Boolean Lexicographic Optimization

Joao Marques-Silva[1], Josep Argelich[2], Ana Graça[3], and Inês Lynce[3]

[1] CSI/CASL, University College Dublin, Ireland jpms@ucd.ie

[2] DIEI, Universitat de Lleida, Spain jargelich@diei.udl.cat

[3] INESC-ID/IST, Technical University of Lisbon, Portugal
{assg,ines}@sat.inesc-id.pt

Abstract

Multi-Objective Combinatorial Optimization (MOCO) problems find a wide range of practical application problems, some of which involving Boolean variables and constraints. This paper develops and evaluates algorithms for solving MOCO problems, defined on Boolean domains, and where the optimality criterion is lexicographic. The proposed algorithms build on existing algorithms for either Maximum Satisfiability (MaxSAT), Pseudo-Boolean Optimization (PBO), or Integer Linear Programming (ILP). Experimental results, obtained on problem instances from haplotyping with pedigrees, show that the proposed algorithms can provide significant performance gains over state of the art MaxSAT, PBO and ILP algorithms.

1 Introduction

Real-world optimization problems often involve multiple objectives, that can represent conflicting purposes. There has been a large body of work on solving multi-objective combinatorial optimization (MOCO) problems, for example [14, 32, 15]. Nevertheless, some of these MOCO problems have natural Boolean formulations, and so Boolean-based optimization solutions could be expected to provide effective alternative solutions. This paper addresses MOCO problems where the variables are Boolean, the constraints are represented by linear inequalities (or clauses), and the optimization criterion is lexicographic. Given a sequence of cost functions, an optimization criterion is said to be lexicographic whenever there is a preference in the order in which cost functions are optimized. There are many examples where optimization is expected to be lexicograhic. For example, suppose that instead of requiring a balance between price, horsepower and fuel consumption for choosing a new car, you have made a clear hierarchy in your mind: you have a strict limit on how much you can afford, then you will not consider a car with less than 150 horsepower and after that the less the fuel consumption the better. Not only you establish a priority in your preferences, but also each optimization criterion is defined in such a way that the set of potential solutions gets subsequently reduced. Such kind of problems are present not only in your daily life but also in many real applications, and representative examples can be found in recent surveys [15, 32, 14].

This paper develops and evaluates algorithms for Boolean lexicographic optimization problems, and has three main contributions. First, it formalizes Boolean

Proceedings of the 17th International RCRA workshop (RCRA 2010):
Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion
Bologna, Italy, June 10–11, 2010

Lexicographic Optimization. Second, it describes practical algorithms for solving Boolean Lexicographic Optimization, either based on pseudo-Boolean optimization (PBO), 0-1 Integer Linear Programming (ILP), or Maximum Satisfiability (MaxSAT) algorithms. Third, the paper illustrates the practical usefulness of the proposed algorithms. The experimental evaluation is focused on a concrete application, namely haplotyping with pedigree information [18]. Nevertheless, the techniques proposed in this paper are general, and have been successfully used in other contexts [1].

The paper is organized as follows. Section 2 overviews MaxSAT, PBO, and Lexicographic Optimization. Afterwards, Section 3 describes four alternative approaches for solving lexicographic optimization problems. Section 4 conducts a detailed experimental evaluation on hard problem instances from haplotyping with pedigree information. State of the art MaxSAT, PBO and ILP solvers are then compared against the algorithms described in this paper. Section 5 summarizes related work and Section 6 concludes the paper.

2 Preliminaries

This section overviews the Maximum Satisfiability (MaxSAT) problem and its variants, as well as the Pseudo-Boolean Optimization (PBO) problem. The main approaches used by state-of-the-art solvers are summarized, including recent unsatisfiability-based MaxSAT algorithms. To conclude, this section provides a brief overview of Multi-Objective Combinatorial Optimization (MOCO) [14, 15], focusing on lexicographic optimization.

2.1 Maximum Satisfiability

Given a CNF formula $\mathcal{C}$, the Maximum Satisfiability (MaxSAT) problem consists in finding an assignment that maximizes the number of satisfied clauses. Well-known variants of the MaxSAT problem include weighted MaxSAT, partial MaxSAT and weighted partial MaxSAT. All these formulations find a wide range of practical applications [21]. The general weighted partial MaxSAT problem formulation assumes a CNF formula $\mathcal{C}$, where each clause $c \in \mathcal{C}$ is associated with a weight w, and where clauses that must be satisfied have weight $w = \top$. The optimization problem is to find a truth assignment such that the cost of the satisfied clauses is maximized.

The last decade has seen a large number of alternative algorithms for MaxSAT. These can be broadly categorized as branch-and-bound with lower bounding, decomposition-based, translation to pseudo-Boolean constraints and unsatisfiability based. Branch-and-bound algorithms integrate lower bounding and inference techniques, and represent the more mature solutions, i.e. which have been studied more extensively in the past. A recent alternative are unsatisfiability-based algorithms, that are built on the success of modern SAT solvers, and which have been

[1] For example, in a recent competition of solvers for solving package upgradeability problems [3].

shown to perform well on problem instances from practical applications. Examples of branch-and-bound algorithms include: MaxSatz [22], IncMaxSatz [23], WMaxSatz [4], and MiniMaxSAT [20]. A well-known example of translation to PB constraints is SAT4J-MaxSAT [8]. Examples of decomposition-based solvers include Clone [30] and sr(w) [31]. In recent years, several unsatisfiability-based MaxSAT algorithms have been proposed. The original approach was proposed in [17], and recent solvers include MSUnCore [28, 27, 26], WBO [26], WPM1 and PM2 [2].

2.2 Pseudo-Boolean Optimization

Pseudo-Boolean Optimization (PBO) is an extension of SAT where constraints are linear inequalities, with integer coefficients and Boolean variables. The objective in PBO is to find an assignment to problem variables such that all problem constraints are satisfied and the value of a linear objective function is optimized. The PBO normal form [7] is defined as follows:

$$\begin{array}{ll} \text{minimize} & \sum\limits_{j \in N} v_j \cdot l_j \\ \text{subject to} & \sum\limits_{j \in N} a_{ij} l_j \geq b_i, \\ & l_j \in \{x_j, \bar{x}_j\}, x_j \in \{0,1\}, a_{ij}, b_i, v_j \in \mathbf{N}_0^+ \end{array} \tag{1}$$

Observe that any pseudo-Boolean formulation can be translated into a normal form.

Modern PBO algorithms generalize the most effective techniques used in modern SAT solvers. These include unit propagation, conflict-driven learning and conflict-directed backtracking [25, 9]. Despite a number of common techniques, there are several alternative approaches for solving PBO. The most often used approach is to conduct a linear search on the value of the objective function. In addition, the use of binary search has been suggested and evaluated in the recent past [12, 10]. SAT algorithms can be generalized to deal with pseudo-Boolean constraints natively [7] and, whenever a solution to the problem constraints is identified, a new constraint is created such that only solutions corresponding to a lower value of the objective function are allowed. The algorithm terminates when the solver cannot improve the value of the cost function. Another often used solution is based on branch and bound search, where lower bounding procedures to estimate the value of the objective function are used, and the upper bound is iteratively refined. Several lower bounding procedures have been proposed over the years, e.g. [11, 25]. There are also algorithms that encode pseudo-Boolean constraints into propositional clauses [34, 6, 13] and solve the problem by subsequently using a SAT solver. This approach has been proved to be very effective for several problem sets, in particular when the clause encoding is not much larger than the original pseudo-Boolean formulation.

2.3 MaxSAT, PBO & BMO

Although MaxSAT and PBO are different formalisms, there are well-known mappings from MaxSAT to PBO and vice-versa [19, 1, 20]. The remainder of the paper

uses both formalisms interchangeably. A set of clauses or constraints is denoted by $\mathcal{C}$. Without loss of generality, linear constraints are assumed to represent clauses, thus representing instances of the Binate Covering Problem [11]. For the general case where linear constraints represent PB constraints, there are well-known mappings from PB constraints to CNF formulas [34, 13, 33], which could be used if necessary.

Mappings from soft clauses to cost functions and vice-versa are also well-known [20]. For example, suppose the cost function $\min \sum_j v_j \cdot x_j$. A set of soft clauses can replace this cost function: for each x_j create a soft clause $(\bar{x}_j)$ with cost v_j. Similarly, a set of soft clauses can be represented with a cost function. Suppose a set of soft clauses $\mathcal{C}_a$, where each clause $c_j \in \mathcal{C}_a$ is associated a weight w_j. Replace each clause c_j with $c'_j = c_j \vee \bar{s}_j$, where s_j is a relaxation variable, and create the cost function $\min \sum_j w_j \cdot s_j$.

Boolean Multilevel Optimization [5] is a restriction of weighted (partial) MaxSAT, with an additional condition on the clause weights. BMO is defined on a set of sets of clauses $C = C_0 \cup C_1 \cup \ldots \cup C_m$, where $\{C_0, C_1, \ldots, C_m\}$ forms a partition of C, and a weight is associated with each set of clauses: $\langle w_0 = \top, w_1, \ldots, w_m \rangle$, such that w_i is associated with each clause c in each set C_i. C_0 represents the *hard* clauses, each with weight $w_0 = \top$. Although C_0 may be empty, it is assumed that $C_i \neq \emptyset$, $i = 1, \ldots, m$.

Definition 1 (BMO)**.** *An instance of Weighted (Partial) Maximum Satisfiability is an instance of BMO iff the following condition holds:*

$$w_i > \sum_{i+1 \leq j \leq m} w_j \cdot |C_j| \qquad i = 1, \ldots, m-1 \tag{2}$$

BMO can be viewed as a technique for identifying lexicographic optimization conditions in MaxSAT and PBO problem instances. This is further highlighted in the following sections.

2.4 Lexicographic Optimization

Multi-Objective Combinatorial Optimization (MOCO) [14, 32, 15] is a well-known area of research, with many practical applications, including Operations Research and Artificial Intelligence. Lexicographic optimization represents a specialization of MOCO, where the optimization criterion is lexicographic. Motivated by the wide range of practical applications, Lexicographic Optimization is also often referred to Preemptive Goal Programming or Lexicographic Goal Programming [32]. This section introduces Boolean Lexicographic Optimization (BLO), a restriction of lexicographic optimization, where variables are Boolean, all cost functions and constraints are linear, and the optimization criterion is lexicographic. The notation and definitions in this section follow [14], subject to these additional constraints.

A set X of variables is assumed, with $X = \{x_1, \ldots, x_n\}$. The domain of the variables is $\mathcal{X} = \{0, 1\}^n$. A point in $\mathcal{X}$ is represented as $\mathbf{x} \in \mathcal{X}$ or $(x_1, \ldots, x_n) \in \mathcal{X}$. A set of p linear functions is assumed, all of which are defined on Boolean

variables, $f_k : \{0,1\}^n \rightarrow \mathbb{Z}$, $1 \leq k \leq p$:

$$f_k(x_1, \ldots, x_n) = \sum_{1 \leq j \leq n} v_{k,j} \cdot l_j \tag{3}$$

where $l_j \in \{x_j, \bar{x}_j\}$, and $v_{k,j} \in \mathbb{N}_0^+$. The p cost functions capturing the optimization problem represent a multi-dimensional function: $\mathbf{f} : \{0,1\}^n \rightarrow \mathbb{Z}^p$, with $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_p(\mathbf{x}))$.

The optimization problem is defined on these p functions, subject to satisfying a set of constraints:

$$\begin{array}{ll} \text{lexmin} & (f_1(x_1, \ldots, x_n), \ldots, f_p(x_1, \ldots, x_n)) \\ \text{subject to} & \sum\limits_{j \in N} a_{ij} l_j \geq b_i, \\ & l_j \in \{x_j, \bar{x}_j\}, x_j \in \{0,1\}, a_{ij}, b_i \in \mathbb{N}_0^+ \end{array} \tag{4}$$

Any point $\mathbf{x} \in \{0,1\}^n$ which satisfy the constraints is called a *feasible* point.

For any two vectors $\mathbf{y}^1, \mathbf{y}^2 \in \mathbb{Z}^p$, with $\mathbf{y}^1 = (y_1^1, \ldots, y_p^1)$ and $\mathbf{y}^2 = (y_1^2, \ldots, y_p^2)$, the lexicographic comparison ($<_{\text{lex}}$) is defined as follows: $\mathbf{y}^1 <_{\text{lex}} \mathbf{y}^2$ iff $y_q^1 < y_q^2$, where $q = \min\{k : y_k^1 \neq y_k^2\}$. For example, $\mathbf{y}^1 = (1,2,3,2) <_{\text{lex}} \mathbf{y}^2 = (1,2,4,1)$, because the coordinate with the smallest index where $\mathbf{y}^1$ and $\mathbf{y}^2$ differ is the third coordinate, with $y_3^1 = 3 < y_3^2 = 4$.

Definition 2 (Lexicographic Optimality)**.** *A feasible point $\hat{\mathbf{x}} \in \{0,1\}^n$ is lexicographically optimal if there exists no other $\mathbf{x}$ such that $\mathbf{f}(\mathbf{x}) <_{\text{lex}} \mathbf{f}(\hat{\mathbf{x}})$.*

3 Boolean Lexicographic Optimization

This section describes three different algorithmic approaches for solving Boolean Lexicographic Optimization problems as defined in Section 2.4. Each algorithm uses the most suitable problem representation, i.e. either PBO or MaxSAT. As a result, Section 3.2 assumes a PBO formulation, whereas Sections 3.3 and 3.4 assume a MaxSAT formulation. Moreover, since MaxSAT problems can be mapped to PBO and vice-versa [1, 20, 26], the algorithms described in this section can be applied to either class of problems.

3.1 Aggregated Cost Function

A simple solution for solving Lexicographic Optimization problems is to aggregate the different functions into a single weighted cost function [29]. In this case, any MaxSAT or PBO algorithm can be used for solving BLO problems. The aggregation is organized as follows. Let $u_k = \sum_j v_{k,j}$ denote the upper bound on the value of f_k. Then define $w_p = 1$, and $w_i = 1 + \sum_{k=i+1}^{p} w_k \cdot u_k$. The aggregated cost function becomes:

$$\min \sum_{k=1}^{p} w_k \cdot \left(\sum_{j=1}^{n} v_{k,j} \cdot l_j \right) \tag{5}$$

Input : $f_1, f_2, \ldots, f_p, \mathcal{C}$
Output: Lexicographic Optimum Solution

for $k \leftarrow 1$ **to** p **do**
  $\mu \leftarrow \texttt{PBO}(\min f_k, \mathcal{C})$
  $\mathcal{C}_k \leftarrow (f_k = \mu)$ `// Cost function` k `must equal` μ
  $\mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{C}_k$ `// Update set of constraints`
  $\mu_k \leftarrow \mu$ `// Record min cost for` k
return $(\mu_1, \ldots, \mu_p)$

Algorithm 1: PBO-based BLO algorithm

subject to the same constraints. Alternatively, the cost function can be represented as a set of weighted soft clauses. In this case, the problem instance can be mapped to PBO, with a unique cost function, where the weights are modified as outlined above.

The main drawback of this solution is that exponentially large weights need to be used in the aggregated cost function. For a MaxSAT approach, this results in large weights being associated with some of the soft clauses.

3.2 Iterative Pseudo-Boolean Solving

An alternative solution for Boolean lexicographic optimization was proposed in the context of BMO [5]. A formalization of this approach is shown in Algorithm 1, and essentially represents an instantiation of the standard approach for solving lexicographic optimization problems [14]. The algorithm executes a sequence of p calls to a PBO solver, in decreasing lexicographic order. At each iteration, the solution of the PBO problem instance is recorded, and a new constraint is added to the set of constraints, requiring the cost function k to be equal to the computed optimum value. After the p iterations, the algorithm identified each of the optimum values for each of the cost functions in the lexicographically ordered cost function. One important remark is that *any* PBO or ILP solver can be used.

The main potential drawbacks of the PB-based approach are: (1) PB constraints resulting from the cost functions need to be handled natively, or at least encoded to CNF; (2) each iteration of the algorithm yields only one new constraint on the value of the cost function k. Clearly, clause reuse could be considered, but would require tighter integration with the underlying solver.

3.3 Iterative MaxSAT Solving with Weight Rescaling

Another alternative approach is inspired on branch-and-bound MaxSAT algorithms, and consists of iteratively rescaling the weights of the soft clauses [5]. Algorithm 2 shows this approach. At each one of the p steps, it finds the optimum value μ_k of the current set $\mathcal{C}_k$. The function `RescaleWeights` computes the weights for the clauses taking into account the previous solutions for each one of the sets. For example, if set $\mathcal{C}_{0<i<k}$ has μ_i unsatisfied clauses, the weight for the set $\mathcal{C}_{i-1}$ can be

Input : Sets of clauses $\langle \mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_p \rangle$ with corresponding weights $\langle \top, w_1, \ldots, w_p \rangle$
Output: Lexicographic Optimum Solution

for $k \leftarrow 1$ **to** p **do**
  $\mathcal{C} \leftarrow \mathcal{C}_0 \cup \cdots \cup \mathcal{C}_k$ `// Current set of clauses`
  $\mathcal{C}_{rsc} \leftarrow \texttt{RescaleWeights}(\mathcal{C}, \langle \mu_1, \ldots, \mu_k - 1 \rangle)$ `// Update clause weights`
  $o \leftarrow \texttt{MaxSAT}(\mathcal{C}_{rsc})$
  $\mu_k \leftarrow \texttt{GetMinCost}(\mathcal{C}_{rsc}, o, \langle \mu_1, \ldots, \mu_k - 1 \rangle)$ `// Record min cost for` k
return $(\mu_1, \ldots, \mu_p)$

Algorithm 2: MaxSAT-based BLO algorithm with weight rescaling

Input : Sets of clauses $\langle \mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_p \rangle$ with corresponding weights $\langle \top, w_1, \ldots, w_p \rangle$
Output: Lexicographic Optimum Solution

$\mathcal{C}_H \leftarrow \mathcal{C}_0$ `// Initial hard clauses`
for $k \leftarrow 1$ **to** p **do**
  $\mathcal{C}_S \leftarrow \mathcal{C}_k$ `// Current soft clauses`
  $(\mu, \{\mathcal{C}_H^r, \mathcal{C}_S^r\}) \leftarrow \texttt{UnsatMaxSAT}(\{\mathcal{C}_H, \mathcal{C}_S\})$
  $\mu_k \leftarrow \mu$ `// Record min cost for` k
  $\mathcal{C}_H \leftarrow \mathcal{C}_H^r \cup \mathcal{C}_S^r$ `// Harden soft clauses`
return $(\mu_1, \ldots, \mu_p)$

Algorithm 3: Unsatisfiability-based MaxSAT BLO algorithm

$w_i \cdot (\mu_i + 1)$, which can be lower than $w_i \cdot (|\mathcal{C}_i| + 1)$. The function `GetMinCost` translates the optimum solution given by the MaxSAT solver, that involves all the sets of clauses up to $\mathcal{C}_k$, to the number of unsatisfied clauses of current set $\mathcal{C}_k$ associated to μ_k. The weights returned by the algorithm may affect the original weights, such that $\mu_i \leq w_i$. The same holds for the weight associated with hard clauses as it depends on the weights given to soft clauses.

Although the rescaling method is effective at reducing the weights that need to be considered, for very large problem instances the challenge of large clause weights can still be an issue.

3.4 Iterative Unsatisfiability-Based MaxSAT Solving

Our final approach for solving BLO problems is based on unsatisfiability-based MaxSAT algorithms. A possible organization is shown in Algorithm 3.

Similarly to the organization of the other algorithms, Algorithm 3 executes p iterations, and each cost function is analyzed separately, in order. At each step a (unsatisfiability-based) partial MaxSAT solver is called on a set of hard and soft

clauses. The result corresponds to the minimum unsatisfiability cost for the set of clauses C_k. In contrast to previous algorithms, the CNF formula is modified in each iteration. Clauses relaxed by the unsatisfiability-based MaxSAT algorithm are kept and become *hard* clauses for the next iterations. The hardening of soft clauses after each iteration can be justified by associating sufficiently large weights with each cost function. When analyzing cost function k, relaxing clauses associated with cost functions 1 to $k-1$ is irrelevant for computing the optimum value at iteration k.

The unsatisfiability-based lexicographic optimization approach inherits some of the drawbacks of unsatisfiability-based MaxSAT algorithms. One example is that, if the minimum cost unsatisfiability cost is large, then the number of iterations may render the approach ineffective. Another drawback is that, when compared to the previous algorithms, a tighter integration with the underlying MaxSAT solver is necessary. Interestingly, a well-known drawback of unsatisfiability-based MaxSAT algorithms is addressed by the lexicographic optimization approach. Unsatisfiability-based MaxSAT algorithms iteratively refine lower bounds on the minimum unsatisfiability cost. Hence, in case the available computational resources are exceeded, the algorithm terminates without providing an approximate solution to the original problem. In contrast, the lexicographic optimization approach allows obtaining intermediate solutions, each representing an upper bound on the minimum unsatisfiability cost.

3.5 Discussion

The previous sections describe four different approaches for solving Boolean lexicographic optimization problems. Some of the main drawbacks were identified, and will be evaluated in the results section. Although the proposed algorithms return a vector of optimum cost function values, it is straightforward to obtain an aggregated result cost function, e.g. using (5). Moreover, and besides serving to solve complex lexicographic optimization problems, the proposed algorithms can provide useful information in practical settings. For example, the iterative algorithms provide partial solutions (satisfying some of the target criteria) during their execution. These partial solutions can be used to provide approximate solutions in case computing the optimum value exceeds available computation resources. Given that all algorithms analyze the cost functions in order, the approximate solutions will in general be much tighter than those provided by algorithms that refine an upper bound (e.g. Minisat+).

The proposed techniques can also be used for solving already existing problem instances. Indeed, existing problem instances may encode lexicographic optimization in the cost function or in the weighted soft clauses. This information can be exploited for developing effective solutions [5]. For example, the BMO condition essentially identifies PBO or MaxSAT problem instances where lexicographic optimization is modelled with an aggregated cost function (represented explicitly or with weighted soft clauses) [5]. In some settings, this is an often used modelling solution. Hence, the BMO condition can be interpreted as an approach to convert from an aggregated cost function to a lexicographic optimization problem.

Finally, the algorithms proposed proposed in the previous sections accept cost functions *implicitly* specified by soft constraints. This provides an added degree of modelling flexibility when compared with the abstract definition of (Boolean) lexicographic optimization.

4 Results

This section evaluates existing state of the art PBO and MaxSAT solvers, as well as the algorithms described in the previous section, on lexicographic optimization problem instances resulting from haplotyping with pedigree information [18]. The problem of haplotyping with pedigrees is an example of lexicographic optimization, because there are two cost functions, and preference is given to one of the cost functions. Albeit this section focuses on a concrete application, similar conclusions have been independently obtained in a recent competition for solving software package upgradeability problems [**?**]. For the haplotyping with pedigree information problem instances, there are two cost functions. For the software upgradeability problem instances there are either two or three cost functions.

All experimental results were obtained on 3 GHz Xeon 5160 servers, with 4 GB of RAM, and running RedHat Enterprise Linux. The CPU time limit was set to 1000 seconds, and the memory limit was set to 3.5 GB. For the experimentation, a well-known commercial ILP solver as well as the best performing PBO and MaxSAT solvers of the most recent evaluations[2] were considered. As a result, the following solvers were used in the experiments: CPLEX, SCIP, Minisat+, BSOLO, MiniMaxSat, MSUnCore, WPM1, SAT4J-PB, SAT4J-MaxSAT. Other well-known MaxSAT solvers (among the best performing in the MaxSAT evaluations) were also considered. However, the large size and intrinsic hardness of the problem instances resulted in these solvers being unable to provide results for any instance. Consequently, these solvers were discarded.

The haplotyping with pedigrees problem instances can be generated with different optimizations to the core model [18]. For the results presented in this section, 500 of the most difficult problem instances were selected. These instances are the most difficult for the best performing solver; hence any other instances would be easier to solve by the best performing solver. The results are organized in two parts. The first part evaluates the number of instances aborted within the CPU time and physical memory limits. The second part compares the CPU times. In all cases, the focus is on evaluating the effectiveness of the proposed algorithms for solving lexicographic optimization problems. The approach considered as default corresponds to the aggregated cost function algorithm.

Table 1 shows the number of aborted instances, i.e. instances that a given solver cannot prove the optimum within the allowed CPU time limit or memory limit. The results allow drawing several conclusions. First, for some solvers, the use of dedicated lexicographic optimization algorithms can provide remarkable performance improvements. A concrete example is Minisat+. The default solver aborts most problem instances, whereas Minisat+ integrated in an iterative

[2] `http://www.maxsat.udl.cat/` and `http://www.cril.univ-artois.fr/PB09/`.

BLO Solution	Solver	# Aborted	
		Plain	BLO
Default Solvers	CPLEX	465	**464**
Iterated PBO	Minisat+	496	**56**
	BSOLO	**456**	500
	SCIP	495	**474**
	SAT4J-PB	463	**435**
Iterated MaxSAT with Rescaling	SAT4J-MaxSAT	464	**404**
	WPM1	**69**	72
	MSUnCore	**84**	85
	MiniMaxSat	500	500
Iterated Unsat-based MaxSAT	MSUnCore	84	**51**

Table 1: Aborted problems instances (out of 500)

pseudo-Boolean BLO solver ranks among the best performing solvers, aborting only 56 problem instances (i.e. the number of aborted instances is reduced in more than 85%). Second, for some other solvers, the performance gains are significant. This is the case with MSUnCore. For MSUnCore, the use of unsatisfiability-based lexicographic optimization reduces the number of aborted faults in close to 40%. SAT4J-PB integrated in an iterative pseudo-Boolean solver, reduces the number of aborted instances in close to 6%. Similarly, SCIP integrated in an iterative pseudo-Boolean solver, reduces the number of aborted instances in more than 4%. Despite the promising results of using iterative pseudo-Boolean solving, there are examples for which this approach is not effective, e.g. BSOLO. This suggests that the effectiveness of this solution depends strongly on the type of solver used and on the target problem instances. The results for the MaxSAT-based weight rescaling algorithm are less conclusive. There are several justifications for this. Given that existing branch and bound algorithms are unable to run large problem instances, the MaxSAT solvers considered are known to be less dependent on clause weights.

Figures 1, 2, 3, 4 and 5 show scatter plots comparing the run times for different solvers on the same problem instances. Each plot compares two different approaches, where each point represents one problem instance, being the x-axis value given by one approach and the y-axis value given by the other. Again, several conclusions can be drawn. Figures 1, 2 and 3 confirm the effectiveness of the algorithms proposed in this paper, namely iterative PB solving and iterative unsatisfiability-based MaxSAT. Despite the remarkable improvements in the performance of Minisat+ when integrated in a lexicographic optimization algorithm, MSUnCore integrated in lexicographic optimization algorithm provides the best performance in terms of aborted problem instances. Nevertheless, the use of BLO adds overhead to the solvers, and so for most instances the best performance is obtained with the default solver WPM1. These conclusions are further highlighted in Figures 4 and 5. Although WPM1 is the best performing algorithm without lexicographic optimization support, MSUnCore with lexicographic optimization provides more robust performance, namely for the hardest problem instances.

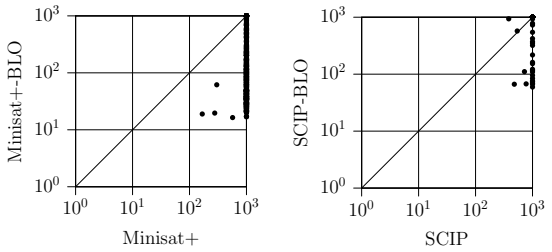

Figure 1: Original Minisat+ and SCIP vs. iterated pseudo-Boolean solving

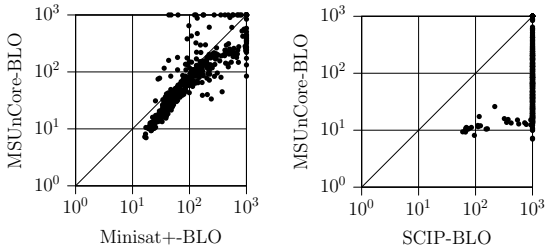

Figure 2: Iterated unsat-based MaxSAT vs. iterated pseudo-Boolean solving

4.1 Discussion

The experimental results provide useful insights about the behavior of the solvers considered. For example, Minisat+ performs poorly when an aggregated cost function is used. However, it is among the best performing solvers when integrated in the iterative pseudo-Boolean solving framework. In contrast, the improvements to SCIP are far less notable. These performance differences are explained by the structure of the problem instances. For Minisat+ the main challenge is the relatively complex aggregated cost function. Hence, the use of iterated pseudo-Boolean solving eliminates this difficulty, and so Minisat+ is able to perform very effectively on the resulting problem instances, which exhibit hard to satisfy Boolean constraints. In contrast, SCIP is less sensitive to the cost function, and the iterative approach does not help with solving the (hard to solve) Boolean constraints. Hence, the use of iterative pseudo-Boolean solving is less effective for SCIP for the problem instances considered. The results for MSUnCore and iterative unsatisfiability-based MaxSAT indicate that the use of BLO solvers provides added robustness, at the cost of additional overhead for problem instances that are easy to solve.

5 Related Work

Recent surveys on MaxSAT and PBO solvers are available in [21, 33]. Lexicographic optimization has a long history of research, with many different algorithms and applications. Examples of recent surveys are provided in [14, 15, 32]. The iterative pseudo-Boolean solving approach (see Section 3.2) is tightly related with the standard organization of Lexicographic Optimization algorithms [14].

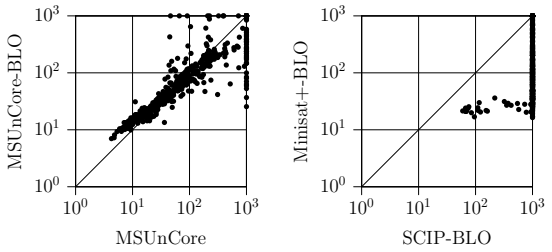

Figure 3: Comparison of BLO solutions

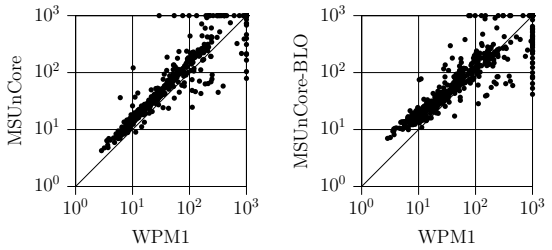

Figure 4: BLO improvement on MSUnCore vs. WPM1

In the area of Boolean-based optimization procedures, there has been preliminary work on solving pseudo-Boolean MOCO problems [24]. Nevertheless, this work addresses exclusively Pareto optimality, and does not cover lexicographic optimization. In the area of constraints and preferences, lexicographic optimization has been the subject of recent work (for example, [16]), but the focus has been on the use of standard CSP algorithms.

6 Conclusions and Future Work

This paper formalizes the problem of Boolean lexicographic optimization (BLO), and develops algorithms for this problem. General lexicographic optimization is a well-known variant of multi-objective combinatorial optimization [14], with a large number of practical applications. The restriction considered in this paper assumes Boolean variables, linear constraints and linear cost functions.

The paper outlines four different algorithmic solutions for BLO, either based on aggregating cost functions in a single cost function, iterative pseudo-Boolean solving, iterative MaxSAT with weight rescaling and iterative unsatisfiability-based MaxSAT. The four algorithmic solutions were evaluated on complex lexicographic optimization problem instances from haplotyping with pedigree information [18]. The experimental evaluation allows drawing several conclusions. First, the use of a single aggregated cost function can impact performance negatively. Second, the use of iterative solutions (either based on PBO solvers or on unsatisfiability-based MaxSAT solvers) can yield significant performance gains when compared with the original solvers, resulting in remarkable reductions in the number of problem

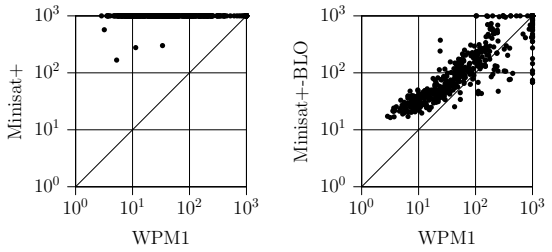

Figure 5: BLO improvements on Minisat+ vs. WPM1

instances unsolved. Despite the experimental evaluation focusing on the concrete case of haplotyping with pedigree information, recent experimental evaluations in the area of software upgradeability [**?**] yielded similar conclusions.

Future work will address tighter integration between default solvers and the algorithms for lexicographic optimization. Concrete examples include clause reuse and incremental interface with the default solvers.

Acknowledgments Claude Michel provided insights on Lexicographic Optimization. This work was partially funded by the SFI PI Grant 09/IN.1/I2618, FP7 EU grants ICT/217069 and ICT/214898, FCT Project PTDC/EIA/64164/2006 and PhD Grant SFRH/BD/28599/2006, and Spanish CICYT Projects TIN2007-68005-C04-01/02 and TIN2009-14704-C03-01/02.

References

[1] F. A. Aloul, A. Ramani, I. L. Markov, and K. A. Sakallah. Generic ILP versus specialized 0-1 ILP: an update. In *International Conference on Computer-Aided Design*, pages 450–457, November 2002.

[2] C. Ansótegui, M. L. Bonet, and J. Levy. Solving (weighted) partial MaxSAT through satisfiability testing. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 427–440, July 2009.

[3] J. Argelich, D. L. Berre, I. Lynce, J. Marques-Silva, and P. Rapicault. Solving linux upgradeability problems using Boolean optimization. In *FLoC Workshop on Logics for Component Configuration (LoCoCo)*, 2010.

[4] J. Argelich, C. M. Li, and F. Manyà. An improved exact solver for partial Max-SAT. In *International Conference on Nonconvex Programming: Local and Global Approaches*, pages 230–231, December 2007.

[5] J. Argelich, I. Lynce, and J. Marques-Silva. On solving Boolean multilevel optimization problems. In *International Joint Conference on Artificial Intelligence*, pages 393–398, July 2009.

[6] O. Bailleux, Y. Boufkhad, and O. Roussel. A translation of pseudo Boolean constraints to SAT. *Journal on Satisfiability, Boolean Modeling and Computation*, 2:191–200, March 2006.

[7] P. Barth. A Davis-Putnam enumeration algorithm for linear pseudo-Boolean optimization. Technical Report MPI-I-95-2-003, Max Plank Institute for Computer Science, 1995.

[8] D. L. Berre. SAT4J library. www.sat4j.org.

[9] D. Chai and A. Kuehlmann. A fast pseudo-Boolean constraint solver. In *Design Automation Conference*, pages 830–835, June 2003.

[10] A. Cimatti, A. Franzén, A. Griggio, R. Sebastiani, and C. Stenico. Satisfiability modulo the theory of costs: Foundations and applications. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 99–113, March 2010.

[11] O. Coudert. On solving covering problems. In *Design Automation Conference*, pages 197–202, June 1996.

[12] N. Eén and N. Sörensson. Translating pseudo-Boolean constraints into SAT. *Journal on Satisfiability, Boolean Modeling and Computation*, 2(3-4):1–25, 2006.

[13] N. Een and N. Sörensson. Translating pseudo-Boolean constraints into SAT. *Journal on Satisfiability, Boolean Modeling and Computation*, 2:1–26, March 2006.

[14] M. Ehrgott. *Multicriteria Optimization*. Springer, 2005.

[15] M. Ehrgott and X. Gandibleux. A survey and annotated bibliography of multiobjective combinatorial optimization. *OR Spektrum*, 22(4):425–460, November 2000.

[16] E. Freuder, R. Heffernan, R. Wallace, and N. Wilson. Lexicographically-ordered constraint satisfaction problems. *Constraints*, 15(1):1–28, 2010.

[17] Z. Fu and S. Malik. On solving the partial MAX-SAT problem. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 252–265, August 2006.

[18] A. Graça, I. Lynce, J. Marques-Silva, and A. L. Oliveira. Haplotype inference combining pedigrees and unrelated individuals. In *Workshop on Constraint Based Methods for Bioinformatics*, September 2009.

[19] P. Hansen and B. Jaumard. Algorithms for the maximum satisfiability problem. *Computing*, 44(4):279–303, December 1990.

[20] F. Heras, J. Larrosa, and A. Oliveras. MiniMaxSat: An efficient weighted Max-SAT solver. *Journal of Artificial Intelligence Research*, 31:1–32, January 2008.

[21] C. M. Li and F. Manyà. MaxSAT, hard and soft constraints. In *SAT Handbook*, pages 613–632. IOS Press, 2009.

[22] C. M. Li, F. Manyà, and J. Planes. New inference rules for Max-SAT. *Journal of Artificial Intelligence Research*, 30:321–359, October 2007.

[23] H. Lin and K. Su. Exploiting inference rules to compute lower bounds for MAX-SAT solving. In *International Joint Conference on Artificial Intelligence*, pages 2334–2339, January 2007.

[24] M. Lukasiewycz, M. Glaß, C. Haubelt, and J. Teich. Solving multi-objective pseudo-Boolean problems. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 56–69, May 2007.

[25] V. Manquinho and J. Marques-Silva. Satisfiability-based algorithms for Boolean optimization. *Annals of Mathematics and Artificial Intelligence*, 40(3-4):353–372, 2004.

[26] V. Manquinho, J. Marques-Silva, and J. Planes. Algorithms for weighted Boolean optimization. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 495–508, July 2009.

[27] J. Marques-Silva and V. Manquinho. Towards more effective unsatisfiability-based maximum satisfiability algorithms. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 225–230, March 2008.

[28] J. Marques-Silva and J. Planes. Algorithms for maximum satisfiability using unsatisfiable cores. In *Design, Automation and Testing in Europe Conference*, pages 408–413, March 2008.

[29] N. V. Phillips. A weighting function for pre-emptive multicriteria assignment problems. *The Journal of the Operational Research Society*, 38(9):797–802, September 1987.

[30] K. Pipatsrisawat, A. Palyan, M. Chavira, A. Choi, and A. Darwiche. Solving weighted Max-SAT problems in a reduced search space: A performance analysis. *Journal on Satisfiability Boolean Modeling and Computation*, 4:191–217, 2008.

[31] M. Ramírez and H. Geffner. Structural relaxations by variable renaming and their compilation for solving MinCostSAT. In *International Conference on Principles and Practice of Constraint Programming*, pages 605–619, September 2007.

[32] C. Romero. Extended lexicographic goal programming: a unifying approach. *Omega*, 29(1):63–71, February 2001.

[33] O. Roussel and V. Manquinho. Pseudo-Boolean and cardinality constraints. In *SAT Handbook*, pages 695–734. IOS Press, 2009.

[34] J. P. Warners. A linear-time transformation of linear inequalities into conjunctive normal form. *Information Processing Letters*, 68(2):63–69, 1998.