Kernighan-Helsgaun algorithm for the travelling salesman problem. Technical Report 01/IM/80, Mathematics and Computing Science. London, UK: University of Greenwich; 2001

[10] Rodney D, Soper A, Walshaw C. The application of multilevel refinement to the vehicle routing problem. In: Fogel D. et al., editors. Proceedings of the CISChed 2007. Piscataway, NJ: IEEE Symposium on Computational Intelligence in Scheduling; 2007. pp. 212-219

[11] Walshaw C. A multilevel algorithm for forced-directed graph-drawing. Journal of Graph Algorithms and Applications. 2003;**7**(3):253-285

[12] Oduntan IO, Toulouse M, Baumgartner R, Bowman C, Somorjai R, Crainic TG. A multilevel tabu search algorithm for the feature selection problem in biomedical data. Computers & Mathematics with Applications. 2008;**55**(5):1019-1033

[13] Bouhmala N. A multilevel genetic algorithm for the clustering problem. International Journal of Information and Communication Technology. 2016;**9**(1):101-116

[14] Bouhmala N. A multilevel learning automata for MAX-SAT. International Journal of machine Learning Cybernetics. Berlin Heidelberg: Springer-Verlag. 2015;**6**(6):911-921. DOI: 10.1007/s13042-015-0355-4

[15] Bouhmala N, Hjelmervik K, Kjell Ivar O. Single vs hierarchical population-based evolutionary algorithm for SAT-encoded industrial problems: A statistical comparison. International Journal of Artificial Intelligence Applications. 2012;**3**(6):57-73

[16] Bouhmala N, Granmo OC. GSAT enhanced with learning automata and multilevel paradigm. International Journal of Computer Science Issues. 2011;**8**(3)

[17] Pirkwieser S, Raidl GR. Multilevel variable neighborhood search for periodic routing problems. In: Cowling PI, Merz P, editors. Proceedings of EvoCOP 2010 10th European Conference on Evolutionary Computation in Combinatorial Optimization. Vol. 6022 of Lecture Notes in Computer Science. Berlin, Germany: Springer-Verlag; 2010. pp. 226-238

[18] Walshaw C. Multilevel refinement for combinatorial optimization: Boosting metaheuristic performance. In: Blum C. et al., editors. Heidelberg, Berlin, Germany: Springer; 2008. pp. 261-289

[19] Spears W. Adapting crossover in evolutionary algorithms. In: Proceedings of the Fourth Annual Conference on Evolutionary Programming. MIT Press; 1995. pp. 367-384

[20] Arcuri A, Briand L. A Hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering. Technical report, simula research laboratory, number 13/2011

[21] Bouhmala N. A multilevel evolutionary algorithm for large SAT-encoded problems. Evolutionary Computation. 2012;**20**(4):641-664

[22] Cohen J. Statistical Power Analysis for the Behavioral Sciences. 2nd ed. New York University: Lawrence Erlbaum; 1998

[23] Mooney CZ, Duval RD. Bootstrapping—A Nonparametric Approach to Statistical Inference. Sage University Press; 1993

IntechOpen