Combining an Evolutionary Algorithm with the Multilevel Paradigm for Solving the Satisfiability Problem

N. Bouhmala
Vestfold University College
Norway
noureddine.bouhmala@hive.no

ABSTRACT: *The satisfiabiltiy problem refers to the task of finding a satisfying assignment that makes a Boolean expression evaluate to True. In this work, an evolutionary algorithm enhanced with the multilevel paradigm is introduced. The multilevel paradigm aims at 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. Results comparing the proposed algorithm with and without the multilevel paradigm are presented.*

Keywords: Satisfiability problem, Memetic algorithm, Multilevel techniques

Received: 30 July 2011, Revised 17 September 2011, Accepted 27 September 2011

1. The Satisfiability Problem

The satisfiability problem (SAT) which is known to be NP-complete [6] plays a central role problem in many pplications in the fields of VLSI Computer-Aided design, Computing Theory, and Artificial Intelligence. Generally, *a* SAT problem is defined as follows. A propositional formula $\Phi = \Lambda^{m}_{j=1}$ 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:

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

where $I_j, \bar{I}_j \subseteq \{1, ...n\}$, $I \cap \bar{I}_j = \phi$, 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. Since we have two choices for each of the n Boolean variables, the size of the search space S becomes $|S| = 2^n$. That is, the size of the search space grows exponentially with the number of variables. Since most known combinatorial optimization problems can be reduced to SAT [9] the design of special methods for SAT can lead to general approaches for solving combinatorial optimization problems. 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. For example, *P* v *Q* is a clause containing the two literals *P* and *Q*. The clause *P* v *Q* is satisfied if either *P* is ***True*** or *Q* is True. When each clause in Φ contains exactly *k* literals, the resulting SAT problem is called *k* - SAT.

The rest of the paper is organized as follows. Section 2 provides an overview of algorithms used for solving the satisfiability problem. Section 3 reviews some of the algorithms combined with the multilevel paradigm. Section 4 gives a general description of an evolutionary algorithm. Section 5 introduces the multilevel evolutionary algorithm. Section 6 shows the results of the

experiments. Finally, in Section 7 we present a summary and some guidelines for future work.

2. Algorithms for Solving SAT: A survey

2.1 Family of GSAT

One of the earliest local search algorithms for solving SAT is GSAT[42]. Basically, GSAT begins with a random generated assignment of values to variables, and then uses the steepest descent heuristic to find the new variable-value assignment which best decreases the number of unsatisfied clauses. After a fixed number of moves, the search is restarted from a new random assignment. The search continues until a solution is found or a fixed number of restarts have been performed. The introduction of an element of randomness (i.e, noise) into a local search method is common practice for improving its effectiveness through diversification [3]. To this end, an extension of GSAT, referred to as random-walk [43] has been realized with the purpose of escaping from local optima. In a random walk step, a randomly unsatisfied clause is selected. Then, one of the variables appearing in that clause is flipped, thus effectively forcing the selected clause to become satisfied. The main idea is to decide at each search step whether to perform a standard GSAT or a random-walk strategy with a probability called the walk probability.

2.2 Family of Walksat

Another widely used variant of GSAT is the Walksat algorithm originally introduced in [44]. It first picks randomly an unsatisfied clause, and then, in a second step, one of the variables with the lowest *break count*, appearing in the selected clause, is randomly selected. The break count of a variable is defined as the number of clauses that would be unsatisfied by flipping the chosen variable. If there exists a variable with break count equals to zero, this variable is flipped, otherwise the variable with minimal break count is selected with a certain probability (noise probability). It turns out that the choice of unsatisfied clauses, combined with the randomness in the selection of variables, can enableWalksat to avoid local minima and to better explore the search space. Extensive tests have led to the introduction of new variants of theWalksat algorithm referred to as Novelty and R-Novelty [34][19]. These two variants use a combination of two criteria when choosing a variable to flip from within an unsatisfied clause. Quite often, these two algorithms can get stuck in local minima and fail to get out. To this end, recent variants have been designed [32][30][21] using a combination of search intensification and diversification mechanisms leading to good performance on a wide range of SAT instances.

2.3 Weighting Techniques

Other algorithms [12] [16] [11] used historybased variable selection strategies in order to avoid flipping the same variable. In parallel to the development of more sophisticated versions of randomized improvement techniques, other methods based on the idea of modifying the evaluation function [50] [21,22] [44] [39] in order to prevent the search from getting stuck in non attractive areas of the underlying search space have become increasingly popular in SAT solving. The key idea is to associate the clauses of the given CNF formula with weights. Although these clause weighting algorithms differ in the way clause weights should be updated (probabilistic or deterministic), they all choose to increase the weights of all the unsatisfied clauses as soon as a local minimum is encountered. A new approach to clause weighting known as Divide and Distribute Fixed Weights (DDFW) [22] exploits the transfer of weights from neighboring satisfied clauses to unsatisfied clauses in order to break out from local minima. Recently, a strategy based on assigning weights to variables [37] instead of clauses greatly enhances the performance of the Walksat algorithm, leading to the best known results on some benchmarks.

2.4 Evolutionary Algorithms

Evolutionary algorithms are heuristic algorithms that have been applied to SAT and many other NP-complete problems. Unlike local search methods that work on a current single solution, evolutionary approaches evolve a set of solutions. GASAT [23] [29] is considered to be the best known genetic algorithm for SAT. GASAT is a hybrid algorithm that combines a specific crossover and a tabu search procedure. Experiments have shown that GASAT provides very competitive results compared with state-of-art SAT algorithms. [5]. proposed several evolutionary algorithms for SAT [13]. Results presented in that paper show that evolutionary algorithms compare favorably to Walksat. Finally, Boughaci et al. introduced a new selection strategy [5] based on both fitness and diversity to choose individuals to participate in the reproduction phase of a genetic algorithm. Experiments showed that the resulting genetic algorithm was able to find solutions of a higher quality than the scatter evolutionary algorithm [4].

2.5 Recent Strategies

Lacking the theoretical guidelines while being stochastic in nature, the deployment of several metaheuristics involves extensive experiments to find the optimal noise or walk probability settings. To avoid manual parameter tuning, new methods have been

designed to automatically adapt parameter settings during the search [33][36], and results have shown their effectiveness for a wide range of problems. The work conducted in [14] introduced Learning Automata (LA) as a mechanism for enhancing local search based SAT solvers, thus laying the foundation for novel LA-based SAT solvers. Finally, a new recent strategy based on an automatic procedure for integrating selected components from various existing solvers have been devised in order to build new efficient algorithms that draw the strengths of multiple algorithms [27,51].

3. Multilevel Techniques

The multilevel paradigm is a simple technique which at its core involves recursive coarsening to produce smaller and smaller problems that are easier to solve than the original one. Figure 1 shows the process of the generic multilevel paradigm in pseudo-code.

The multilevel paradigm starts by merging the variables associated with the problem to form clusters. The clusters are used in a recursive manner to construct a hierarchy of problems each representing the original problem but with fewer degrees of freedom. The coarsest level can then be used to compute an initial solution. The solution found at the coarsest level is extended to give an initial solution for the next level and then improved using a chosen algorithm. A common feature that characterizes multilevel algorithms, is that any solution in any of the coarsened problems is a legitimate solution to the original problem. Techniques using the the multilevel paradigm draw their strength form coupling the refinement process across different levels. Multilevel techniques were first introduced when dealing with the graph partitioning problem [1] [15] [17] [25] [26] [46] and have proved to be effective in producing high quality solutions at a lower cost than single level techniques. The traveling salesman problem was the second combinatorial optimization problem to which the multilevel paradigm was applied [47] [48] and has clearly shown a clear improvement in the asymptotic convergence of the solution quality. When the multilevel paradigm was applied to the graph coloring problem [38], the results do not seem to be in line with the general trend.

Input: Problem P_0;
Output: Solution $S_{final}(P_0)$;
Begin
level := 0;
While (**NOT** reached the desired number of levels) **Do**
$P_{level+1}$:= Coarsen (P_{level});
level := level + 1 ,
EndWhile
/* **Initial Solution is computed at the lowest level** */
$S(P_{level})$:= Initial Solution (P_{level});
While(($level > 0$)) Do
$S_{start}(P_{level-1})$:= *Uncoarsen* ($S_{final}(P_{level})$) ;
$S_{final}(P_{level-1})$:= Refine ($S_{start}(P_{level-1})$);
level := level - 1;
EndWhile
End.

Figure 1. The multilevel generic algorithm

observed in graph partitioning problem and traveling salesman problem as its ability to enhance the convergence behavior of the local search algorithms was rather restricted to some class of problems. Graph drawing is another area where multilevel techniques gave a better global quality to the drawing and the author suggests its use to both accelerate and enhance force drawing placement algorithms [45].

4. Memetic Algorithms (MAs)

An important prerequisite for the the multilevel paradigm is the use of an optimization strategy in order to carry out the refinement during each level. In this work we look at memetic algorithms [49] [28] (MAs). MAs represent the set of hybrid algorithms that combine genetic algorithms and local search. In general the genetic algorithm improves the solution while the local fine tunes the solution. They are adaptive based search optimizations algorithms that take their inspiration form genetics and evolution process [35]. Figure 2 provides a canonical memetic 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 or to a selected subset of individuals.
        Evaluate the fitness of the intermediate population;
        Replace the parent population with a new generation;
    EndWhile
End.
```

Figure 2. A canonical memetic algorithm

MAs simultaneously examine and manipulate a set of possible solution. Given a specific problem to solve, the input to MAs is an initial population of solutions called individuals or chromosomes. A gene is part of a chromosome, which is the smallest unit of genetic information. Every gene is able to assume different values called allele. All genes of an organism form a genome which affects the appearance of an organism called phenotype. The chromosomes are encoded using a chosen representation and each can be thought of as a point in the search space of candidate solutions. Each individual is assigned a score (fitness) value that allows assessing its quality. The members of the initial population may be randomly generated or by using sophisticated mechanisms by means of which an initial population of high quality chromosomes is produced. The reproduction operator selects (randomly or based on the individual's fitness) chromosomes form the population to be parents and enters them in a mating pool. Parent individuals are drawn form the mating pool and combined so that information is exchanged and passed to offspring depending on the probability of the cross-over operator. The new population is then subjected to mutation and entered into an intermediate population. The mutation operator acts as an element of diversity into the population and is generally applied with a low probability to avoid disrupting crossover results. The individuals from the intermediate population are then enhanced with a local search and evaluated. The local search used is a simple one that chooses the variable-value assignment with the largest decrease or the smallest increase in the numbers of unsatisfied clauses. This heuristic is executed for one iteration in order to reduce the time complexity. Random tie breaking strategy is used between variables with identical score. Finally, a selection scheme is used to update the population giving rise to a new generation. The individuals from the set of solutions which is called population will evolve form generation to generation by repeated applications of an evaluation procedure that is based on genetic operators and a local search scheme. Over many generations, the population becomes increasingly uniform until it ultimately converges to optimal or near-optimal solutions.

5. The Multilevel Memetic Algorithm (MLVMA)

The implementation of a multilevel algorithm requires four basic components: a coarsening algorithm, an initialization algorithm , an extension algorithm (which takes the solution on one problem and extends it to the parent problem), and an improvement algorithm. In this section we describe all these components which are necessary for the derivation of a memetic algorithm operating in a multilevel context.

5.1 Coarsening

Let $G^0_\Phi = (V,E)$ (the subscript represents the level of graph scale) be an undirected graph of vertices V and edges E. The set V denotes the Boolean variables and each edge $(i, j) \in E$ implies that the variables i and j belong to the same clause. Let $w_{i,j}$ denotes the non-negative weight of the edge (i, j). Coarsening aims at transforming G^0_Φ into a sequence of smaller graphs G^1_Φ, G^1_Φ,.....G^m_Φ such that $|G^0_\Phi| > |G^1_\Phi| > ....|G^m_\Phi|$. Let G^0_Φdenotes the original graph. The next level coarser graph G is constructed form G^0_Φ by fusing adjacent vertices. Thus, the edge joining two vertices is contracted and a new vertex consisting of these two vertices is created. This edge contraction procedure can be accomplished by finding a maximal matching. The maximal matching is a maximal subset $M \subset E$ of edges no two of which share a vertex. Given the matching M, the coarser graph G is constructed as follows: for each edge $(v_i,v_j) \in M$, we contract its endpoints vi and vJ into a new vertex. The weight of the new vertex is set equal to the sum of the weights of its constituent vertices. Its neighbors are the combined neighbors of the merged vertices viand v_j .Our maximal matching is computed using a randomized algorithm similar to [17]. The vertices are visited in a random order. If a variable vi has not been matched yet, then we randomly select one of its randomly unmatched variable v_j , and a new variable consisting of these two variables is created. If vertex v_j exits, we include the edge (v_i,v_j) in the matching. Unmatched vertices are simply copied to the next level. The weight of a vertex v_i in a coarser graph and that of its incident vertices provide an upper bound of the clauses that can be affected if vi is flipped. This coarsening procedure has the property of retaining the information regarding the overall structure of the original graph. Thus, any assignment of the coarse graph corresponds naturally to an assignment of the original graph. It will require at least O ($log\ N / N'$) coarsening phases to coarsen the problem down to N' vertices. The new formed vertives 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.

5.2 Start solution

The coarsening procedure ceases when the problem size shrinks to a desired threshold. Initialization is then trivial and consists of generating an initial population P_m of the graph $|G^m_\Phi|$ using a random procedure.

5.3 Uncoarsening

The population P_{m+1} of the graph $|G^{m+1}_\Phi|$ is projected onto its parent $|G^m_\Phi|$. Since the gene of each chromosome of $|G^{m+1}_\Phi|$ contains a distinct subset of variables of $|G^m_\Phi|$, obtaining P_m form P_{m+1} is done by simply assigning the set of variables V^v_m collapsed to $v \in$ m+1 the same value as v (i.e., $C_i[u] = C_{i+1}[v]$, $\forall\ u \in V^u_i$, where C_i denotes the chromosome of $|G^i_\Phi|$).

5.4 Refinement

The idea of improvement is to use the projected population P_{m+1} onto P_m as the initial population for further improvement using a memetic algorithm. Even though P_{m+1} is a local minimum of $|G^{m+1}_\Phi|$, the projected population Pm may not be at a local optimum with respect to G^m_Φ . The projected population is already a good solution and contains individuals with high fitness value, MA will converge quicker within a few generation to a better assignment. The next subsection describes the main features of the memetic algorithm used in this work.

5.4.1 Implementation Issues

a) Fitness function:

The notion of fitness is fundamental to the application of memetic algorithms. It is a numerical value that expresses the performance of an individual (solution) so that different individuals can be compared. The fitness of a chromosome is equal to the number of clauses that are unsatisfied by the truth assignment represented by the chromosome.

b) Representation :

A representation is a mapping from the state space of possible solutions to a state of encoded solutions within a particular data structure. The chromosomes which are assignments of values to the variables are encoded as strings of bits, the length of which is the number of variables. The values ***True*** and ***False*** are represented by 1 and 0 respectively. In this representation , an individual X corresponds to a truth assignment and the search space is the set $S = \{0, 1\}^n$.

c) Initial population:

The initial population consists of individuals (assignments) generated randomly in which each gene's allele is assigned the

value 0 or 1.

d) Cross-over:
The task of the cross-over operator is to reach regions of the search space with higher average quality. New solutions are created by combining pairs of individuals in the population and then applying a crossover operator to each chosen pair. 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 twopoint crossover selects two randomly points within a chromosome and then interchanges the two parent chromosomes between these points to generate two new offspring. Recombination can be defined as a process in which a set of configurations (solutions referred as parents) undergoes a transformation to create a set of configurations (referred as offspring). The creation of these descendants involves the location and combinations of features extracted from the parents.

e) Mutation:
The purpose of mutation which is the secondary search operator used in this work, is to generate modified individuals by introducing new features in the population. By mutation, the alleles of the produced child have a chance to be modified, which enables further exploration of the search space. The mutation operator takes a single parameter p_m ,which specifies the probability of performing a possible mutation. Let $C = c_1, c_2, ...c_m$ be a chromosome represented by a binary chain where each of whose gene c_i is either 0 or 1. In our mutation operator, each gene c_i is mutated through flipping this gene's allele from 0 to 1 or vice versa if the probability test is passed. The mutation probability ensures that, theoretically, every region of the search space is explored. If on the other hand, mutation is applied to all genes, the evolutionary process will degenerate into a random search with no benefits of the information gathered in preceding generations. The mutation operator prevents the searching process form being trapped into local optimum while adding to the diversity of the population and thereby increasing the likelihood that the algorithm will generate individuals with better fitness values.

f) Selection:
The selection operator acts on individuals in the current population. During this phase, the search for the global solution gets a clearer direction, whereby the optimization process is gradually focused on the relevant areas of the search space. Based on each individual quality (fitness), it determines the next population. In the roulette method, 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 $P_{Selection_i} = f_i / \Sigma_1^N f_i$.

g) Local Search:
Finally, the last component of the proposed MA is the use of local improvers. By introducing local search at this level, the search within promising areas is intensified. This local search should be able to quickly improve the quality of a solution produced by the crossover operator, without diversifying it into other areas of the search space. In the context of optimization, this rises a number of questions regarding how best to take advantage of both aspects of the whole algorithm. With regard to local search there are issues of which individuals will undergo local improvement and to what degree of intensity. However care should be made in order to balance the evolution component (exploration) against exploitation (local search component). Bearing this thought in mind, the strategy adopted in this regard is to let each chromosome go through a low rate intensity local improvement. This heuristic is similar to that used for the memetic algorithm except that it works on a cluster of variables rather than a single variable. During one iteration, the heuristic seeks for the cluster-value assignment with the largest decrease or the smallest increase in the numbers of unsatisfied clauses. Random tie breaking strategy is used between clusters with identical score.

6. Experimental Results

6.1 Test suite
Tables 1 and 2 list the test suite of 33 problem instances used to compare the two algorithms. The suite consists of boundary model checking instances (BMC) taken from industrial hardware designs and SAT-encoded quasigroup instances. Due to the randomization nature of the two algorithms, each problem instance was run 50 times. The time limit was set to 300 sec for short runs and to 1 hour for long runs. The reason to compare algorithms on short and on long runs is to evaluate their ability in producing relatively good solutions under strong time constraints versus producing high quality solutions where additional computational resources are available. The tests were carried out on a a DELL machine with 800 MHz CPU and 2 GB of memory.

instance	number of variables	number of clauses
qg1-07.cnf	343	68083
qg1-08.cnf	512	148957
qg2-07.cnf	343	68083
qg2-08.cnf	512	148957
qg3-08.cnf	512	10469
qg3-09.cnf	729	16732
qg4-08.cnf	512	9685
qg4-09.cnf	729	15580
qg5-09.cnf	729	28540
qg5-10.cnf	1000	43636
qg5-11.cnf	1331	64054
qg5-12.cnf	1728	90919
qg5-13.cnf	2197	125464
qg6-09.cnf	729	21844
qg6-10.cnf	1000	33466
qg6-11.cnf	1331	49204
qg6-12.cnf	1728	69931
qg7-09.cnf	729	22060
qg7-10.cnf	1000	33736
qg7-11.cnf	1331	49534
qg7-12.cnf	1728	70327
qg7-13.cnf	2197	97072

Table 1. SAT-encoding of quasigroup instances

instance	number of variables	number of clauses
bmc-ibm-1.cnf	9658	55870
bmc-ibm-2.cnf	3628	14468
bmc-ibm-3.cnf	14930	72106
bmc-ibm-5.cnf	9396	41207
bmc-ibm-7.cnf	8710	39774
bmc-galileo-8.cnf	58074	294821
bmc-galileo-9.cnf	63624	326999
bmc-ibm-10.cnf	61088	334861
bmc-ibm-11.cnf	32109	150027
bmc-ibm-12.cnf	39598	19477
bmc-ibm-13.cnf	13215	6527

Table 2. Boundary model checking instances

code was written in C and compiled with the GNU C compiler version 4.6. The parameters used in the experiments are listed below:

• 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 coarsest problem reaches 100 vertices.

• Convergence during the improvement phase: If no improvement of the fitness function of the best individual has not been observed during 50 consecutive generations, MA is assumed to have reached convergence and moves to a higher level.

6.2 Results

6.2.1 Mean Fitness

Figure 3 plots the average fitness of the entire population at each generation for the two algorithms using bmc-ibm-13.cnf as an example. This example was selected as it represents the general trend observed for all the instances tested. It can be seen in general that the average fitness improves consistently as the number of generations increases. However, MLVMA evolves to a relatively lower average fitness value and at a faster rate compared to MA. As the number of generations gets large, the population tends to loose its diversity, and all individuals in the population tend to be identical with almost the same average fitness value.

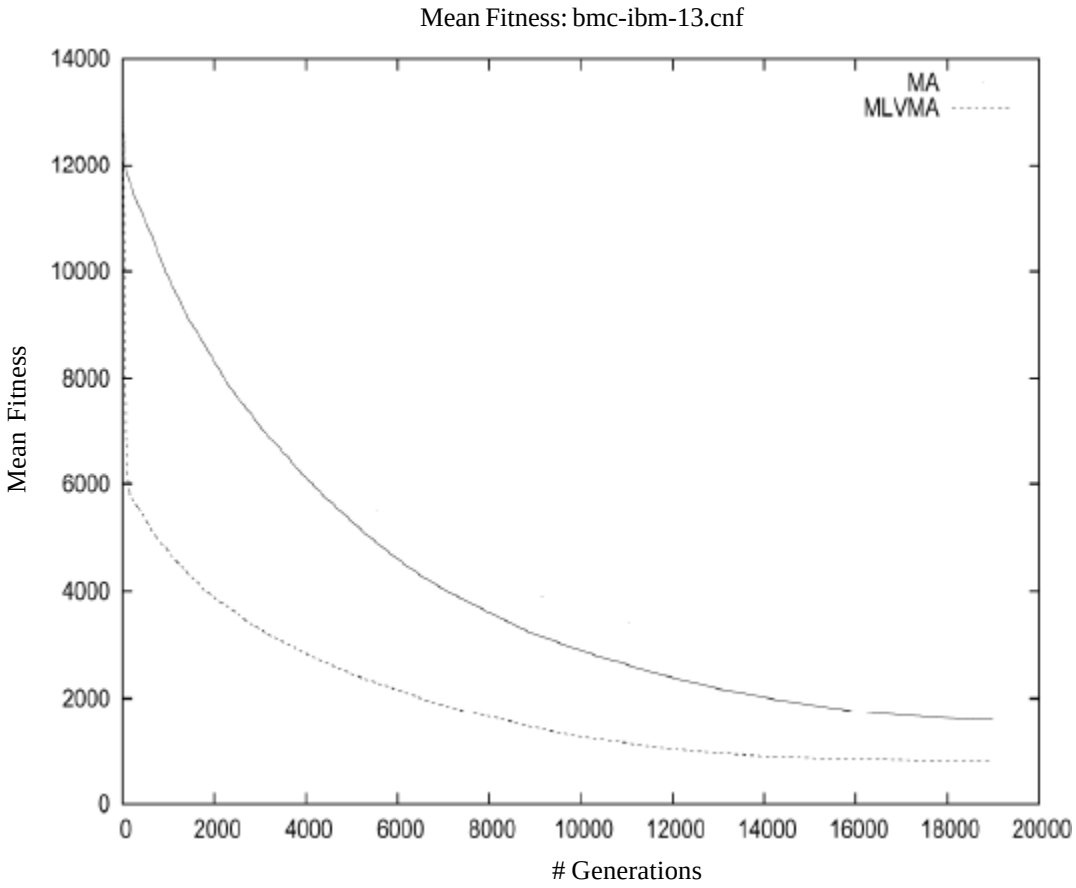

Figure 3. Evolution of Mean Fitness for bmc-ibm-13.cnf: $|V| = 13215$, $|C| = 6572$. Along the horizontal axis we give the number of generations and along the vertical axis the mean fitness

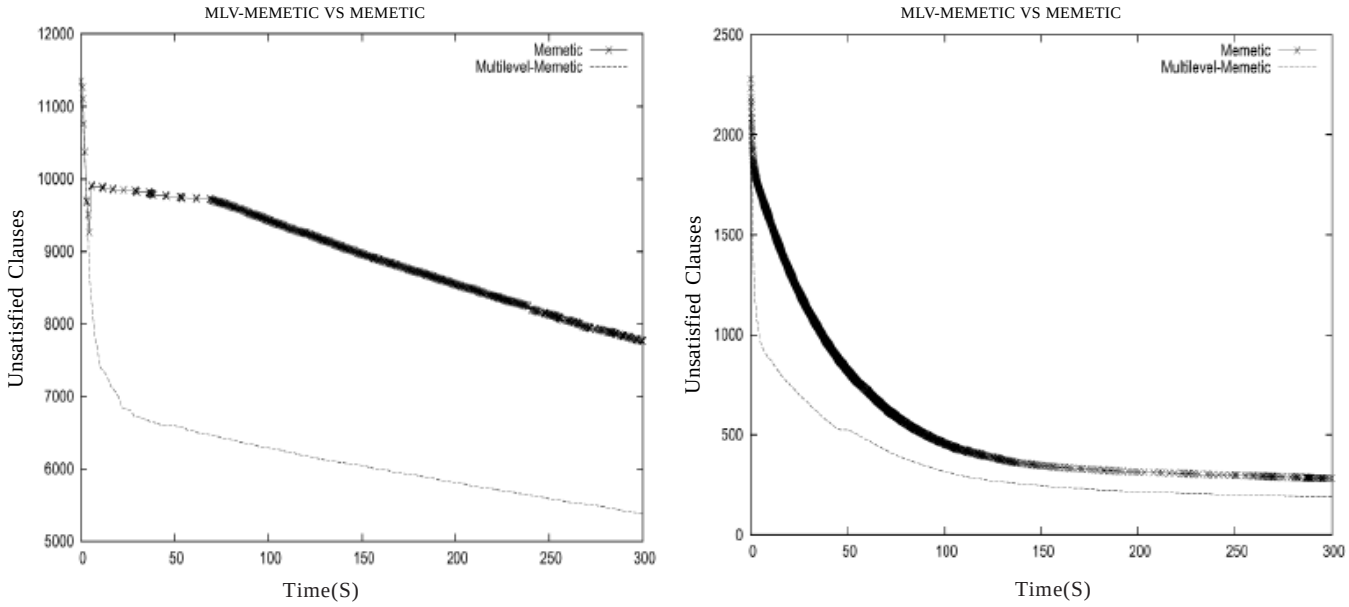

Figure 4. (Left) bmc-ibm-1.cnf: |V | = 9685 , |C| = 55870. (Right) bmc-ibm-2.cnf: |*V*| = 3628, |*C* | = 14468. Along the horizontal axis we give the time in seconds, and along the vertical axis the number of unsatisfied clauses

6.2.2 Evolution of the best solution

Figures 4 - 6 show how the best assignment (fittest chromosome) progresses during the search.The plots show immediately the dramatic improvement obtained using the multilevel paradigm. The performance of MA is unsatisfactory and is getting even far more dramatic for larger problems as the percentage excess over the solution is higher compared to that of MLVMA. The curves show no cross-over implying that MLVMA dominates MA. The plots suggest that problem solving with MLVMA happens in two phases. The first phase which corresponds to the early part of the search, MLVMA behaves as a hill-climbing method. This phase which can be described as a long one, up to 85% of the clauses are satisfied. 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 flips typically leave the best assignment unchanged, and occurs more specifically once the refinement reaches the finest level. Comparing the multilevel version with the single level version, MLVMA is far better than MA, making it the clear leading algorithm. The key success behind the efficiency of MLVMA relies on the multilevel paradigm. MLVMA uses the multilevel paradigm and draw its strength from 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 a 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 allowingMA 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. We turn towards the non-parametric Wilcoxon Rank test in order to test the level of statistical confidence in differences between the mean cost of the two algorithms. The test requires that the absolute values of the differences between the mean costs of the two algorithms are sorted from smallest to largest and these differences are ranked according to absolute magnitude. The sum of the ranks is then formed for the negative and positive differences separately. As the size of the trials increases, the rank sum statistic becomes normal. If the null hypothesis is true, the sum of ranks of the positive differences should be about the same as the sum of the ranks of the negative differences. Using two-tailed ***P*** value (< 0.05), significance performance differences has been observed for all the test problems.

6.2.3 Convergence Rate & Variable to clauses ratio

Figure 7 shows the convergence behavior expressed as the ratio between the best chromosome of the two algorithms as a function of time. The comparison is based on short runs (300 sec). The plot selected three instances as it represents the general

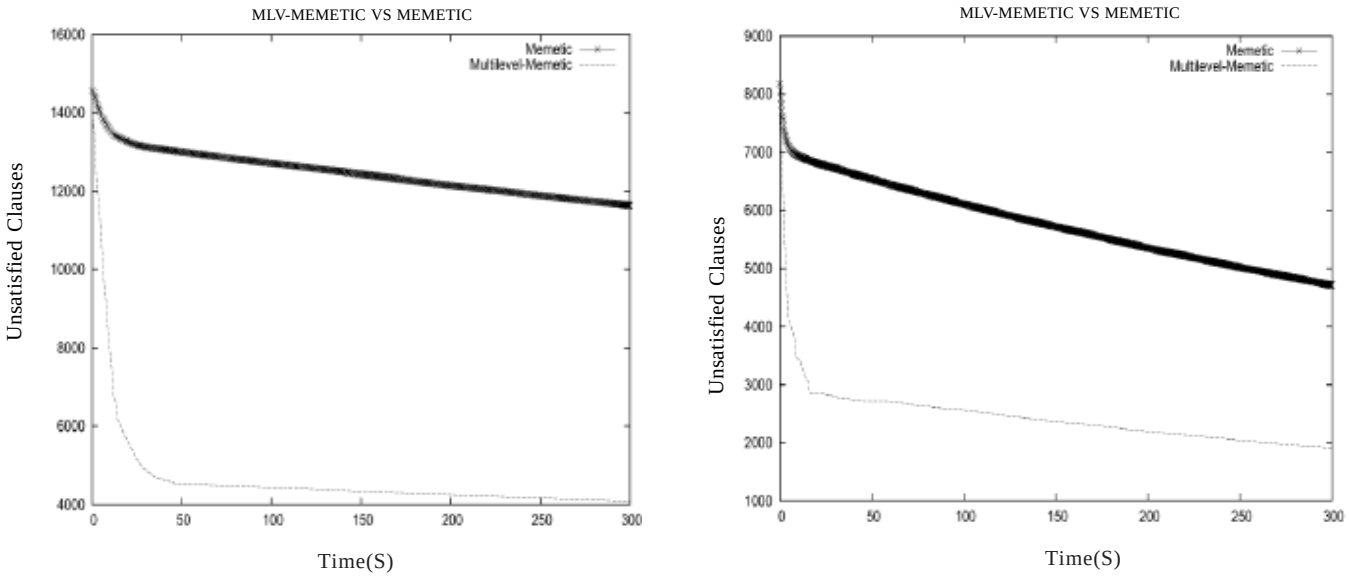

Figure 5. (Left) bmc-ibm-3.cnf: |V | = 14930, |C| = 72106. (Right) bmc-ibm-5: |V | = 9396, |C| = 41207. Along the horizontal axis we give the time in seconds , and along the vertical axis the number of unsatisfied clauses

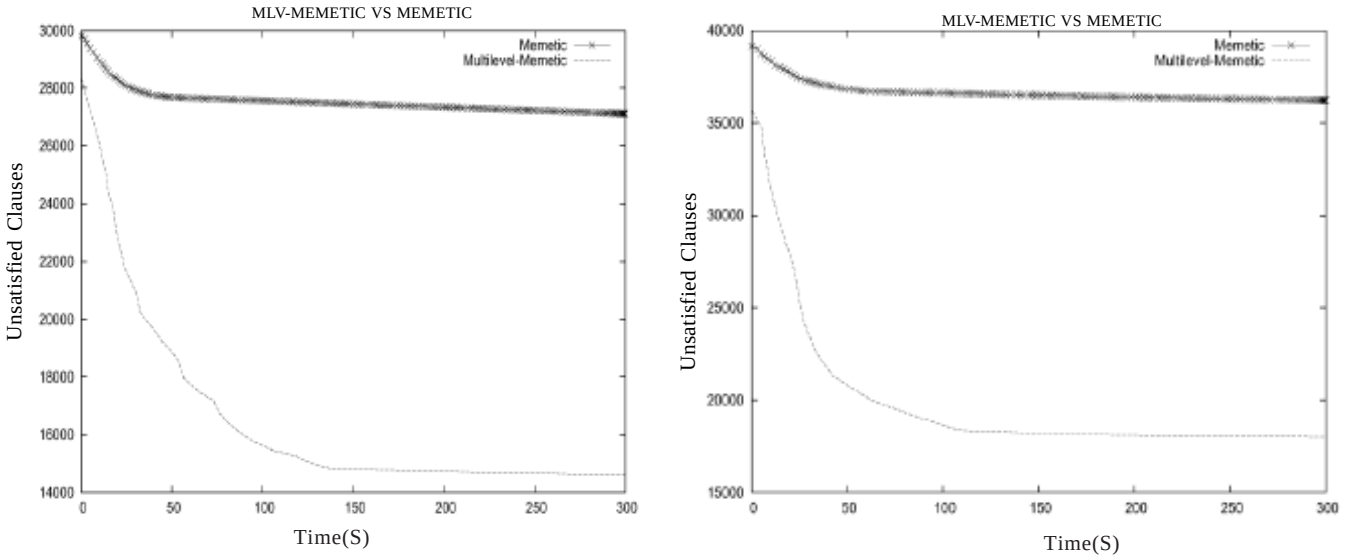

Figure 6. (Left) bmc-ibm-11.cnf: |V | = 32109, |C| = 150027. (Right) bmc-ibm-12.cnf: |V | = 39598, |C| = 19477. Along the horizontal axis we give the time in seconds, and along the vertical axis the number of unsatisfied clauses

trend observed. The curves are below the value 1 leading to conclude that MLVMA is faster compared to MA. The asymptotic performance offered by MLVMA is impressive, and dramatically improves on MA. In some cases, the difference in performance reaches 30% during the first seconds, and maintains it during the whole search process. However, on other cases, the difference in performance continues to increase as the search progresses. Figures 8 and 9 show how the quality of the solution produced by two algorithms varies with the ratio of variables to clauses for BMC and SAT-encoded quasigroup instances. Both algorithms were allowed to run for 1 hour. Due to the small number of instances available, it would be unwise to draw firm conclusions. However some interesting observations can be drawn from these two plots. The first thing to notice is that as the ratio of variables to clauses increases, the percentage of unsatisfied clauses produced by MLVMA remains lower while not showing a substantial variation compared to that of MA. The second thing is the existence of a 'crossover' point at which the difference in the solution quality between the two algorithms is the highest (the point occurs at 2.01 for BMC instances; for SAT-encoded

quasigroups; at 0.01.). More tests are needed in order to investigate if each of the two observed values represents the correct ratio around which the region of hard instances occurs.

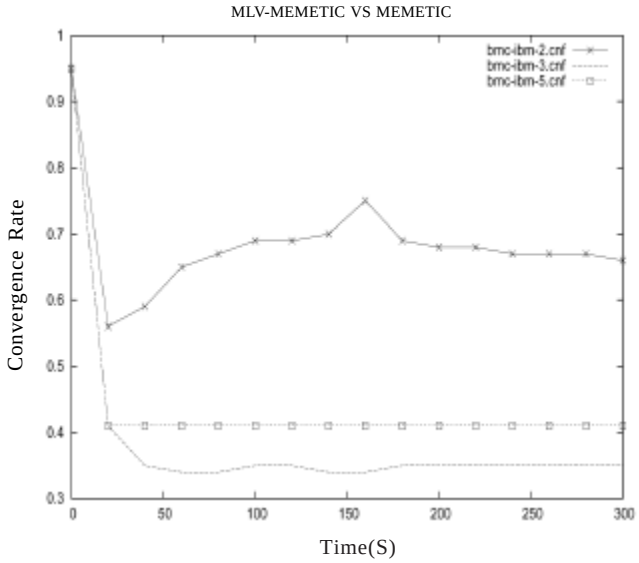

Figure 7. Results on the convergence behavior for bmc-ibm-2.cnf, bmc-ibm-3.cnf, bmc-ibm-5.cnf. Along the horizontal axis we give the time (in seconds) , and along the vertical axis the convergence rate

7. Conclusion

In this work, we have described a new approach for addressing the satisfiability problem. which combines the multilevel paradigm with a simple memetic algorithm. The experiments have shown that MLVMA works quite well with a random coarsening scheme combined with a simple MA used as a refinement algorithm. The broad conclusion that we may draw form the results is that the multilevel framework is able to significantly improve the convergence of MA. It can be seen from the results that MLVMA always returns a better solution compared to MA for the equivalent runtime. A scale up test showed that the difference in performances between the two algorithms increases with larger problems. Finally, MLVMA showed competitive results when compared gainst three state of art algorithms. An obvious subject for further work would be the use of other types of crossover operators which may enhance the performance of MLVMA.

Appendix

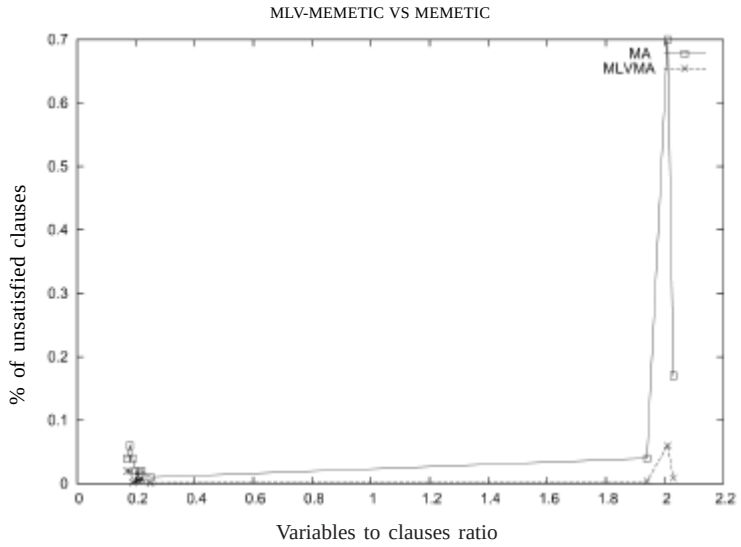

Figure 8. BMC instances: Along the horizontal axis we give the ratio of variables to clauses, and along the vertical axis the percentage of unsatisfied clauses

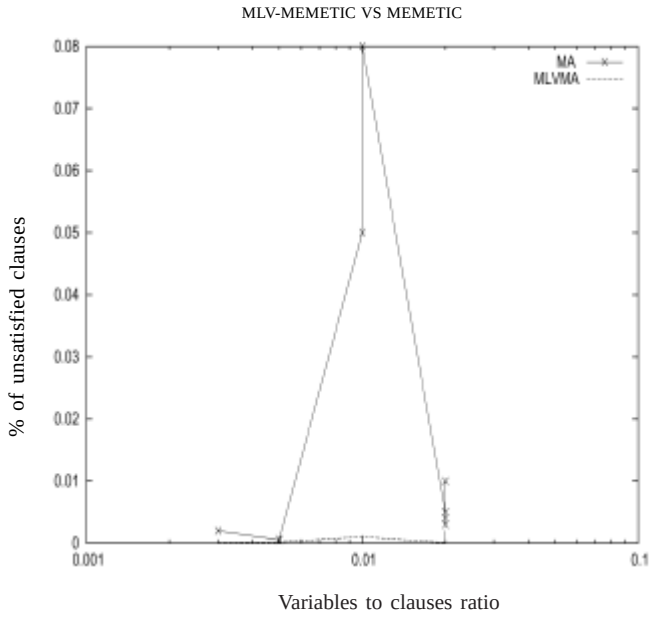

Figure 9. SAT-encoded quasigroup instances : Along the horizontal axis we give the ratio of variables to clauses , and along the vertical axis the percentage of unsatisfied clauses

References

[1] Barnard, S. T., Simon, H. D. (1994). A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. *Concurrency: Practice and Experience*, 6 (2) 101-117.

[2] Biere, A. (2003). Bounded model checking. Advances in Computers.

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

[4] Boughaci, D., Drias, H. (2005). Efficient and experimental meta-heuristics for MAX-SAT problems. *In:* Lecture Notes in Computer Sciences, WEA 2005, V. 3503/2005, p. 501-512.

[5] Boughaci, D., Benhamou, B., Drias, H. (2008). Scatter Search and Genetic Algorithms for MAXSAT Problems. J.Math.Model.Algorithms, p. 101-124.

[6] Cook, S. A. (1971). The complexity of theorem-proving procedures, In: Proceedings of the Third ACM Symposium on Theory of Computing, p. 151-158.

[7] Davis, M., Putnam, H. (1960). A computing procedure for quantification theory, *Journal of the ACM*, 7. 201-215.

[8] Eiben, A. E. and Van der Hauw, J. K. (1997). Solving 3-SAT with Adaptive Genetic Algorithms, *In*: Proceedings of the 4th IEEE Conference on Evolutionary Computation, p. 81-86. IEEE Press.

[9] Gary, M. R., Johnson, D. S. (1979). Computers and intractability: A guide to the theory of NPcompleteness. W.H. Freeman and Company, New York.

[10] Gent, I. and Walsh, T. (1995). Unsatisfied Variables in Local Search. *In*: J. Hallam, editor, Hybrid Problems, Hybrid Solutions, p. 73-85. IOS Press.

[11] Gent, L. P., Walsh, T. (1993). Towards an Understanding of Hill-Climbing Procedures for SAT, *In:* Proceedings of AAAI'93, p. 28-33. MIT Press.

[12] Glover, F. (1989). Tabu Search-Part1. *ORSA Journal on Computing*, 1 (3)190-206.

[13] Gottleib, J., Marchiori, E., Rossi, C. (2002). Evolutionary Algorithms for the satisfiability problem. *Evolutionary Computation*, 10 (1) 35-50.

[14] Granmo, O. C., Bouhmala, C. (2007). Solving the Satisfiability Problem Using Finite Learning Automata. *International Journal of Computer Science and Applications*, 4 (3) 15-29.

[15] Hadany, R.,Harel, D. (1999). A multi-scale algorithm for drawing graphs nicely. Tech.Rep.CS99- 01, Weizmann Inst.Sci., Faculty Maths.Computer Science.

[16] Hansen, P., Jaumand, B.(1990). Algorithms for the Maximum Satisfiability Problem, *Computing*, 44, 279-303.

[17] Hendrickson, B., Leland, R. (1995). A multilevel algorithm for partitioning graphs. *In*: S. Karin, editor, Proc. Supercomputing'95, San Diego, ACM Press, New York.

[18] Hirsch,E. A., Kojevnikov, A (2001). UnitWalk: A new SAT solver that uses local search guided by unit clause elimination. PDMI preprint 9/2001, Steklov Institute of Mathematics at St.Petersburg.

[19] Hoos, H (1999). On the run-time behavior of stochastic local search algorithms for SAT. *In:* Proceedings of AAAI-99, p. 661-666.

[20] Hoos, H. (2002). An adaptive noise mechanism for Walksat. *In:* Proceedings of the Eighteen National Conference in Artificial Intelligence (AAAI-02), p.655-660.

[21] Hutter, F., Tompkins, D., Hoos, H. (2002). Scaling and probabilistic smoothing:Efficient dynamic local search for SAT. *In:* Proceedings of the Eight International Conference of the Principles and Practice of Constraint Programming (CP'02), p. 233-248.

[22] Ishtaiwi, A., Thornton, J., Sattar, A., Pham, D.N. (2005). Neighborhood clause weight redistribution in local search for SAT. *In:* Proceedings of the Eleventh International Conference on Principles and Practice Programming(CP-05), V. 3709 of Lecture Notes in Computer Science, p. 772-776.

[23] Jin-Kao, H., Lardeux, F., Saubion, F. (2003). Evolutionary computing for the satisfiability problem. *In:* Applications of Evolutionary Computing, V. 2611 of LNCS, p. 258-267, University of Essex, England, UK.

[24] Johnson, D.S., Trick, M.A. editors. (2006). Cliques, Coloring, and Satisfiability, V. 26 of DIMACS Series on Discrete Mathematics and Theoretical Computer Science. American Mathematical Society.

[25] Karypis, G., Kumar,V. (1998). A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, 20 (1) 359-392.

[26] Karypis, G., Kumar,V. (1998). Multilevel k-way partitioning scheme for irregular graphs, *J. Par. Dist. Comput.*, 48 (1) 96-129.

[27] Khuda Bukhsh, A. R. Xu, L., Hoos, H. H., Leyton-Brown, K. SATenstein. (2009). Automatically Building Local Search SAT Solvers From Components. *In:* Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI-09), accepted.

[28] Kramer, O. (2010). Iterated local search Powell's method: a memetic algorithm for continuous global optimization. *Memetic Comp.*, 2, p. 69-83.

[29] Lardeux, F., Saubion, F., Jin-Kao. (2006). GASAT: A Genetic Local Search Algorithm for the Satisfiability Problem. *Evolutionary Computation,* 14 (2) MIT Press.

[30] Li, C. M, Wei, W., Zhang, H. (2007). Combining adaptive noise and look-ahead in local search for SAT. *In:*Proceedings of the Tenth International Conference on Theory and Applications of Satisfiability Testing(SAT-07), V. 4501 of Lecture Notes in Computer Science, p. 121-133.

[31] Lozano, M., Martinez, C.G. (2010). Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: Overview and progress report. *Computers and operations Research*, 37, 481-497.

[32] Li, C. M, Huang, W. Q. (2005). Diversification and determinism in local search for satisfiability. *In:* proceedings of the Eight International Conference on Theory and Applications of Satisfiability Testing (SAT-05), V. 3569 of Lecture Notes in Computer Science, p. 158-172.

[33] Li, C. M, Wei, W., Zhang, H. (2007). Combining adaptive noise and look-ahead in local search for SAT. *In:* Proceedings of the Tenth International Conference on Theory and Applications of Satisfiability Testing (SAT-07), V. 4501 of Lecture Notes in Computer Science, p. 121-133.

[34] McAllester, D., Selman, B., Kautz, H. (1997). Evidence for Invariants in Local Search. Proceedings of AAAI'97, p. 321-326. MIT Press.

[35] Moscato, P. A. (1989). On evolution search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Technical Report Caltech Concurrent Computation Program, Caltech, Pasadena, California.

[36] Patterson, D. J, Kautz, H. (2001). Auto-Walksat: A Self-Tuning Implementation of Walksat. Electronic Notes on Discrete Mathematics 9.

[37] Prestwich, S. (2005). Random walk with continuously smoothed variable weights. *In:* Proceedings of the Eight International Conference on Theory and Applications of Satisfiability Testing (SAT-05), V.3569 of Lecture Notes, p. 203-215.

[38] Rodney, D., Soper, A., Walshaw, C. (2007). The application of multilevel refinement to the vehicle routing problem. *In:* D. Fogel et al., editors, *In*: Proc. CISChed , IEEE Symposium on Computational Intelligence in Scheduling, p. 212-219. IEEE.

[39] Schuurmans, D., Southey, F. (2000). Local search characteristics of incomplete SAT procedures. *In:* Proc.AAAI-2000, p. 297-302, AAAI Press.

[40] Schuurmans, D., Southey, F., Holte, R.C. (2001). The exponentiated sub-gradient algorithm for heuristic Boolean programming. *In*: Proc. IJCAI-01, p. 334-341, Morgan Kaufman Publishers.

[41] Selman, B., Levesque, H., Mitchell, D. (1992). A New Method for Solving Hard Satisfiability Problems. *In:* Proceedings of AAA'92, p. 440-446, MIT Press.

[42] Selman, B., Kautz, H. A., Cohen, B. (1994). Noise Strategies for Improving Local Search. *In:* Proceedings of AAAI'94, p. 337-343. MIT Press.

[43] Selman, B., Kautz, H. A. (1993). Domain-Independent extensions to GSAT: Solving large structured satisfiability problems. *In:* R.Bajcsy editor, Proceedings of the international Joint Conference on Artificial Intelligence, V 1, p. 290-295. Morgan Kaufmann Publishers Inc.

[44] Thornton, J., Pham, D. N ., Bain, S., Ferreira Jr, V. (2004). Additive versus multiplicative clause weighting for SAT. Proceedings of the Nineteenth National Conference of Artificial Intelligence (AAAI-04), p. 191-196.

[45] Walshaw, C. (2003). A multilevel algorithm for Forced-Directed Graph-Drawing. *Journal of Graph Algorithms and Applications*, 7 (3) 253-285.

[46] Walshaw, C., Cross, M. (2000). Mesh partitioning: A multilevel balancing and refinement algorithm, *SIAM J. Sci. Comput.*, 22 (1) 63-80.

[47] Walshaw, C. (2002). A multilevel approach to the traveling salesman problem. *Oper. Res.*, 50 (5) 862-877.

[48] Walshaw, C. (2001). A Multilevel Lin-Kernighan-Helsgaun Algorithm for the Traveling Salesman Problem. Tech. Rep. 01/IM/80, Comp. Math. Sci., Univ. Greenwich.

[49] Wang, Y., Li, B. (2010). Multi-strategy ensemble evolutionary algorithm for dynamic multiobjective optimization. Memetic Comp., 2, p. 3-24.

[50] Wu, Z., Wah, B. (2000). An efficient global-search strategy in discrete Lagrangian methods for solving hard satisfiability problems. *In:* Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-00), p. 310-315.

[51] Xu, L., Hutter, F., Hoos, H., Leyton-Brown, K. (2008). SATzilla: Portfolio-based Algorithm Selection for SAT. *Journal of Artificial Intelligence Research (JAIR)*, 32, 565-606.