Combining Genetic Algorithm with Variable Neighbourhood Search for MAX-SAT

Noureddine Bouhmala[1], Kjell Ivar Øvergård[2]
{noureddine.bouhmala@usn.no[1], koe@usn.no[2]}

University College of Southeast Norway, Department of Southeast Norway,
Postboks 4, 3199 Borre, Norway[1,2]

Abstract. Metaheuristics which have gain popularity for solving different optimization problems that arise in various fields aim at finding good solutions at a low cost. In this paper, a hybrid approach combining genetic algorithm (GA) with variable neighbourhood search (VNS) is proposed for solving the maximum satisfiability problem (MAX-SAT). VNS works by moving from one neighbourhood to another in order to avoid getting trapped in local optima. Results comparing GA with and without VNS are presented. Finally, GA combined with VNS is compared to highly efficient solvers from the MAX-SAT 2014 competition.

Keywords: genetic algorithm, variable neighbourhood search, maximum satisfiability problem.

1 Introduction

The MAX-SAT problem which is proved to be NP-complete [1] is formulated as follows. Given a positive constant k, a propositional formula $\Phi = \wedge_{j=1}^{m} C_j$ with M clauses and N Boolean variables. Each Boolean variable, x_i, $i \in \{1, . . . , n\}$, can be assigned one of the two values, TRUE or FALSE. Each clause C_j, in turn, is a disjunction of Boolean variables and has the form as presented in formula (1) where $I_j, \bar{I}_j \subseteq \{1, ...n\}$, $I \cap \bar{I} = \varnothing$, and $\bar{x}$ denotes the negation of x.

$$C_j = \left(\vee_{l \in I_j} x_l\right) \vee \left(\vee_{l \in \bar{I}_j} \bar{x}_l\right) . \quad (1)$$

Let $\phi(x)$ be the following formula containing 4 variables and 3 clauses. The goal is to determine if there exists a truth assignment for ϕ that satisfies the maximum number k of clauses.:

$$\phi(x) = (x_1 \vee \neg x_4) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_1 \vee x_4 \vee x_2) . \quad (2)$$

Several state-of-the-art local search algorithms are enhanced versions of WalkSAT (WSAT) [2] algorithm. Examples include WalkSAT/Tabu [3], Novelty+ and R-Novelty+

heuristics [4], G2WSAT [5]. 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 [6] is considered to be the best known genetic algorithm for MAX-SAT. GASAT is a hybrid algorithm that combines a specific crossover and a taboo 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 Neighbourhood Search (VNS) [7] is a meta-heuristic for solving combinatorial and global optimization problems. Most metaheuristics are based on using a single neighbourhood during the whole search. VNS exploits different neighbourhoods within a local search. VNS is based on the fact that a local optimum reached within one neighbourhood is different from the local optimum of another neighbourhood , thus the search can systematically explore different search areas which are defined by different neighbourhood structures.

Genetic Algorithms (GA) [8] are non-deterministic methods for global optimization and belong to the class of evolutionary algorithms. GA work with a set of solutions in parallel. A gene is defined to be the smallest unit of genetic information. Every gene is able to take various values called allele. The chromosomes are encoded using an appropriate representation and each can be regarded of as a solution in the search space . Each individual is assigned a value called (fitness) that permits assessing its quality. The chromosomes that form the initial population may be randomly constructed or by using greedy techniques leading to chromosomes having high fitness. GA proceeds afterwards by grouping chromosomes into pairs and enters them in a mating pool. Each pair is drawn from the mating pool and combined using the cross-over operator leading to offspring. The new population goes through the mutation operator which brings diversity into the population. The probability of the mutation operator is generally chosen to be very low so that the results achieved during the cross-over phase are not completely disrupted. A selection strategy based on the fitness is usually used to choose the chromosomes that will be part of the next generation. The procedure described above iterates over many generations until the population converges to optimal or near-optimal solutions. The proposed VNS-GA is described in section 2.1 to 2.9.

2.1 Neighbourhood selection

Let L denotes the set of variables of the MAX-SAT problem to be solved. The first phase of the algorithm consists in constructing a set of neighbourhoods satisfying the following property: $N_1(x) \subset N_2(x) \subset \ldots . N_{k_{max}}(x)$. The initial neighbourhood (k = 1) is based on flipping the value of a single variable. The state of a variable is changed from (True → False) or (False → True). The first neighbourhood N_2 is computed by grouping variables into clusters using a non-deterministic algorithm. The variables are traversed in a random order. The procedure selects for each unmatched variable v_i the first unmatched variable v_j and a cluster v_k made of the two variables is formed. The neighbourhood N_2 allow flips based on changing clusters each having 2 variables. The new formed clusters are used to construct a larger neighbourhood N_3 and the procedure iterates until the desired number of neighbourhood (k_{max}) allowed by the user is reached.

2.2 Initial population

The chromosomes are represented as strings of bits. The number of the variables determine the length of the chromosomes . The values (True and False) are represented by 1 and 0 respectively. In this representation, a chromosome X corresponds to a truth assignment and the space of configuration is the set $S = \{0, 1\}^n$. A random initial population is computed from neighbourhood ($N_{k_{max}}$). The value assigned to each cluster of a given individual is broadcast to all the variables belonging to that clusters.

2.3 Fitness function

GA exploits the fitness in order to guide the search. It is a value that quantify the quality of an individual (solution) so that different solutions can be compared. The fitness of a chromosome (individual) in the population is chosen to be equal to the number of clauses that are un- satisfied.

2.4 What neighbourhood to start from

The core of VNS is the selection of the different neighbourhoods 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 neighbourhood $N_{k_{max}}$ and continues to move towards smaller neighbourhoods. The chosen order enables GA to perform the search in an efficient manner. The largest neighbourhood N_{max} makes GA to treat a cluster as a single object making the search to become directed and limited only to those solutions in which the variables grouped within a cluster are given the same value. The change of neighbourhood leads the search to be is intensified close to solutions from previous neighbourhoods in order to attain solutions of better quality.

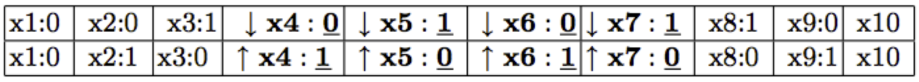
2.5 Matching process

Pairs of individuals are combined in order to create offspring using the cross-over operator. Combining pairs of individuals is regarded as a matching process. GA uses the same random procedure described in section 2.1. The chromosomes are traversed in random order. An unmatched chromosome i_k is matched randomly with an unmatched chromosome i_l.

2.6 Cross-over operator

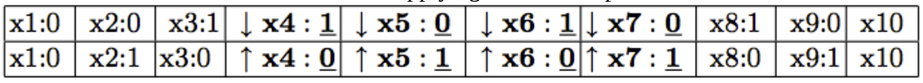
The crossover operator is regarded as the most important in GA. It allows GA to perform the search hoping to reach regions of the search space with higher fitness value. The two-point crossover operator is applied to each matched pair of individuals. It works by selecting two randomly points within a chromosome and then interchanges the two parent chromosomes between these points to generate two new offspring. Cross-over operator can be defined as a process in which a set of solutions referred as parents undergoes a transformation to create a new set of solutions configurations referred to as offspring. The creation of these descendants involves the location and combinations of patterns extracted from the parents. Examples of this process can be seen in Tables 1 and 2.

Table 1. Before applying cross-over operator.



x1:0	x2:0	x3:1	↓ **x4 : 0**	↓ **x5 : 1**	↓ **x6 : 0**	↓ **x7 : 1**	x8:1	x9:0	x10
x1:0	x2:1	x3:0	↑ **x4 : 1**	↑ **x5 : 0**	↑ **x6 : 1**	↑ **x7 : 0**	x8:0	x9:1	x10

Table 2. After applying cross-over operator.



x1:0	x2:0	x3:1	↓ **x4 : 1**	↓ **x5 : 0**	↓ **x6 : 1**	↓ **x7 : 0**	x8:1	x9:0	x10
x1:0	x2:1	x3:0	↑ **x4 : 0**	↑ **x5 : 1**	↑ **x6 : 0**	↑ **x7 : 1**	x8:0	x9:1	x10

2.7 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 neighbourhood ($k > 0$), the mutation is applied to a cluster of variables.

2.8 Selection

The selection process applies on chromosomes in the current population. Based on the fitness of each individual, the selection step chooses the next population based on the roulette method which is stochastic and biased toward the best individuals. The method works by first calculating the cumulative fitness of the whole population through the sum of the fitness of all individuals. Then, the probability of selection for each individual is computed as being $P_{Selection_i} = f_i / \sum_1^N f_i$, where f_i denotes the fitness of individual i.

2.9 Switch to a smaller Neighbourhood

As soon as GA has reached the stopping criterion according to a neighbourhood N_j. It moves to another neighbourhood and the assignment obtained on the neighbourhood N_i must be projected on its parent neighbourhood N_{i-1}. The projection algorithm is simple; if a cluster $c_j \in N_m$ is given for example the value of false, then the grouped pair of clusters that it represents, c_j , $c_k \in N_{m-1}$ are also given the false value. Finally, GA is applied at the default neighbourhood.

3 Experimental results

3.1 Test suite and parameter settings

VNS-GA is compared against GA using a set of random problems (MAX-2SAT, MAX-3SAT). This benchmark 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). Each problem instance was run 50 times with a cut–off parameter (max-time) set to 15 minutes. The experiments 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 maximum size of neighbourhood and the stopping criterion of GA have been chosen experimentally and are defined as follows:

k

$\boldsymbol{k_{max}}$: The maximum size of the neighbourhood is computed 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 stopping criterion and move to a smaller neighbourhood if the fitness of the best chromosome remains without changed during five consecutive generations.

3.2 Statistical analysis

Tables 3 and 4 compares VNS-GA against GA. The mean (x) and the standard deviation (σ) of unsolved clauses for the two algorithms are shown. The range of solutions is also calculated in order to detect the possibility of an overlap between solutions for any given instance.

Statistical inferential was conducted using independent samples t-test which compares the difference in means between the two groups. The non-parametric Mann-Whitney U-test gave identical results. The non-parametric effect size measure Probability of Superiority (PS) [9] was adopted to show the relative dominance of one algorithm over the other. The PS effect size measure is also called $\hat{A}_{12}$ [10] and is calculated using the rank sum which is a known component in any non-parametric analysis such as the Mann-Whitney U-test [11].

Calculating PS is done according to formula (3) [9, 10, 11], where R_1 denotes the rank sum of algorithm VNS-GA, m is the number of observations in the first data sample, and n is the number of observations in the second data sample.

$$PS = (R_1/m - (m + 1)/2)/n \,. \quad (3)$$

Table 3 and 4 indicate that VNS-GA algorithm beats GA for all MAX2SAT in- stances with less than 1300 clauses, except for one instance (s2v140c1300-2) where GA wins , although the best solution (171 unsolved clauses) is the same for both algorithms. For the five remaining instances, both algorithms reach the same performance.

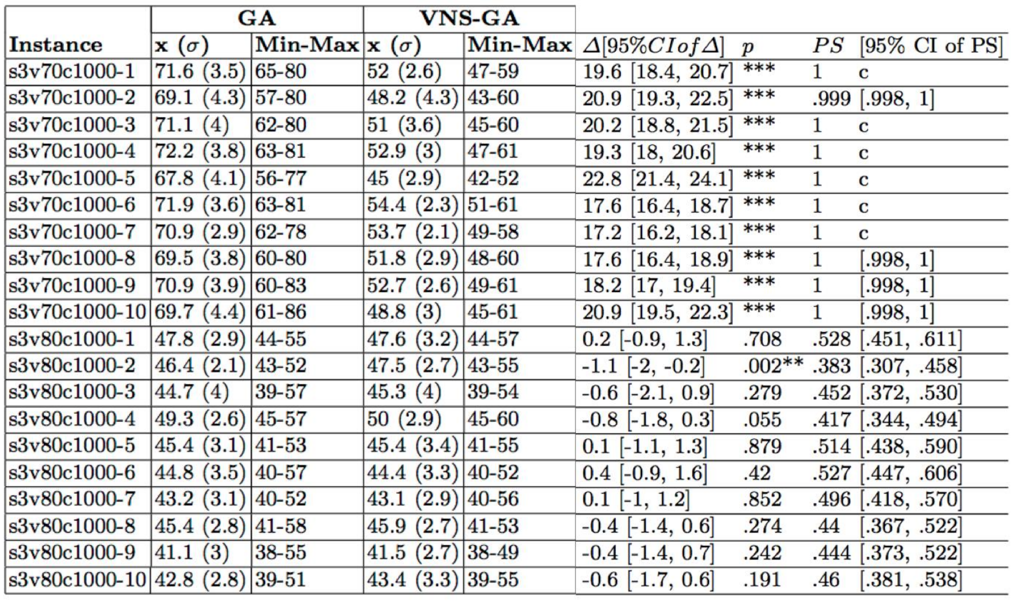
Table 3. Statistical comparison of the solutions for 2SAT-instances given by VNS-GA and GA.

	GA		VNS-GA					
Instance	x (σ)	Min-Max	x (σ)	Min-Max	Δ[95%$CI of \Delta$]	p	PS	[95% CI of PS]
s2v100c1200-1	210.6 (6.2)	192-229	171.5 (3.4)	169-180	39.2 [37.3, 41]	***	1	c
s2v100c1200-2	203.1 (6.3)	186-217	163.9 (1.3)	163-169	39.3 [37.6, 41]	***	1	c
s2v100c1200-3	212.8 (5.5)	199-226	173.7 (2.7)	172-182	39 [37.4, 40.6]	***	1	c
s2v100c1200-4	207.2 (6.2)	191-223	171 (2.2)	169-178	36.2 [34.5, 37.9]	***	1	c
s2v100c1200-5	207.2 (6.2)	191-223	171 (2.2)	169-178	41 [39, 42.9]	***	1	c
s2v100c1200-6	195.5 (7.3)	179-215	154.5 (1.8)	153-160	36.1 [34.6, 37.7]	***	1	c
s2v100c1200-7	204 (5.9)	188-221	167.8 (0.9)	167-172	35.4 [33.8, 37.1]	***	1	c
s2v100c1200-8	207.9 (5.9)	192-222	172.5 (2.3)	171-181	32.8 [31.3, 34.3]	***	1	c
s2v100c1200-10	206.7 (6.7)	184-224	163.5 (3.2)	162-176	43.2 [41.2, 45.1]	***	1	c
s2v120c1200-1	203.2 (5.4)	185-217	163 (2.5)	161-171	40.2 [38.6, 41.7]	***	1	c
s2v120c1200-2	200.1 (5.7)	184-212	161.2 (1.9)	159-167	38.9 [37.4, 40.5]	***	1	c
s2v120c1200-3	204.6 (4.8)	192-217	162.8 (4.7)	160-176	41.8 [40.1, 43.6]	***	1	c
s2v120c1200-4	201.2 (6.3)	185-219	158.2 (1.5)	157-164	43 [41.3, 44.7]	***	1	c
s2v120c1200-5	188.1 (6.7)	172-211	144.2 (1.1)	143-148	43.8 [42.1, 45.6]	***	1	c
s2v120c1200-6	207 (5.6)	194-225	170.2 (2.2)	167-176	36.9 [35.3, 38.4]	***	1	c
s2v120c1200-7	201.2 (5.3)	188-215	164.2 (2.7)	162-170	36.9 [35.4, 38.5]	***	1	c
s2v120c1200-8	203.6 (5.7)	192-224	167.4 (2.1)	165-173	36.2 [34.6, 37.8]	***	1	c
s2v120c1200-9	195 (6.2)	178-209	148.4 (0.7)	148-151	46.5 [44.9, 48.2]	***	1	c
s2v120c1200-10	196.6 (4.8)	186-207	154.3 (1.4)	154-165	42.3 [41, 43.6]	***	1	c
s2v140c1300-1	163.8 (1.8)	162-169	163.6 (1.7)	162-168	0.2 [-0.5, 0.8]	.526	.528	[.456, .607]
s2v140c1300-2	171.9 (1.5)	171-179	172.6 (2.2)	171-181	-0.7 [-1.4, 0]	.014*	.422	[.351, .494]
s2v140c1400-1	183.7 (2.2)	182-192	184.1 (2.3)	182-197	-0.4 [-1.3, 0.4]	.176	.431	[.356, .509]
s2v140c1400-2	179 (2.9)	178-190	179 (3.1)	178-192	0[-1.1, 1.1]	.944	.523	[.465, .579]
s2v140c1500-1	205 (0.7)	205-208	205.6 (1)	205-211	0[-0.3, 0.3]	.844	.534	[.465, .601]
s2v140c1500-2	201 (1.7)	199-207	200.9 (1.7)	199-208	0 [-0.6, 0.7]	.901	.496	[.418, .577]

Table 3: 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), ⋆ ⋆ ⋆ = p < .001. ⋆ ⋆ = p < .01, ⋆ 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. Notes: x = Mean, σ = Standard Deviation, Min = Minimum observed value, Max = Maximum observed value.

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 neighbourhood scheme stabilizes GA in some way.

Table 4: Statistical comparison of the solutions for 3SAT instances given by VNS-GA and GA.



Instance	GA		VNS-GA					
	x (σ)	Min-Max	x (σ)	Min-Max	$\Delta[95\% CI of \Delta]$	p	PS	[95% CI of PS]
s3v70c1000-1	71.6 (3.5)	65-80	52 (2.6)	47-59	19.6 [18.4, 20.7]	***	1	c
s3v70c1000-2	69.1 (4.3)	57-80	48.2 (4.3)	43-60	20.9 [19.3, 22.5]	***	.999	[.998, 1]
s3v70c1000-3	71.1 (4)	62-80	51 (3.6)	45-60	20.2 [18.8, 21.5]	***	1	c
s3v70c1000-4	72.2 (3.8)	63-81	52.9 (3)	47-61	19.3 [18, 20.6]	***	1	c
s3v70c1000-5	67.8 (4.1)	56-77	45 (2.9)	42-52	22.8 [21.4, 24.1]	***	1	c
s3v70c1000-6	71.9 (3.6)	63-81	54.4 (2.3)	51-61	17.6 [16.4, 18.7]	***	1	c
s3v70c1000-7	70.9 (2.9)	62-78	53.7 (2.1)	49-58	17.2 [16.2, 18.1]	***	1	c
s3v70c1000-8	69.5 (3.8)	60-80	51.8 (2.9)	48-60	17.6 [16.4, 18.9]	***	1	[.998, 1]
s3v70c1000-9	70.9 (3.9)	60-83	52.7 (2.6)	49-61	18.2 [17, 19.4]	***	1	[.998, 1]
s3v70c1000-10	69.7 (4.4)	61-86	48.8 (3)	45-61	20.9 [19.5, 22.3]	***	1	[.998, 1]
s3v80c1000-1	47.8 (2.9)	44-55	47.6 (3.2)	44-57	0.2 [-0.9, 1.3]	.708	.528	[.451, .611]
s3v80c1000-2	46.4 (2.1)	43-52	47.5 (2.7)	43-55	-1.1 [-2, -0.2]	.002**	.383	[.307, .458]
s3v80c1000-3	44.7 (4)	39-57	45.3 (4)	39-54	-0.6 [-2.1, 0.9]	.279	.452	[.372, .530]
s3v80c1000-4	49.3 (2.6)	45-57	50 (2.9)	45-60	-0.8 [-1.8, 0.3]	.055	.417	[.344, .494]
s3v80c1000-5	45.4 (3.1)	41-53	45.4 (3.4)	41-55	0.1 [-1.1, 1.3]	.879	.514	[.438, .590]
s3v80c1000-6	44.8 (3.5)	40-57	44.4 (3.3)	40-52	0.4 [-0.9, 1.6]	.42	.527	[.447, .606]
s3v80c1000-7	43.2 (3.1)	40-52	43.1 (2.9)	40-56	0.1 [-1, 1.2]	.852	.496	[.418, .570]
s3v80c1000-8	45.4 (2.8)	41-58	45.9 (2.7)	41-53	-0.4 [-1.4, 0.6]	.274	.44	[.367, .522]
s3v80c1000-9	41.1 (3)	38-55	41.5 (2.7)	38-49	-0.4 [-1.4, 0.7]	.242	.444	[.373, .522]
s3v80c1000-10	42.8 (2.8)	39-51	43.4 (3.3)	39-55	-0.6 [-1.7, 0.6]	.191	.46	[.381, .538]

Table 4. Notes: x = Mean, σ = Standard Deviation, Min = Minimum observed value, Max = Maximal observed value, 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), ⋆ ⋆ ⋆ = p < .001. ⋆ ⋆ = p < .01, ⋆ 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. Notes:

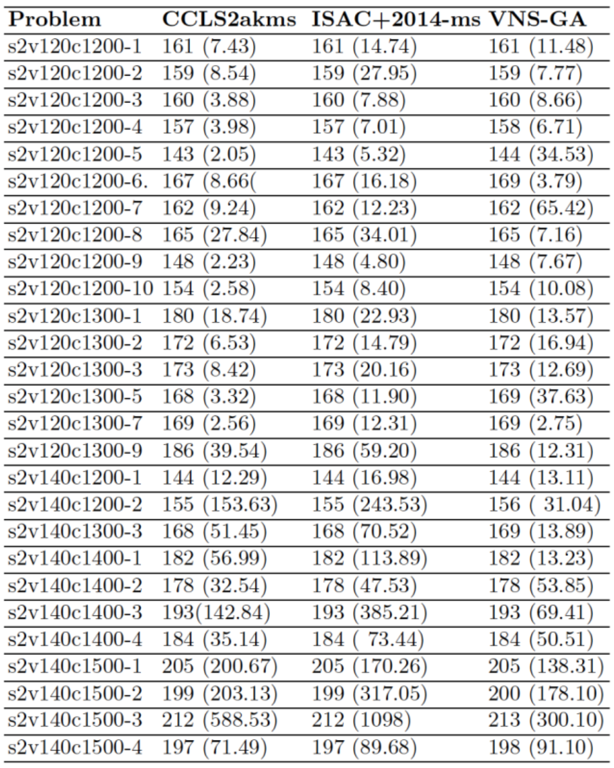
Table 5 compares VNS-GA with solvers used at the 2014 MAX-SAT competition. For each solver, the number of unsatisfied clauses is given while the number between parenthesis shows the running time in seconds. CCLS2akms and ISAC+2014-ms produce identical solution. The difference in performance depends on the time invested to produce such solutions. VNS-GA is found to be able to produce the same quality compared to these two solvers in 18 cases out of 27. The comparison against CCLS2akms shows that VNS-GA produces solutions at a lower cost in 8 cases. The time of VNS-GA ranges between 10% and 77% of the time of CCLS2akms. On the other hand, CCLS2akms happens to be faster than VNS-GA in the remaining cases. CCLS2akms was found between 31% and 86% of the time of VNS-GA. The comparison between VNS-GA and ISAC+2014-ms shows that VNS-GA is faster in 12 cases. The time of VNS-GA varies between 19% and 89% of the time of ISAC+2014-ms. The time of ISAC+2014-ms varies between 10% and 82% in the remaining 6 cases. Finally, VNS-GA gave solution of lower quality in 9 cases compared to these two solvers. The difference never exceeds 1one unsatisfied clause.

4 Conclusions

In this paper , a hybrid approach combining GA and VNS is proposed. VNS exploits the notion of neighbourhood change to escape local optima. The set of neighbourhoods proposed in this paper is based on one type of neighbourhood with varying size. Making the search

begins from the largest neighbourhood and moving systematically towards the smallest neighbourhood is a better strategy for conducting an efficient search.
The results reported in this paper shows that the variable neighbourhood search strategy can prevent GA from premature convergence. The results provided by of VNS-GA are better compared to that of GA. When compared to top ranked solvers, VNS-GA is capable of producing identical results while requiring the least amount of time for several test cases. At the present time, a better strategy for constructing the different neighbourhoods is under implementation. The method is based on grouping variables by taking into account the number of clauses they share in common rather than doing it randomly.

Table 5. Comparison of VNS-GA with CCLS2akms and ISAC+2014-ms



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 5. Comparing VNS-GA with state-of-the-art incomplete solvers. Numbers outside brackets indicate the number of unsolved clauses, while number inside brackets is the time in seconds.

References

[1] Cook, S. A: The complexity of theorem-proving procedures. Proceedings of the Third ACM Symposium on Theory of Computing, 151–158, (1971).
[2] Selman, B., Kautz, H. A., Cohen, B.: Noise Strategies for Improving Local Search. Proceedings of AAAI'94, pp. 337–343. MIT Press, (1994).

[3] McAllester, D., Selman, B. Kautz, H.: Evidence for invariants in local search. proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97), pp 321-326, (1997).

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

[5] Li,C.M., Huang, W.Q.: Diversification and determinism in local search for satisfiability. Proceedings of the Eighth International Conference on Theory and Applications of Satisfiability Testing (SAT-05), 569, Lecture Notes in Computer Science, 158–172, (2005).

[6] Jin-Kao, H., Lardeux, H., Saubion, F.: Evolutionary computing for the satisfiability problem. In Applications of Evolutionary Computing, volume 2611 of LNCS, pp. 258-267, University of Essex, (2003).

[7] Hansen, P., Jaumard, B., Mladenovic, N., Parreira, A.D.: Variable neighbourhood search for maximum weighted satisfiability problem. Technical Report G-2000-62, Les Cahiers du GERAD, Group for Research in Decision Analysis, 2000.

[8] Frank, J.: A study of genetic algorithms to find approximate solutions to hard 3CNF problem. In proceedings of Golden West international conference on artificial intelligence, (1994)

[9] Grissom, R. J., Kim, J. J..: Effect sizes for research: Univariate and multivariate applications. Routledge (2012).

[10] 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 Behavioral Statistics, 25(2), 101-132, (2000).

[11] Arcuri, A., Briand, L.: A Hitchhiker's Guide to Statistical Tests for Assessing Randomized Algorithms in Software Engineering. Technical report, Simula Research Laboratory, number 13/2011, (2011).