SINGLE VS HIERARCHICAL POPULATION-BASED MEMETIC ALGORITHM FOR SAT-ENCODED INDUSTRIAL PROBLEMS: A STATISTICAL COMPARISON

Noureddine Bouhmala, Karina Hjelmervik, and Kjell Ivar Øvergård

Department of Maritime Technology and Innovation
Vestfold University College, Norway
{noureddine.bouhmala, karina.hjelmervik, kjell.oevergaard}@hive.no

ABSTRACT

In this work, a hierarchical population-based memetic algorithm for solving the satisfiability problem is presented. The approach suggests looking at the evolution as a hierarchical process evolving from a coarse population where the basic unit of a gene is composed of cluster of variables that represent the problem to a fine population where each gene represents a single variable. The optimization process is carried out by letting the converged population at a child level serve as the initial population to the parent level. A benchmark composed of industrial instances is used to compare the effectiveness of the hierarchical approach against its single-level counterpart.

KEYWORDS

Satisfiability problem, Memetic algorithm, Hierarchical algorithm,

1. INTRODUCTION

Complex optimization problems arise in several areas of artificial intelligence and computer science. In their full generality, these problems are NP-complete and consequently algorithmically intractable. With the growing popularity of artificial intelligence (AI), several researchers have applied AI techniques in extensive various fields such as expert systems, neural network, genetic algorithms, supervised learning methods, Multi-agent systems, and fuzzy set theory to various problems. AI methods have been applied in the field of high energy physics where the goal is to discover the fundamental properties of the physical universe [62], predicting hard drive failures to allow users to back up their data [40]. The field of software engineering turns out to be a fertile ground where many software development tasks could be formulated as learning problems and approached in terms of AI learning algorithms [66]. AI methods have proven to provide better accuracy than statistical methods for the prediction of tumour behaviour [45]. AI techniques have shown their superiority compared to logistic regression when predicting the childhood obesity [65] by over 10%. In the field of logistics, travel-time information is essential to reduce the delivery costs, increase the reliability of delivery, and improve the service quality. Many research studies revealed the good capacity of AI techniques to estimate and predict travel-time. The expected travel time prediction error with AI methods is approximately 4% of practical travel time [36]. Predicting energy production and consumption is an elusive task since it has a major impact to policy and high-stakes decision making. Artificial neural networks were used for the first time to build a predictive model to forecast United States natural gas

DOI : 10.5121/ijaia.2012.3607

production [38]. In recent years, artificial intelligence, in its many integrated flavors from artificial neural networks to memetic algorithms to fuzzy logic, has been making solid steps toward becoming more and more accepted in the field of oil and gas industry such as reservoir characterization [13], production engineering issues [1], and drilling [3].

Memetic algorithms (MAs) algorithms like other metaheuristics 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. MAs are among the most successful hybrid approaches that were proposed to improve the performance of genetic algorithms. While the combination of a population of solutions and genetic operators constitutes the main component of MAs that act as diversification factor on the search space, the use of local search methods helps to quickly identify better solutions in a localized region of the search space. Nevertheless, even MAs may still suffer from either slow or premature convergence [52]. The performance of MAs as well as other available optimization techniques 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.

In this paper, we present a hierarchical population-based memetic algorithm for SAT encoded industrial problems. The concept suggests looking at the evolution of the population as a multilevel process operating in a coarse-to-fine strategy (evolving from a coarse population where each gene is composed of a cluster of variables to a fine population where each gene represents a single variable). The experimental results are provided to demonstrate the efficiency of the hierarchical paradigm at improving the asymptotic convergence of the single level memetic algorithm.

The paper is organized as follows. Section 2 describes the satisfiability problem. Section 3 reviews some of the metaheuristics used with the hierarchical paradigm. Section 4 describes the hierarchical memetic algorithm. In Section 5 we report the experimental results. Finally Section 6 discusses the main conclusions and provides some guidelines for future work.

2. THE SATISFIABILITY PROBLEM

The Satisfiability (SAT) problem which is known to be NP-complete [11] plays a central problem in many applications in the fields of Very Large Scale Integration (VLSI) Computer-Aided design, Computing Theory, and Artificial Intelligence. Generally, an instance of the SAT problem is defined by a set of Boolean variables $V = \{v_1, \ldots, v_n\}$ and a Boolean formula $\Phi : \{0, 1\}^n \rightarrow \{0, 1\}$. The formula Φ is in conjunctive normal form (CNF) if it is a conjunction of clauses. Each clause in turn is a disjunction of literals and a literal is a variable or its negation. 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 [14], 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, and a literal being a Boolean variable or its negation. For example, $P \vee Q$ is a clause containing the two literals P and Q. The clause $P \vee Q$ 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.

3. A SHORT SURVEY OF HIERARCHICAL APPROACHES

Hierarchical approaches are special techniques that aim at producing produce smaller and smaller problems that are easier to solve than the original one. These techniques were first introduced when dealing with the graph partitioning problem (GPP) [4][20][24][30] [31][58] and have proved to be effective in producing high quality solutions at a lower cost than single level techniques. Recently, a memetic algorithm integrating a new hierarchical crossover operator and a perturbation-based tabu algorithm has been introduced in [5] for GPP. Experimental studies showed that the proposed approach performs far better than any of the existing graph partitioning algorithms in terms of solution quality. The traveling salesman problem (TSP) was the second combinatorial optimization problem to which the hierarchical paradigm was applied [59][60] and has shown a clear improvement in the asymptotic convergence of the solution quality. When the hierarchical paradigm was applied to the graph coloring problem [46], the results did not seem to be in line with the general trend observed in GPP and TSP 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 such techniques gave a better global quality to the drawing and is suggested to both accelerate and enhance force drawing placement algorithms [57]. A tabu search operating in a hierarchical context has been developed for the feature selection problem in biomedical data [41]. The empirical results showed that the approach obtained more accurate and stable classification than those obtained by using the other feature selection techniques. The hierarchical paradigm has been combined with the greedy GSAT algorithm [9] for the satisfiability problem and the broad conclusions drawn from this work was that the multilevel context can either speed up GSAT or improve its asymptotic convergence. A recent survey over existing hierarchical techniques can be found in [61][6][43].

4. THE HIERARCHICAL APPROACH

4.1. General Strategy

The hierarchical strategy involves recursive coarsening to create a hierarchy of increasingly smaller and coarser versions of the original problem. The reduction phase works by grouping the variables representing the problem into clusters. This phase is repeated until the size of the smallest problem falls below a specified reduction threshold. Then, a solution for the problem at the coarsest level is generated, and then successively projected back onto each of the intermediate levels in reverse order. The solution at each child level is improved before moving to the parent level. The hierarchical memetic algorithm is described in Algorithm 1.

4.2. Reduction Phase

Let P_0 (the subscript represents the level of problem scale) be the set of literals. The next coarser level P_1 is constructed from P_0 by merging literals. The merging is computed using a randomized algorithm similar to [24]. The literals are visited in a random order. If a literal l_i has not been matched yet, then a randomly unmatched literal l_j is selected, 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 reduction process until the size of the problem reaches some desired threshold (lines 3-5 of Alg.1).

Algorithm 1: The Hierarchical Memetic 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) = MA-Refinement (P_level);
 9       while (level > 0) do
10           S_start(P_level-1): = Project (S_final (P_level)) ;
11           S_final (P_level-1) := MA-Refinement (S_start (P_level-1)) ;
12           level := level - 1 ;
13   End
```

4.3. Initial Solution

The reduction phase ceases when the problem size shrinks to a desired threshold. Initialization is then trivial and consists of generating an initial solution for the population of the problem (P_m) using a random procedure. The clusters of every individual in the population are assigned the value of true or false in a random manner (line 7 of Alg. 1).

4.4. Projection Phase

The Projection phase refers to the inverse process followed during the reduction phase. Having improved the assignment on $level_{m+1}$, the assignment must be extended on is parent $level_m$. The extension algorithm is simple; if a cluster $c_i \in S_{m+1}$ is assigned the value of true then the merged pair of clusters that it represents, c_j, $c_k \in S_m$ are also assigned the true value (line 10 of Alg.1).

4.5. Improvement Phase

The idea behind the improvement phase is to use the projected population at $level_{m+1}$ as the initial population for $level_m$ for further refinement using MA described in the next subsections. Even though the population at the levelm+1 is at a local minimum, the projected population may not be at a local optimum with respect to $level_m$. 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 (line 8,11 of Alg.1). As soon as the population tends to lose its diversity, premature convergence occurs and all individuals in the population tend to be identical with almost the same fitness value. During each level, the memetic algorithm is assumed to reach convergence when no further improvement of the best solution has been made during five consecutive generations.

4.5.1 Memetic Algorithms (MAs)

Memetic algorithms (MAs) represent the set of hybrid algorithms that combine Genetic algorithms (GAs) and local search. In general the genetic algorithm improves the solution while the local search fine-tunes the solution. They are adaptive based search optimizations algorithms that take their inspiration from genetics and evolution process. Memetic algorithms simultaneously examine and manipulate a set of possible solution. Given a specific problem to solve, the input 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 that 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 from the population to be parents and enters them in a mating pool. Parent individuals are drawn from the mating pool and combined so that information is exchanged and passed to offsprings 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. Finally, a selection scheme is used to update the population giving rise to a new generation. The individuals from the set of solutions that is called population will evolve from 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.

4.5.2. Implementation Issues

- **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 function used by our memetic algorithm is simply the minimization of the number of unsatisfied clauses.

- **Representation:**

  A representation is a mapping from the state space of possible solutions to a state of encoded solutions within a particular data structure. Given a set S_m at level m and an integer l, an individual I of length l corresponds to a truth assignment of S_m such that all the literals that make the cluster C_i, $i \in \{1, \ldots, l\}$ are assigned the same value as C_i. 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 $S = \{0, 1\}^n$.

- **Initial population:**

  The initial population consists of individuals generated randomly in which each gene's allele is assigned randomly the value 0 or 1.

- **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 two-point 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 offsprings). The creation of these descendants involves the location and combinations of features extracted from the parents. The reason behind choosing the two-point crossover, are the results presented in [56] where the difference between the different crossovers are not significant when the problem to be solved is hard. The work conducted in [53] shows that the two-point crossover is more effective when the problem at hand is difficult to solve.

- **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 individuals 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 $C = c_1, c_2,\ldots, c_m$ 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 from being trapped into local optima while adding to the diversity of the population and thereby increasing the likelihood that the algorithm will generate individuals with better fitness values.

- **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, it determines the next population. In the roulette method, the selection is stochastic and biased towards 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 $P_{\text{Selection}i} = f_i / \sum_1^N f_i$.

Algorithm 2: local-refinement

```
input:  Chromosome_i
output: A possibly improved Chromosome_i
1 begin
2     PossFlips ← a randomly selected variable with the largest decrease (or smallest
      increase) in unsatisfied clauses ;
3     v ← Pick (PossFlips); Chromosome_i ← Chromosome_i with v flipped ;
4     if Chromosome_i satisfies Φ then
5         return Chromosome_i;
6 end
```

- **Local search:**

Finally, the last component for MAs is the use of local improvers. Algorithm 2 shows the pseudo-code of the local search. 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 the local search component (exploitation). Bearing this thought in mind, the strategy adopted in this regard is to let each chromosome go through a low rate intensity local improvement. A fast and simple heuristic is used for one iteration during which it seeks for the new variable-value assignment which best decreases the numbers of unsatisfied clauses is identified.

5. EXPERIMENTAL RESULT

5.1. Benchmark Instances

We evaluated the performance of the hierarchical memetic algorithm (MLVMA) against its single variant (MA) using a set of instances taken from real industrial problems. This set is taken from the SATLIB website (http://www.informatik.tu-darmstadt.de/AI/SATLIB).

instance	number of variables	number of clauses
$2\text{bitadd}_{10}.cnf$	590	1422
$2\text{bitadd}_{11}.cnf$	649	1562
$2\text{bitadd}_{12}.cnf$	708	1702
$2\text{bitcomp}_{5}.cnf$	125	310
$2\text{bitmax}_{6}.cnf$	252	766
$3\text{bitadd}_{31}.cnf$	8432	31310
$3\text{bitadd}_{32}.cnf$	8704	32316
3block.cnf	283	9690
4blocks.cnf	758	47820
4blocksb.cnf	410	24758
e0ddr2-10-by-5-1.cnf	19500	103887
e0ddr2-10-by-5-4.cnf	1728	104527
enddr2-10-by-5-1.cnf	20700	111567
enddr2-10-by-5-8.cnf	21000	113729
ewddr2-10-by-5-1.cnf	21800	118607
ewddr2-10-by-5-8.cnf	22500	123329

Table 1. Benchmark set of the SAT Competition Beijing

Table 1 show 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 cut–off parameter (max–time) set to 15 minute. 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 [2].

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 following parameters have been fixed experimentally and are listed below:

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

6. RESULTS

6.1. Observed Trends

The time development of the hierarchical memetic algorithm against its single variant in solving the instances is shown in Figs. 1–8. The plots show the 100 runs of both algorithms with a cut-off at 15 minutes as well as the mean of these runs. The curves suggest that problem solving with the hierarchical variant 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 reach the plateau region, marking the start of the second phase. The plateau region 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. It is clear from the plots that MLVMA offers a better asymptotic convergence compared to MA especially for large instances. The instances where both algorithms reach approximately the same solution quality (with MLVMA being marginally better), the hierarchical variant offers a cost effective solution strategy considering the amount of time required. In our view, the efficiency of the hierarchical approach relies on coupling the refinement process across different levels. This paradigm offers two main advantages which enables the memetic algorithm to become much more powerful in the hierarchical context. During the refinement phase the memetic algorithm applies a local transformation (i.e. a move) within the neighbourhood (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 the gene of each individual representing a cluster of variables at different levels, 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 neighbourhood becomes adaptive and allows the possibility for both cross-over and mutation operators 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.

6.2. Statistical Analysis

The results showing the statistical comparison of the two algorithms are presented in Table 2. We report the mean (M) and the standard deviation (SD) of unsolved clauses for the MLVMA and MA algorithms. The range of solutions from each algorithm is also presented in order to show the overlap between solution spaces for any given instance. Statistical inferential analysis was done with an independent samples t-test that 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 $\hat{A}_{12}$ [55] was used to evaluate the relative dominance of one algorithm over the other. The $\hat{A}_{12}$ effect size measure is calculated using the rank sum which

Table 2: Comparison of multi-level (MLVMA) and single-level (MA) memetic algorithms

Instance	MLVMA		MA						
	M (*SD*)	Range	*M* (*SD*)	Range	*M* diff.	95% *CI* of *M* diff.	*p*	Obs. $\hat{A}_{12}$	A12 [95% *CI* of $\hat{A}_{12}$][a]
2bitadd_10	2.0 (.7]	[1-3]	16.3 (2.8)	[11-25]	14.4	[13.8, 14.9]	***	1	c
2bitadd_11	1.7 (.7)	[1-4]	16.3 (3.2)	[8-24]	14.5	[13.9, 15.2]	***	1	c
2bitadd_12	1.5 (.7)	[1-3]	1.6 (.7)	[1-4]	0.1	[-0.1, 0.3]	.247	.548	.547 [.476, .622]
2bitcomp_5[b]	1.0 (0)		1.0 (0)		0			.5	c
2bitmax_6	1.0 (.2)	[1-2]	1.0 (0.1)	[1-2]	0	[-0.1, 0.3]	.653	.495	.495 [.475, .515]
3bitadd_31	132.6 (10.9)	[122-216]	1106.2 (142.1)	[923-2620]	973.6	[945.5, 1001.7]	***	1	c
3bitadd_32	135.7 (11.9)	[123-186]	1366.9 (179.1)	[1125-1974]	1231.2	[1195.8, 1266.6]	***	1	c
3blocks	4.0 (1.8)	[2-9]	7.2 (1.0)	[4-9]	3.2	[2.8, 3.6]	***	.918	.920 [.877, .958]
4blocks	8.2 (3.1)	[2-14]	13.0 (1.0)	[11-18]	4.8	[4.1, 5.4]	***	.916	.917 [.878, .953]
4blocksb	5.2 (1.8)	[2-8]	7.3 (0.7)	[5-8]	2.1	[1.7, 2.4]	***	.847	.840 [.780, .891]
e0ddr2-10-by-5-1	343.4 (119.0)	[261-697]	10871.1 (324.5)	[9895-11527]	10527.7	[10459.5, 10595.8]	***	1	c
e0ddr2-10-by-5-4	320.6 (80.8)	[271-718]	10969.1 (360.1)	[10190-11784]	10648.5	[10575.8, 10721.3]	***	1	c
enddr2-10-by-5-1	371.9 (144.0)	[281-1021]	12042.9 (378.1)	[11064-12879]	11671.0	[11591.2, 11750.8]	***	1	c
enddr2-10-by-5-8	358.9 (136.1)	[278-967]	12241.3 (400.0)	[11169-13446]	11882.4	[11799.1, 11965.7]	***	1	c
ewddr2-10-by-5-1	399.8 (166.9)	[289-1124]	12939.7 (407.9)	[11960-13835]	12539.9	[12453.0, 12626.8]	***	1	c
ewddr2-10-by-5-8	354.6 (107.0)	[293-710]	13537.5 (423.8)	[12393-14736]	13182.9	[13096.7, 13269.1]	***	1	c

Note: *M* = Mean, *SD* = Standard Deviation, *M* diff = Mean Difference, *CI* = Confidence Interval, Obs. = Observed Value, *p* = *p*-value.
[a] Based upon 10000 bootstrapped samples, deviations from bootstrapped $\hat{A}_{12}$ is due to stochastic variance. [b] All runs found an optimal solution, for this instance, hence no inferential statistics is computed. [c] Effect size $\hat{A}_{12}$ cannot be computed because there is no overlap between MLVMA and MA for these instances. *** = $p < .001$.

is a common component in any non-parametric analysis such as the Mann-Whitney U-test [2]. Calculating $\hat{A}_{12}$ is done according to the following formula

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

where R1 is the rank sum of algorithm MLVMA, m is the number of observations in the first data sample, and n is the number of observations in the second data sample. Calculating $\hat{A}_{12}$ results in a number between 0 and 1 that represent the probability that MLVMA will yield a better solution than MA. If the two algorithms are equivalent then $\hat{A}_{12} = .5$, while a complete dominance of algorithm MLVMA over MA would entail $\hat{A}_{12} = 1$.

$\hat{A}_{12}$ is more easily interpreted than the more common parametric Cohen's *d* [10] that 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 [2]. Second, when dealing with solutions to optimization problems, a researcher or 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 $\hat{A}_{12}$ shown in Table 2 (where applicable) are calculated using a bootstrapping procedure [39] that is used to estimate the 95% confidence interval of $\hat{A}_{12}$. 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 Â12 with the equation 1. The three steps are then repeated 1000 times and the resulting statistic Â12 is saved to create a sampling distribution of the statistic Â12

The results indicate that MLVMA is always better than MA in 10 out of the 16 instances. MLVMA dominates MA in three instances (for the 3blocks, 4blocks and 4blocksb-instances, $\hat{A}_{12}$ 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 after 900 seconds run time. 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 ($t_{(15)} = 3.059$, $b = 2.041^{-8}$, 95% *CI* [1.163^{-8}, 2.714^{-8}], $p = .008$, $r = .633$) than for the MA ($t_{(15)} = 10.067$, $b = 9.341^{-7}$, 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 memetic algorithm (see Figure 9 for a graphical representation).

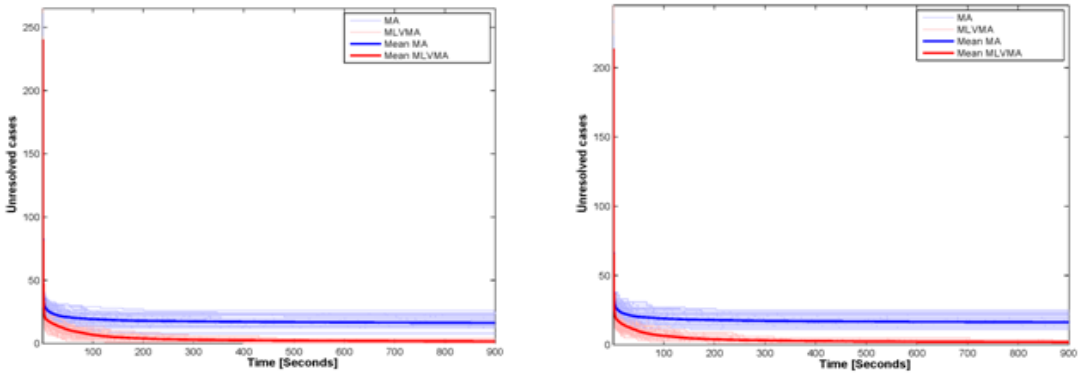

Figure 1:
MLVMA Vs MA: (left) 2bitadd10.cnf, (right) 2bitadd11.cnf - Time development for 100 runs in 15 minutes.

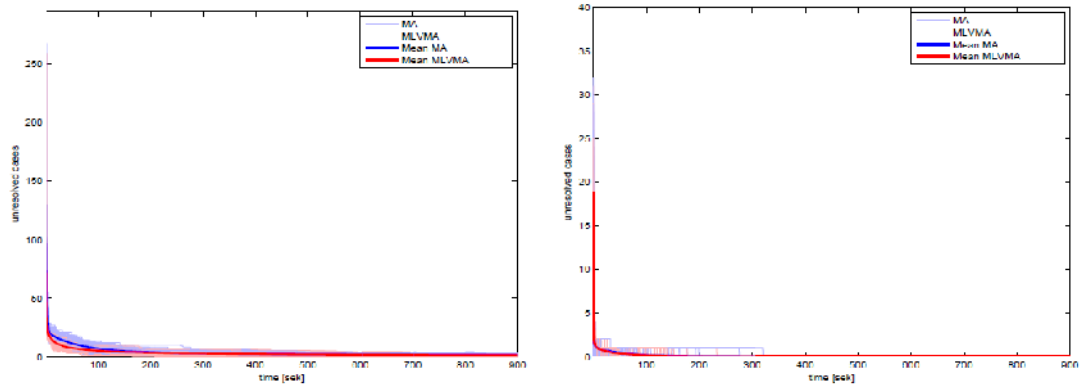

Figure 2:
MLVMA Vs MA: (left) 2bitadd12.cnf,(right) 2bitcomp5.cnf - Time development for 100 runs in 15 minutes.

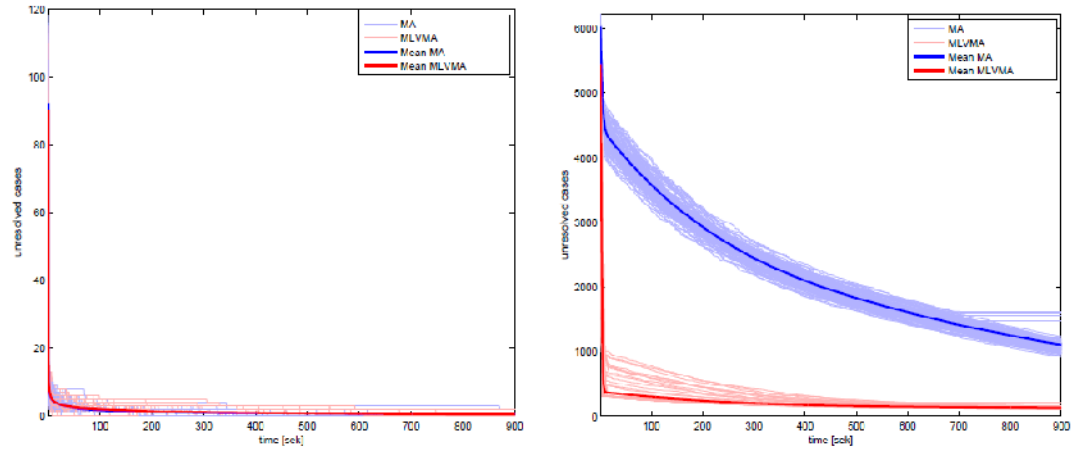

Figure 3:
MLVMA Vs MA: (left) 2bitmax6.cnf,(right) 3bitadd31.cnf - Time development for 100 runs in 15 minutes.

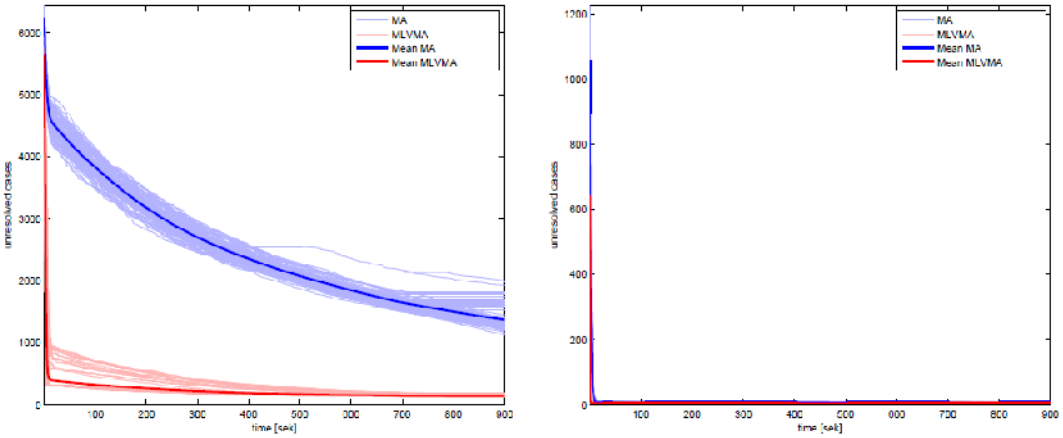

Figure 4:
MLVMA Vs MA: (left) 3bitadd32.cnf, (right) 3block.cnf - Time development for 100 runs in 15 minutes.

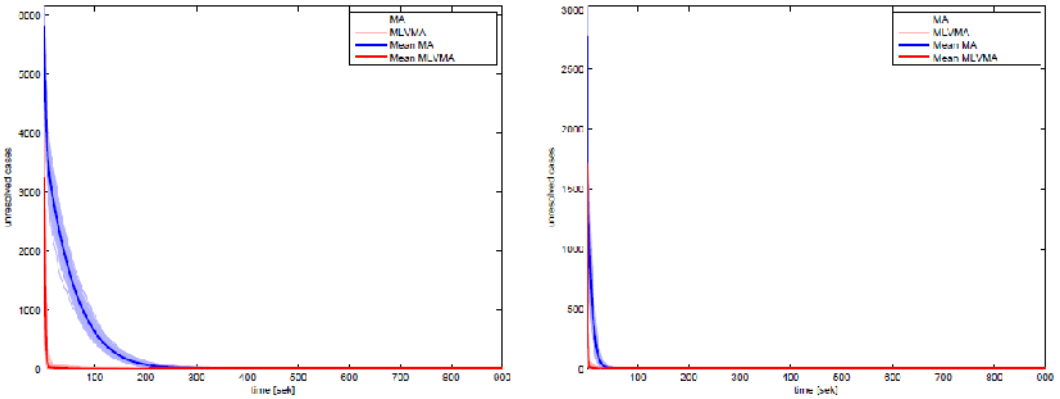

Figure 5:
MLVMA Vs MA: (left) 4blocks.cnf, (right) 4blocksb.cnf - Time development for 100 runs in 15 minutes.

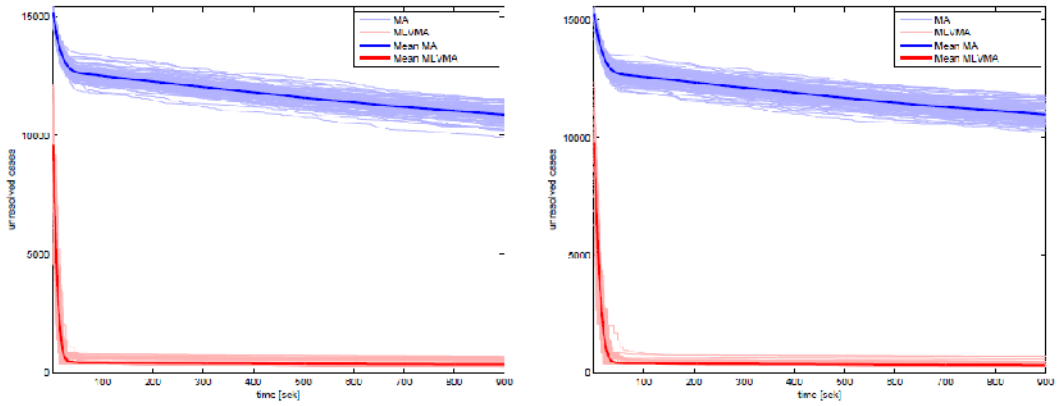

Figure 6:
MLVMA Vs MA: (left) e0ddr2-10-by-5-1.cnf, (right) e0ddr2-10-by-5-4.cnf - Time development for 100 runs in 15 minutes.

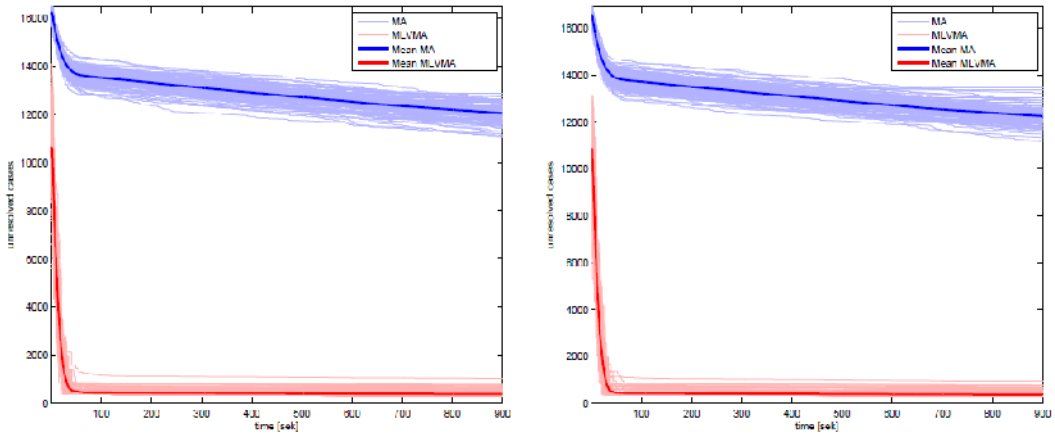

Figure 7:
MLVMA Vs MA: (left) enddr2-10-by-5-1.cnf, (right) enddr2-10-by-5-8.cnf- Time development for 100 runs in 15 minutes.

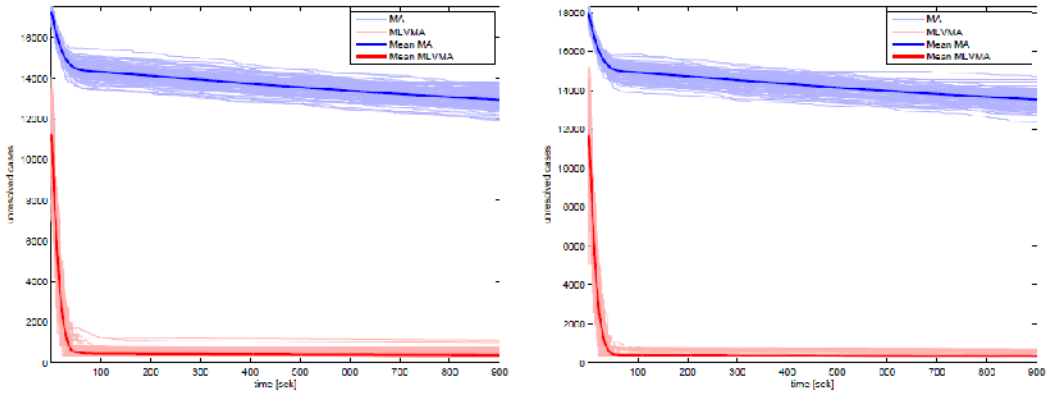

Figure 8:
MLVMA Vs MA: (left) ewddr2-10-by-5-1.cnf, (right) ewddr2-10-by-5-8.cnf - Time development for 100 runs in 15 minutes.

7. CONCLUSION

A new approach for solving the satisfiability problem based on a hierarchical memetic algorithm is presented. The approach permits the population to go through several optimization levels, where the converged population at the previous level will serve as the starting population for the next level. A set of industrial instances was used to get a comprehensive picture of the performance of the new approach. The results obtained ensure that the hierarchical paradigm greatly improves MA and always returns a better solution for the equivalent runtime compared to MA. Our future work aims at investigating other reduction schemes and identifying other parameters that may influence the interaction between the memetic algorithm and the hierarchical paradigm.

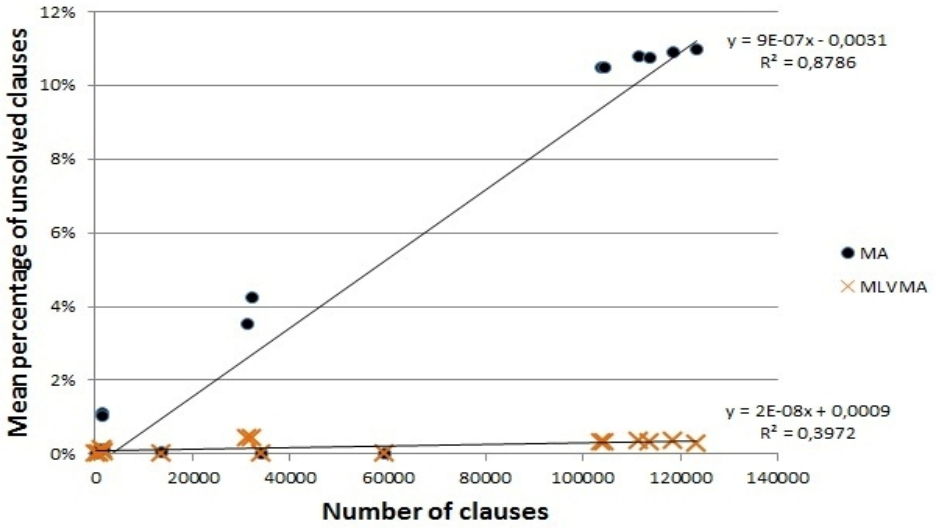

Figure 9: Scatterplot with regression lines showing the relationship between number of clauses per instance for the multi-level (MLVMA) and the single-level (MA) algorithms.

REFERENCES

[1] C. Alimonti and G. Falcone, (2002) "Knowledge Discovery in Databases and Multiphase Flow Metering: The Integration of Statistics, Data Mining, Neural Networks, Fuzzy Logic, and Ad Hoc Flow Measurements Towards Well Monitoring and Diagnosis", SPE 77407, Proc., SPE Annu. Technical Conf. and Expo. held in San Antonio, Texas.

[2] A. Arcuri and L. Briand, (2011) "A Hitchhiker's Guide to Statistical Tests for Assessing Randomized Algorithms in Software Engineering", Technical report, Simula research lab- oratory, number 13/2011.

[3] R. S. Balch, D. M. Hart, W. W. Weiss, and R. F. Broadhead, (2002) "Regional Data Analysis to Better Predict Drilling Success. "Brushy Canyon Formation, Delaware Basin, New Mexico. SPE 75145, Proc., SPE/DOE Improved Oil Recovery Symp. held in Tulsa, Oklahoma.

[4] S.T. Barnard and H.D. Simon, (1994) "A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems", Concurrency: Practice and Experience, Vol. 6, No. 2, pp101-117.

[5] U. Benlik and J-K. Hao, (2011) "A Multilevel Memetic Approach for Improving Graph k- Partitions". IEEE Trans. Evol. Comput., , vol.15, no.5, pp624-642.

[6] C. Blum, J. Puchinger, G. R. Raidl, and A. Roli, (2011) "Hybrid metaheuristics in combinatorial optimization: A survey", Applied Soft Computing, Vol. 11, pp4135-4151.

[7] D. Boughaci, B. Benhamou, and H. Drias, (2008) "Scatter Search and Genetic Algorithms for MAX-SAT Problems", J. Math. Model. Algorithms, Vol. 7, No. 2, pp101-124.

[8] D. Boughaci and H. Drias, (2005) "Efficient and experimental meta-heuristics for MAX-SAT problems", In S. E. Nikoletseas (ed.) Lecture Notes in Computer Sciences, Experimental and Efficient Algorithms, vol. 3503, pp501-512.

[9] N. Bouhmala and O.C. Granmo, (2011) "GSAT Enhanced with Learning Automata and Multilevel Paradigm", International Journal of Computer Science Issues, Vol. 8, No. 3., pp.38-54.

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

[11] S.A. Cook, (1971) "The complexity of theorem-proving procedures", Proc. of the 3rd ACM Symp. on Theory of Computing, pp151-158.

[12] B. Efron, (1982) "The Jacknife, the Bootstrap and Other Resampling Plans", Philadelphia, PA: Society for Industrial and Applied Mathematics.

[13] J. Finol., C. Romero., and P. Romero, (2002) ”An Intelligent Identification Method of Fuzzy Models and Its Applications to Inversion of NMR Logging Data”, SPE 77605, Proc., SPE Annu. Technical Conf. and Exhibition held in San Antonio.
[14] M. R. Gary and D. S. Johnson, (1979) ”Computers and intractability: A guide to the theory of NP-completeness”, W.H. Freeman and Company, New York.
[15] I. Gent and T. Walsh, (1995) ”Unsatisfied Variables in Local Search”, In J. Hallam, (ed.), Hybrid Problems, Hybrid Solutions, pp73-85, IOS Press.
[16] L.P. Gent and T. Walsh, (1993) ”Towards an Understanding of Hill-Climbing Procedures for SAT”, Proc. of AAAI’93, pp.28-33, MIT Press.
[17] F. Glover, (1989) ”Tabu Search-Part 1”, ORSA Journal on Computing, Vol. 1, No. 3, pp190-206.
[18] J. Gottleib, E. Marchiori, and C. Rossi, (2002) ”Evolutionary Algorithms for the satisfiability problem”, Evolutionary Computation, Vol. 10, No. 1, pp35-50.
[19] O.C. Granmo and N. Bouhmala, (2007) ”Solving the Satisfiability Problem Using Finite Learning Automata”, International Journal of Computer Science and Applications, Vol. 4, No. 3, pp15-29.
[20] R. Hadany and D. Harel, (2001) ”A multi-scale algorithm for drawing graphs nicely”, Discrete Applied Mathematics, Vol. 113, No. 1, pp3-21.
[21] P. Hansen and B. Jaumand, (1990) ”Algorithms for the Maximum Satisfiability Problem”, Computing, Vol. 44, pp279-303.
[22] K. R. Hassnain, (2011) ”Soft Computing Approaches to DPLL SAT Solver Optimization”, Phd Thesis, TU Darmstadt.
[23] J. S. Haukoos and R. J. Lewis, (2005) “Advanced statistics: Bootstrapping confidence intervals for statistics with difficult distributions”, Academy of Emergency Medicine, Vol. 12, No. 4, pp360-365.
[24] B. Hendrickson and R. Leland, (1995) ”A multilevel algorithm for partitioning graphs”, In S. Karin (ed.), Proc. Supercomputing’95, San Diego.
[25] H. Hoos, (1999) ”On the run-time behavior of stochastic local search algorithms for SAT”, Proc. of AAAI-99, pp661-666.
[26] H. Hoos, (2002) “An adaptive noise mechanism for Walksat”, Proc. of the 18th Nat. Conf. in Artificial Intelligence (AAAI-02), pp655-660.
[27] F. Hutter, D. Tompkins, and H. Hoos, (2002) ”Scaling and probabilistic smoothing: Efficient dynamic local search for SAT”, In P. van Hentenryck (ed.) Lecture notes in computer science: Principles and Practice of Constraint Programming, vol. 2470, pp233-248.
[28] A. Ishtaiwi, J. Thornton, A. Sattar, and D. N. Pham, (2005) ”Neighborhood clause weight re-distribution in local search for SAT”, In P. van Beek (ed.) Lecture Notes in Computer Science: Principles and Practice of Constraint Programming, vol. 3709, pp772-776.
[29] H. Jin-Kao, F. Lardeux, and F. Saubion, (2003) ”Evolutionary computing for the satisfiability problem”, In G. R. Raidl et al., (Eds.), Lecture Notes in Computer Science: Applications of Evolutionary Computing, vol. 2611, pp258-267.
[30] G. Karypis and V. Kumar, (1998) ”A fast and high quality multilevel scheme for partitioning irregular graphs”, SIAM J. Sci. Comput., Vol. 20, No. 1, pp359-392.
[31] G. Karypis and V. Kumar, (1998) ”Multilevel k-way partitioning scheme for irregular graphs”, J. Par. Dist. Comput., Vol. 48, No. 1, pp96-129.
[32] A. R. KhudaBukhsh, L. Xu, H. H. Hoos, and K. Leyton-Brown, (2009) ”SATenstein: Automatically Building Local Search SAT Solvers From Components”, Proc. 25th Intl. Joint Conf. Artificial Intelligence (IJCAI-09).
[33] F. Lardeux, F. Saubion, and Jin-Kao, (2006) ”GASAT: A Genetic Local Search Algorithm for the Satisfiability Problem”, Evolutionary Computation, Vol. 14, No. 2, pp223-253.
[34] C. M. Li, W. Wei, and H. Zhang, (2007) ”Combining adaptive noise and look-ahead in local search for SAT”, In J. Marques-Silva and K.A. Sakallah (eds.), Lecture Notes in Computer Science: Theory and Applications of Satisfiability Testing (SAT-07), vol. 4501, pp121-133.
[35] C. M. Li, and W. Q. Huang, (2005) ”Diversification and determinism in local search for satisfiability”, In F. Bacchus and T. Walsh (eds.), Lecture Notes in Computer Science: Theory and Applications of Satisfiability Testing (SAT-05), vol. 3569, pp158-172.
[36] H. Lin, M. Taylor, and R. Zito, (2005) ”A Review Of Travel-Time Prediction in Transport and Logistics”, Proc. of the Eastern Asia Society for Transportation Studies, Vol. 5, pp1433-1448.
[37] D. McAllester, B. Selman, and H.Kautz(1997), ”Evidence for Invariants in Local Search”, Proc. of AAAI’97, pp321-326, MIT Press.

[38] S.D. Mohaghegh, (2000) ”Virtual Intelligence Applications in Petroleum Engineering: Part 1-Artificial Neural Networks”, Journal of Petroleum Technology, Distinguished Author Series, pp64-73.
[39] C. Z. Mooney and R. D. Duval, (1993) ”Bootstrapping - A Nonparametric Approach to Statistical Inference”, Sage University Press.
[40] J. F. Murray, G. F. Huges, and K. K. Delgado, (2005) ”Machine Learning Methods for Predicting Failures in Hard Drives: A Multiple-Instance Application”, Journal of Machine Learning Research, Vol. 6, pp783-816.
[41] I. O. Oduntan, M. Toulouse, R. Baumgartner, C. Bowman, R. Somorjai, and T. G. Crainic, (2008) ”A multilevel tabu search algorithm for the feature selection problem in biomedical data”, Computers & Mathematics with Applications, Vol.55, No. 5, pp1019-1033.
[42] D. J. Patterson and H. Kautz, (2001) ”Auto-Walksat: A Self-Tuning Implementation of Walk- sat”, Electronic Notes on Discrete Mathematics, Vol. 9, pp360-368.
[43] S. Pirkwieser and G. R. Raidl, (2010) ”Multilevel variable neighborhood search for periodic routing problems”,. In. P. I. Cowling and P. Merz, (Eds.), Lecture Notes in Computer Science: Evolutionary Computation in Combinatorial Optimization, Vol. 6022 of pp226-238, Springer-Verlag, Berlin, Germany.
[44] S. Prestwich, (2005) ”Random walk with continuously smoothed variable weights”, In F. Bacchus and T. Walsh (eds.), Lecture Notes in Computer Science: Theory and Applications of Satisfiability Testing (SAT-05), vol. 3569, pp203-215.
[45] K. N. Qureshi, R. N. G. Naguib, F. C. Hamdy, D. E. Neal and J. K. Mellon, (2000) ”Neural network analysis of clinicopathological and molecular markers in bladder cancer”, Journal of Urology, Vol. 163, pp630-633.
[46] D. Rodney, A. Soper, and C. Walshaw, (2007) ”The application of multilevel refinement to the vehicle routing problem”, In D. Fogel et al., eds, Proc. CISChed 2007, IEEE Symp. Computational Intell. Scheduling, pp212-219.
[47] D. Schuurmans, and F. Southey, (2000) ”Local search characteristics of incomplete SAT procedures”, In Proc.AAAI-2000, pp297-302.
[48] D. Schuurmans, F. Southey, and R. C. Holte, (2001) ”The exponentiated sub-gradient algorithm for heuristic Boolean programming”, In Proc. IJCAI-01, pp334-341.
[49] B. Selman, H. Levesque, and D. Mitchell, (1992)” A New Method for Solving Hard Satisfiability Problems”, In Proc. of AAA’92, pp440-446.
[50] B. Selman, H. A. Kautz, and B. Cohen, (1994) ”Noise Strategies for Improving Local Search”, Proc. of AAAI’94, pp337-343, MIT Press.
[51] B. Selman, and H. A . Kautz, (1993) ”Domain-Independent extensions to GSAT: Solving large structured satisfiability problems”, In R.Bajcsy, (ed.), Proc. of the Intl. Joint Conf. on Artificial Intell., pp290-295.
[52] W. Sheng, X. Liu, and M. Fairhurst, (2008) ”A Niching Memetic Algorithm for Simultaneous Clustering and Feature Selection”, IEEE Trans. on Knowledge And Data Engineering, Vol.20, No.7., pp868-879.
[53] W. Spears, (1995) ”Adapting Crossover in Evolutionary Algorithms”, Proc 4th Annu. Conf. Evol. Programming, pp367-384, MIT Press,
[54] J. Thornton, D. N. Pham, S. Bain, and V. Ferreira Jr., (2004) ”Additive versus multiplicative clause weighting for SAT”, Proc. of the 19th Nat. Conf. of Artificial Intell. (AAAI-04), pp191-196.
[55] A. Vargha, 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, Vol. 25, No. 2, pp101-132.
[56] D. Vrajitoru, (1999) ”Genetic programming operators applied to genetic algorithms”, In Proc. Genetic and Evol. Computation Conf., pp686-693, Orlando, FL: Morgan Kaufmann Publishers.
[57] C. Walshaw, (2003) ”A multilevel algorithm for Forced-Directed Graph-Drawing”, Journal of Graph Algorithms and Applications, Vol. 7, No. 3, pp253-285.
[58] C. Walshaw and M. Cross, (2000) ”Mesh partitioning: A multilevel balancing and refinement algorithm”, SIAM J. Sci. Comput., Vol. 22, No. 1, pp63-80.
[59] C. Walshaw, (2002) ”A multilevel approach to the traveling salesman problem”, Oper. Res., Vol. 50, No. 5, pp862-877.
[60] C. Walshaw, (2001) ”A Multilevel Lin-Kernighan-Helsgaun Algorithm for the Travel-ling Salesman Problem”, Tech. Rep. 01/IM/80, Comp. Math. Sci., Univ. Greenwich.

[61] C. Walshaw, (2008) ”Multilevel Refinement for Combinatorial Optimization: Boosting Metaheuristic Performance”, In C. Blum et al., Hybrid Metaheuristics: Studies in Computational Heuristics, pp261-289, Springer Verlag, Berlin, Germany.

[62] S. Whiteson and D. Whiteson, (2009) ”Machine learning for event selection in high energy physics”, Engineering Applications of Artificial Intelligence, Vol. 22, pp1203-1217.

[63] Z.Wu., and B. Wah, (2000) ”An efficient global-search strategy in discrete Lagrangian methods for solving hard satisfiability problems”, In Proc. of the 17th Nat. Conf. on Artificial Intell. (AAAI-00), pp310-315.

[64] L. Xu, F. Hutter, H. Hoos, and K. Leyton-Brown, (2008) ”SATzilla: Portfolio-based Algorithm Selection for SAT”, Journal of Artificial Intelligence Research (JAIR), Vol. 32, pp565-606.

[65] S. Zhang, C. Tjortjis, X. Zeng, H. Qiao, I. Buchan, and J. Keane, (2009) ”Comparing data mining methods with logistic regression in childhood obesity prediction”, Journal of information Systems Frontiers, Vol. 11, No. 4, pp449-460.

[66] D. Zhang, and J. J. P Tsai, (2003) ”Machine Learning and Software Engineering”, Software Quality Journal, Vol. 11, No. 2, pp87-119.