Advances and Applications in Discrete Mathematics

Published Online: April 2016
http://dx.doi.org/10.17654/DM017020125
Volume 17, Number 2, 2016, Pages 125-149 **ISSN: 0974-1658**

ENHANCED META-HEURISTICS WITH VARIABLE NEIGHBORHOOD SEARCH STRATEGY FOR COMBINATORIAL OPTIMIZATION PROBLEMS

Noureddine Bouhmala

Department of Maritime Technology and Innovation
Vestfold University College
Norway
e-mail: noureddine.bouhmala@hbv.no

Abstract

Variable neighborhood search (VNS) is a simple meta-heuristic that systematically changes the size and type of neighborhood during the search process in order to escape from local optima. In this paper, enhanced versions of tabu search and memetic algorithm with variable neighborhood search for combinatorial optimization problems are introduced. The set of constructed neighborhoods satisfies the property that each small neighborhood is a subset of a larger one. Most of the work published earlier on VNS starts from the first neighborhood and moves on to higher neighborhoods without controlling and adapting the ordering of neighborhood structures. The order in which the neighborhood structures have been selected in this paper during the search process offers a better mechanism for performing diversification and intensification. A set of industrial and random problems is used to test the effectiveness of the two enhanced meta-heuristics using the maximum satisfying problem as a test case.

Received: August 12, 2015; Revised: December 12, 2015; Accepted: December 18, 2015
2010 Mathematics Subject Classification: 68-XX, 68Txx, 68Wxx.
Keywords and phrases: maximum satisfiability problem, memetic algorithm, tabu search, variable neighborhood search.
Communicated by K. K. Azad

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 and brought benefits to our societies 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 [44], predicting hard drive failures to allow users to back up their data [31]. 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 [47]. AI methods have proved to provide better accuracy than statistical methods for the prediction of tumor behavior [33]. AI techniques have shown their superiority compared to logistic regression when predicting the childhood obesity [46] 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 [23]. 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 production [28]. 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 [12], production engineering issues [1], and drilling [3]. Memetic algorithms (MAs) and tabu search (TS) like other meta-heuristics offer the advantage of being flexible as

they can be applied to any problem (discrete or continuous). 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 [37]. On the other hand, TS which is based on explicit memory structures, uses memory to record the search trajectory and guide the search in order to consider both intensification and diversification. In TS, exploration is achieved by examining new candidate solutions by a tabu-restricted neighborhood search, while exploitation takes place when there is a move to the best of these candidates to start a new search cycle. The key components in meta-heuristic algorithms for optimization are local intensive exploitation and global diverse exploration, and their interaction can significantly affect the efficiency of a meta-heuristic algorithm. However, up to date there exists no simple rule for how to balance these important components. Mladenović and Hansen [27] have recently proposed a new meta-heuristic called *variable neighborhood search*. VNS overcomes local optimality using a combination of a local search with systematic changes of neighborhood. Unlike many standard meta-heuristics where only a single neighborhood is employed, VNS systematically changes different neighborhoods within a local search.

In this paper, two new versions of MA and TS enhanced with VNS are introduced. The resulting versions offer two main advantages which enable GA and TS to become much more powerful in the variable neighborhood context. The neighborhoods designed in VNS are constructed in a such manner that the process of switching from one neighborhood to another allows the possibility for both MA and TS to explore different regions in the search space in a structured manner while intensifying the search by exploiting the solutions from previous neighborhoods in order to reach better solutions.

The paper is organized as follows. Section 2 defines the maximum satisfiability problem. Sections 3 and 4 describe the enhanced versions of TS

and MA with variable neighborhood search together with the experimental results. Finally, Section 5 discusses the main conclusions and provides some guidelines for future work.

2. The Maximum Satisfiability Problem (MAX-SAT)

The MAX-SAT problem which is known to be NP-complete [10] is defined as follows. Given a set of *n* Boolean variables and a conjunctive normal form (CNF) of a set of *m* disjunctive clauses of literals, where each literal is a variable or its negation which takes one of the two values True or False and a positive constant *k*, the task is to determine whether there exists an assignment of truth values of the variables that satisfies the maximum number *k* of clauses. MAX-SAT still deserves much research attention from a wider community of researchers due to its theoretical and practical importance. MAX-SAT is a widely used modeling framework for simulating complex systems that turn out to be of combinatorial nature. Examples include model-checking [4] of finite state systems, design debugging [38], AI planning [34] to name just a few. Different state-of-the-art local search algorithms for solving MAX-SAT have been developed. Several of these algorithms are enhanced versions of earliest GSAT [36] and WalkSAT [35] algorithms. Examples include GSAT/Tabu [25], Walk-SAT/Tabu [26], Novelty+ [18] and R-Novelty+ [24] heuristics, variable and clause weighting algorithms [19, 32, 41], dynamic parameter tuning algorithms [17], adaptive memory-based local search hybrid approaches (GASAT) [20], larger neighborhood search algorithms [45], learning automata [15], adaptive memory-based local search algorithms [48], Iterated Robust Tabu Search (IRoTS) [39], new algorithms based on a new diversification scheme to prevent cycling [6-8], variable neighborhood search [5], CCLS which is a local search solver based on the configuration checking strategy [22].

3. Variable Neighborhood Search Tabu-based Algorithm (VNS-TS)

3.1. VNS-TS in detail

Variable neighborhood search [5, 27] is a relatively recent meta-heuristic

jointly which has been applied to a wide variety of combinatorial optimization problems such as feature selection in data mining [30], scheduling [21], vehicle routing problem [13] and maximum satisfiability problem [5]. Unlike many standard meta-heuristics where only a single neighborhood is employed, 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 different search areas which are defined by different neighborhood structures. Tabu search algorithm was proposed by Glover [14]. The main feature of the algorithm is the ability to avoid returning in a previous state by keeping a trace of the optimization history. Algorithm 1 shows TS working in a variable neighborhood context. Let L denote the set of variables of the MAX-SAT problem to be solved. The first phase of the algorithm consists in constructing a set of neighborhood satisfying the following property: $N_1(x) \subset N_2(x) \subset \cdots N_{k_{\max}}(x)$. 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 $\rightarrow$ False or False $\rightarrow$ 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 v_i and v_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 (lines 3, 4, 5 of Algorithm 1). A random initial solution of the problem is computed (line 7 of Algorithm 1). Then, during each pass of the algorithm, given the current solution, one examines its corresponding neighborhood and chooses to move to the solution that most improves the objective function. At the end of each

pass, the variable with the highest gain is selected (lines 10-17 of Algorithm 1) (break ties randomly). The gain of a variable v_i is defined as the number of clauses that would be unsatisfied if v_i is flipped. To avoid getting stuck in a local minimum, historical information from the last *j* iterations is used. The value *j* may be fixed or a variable that depends on the search. The set of moves determined by this information forms a tabu list. Hence, the method has a short term memory remembering which trajectories have been recently explored. To prevent the method from cycling between the same solutions, one forbids the reverse of any move contained in the tabu list. The algorithm proceeds by choosing a random unsatisfied clause (line 11 of Algorithm 1). Thereafter, a non-tabu and unvisited variable is chosen randomly and flipped (lines 12, 13 of Algorithm 1). The tabu list is updated before the start of every new pass (line 17 of Algorithm 1). The selected variable during each pass is inserted into the tabu list together with a tabu-value that will determine the number of iterations it will remain tabu. During each pass, the tabu-value assigned to each tabu variable is decremented by 1. When tabu-value reaches the value 0, its corresponding variable becomes non-tabu. The effectiveness of VNS is strongly affected by the ordering in which a given type of neighborhood is considered. Both the choice and the order of neighborhood structures are critical for the performance of the algorithm. Most of the work published earlier on VNS starts from the first neighborhood and moves on to higher neighborhoods without controlling and adapting the ordering of neighborhood structures. The strategy adopted in this work is to let the search process start 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 better mechanism for performing diversification and intensification. This issue will be explained when presenting the experimental results.

Algorithm 1: Variable Neighborhood Search Tabu-based Algorithm

```
input  : Problem P
output: Solution S(P)
/*Select the set of neighborhood structures N_k (k=1,2,....k_max) /* ;
k := 0 ;
while (Not reached the desired set of neighborhood) do
    N_{k+1}:= Define (N_k) ;
    k := k + 1 ;
begin
    Generate a random initial solution from the k^th_max neighborhood ;
    while (k >0) do
        while (Not stop) do
            for i ← 1 to NumberOfUnsatisfiedClauses do
                C_k ← A randomly chosen unsatisfied clauses ;
                l_j ← A randomly chosen unvisited and a non tabu variable;
                l_j ← Flip (l_j) ;
                G_j ← ComputeGain(l_j) ;
                M(l_j) ← Mark l_j visited ;
            l_k ← Choosing the variable with highest gain ;
            Update tabu conditions () ;
end
```

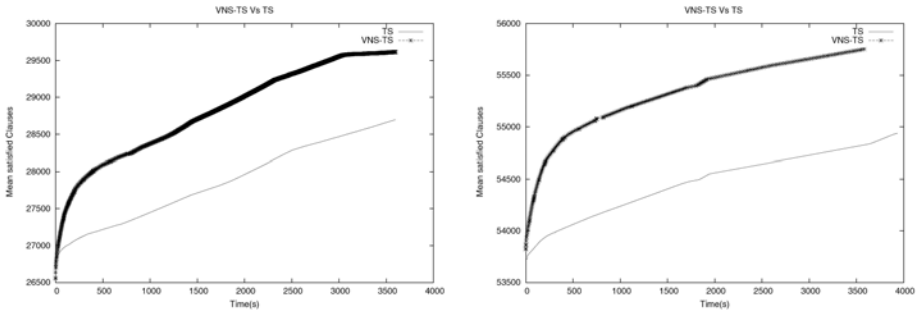

Figure 1. VNS-TS vs TS: (left) alu4mul.miter.shuffled-as.sat03-344.cnf: $|V| = 4736\ |C| = 30465$, (right) C6288mul.miter.shuffled-as.sat03-346.cnf: $|V| = 9540\ |C| = 61421$. Evolution of the mean satisfied clause over time.

3.2. Experimental results

The performance of VNS-TS is evaluated against TS using a set of real industrial problems. This set is taken from the sixth MAX-SAT 2011 organized as an affiliated event of the Fourteenth International Conference on Theory and Applications of Satisfiability Testing (SAT-2011). Due to the

randomization nature of both algorithms, each problem instance was run 100 times with a cut-off parameter (max-time) set to 60 minutes. The symbols $|V|$ and $|C|$ denote, respectively, the number of variables and the number of clauses. 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 length of tabu list is set to be equal to: $0.01875 \times n + 2.8125$ as proposed in [25] where n is the number of variables in the problem. The cardinality of the neighborhood $k_{\max}$ 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 variables will lead to $k_{\max}$ equals to 3). TS is assumed to have reached convergence and switch to a smaller neighborhood if the best solution remains unchanged during 50 consecutive iterations. Figure 1 shows the development of the mean satisfied clause for both algorithms. Both algorithms start from nearly identical initial solutions. The plots show immediately the dramatic improvement obtained with VNS. The curves show no crossover implying that VNS-TS dominates TS. The mean number of unsatisfied clauses improves rapidly at first and continues to improve before it flattens off as we mount the plateau marking the start of the second phase. The plateau spans a region in the search space where flips typically leave the best assignment unchanged, and occurs more specifically once the refinement reaches the default neighborhood. Comparing VNS-TS against TS, VNS-TS is far better than TS, making it the clear leading algorithm. The key success behind the efficiency of VNS-TS relies on the VNS context. VNS-TS draws its strength from coupling the optimization process across different neighborhoods. This paradigm offers two main advantages which enables TS to become much more powerful in the VNS context. During the refinement phase, TS applies a local a transformation (i.e. a move) within the neighborhood (i.e. the set of solutions that can be reached from the current one) of the current solution to generate a new one. The variety of neighborhoods 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 TS to view a cluster of variables

as a single entity, the search becomes guided and restricted to only those configurations in the solution space in which the variables grouped within a cluster are assigned the same value. As one moves from a larger neighborhood to a smaller one, the size of the clusters varies and the size of the neighborhood becomes adaptive and allows the possibility of exploring different regions in the search space while intensifying the search by exploiting the solutions from previous neighborhoods in order to reach better solutions. Tables 1-2 show the results of comparing VNS-TS against TS. The tables show that VNS-TS shows a better asymptotic convergence compared to TS in 32 cases out of 50 with an improvement lying within 9%. The cases where TS gives better results than VNS-TS, the difference in quality does not exceed 2%.

Table 1. Comparison of VNS-TS against TS

Instances	$\|V\|$	$\|C\|$	**TS**	**VNS-TS**
aim-200-3_4 − *yes*1 − 2	200	680	673	663
alu4mul.miter.shuffled-as.sat03-344	4736	30465	28698	29613
am4-4.shuffled-as.sat03-360	433	1458	1457	1404
am5-5.shuffled-as.sat03-361	1076	3677	3610	3544
am6-6.shuffled-as.sat03-362	2269	7814	7589	7493
am7-7.shuffled-as.sat03-363	4264	14751	14032	14055
am8-8.shuffled-as.sat03-364	7361	25538	22416	23249
am9-9.shuffled-as.sat03-365	11908	41393	35318	35933
bw-large.c	3016	50457	46631	50450
bw-large.d	6325	131973	107713	118272
c3540mul.miter.shuffled-as.sat03-345	5248	33199	30958	31734
c6288mul.miter.shuffled-as.sat03-346	9540	61421	54941	55752
c7552mul.miter.shuffled-as.sat03-347	11282	69529	62258	62450
cnt10.shuffled-as.sat03-418	20470	68561	58204	58683
comb1.shuffled-as.sat03-419	5910	16804	14653	15505
comb3.shuffled-as.sat03-421	4774	16331	14968	15556
c880mul.miter.shuffled-as.sat03-348	1612	9373	9342	9188
dalumul.miter.shuffled-as.sat03-349	9426	59991	54188	54940
ewddr2-10-by-5-1	21800	118607	104713	104835
f2clk40.shuffled-as.sat03-424	27568	80439	63674	64028

ferry8.shuffled-as.sat03-384	1918	12311	12306	12116
ferry8u.shuffled-as.sat03-385	1875	11915	11909	11728
ferry9.shuffled-as.sat03-386	2410	16209	16197	15954
ferry9u.shuffled-as.sat03-387	2342	15747	15735	15502
ferry10.shuffled-as.sat03-378	2958	20791	20768	20410

Table 2. Comparison of VNS-TS against TS

Instances	$\|V\|$	$\|C\|$	**TS**	**VNS-TS**
ferry11.shuffled-as.sat03-380	3562	26105	25355	25656
ferry12u.shuffled-as.sat03-383	4133	31515	30069	30866
frg1mul.miter.shuffled-as.sat03-351	3230	20575	20360	20103
frg2mul.miter.shuffled-as.sat03-352	10316	62943	56680	57198
gripper13u.shuffled-as.sat03-395	4268	38965	37229	38337
gripper14.shuffled-as.sat03-396	4758	45056	42412	43167
gripper14u.shuffled-as.sat03-397	4584	43390	40967	42688
homer17.shuffled-as.sat03-428	286	1742	1738	1731
homer18.shuffled-as.sat03-429	308	2030	2014	2014
homer19.shuffled-as.sat03-430	330	2340	2332	2313
homer20.shuffled-as.sat03-431	440	4220	4202	4184
i8mul.miter.shuffled-as.sat03-354	14524	91139	81541	81821
i10mul.miter.shuffled-as.sat03-353	12998	77941	69475	70332
k2mul.miter.shuffled-as.sat03-355	11680	74581	66791	67367
logistics.d	4713	21991	19522	20952
mot-comb2-red-gate-0.dimacs.seq.filtered	5484	13894	11219	11391
motcomb3-red-gate-0.dimacs.seq.filtered	11265	29520	23249	23460
qg6-11	1331	49204	49104	49203
qg6-12	1728	69931	69786	69930
rotmul.miter.shuffled-as.sat03-356	5980	35229	32469	32819
term1mul.miter.shuffled-as.sat03-357	3504	22229	21809	21664
vdamul.miter.shuffled-as.sat03-358	5444	34509	32320	32656
x1mul.miter.shuffled-as.sat03-359	8760	55571	50310	50861

4. Variable Neighborhood Search Memetic-based Algorithm (VNS-MA)

4.1. VNS-MA in detail

Memetic algorithms (MAs) [29] are population-based meta-heuristic optimization methods inspired by biological evolution. They simultaneously examine and manipulate a set of possible solutions. These are usually randomly initialized across the search space, although heuristics may also be used. In the context of MAs, the objective function assigning values to each solution is termed a fitness function, and is used to guide the search. MAs are characterized by the importance they place on recombination (crossover, mutations) of solutions. In a well-designed MA, crossover is able to combine successful sub-solutions (parent solutions) to form new solutions (off-springs) with the best of both. In other words, once a sub-solution is discovered by any member of the population, it can be propagated to the rest of the population without needing to be rediscovered. This is considered to be the primary source of novel solutions in a MA, though small mutations are usually used as well. Hence, designers of memetic algorithms are careful to design the problem representation and crossover operator to work well together. After the initialization, four different components are iteratively performed until a convergence criterion is reached: selection, reproduction (which encompasses recombination and mutation), replacement, and local search. Algorithm 2 shows the general outline of a MA combined with a variable neighborhood search.

Algorithm 2: Variable Neighborhood Search Memetic-based Algorithm

input : Problem in CNF Format
output: Number of unsatisfied clauses
1 /*Select the set of neighborhood structures N_k (k=1,2,....k_{max}) /* ;
2 k := 0 ;
3 **while** *(Not reached the desired set of neighborhood)* **do**
4 N_{k+1}:= Define (N_k) ;
5 $k := k + 1$;
6 /*Generate a random initial population from the k^{th}_{max} neighborhood */ ;
7 $S^k_{current} \leftarrow RandomSolution$ () ;
8 Evaluate the fitness of each individual in the population $S^k_{current}$;
9 **while** *(k >0)* **do**
10 **while** *(Not Stop)* **do**
11 Select individuals from $S^k_{current}$ according to a scheme to reproduce ;
12 Breed if necessary each selected pairs from $S^k_{current}$ through crossover;
13 Apply mutation operator if necessary to each offspring in $S^k_{current}$;
14 Apply local search to each individual ;
15 Evaluate the fitness of the intermediate population ;
16 $S^k_{new} \leftarrow S^k_{current}$;
17 $S^{k-1}_{current} \leftarrow ChangeNeighborhood(S^k_{new})$;
18 $k \leftarrow k - 1$;
19 return (Best-Individual (S^{k+1}_{new}));

- **Representation and search space**

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

- **Fitness function**

The notion of fitness is fundamental to the application of memetic algorithms. It is a numerical value that expresses the performance of an individual (solution) so that different individuals can be compared. The fitness of a chromosome (individual) in the population is equal to the number of clauses that are unsatisfied by the truth assignment represented by the chromosome. All the individuals of the initial population are evaluated and assigned a fitness (line 8 of Algorithm 2).

• **Construction of neighborhoods**

The procedure for generating the different neighborhoods (lines 3, 4, 5 of Algorithm 2) is similar to the one describe for TS in Subsection 3.1.

• **Optimization cycle**

The following four components describe an optimization cycle performed by a MA within a specific neighborhood.

- **Choosing starting neighborhood**

VNS-MA starts the search from the largest neighborhood $N_{k_{\max}}$ (lines 9-16 of Algorithm 2) as VNS-TS and continues to move towards smaller neighborhood structures.

- **Selection**

The selection component aims to determine the candidate solutions to be used to create new solutions or off-springs. Selection often operates in relation with the best fitness value. The best fitness value reflects typically amounts the extent to which the solution maximizes/minimizes the objective function. In this work, selection is performed in a random manner. The individuals are visited in random order. An unmatched individual i_k is matched randomly with an unmatched individual i_l (line 11 of Algorithm 2).

- **Crossover operator**

The task of the crossover operator (line 12 of Algorithm 2) 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 chromosomes between these points to generate two new offspring. Recombination can be defined as a process in which a set of configurations (solutions referred as parents) undergoes a transformation to create a set of configurations (referred as offspring). The creation of these descendants involves the location and combinations of features extracted from the parents. The reasons behind choosing the two-point crossover are

the results presented in [43] where the difference between the different crossovers is not significant when the problem to be solved is hard. The work conducted in [40] shows that the two-point crossover is more effective when the problem at hand is difficult to solve.

- Mutation

The purpose of mutation (line 13 of Algorithm 2) 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 from 1 to 0 if the probability test is passed. In case of a large neighborhood $(k > 1)$, the mutation is applied to a cluster of variables. The mutation probability ensures that, theoretically, every region of the search space is explored. If, on the other hand, mutation is applied to all genes, the evolutionary process will degenerate into a random search with no benefits of the information gathered in preceding generations. The mutation operator prevents the searching process form being trapped into local optimum while adding to the diversity of the population and thereby increasing the likelihood that the algorithm will generate individuals with better fitness values.

- Local search

By introducing local search at this level (line 14 of Algorithm 2), the search within promising areas is intensified. The local search should be designed 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. In order to balance the exploration against exploitation a fast and simple heuristic is used for one iteration during which it seeks for the new variable-value assignment which best decreases the number of unsatisfied clauses is identified.

- **Replacement**

The replacement component (line 16 of Algorithm 2) 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 $P_{Selection_i} = f_i \Big/ \sum_1^N f_i$, where f_i is the fitness of individual *i* (line 15 of Algorithm 2).

- **Move to a new neighborhood**

Once MA 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 (line 17 of Algorithm 2) is simple; if a cluster $c_i \in N_m$ is assigned the value of true, then the merged pair of clusters that it represents, $c_j, c_k \in N_{m-1}$ are also assigned the true value.

4.2. Experimental results

Table 3. Benchmark instances

Cases	Instance	Number of variables	Number of clauses
1	bmc-ibm-1.cnf	9 658	55 870
2	bmc-ibm-2.cnf	3 628	14 468
3	bmc-ibm-3.cnf	14 930	72 106

4	bmc-ibm-5.cnf	9 396	41 207
5	bmc-ibm-7.cnf	8 710	39 774
6	bmc-galileo-8.cnf	58 074	294 821
7	bmc-galileo-9.cnf	63 624	326 999
8	bmc-ibm-10.cnf	61 088	334 861
9	bmc-ibm-11.cnf	32 109	150 027
10	bmc-ibm-12.cnf	39 598	194 778
11	bmc-ibm-13.cnf	13 215	65 278
12	g125.18.cnf	2 250	70 163
13	g125.17.cnf	2 125	66 272
14	g250.15.cnf	3 750	233 965
15	g250.29.cnf	7 250	454 622

The performance of the VNS-MA is evaluated against MA using a set of instances taken from real industrial problems (bmc instances) and random graph coloring problems (g125-g250). This set is taken from (http://www.informatik.tu-darmstadt.de/AI/SATLIB). Table 3 shows the instances used in the experiment. IBM SPSS Statistics version 19 was used for statistical analysis. Due to the randomization nature of the algorithms, each problem instance was run 100 times with a cut-off parameter (max-time) set to 15 minute. The 100 runs were chosen because pilot runs had shown the size of the difference to be so large that 100 runs were enough for an acceptable statistical power $(power > .95)$, this is in accordance with the suggestions given in a recent report on statistical testing of randomized algorithms [2].

The following parameters have been fixed experimentally and are listed below:

- Mutation probability = 0.1.
- Population size = 50.
- Stopping criteria within a specific neighborhood: 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 new neighborhood.

Table 4. VNS-MA vs MA: mean, standard deviation and range of unsolved clauses

#Case	VNS-MA		MA	
	Mean (SD)	**Range**	**Mean (SD)**	**Range**
1	3872.8 (233.7)	3632-4830	5700.5 (427.6)	5184-7773
2	157.8 (8.1)	137-185	234.6 (16.1)	190-273
3	3320.9 (279.1)	2911-4187	10049.3 (356.8)	9547-11559
4	1125.8 (138.1)	976-1675	3282.5 (276.0)	3024-4651
5	1475.9 (101.3)	1167-1668	3107.1(152.3)	2870-4044
6	12232.9 (2464)	9809-23178	51062.3 (814.1)	49375-53517
7	14412.6 (2877.1)	11318-22705	57301.2 (877.1)	55292-59723
8	2206.1 (1117.6)	20621-25275	63720.2 (475.3)	62575-64626
9	13730.9 (547.5)	13045-15413	26207.8 (353.7)	25461-27311
10	17477.0 (410.7)	17079-18908	35289.7 (440.3)	34363-36871
11	3875.7 (273.4)	3529-5055	8233.4 (384.2)	7886-10080
12	37.8 (3.5)	29-48	323.9 (47.4)	225-481
13	40.7 (3.5)	30-50	154.2 (20.7)	92-222
14	193.7 (30.9)	173-425	29565.3 (1068.7)	27587-33040
15	214.8 (64.3)	177-637	85410.6 (1759.2)	80704-89626

Table 5. VNS-MA vs MA: mean difference and the 99% confidence interval

#Case	Bootstrapped inferential statistics[a]	
	MD [99%CI]	**Cohen's delta**
1	1895.2 [1827.6, 1962.8]	5.3
2	76.8 [71.9, 81.3]	6.02
3	6728.4 [6610.1, 6844.2]	21.01
4	2156.7 [2082.4, 2238.7]	9.88
5	1631.2 [1585.7, 1682.1]	12.61
6	38829.4 [38119.8, 39450.1]	21.16
7	42888.6 [42100.6, 43645.6]	20.17
8	41713.8 [41396.1, 42020.2]	48.57
9	12476.9 [12314.7, 12614.4]	27.07
10	17812.7 [17656.6, 17963]	41.83

11	4446.7 [4329.1, 4571.1]	13.33
12	286.1 [274.0, 298.4]	8.51
13	113.6 [108.3, 119.0]	7.66
14	29371.6 [29097.1, 29650.6]	38.85
15	85195.8 [84753.5, 85638.3]	68.44

a: Based upon 10000 bootstrapped samples.

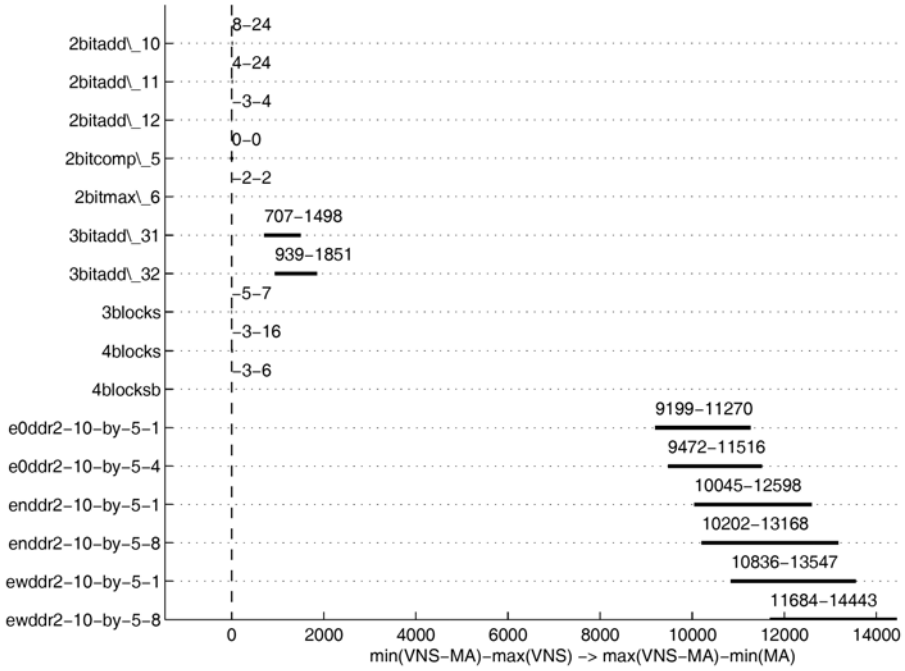

Figure 2. The range from observed minimum $(\mathrm{Min}_{\text{VNS-MA}} - \mathrm{Max}_{\text{MA}})$ to maximum $(\mathrm{Max}_{\text{VNS-MA}} - \mathrm{Min}_{\text{MA}})$ differences for each case.

The comparison of the MA and the VNS-MA algorithms for each instance is shown in Table 4. The table shows the mean, standard deviation and the range of unsolved clauses for VNS-MA and MA for each instance. Table 5 also shows the mean difference and the 99% confidence interval for the difference between MA and VNS-MA. Bootstrapping [11] was chosen due to the lack of knowledge regarding the statistical distribution of the

underlying population. Bootstrapping is a non-parametric re-sampling method that can be used to estimate statistical parameters without strong assumptions about the distribution of the populations from which the data samples have been drawn. Bootstrapping works by random sampling with replacement from the observed empirical distribution. This re-sampling procedure creates new data sets that are estimates of the original sampling distribution. Because of the random re-sampling each re-sample would be likely to be similar but slightly different from the original empirical distribution. Each sample would contain a largely similar distribution as the original sample, but with some observations missing and other observations present several times. This procedure allows for an evaluation of the sampling variability inherent in the observed data set and allows for a computer-intensive but conceptually simple estimation of statistical parameters that otherwise would be difficult to calculate using standard parametric estimation techniques [16]. As can be observed from Table 4, there is no overlap between the ranges of MA and VNS-MA. VNS-MA completely dominates MA in all instances. The performance of VNS-MA is strengthened by the fact that none of the confidence intervals for the mean difference between VNS-MA and MA contains zero (0)[1]. The standardized effect size measure Cohen's delta [9] is very high[2] and indicates that the effect of adding a variable neighborhood search strategy to a memetic algorithm leads to a great improvement of the solutions. To create a visualization of the difference between VNS-MA and MA, we computed the observed minimum $(\text{Min}_{\text{VNS-MA}} - \text{Max}_{\text{MA}})$ and maximum $(\text{Max}_{\text{VNS-MA}}$

[1]We used mean-based statistics to evaluate the difference between the two algorithms because there was no overlap between the compared data sets. This is not in accordance with recent recommendations [2] but was found to be necessary because non-parametric rank-based tests of significance such as Mann-Whitney *U*-test would not entail extra information as non-overlapping data sets would give identical solutions for different instances.

[2]The mean-based effects size measure Cohen's delta [9] was used because we used mean-based *t*-tests to evaluate the difference between the data sets. We also included Cohen's delta because the common language effect size measure $\hat{A}_{12}$ [42] would entail no extra information as one model would always be better, e.g., $P_{(VNS-MA>MA)} = 1$ for all instances.

$-\text{Min}_{\text{MA}}$) differences in solved clauses, see Figure 2. The range from minimum to maximum differences does not include zero, hence showing that VNS-MA produced uniformly better solutions than MA in all instances. The relationship between the size of problem spaces and the output from the VNS-MA and MA is also an interesting parameter for evaluating the success of the variable neighborhood concept. The correlation between the size of the problem spaces (number of clauses) and the difference in mean solved clauses for VNS-MA and MA (mean percentage solved clauses for VNS-MA minus the mean percentage solved clauses for MA, see Table 1 for the data) was very high $(r = .969; \rho < .001)$ indicating that enhancing the memetic algorithm with a variable neighborhood search leads to a improvement that increases for larger problem spaces. Even though it seems like there is a positive correlation between the sample space and the quality of solutions we cannot determine the exact shape of this relationship because of the small sample space $(n = 15)$. 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 VNS-MA was much lower $(t(15) =$ 3.059, = 2.041-8, 95% CI [1.163-8, 2.714-8], $p = .008, r = .633)$ than for the MA $(t(15) = 10.067,$ = 9.341-7, 95% CI [8.232-7, 1.04-6], $p < .001, r = .937)$ indicating that VNS-MA is less affected by the size of the problem than MA.

5. Conclusion

VNS follows a simple principle that is based on systematic changes of neighborhood within the search. In this work, enhanced versions of TS and MA with VNS are introduced. The set of neighborhoods proposed in this paper is designed so that each generated neighborhood will serve as the basis for generating a larger one. By allowing TS and MA to view a cluster of variables as a single entity, the search becomes guided and restricted to only

those configurations in the solution space in which the variables grouped within a cluster are assigned the same value. As the size of the clusters varies from one neighborhood to another, the neighborhood becomes adaptive and allows the possibility of exploring different regions in the search space while intensifying the search by exploiting the solutions from previous neighborhoods in order to reach better solutions. Starting the search from the largest neighborhood and moving systematically towards the smallest neighborhood provides a better strategy to cope with the diversification and intensification search. The results obtained ensure that VNS greatly improves MA and TS and always returns a better solution for the equivalent run-time compared to TS and MA. Future work aims at investigating other strategies for generating neighborhoods satisfying the property introduced in this paper and identifying other parameters which may improve their performances.

Acknowledgement

The author thanks the anonymous referees for their valuable suggestions which let to the improvement of the manuscript.

References

[1] C. Alimonti and G. Falcone, 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, Proceedings, SPE Annual Technical Conference and Exhibition held in San Antonio, Texas, 2002.

[2] A. Arcuri and L. Briand, A hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering, Technical Report, Simula Research Laboratory, number 13/2011, 2011.

[3] R. S. Balch, D. M. Hart, W. W. Weiss and R. F. Broadhead, Regional data analysis to better predict drilling success, brushy canyon formation, Delaware Basin, New Mexico, SPE 75145, Proceedings, SPE/DOE Improved Oil Recovery Symposium held in Tulsa, Oklahoma, 2002.

[4] A. Biere, A. Cimatti, E. Clarke and Y. Zhu, Symbolic model checking without BDDs, Tools and Algorithms for the Construction and Analysis of Systems, Springer, 1999, pp. 193-207

[5] N. Bouhmala, A variable neighborhood Walksat-based algorithm for MAX-SAT problems, The Scientific World Journal 2014 (2014), Article ID 798323, 11 pp. http://dx.doi.org/10.1155/2014/798323.

[6] S. Cai, C. Luo and K. Su, CCASat: solver description, Proc. of SAT Challenge: Solver and Benchmark Descriptions, 2012, pp. 13-14.

[7] S. Cai and K. Su, Configuration checking with aspiration in local search for SAT, Proc. of AAAI-12, 2012, pp. 434-440.

[8] S. Cai, K. Su and A. Sattar, Local search with edge weighting and configuration checking heuristics for minimum vertex cover, Artif. Intell. 175(9-10) (2011), 1672-1696.

[9] J. Cohen, Statistical Power Analysis for the Behavioral Sciences, 2nd ed., Lawrence Erlbaum, 1988.

[10] S. A. Cook, The complexity of theorem-proving procedures, Proceedings of the Third ACM Symposium on Theory of Computing, 1971, pp. 151-158.

[11] B. Efron, The Jacknife, the Bootstrap and Other Resampling Plans, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1982.

[12] J. Finol, C. Romero and P. Romero, An intelligent identification method of fuzzy models and its applications to inversion of NMR logging data, SPE 77605, Proceedings, SPE Annual Technical Conference and Exhibition held in San Antonio, 2002.

[13] K. Fleszar, I. H. Osman and K. Hindli, A variable neighbourhood search algorithm for the open vehicle routing problem, Euro. J. Oper. Res. 195(3) (2009), 803-809.

[14] F. Glover, Tabu search - Part 1, ORSA J. Comput. 1(3) (1989), 190-206.

[15] O. C. Granmo and N. Bouhmala, Solving the satisfiability problem using finite learning automata, Inter. J. Comp. Sci. Appl. 4(3) (2007), 15-29.

[16] J. S. Haukoos and R. J. Lewis, Advanced statistics: bootstrapping confidence intervals for statistics with difficult distributions, Acad. Emergency Medicine 12(4) (2005), 360-365.

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

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

[19] F. Hutter, D. Tompkins and H. Hoos, Scaling and probabilistic smoothing: efficient dynamic local search for SAT, Lecture Notes in Computer Science:

Principles and Practice of Constraint Programming, P. van Hentenryck, ed., Vol. 2470, 2002, pp. 233-248.

[20] H. Jin-Kao, F. Lardeux and F. Saubion, Evolutionary computing for the satisfiability problem, Lecture Notes in Computer Science: Applications of Evolutionary Computing, G. R. Raidl et al., eds., Vol. 2611, 2003, pp. 258-267.

[21] D. Lei and X. Guo, Variable neighbourhood search for dual-resource constrained flexible job shop scheduling, Inter. J. Prod. Res. 52(9) (2014), 2519-2529. DOI: 10.1080/00207543.2013.849822.

[22] C. Luo, S. Cai, W. Wu, Z. Jie and K. Su, CCLS: an efficient local search algorithm for weighted maximum satisfiability, IEEE Trans. Computers 64(7) (2015), 1830-1843.

[23] H. Lin, M. Taylor and R. Zito, A review of travel-time prediction in transport and logistics, Proceedings of the Eastern Asia Society for Transportation Studies, Vol. 5, 2005, pp. 1433-1448.

[24] C. M. Li and W. Q. Huang, Diversification and determinism in local search for satisfiability, Lecture Notes in Computer Science: Theory and Applications of Satisfiability Testing (SAT-05), F. Bacchus and T. Walsh, eds., Vol. 3569, 2005, pp. 158-172.

[25] B. Mazure, L. Saïs and E. Grégoire, Tabu search for SAT, Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97), 1997, pp. 281-285.

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

[27] N. Mladenović and P. Hansen, Variable neighborhood search, Comp. Oper. Res. 24 (1997), 1097-1100.

[28] S. Mohaghegh, Virtual-intelligence applications in petroleum engineering: Part 1 – Artificial neural networks, J. Petroleum Tech. 52(09) (2000), 64-73.

[29] P. Moscato, On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms, Technical Report C3P Report 826, Caltech Concurrent Computation Program, California Institute of Technology, 1989.

[30] A. Mucherino and L. Liberti, A VNS-based heuristic for feature selection in data mining, Hybrid Metaheuristics, Series: Studies in Computational Intelligence, Vol. 434, 2013, pp. 353-368.

[31] J. F. Murray, G. F. Huges and K. K. Delgado, Machine learning methods for predicting failures in hard drives: a multiple-instance application, J. Mach. Learn. Res. 6 (2005), 783-816.

[32] S. Prestwich, Random walk with continuously smoothed variable weights, Lecture Notes in Computer Science: Theory and Applications of Satisfiability Testing (SAT-05), F. Bacchus and T. Walsh, eds., Vol. 3569, 2005, pp. 203-215.

[33] K. N. Qureshi, R. N. G. Naguib, F. C. Hamdy, D. E. Neal and J. K. Mellon, Neural network analysis of clinicopathological and molecular markers in bladder cancer, J. Urology 163 (2000), 630-633.

[34] J. Rintanen, K. Heljanko and I. Niemel, Planning as satisfiability: parallel plans and algorithms for plan search, Artif. Intell. 170(12-13) (2006), 1031-1080.

[35] B. Selman, H. A. Kautz and B. Cohen, Noise strategies for improving local search, Proceedings of AAAI'94, MIT Press, 1994, pp. 337-343.

[36] B. Selman, H. Levesque and D. Mitchell, A new method for solving hard satisfiability problems, Proceedings of AAA'92, 1992, pp. 440-446.

[37] W. Sheng, X. Liu and M. Fairhurst, A niching memetic algorithm for simultaneous clustering and feature selection, IEEE Trans. Knowledge and Data Engineering 20(7) (2008), 868-879.

[38] A. Smith, A. G. Veneris, M. F. Ali and A. Viglas, Fault diagnosis and logic debugging using Boolean satisfiability, IEEE Trans. Computer-Aided Design 24(10) (2005), 1606-1621.

[39] K. Smyth, H. H. Hoos and T. Stützle, Iterated robust tabu search for MAX-SAT, Advances in Artificial Intelligence, Series: Lecture Notes in Computer Science, Vol. 2671, 2003, pp. 129-144.

[40] W. Spears, Adapting crossover in evolutionary algorithms, Proc. of the Fourth Annual Conference on Evolutionary Programming, MIT Press, 1995, pp. 367-384.

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

[42] A. Vargha and H. D. Delaney, A critique and improvement of the CL common language effect size statistics of McGraw and Wong, J. Edu. Behav. Stat. 25(2) (2000), 101-132.

[43] D. Vrajitoru, Genetic programming operators applied to genetic algorithms, Proceedings of the Genetic and Evolutionary Computation Conference, Orlando, FL, Morgan Kaufmann Publishers, 1999, pp. 686-693.

[44] S. Whiteson and D. Whiteson, Machine learning for event selection in high energy physics, Eng. Appl. Artif. Intell. 22 (2004), 1203-1217.

[45] M. Yagiura and T. Ibaraki, Efficient 2 and 3-flip neighborhood search algorithms for the MAX SAT: experimental evaluation, J. Heuristics 7(5) (2001), 423-442.

[46] S. Zhang, C. Tjortjis, X. Zeng, H. Qiao, I. Buchan and J. Keane, Comparing data mining methods with logistic regression in childhood obesity prediction, J. Inform. Sys. Frontiers 11(4) (2009), 449-460.

[47] D. Zhang and J. J. P. Tsai, Machine learning and software engineering, Software Quality J. 11(2) (2003), 87-119.

[48] Z. Lü and J.-K. Hao, Adaptive memory-based local search for MAX-SAT, Appl. Soft Comput. 12(8) (2012), 2063-2071.