Combining Genetic Algorithm with Variable Neighborhood Search for MAX-SAT

Noureddine Bouhmala and Kjell Ivar

University College of Southeast Norway,
Department of Maritime Technology and Innovation
nouredine.bouhmala@hbv.no

Abstract. Metaheuristics used to tackle different optimization prob-lems that arise in various fields aim at finding a tactical interplay be-tween diversification and intensification to overcome local optimality. In this paper, a genetic algorithm (GA) combined with variable neighbor-hood search (VNS) is proposed for solving the maximum satisfiability problem. VNS systematically changes different neighborhoods within a local search. The idea is that a local optimum reached within one neighborhood structure is not necessarily the same local optimum of another neighborhood structure, thus the search can systematically explore differ-ent search areas which are defined by different neighborhood structures. Results comparing the genetic with and without VNS are presented. In addition, GA combined with VNS is compared against state-of-the-art algorithms in order to show its effectiveness.

Key words: genetic algorithm, variable neighborhood search , maxi-mum satisfiability problem.

1 Introduction

The MAX-SAT problem which is known to be NP-complete [4] is defined as follows. Given a positive constant k, a propositional formula $\Phi = \bigwedge_{j}^{m}{}_{=1} C_j$ with M clauses and N Boolean variables. Each Boolean variable, x_i, $i \in \{1, \ldots, n\}$, takes one of the two values, or . Each clause C_j, in turn, is a disjunction of Boolean variables and has the form:

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

where I_j, $I^{-}{}_j \subseteq \{1, ...n\}$, $I \cap I^{-}{}_j = \emptyset$, and $x^{-}{}_i$ denotes the negation of x_i. As a simple example let $\Phi(x)$ be the following formula containing 4 variables and 3 clauses:

$$\Phi(x) = (\mathrm{x1} \vee \neg x4) \wedge (\neg x1 \vee x3) \wedge (\neg x1 \vee x4 \vee x2).$$

The task is to determine whether or not there exist a truth assignment for Φ that satisfies the maximum number k of clauses. Several state-of-the-art local search algorithms are enhanced versions of WalkSAT (WSAT) [12] algorithm.

Examples include WalkSAT/Tabu[10], Novelty+ and R-Novelty+ heuristics [7], G2WSAT[9]. Evolutionary algorithms are heuristic algorithms that have been applied to MAX-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[8] is considered to be the best known genetic algorithm for MAX-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 MAX-SAT algorithms.

2 Combining GA with VNS (VNS-GA)

Variable Neighborhood Search (VNS) [6] is a relatively recent meta-heuristic for solving combinatorial and global optimization problems. Unlike many standard metaheuristics where only a single neighborhood is employed, VNS systemati-cally changes different neighborhoods within a local search. The idea is that a local optimum reached within one neighborhood structure is not necessarily the same local optimum of another neighborhood structure, thus the search can sys-tematically explore different search areas which are defined by different neighbor-hood structures. Genetic Algorithms [5] are stochastic methods for global search and optimization and belong to the group of evolutionary algorithms. They si-multaneously examines and manipulates a set of possible solution. 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 chro-mosomes 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 off-springs depending on the probability of the cross-over operator. The new population is then subjected to mutation and enters into an intermediate population. The mutation operator acts as an element of diversity into the population and is generally applied with a low prob-ability to avoid disrupting cross-over results. 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 from generation to generation by repeated applications of an evaluation procedure that is based on genetic operators. Over many generations, the population becomes increasingly uniform until it ultimately converges to optimal or near-optimal solutions. The proposed VNS-GA is described as follows:

– **Neighborhood Selection:** Let L denotes the set of variables of the MAX-SAT problem to be solved. The first phase of the algorithm consists in con-

structing a set of neighborhood satisfying the following property: $N_1(x) \subset N_2(x) \subset .....N_{k_{max}}(x)$. The starting (default) neighborhood with $k = 1$ consists of a move based on the flip of a single variable. A flip means assigning the opposite state to a variable (i.e., change True → False or False → True). The first neighborhood N_2 is constructed from L by merging variables. The merging procedure is computed using a randomized algorithm. The variables are visited in a random order. If a variable v_i has not been matched yet, then a randomly unmatched variable v_j is selected, and a cluster v_k consisting of the two variables l_i and l_j is created. The set N_2 consists of moves based on flipping predefined clusters each having 2^1 variables. The new formed clusters are used to define a new and larger neighborhood N_3 and recursively iterate the process until the desired number of neighborhood (k_{max}) is reached.

- **Initial Population**
  The chromosomes which are assignments of values to the variables are en-coded as strings of bits, the length of which is the number of variables. The values and are represented by 1 and 0 respectively. In this representation , a chromosome X corresponds to a truth assignment and the search space is the set $S = \{0, 1\}^n$. A random initial population is generated from the largest neighborhood ($N_{k_{max}}$). The random initial population consists in assigning to each chromosome of the population a random value (True or False) to each cluster and all the variables within that cluster will get the same value.
- **Fitness Function**
  The notion of fitness is fundamental to the application of genetic 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 (individual) in the population is equal to the number of clauses that are un-satisfied by the truth assignment represented by the chromosome.
- **What Neighborhood to start from**
  The most crucial part when applying VNS is the selection of the different neighborhoods according to some strategy for the effectiveness of the search process. The strategy adopted in this work is to let VNS-GA start the search process from the largest neighborhood $N_{k_{max}}$ and continues to move towards smaller neighborhood structures. The motivation behind this strategy is that the order in which the neighborhood structures have been selected offers a bet-ter mechanism for performing diversification and intensification. The largest neighborhood N_{max} allows GA to view any cluster of variables as a single entity leading the search to become guided in far away regions of the solu-tion space 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 switch from one neighborhood to another implies a decrease in the size of the neighborhood, the search is intensified around solutions from previous neighborhoods in order to reach better ones.

- **Matching Process** New solutions are created by combining pairs of individ-uals in the population and then applying a crossover operator to each chosen pair. Combining pairs of individuals can be viewed as a matching process. In

the version of GA used in this work, the individuals are visited in random order. An unmatched individual i_k is matched randomly with an unmatched individual i_l.

– **Cross-Over Operator**
The task of the crossover operator is to reach regions of the search space with higher average quality. The two-point crossover operator is applied to each matched pair of individuals. The two-point crossover selects two randomly points within a chromosome and then interchanges the two parent chromo-somes 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 (re-ferred as offspring). The creation of these descendants involves the location and combinations of features extracted from the parents.

x1:0	x2:0	x3:1	$\downarrow$ **x4** : **0**	$\downarrow$ **x5** : **1**	$\downarrow$ **x6** : **0**	$\downarrow$ **x7** : **1**	x8:1	x9:0	x10
x1:0	x2:1	x3:0	$\uparrow$ **x4** : **1**	$\uparrow$ **x5** : **0**	$\uparrow$ **x6** : **1**	$\uparrow$ **x7** : **0**	x8:0	x9:1	x10

Table 1. Two parents before the cross-over operator .

x1:0	x2:0	x3:1	$\downarrow$ **x4** : **1**	$\downarrow$ **x5** : **0**	$\downarrow$ **x6** : **1**	$\downarrow$ **x7** : **0**	x8:1	x9:0	x10
x1:0	x2:1	x3:0	$\uparrow$ **x4** : **0**	$\uparrow$ **x5** : **1**	$\uparrow$ **x6** : **0**	$\uparrow$ **x7** : **1**	x8:0	x9:1	x10

Table 2. Results of the cross-over operator.

– **Mutation**
The purpose of mutation 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. In case of a large neighborhood ($k > 0$), the mutation is applied to a cluster of variables.

Selection
The selection acts on individuals in the current population. 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 $P_{Selection_i} = f_i / \sum_1^N f_i$,
where f_i is the fitness of individual i.

– **Switch to a smaller Neighborhood**
Once GA has reached the convergence criterion with respect to a neighborhood N_i, It switches to another neighborhood and the assignment reached on the neighborhood N_i must be projected on its parent neighborhood N_{i-1}. The projection algorithm is simple; if a cluster $c_i \in \mathrm{N}_m$ is assigned the value of true then the merged pair of clusters that it represents, c_j , $c_k \in \mathrm{N}_{m-1}$ are also assigned the true value. Finally, GA is applied at the default neighborhood.

3 Experimental Results

3.1 Test Suite & Parameter Settings

The performance of VNS-GA is evaluated against GA using a set of random problems (MAX-2SAT, MAX-3SAT). This set is taken from the Ninth MAX-SAT 2014 evaluation (http://maxsat.ia.udl.cat/benchmarks/) organized as an affiliated event of the 17th International Conference on Theory and Applications of Satisfiability Testing(SAT 2014). Due to the randomization nature of both algorithms, each problem instance was run 50 times with a cut–off parameter (max-time) set to 15 minutes. 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:

– k_{max}: The cardinality of the neighborhood is set such that the number of the formed clusters is 10% of the size of the problem instance (i.e, a problem with 100 problems will lead to k_{max} equals to 3).
– GA is assumed to have reached convergence and switch to a smaller neighbor-hood if the fitness of the fittest chromosome remains unchanged during five consecutive generations.

3.2 Statistical Analysis

The results showing the statistical comparison between the two algorithms are presented in Tables 3-6. The mean (x) and the standard deviation (σ) of unsolved clauses for VNS-GA and GA algorithms are reported. 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 which compares the difference in means between the two groups. Comparison using the non-parametric Mann-Whitney U-test gave identical results. The non-parametric effect size measure $\hat{A}_{12}$ [14] 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 is a common component in any non-parametric analysis such as the Mann-Whitney U-test [1]. Calculating $\hat{A}_{12}$ is done according to the following formula:

	GA		VNS-GA	
Instance	**x (σ)**	**Min-Max**	**x (σ)**	**Min-Max**
s2v100c1200-1	210.6 (6.2)	192-229	171.5 (3.4)	169-180
s2v100c1200-2	203.1 (6.3)	186-217	163.9 (1.3)	163-169
s2v100c1200-3	212.8 (5.5)	199-226	173.7 (2.7)	172-182
s2v100c1200-4	207.2 (6.2)	191-223	171 (2.2)	169-178
s2v100c1200-5	207.2 (6.2)	191-223	171 (2.2)	169-178
s2v100c1200-6	195.5 (7.3)	179-215	154.5 (1.8)	153-160
s2v100c1200-7	204 (5.9)	188-221	167.8 (0.9)	167-172
s2v100c1200-8	207.9 (5.9)	192-222	172.5 (2.3)	171-181
s2v100c1200-9	206.5 (5.6)	195-221	173.8 (1.6)	171-179
s2v100c1200-10	206.7 (6.7)	184-224	163.5 (3.2)	162-176
s2v120c1200-1	203.2 (5.4)	185-217	163 (2.5)	161-171
s2v120c1200-2	200.1 (5.7)	184-212	161.2 (1.9)	159-167
s2v120c1200-3	204.6 (4.8)	192-217	162.8 (4.7)	160-176
s2v120c1200-4	201.2 (6.3)	185-219	158.2 (1.5)	157-164
s2v120c1200-5	188.1 (6.7)	172-211	144.2 (1.1)	143-148
s2v120c1200-6	207 (5.6)	194-225	170.2 (2.2)	167-176
s2v120c1200-7	201.2 (5.3)	188-215	164.2 (2.7)	162-170
s2v120c1200-8	203.6 (5.7)	192-224	167.4 (2.1)	165-173
s2v120c1200-9	195 (6.2)	178-209	148.4 (0.7)	148-151
s2v120c1200-10	196.6 (4.8)	186-207	154.3 (1.4)	154-165
s2v140c1300-1	163.8 (1.8)	162-169	163.6 (1.7)	162-168
s2v140c1300-2	171.9 (1.5)	171-179	172.6 (2.2)	171-181
s2v140c1400-1	183.7 (2.2)	182-192	184.1 (2.3)	182-197
s2v140c1400-2	179 (2.9)	178-190	179 (3.1)	178-192
s2v140c1500-1	205 (0.7)	205-208	205.6 (1)	205-211
s2v140c1500-2	201 (1.7)	199-207	200.9 (1.7)	199-208

Table 3. 2SAT: VNS-GA Vs GA. Notes: x = Mean, σ = Standard Deviation, Min = Minimum observed value, Max = Maximal observed value.

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

where R1 is the rank sum of algorithm VNS-GA, m is the number of ob-servations in the first data sample, and n is the number of observations in the second data sample. VNS-GA algorithm dominates GA for all MAX2SAT in-stances with less than 1300 clauses. However, for one instance (s2v140c1300-2) GA shows better performance, although the best solution (171 unsolved clauses) is identical for both algorithms. For the five remaining instances there is no sta-tistically significant difference between the two algorithms. VNS-GA dominates GA on all MAX3SAT instances with 70 variables (s3v70c1000-1 to s3v70c1000- 10). GA is significantly better on one instance (s3v80c1000-2) with no statistical difference between the remaining algorithms with 80 variables (s3v80c1000-1 to s3v80-1000-10). When GA is significantly better the difference with respect to best solutions is marginal or non-existent. VNS-GA has a much lower variation of results across the test suite and we take this to mean that the variable neigh-

borhood scheme stabilizes GA in some way. Table 7 compares VNS-GA with two highly efficient incomplete solvers used at the 2014 MAX-SAT competition. For each solver, we report the number of unsatisfied clauses, while the number between parenthesis denotes the time in seconds. CCLS2akms and ISAC+2014-ms gives similar solution quality while the main differences resides on the time required to produce such quality. VNS-GA is capable of delivering the same solution quality as these two solvers in 18 cases out of 27. When compared to CCLS2akms, VNS-GA converges faster in 8 cases. The time of VNS-GA in these cases lies between 10% and 77% of the time of CCLS2akms. For the the remaining cases, the time of CCLS2akms is between 31% and 86% of the time of VNS-GA. The comparison between VNS-GA and ISAC+2014-ms reveals that VNS-GA converges faster in 12 cases. The time of VNS-GA lies between 19% and 89% of the time of ISAC +2014-ms. The time of ISAC+2014-ms is between 10%and 82% in the remaining 6 cases. VNS-GA was beaten by these two solvers in 9 cases. The difference in quality expressed as the number of unsatisfied clauses never exceeds 1.

	GA		**VNS-GA**	
Instance	**x (σ)**	**Min-Max**	**x (σ)**	**Min-Max**
s3v70c1000-1	71.6 (3.5)	65-80	52 (2.6)	47-59
s3v70c1000-2	69.1 (4.3)	57-80	48.2 (4.3)	43-60
s3v70c1000-3	71.1 (4)	62-80	51 (3.6)	45-60
s3v70c1000-4	72.2 (3.8)	63-81	52.9 (3)	47-61
s3v70c1000-5	67.8 (4.1)	56-77	45 (2.9)	42-52
s3v70c1000-6	71.9 (3.6)	63-81	54.4 (2.3)	51-61
s3v70c1000-7	70.9 (2.9)	62-78	53.7 (2.1)	49-58
s3v70c1000-8	69.5 (3.8)	60-80	51.8 (2.9)	48-60
s3v70c1000-9	70.9 (3.9)	60-83	52.7 (2.6)	49-61
s3v70c1000-10	69.7 (4.4)	61-86	48.8 (3)	45-61
s3v80c1000-1	47.8 (2.9)	44-55	47.6 (3.2)	44-57
s3v80c1000-2	46.4 (2.1)	43-52	47.5 (2.7)	43-55
s3v80c1000-3	44.7 (4)	39-57	45.3 (4)	39-54
s3v80c1000-4	49.3 (2.6)	45-57	50 (2.9)	45-60
s3v80c1000-5	45.4 (3.1)	41-53	45.4 (3.4)	41-55
s3v80c1000-6	44.8 (3.5)	40-57	44.4 (3.3)	40-52
s3v80c1000-7	43.2 (3.1)	40-52	43.1 (2.9)	40-56
s3v80c1000-8	45.4 (2.8)	41-58	45.9 (2.7)	41-53
s3v80c1000-9	41.1 (3)	38-55	41.5 (2.7)	38-49
s3v80c1000-10	42.8 (2.8)	39-51	43.4 (3.3)	39-55

Table 4. 3SAT: VNS-GA Vs GA. Notes: x = Mean, σ = Standard Deviation, Min = Minimum observed value, Max = Maximal observed value

Prob	Δ[95%$CI of \Delta$]	p	PS	[95% CI of PS]
s2v100c1200-1	39.2 [37.3, 41]	***	1	c
s2v100c1200-2	39.3 [37.6, 41]	***	1	c
s2v100c1200-3	39 [37.4, 40.6]	***	1	c
s2v100c1200-4	36.2 [34.5, 37.9]	***	1	c
s2v100c1200-5	41 [39, 42.9]	***	1	c
s2v100c1200-6	36.1 [34.6, 37.7]	***	1	c
s2v100c1200-7	35.4 [33.8, 37.1]	***	1	c
s2v100c1200-8	32.8 [31.3, 34.3]	***	1	c
s2v100c1200-10	43.2 [41.2, 45.1]	***	1	c
s2v120c1200-1	40.2 [38.6, 41.7]	***	1	c
s2v120c1200-2	38.9 [37.4, 40.5]	***	1	c
s2v120c1200-3	41.8 [40.1, 43.6]	***	1	c
s2v120c1200-4	43 [41.3, 44.7]	***	1	c
s2v120c1200-5	43.8 [42.1, 45.6]	***	1	c
s2v120c1200-6	36.9 [35.3, 38.4]	***	1	c
s2v120c1200-7	36.9 [35.4, 38.5]	***	1	c
s2v120c1200-8	36.2 [34.6, 37.8]	***	1	c
s2v120c1200-9	46.5 [44.9, 48.2]	***	1	c
s2v120c1200-10	42.3 [41, 43.6]	***	1	c
s2v140c1300-1	0.2 [-0.5, 0.8]	.526	.528	[.456, .607]
s2v140c1300-2	-0.7 [-1.4, 0]	.014*	.422	[.351, .494]
s2v140c1400-1	-0.4 [-1.3, 0.4]	.176	.431	[.356, .509]
s2v140c1400-2	0[-1.1, 1.1]	.944	.523	[.465, .579]
s2v140c1500-1	0[-0.3, 0.3]	.844	.534	[.465, .601]
s2v140c1500-2	0 [-0.6, 0.7]	.901	.496	[.418, .577]

Table 5. Statistical comparison of the solutions given by VNS-GA and GA. Notes: Δ = Mean difference, CI = Confidence Interval, p = p-value, PS = Probability of Superiority (i.e. that VNS-GA will have less unsolved clauses than GA for each instance), $\star\star\star$ = $p < .001$. $\star\star = p < .01$, $\star p < .05$, c = not possible to calculate CI of PS due to no overlap between VNS-GA and GA. The 95% CI for PS is calculated in Excel using a bootstrapping procedure (random selection with replacement) performed 1000 times.

4 Conclusions

In this work, a variable neighborhood search combined with the genetic algorithm for the maximum satisfiability problem is introduced. VNS follows a simple prin-ciple that is based on systematic changes of neighborhood within the search. The set of neighborhood proposed in this paper can easily be incorporated into any metaheuristic when dealing with various combinatorial optimization problems. Starting the search from the largest neighborhood and moving systematically towards the smallest neighborhood is a better strategy for performing diversifi-cation and intensification. The results indicate that the variable neighborhood search strategy can enhance the convergence behavior of GA. It appears clearly from the results that the performance of VNS-GA is better compared to that of GA. The larger the problem, the larger the size of the neighborhood is needed,

Prob	Δ[95%$CI of \Delta$]	p	PS	[95% CI of PS]
s3v70c1000-1	19.6 [18.4, 20.7]	***	1	c
s3v70c1000-2	20.9 [19.3, 22.5]	***	.999	[.998, 1]
s3v70c1000-3	20.2 [18.8, 21.5]	***	1	c
s3v70c1000-4	19.3 [18, 20.6]	***	1	c
s3v70c1000-5	22.8 [21.4, 24.1]	***	1	c
s3v70c1000-6	17.6 [16.4, 18.7]	***	1	c
s3v70c1000-7	17.2 [16.2, 18.1]	***	1	c
s3v70c1000-8	17.6 [16.4, 18.9]	***	1	[.998, 1]
s3v70c1000-9	18.2 [17, 19.4]	***	1	[.998, 1]
s3v70c1000-10	20.9 [19.5, 22.3]	***	1	[.998, 1]
s3v80c1000-1	0.2 [-0.9, 1.3]	.708	.528	[.451, .611]
s3v80c1000-2	-1.1 [-2, -0.2]	.002**	.383	[.307, .458]
s3v80c1000-3	-0.6 [-2.1, 0.9]	.279	.452	[.372, .530]
s3v80c1000-4	-0.8 [-1.8, 0.3]	.055	.417	[.344, .494]
s3v80c1000-5	0.1 [-1.1, 1.3]	.879	.514	[.438, .590]
s3v80c1000-6	0.4 [-0.9, 1.6]	.42	.527	[.447, .606]
s3v80c1000-7	0.1 [-1, 1.2]	.852	.496	[.418, .570]
s3v80c1000-8	-0.4 [-1.4, 0.6]	.274	.44	[.367, .522]
s3v80c1000-9	-0.4 [-1.4, 0.7]	.242	.444	[.373, .522]
s3v80c1000-10	-0.6 [-1.7, 0.6]	.191	.46	[.381, .538]

Table 6. Statistical comparison of the solutions given by VNS-GA and GA. Notes: Δ = Mean difference, CI = Confidence Interval, p = p-value, PS = Probability of Superiority (i.e. that VNS-GA will have less unsolved clauses than the GA for each instance), $\star\star\star = p < .001$. $\star\star = p < .01$, $\star p < .05$, c = not possible to calculate CI of PS due to no overlap between VNS-GA and GA. The 95% CI for PS is calculated in Excel using a bootstrapping procedure (random selection with replacement) performed 1000 times.

and consequently the more efficient is GA at different neighborhoods. When compared to state of art solvers, VNS-GA is capable of producing similar re-sults than the top ranked solvers while requiring the least amount of time for several test cases. In order to improve VNS-GA, a better strategy would be to construct the different neighborhoods based on merging variables by exploiting the number of clauses they have in common rather randomly.

References

1. Arcuri, A., Briand, L.: A Hitchhiker's Guide to Statistical Tests for Assessing Ran-domized Algorithms in Software Engineering. Technical report, simula research lab-oratory, number 13/2011, (2011).
2. Audemard,G., Simon,L.: in Twenty-first International Joint Conference on Artificial Intelligence (IJCAI'09), july (2009)
3. Cai,S., Luo,C., Su,K.: CCASat: Solver description. In: Proc. of SAT Challenge 2012: Solver and Benchmark Descriptions, 1314 (2012)

Problem	CCLS2akms	ISAC+2014-ms	VNS-GA
s2v120c1200-1	161 (7.43)	161 (14.74)	161 (11.48)
s2v120c1200-2	159 (8.54)	159 (27.95)	159 (7.77)
s2v120c1200-3	160 (3.88)	160 (7.88)	160 (8.66)
s2v120c1200-4	157 (3.98)	157 (7.01)	158 (6.71)
s2v120c1200-5	143 (2.05)	143 (5.32)	144 (34.53)
s2v120c1200-6.	167 (8.66(	167 (16.18)	169 (3.79)
s2v120c1200-7	162 (9.24)	162 (12.23)	162 (65.42)
s2v120c1200-8	165 (27.84)	165 (34.01)	165 (7.16)
s2v120c1200-9	148 (2.23)	148 (4.80)	148 (7.67)
s2v120c1200-10	154 (2.58)	154 (8.40)	154 (10.08)
s2v120c1300-1	180 (18.74)	180 (22.93)	180 (13.57)
s2v120c1300-2	172 (6.53)	172 (14.79)	172 (16.94)
s2v120c1300-3	173 (8.42)	173 (20.16)	173 (12.69)
s2v120c1300-5	168 (3.32)	168 (11.90)	169 (37.63)
s2v120c1300-7	169 (2.56)	169 (12.31)	169 (2.75)
s2v120c1300-9	186 (39.54)	186 (59.20)	186 (12.31)
s2v140c1200-1	144 (12.29)	144 (16.98)	144 (13.11)
s2v140c1200-2	155 (153.63)	155 (243.53)	156 (31.04)
s2v140c1300-3	168 (51.45)	168 (70.52)	169 (13.89)
s2v140c1400-1	182 (56.99)	182 (113.89)	182 (13.23)
s2v140c1400-2	178 (32.54)	178 (47.53)	178 (53.85)
s2v140c1400-3	193(142.84)	193 (385.21)	193 (69.41)
s2v140c1400-4	184 (35.14)	184 (73.44)	184 (50.51)
s2v140c1500-1	205 (200.67)	205 (170.26)	205 (138.31)
s2v140c1500-2	199 (203.13)	199 (317.05)	200 (178.10)
s2v140c1500-3	212 (588.53)	212 (1098)	213 (300.10)
s2v140c1500-4	197 (71.49)	197 (89.68)	198 (91.10)

Table 7. Comparing VNS-GA with state-of-art Incomplete solvers.

4. Cook.S.A: The complexity of theorem-proving procedures. Proceedings of the Third ACM Symposium on Theory of Computing, 151–158, (1971)
5. Frank,J.: A study of genetic algorithms to find approximate solutions to hard 3CNF problem. In proceedings of Golden West international conference on artificial intel-ligence, (1994).
6. Hansen, P., Jaumard,B., Mladenovic,N., Parreira, A.D.: Variable neighborhood search for maximum weighted satisfiability problem. Technical Report G-2000-62, Les Cahiers du GERAD, Group for Research in Decision Analysis, 2000.
7. Hoos,H.: On the run-time behavior of stochastic local search algorithms for SAT. In Proceedings of AAAI-99, 661-666, (1999)
8. Jin-Kao, H., Lardeux, H., Saubion, F.: Evolutionary computing for the satisfia-bility problem. In Applications of Evolutionary Computing, volume 2611 of LNCS, 258-267, University of Essex, 2003.
9. Li,C.M., Huang, W.Q.: Diversification and determinism in local search for satisfi-ability. Proceedings of the Eighth International Conference on Theory and Appli-cations of Satisfiability Testing (SAT-05), 569, Lecture Notes in Computer Science, 158–172, (2005).

10. McAllester,D., Selman,B. Kautz, H.: Evidence for invariants in local search. proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97), pages 321-326, 1997.
11. Narendra, K.S., M.A.L. Thathachar, M.A.L.: Learning Automata: An Introduction. Prentice Hall, (1989)
12. Selman, B., Kautz,H.A., Cohen, B.: Noise Strategies for Improving Local Search. Proceedings of AAAI'94, 337–343. MIT Press, (1994)
13. Tsetlin.M.L.: Automaton Theory and Modeling of Biological Systems. Academic Press, (1973)
14. Vargha, A., Delaney,H.D.: A critique and improvement of the CL Common Language Effect Size statistics of McGraw and Wong. Journal of Educational and Be-havioral Statistics, 25(2), 101-132, (2000).