Iyad Abu Doush*, Amal Lutfi Quran, Mohammed Azmi Al-Betar
and Mohammed A. Awadallah

MAX-SAT Problem using Hybrid Harmony Search Algorithm

DOI 10.1515/jisys-2016-0129
Received August 1, 2016; previously published online May 4, 2017.

Abstract: Maximum Satisfiability problem is an optimization variant of the Satisfiability problem (SAT) denoted as MAX-SAT. The aim of this problem is to find Boolean variable assignment that maximizes the number of satisfied clauses in the Boolean formula. In case the number of variables per clause is equal or greater than three, then this problem is considered NP-complete. Hence, many researchers have developed techniques to deal with MAX-SAT. In this paper, we investigate the impact of different hybrid versions of binary harmony search (HS) algorithm on solving MAX 3-SAT problem. Therefore, we propose two novel hybrid binary HS algorithms. The first hybridizes Flip heuristic with HS, and the second uses Tabu search combined with Flip heuristic. Furthermore, a distinguished feature of our proposed approaches is using an objective function that is updated dynamically based on the stepwise adaptation of weights (SAW) mechanism to evaluate the MAX-SAT solution using the proposed hybrid versions. The performance of the proposed approaches is evaluated over standard MAX-SAT benchmarks, and the results are compared with six evolutionary algorithms and three stochastic local search algorithms. The obtained results are competitive and show that the proposed novel approaches are effective.

Keywords: Maximum satisfiability problem, harmony search, local search, optimization, 3SAT problem, evolutionary algorithms, MAX-SAT problem, metaheuristic.

1 Introduction

The combinatorial optimization is the core of several research areas including operational research, computer science, discrete mathematics, and many other domains. This field tries to solve various combinatorial optimization problems (COP) that are not easy to solve due to their size and combinatorial nature. The satisfiability problem (SAT) is one of the popular optimization problems. In this problem, the solver determines the satisfiability of a formula by searching for the Boolean variable assignment, which makes such evaluation to be true [14].

The SAT problem can be used to simulate challenges in many areas, whereas many real-world problems can be easily represented in the propositional logic such as graph coloring [11], checking [39], and task planning [27]. The SAT problem is recognized as NP-complete in almost all its variations [14].

An NP-complete variation of the SAT problem is when we want to count the satisfied clauses. This type of SAT problem is named MAX-SAT (maximum satisfiability). The objective of the MAX-SAT problem is to specify what the maximum number of clauses that are satisfied of a propositional formula is. This propositional

***Corresponding author: Iyad Abu Doush,** Computer Science Department, Yarmouk University, Irbid, Jordan; and Department of Computer Science and Information Systems, American University of Kuwait, Salmiya, Kuwait,
e-mail: iyad.doush@yu.edu.jo
Amal Lutfi Quran: Computer Science Department, Yarmouk University, Irbid, Jordan
Mohammed Azmi Al-Betar: Department of Information Technology, Al-Huson University College, Al-Balqa Applied University, P.O. Box 50, Al-Huson, Irbid, Jordan
Mohammed A. Awadallah: Department of Computer Science, Al-Aqsa University, P. O. Box 4051, Gaza, Palestine

formula is written in conjunctive normal form (CNF), where each literal is either negated or not, and the variable can be repeated in the logic clauses as shown in eq. (1):

$$F = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_3 \vee x_4 \vee x_5) \wedge \ldots (x_4 \vee \neg x_6 \vee \neg x_9) \quad (1)$$

Furthermore, MAX-SAT problem can be considered as a generalization of the SAT problem. Solving the MAX-SAT can lead to solve the SAT problem easily, but not vice versa. Thus, solving the MAX-SAT problem is harder than solving the SAT problem. Additionally, even in the case of having two literals for each clause, the MAX-SAT is considered NP-hard. On the other hand, the SAT problem can be solved in polynomial time if we have two literals per clause [44].

MAX-SAT is broadly used as modeling framework to solve various combinatorial optimization problems. There are many applications expressed as MAX-SAT such as scheduling [56], routing [56], model checking [10] of finite state systems, AI planning [46], electronic markets [47], and design debugging [49].

The MAX k-SAT problem is a variant of MAX-SAT that relates to the clause instances with at most k literals [28]. The MAX k-SAT problem was demonstrated as NP-complete for any $k \geq 3$ [57]. A special case of MAX k-SAT is MAX 3-SAT where each Boolean expression is presented in CNF, and each clause contains only three variables. Other variants of the MAX-SAT problem are partial MAX-SAT [43] and weighted MAX-SAT [13].

In order to solve the MAX k-SAT problem, we have two options: (i) the exact algorithms and (ii) metaheuristic methods. The exact algorithms are used to find the exact solution and to verify the satisfiability of the SAT problem or its unsatisfiability, but they typically have an exponential complexity [43]. The exact algorithms are based on the Davis–Putnam–Loveland algorithm (DPLL) [15], such as the branch and bound algorithm based on DPLL [37]. While the metaheuristics methods can find an optimal solution faster, they do not guarantee to find the exact solution of the problem. They are fundamentally based on local search and evolutionary algorithms. The metaheuristic methods for MAX k-SAT problem include stochastic local search (SLS) methods [25, 40, 41, 48], evolutionary algorithms (EA) [22, 34, 38], and hybrid methods of EA and SLS or exact methods [16, 30, 31, 35].

Harmony search (HS) algorithm, a recent evolutionary algorithm, has attracted the attention of several researchers since its appearance in 2001. The HS had been successfully applied to a variety of problems that have covered many areas as follows: mechanical component design [26], timetabling [3, 7], nurse rostering [9], office space allocation [8], structural engineering problems [36], multi-buyer multi-vendor supply chain problem [50], control systems [26], optimization benchmarks [1, 24], flow shop scheduling [54], water distribution network [19], power systems [26], information technology [26], industry [26], medical images [6], construction design [26], soil stability analysis [12], Internet routing [17], and robotics [51]. Furthermore, the structure population of HS is studied in Refs. [5] and [4].

Previous research on HS applies the algorithm to solve discrete or continuous optimization problems, and only a few research study the application of the algorithm on binary problems such as MAX k-SAT. Geem [18] introduced the first application of HS using binary code to solve water pump switching problems. Other researchers tackled other binary problems: ecologic optimization problem [20], one-dimensional binary knapsack problems [23], and 0–1 knapsack problem [58]. The research on binary-coded HS algorithms still needs more investigation, and the performance of such algorithm still needs to be improved [55].

In this paper, we investigate the effectiveness of introducing two novel hybrid variations of binary HS algorithm for MAX 3-SAT problem. The first uses flip heuristic, and the second uses Tabu search joined with flip heuristic. Also, we introduce the use of adaptive objective function based on the stepwise adaptation of weights (SAW) to evaluate the solutions resulting from the proposed hybrid variants of HS algorithm. The performance of the proposed algorithms is evaluated on solving the MAX 3-SAT problem. A comparison with nine state-of-the-art techniques shows the effectiveness of the proposed algorithms.

The organization of this paper is as follows: The background of MAX 3-SAT problem and binary HS algorithm is overviewed in Section 2. The adaptation of binary HS algorithm and its hybrid versions for the MAX 3-SAT problem are proposed in Section 3. Experiments and comparative results are presented in Section 4. Finally, the conclusion and future research directions are provided in Section 5.

2 Background

In this section, a detailed background about the MAX 3-SAT problem is discussed, and then, the procedural steps of the binary HS are illustrated.

2.1 MAX *k*-SAT Problem

CNF representation is used by the SAT solvers. Using CNF, a conjunction of propositional clauses is inserted in which a disjunction of literals form each clause.

Here, n is used to denote the number of Boolean variables, while m is used to represent the number of clauses in the propositional formula F that expressed in CNF, if F has n Boolean variables $x_1, x_2, \ldots, x_n$, and m clauses $C_1, C_2, \ldots, C_m$; therefore, the MAX k-SAT problem can be formulated as follows [52]:

- Each assignment of the Boolean variables is viewed as a binary vector V, where $V = (v_1, v_2, \ldots, \ldots, v_n) \in \{0, 1\}^n$.
- Each clause of length k is a disjunction of k literals, a clause $C_i = (x_1 \vee x_2 \vee \ldots \vee x_k)$.
- Each literal L_i is either a variable x_i or its complement $\neg x_i$.
- Each variable or its negation can be shown more than one time in the logical expression.
- The assignment is complete if all the variables are assigned, and it is called partial otherwise.
- A solution of F is an assignment satisfying all clauses of F.

For some k, the k-SAT problem searches about a complete variable assignment that makes a CNF formula $F = C_1 \wedge C_2 \wedge \ldots \wedge C_m$ evaluate to true. If such assignment exists for F, then F is a satisfiable formula, else F is said to be unsatisfiable. The MAX k-SAT problem is an important version of the SAT problem and is considered as a combinatorial optimization problem. It seeks to find the best assignment $V \in \{0, 1\}^n$ in which we maximize the number of satisfied clauses in the Boolean formula.

The MAX k-SAT is defined by determining the set of all potential solutions ($\{0, 1\}^n$) and a function $SC \rightarrow N$ named score of the assignment, which equals the number of true clauses. The objective is to find the best binary vector that maximizes the number of satisfied clauses in the Boolean formula.

For this problem, there are 2^n possible satisfying assignments. The MAX k-SAT problem is proven to be NP-complete for any $k > 2$. In this work, we concentrate on solving the MAX 3-SAT problem, which is a combinatorial optimization problem and considered a special case of the weighted MAX 3-SAT, where each clause weight equals one. In MAX 3-SAT, each clause of F contains only three literals as shown in eq. (1).

2.2 The Binary Harmony Search Algorithm

A close relationship can be found between optimization and music. Jazz musicians when they are creating their music, they either play notes (pitches) randomly, play notes based on their experiences, or modify the pitch in order to find a perfect harmony. Therefore, in order to find an optimal solution, the variables in the HS algorithm are assigned with values, which are either selected randomly or chosen from good values previously memorized.

There are similarities between the musicians' behavior when they are composing their music and the optimization process; this analogy between improvisation and optimization is summarized below:

- Each decision variable represents a musician.
- A musical note (or a pitch) represents the value of each variable.
- A solution vector at certain iteration represents the musical harmony at a certain time.
- The audience's aesthetics corresponds to the objective function.
- As musical harmony is improved practice after practice, the solution vector is improved iteration by iteration.

The literature has many research work to solve problems using binary HS algorithm. Kong et al. [29] proposed a new binary HS (NBHS) algorithm for solving multidimensional knapsack problems. The classical HS algorithm is modified in the NBHS algorithm as follows: (i) the value of the decision variables in the solution includes the probability distribution of 0 and 1 rather than the exact value; (ii) the mean harmony concept is used in the improvisation process rather than the original concept of the memory consideration. In the evaluation process, their method obtained satisfactory results for multidimensional knapsack problems with large dimension sizes.

Afkhami et al. [2] introduced a binary HS algorithm for solving a maximum clique problem (MCP). The solution represented as a series of ones and zeros. The pitch adjustment operator of the classical HS algorithm is adjusted to be flipping the decision variables from 0 to 1 or from 1 to 0.

Nasrollahi et al. [45] presented a binary HS algorithm for highway rehabilitation decision making problems. The value of the decision variable in the solution is one if the road segment must be reconstructed and zero if not. Their method is tested using real-world dataset sampled from Iran with good results.

Wang et al. [53] introduced a binary HS algorithm for optimization benchmark functions. The solution is represented as a series of ones and zeros. The pitch adjustment operator is modified to choose the value from its structural neighborhood rather an adjacent value in HS memory. The performance of their method is better than the performance of the classical HS.

As musicians obtain a random pitch from the instrument range, random values in random selection are taken from the variable possible range of values. This is similar when a musician plays any favored pitch from his memory. In memory consideration, the values are picked from the vectors of harmony memory. When a pitch is taken from memory, the pitch can be further adjusted by a musician to the neighboring pitches to get a better harmony. In pitch adjustment, the value is updated with a predefined probability. This value can change the value in memory with adjacent values according to a defined probability. The overall HS algorithm is shown as a flowchart in Figure 1. HS has five main steps that are described below:

2.2.1 Initialize the Problem and the Algorithm Parameters

Generally, the optimization problem is formulated as:

$$\text{Minimize}\{f(x)\,|\,x_j \in X_j = 1,\ 2,\ \ldots,\ N\} \tag{2}$$

where $f(x)$ represents an objective function, and x is the set of the decision variables x_j; X_j represents the set of potential range of values for each decision variable, and N is the number of decision variables. That is $X_j \in [LB_j, UB_j]$, where LB_j and UB_j are both the lower and upper boundaries for each decision variable,

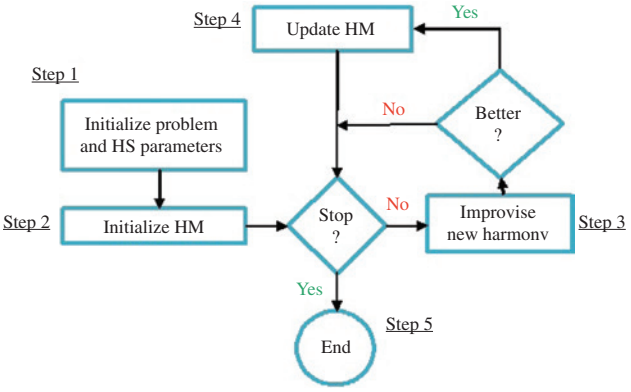

Figure 1: The Flowchart of HS Algorithm.

respectively. In the binary problem the lower bound is 0, and the upper bound is 1, and the variable x_j value can be either 0 or 1.

The HS algorithm parameters are also initialized in this step:

- The number of improvisations (NI): NI is a termination condition of the optimization process that corresponds to the number of iterations.
- Harmony memory size (HMS): HMS indicates how many solutions are stored in the harmony memory.
- Harmony memory consideration rate (HMCR): the rate HMCR $\in [0,1]$ is used to decide whether the value of a decision variable of the new harmony is picked from the harmony memory (HM).
- Pitch adjustment rate (PAR): the probability PAR $\in [0,1]$ decides whether the decision variable picked from the harmony memory will be adjusted to the adjacent value by a certain amount. In the case of the binary coded HS, the solution can have only two values 0 or 1. This operator is used to flip the current value of the solution.

Moreover, the HMCR and PAR are the two parameters that control the three operators of the HS algorithm: (1) memory consideration controlled by HMCR, (2) random consideration controlled by 1 – HMCR, and (3) pitch adjustment rate that is controlled by PAR.

2.2.2 Initialize the Harmony Memory (HM)

The HM is a matrix of size $N \times$ HMS filled by sets of solution vectors determined by HMS. These vectors are randomly generated as follows:

In binary HS $x_j^i \in \{0, 1\}$, where $i \in \{1, 2, \ldots, \text{HMS}\}$ and $j \in \{1, 2, \ldots, N\}$.

Those generated solutions are stored in HM in ascending order (or descending order) according to their objective function values as follows:

$$\text{HM} = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_N^1 \\ x_1^2 & x_2^2 & \cdots & x_N^2 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{\text{HMS}} & x_2^{\text{HMS}} & \cdots & x_N^{\text{HMS}} \end{bmatrix}. \tag{3}$$

2.2.3 Improvise a New Harmony Memory

In this step, a new harmony vector $\mathbf{x}' = (x_1', x_2', \ldots, x_N')$ is created according to three operators:

Memory consideration: the value of the first decision variable x_1' for the new vector is randomly selected from any of the values in the specified HM range $\{x_1^1, x_1^2, \ldots, x_1^{\text{HMS}}\}$. The values of the other decision variables $(x_2', x_3', \ldots, x_N')$ are selected sequentially in the same way with probability (w.p.) HMCR, where HMCR $\in (0, 1)$, is the rate of selecting one value from the possible values stored in the HM.

Random consideration: Decision variables that are not selected with values based on memory consideration are randomly selected according to their possible value range based on random consideration with a probability (1 – HMCR) as follows:

$$x_j' \leftarrow \begin{cases} \{x_j^1, x_j^2, \ldots, x_j^{\text{HMS}}\} & \text{w. p. HMCR,} \\ \mathbf{X}_j & \text{w. p. } (1-\text{HMCR}). \end{cases} \tag{4}$$

For example, if HMCR is 0.85, this indicates that the HS algorithm will choose the decision variable value from stored values in the HM *w.p.* 85% or from the whole possible range *w.p.* (100–85)%. For the binary HS, the random consideration means generating $x_j \in \{0, 1\}$ randomly with the rate (1 – HMCR).

Pitch adjustment: Every variable x'_j of a new harmony vector obtained by the memory consideration is checked to determine whether it should be modified with the probability of PAR, where PAR ∈ (0, 1), which is the rate of pitch adjustment as bellow:

$$\text{pitch adjustment for } x'_j? \leftarrow \begin{cases} x''_j & \text{w. p. PAR,} \\ x'_j & \text{w. p. } (1-\text{PAR}). \end{cases} \tag{5}$$

The (1 – PAR) value refers to the rate of doing nothing. If the pitch adjustment decision for x'_j is yes, the value of x'_j is replaced to its adjacent value as follows:

$$x''_j? \leftarrow \begin{cases} 1 & \text{if } x'_j = 0, \\ 0 & \text{if } x'_j = 1. \end{cases} \tag{6}$$

2.2.4 Update the Harmony Memory (HM)

In this step, the objective function $f(x')$ value is calculated for the new harmony vector $\mathbf{x}'$. In case the vector of the new harmony is better than the worst harmony $\mathbf{x}^{\text{worst}}$ in the HM (i.e. $\mathbf{x}^{\text{worst}} = \mathbf{x}^1$ if HM is sorted in descending order), the new harmony is kept in the HM, and the worst harmony vector is deleted from the HM.

2.2.5 Check the Stopping Criterion

The computation is terminated in case the maximum number of improvisations (i.e. the stopping criterion) is satisfied. This is specified by the NI parameter or else, Steps 3 and 4 of HS algorithm are repeated, and finally, the best solution to the problem will be the best harmony memory vector. The HS procedure can be presented as in Algorithm 1.

Algorithm 1: Binary HS Algorithm Pseudo-Code.

```
Set HS parameters: HMCR, PAR, NI, HMS.
x_i^j = Random(0, 1), ∀i = 1, 2, ..., N and ∀j = 1, 2, ..., HMS {generate HM solutions}
Calculate(f(x^j)), ∀j =(1, 2, ..., HMS)
Sort(HM)
itr = 0
while (itr ≤ NI) do
  x' = φ
  for i = 1, ..., N do
    if (U(0, 1) ≤ HMCR) then
      x'_i ∈ {x_i^1, x_i^2, ..., x_i^HMS} {memory consideration}
      if (U(0, 1) ≤ PAR) then
        x'_i = Flip(x'_i) {pitch adjustment}
      end if
    else
      x'_i ∈ {0, 1} {random consideration}
    end if
  end for
  if (f(x') < f(x^worst)) then
    Include x' to the HM
    Exclude x^worst from HM
  end if
  itr = itr + 1
end while
```

3 The Methodology

In this section, the proposed approaches are presented for MAX 3-SAT problem in three consecutive versions as follows: the first proposed algorithm called HS algorithm for SATisfiability problem (HSASAT) where it applies the pure HS for solving MAX 3-SAT. The second proposed algorithm refers to as weighted HS algorithm with flip heuristic for SATisfiability problem (WHSFLIP), which is an enhanced version of HSASAT. It maintains a local search procedure that is based on the Flip heuristic used in flipGA [38]. The last proposed algorithm is an enhancement to the WHSFLIP to develop a new algorithm named weighted HS with Tabu search for a SATisfiability problem (WHSTS), which is based on a local search algorithm that mixes the Flip heuristic with Tabu search principles.

In WHSFLIP and WHSTS, a new objective function is utilized based on the clause weights as will be discussed in the following subsection.

3.1 Objective Function

The objective function (or evaluation function) is utilized to decide whether the solution for 3-SAT problem has a good quality or not. There are several proposed fitness functions [22] used by the evolutionary algorithms for the MAX 3-SAT problem. Most of the previous meta-heuristic methods use the standard objective function (i.e. the number of satisfied clauses) as shown in eq. (7), which maximizes the number of true clauses:

$$F(x) = \sum_{i=1}^{m} C_i(x) \tag{7}$$

in which $C_i(x)$ represents the truth value of the i-th clause. Thus, we use a dynamic objective function that is based on the SAW mechanism [21]. This adaptive objective function is given as shown in eq. (8).

$$F(\boldsymbol{x}) = \sum_{i=1}^{m} W_i \times C_i(\boldsymbol{x}) \tag{8}$$

A weight W_i is added to each clause C_i. The weights are used to specify which clauses are difficult to satisfy in the current iteration. The weights are initialized to one at the beginning as we give same difficulty rate for all the clauses. The algorithm automatically modifies the weights. After some iterations (say 250), we adjust the weights according to eq. (9). This change will make the unsatisfied clauses have more weights, and thus, the search focuses more on these clauses (i.e. guided search).

$$W_{i+1} = W_i + 1 - C_i(\boldsymbol{x}^*) \tag{9}$$

3.2 The HS Representation of MAX-SAT Solutions

We map the candidate solutions for MAX 3-SAT into harmony representation then we can apply HS to solve this problem. There are different possible representations of the MAX-SAT search space such as binary representation, floating point representation, the clausal representation, and the path representation [22]. In our proposed algorithms, we used the binary representation because many state-of-the-art 3-SAT evolutionary algorithm solvers use this representation [22]. Using this representation, a candidate solution is represented using a binary vector of length n. Note that n represents the number of Boolean variables in the CNF formula (see Figure 2).

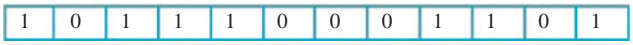

Figure 2: Binary Representation of Harmony.

Algorithm 1 illustrates the overall flow of the HSASAT approach.

3.3 Local Search used in HS Algorithm

Local search techniques can be used to improve the performance of HS. We have developed a new version of HS for MAX-SAT called weighted HS with flip heuristic satisfiability problems (WHSFLIP), where the PAR operator is substituted with flip heuristic used in flipGA [38] as a local search (see Algorithm 2). Such modification in the algorithm can accelerate the convergence, and thus, the algorithm produces better outcomes.

Algorithm 2: Local Search Procedure based on Flip Heuristic Pseudo-Code.

Input: Boolean assignment, formula ϕ in CNF, maxflip {is the maximum number of flips allowed}.
enhancement = 1
numFlip = 0 {The current number of flips applied}
while {enhancement > 0 numFlip < maxflip} **do**
  enhancement = 0
  for $i = 1, \ldots, nVar$ **do**
    {nVar is the number of variables to be resolved}
    flip var_i {flip the ith variable}
    numFlip ++
    calculate the flip gain
    if gain ≥ 0 **then**
      the flip is confirmed
      enhancement = enhancement + gain
    else
      the flip is rolled back var_i
    end if
  end for
end while
Output: Boolean assignment

Flip heuristic means the Boolean formula variables are flipped. We agree on the flip if and only if the gain ≥ 0 is satisfied, where the gain is

$$\text{Gain} = \text{number of satisfied clauses after flip} - \text{number of satisfied clauses before flip}$$

We repeat the process until no enhancement in the number of satisfied clauses is found. Therefore, a new harmony or a potential solution is produced through random selection, harmony memory consideration, or flip heuristic (see Algorithm 2).

It is apparently noted that the problem encountered using this heuristic is the large flipped variables to get the best solution. Thus, in order to reduce this overhead, we developed another version of HS for MAX 3-SAT named weighted HS with Tabu search for SATisfiability problem (WHSTS). Tabu search is inserted to the Flip heuristic to minimize the number of flips applied. The variable is added to the Tabu list if it does not improve the objective function (see Algorithm 3).

Algorithm 3: Local Search Procedure based on Tabu Search and Flip Heuristic Pseudo-Code.

```
Input: Boolean assignment, formula φ in CNF, maxflip. {maxflip is the maximum number of flips allowed}
enhancement = 1
numFlip = 0 {The current number of flips applied}
While enhancement > 0 and numFlip < maxflip do
  enhancement = 0
  for i = 1, …, nVar do
    {nVar is the number of variables to be resolved}
    if var_i ∈ TabuList then
      flip var_i {flip the ith variable}
      numFlip++
      calculate the flip gain
    end if
    if gain ≥ 0 then
      the flip is confirmed
      enhancement = enhancement + gain
    else
      TabuList.push(var_i)
      the flip is rolled back var_i
    end if
    if TabuList.size() == TabuListMax then
      TabuList.pop()
    end if
  end for
end while
Output: Boolean assignment
```

The Tabu variables are stored in an ordered FIFO list. Then, during the iterations, these variables are not allowed to be flipped again. The length of the Tabu list is set experimentally according to the number of variables in the CNF formula. The optimal length of the Tabu list is set based on the experiments of Mazure according to eq. (10) [42]:

$$\text{Optimal length of tabu list} = 0.01875n + 2.8125 \tag{10}$$

The overall flow of WHSTS is illustrated in Algorithm 3).

4 Experiments and Results

The three versions of the proposed HS-based algorithms for MAX 3-SAT are experimentally evaluated and compared with some evolutionary algorithms along with some state-of-the-art MAX 3-SAT solvers. The experiments were executed on an Intel core i7 processor running with 8 GB of RAM, where the proposed algorithms were coded using MathWorks MATLAB R2015a (The MathWork, Natick, MA, USA) under Microsoft Windows 7 (Microsoft, Redmond, WA, USA).

4.1 Datasets used for Evaluation

In order to estimate the experimental performance of the proposed algorithms, several test sets are carried out using different AIM benchmark instances taken from Ref. [42]. The AIM benchmark instances were generated by a particular random 3-SAT instance generator that produces yes-instances and no-instances separately for wide ranges.[1] We used 10 yes-instances from the AIM family benchmarks that have exactly one

1 http://www.cs.ubc.ca/hoos/SATLIB/benchm.html.

Table 1: The Characteristics of the Dataset used in the Evaluation.

Dataset name	Number of clauses	Number of variables
aim – 50 – 1_6 – *yes*1 – 4	80	50
aim – 50 – 2_0 – *yes*1 – 1	100	50
aim – 50 – 3_4 – *yes*1 – 1	170	50
aim – 50 – 6_0 – *yes*1 – 1	300	50
aim – 100 – 1_6 – *yes*1 – 1	160	100
aim – 100 – 2_0 – *yes*1 – 1	200	100
aim – 100 – 3_4 – *yes*1 – 1	340	100
aim – 100 – 6_0 – *yes*1 – 1	600	100
aim – 200 – 2_0 – *yes*1 – 1	400	200
aim – 200 – 6_0 – *yes*1 – 1	1200	200

satisfying assignment to assess the capabilities of our proposed approaches in finding the exact solutions (see Table 1).

4.2 Experimental Results

The proposed algorithms are tested on different AIM yes-instance benchmarks. The obtained results are compared with the algorithms presented in Refs. [32, 33]. We run the HSASAT, WHSFLIP, and WHSTS programs using the following parameters' setting: population size (HMS) = 20, total number of generations = 1000, Maxflip in flip heuristic = 30,000, HMCR = 0.97, PAR = 0.3, and FW = 0.01. Each instance is tested 30 times independently. These parameter settings are chosen based on several preliminary experimental testing.

The proposed algorithms compared with the state-of-the-art evolutionary algorithms presented in Ref. [33] are as follows:

- PSO-LS: a particle swarm optimization (PSO) that uses SAW objective function and uses a standard PSO-flight operation based on sigmoid transformation.
- PSOSAT: PSO algorithm with the standard objective function (i.e. number of true clauses).
- WPSOSAT: weighted particle swarm optimization for satisfiability problems, which is an evolutionary algorithm based on a hybrid PSO algorithm and flip heuristic using the SAW objective function.

The comparison results among those evolutionary algorithms along with HSASAT, WHSFLIP, and WHSTS are summarized in Table 2. Considering the results in the table, we can note that WHSFLIP, WHSTS, and WPSOSAT succeeded in finding the exact solutions for the 10 benchmark instances, while HSASAT,

Table 2: Comparative Results 1.

Tests	HSASAT	WHSFLIP	WHSTS	PSO-LS	PSOSAT	WPSOSAT
aim – 50 – 1_6 – *yes*1 – 4	79	**80**	**80**	79	79	**80**
aim – 50 – 2_0 – *yes*1 – 1	**100**	**100**	**100**	98.25	**100**	**100**
aim – 50 – 3_4 – *yes*1 – 1	**170**	**170**	**170**	165	**170**	**170**
aim – 50 – 6_0 – *yes*1 – 1	**300**	**300**	**300**	288.25	**300**	**300**
aim – 100 – 1_6 – *yes*1 – 1	159	**160**	**160**	154.25	159	**160**
aim – 100 – 2_0 – *yes*1 – 1	199	**200**	**200**	191	199	**200**
aim – 100 – 3_4 – *yes*1 – 1	**340**	**340**	**340**	319.5	**340**	**340**
aim – 100 – 6_0 – *yes*1 – 1	**600**	**600**	**600**	557	**600**	**600**
aim – 200 – 2_0 – *yes*1 – 1	399	**400**	**400**	373.25	399	**400**
aim – 200 – 6_0 – *yes*1 – 1	**1200**	**1200**	**1200**	1095.25	**1200**	**1200**

Bold values represent best solution.

PSOSAT failed in benchmarks *aim* – 50 – 1_6 – *yes*1 – 4, *aim* – 100 – 1_6 – *yes*1 – 1, *aim* – 100 – 2_0 – *yes*1 – 1, *aim* – 200 – 2_0 – *yes*1 – 1. In contrast, PSO-LS failed in all the benchmark instances.

Furthermore, the proposed algorithms are compared with evolutionary algorithms tested in Ref. [32], which are:

- ClonTS: a clonal selection algorithm based on flip heuristic and the standard objective function (i.e. number of true clauses).
- WClonTS: weighted clonal selection with Tabu for satisfiability problems. This is based on the SAW objective function.
- Clonsat: an evolutionary algorithm based on a hybrid clonal selection algorithm and Walksat procedure.

Comparison of those evolutionary algorithms with the proposed HSASAT, WHSFLIP, and WHSTS is shown in Table 3. The results show that WHSFLIP, WHSTS, and WClonTS succeeded in finding the exact solutions for 10 benchmark instances, while HSASAT, ClonTS, and Clonsat failed in benchmarks *aim* – 50 – 1_6 – *yes*1 – 4, *aim* – 100 – 1_6 – *yes*1 – 1, *aim* – 100 – 2_0 – *yes*1 – 1, and *aim* – 200 – 2_0 – *yes*1 – 1.

In addition, we have made a comparison with three state-of-the-art stochastic local search algorithms: Novelty+, Walksat, and IROTS that are presented in Refs. [32, 33]. These algorithms are compared with the proposed algorithms shown in Table 4

Comparison of those evolutionary algorithms beside HSASAT, WHSFLIP, and WHSTS is shown in Table 3. The evaluation results prove that WHSFLIP, WHSTS, and WClonTS succeeded in finding the exact solutions for the 10 benchmark instances, while HSASAT, ClonTS, and Clonsat failed in the benchmarks *aim* – 50 – 1_6 – *yes*1 – 4, *aim* – 100 – 1_6 – *yes*1 – 1, *aim* – 100 – 2_0 – *yes*1 – 1, and *aim* – 200 – 2_0 – *yes*1 – 1.

Table 3: Comparative Results 2.

Tests	HSASAT	WHSFLIP	WHSTS	ClonTS	WClonTS	Clonsat
aim – 50 – 1_6 – *yes*1 – 4	79	**80**	**80**	79	**80**	79.75
aim – 50 – 2_0 – *yes*1 – 1	**100**	**100**	**100**	**100**	**100**	**100**
aim – 50 – 3_4 – *yes*1 – 1	**170**	**170**	**170**	**170**	**170**	**170**
aim – 50 – 6_0 – *yes*1 – 1	**300**	**300**	**300**	**300**	**300**	**300**
aim – 100 – 1_6 – *yes*1 – 1	159	**160**	**160**	159	**160**	159
aim – 100 – 2_0 – *yes*1 – 1	199	**200**	**200**	199	**200**	199
aim – 100 – 3_4 – *yes*1 – 1	**340**	**340**	**340**	**340**	**340**	**340**
aim – 100 – 6_0 – *yes*1 – 1	**600**	**600**	**600**	**600**	**600**	**600**
aim – 200 – 2_0 – *yes*1 – 1	399	**400**	**400**	399	**400**	399
aim – 200 – 6_0 – *yes*1 – 1	**1200**	**1200**	**1200**	**1200**	**1200**	**1200**

Bold values represent best solution.

Table 4: Comparative Results 3.

Tests	HSASAT	WHSFLIP	WHSTS	Novelty+	Walksat	IROTS
aim – 50 – 1_6 – *yes*1 – 4	79	**80**	**80**	79	79	79.75
aim – 50 – 2_0 – *yes*1 – 1	**100**	**100**	**100**	**100**	**100**	**100**
aim – 50 – 3_4 – *yes*1 – 1	**170**	**170**	**170**	**170**	**170**	**170**
aim – 50 – 6_0 – *yes*1 – 1	**300**	**300**	**300**	**300**	**300**	**300**
aim – 100 – 1_6 – *yes*1 – 1	159	**160**	**160**	159	159	159
aim – 100 – 2_0 – *yes*1 – 1	199	**200**	**200**	199	199	199
aim – 100 – 3_4 – *yes*1 – 1	**340**	**340**	**340**	**340**	**340**	**340**
aim – 100 – 6_0 – *yes*1 – 1	**600**	**600**	**600**	**600**	**600**	**600**
aim – 200 – 2_0 – *yes*1 – 1	399	**400**	**400**	399	399	399
aim – 200 – 6_0 – *yes*1 – 1	**1200**	**1200**	**1200**	**1200**	**1200**	**1200**

Bold values represent best solution.

4.3 Friedman Test Results

A Friedman test was conducted to test the differences in the means of the 12 algorithms. The hypotheses are:
- H_0: all the means are equal.
- H_1: Not all the means are equal.

The analysis results of the Friedman test are presented in Tables 5–7. The descriptive statistics of each algorithm is shown in Table 5. Table 6 presents the mean ranks of the 12 algorithms, and Table 7 reports the Friedman test statistics.

In order to determine whether the 12 algorithms are different significantly, we check the *Chi square* statistics, *df*, and *Asymp. Sig.* value. According to Table 7, we can see that the *P*-value is less than 0.05. So we reject

Table 5: Descriptive Statistics.

	N	Mean	Stdv.	Min	Max	Percentiles		
						25th	50th	75th
WHSFLIP	10	355	336.03075	80	1200	145	250	450
WHSTS	10	355	336.03075	80	1200	145	250	450
WPSOSAT	10	355	336.03075	80	1200	145	250	450
WClonTS	10	355	336.03075	80	1200	145	250	450
IROTS	10	354.675	336.15464	79.75	1200	144.25	249.5	449.25
Clonsat	10	354.675	336.15464	79.75	1200	144.25	249.5	449.25
HSASAT	10	354.6	336.22288	79	1200	144.25	249.5	449.25
PSOSAT	10	354.6	336.22288	79	1200	144.25	249.5	449.25
ClonTS	10	354.6	336.22288	79	1200	144.25	249.5	449.25
Novelty+	10	354.6	336.22288	79	1200	144.25	249.5	449.25
Walksat	10	354.6	336.22288	79	1200	144.25	249.5	449.25
PSO-LS	10	332.075	304.58724	79	1095.25	140.25	239.625	419.1875

Table 6: Friedman Test.

	Mean rank
WHSFLIP	8.4
WHSTS	8.4
WPSOSAT	8.4
WClonTS	8.4
IROTS	6.45
Clonsat	6.45
HSASAT	6.05
PSOSAT	6.05
ClonTS	6.05
Novelty+	6.05
Walksat	6.05
PSO-LS	1.25

Table 7: Test Statistics.

N	10
Chi square	73.028
Df	11
Asymp. Sig.	0

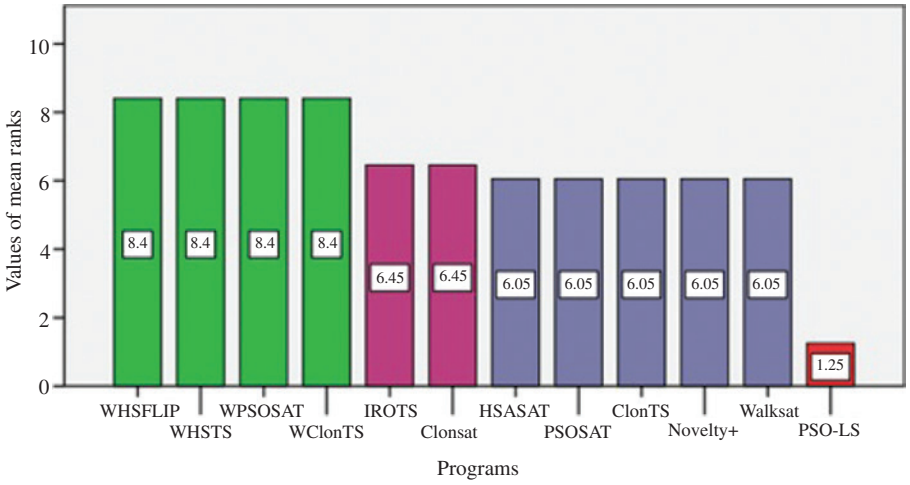

Figure 3: Friedman Test for Satisfiable Tests.

H_0 at 5% significance level and accept H_1. So there is a statistically significant difference in the means of the 12 algorithms.

By looking at Table 6 we can see that WHSFLIP, WHSTS, WPSOSAT, and WClonTS had the heights rank (i.e. 8.40), thanks to the SAW objective function and using hybrid evolutionary algorithms with local search, which improves the algorithms' performance significantly.

IROTS and Clonsat ranked second in the Friedman test and were close to the exact solutions as IROTS executed RoTS algorithm at each local search phase and Clonsat combined clonal selection algorithm and Walksat procedure.

Also, the analysis results show that HSASAT, PSOSAT, ClonTS, Novelty+, and Walksat have similar performances, but they ranked second in the Friedman test, where HSASAT and PSOSAT used, respectively, pure HS and pure PSO algorithms with the standard objective function, while ClonTS used clonal selection algorithm based on flip heuristic and the standard objective function. Novelty+ and Walksat are state-of-the-art stochastic local search algorithms.

The analysis results showed that PSO-LS is not competitive to deal with satisfied instances. It was ranked the last in the Friedman test because it used a pure flight operation based on sigmoid transformation. According to the analysis results, our proposed approaches WHSFLIP, WHSTS succeeded in finding the exact solutions of the yes-instances, while HSASAT had good results (see Figure 3).

5 Conclusion

In this paper, three new approaches are proposed to solve the MAX 3-SAT problem. The first developed approach is based on the HS algorithm with the standard MAX-SAT objective function (HSASAT). The second proposed approach is a novel HS algorithm with a local search algorithm based on flip heuristic (WHSFLIP). The third developed approach is a novel HS algorithm with Tabu search and flip heuristic (WHSTS). In order to enhance the performance of the second and third approaches, we use adaptive objective function based on the SAW mechanism.

According to the experimental results, HSASAT algorithm provides competitive results in most yes-instances. While WHSFLIP gave better solutions compared with other programs in all conducted tests, it suffers from the large number of flips used to find the best solution. On the other hand, WHSTS approach overcome that problem using Tabu search combined with flip heuristic and gave high-quality solutions over all benchmark instances. Furthermore, the framework of the proposed procedures was considered as an extended platform for evaluating the different variants of the satisfiability problems.

The proposed novel HS algorithms can be used to solve other optimization problems as they prove to provide better solutions than the original HS algorithm.

Our future work will focus on testing the proposed procedures over larger benchmarks and over unsatisfied benchmark instances to ensure the effectiveness of the proposed approaches. Also, we plan to apply the proposed novel HS algorithms on other optimization problems.

Bibliography

[1] I. Abu Doush, Harmony search with multi-parent crossover for solving ieee-cec2011 competition problems, in: *Proceedings of the 19th International Conference on Neural Information Processing – Volume Part IV, ICONIP'12*, Lecture Notes in Computer Science, vol. 7666. Springer, Berlin, Heidelberg, Doha, Qatar, pp. 108–114, 2012.

[2] S. Afkhami, O. R. Ma and A. Soleimani, A binary harmony search algorithm for solving the maximum clique problem, *Int. J. Comput. Appl.* **69** (2013), 38–43.

[3] M. A. Al-Betar and A. T. Khader, A harmony search algorithm for university course timetabling, *Ann. Oper. Res.* **194** (2012), 3–31.

[4] M. A. Al-Betar, A. T. Khader, M. A. Awadallah, M. H. Alawan and B. Zaqaibeh, Cellular harmony search for optimization problems, *J. Appl. Math.* **20** (2013), 2013.

[5] M. A. Al-Betar, M. A. Awadallah, A. T. Khader and Z. A. Abdalkareem, Island-based harmony search for optimization problems, *Exp. Syst. Appl.* **42** (2015), 2026–2035.

[6] O. M. Alia, R. Mandava, D. Ramachandram and M. E. Aziz, Harmony search-based cluster initialization for fuzzy c-means segmentation of MR images, in: *TENCON 2009–2009 IEEE Region 10 Conference, IEEE*, Singapore, pp. 1–6, 2009.

[7] K. Anwar, A. T. Khader, M. A. Al-Betar and M. A. Awadallah, Harmony search-based hyper-heuristic for examination timetabling, in: *2013 IEEE 9th International Colloquium on Signal Processing and its Applications (CSPA)*, IEEE, Kuala Lumpur, Malaysia, pp. 176–181, 2013.

[8] M. A. Awadallah, A. T. Khader, M. A. Al-Betar and P. C. Woon, Office-space-allocation problem using harmony search algorithm, in: *Neural Information Processing: 19th International Conference, ICONIP 2012, Doha, Qatar, November 12–15, 2012, Proceedings, Part II*, pp. 365–374, Springer, Berlin, Heidelberg, 2012.

[9] M. A. Awadallah, M. A. Al-Betar, A. T. Khader, A. L. Bolaji and M. Alkoffash, Hybridization of harmony search with hill climbing for highly constrained nurse rostering problem, *Neural Comput. Appl.* **28** (2017), 463–482.

[10] A. Biere, A. Cimatti, E. Clarke and Y. Zhu, Symbolic model checking without BDDs, in: *Tools and Algorithms for the Construction and Analysis of Systems: 5th International Conference, TACAS'99 Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS'99 Amsterdam, The Netherlands, March 22–28, 1999 Proceedings*, pp. 193–207, Springer, Berlin, Heidelberg, 1999.

[11] N. Bouhmala and O.-C. Granmo, Solving graph coloring problems using learning automata, in: *Evolutionary Computation in Combinatorial Optimization: Proceedings of the 8th European Conference, EvoCOP 2008*, Naples, Italy, March 26–28, 2008, pp. 277–288, Springer, Berlin, Heidelberg, 2008.

[12] Y. Cheng, L. Li, T. Lansivaara, S. Chi and Y. Sun, An improved harmony search minimization algorithm using different slip surface generation methods for slope stability analysis, *Eng. Optim.* **40** (2008), 95–115.

[13] A. Choi, T. Standley and A. Darwiche, Approximating weighted max-SAT problems by compensating for relaxations, in: *Principles and Practice of Constraint Programming – CP 2009: 15th International Conference*, *CP 2009 Lisbon, Portugal, September 20–24, 2009 Proceedings*, pp. 211–225, Springer, Berlin, Heidelberg, 2009.

[14] S. A. Cook, The complexity of theorem-proving procedures, in: *Proceedings of the Third Annual ACM Symposium on Theory of Computing, STOC '71*, pp. 151–158, ACM, New York, NY, USA, 1971.

[15] M. Davis, G. Logemann and D. Loveland, A machine program for theorem-proving, *Commun. ACM* **5** (1962), 394–397.

[16] S. Fernandes and H. R. Lourenço, Hybrids combining local search heuristics with exact algorithms, in: *V Congreso Espanol sobre Metaheursticas, Algoritmos Evolutivos y Bioinspirados*, pp. 269–274, 2007.

[17] R. Forsati, A. Haghighat and M. Mahdavi, Harmony search based algorithms for bandwidth-delay-constrained least-cost multicast routing, *Comput. Commun.* **31** (2008), 2505–2519.

[18] Z. W. Geem, Harmony search in water pump switching problem, in: *Advances in Natural Computation: First International Conference, ICNC 2005, Changsha, China, August 27–29, 2005, Proceedings, Part III*, pp. 751–760, Springer, Berlin, Heidelberg, 2005.

[19] Z. W. Geem, Particle-swarm harmony search for water network design, *Eng. Optim.* **41** (2009), 297–311.

[20] Z. W. Geem and J. C. Williams, Harmony search and ecological optimization, *IJEST* **1** (2007), 150–154.

[21] J. Gottlieb and N. Voss, Adaptive fitness functions for the satisfiability problem, in: *Parallel Problem Solving From Nature PPSN VI: 6th International Conference Paris, France, September 18–20, 2000 Proceedings*, pp. 621–630, Springer, Berlin, Heidelberg, 2000.

[22] J. Gottlieb, E. Marchiori and C. Rossi, Evolutionary algorithms for the satisfiability problem, *Evol. Comput.* **10** (2002), 35–50.
[23] J. Greblicki and J. Kotowski, Analysis of the properties of the harmony search algorithm carried out on the one dimensional binary knapsack problem, in: *Computer Aided Systems Theory – EUROCAST 2009: 12th International Conference, Las Palmas de Gran Canaria, Spain, February 15–20, 2009*, Revised Selected Papers, pp. 697–704, Springer, Berlin, Heidelberg, 2009.
[24] B. H. F. Hasan, I. A. Doush, E. A. Maghayreh, F. Alkhateeb and M. Hamdan, Hybridizing harmony search algorithm with different mutation operators for continuous problems, *Appl. Math. Comput.* **232** (2014), 1166–1182.
[25] H. H. Hoos and T. Stützle, Local search algorithms for SAT: an empirical evaluation, *J. Autom. Reason.* **24** (2000), 421–481.
[26] G. Ingram and T. Zhang, Overview of applications and developments in the harmony search algorithm, in: *Music-Inspired Harmony Search Algorithm: Theory and Applications*, pp. 15–37, Springer, Berlin, Heidelberg, 2009.
[27] H. Kautz and B. Selman, Pushing the envelope: planning, propositional logic, and stochastic search, in: *Proceedings of the National Conference on Artificial Intelligence*, Portland, Oregon, pp. 1194–1201, 1996.
[28] H. Kautz and B. Selman, The state of sat, *Discrete Appl. Math.* **155** (2007), 1514–1524.
[29] X. Kong, L. Gao, H. Ouyang and S. Li, Solving large-scale multidimensional knapsack problems with a new binary harmony search algorithm, *Comput. Oper. Res.* **63** (2015), 7–22.
[30] L. Kroc, A. Sabharwal, C. P. Gomes and B. Selman, Integrating systematic and local search paradigms: a new strategy for MaxSAT, in: *Proceeding IJCAI'09 – Proceedings of the 21st international joint conference on Artifical intelligence*, Pasadena, California, USA, pp. 544–551, 2009.
[31] F. Lardeux, F. Saubion and J.-K. Hao, Gasat: a genetic local search algorithm for the satisfiability problem, *Evol. Comput.* **14** (2006), 223–253.
[32] A. Layeb, A clonal selection algorithm based tabu search for satisfiability problems, *J. Adv. Inform. Technol.* **3** (2012), 138–146.
[33] A. Layeb, A particle swarm algorithm for solving the maximum satisfiability problem, in: *12th International Arab Conference in Information Technology*, Naif Arab University for Security Sciences, Saudia Arabia, pp. 175–184, 2011.
[34] A. Layeb and D.-E. Saidouni, A new quantum evolutionary local search algorithm for MAX 3-SAT problem, in: *Hybrid Artificial Intelligence Systems: Third International Workshop, HAIS 2008, Burgos, Spain, September 24–26, 2008, Proceedings*, pp. 172–179, Springer, Berlin, Heidelberg, 2008.
[35] A. Layeb, A. H. Deneche and S. Meshoul, A new artificial immune system for solving the maximum satisfiability problem, in: *Trends in Applied Intelligent Systems: 23rd International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2010, Cordoba, Spain, June 1–4, 2010, Proceedings, Part II*, pp. 136–142, Springer, Berlin, Heidelberg, 2010.
[36] K. S. Lee and Z. W. Geem, A new structural optimization method based on the harmony search algorithm, *Comput. Struct.* **82** (2004), 781–798.
[37] C. M. Li, F. Manyà and J. Planes, Exploiting unit propagation to compute lower bounds in branch and bound Max-SAT solvers, in: *Principles and Practice of Constraint Programming – CP 2005: 11th International Conference, CP 2005, Sitges, Spain, October 1–5, 2005. Proceedings*, Springer, Berlin, Heidelberg, 2005, pp. 403–414.
[38] E. Marchiori and C. Rossi, A flipping genetic algorithm for hard 3-SAT problems, in: *Proceedings of the Genetic and Evolutionary Computation Conference*, Morgan Kaufmann Publishers, Orlando, Florida, USA, Vol. 1, pp. 393–400, 1999.
[39] F. Marić, Formal verification of a modern sat solver by shallow embedding into Isabelle/HOL, *Theor. Comput. Sci.* **411** (2010), 4333–4356.
[40] J. P. Marques-Silva and K. A. Sakallah, Grasp: a search algorithm for propositional satisfiability, *IEEE Trans. Comput.* **48** (1999), 506–521.
[41] M. Mastrolilli and L. M. Gambardella, Maximum satisfiability: how good are tabu search and plateau moves in the worst-case? *Eur. J. Oper. Res.* **166** (2005), 63–76.
[42] B. Mazure, L. Sas and É. Grégoire, Tabu search for sat, in: *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Innovative Applications of Artificial Intelligence Conference,* AAAI 97, IAAI 97, Providence, Rhode Island, pp. 281–285, 1997.
[43] M. E. B. Menaï and M. Batouche, A backbone-based co-evolutionary heuristic for partial MAX-SAT, in: *Artificial Evolution: 7th International Conference, Evolution Artificielle*, EA 2005, Lille, France, October 26–28, 2005, Revised Selected Papers, pp. 155–166, Springer, Berlin, Heidelberg, 2006.
[44] R. G. Michael and S. J. David, Computers and intractability: a guide to the theory of np-completeness, WH Freeman and Co., San Francisco, 1979.
[45] A. Nasrollahi, M. Saffarzadeh, A. Isfahanian and M. Ghayekhloo, Application of a new binary harmony search algorithm in highway rehabilitation decision-making problems: a case study in Iran, *Civil Eng. Environ. Syst.* **32** (2015), 335–350.
[46] J. Rintanen, K. Heljanko and I. Niemelä, Planning as satisfiability: parallel plans and algorithms for plan search, *Artif. Intell.* **170** (2006), 1031–1080.
[47] T. Sandholm, Algorithm for optimal winner determination in combinatorial auctions, *Artif. Intell.* **135** (2002), 1–54.
[48] B. Selman, H. J. Levesque, D. G. Mitchell, A new method for solving hard satisfiability problems, *AAAI*, San Jose, CA, **92** (1992), 440–446.
[49] A. Smith, A. Veneris, M. F. Ali and A. Viglas, Fault diagnosis and logic debugging using boolean satisfiability, *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.* **24** (2005), 1606–1621.

[50] A. A. Taleizadeh, S. T. A. Niaki and F. Barzinpour, Multiple-buyer multiple-vendor multi-product multi-constraint supply chain problem with stochastic demand and variable lead-time: a harmony search algorithm, *Appl. Math. Comput.* **217** (2011), 9234–9253.
[51] P. Tangpattanakul and P. Artrit, Minimum-time trajectory of robot manipulator using harmony search algorithm, in: *6th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2009. ECTI-CON 2009*, Vol. 1, IEEE, Bangkok, Thailand, pp. 354–357, 2009.
[52] R. Walter, C. Zengler and W. Küchlin, Applications of maxsat in automotive configuration, in: *Configuration Workshop*, Vienna, Austria, pp. 21–28, 2013.
[53] L. Wang, Y. Xu, Y. Mao and M. Fei, A discrete harmony search algorithm, in: *Life System Modeling and Intelligent Computing Conference*, Springer, Wuxi, China, pp. 37–43, 2010.
[54] L. Wang, Q.-K. Pan and M. F. Tasgetiren, A hybrid harmony search algorithm for the blocking permutation flow shop scheduling problem, *Comput. Ind. Eng.* **61** (2011), 76–83.
[55] L. Wang, R. Yang, Y. Xu, Q. Niu, P. M. Pardalos and M. Fei, An improved adaptive binary harmony search algorithm, *Inform. Sci.* **232** (2013), 58–87.
[56] H. Xu, R. A. Rutenbar and K. Sakallah, sub-SAT: a formulation for relaxed boolean satisfiability with applications in routing, *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.* **22** (2003), 814–820.
[57] H. Zhang, H. Shen and F. Manya, Exact algorithms for max-sat, *Electron. Notes Theor. Comput. Sci.* **86** (2003), 190–203.
[58] D. Zou, L. Gao, S. Li and J. Wu, Solving 0–1 knapsack problem by a novel global harmony search algorithm, *Appl. Soft Comput.* **11** (2011), 1556–1564.