A Penalty Function Approach to Max 3-SAT Problems

Christian Kofler, Peter Greistorfer, Haibo Wang, Gary Kochenberger

Working Paper 2014-04

December 11, 2014

A Penalty Function Approach to Max 3-SAT Problems

Christian Kofler[a], Peter Greistorfer[a,*], Haibo Wang[b], Gary Kochenberger[c]

Working Paper 2014-04

December 11, 2014

Abstract

We consider a penalty function approach for the solving of the Max 3-SAT problem. The algorithm introduced is a multi-start approach that makes use of elite-solution techniques derived from scatter search. More precisely, it is based on the so-called adaptive memory projection metaphor. The main focus of this paper is to demonstrate the usefulness of this projection idea in the context of the binary Max 3-SAT. We review the literature in that field, explain the metaheuristic proposed and present results on the basis of a DIMACS test set.

Keywords: Satisfiability, Max 3-SAT, adaptive memory programming, projection, metaheuristic

[a] University of Graz, Institute of Production and Operations Management, Universitaetsstrasse 15/E3, 8010 Graz, AUSTRIA

[b] Texas A&M International University, Division of International Business & Technology Studies 5201 University Boulevard, Laredo, Texas 78041, USA

[c] University of Colorado, Business School, 1475 Lawrence Street, Denver, Colorado 80217-3364, USA

[*] Corresponding author. Tel.: +43(316)380-7244, Fax: +43(316)380-9560, E-Mail peter.greistorfer@uni-graz.at.

Working Paper Series
Faculty of Social and Economic Sciences
Karl-Franzens-University Graz
ISSN 2304-7658
sowi.uni-graz.at/de/forschen/working-paper-series/
sowi-wp@uni-graz.at

A Penalty Function Approach to Max 3-SAT Problems

Christian Kofler[1], Peter Greistorfer[1], Haibo Wang[2], Gary Kochenberger[3]

[1] Karl-Franzens-Universität Graz
Universitätsstraße 15/E3, 8010 Graz, Austria
christian.kofler@edu.uni-graz.at, peter.greistorfer@uni-graz.at

[2] Texas A&M International University
5201 University Boulevard, Laredo, Texas 78041, USA
hwang@tamiu.edu

[3] University of Colorado
1475 Lawrence Street, Denver, Colorado 80217-3364, USA
gary.kochenberger@ucdenver.edu

Abstract

We consider a penalty function approach for the solving of the Max 3-SAT problem. The algorithm introduced is a multi-start approach that makes use of elite-solution techniques derived from scatter search. More precisely, it is based on the so-called adaptive memory projection metaphor. The main focus of this paper is to demonstrate the usefulness of this projection idea in the context of the binary Max 3-SAT. We review the literature in that field, explain the metaheuristic proposed and present results on the basis of a DIMACS test set.

Keywords: Satisfiability, Max 3-SAT, adaptive memory programming, projection, metaheuristic

1 Introduction

Satisfiability (SAT) problems play an important role in operations research, computer science and in the area of artificial intelligence. Additionally, they provide important insights into complexity theory. SAT problems are binary, i.e. boolean decision problems defined on a number of 0-1 literals, often denoted as x_j ($j = 1, \ldots, n$) building clauses ($i = 1, \ldots, m$) of literals using logical operators *and*, *or*, *not*. Constructing SAT solutions requires finding true or false values for those literals with the objective to determine whether an expression is true or not (propositional SAT and formulas, respectively), or to maximize the number of true clauses in Max-SAT. This category includes the variants of 2-SAT and 3-SAT (at most 2 and 3 literals per clause). Special forms are Exact-3-SAT, Not-All-Equal-3-SAT, Horn-SAT and Xor-SAT. It should be noted that SAT problems are NP-complete (Cook 1971, Levin 1973) and that they belong to the general class of *constraint satisfaction problems* (CSPs).

Max 3-SAT problems have been the object of many research efforts over the past few decades. They remain an important research area today due to their computational challenge and application importance. Applications are reported in diverse areas such as physics (ising models), telecommunications (frequency assignment), scheduling (satellite scheduling), bioinformatics (protein alignment), medical treatment (cancer therapy) and circuit design (debugging, logic synthesis). Other application area can be found in the literature as well, e.g. see Larrosa (2008).

In this research we investigate the use of a penalty function approach for solving these important problems. This approach was first proposed in general by Hammer and Rudeanu (1968) and has been mentioned more recently in Hansen and Jaumard (1990) and Boros and Hammer (2002). Little evidence appears in the literature, however, regarding the potential usefulness of this approach for actually solving Max 3-SAT problems. In Kochenberger et al. (2005), this penalty function approach was successfully employed to solve a large number of Max 2-SAT problems. The success on this class of Max SAT problems motivated our interest here in investigating the effectiveness of the penalty function approach for the more general Max 3-SAT problem.

We present a penalty function formulation that is combined with a specific metaheuristic approach, the so-called *adaptive memory projection* (AMP), for solving the nonlinear penalty function. In doing so, firstly Section 2 introduces and exemplifies the general penalty idea, followed by Section 3, in which the projection framework is outlined and adopted. This section also covers the specification of the overall Max 3-SAT multi-start metaheuristic. The computational results are presented and discussed in Section 4. Finally, the paper concludes with a summary and an outlook in Section 5.

2 Penalty Function Approach

Using penalties as an alternative to the explicit imposition of constraints is a frequently employed strategy in a variety of problem settings. This approach was successfully employed by Kochenberger et al. (2005) for the Max 2-Sat problem, where the penalty function for that class of problems took the form of an *unconstrained quadratic binary program* (UQBP, *xQx*). Other recently reported applications of this approach, also taking the form of UQBP, are given in Alidaee et al. (2005). The traditional Max 3-SAT problem concerns finding assignments for the true/false literals comprising the clauses such that the maximum number of clauses is satisfied. For an LP-formulation, in which the number of satisfied clauses is maximized, the reader may be directed to the foundations of Motwani and Raghavan (1995). The logical alternative, which we pursue here, is to minimize the number of clauses not satisfied via a penalty function approach. Unlike the Max 2-Sat case, where the penalty function approach results in an unconstrained quadratic minimization problem, the penalty approach for the Max 3-Sat case gives rise to an unconstrained minimization problem in binary variables of a cubic penalty function as outlined below.

We assume a standard Max 3-SAT representation in *conjunctive normal form* (CNF). For such problems, there are four possible types of clauses, each with a classical constraint of a “covering” form denoting whether or not the clause is satisfied. Associated with each such constraint is a cubic penalty function that takes the value *0* for feasible solutions and otherwise is equal to *1*. Combining these penalty functions into an aggregate penalty gives rise to an unconstrained cubic penalty function. Minimizing this function is equivalent to maximizing the number of clauses satisfied. An appropriate penalty function is revealed in the following development. For Max 3-SAT problems, there are four possible types of clauses corresponding to the number of negotiations within 3 variables. Together with their classical constraints and associated cubic penalties, these types are enumerated in Tab. 1.

Negations	*Classical constraint*	*Equivalent penalty function*
0	$x_i + x_j + x_k \geq 1$	$(1 - x_i x_j x_k - x_i - x_j - x_k + x_i x_j + x_i x_k + x_j x_k)$
1	$x_i + x_j + \bar{x}_k \geq 1$	$(x_k - x_i x_k - x_j x_k + x_i x_j x_k)$
2	$x_i + \bar{x}_j + \bar{x}_k \geq 1$	$(x_j x_k - x_i x_j x_k)$
3	$\bar{x}_i + \bar{x}_j + \bar{x}_k \geq 1$	$(x_i x_j x_k)$

Tab. 1: MAX 3-SAT constraints and penalties.

As mentioned, each penalty function takes on the value 0 or 1 depending whether or not the classical constraint is satisfied or not. The approach, then, is to minimize the sum of the penalty terms and in doing so, we minimize the number of clauses not satisfied. If we are able to drive the objective function to zero, we have a 3-SAT solution, otherwise we have a solution for the *Max* 3-SAT problem. This procedure is illustrated by the following *example*, an instance with *5* variables and *12* clauses:

$$
\begin{array}{cccc}
x_1 \vee x_2 \vee x_3 & x_2 \vee \bar{x}_3 \vee x_4 & x_2 \vee x_4 \vee x_5 & x_3 \vee x_4 \vee x_5 \\
x_1 \vee \bar{x}_2 \vee x_3 & \bar{x}_2 \vee x_3 \vee x_4 & \bar{x}_2 \vee x_3 \vee x_5 & x_3 \vee \bar{x}_4 \vee \bar{x}_5 \\
\bar{x}_1 \vee x_2 \vee \bar{x}_3 & \bar{x}_2 \vee \bar{x}_3 \vee \bar{x}_4 & x_2 \vee \bar{x}_3 \vee x_5 & \bar{x}_3 \vee \bar{x}_4 \vee \bar{x}_5
\end{array}
$$

Summing the penalties for these twelve clauses forms the penalty function:

$$
\begin{aligned}
p(x) = 3 - x_1 + x_2 - 2x_4 - 2x_5 + 2x_1x_3 - 4x_2x_3 + 3x_4x_5 \\
- x_1x_2x_3 + 3x_2x_3x_4 - x_2x_4x_5 - x_3x_4x_5 + 2x_2x_3x_5
\end{aligned}
$$

Minimizing $p(x)$ gives the solution $x_1 = x_2 = x_3 = 1,\ x_4 = x_5 = 0$ for which $p(x) = 0,$ meaning that all 12 clauses are satisfied. Each Max 3-SAT penalty, in general, is of the above form, consisting of an additive constant plus a number of linear, quadratic and cubic terms. Assuming *i < j < k*, ranging within *1, ..., n,* with *n* is the number of variables, and taking all sums over variable indices with non-zero coefficients, it can be given as $p(x) = c + \sum c_i x_i + \sum c_{ij} x_i x_j + \sum c_{ijk} x_i x_j x_k$. This penalty is the objective of an unconstrained optimization problem and represents the aggregated number of unsolved clauses, which has to be minimized. Several remarks are in order here:

- In general, solving the Max 3-Sat problem is equivalent to the problem minimize *p(x)* with $x = x_1, \ldots, x_n$.
- In forming *p(x)*, several terms from the penalty function, coming from the individual clauses, combine and/or cancel, yielding a “reduced” aggregate objective function.
- Note that *p(x)* and *c* are both non-negative, while the linear, quadratic and cubic coefficients are unrestricted in sign.
- The size of the penalty function to be minimized is independent of the number of clauses in the original CNF-formulation, depending only on the number of literals (variables) in the original problem. Thus, a MAX 3-Sat problem with, e.g., 500 variables and 1000 clauses and another with 500 variables but 20,000 clauses would each give rise to cubic minimization problems with 500 variables, where the penalty functions differ from one another in their coefficients.

There are several obvious ways to deal with this type of function. Prominent approaches are the solving of the cubic function with some non-linear solver or the reduction of *p(x)* to a quadratic function, in order to solve it as an unconstrained *xQx* (e.g. see the work of Boros et al. (1989, 2002), Hansen and Jaumard (1990), Crama et al. (1997) and Kochenberger et al. (2005)). However, in the case of the Max 3-SAT problem, naturally substitutions have to take place to eliminate cubic terms, i.e. variable settings $x_k := x_i x_j$ are needed. The approach of the present paper is a direct application of an approximation method to the cubic function and, as already noted, originates in the metaheuristic concept called adaptive memory programming.

3 Adaptive Memory Projection for Max 3-SAT

Since minimizing *p(x)* is in general NP-hard, heuristic methods are most likely needed to find high quality solutions in a reasonable amount of computing time. The approach we report on here is a variation of AMP, originally employed in the context of Integer Linear Programming (Glover 2005). An AMP method is designed to exploit what is called the *Projection Principle* for combinatorial optimization. This principle may be successfully applied to problem settings where a good method *M* is available to quickly solve problem *P* to (near) optimality if *P* has no more than n^* variables, when n^* is suitably chosen. One can say that an original problem *P* with *n* variables is *projected* onto a problem P^* by fixing $n - n^*$ variables. A consequence of this reduction is that for proper choices of n^* the free problem P^* can be solved more reliably and rapidly. The base idea is to summarize information from a small number of high-quality solutions, usually collected in a so-called reference set, in order to (1) successfully pass high-quality solution attributes to the projected problem P^*, and in order to (2) shrink the original problem *P*, so that its projected residual is smaller, thus making it easier to solve it efficiently. Efficiency can have several aspects, e.g. the fact that the overall solution time can be reduced or the possibility to enhance solution quality in contrast to a standard algorithmic design. All in all, AMP can be understood as problem reduction technique similar to the classical *Branch & Bound* or to more recent developments, like the *Core-concept* in Puchinger et al. (2006). Each of these techniques relies on a dedicated variable fixing philosophy, the one employed in AMP is population-based.The overall approach for the solving of Max 3-SAT is a 2-phase approach with both phases built to run a straight forward construction heuristic named *Basic One-Pass Heuristic (*BOPH). In each phase BOPH is embedded in a multi-start framework that re-runs this construction procedure. Adaptive memory information is gathered in phase-1 by maintaining a reference set, *RefSet*, of high quality solutions to keep track of persistently attractive variables. After termination of phase-1, variables are fixed, simultaneously deriving P^*, which then is input into the multi-start of phase-2.

We start with the definition of BOPH which iteratively applies a greedy setting to a specific decision variable. Influence is measured by a change in *p(x)*. BOPH uses weights α and β to evaluate the coefficients of the cubic penalty function mentioned above. These weights are input parameters to the procedure and control the search trajectory heuristically. Fig. 1 summarizes this greedy heuristic.

BOPH

1) *Evaluate* each variable x_j by means of $e_j = c_j + (\sum c_{ij})/\alpha + (\sum c_{ijk})/\beta$, where the sums are taken over all terms of the cubic penalty function containing variable x_j.
2) *Examine* the absolute evaluation values $\varepsilon_j = | e_j |$ and find the biggest value, i.e. set $p = arg\ max \{ \varepsilon_j \mid j = 1, \ldots, n \}$. If $e_p > 0$ then set $x_p = 0$, else $x_p = 1$.
3) *Reduce* the penalty function to reflect the assignments just made, i.e. eliminate and rearrange terms in the objective.
4) *Repeat* Steps 1 to 3 with the reduced problem until all variables are fixed. Output the objective function value, i.e. the number of the unsolved clauses.

Fig. 1: Basic-one-pass heuristic BOPH.

For $0 < \alpha < \beta$ in Step 1 of the above procedure, we are giving less weight to the quadratic terms and even less to the cubic ones. Since the maximum choice in the subsequent step depends on the preceding evaluation, different weights generate different solution. We make use of this ability to generate a set of diverse quality solutions by means of the multi-start frame, called M-BOPH. The latter iteratively calls BOPH as a sub-routine, using different weights in Step 1 according to a pre-specified (parameter) weighing schedule.

Next we describe the building of P^* between the two phases. This core-process of AMP, i.e. the exploitation of the aggregated *RefSet*-information, initializes the so-called referent optimization (compare Glover (2005)). Upon termination of phase-1, a number of b reference set solutions is stored in the $RefSet = \{ x(r), r = 1, ..., b \}$. These solutions are exploited in two steps, determining frequency-values and the actual reduction from P to P^*. Note that n^*, the size of the free problem P^* is an input parameter. Fig. 2 summarizes these two steps.

AMP – Base Reduction Process

1) *Frequencies*. For each variable x_j $(j = 1, ..., n)$ let $b_j(0)$ and $b_j(1)$ denote the number of solutions $x(r)$ in RefSet such that $x_j(r) = 0$ and $x_j(r) = 1$, respectively. (Hence $b_j(0) + b_j(1) = b$.) Determine $b_j = Max \{ b_j(0), b_j(1) \}$ and index the variables in ascending order of the b_j values so *that* $b_1 \leqslant b_2 \leqslant \cdots \leqslant b_n$.

2) *Reduction.* Determine problem P^* by selecting the first n^* variables to be free, and fixing the remaining variables x_j for $j = n^* + 1, ..., n$ by setting $x_j = x_{j^*}$ where $x_{j^*} = 0$ if $b_j = b_j(0)$ and $x_{j^*} = 1$ *if* $b_j = b_j(1)$.

Fig. 2: Exploiting the information stored in *RefSet*.

It should be noted that, in contrast to BOPH, highly-influencing variables are not greedily selected in AMP, but are identified in a pool-oriented manner (see Greistorfer and Voß (2005)) using a sort of frequency memory. Since the different elements for our Max 3-SAT approach now are sufficiently defined, the overall procedure can be put together in Fig.3.

M-BOPH and AMP for Max 3-SAT, a 2-phases approach

1) *Input* data, a weighing schedule for α and β, *RefSet*-size b and the dimension of the free problem, n^*.

2) *Phase-1.* While doing M-BOPH with the full instance P, i.e. with $x = x_j, .., x_n$, maintain a reference set *RefSet* of size b. Stop whenever the best solution found so far has a value $p(x_{best}) = 0$.

3) Build P^* using the **AMP – Base Reduction Process**, i.e. determine fixed variables and the reduced vector x^*.

4) *Phase-2.* Perform M-BOPH with the reduced Problem P^* (while keeping $x_{n^*+1}, ..., x_{n^*}$ in each iteration constant and fixed) to get solution values for $x^* = x_1, ..., x_{n^*}$. Again, stop if a $p(x_{best}) = 0$ is reached.

5) Output the overall result as the combined solution from phases 1 and 2, i.e. the combination of the fixed variables of Step 3 and the best solution found in Step 4.

Fig. 3: Overall Max 3-SAT approach.

For the maintenance of the *RefSet* the procedure uses a standard reference set update method with duplication checks as described in Glover (1998). Steps 1, 3 and 5 are computationally inexpensive, so the overall computing time mainly depends on the number of BOPH-calls in the M-BOPHs of Step 2 and 4. The phase-1-termination stated (and termination of the whole process) is obvious. In contrast, there are lots of termination criteria available if the zero-minimum is not reached. These include sophisticated ones, e.g. with the use of convergence measures applied to the results in the *RefSets*. For the application tested, the number of M-BOPH iterations was simply bounded with the number of the (α, β)-variations determined by the weighing schedule. This schedule, as well as the remaining parameters, will be given in the next section.

4 Computational Experience

The hardware used to test the metaheuristic introduced was a Lenovo T61p with an Intel Core2Duo T9500 at 2.6GHz and 3GB RAM with a 32-Bit Windows-Vista Ultimate OS, 2007, SP1. The code was programmed in C/C++ and compiled with MingW32 and GNU gcc using the Codeblocks 12.11 IDE. Computational tests were performed on *72* artificially generated random 3-SAT instances ("aim") from the DIMACS benchmark set for SAT problems (Asahiro et al. 1996). Their sizes *n* range from *50* to *200* variables with *m* between *80* to *1200* clauses (see Tab. 2).

(1)	(2)	(3)	(4)			
group	*n*	*#*	*m*			
aim-50	50	24	80	100	170	300
aim-100	100	24	160	200	340	600
aim-200	200	24	320	400	680	1200

Tab. 2: Structure of the test set.

In provide a means for comparisons, additionally we formulated each instance as a linear 0-1 program and found optimal solutions using CPLEX. For this purpose, the formulation of Givry et al. (2003) was used. Since our test problems are of modest size, the solver had little trouble optimally solving each instance, giving us a benchmark for comparison.

After some preliminary testing, the weighing schedule used for the M-BOPH was set at an ad-hoc basis: weights (α, β) are determined as 100 combinations of $\alpha = 1, 2, \ldots$, each of it combined with a $\beta = 2, \ldots, 10$, giving BOPH input parameters (1,2) ,(1,3), … (1,10); (2,2), (2,3), …, (2,10); … The number of these weight settings is equivalent to the number of BOPH-calls, which implies 100 plus 100 overall iterations (in phase-1 plus phase-2) before termination the whole algorithm. Again, after short experimental studies, the size of the *RefSet, b,* was set to *10*, while the free problem size turned out to be preferably a function of the problem size, i.e. $n^* = int(0.85\ n)$.

The detailed computational results for all instances are given in Tab. 3. Basically, in all tables the variable *C* ("cost") stands for the objective function value, i.e. the number auf unsatisfied clauses, represented by the value of the penalty function *p(x).*

Following the problem name, the number of clauses and the number of variables in columns (1) to (3), we give the CPLEX outcome of the linear programming formulation, i.e. the known optimal solution value in (4) and the CPU seconds in (5), both values used for evaluation purposes. Columns (6) to (12) summarize the results for the M-BOPH-AMP-approach: *C1** and *C2**, in (6) and (8), are the objective function values calculated in phases 1 and 2, while *k1** and *k2**, in (7) and (9), are their respective best iterations (of a total of *100* in each phase) and show, when the values of *C1** and *C2** were obtained. Note that there exist some not-available-entries, "-", in (8) and (9) which means that no phase-2 was active for that instance, since the M-POPH of phase-1 already obtained a proven optimal result of $C1^* = 0$. Finally, (10), for the sake of clearance repeats the best result achieved, followed by the best phase and the total CPU-time for the metaheuristic introduced in (11) and (12).

< preferably insert Tab. 3 here >

For the discussion of the results' quality we refer to the (aggregated) group-results of Tab. 4, which counts the number of optimal solutions reached and shows the improvement between the two phases. The upper part, columns (2) to (7), presents the (absolute) counts, whereas the lower one, with corresponding columns (8) to (13), gives the same information in terms of (relative) percentages.

< preferably insert Tab. 4 here >

The first observation, deductible from (2) of Tab.4, is that the number of optimal solutions (either reached in phase-1 or phase-2) is quite acceptable with 33 (of 72, i.e. 45.8%), above all, since it was not the prime intention of this study to develop a competitive state-of-the-art algorithm. However, the main research topic, the usefulness of the multi-start-framework implemented with M-BOPH and split into two phases with the AMP-reduction in between, can be clearly manifested with the entries of the next columns, (3) to (7).

Firstly, the amount of optimal solutions already reached with the first M-BOPH application is visible in (3), a fact that is underlined by the best phase-1-iterations, *k1**, of Tab. 3, which are widely spread between their theoretic boundaries *1* and *100*, with values between *10* and *91* (see column (7) of the preceding table, Tab. 3). Note that these entries of (3) in Tab. 4 are based on an ex-post observation, a comparison of *C1* with CPLEX' solution values. Naturally, during the run of the AMP-approach, after termination of phase-1, this knowledge is not accessible for the heuristic. So the problem reduction in Step 3 of the overall approach and the subsequent M-BOPH-run of phase-2 have to be performed whenever phase-1 terminates with a $p(x_{best}) > 0$. These cases are counted in column (4) of Tab. 4. The high total number of phase-2-runs (68 of 72, 94.4%) and its apparent correlation with the attractive outcome can be seen as second advantage of the approach: said outcome, shown in columns (5), the number of improvements in phase-2 upon phase-1 (including tiebreaks), and (6), the number of exclusive improvements in phase-2 upon phase-1, is considerably high. The objective function value of 57 of 68 instances, i.e. 83.8% of all cases, did *not* deteriorate, which means that the fixing of variables and the forwarding of a reduced problem to phase-2 was indeed productive. Even more, in almost one third (compare the last entry in (12)) the AMP-approach could outmatch the plain M-BOPH of phase-1 and provide a better result. Thirdly and finally, there is a significant amount of problems, 8.8%, for which not only an improvement, but also the optimal solution could be reached by the exclusive help of phase-2 (see (7) and (13)).

As emphasized, the goal of this research was to demonstrate the usefulness of an AMP-related approach to improve a plain heuristic search algorithm. However, even if not directly targeted as a research topic, we extend the discussion of the computational results with a short remark regarding the CPLEX-gap. For the three groups of instances, aim-50, aim-100 and aim-200, the relative gap between the optimal solution and the heuristic solution obtained with the AMP-approach is on average 0.52%, 0.47% and 0.56%. (Note, due to zero-optimal-solutions, the gap is not calculated with the base CPLEX but standardized with the total number of clauses.) The median, average and maximum gap, with respect to the overall test set, is 0.31%, 0.52% and 2.83%, which appears to be a solid series of small to moderate deviations.

Finally, we conclude with an observation of the running times. CPLEX was run on a different machine, a 2.2 GHz Pentium Celeron PC. In order to accurately compare the CPU usage, we refer to the *Standard Performance Evaluation Corporation* (2014) and their SPEC CPU2006 benchmark suite, assigning a CFP2006 base result of 21.8 for the CPLEX-PC and a comparative value of 16.5 for the AMP-notebook. The average group running times of the AMP-approach are 4.1, 28.3 and 212.5 seconds with a total of 244.9 seconds. This translates to benchmarked CPLEX-times of 7.0, 24.0 and 607.9 with a total of 638.8 seconds. The fact that the heuristic is a little slower for the middle group, aim-100, is positively contradicted by the superior running time performance in aim-50 and aim-100, where the AMP-times improve over CPLEX with 42% and 65%. All in all, the metaheuristic total running time could be reduced to 38% of the LP-CPLEX approach.

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)
Name	*m*	*n*	CPLEX		AMP						
			Copt	*sec.*	*C1**	*k1**	*C2**	*k2**	*C**	*p**	*sec.*
aim-50-1_6-no-1	80	50	1	0.22	1	19	1	1	1	2	0.17
aim-50-1_6-no-2	80	50	1	0.27	1	19	1	1	1	2	0.17
aim-50-1_6-no-3	80	50	1	0.36	1	25	1	10	1	2	0.17
aim-50-1_6-no-4	80	50	1	0.11	1	21	1	1	1	2	0.17
aim-50-1_6-yes1-1	80	50	0	0.22	1	17	1	1	1	2	0.17
aim-50-1_6-yes1-2	80	50	0	0.25	1	15	1	1	1	2	0.17
aim-50-1_6-yes1-3	80	50	0	0.13	1	13	1	10	1	2	0.17
aim-50-1_6-yes1-4	80	50	0	0.16	1	15	1	13	1	2	0.19
aim-50-2_0-no-1	100	50	1	0.22	1	21	1	1	1	2	0.2
aim-50-2_0-no-2	100	50	1	0.56	1	15	1	12	1	2	0.19
aim-50-2_0-no-3	100	50	1	0.11	1	21	1	2	1	2	0.19
aim-50-2_0-no-4	100	50	1	0.48	1	22	1	1	1	2	0.19
aim-50-2_0-yes1-1	100	50	0	0.13	1	35	1	10	1	2	0.19
aim-50-2_0-yes1-2	100	50	0	0.13	2	26	1	12	1	2	0.19
aim-50-2_0-yes1-3	100	50	0	0.06	1	11	2	11	1	1	0.19
aim-50-2_0-yes1-4	100	50	0	0.13	1	10	1	15	1	2	0.19
aim-50-3_4-yes1-1	170	50	0	0.19	2	13	1	11	1	2	0.23
aim-50-3_4-yes1-2	170	50	0	0.33	1	12	3	12	1	1	0.23
aim-50-3_4-yes1-3	170	50	0	0.08	1	13	2	3	1	1	0.23
aim-50-3_4-yes1-4	170	50	0	0.42	4	16	3	12	3	2	0.22
aim-50-6_0-yes1-1	300	50	0	0.06	0	11	-	-	0	1	0
aim-50-6_0-yes1-2	300	50	0	0.23	3	22	0	13	0	2	0.27
aim-50-6_0-yes1-3	300	50	0	0.09	0	12	-	-	0	1	0
aim-50-6_0-yes1-4	300	50	0	0.36	0	11	-	-	0	1	0
aim-100-1_6-no-1	160	100	1	0.44	1	56	1	4	1	2	1.09
aim-100-1_6-no-2	160	100	1	0.55	1	27	1	11	1	2	1.04
aim-100-1_6-no-3	160	100	1	0.95	1	19	1	1	1	2	1.11
aim-100-1_6-no-4	160	100	1	0.19	1	16	1	1	1	2	1.04
aim-100-1_6-yes1-1	160	100	0	0.41	2	27	2	12	2	2	1.11
aim-100-1_6-yes1-2	160	100	0	0.42	1	24	2	11	1	1	1.12
aim-100-1_6-yes1-3	160	100	0	0.22	2	25	1	10	1	2	1.09
aim-100-1_6-yes1-4	160	100	0	1.28	1	12	2	1	1	1	1.04
aim-100-2_0-no-1	200	100	1	0.30	1	17	1	11	1	2	1.25
aim-100-2_0-no-2	200	100	1	1.09	1	27	1	11	1	2	1.2
aim-100-2_0-no-3	200	100	1	0.50	1	44	1	1	1	2	1.22
aim-100-2_0-no-4	200	100	1	1.22	1	91	1	1	1	2	1.22
aim-100-2_0-yes1-1	200	100	0	0.53	2	15	2	11	2	2	1.2
aim-100-2_0-yes1-2	200	100	0	1.05	2	27	1	13	1	2	1.22
aim-100-2_0-yes1-3	200	100	0	0.44	2	18	3	12	2	1	1.19
aim-100-2_0-yes1-4	200	100	0	0.36	2	16	1	11	1	2	1.23
aim-100-3_4-yes1-1	340	100	0	0.36	3	21	7	11	3	1	1.36
aim-100-3_4-yes1-2	340	100	0	1.41	5	13	5	21	5	2	1.34
aim-100-3_4-yes1-3	340	100	0	2.63	5	16	4	15	4	2	1.34

aim-100-3_4-yes1-4	340	100	0	1.59	6	25	5	10	5	2	1.34
aim-100-6_0-yes1-1	600	100	0	0.34	0	14	-	-	0	1	0
aim-100-6_0-yes1-2	600	100	0	0.72	6	12	0	11	0	2	1.51
aim-100-6_0-yes1-3	600	100	0	0.44	2	13	1	11	1	2	1.5
aim-100-6_0-yes1-4	600	100	0	0.69	4	13	0	11	0	2	1.51
aim-200-1_6-no-1	320	200	1	0.20	1	18	1	11	1	2	7.96
aim-200-1_6-no-2	320	200	1	1.13	1	16	1	10	1	2	7.88
aim-200-1_6-no-3	320	200	1	1.59	2	17	1	12	1	2	7.91
aim-200-1_6-no-4	320	200	1	1.36	1	18	1	1	1	2	8.08
aim-200-1_6-yes1-1	320	200	0	0.58	1	24	2	23	1	1	7.82
aim-200-1_6-yes1-2	320	200	0	1.34	2	13	2	11	2	2	7.54
aim-200-1_6-yes1-3	320	200	0	1.36	1	15	1	11	1	2	7.52
aim-200-1_6-yes1-4	320	200	0	1.13	1	12	1	11	1	2	7.78
aim-200-2_0-no-1	400	200	1	1.20	1	12	1	11	1	2	8.6
aim-200-2_0-no-2	400	200	1	3.95	1	16	2	11	1	1	8.7
aim-200-2_0-no-3	400	200	1	1.16	1	15	2	13	1	1	8.78
aim-200-2_0-no-4	400	200	1	1.06	1	27	1	12	1	2	8.88
aim-200-2_0-yes1-1	400	200	0	6.86	3	13	3	12	3	2	8.42
aim-200-2_0-yes1-2	400	200	0	2.14	3	15	3	13	3	2	8.63
aim-200-2_0-yes1-3	400	200	0	1.16	4	15	2	12	2	2	8.58
aim-200-2_0-yes1-4	400	200	0	2.38	4	17	3	12	3	2	8.53
aim-200-3_4-yes1-1	680	200	0	59.78	14	18	11	13	11	2	9.45
aim-200-3_4-yes1-2	680	200	0	80.97	11	12	12	15	11	1	9.63
aim-200-3_4-yes1-3	680	200	0	48.44	12	25	8	18	8	2	9.59
aim-200-3_4-yes1-4	680	200	0	51.03	12	13	11	12	11	2	9.58
aim-200-6_0-yes1-1	1200	200	0	46.06	29	27	2	12	2	2	10.7
aim-200-6_0-yes1-2	1200	200	0	56.81	24	26	0	11	0	2	10.6
aim-200-6_0-yes1-3	1200	200	0	82.47	35	14	34	16	34	2	10.6
aim-200-6_0-yes1-4	1200	200	0	5.94	17	25	0	11	0	2	10.7

Tab. 3: Detailed results.

(1)	(2)	(3)	(4)	(5)	(6)	(7)
group	*C = Copt*	*C1 = Copt*	*phase-2 (C1 > 0)*	*C2 ≤ C1*	*C2 < C1*	*C2opt (C1 > 0)*
aim-50	12	11	21	18	4	1
aim-100	11	9	23	19	8	2
aim-200	10	7	24	20	10	3
sum (#)	33	27	68	57	22	6
	(8)	(9)	(10)	(11)	(12)	(13)
aim-50	16.7%	15.3%	29.2%	26.5%	5.9%	1.5%
aim-100	15.3%	12.5%	31.9%	27.9%	11.8%	2.9%
aim-200	13.9%	9.7%	33.3%	29.4%	14.7%	4.4%
rel. (%)	45.8%	37.5%	94.4%	83.8%	32.4%	8.8%

Tab. 4: Aggregated results – optima and improvements.

5 Conclusion

In this paper we have experimented with a simple multi-start heuristic, utilizing variants of adaptive memory projection whose base tools are pools of high-quality solutions, so-called reference sets, from which information is drawn to help guide the search process. In our approach a fast greedy heuristic (BOPH) is embedded in a multi-start-framework (M-BOPH), running M-BOPH in two subsequent phases. In between these phases, according to the principles of AMP, a problem reduction process is applied, giving the advantage that the problem can be solved more efficiently. This approach quickly produced high quality solutions to a test bed of modest sized test problems. Comparisons of solution quality were made possible via optimal solutions computed with CPLEX.

All told, our results indicate that the approach is viable and holds promise as a competitive methodology for solving the general binary problems. Also, the method has proved being quite an attractive way to tackle Max 3-SAT. However, our results also indicate that there is apparently much room for improvements. Possibilities for future improvement include exploring different settings for *(α,β)*, advanced weighing schedules, which might be controlled by an adaptive memory-oriented mechanism, and more sophisticated base-procedures. Also, more ambitious AMP core-procedures could be developed, like the clustering of the reference set, which could be able to supply more focused variable fixings.

In addition, we have carried out preliminary tests involving the conversion of the cubic penalty function into an equivalent quadratic penalty function and then solving the quadratic problem via well-developed metaheuristics designed for UQBP models and MAX 2-Sat approaches, respectively. These avenues for further investigation are a part of our on-going research agenda.

References

Alidaee, B., Kochenberger, G., Lewis, K., Lewis, M., Wang, H.: Computationally attractive non-linear models for combinatorial optimization. *Int. J. of Math. in OR* 1, 9–20 (2005)

Asahiro Y., Iwama, K., Miyano, E.: Random generation of test instances with controlled attributes. In: Johnson, D.S., Trick, M.A. (eds) *Cliques, coloring, and satisfiability: The second DIMACS implementation challenge.* DIMACS Series on Discrete Mathematics and Theoretical Computer Science 26, pp. 377–394 (1996)

Boros, E., Prekopa, A.: Probabilistic bounds and algorithms for the maximum satisfiability problem. *Annals of OR* 21, 109–126 (1989)

Boros, E., Hammer, P.L.: Pseudo boolean optimization. *Discrete Appl. Math.* 123(1–3), 155–225 (2002)

Cook, S.: The complexity of theorem proving procedures. In: *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, pp. 151–158 (1971)

Crama, Y., Ekin, O., Hammer, P.L.: Variable and term removal from boolean formulae. *Discrete Appl. Math.* 75(3), 217–230 (1997)

Givry, S. de, Larrosa, J., Meseguer, P., Schiex, T.: Solving max-sat as weighted CSP. In: *Proceedings of the 9th CP*, Kinsale, Ireland, 2833, Lecture Notes in Computer Science, pp. 363–376, Springer Verlag (2003)

Glover, F.: A template for scatter search and path relinking. In: Hao, J.-K., Lutton, E., Ronald, E., Schoenauer, M., Snyers, D. (eds.) *Artificial Evolution*, 1363, Lecture Notes in Computer Science, pp. 1–51, Springer, Berlin - Heidelberg (1998)

Glover, F.: Adaptive memory projection methods for integer programming. In: Rego, C., Alidaee, B. (eds.) *Metaheuristic optimization via memory and evolution. Tabu search and scatter search,* Operations Research / ComputerScience Interfaces Series 30, pp. 425–440, Kluwer Academic Publishers, Boston (2005)

Greistorfer, P., Voß, S.: Controlled pool maintenance for metaheuristics. In: Rego, C., Alidaee, B. (eds.) *Metaheuristic optimization via memory and evolution. Tabu search and scatter search,* Operations Research / Computer Science Interfaces Series 30, pp. 387–424, Kluwer Academic Publishers, Boston (2005)

Hammer, P.L., Rudeanu, S.: *Boolean methods in OR.* Springer-Verlag, New York (1968)

Hansen, P., Jaumard, B.: Algorithms for the maximum satisfiability Problem. *Computing* 44, 279–303 (1990)

Kochenberger, G., Glover, F., Alidaee, B., Lewis, K.: Using the unconstrained quadratic program to model and solve max 2-sat problems. *Int. J. of OR* 1, 89–100 (2005)

Larrosa, J.: Current trends in max-sat solving. Working Paper, Llenguatges i Sistemes Informàtics (LSI) department, Universitat Politècnica de Catalunya, Barcelona, Spain (2008)

Levin, L.A.: Universal sequential search problems (Problemy Peredachi Informatsii), 9(3):115–116, Russian Academy of Sciences, Branch of Informatics, Computer Equipment and Automatization (1973)

Motwani, R., Raghavan, P.: *Randomized algorithms.* Cambridge University Press, Cambridge (1995)

Puchinger, J., Raidl, G.R., Pferschy, U.: The core concept for the multidimensional knapsack problem. In: Gottlieb, J., Raidl, G.R. (eds.) *Evolutionary computation in combinatorial optimization*, 3906, Lecture Notes in Computer Science, pp. 195–208, Springer, Berlin - Heidelberg (2006)

Standard Performance Evaluation Corporation (SPEC), 7001 Heritage Village Plaza, Suite 225, Gainesville, VA 20155, www.spec.org (2014). Accessed 15 January 2014