J Intell Inf Syst (2017) 49:87–118
DOI 10.1007/s10844-016-0422-7

Constraint-based and SAT-based diagnosis of automotive configuration problems

Rouven Walter[1] · **Alexander Felfernig**[2] · **Wolfgang Küchlin**[1]

Received: 15 November 2015 / Revised: 19 May 2016 / Accepted: 29 June 2016 /
Published online: 12 July 2016

Abstract We compare the concepts and computation of optimized diagnoses in the context of Boolean constraint based knowledge systems of automotive configuration, namely the *preferred minimal diagnosis* and the *minimum weighted diagnosis*. In order to restore the consistency of an over-constrained system w.r.t. a strict total order of the user requirements, the preferred minimal diagnosis tries to keep the most preferred user requirements and can be computed, for example, by the FASTDIAG algorithm. In contrast, partial weighted MinUNSAT solvers aim to find a set of unsatisfied clauses with the minimum sum of weights, such that the diagnosis is of minimum weight. It turns out that both concepts have similarities, i.e., both deliver an optimal minimal correction subset. We show use cases from automotive configuration where optimized diagnoses are desired. We point out theoretical commonalities and prove the reducibility of both concepts to each other, i.e., both problems are FP^{NP}-complete, which was an open question. In addition to exact algorithms we present greedy algorithms. We evaluate the performance of exact and greedy algorithms on problem instances based on real automotive configuration data from three different German car manufacturers, and we compare the time and quality tradeoff.

Keywords Diagnosis · Preferences · Optimization · Constraints · SAT · Automotive · Configuration

✉ Rouven Walter
rouven.walter@googlemail.com

Alexander Felfernig
alexander.felfernig@ist.tugraz.at

[1] Symbolic Computation Group, WSI Informatics, Eberhard-Karls-Universität, Tübingen, Germany
www-sr.informatik.uni-tuebingen.de

[2] Institute for Software Technology, Graz University of Technology, Graz, Austria www.felfernig.eu

1 Introduction

Constraint Programming (CP) has been successfully applied in many different areas, such as planning, scheduling, and configuration. In this approach, configuration is treated as a Constraint Satisfaction Problem (CSP) on the configuration rules or restrictions. Many car companies (such as GM, VW, AUDI, Daimler and BMW), also known as automotive Original Equipment Manufacturers (OEMs), denote their configuration rules in (dialects of) the more restrictive Propositional Logic. In this case, the configuration constraints can be compiled directly into a single formula, and a SAT-solver can then be used for various verification or configuration purposes. This approach was initiated by Sinz et al. (Küchlin and Sinz 2000; Sinz et al. 2003) in the late 1990s for Mercedes-Benz cars and trucks. There, the configuration knowledge base is known as the *product overview*, and hence the compilation has been termed the *product overview formula* (POF); a more neutral term would be *product description*. This approach has since been commercialized, and corresponding verification software is now in daily productive use at three German premium OEMs.

In many practical use cases the configuration knowledge base can become temporarily over-constrained, e.g., by overly restrictive rules or user requirements (Walter et al. 2013). A typical situation would be a customer configuring a car, with her wishes conflicting with the knowledge base. Another typical situation would be an engineer given the task that new features should be made available for an existing model type. But the features were not available (and hence not configurable) for the existing model type before. In both situations we would like to have an automatic reasoning procedure for assistance in order to restore consistency.

Whenever we face a situation where a knowledge base is over-constrained, there are two possibilities to provide guidance to restore consistency. One approach is to guide the user by computing *minimal unsatisfiable cores* (conflicts), which can be considered as a problem explanation. However, more than one conflict is involved in general. Another approach is to try to satisfy as many of the constraints as possible by finding a *maximal satisfiable subset* (MSS), or by the opposite, finding a *minimum correction subset* (MCS) which can be considered as a repair suggestion or a diagnosis. All the constraints of an MCS have to be removed or altered in order to restore consistency, showing the customer which wishes to change, or the engineer where to redesign the car.

An MCS can be optimized in different ways: (i) when considering a strict total order on the user requirements we can optimize an MCS such that the most preferred user requirements will be kept and the MCS consists of less preferred user requirements, called the *preferred minimal diagnosis*. For example, the FASTDIAG algorithm (Felfernig et al. 2012; O'Callaghan et al. 2005) computes the preferred minimal diagnosis. Or (ii) when considering weights assigned to the user requirements we can compute a MCS with the minimum sum of weights, called the *minimum weighted diagnosis* which corresponds to the *partial weighted MinUNSAT* problem (Li and Manyà 2009) (resp. the dual problem *partial weighted MaxSAT*). The partial weighted MinUNSAT problem is a generalization of the well-known *satisfiability* (SAT) problem (Franco and Martin 2009). Both optimization approaches can be considered as an optimal diagnosis w.r.t. their definition of optimum. In this work, we study both approaches, and make the following contributions:

1. (∗) We give a detailed list of use cases from automotive configuration using optimal diagnoses for re-configuration scenarios and for improving example configurations.
2. We introduce definitions of both concepts, list established approaches (including an enhanced version of FASTDIAG) and point out theoretical similarities.

3. (∗) We show that, in the context of Propositional Logic, both problems, the computation of the preferred minimal diagnosis and the computation of the minimum weighted diagnosis, are reducible to each other and that both are FP^{NP}-complete.
4. We present a greedy approach from one concept to the other and vice versa.
5. (∗) We show experimental evaluations based on real automotive configuration data from three different premium German car manufacturer for the (exact and greedy) computation of an optimized diagnosis.

This work is a significantly extended version of (Walter et al. 2015). New or significantly extended contributions are marked by an asterisk (∗) in the list above.

To the best of our knowledge, it has not been proven before that the computation of the preferred minimal diagnosis (corresponding to the A-preferred MCS / L-preferred MSS in the context of Propositional Logic) is FP^{NP}-hard. Marques-Silva et al. (Marques-Silva et al. 2013, Remark 1) raised this question as an open issue.

The remainder of the paper is structured as follows: Section 2 introduces the formal background and notations. Section 3 shows the definitions of the preferred minimal diagnosis and the minimum weighted diagnosis. Section 4 shows a detailed list of use cases from the automotive configuration. In Section 5 we give an overview of solving techniques for the computation of the optimal diagnosis and point out similarities. Section 6 is an analysis of the computational complexity of the preferred minimal diagnosis and the minimum weighted diagnosis. In Sections 7 and 8 we show how to reduce one problem to each other and how to build up a greedy approach. Section 9 we present experimental evaluations. In Section 10 we discuss related work and finally, in Section 11 we conclude this work and discuss future research topics.

2 Preliminaries

Within the scope of this work we focus on Propositional Logic over the standard operators $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$ with constants $\bot$ and $\top$, representing false and true, respectively. The set of variables of a Boolean formula φ is denoted by $\texttt{vars}(\varphi)$. A variable assignment β is a mapping from $\texttt{vars}(\varphi)$ to $\{0, 1\}$. The evaluation of formula φ by a variable assignment β is denoted by $\beta(\varphi)$ and follows the standard evaluation of Propositional Logic. If $\beta(\varphi) = 1$, then β is a *satisfying assignment* or *model* of φ and φ is called *satisfiable*. Otherwise, if $\beta(\varphi) = 0$, then β is an *unsatisfying* assignment. We denote these two situations by $\beta \models \varphi$ and $\beta \not\models \varphi$, respectively. The well-known NP-complete SAT problem (Cook 1971; Franco and Martin 2009) asks for a given formula whether it is satisfiable.

A Boolean formula φ is in *conjunctive normal form* (CNF) if, and only if, it consists of a conjunction of clauses $\varphi = \bigwedge_{i=1}^{m} c_i$, where a clause c_i is a disjunction of literals. A literal is a variable or its negation. A formula in CNF can be interpreted as a set of clauses $\varphi = \{c_1, \ldots, c_m\}$ and further, a clause can be interpreted as a set of literals. Most of the time we will use the notation of a set of clauses to simplify reading.

Two formulas are *semantically equivalent* if, and only if, their evaluation by any variable assignment β is the same. Two formulas are *equisatisfiable* if, and only if, either both formulas are satisfiable or both formulas are unsatisfiable.

The transformation of a Boolean formula into a semantically equivalent CNF takes an exponential number of steps in the worst case. However, any Boolean formula can be transformed into an equisatisfiable CNF in polynomial time by Tseitin (Tseitin 1970) or the more compact Plaisted-Greenbaum (Plaisted and Greenbaum 1986) transformation using

fewer clauses. We denote such a transformation by $\texttt{tseitin}(\varphi)$. Both transformations have the useful property that, if φ is satisfiable, the satisfying assignments are the same when restricted to the variables of φ. We denote a variable assignment restricted to the variables of $\texttt{vars}(\varphi)$ by $\beta|_{\texttt{vars}(\varphi)}$.

Proposition 1 *(Tseitin/Plaisted-Greenbaum Model Property) Let φ be a Boolean formula.*

$$\{\beta \mid \beta \models \varphi\} = \{\beta|_{\texttt{vars}(\varphi)} \mid \beta \models \texttt{tseitin}(\varphi)\}$$

Nowadays SAT solvers are mostly CNF solvers. With the Tseitin/Plaisted-Greenbaum transformation we can efficiently transform any Boolean formula into a CNF in polynomial time and provide a model, if φ is satisfiable, by simply discarding all auxiliary variables introduced during the transformation.

3 Optimal diagnosis

We will now consider different ways of optimizing a diagnosis. We start by defining a *minimal diagnosis*, afterwards consider preferences resulting in the *preferred minimal diagnosis* and consider weights resulting in the *minimum weighted diagnosis*.

3.1 Minimal diagnosis

In applications, we typically consider a set of *hard* constraints φ_{h} (arbitrary Boolean formulas) which must be satisfied, such as technical or legal constraints of the product overview φ_{POF} (see Section 4). On the other hand, we consider a set φ_{s} of *soft* constraints (arbitrary Boolean formulas) which may also be relaxed, such as constraints from marketing or from user requirements.

Remark 1 Without loss of generality we can assume that the sets φ_{h} and φ_{s} are clause sets. By applying the following two steps, we can transfer the constraint sets φ_{h} and φ_{s} into clause sets in polynomial time:

1. For each Boolean formula $\psi \in \varphi_{\text{s}}$ introduce a fresh variable b and add the implication $b \rightarrow \psi$ to the hard part φ_{h}. Then replace ψ in φ_{s} by the unit clause $\{b\}$ (cf. Argelich and Manyà 2006; Heras et al. 2012a).
2. Replace each Boolean formula $\psi \in \varphi_{\text{h}}$ by the clause set $\texttt{tseitin}(\psi)$. The constraint ψ and the transformation $\texttt{tseitin}(\psi)$ share the same models w.r.t. the original variables, see Proposition 1. Thus, the search space between both remains the same.

Because of Remark 1 we will from now on assume that both, φ_{h} and φ_{s}, are clause sets. Further, we assume that φ_{h} is consistent, i.e., satisfiable. If $\varphi_{\text{h}} \cup \varphi_{\text{s}}$ is inconsistent, we face an over-constrained situation and want to adjust or remove constraints from φ_{s}. A *diagnosis* or *correction subset* is a set of clauses which, when removed, restores consistency:

Definition 1 (Diagnosis/Correction Subset) Let φ_{h} and φ_{s} be sets of clauses. A set $\Delta \subseteq \varphi_{\text{s}}$ is a *diagnosis* or *correction subset* if, and only if, $\varphi_{\text{h}} \cup (\varphi_{\text{s}} \setminus \Delta)$ is satisfiable.

Obviously, the empty set $\emptyset$ is a correction subset if the union $\varphi_h \cup \varphi_s$ is already consistent. Also, φ_s is always a diagnosis itself. The complement of a diagnosis is a *satisfying subset*, i.e., a subset $\Gamma \subseteq \varphi_s$ such that $\varphi_h \cup \Gamma$ is consistent.

Since we are interested in finding an optimized diagnosis we demand a diagnosis to be *minimal*. The following definition introduces a local minimality property.

Definition 2 (MSS/MCS) Let φ_h and φ_s be sets of clauses. A set $\Gamma \subseteq \varphi_s$ is a *maximal satisfiable subset* (MSS) if, and only if, $\varphi_h \cup \Gamma$ is satisfiable and for every set $\Gamma' \subseteq \varphi_s$ with $\Gamma \subsetneq \Gamma'$ it holds that $\varphi_h \cup \Gamma'$ is unsatisfiable.

A set $\Delta \subseteq \varphi_s$ is a *minimal correction subset* (MCS) if, and only if, $\varphi_h \cup (\varphi_s \setminus \Delta)$ is satisfiable and for every $\Delta' \subseteq \varphi_s$ with $\Delta' \subsetneq \Delta$ it holds that $\varphi_h \cup (\varphi_s \setminus \Delta')$ is unsatisfiable.

Clearly, the complement $\varphi_s \setminus \psi$ of an MSS (resp. an MCS) ψ of φ_h and φ_s is an MCS (resp. an MSS) of φ_h and φ_s (Liffiton and Sakallah 2008). In this work we consider the computation of a diagnosis or its complement always with respect to a clause set φ_h of hard clauses which have to be satisfied if not explicitly stated otherwise.

3.2 Preferred minimal diagnosis

In analogy to Marques-Silva and Previti (2014) we introduce the following definitions:

Definition 3 (L- and A-Preference) Let $<$ be a strict total order over a set $\varphi = \{c_1, \ldots, c_m\}$ of clauses with $c_i < c_{i+1}$ for $1 \leq i < m$, i.e., clause c_i is *preferred* to clause c_{i+1}.

We define the lexicographical order $<_{\text{lex}}$ as follows: For two sets $\psi_1, \psi_2 \subseteq \varphi$ we say set ψ_1 is lexicographically preferred to ψ_2, denoted as $\psi_1 <_{\text{lex}} \psi_2$, if, and only if,

$$\exists_{1 \leq k \leq m} : c_k \in \psi_1 \setminus \psi_2 \quad \text{and}$$
$$\psi_1 \cap \{c_1, \ldots, c_{k-1}\} = \psi_2 \cap \{c_1, \ldots, c_{k-1}\}.$$

Furthermore, we define the anti-lexicographical order $<_{\text{antilex}}$ as follows: For two sets $\psi_1, \psi_2 \subseteq \varphi$ we say set ψ_1 is anti-lexicographically preferred to ψ_2, denoted as $\psi_1 <_{\text{antilex}} \psi_2$, if, and only if,

$$\exists_{1 \leq k \leq m} : c_k \in \psi_2 \setminus \psi_1 \quad \text{and}$$
$$\psi_1 \cap \{c_{k+1}, \ldots, c_m\} = \psi_2 \cap \{c_{k+1}, \ldots, c_m\}.$$

For a strict total order $c_1 < \ldots < c_m$ we denote the inverse order $c_m < \ldots < c_1$ by $<^{-1}$. When we want to relax an over-constrained system, we want to find an MSS which is the lexicographically *most* preferred one or, the other way round, we want to find an MCS which is the anti-lexicographically *most* preferred one for the inverse order $<^{-1}$. The following definition captures this motivation:

Definition 4 (Preferred MCS/MSS) Let φ_h and $\varphi_s = \{c_1, \ldots, c_m\}$ be sets of clauses. Let $<$ be a strict total order over φ_s with $c_i < c_{i+1}$ for $1 \leq i < m$.

An MSS Γ is L-preferred (resp. A-preferred) if for all MSS $\Gamma' \neq \Gamma$ it holds that $\Gamma <_{\text{lex}} \Gamma'$ (resp. $\Gamma <_{\text{antilex}} \Gamma'$). Analogously, an MCS Δ is L-preferred (resp. A-preferred) if for all MCS $\Delta' \neq \Delta$ it holds that $\Delta <_{\text{lex}} \Delta'$ (resp. $\Delta <_{\text{antilex}} \Delta'$).

The lexicographical order appears to be the more intuitive one. Whereas an L-preferred set tries to include the most preferred clauses, an A-preferred set tries to exclude the most non-preferred clauses. We focus on the computation of the L-preferred MSS w.r.t. $<$ or, analogously, the A-preferred MCS w.r.t. $<^{-1}$ (cf. use cases in Section 4).

If ψ is an L-preferred (resp. A-preferred) MSS/MCS of φ_s w.r.t. to the order $<$, then $\varphi_s \setminus \psi$ is an A-preferred (resp. L-preferred) MSS/MCS of φ_s w.r.t. to the inverse order $<^{-1}$ (see Marques-Silva and Previti 2014, Proposition 12). Therefore, algorithms for the computation of an L-preferred MSS/MCS can also be used for the computation of the corresponding A-preferred MCS/MSS.

The A-preferred MCS is also called the *preferred minimal diagnosis* (PMD), since it represents a minimal diagnosis trying to contain the most preferred constraints of the reverse order $<^{-1}$, i.e., trying to avoid the most preferred constraints of the original order $<$.

For comparison, the definition of a *preferred minimal diagnosis* used in (Felfernig et al. 2012) is in the context of a *constraint satisfaction problem* (CSP). The set C_{KB} (resp. C_{R}) represents the constraints of the knowledge base (resp. the user requirements). In the context of Propositional Logic the set C_{KB} (resp. C_{R}) is represented by φ_h (resp. φ_s). Note that the strict total order in (Felfernig et al. 2012) is defined the other way round, i.e., if $c_i < c_j$ then constraint c_j is preferred to c_i. Our definition follows (Junker 2004; Marques-Silva and Previti 2014).

3.3 Minimum weighted diagnosis

A minimum weighted diagnosis is a diagnosis where the sum of weights of the unsatisfied clauses (the clauses to remove) is optimized to be the minimum. In the context of Propositional Logic we are considering the *partial weighted MinUNSAT* problem (Li and Manyà 2009).

Definition 5 (MinUNSAT) Let φ_h and $\varphi_s = \{c_1, \ldots, c_m\}$ be sets of clauses over variables $\mathtt{vars}(\varphi_h \cup \varphi_s) = \{x_1, \ldots, x_n\}$. Let $w_1, \ldots, w_m$ be weights corresponding to the clauses $c_1, \ldots, c_m$ with $w_i \in \mathbb{N}_{\geq 1}$. The *partial weighted minimum unsatisfiable* problem, denoted by MinUNSAT, is defined as the subset $\Delta \subseteq \varphi_s$ such that:

$$\sum_{c_i \in \Delta} w_i = \min \left\{ \sum_{i=1}^{m} w_i \cdot (1 - \beta(c_i)) \;\middle|\; \beta \models \varphi_h \right\}$$

In the context of this paper, we will focus on the partial weighted MinUNSAT problem. The dual problem, the *partial weighted maximum satisfiability* problem (MaxSAT), is analogously defined by finding the maximum sum of weights of satisfied clauses. Partial weighted MinUNSAT and partial weighted MaxSAT are closely connected:

Proposition 2 *(MinUNSAT Complement Property) Let φ_h and $\varphi_s = \{c_1, \ldots, c_m\}$ be clause sets. Let $w_1, \ldots, w_m$ be weights corresponding to the clauses $c_1, \ldots, c_m$ with $w_i \in \mathbb{N}_{\geq 1}$. Let Δ be the MinUNSAT result and Γ be the MaxSAT result, then holds:*

$$\varphi_s = \Delta \,\dot\cup\, \Gamma$$

Symbol $\dot\cup$ represents the disjoint union of two sets.

Proof For the complement $\varphi \setminus \Delta$ the following equations hold:

$$\begin{aligned}
&\sum_{i=1}^{m} w_i - \sum_{c_i \in \Delta} w_i \\
&= \sum_{i=1}^{m} w_i - \min \left\{ \sum_{i=1}^{m} w_i \cdot (1 - \beta(c_i)) \;\middle|\; \beta \models \varphi_{\mathrm{h}} \right\} \\
&= \sum_{i=1}^{m} w_i + \max \left\{ - \sum_{i=1}^{m} w_i \cdot (1 - \beta(c_i)) \;\middle|\; \beta \models \varphi_{\mathrm{h}} \right\} \\
&= \sum_{i=1}^{m} w_i + \max \left\{ - \sum_{i=1}^{m} w_i + \sum_{i=1}^{m} \beta(c_i) \;\middle|\; \beta \models \varphi_{\mathrm{h}} \right\} \\
&= \max \left\{ \sum_{i=1}^{m} \beta(c_i) \;\middle|\; \beta \models \varphi_{\mathrm{h}} \right\}
\end{aligned}$$

The last term is the property for the solution of MaxSAT. □

With Proposition 2 a solution for one problem directly leads to a solution for the other one and vice versa. Therefore, algorithms for solving the partial weighted MinUNSAT problem can also be used for solving the partial weighted MaxSAT problem.

In the context of this paper, we will refer to partial weighted MinUNSAT as MinUNSAT to simplify reading. Note that in the literature often the name MaxSAT is used to refer to MinUNSAT. But we will use its original name to make the dinstinction clear.

4 Use cases from automotive configuration

Küchlin and Sinz (2000) developed the *product overview formula* (POF) in the context of automotive configuration, which is a Boolean formula, denoted by φ_{POF}, where each model β represents a valid configuration (because it satisfies all constraints), and hence a constructible car. A POF captures the *high level configuration* of a car, that is the configurable features. In contrast, the *low level configuration* is concerned with the assembly of each individual constructible car from the multitude of alternative materials. Those are listed in the bill-of-materials (BOM), which contains all materials needed for the entire type series, i.e., for the assembly of *each* constructible car. A BOM is a structured list consisting of *positions* with alternative position *variants* (e.g., the Engine position with Engine 1, Engine 2, etc., as variants). An attached selection condition of each position variant is a Boolean formula ψ_i, such that ψ_i evaluates to true for a car represented by a model β of φ_{POF} if, and only if, the material of this position variant belongs in the-car β.

We briefly recapitulate the three main verification tests (Küchlin and Sinz 2000; Sinz et al. 2003) for the BOM:

> **Test-1: Redundant Part.** For each position and for each position variant with selection constraint ψ, check whether $\varphi_{\mathrm{POF}} \wedge \psi$ is satisfiable. If not, the part is *redundant*.

Test-2: Overlap Error (double hit). For each position, check if there are two position variants with selection constraints ψ_i and ψ_j, $i \neq j$, such that $\varphi_{\text{POF}} \wedge \psi_i \wedge \psi_j$ is satisfiable. If this is the case, the position has an *overlap error*, since there is a car configuration which selects both variants.
Test-3: Incomplete Position (no-hit). For each position, check if $\varphi_{\text{POF}} \wedge \bigwedge_{i=1}^{t} \neg\psi_i$ is satisfiable, where $\psi_1, \ldots, \psi_t$ are the selection constraints of all position variants of the considered position. If this is the case, the position is *incomplete*, i.e., there exists at least one car configuration for which no part is selected at this position.

Use cases from automotive configuration can be divided into two categories. On the one hand, we want to re-configure constraints or options to restore consistency in an optimal way when facing an over-constrained knowledge base. On the other hand, we want to have optimal example configurations for situations where the knowledge base is consistent. Both kinds of use cases can be covered by computing an optimal diagnosis, either by the computation of the preferred minimal diagnosis or computation of the minimum weighted diagnosis. For this we consider preferences (a strict total order) or priorities (e.g., weights in kg, CO_2 emission, price in Euro/Dollar, etc) on the constraints.

1. **Re-Configuration of an over-constrainted knowledge base.** An optimal diagnosis of an over-constrainted knowledge base is desired in the following use cases.
   (a) **Re-Configuration / Repair suggestion.** During an interactive configuration session (cf. Walter and Küchlin 2014) a customer may be confronted with a situation where a desired option is not selectable because it conflicts with the knowledge base and previously selected options. By calculating an optimal diagnosis we can provide the customer a repair suggestion where we try to keep the most important constraints.
   (b) **Engineering guidance for non-constructible options.** For a given set of options which are in conflict with the knowledge-base, an engineer is given the task to adjust the constraints of the knowledge-base such that the options will be constructible for the next product cycle. To guide the engineer we can compute the optimal diagnosis which tries to keep the most important constraints and shows a set of less important constraints that have to be removed or adjusted.
   (c) **Engineering guidance for redundant parts.** If a part of the BOM has become accidentally redundant, see **Test-1**, we can compute the optimal diagnosis to retrieve a set of less important constraints which have to be removed or adjusted.
   (d) **Enumeration of the best k diagnoses.** In order to provide alternative solutions we can compute the best k optimal diagnoses in descending order. The number k can be small compared to the number of all existing diagnoses, e.g., we can compute the $k = 10$ best diagnoses during a configuration process for a customer to provide a set of alternative consistent re-configurations.
   (e) **Enumeration of all diagnoses.** By the enumeration of all diagnoses we can extract all *minimal cardinality conflicts* (called *minimal unsatisfiable subset* in the context of Propositional Logic) in order to provide a precise explanation *why* the knowledge-base is over-constrained (Liffiton and Sakallah 2008).

2. **Optimal example configurations.** An example configuration can be more intuitive when it respects the importance of features. We can provide such an example configuration by the computation of an optimal diagnosis. The complement of the optimal diagnosis represents the optimal example configuration. There are many use cases, where an optimized example configuration is desired.

(a) **Optimal example configuration for a BOM position with overlap errors.** After checking each position of a BOM for overlap errors (cf. **Test-2**), we want to provide the engineer with an example of a constructible car (variable assignment) for which the overlap error occurs. We can compute the optimal diagnosis and the complement of the result is the optimal example configuration of a car with an overlap error.

(b) **Optimal example configuration covering important overlap errors of a BOM position.** It can happen that a position of the BOM contains multiple overlap errors. Let $OE \subseteq \{\{i, j\} \mid i, j = 1, \ldots, k \text{ and} i \neq j\}$ be the set of all overlap errors of a position. To provide an important overlap error to the engineer, we can consider the importance of the parts by assigning preferences or priorities to the selection conditions. From a strict total order (resp. priorities) of the selection conditions we can deduce a strict total lexicographical order on sets of selection conditions, such that we have preferences (resp. priorities) on the overlap errors in OE. Then we compute the optimal diagnosis. The complement of the result is the optimal example configuration covering the most important overlap errors.

(c) **Optimal example configuration for an incomplete BOM position.** For an incomplete BOM position (cf. **Test-3**), we can consider the importance of features to provide the engineer the optimal example configuration of a constructible car which is not covered by the incomplete position.

(d) **Optimal example configuration during the configuration process.** During the configuration process of a car, a customer provides a partial selection of options. Such a partial selection, if consistent with the knowledge base, can be extended to a constructible car. To provide the customer with important examples of cars, we can consider the importance of the features. The complement of the optimal diagnosis represents the optimal example configuration of a constructible car.

The computation of such an example also makes the configuration process deterministic in the sense that the most important constructible car is computed, while any satisfying assignment would fit for an example of a constructible car.

(e) **Enumeration of the best k diagnoses.** In order to provide alternative optimal example configuration, as described in the previous use cases, we can compute the best k diagnoses in descending order. The number k can be small compared to the number of all existing diagnoses, e.g., we can compute the $k = 10$ best diagnoses during a configuration process for a customer to provide a set of alternative optimal example configurations. For example, a re-configuration tool could provide a "next" functionality to step through the optimal example configurations.

5 Computation of the optimal diagnosis

In this section we present different methods and improvements for the computation of the optimal diagnosis. We begin with approaches for the computation of the preferred minimal diagnosis and afterwards show approaches for the computation of the minimum weighted diagnosis. Afterwards, we point out similarities of both diagnosis optimizations.

5.1 Computation of the preferred minimal diagnosis

A straight forward approach is a linear search (see the constructive definition of the preferred explanation in Junker (2004)): We iterate in descending order through all constraints and

check whether they conflict with the hard constraints and the previously added constraints or not. If there is a conflict, the constraint is part of the A-preferred MCS. Otherwise, the constraint is part of the L-preferred MSS and will be added. The complexity of linear search in terms of the number of consistency checks is $\mathcal{O}(m)$, where m is the number of clauses in φ_s.

Algorithm 1 shows the linear search procedure. Variable `Solver` holds the solver for checking the formula for satisfiability or unsatisfiability by calling `Solver.Consistent` or `Solver.Inconsistent`, respectively. In the context of Propositional Logic the solver is a SAT solver. When considering another logic, the reasoning procedure used has to be an appropriate solver, e.g., when considering CSP instances the solver is a CSP solver.

Algorithm 1 Linear Search with backbones and inc-/decremental optimization

```
   Input: φ_h, φ_s = {c_1, ..., c_m} with c_1 < ··· < c_m
   Output: Preferred minimal diagnosis Δ
 1 global : Δ ← ∅
 2 if Solver.Inconsistent(φ_h) then
 3     return "no solution"
 4 Solver.Add(φ_h)                          // Speed-Up 1: Inc-/Decremental solver
 5 for i ← 1 to m do
 6     Solver.Mark()
 7     Solver.Add(c_i)                      // Speed-Up 1: Inc-/Decremental solver
 8     if Solver.Inconsistent() then
 9         Solver.Undo()                    // Speed-Up 1: Inc-/Decremental solver
10         Solver.AddUnits(¬c_i)     // Speed-Up 2: Add negated literals of c_i
11         Δ ← Δ ∪ {c_i}       // Clause c_i is element of the A-preferred MCS
12 return Δ
```

Remark 2 1. (Exploiting inc-/decremental SAT interface) Modern SAT solvers often provide an inc-/decremental interface for adding clauses and removing them afterwards. This can be useful, e.g., when we have a huge formula, like φ_{POF}, representing the configuration model and small test instances to check against the configuration model.

We can improve the linear search by using the incremental interface and adding the consistent clauses incrementally. Thus, we do not have to add all constraints again in each iteration (see Speed-Up 1 in Algorithm 1).

2. (Exploiting backbones) The negation of identified diagnosis clauses are backbone literals of $\varphi_h \cup \Gamma$, where Γ is the resulting L-preferred MSS. Therefore, we can narrow the search space for the SAT solver by adding the negation of literals of an identified diagnosis clause as unit constraints (cf. Marques-Silva et al. 2013). The SAT solver will perform unit propagation on these literals instead of costly decisions (see Speed-Up 2 in Algorithm 1).

Felfernig et al. (2012) developed a divide-and-conquer procedure, called FastDiag, for computing the preferred minimal diagnosis of Constraint Satisfaction Problems (CSP) which has been successfully exploited in the QuickXplain algorithm (Junker 2004) for computing preferred explanations (a preferred minimal unsatisfiable subset). FastDiag can be used for the computation of the preferred minimal diagnosis in the context of Propositional Logic, too. The idea behind FastDiag is to split the set of soft clauses $\varphi_s = \{c_1, \ldots, c_m\}$ into two equally sized subsets $\psi_1 = \{c_1, \ldots, c_{\lfloor \frac{m}{2} \rfloor}\}$ and $\psi_2 = \{c_{\lfloor \frac{m}{2} \rfloor + 1}, \ldots, c_m\}$. Then check whether $\varphi_h \cup \psi_1$ is consistent. If $\varphi_h \cup \psi_1$ is consistent, then no element of the more preferred clauses of ψ_1 belongs to the result (and does not have to

be checked separately) and we know that at least one element of ψ_2 belongs to the result (one consistency check is omitted). If $\varphi_h \cup \psi_1$ is not consistent, then we recursively proceed by splitting ψ_1 into two equally sized subsets. Algorithm 2 shows the procedure adjusted to our notation.

Before FastDiag (Felfernig et al. 2012) O'Callaghan et al. (2005) developed CorrectiveExp which is very similar to FastDiag but with a subtle difference: Basically, FastDiag (Felfernig et al. 2012) first generates a preferred minimal diagnosis Δ_1 for the set of constraints ψ_1 (line 17). Next, a preferred minimal diagnosis Δ_2 for the set ψ_2 is generated taking the constraints in $\psi_1 \setminus \Delta_1$ into account (line 18). Eventually, both minimal diagnoses are combined to the final minimal diagnosis (line 19). Broadly speaking the algorithm CorrectiveExp (O'Callaghan et al. 2005) would search in the whole set $\psi \setminus \Delta_1$ for generating a preferred minimal diagnosis Δ_2, which leads to unnecessary consistency checks. Note, the performance of the system depends critically on the number of consistency checks which are calls to an NP-oracle (if arbitrary constraints are allowed). To see the difference, a simple example can be generated by a set of soft constraints where the only minimal conflict consists of the first two most preferred constraints.

Algorithm 2 FastDiag with backbones and inc-/decremental optimization

Input: $\varphi_h, \varphi_s = \{c_1, \ldots, c_m\}$ with $c_1 < \cdots < c_m$
Output: Preferred minimal diagnosis Δ

```
1  if Solver.Inconsistent(φ_h) then
2      return "no solution"
3  else
4      Solver.Add(φ_h)                    // Speed-Up 1: Inc-/Decremental solver
5      return FD(false, φ_s)

6  func FD(isRedundant, ψ = {c_1, ..., c_q}) : Preferred minimal diagnosis Δ
7  Solver.Mark()
8  Solver.Add(ψ)                          // Speed-Up 1: Inc-/Decremental solver
9  if ¬isRedundant and Solver.Consistent() then
10     return ∅
11 else
12     Solver.Undo()                      // Speed-Up 1: Inc-/Decremental solver
13 if ψ = {c_i} then
14     Solver.AddUnits(¬c_i)          // Speed-Up 2: Add negated literals of c_i
15     return ψ
16 k = ⌊q/2⌋;   ψ_1 = {c_1, ..., c_k};   ψ_2 = {c_{k+1}, ..., c_q}
17 Δ_1 = FD(IsEmpty(ψ_2), ψ_1)
18 Δ_2 = FD(IsEmpty(Δ_1), ψ_2)
19 return Δ_1 ∪ Δ_2
```

The same speed ups as seen for the linear search can be applied to FastDiag as well (see Remark 2). We can improve the FastDiag procedure by adding all constraints φ_h first and executing the consistency checks by using the inc-/decremental interface. Another improvement can be made for the help procedure FD: We add all clauses of the set $\psi = \{c_1, \ldots, c_q\}$. If they are consistent, we leave them in the solver. Therefore, once clauses are identified to be in the L-preferred MSS they will be retained for the rest of the algorithm execution. Otherwise, we remove them (see Speed-Up 1 in Algorithm 2).

The worst case complexity of FastDiag in terms of the number of consistency checks is $\mathcal{O}(2d \cdot \log_2(\frac{m}{d}) + 2d)$, where d is the minimal diagnosis set size and m is the number of clauses in φ_s. For small sized diagnoses compared to the number of overall soft clauses, the

number of consistency checks is less than the number of consistency checks of the linear search. In automotive configuration we typically face a small number of diagnosis clauses.

5.2 Computation of the minimum weighted diagnosis

Different approaches have been developed to solve the minimum weighted diagnosis problem (corresponding to the MinUNSAT problem) and we want to point out some important approaches in this section. Branch-and-Bound for general optimization problems has been adopted for MinUNSAT, e.g., Heras et al. (2012b); Kügel (2012). In recent years, many MinUNSAT solvers make use of SAT solvers as a black box. A basic linear approach is to add a blocking variable to each soft clause, iteratively checking the instance for satisfiability and restricting the blocking variables further each time to narrow the search space. Since the input length is the binary representation of the sum of weights $\log_2\left(\sum_{i=1}^{m} w_i\right)$, we have an exponential number of SAT calls in the worst case. Additionally, we can use the delivered model of the SAT solver to avoid iterations, and thus to avoid SAT calls. Furthermore, the model can be reduced to a prime implicant such that the don't care literals can be assigned arbitrarily to reach the optimum faster. The Java library SAT4J (Le Berre and Parrain 2010) uses this approach.

A binary search approach is also possible. The range of the binary search would be from 0 to the sum of weights $\sum_{i=1}^{m} w_i$. The number of SAT calls in the worst case is the input length, which is the binary representation of the sum of weights $\log_2\left(\sum_{i=1}^{m} w_i\right)$. Thus, we have a linear number of SAT calls by binary search. Even though complexity in terms of the number of SAT calls is exponentially better compared to linear search, the practical disadvantage of binary search lies in the SAT calls with an unsatisfiable result. Because industrial instances from automotive configuration often contain a huge number of models, the search for models is usually fast. In contrast, the verification of unsatisfiability takes much longer. With binary search, half the number of SAT calls result in an unsatisfiable result, whereas all but the last SAT call will find a model with linear search.

Nowadays MinUNSAT solvers make usage of SAT solvers providing an unsatisfiable core for the unsatisfiable case, which was first presented by Fu and Malik in (2006) for *unweighted* partial MinUNSAT. This approach was later extended to deal with weights in Ansótegui et al. (2009) and is known as WPM1 algorithm. The idea for the *unweighted* version is to iteratively check whether the instance is satisfiable or not. If it is satisfiable, we are finished. If it is not satisfiable, we exploit the unsatisfiable core provided by the SAT solver and relax all soft clauses by adding a fresh blocking variable. The set of satisfiable blocking variables is then restricted to 1 by a cardinality constraint (Eén and Sörensson 2006; Sinz 2005). The focus of the clauses which have to be relaxed is thereby narrowed to speed up the search process of the SAT solver. To handle weights, we have to split each soft clause c_i with weight w_i of the unsatisfiable core into two clauses: (i) a clause $c_i \vee b_i$ extended by fresh blocking variable b_i with the minimum weight $w_{\min}$ of all weights of the soft clauses in the unsatisfiable core, and (ii) a clause c_i assigned to the weight $w_i - w_{\min}$. Algorithm 2 shows the WPM1 procedure adjusted to return a diagnosis and exploiting an inc-/decremental SAT interface. The unsatisfiable core φ_c need not necessarily be an MUS. Thus, the worst case complexity of WPM1 in terms of the number of consistency checks is $\mathcal{O}(d)$, where d is the minimal sum of weights of unsatisfied clauses, i.e., only costs of 1 are added in each iteration (Heras et al. 2011). This would mean an exponential number of SAT calls compared to the input length. The exact relation between the number of iterations and the quality of the provided unsatisfiable core is an open issue (Heras et al. 2011). In practice, however, the provided unsatisfiable core tends to be minimal or with only few redundant clauses.

Another approach, not based on calling an underlying SAT solver, was evaluated by Ansótegui and Gabàs (2013) by translating a MinUNSAT instance into an ILP (Schrijver 1998) instance. Ansótegui and Gabàs evaluated the commercial Mixed Integer Programming (MIP) solver CPLEX from IBM and showed that the performance is competitive on crafted instances.

Further MinUNSAT approaches have been developed, see Morgado et al. (2013) for a good overview.

Algorithm 3 WPM1

Input: $\varphi_{\mathrm{h}}, \varphi_{\mathrm{s}} = \{c_1, \ldots, c_m\}$ with weights $w_1, \ldots, w_m \in \mathbb{N}_{\geq 1}$
Output: Minimum weighted diagnosis Δ

```
if Solver.Inconsistent(φ_h) then
    return "no solution"
Solver.Add(φ_h)
φ_pairs ← {(c_i, w_i) | i = {1, ..., m}}
while true do
    Solver.Mark()
    Solver.Add({c_i | (c_i, w_i) ∈ φ_pairs})
    (st, φ_c) ← Solver.Consistent()
    if st = SAT then
        β ← Solver.GetModel()
        return {c_i | β ⊭ c_i and (c_i, w_i) ∈ φ}
    Solver.Undo()
    BV ← ∅
    w_min ← min{w_i | c_i ∈ φ_c and (c_i, w_i) ∈ φ_pairs}
    foreach c_i ∈ φ_c ∩ {c_i | (c_i, w_i) ∈ φ_pairs} do
        b_i ← fresh blocking variable
        φ_pairs ← φ_pairs \ {(c_i, w_i)} ∪ {(c_i, w_i − w_min)} ∪ {(c_i ∨ b_i, w_min)}
        BV ← BV ∪ {b_i}
    Solver.Add(CNF(∑_{b∈BV} b ≤ 1))
```

5.3 Similarities

We want to identify and discuss similarities of the preferred minimal diagnosis and the minimum weighted diagnosis. As described in the previous subsections the problem of finding the preferred minimal diagnosis corresponds to the computation of the A-preferred MCS in Propositional Logic. The hard parts φ_{h} of both problems are equal in terms of expressive power, since both can contain arbitrary constraints which can be reduced to clauses (see Remark 1). The interesting part is the set φ_{s} with its strict total ordering $<$ and assigned weights, respectively. Both can be defined as a special case of an MCS problem. The preferred minimal diagnosis problem can be interpreted as an optimal MCS:

$$\min_{<^{-1}_{\text{antilex}}} \{\Delta \mid \Delta \text{ is an MCS w.r.t. } \varphi_{\mathrm{h}} \text{ and } \varphi_{\mathrm{s}}\}$$

The minimum weighted diagnosis (corresponding the partial weighted MinUNSAT) is also an MCS with the minimum sum of weights:

$$\min_{\leq_{\sum_{c_i \in \Delta} w_i}} \left\{ \Delta \;\middle|\; \Delta \text{ is an MCS w.r.t. } \varphi_{\mathrm{h}} \text{ and } \varphi_{\mathrm{s}} \right\}$$

The result of the A-preferred MCS problem is an optimal MCS in terms of preferences, whereas the result of the partial weighted MinUNSAT problem is an optimal MCS in

Table 1 Complement comparison

Diagnosis Δ	Complement $\varphi_s \setminus \Delta$	Reference
(Partial) MCS	(Partial) MSS	see (Liffiton and Sakallah 2008)
(Partial) A-preferred MCS w.r.t. $<$	(Partial) L-preferred MSS w.r.t. $<^{-1}$	See (Marques-Silva and Previti 2014, Prop. 12)
(Partial) (Weighted) MinUNSAT	(Partial) (Weighted) MaxSAT	See Prop. 2

terms of weights. The same similarities hold for the corresponding complement problems MaxSAT and L-preferred MSS.

For any given MCS, the complement is an MSS. This complement property holds for the A-preferred MCS problem and the MinUNSAT problem, too. Table 1 shows an overview. With "partial" in parantheses we indicate that the complementary property also holds even if no set of hard clauses φ_h is considered. For MinUNSAT the complement property holds for all combinations, with or without hard clauses and with or without weights.

Table 2 shows a comparison concerning the uniqueness of the result. All results also hold for the corresponding dual problem, i.e., the computation of the complement as seen in Table 1. We distinguish four categories: (i) Clause set: The question is whether the clauses of the result is always the same; (ii) Cardinality of the diagnosis Δ: The question is whether the number of clauses of the result are always the same; (iii) The sum of weights $\sum_{c_i \in \Delta} w_i$ of the diagnosis Δ: The question is whether the sum of weights of the result is always the same, and (iv) The model of the complement $\varphi_s \setminus \Delta$: The question is whether the model satisfying $\varphi_h \cup (\varphi_s \setminus \Delta)$ is unique.

For entry Yes[1]: If we consider the preferences of the A-preferred MCS problem as weights so that we have a correct reduction (see Section 7), then the sum of weights is unique.

6 Computational complexity of the optimized diagnosis

An established measurement of the complexity of function problems, where the output can be of an arbitrary structure and is not restricted to false and true, is the complexity in terms of the number of calls to an NP-oracle (Gottlob and Fermüller 1993; Krentel 1988). In this section, we will study the lower and upper bounds in terms of the number of calls to an NP-oracle the computation of a diagnosis (minimal correction subset), the preferred minimal diagnosis (A-preferred MCS) and minimum weighted diagnosis (partial weighted MinUNSAT).

Table 2 Result uniqueness comparison

Diagnosis Δ	Result Uniqueness			
	Clause Set	Cardinality $\lvert\Delta\rvert$	Weights $\sum_{c_i \in \Delta} w_i$	Model
(Partial) MCS	No	No	No	No
(Partial) A-preferred MCS	Yes	Yes	Yes[1]	No
(Partial) (Weighted) MinUNSAT	No	No	Yes	No

We use the standard notation for the complexity class $\mathrm{FP}^{\mathrm{NP}}$ (resp. $\mathrm{FP}^{\mathrm{NP}[\log n]}$), the class of function problems solvable in deterministic polynomial time using a polynomial (resp. logarithmic) number of calls to an NP oracle (Papadimitriou 1994). Furthermore, the complexity class $\mathrm{FP}_{||}^{\mathrm{NP}}$ is the class of function problems solvable in deterministic polynomial time using a polynomial number of non-adaptive queries to an NP oracle (see Selman 1994 for a definition). The relation between these three complexity classes is (Selman 1994, Section 1.2):

$$\mathrm{FP}^{\mathrm{NP}[\log n]} \subseteq \mathrm{FP}_{||}^{\mathrm{NP}} \subseteq \mathrm{FP}^{\mathrm{NP}}$$

It is not known if $\mathrm{FP}^{\mathrm{NP}[\log n]} = \mathrm{FP}_{||}^{\mathrm{NP}}$ or $\mathrm{FP}_{||}^{\mathrm{NP}} = \mathrm{FP}^{\mathrm{NP}}$ holds, but it is believed that neither of the two equations is the case (Selman 1994, Section 1.2).

Table 3 shows the main result of this section, i.e., a comparison of computational complexity with the result that both problems, the computation of the preferred minimal diagnosis and a minimum weighted diagnosis, are $\mathrm{FP}^{\mathrm{NP}}$-complete and therefore equally hard to solve. We will explain Table 3 in detail in the rest of this section, beginning with the computation of an MCS. Since the complement of an MSS is an MCS, it is sufficient to prove the complexity for one of the two problems. The computation of an MCS is in $\mathrm{FP}^{\mathrm{NP}}$, since an MCS can be computed by a linear search (see Marques-Silva et al. 2013, Algorithm 1). The computation of an MCS is $\mathrm{FP}_{||}^{\mathrm{NP}}$-hard as shown in (Chen and Toda 1995, Theorem 4.8(3)). Since $\mathrm{FP}^{\mathrm{NP}[\log n]} \subseteq \mathrm{FP}_{||}^{\mathrm{NP}}$ (see Jenner and Torán 1995, Theorem 2.2), the computation of an MCS is also $\mathrm{FP}^{\mathrm{NP}[\log n]}$-hard. However, we will prove that a logarithmic number of NP-oracle calls is not sufficient for the computation of an MSS and therefore, the problem is no member of $\mathrm{FP}^{\mathrm{NP}[\log n]}$ unless $\mathrm{P} = \mathrm{NP}$.

A *maximal model* of a Boolean formula is a satisfying assignment such that the set of variables assigned to true cannot be extended to another satisfying assignment. That is, the set of true assigned variables forms a local maximum.

Since the computation of a satisfying assignment for a Boolean formula cannot be solved by a logarithmic number of calls to an NP-oracle unless $\mathrm{P} = \mathrm{NP}$ (Gottlob and Fermüller 1993, Theorem 5.4.), the computation of a maximal model cannot be solved by a logarithmic number of calls to an NP-oracle unless $\mathrm{P} = \mathrm{NP}$, either.

The problem of finding a maximal model for a Boolean formula can be polynomially reduced to the problem of finding an MSS w.r.t. clause sets φ_{h} and φ_{s}. Therefore, the computation of an MSS can not be solved by a logarithmic number of calls to an NP-oracle

Table 3 Complexity comparison in terms of NP-oracle calls

Problem	Lower Bound / Hardness	Upper Bound
MCS / MSS	$\mathrm{FP}_{\|\|}^{\mathrm{NP}}$-hard, (see Chen and Toda 1995, Thm. 4.8) but $\notin \mathrm{FP}^{\mathrm{NP}[\log n]}$, see Thm. 1 unless $\mathrm{P} = \mathrm{NP}$	$\in \mathrm{FP}^{\mathrm{NP}}$ (e.g., linear search), (see Marques-Silva et al. 2013, Alg. 1)
Partial Weighted MinUNSAT / Partial Weighted MaxSAT	$\mathrm{FP}^{\mathrm{NP}}$-hard (Papadimitriou 1994, Thm. 17.4)	$\in \mathrm{FP}^{\mathrm{NP}}$ (Papadimitriou 1994, Thm. 17.4) (e.g., binary search, see (Morgado et al. 2013))
A-preferred MCS / L-preferred MSS	$\mathrm{FP}^{\mathrm{NP}}$-hard, see Corollary 1	$\in \mathrm{FP}^{\mathrm{NP}}$ (e.g., linear search, see Alg. 1)

either unless P = NP. Thus, the problem is not in the class $\text{FP}^{\text{NP}[\log n]}$ unless P = NP. The following theorem captures this statement.

Theorem 1 *Let φ_{h} and φ_{s} be clause sets. The computation of an MSS w.r.t. φ_{h} and φ_{s} cannot be solved with a logarithmic number of calls to an NP-oracle unless* P = NP.

Proof We reduce the problem of finding a maximal model of a Boolean formula ψ to the problem of computing an MSS w.r.t. clause sets φ_{h} and φ_{s}. We define:

$$\varphi_{\text{h}} := \texttt{tseitin}(\psi)$$
$$\varphi_{\text{s}} := \{\{x_i\} \mid x_i \in \texttt{vars}(\psi)\}$$

The resulting MSS induces a model by assigning to true all variables contained within the MSS. All other variables are assigned to false. The resulting model is a maximal model, since otherwise the MSS properties are violated. □

The partial weighted MinUNSAT (resp. partial weighted MaxSAT) problem is FP^{NP}-complete (Papadimitriou 1994, Theorem 17.4). Partial weighted MinUNSAT can be solved, for example, by a binary search where the lower bound is 0 and the upper bound is the sum of all weights $\sum_{i=1}^{m} w_i$. Since the input length of the problem is $\log_2\left(\sum_{i=1}^{m} w_i\right)$, the number of calls to an NP-oracle is linear. Pseudo-Boolean Constraints, encoded as Boolean formulas, can be used to narrow the search space (Li and Manyà 2009).

The computation of the A-preferred MCS can be performed with a linear number of calls to an NP-oracle, such as by linear search as in Algorithm (Marques-Silva et al. 2013, Algorithm 1), and therefore the problem is an element of the class FP^{NP}.

It was an open question stated in (Marques-Silva and Previti 2014, Remark 1) whether the computation of an A-preferred MCS is FP^{NP}-hard. We will prove that the computation of an A-preferred MCS is FP^{NP}-hard by proving that the computation of the complement set, the L-preferred MSS, is FP^{NP}-hard.

Theorem 2 *Let φ_{h} and φ_{s} be clause sets. The computation of the L-preferred MSS w.r.t. φ_{h} and φ_{s} is FP^{NP}-hard.*

Proof We consider the *Maximum Satisfying Assignment* (MSA) problem: For a Boolean formula ψ with variables $\texttt{vars}(\psi) = \{x_1, \ldots, x_n\}$ find a satisfying assignment with the lexicographical maximum of the word $x_1 \cdots x_n \in \{0,1\}^n$ or 0 if ψ is not satisfiable. The MSA problem is FP^{NP}-complete as proved in Krentel (1988). We can polynomially reduce the MSA problem to the L-preferred MSS problem. We define:

$$\varphi_{\text{h}} := \text{Tseitin}(\psi)$$
$$\varphi_{\text{s}} := \{\{x_1\}, \ldots, \{x_n\}\}$$
$$< := x_1 < \cdots < x_n$$

Since the Tseitin-Transformation has the same models on the set of the original variables $\{x_1, \ldots, x_n\}$ as formula ψ (see Proposition 1), our reduction is sound. The solution of the L-preferred MSS problem induces a solution for the MSA problem. □

Furthermore, we show that the computation of the L-preferred MSS w.r.t. to a hard clause set φ_{h} and a soft clause set φ_{s} can be polynomially reduced to the computation of the L-preferred MSS w.r.t. a soft clause set only. That means, the computation of the L-preferred MSS w.r.t. a soft clause set only is also FP^{NP}-hard.

Proposition 3 *Let φ_h and φ_s clause sets. The computation of the L-preferred MSS w.r.t. φ_h and φ_s is polynomially reducible to the computation of the L-preferred MSS w.r.t. to a soft clause set only.*

Proof Let $\varphi_h = \{b_1, \ldots, b_k\}$ and $\varphi_s = \{c_1, \ldots, c_m\}$ be clause sets with the strict total order $c_1 < \cdots < c_m$. We include the clauses of φ_h by extending the strict total order: $b_1 < \cdots < b_k < c_1 < \cdots < c_m$. With the extended strict total order we ensure that the clauses of φ_h are the most preferred clauses and therefore have to be satisfied (if satisfiable at all). If φ_h is not satisfiable, the original problem will return "no solution" and the new problem will return the L-preferred MSS where at least one of the clauses $b_1, \ldots, b_k$ is not satisfied. If φ_h is satisfiable, we can extract the result of for the original problem by removing the clauses $b_1, \ldots, b_k$ from the calculated L-preferred MSS. □

Note that Proposition 3 shows that an additional set of hard clauses does not affect the complexity of the L-preferred MSS problem. We summarize our results about the computation of the L-preferred MSS and the A-preferred MCS in the following corollary.

Corollary 1 *Let φ_h and φ_s be clause sets.*

1. *The computation of the A-preferred MCS (resp. L-preferred MSS) w.r.t. to a hard clause set φ_h and a clause set φ_s is $\mathrm{FP}^{\mathrm{NP}}$-complete.*
2. *The computation of the A-preferred MCS (resp. L-preferred MSS) w.r.t. a clause set φ_s only is $\mathrm{FP}^{\mathrm{NP}}$-complete.*

Proof 1. $\mathrm{FP}^{\mathrm{NP}}$-Hardness follows from Theorem 2. Since the complement of the L-preferred MSS w.r.t. the order $<$ is the A-preferred MCS w.r.t. the inverse order $<^{-1}$ (see Marques-Silva and Previti 2014, Proposition 12), the same complexity holds. Membership in $\mathrm{FP}^{\mathrm{NP}}$ follows by linear search, see Section 5.1.

2. $\mathrm{FP}^{\mathrm{NP}}$-Hardness follows from Theorem 2, Proposition 3 and statement 1. of this corollary. Membership of $fpnp$ follows by linear search, see Section 5.1.

□

Corollary 1 negatively answers the open question, stated in (Marques-Silva and Previti 2014, Remark 1), whether computing L-preferred MSSes and A-preferred MCSes could be in $\mathrm{FP}^{\mathrm{NP}[\log n]}$.

Unless P = NP, problem which are $\mathrm{FP}^{\mathrm{NP}}$-complete are strictly harder than problems in $\mathrm{FP}^{\mathrm{NP}[\log n]}$ (Krentel 1988). Intuitively, the computation of the A-preferred MCS is solved by checking each clause separately in the worst case. The computation of any MCS is expected to be an easier task, since no order has to be respected. But the exact lower bound is unknown to the best of our knowledge.

In summary, the computation of the preferred minimal diagnosis and the minimum weighted diagnosis are both $\mathrm{FP}^{\mathrm{NP}}$-complete and therefore equally hard to solve.

7 Reductions

We have shown in the previous section that both, the preferred minimal diagnosis problem and the minimum weighted diagnosis problem, are $\mathrm{FP}^{\mathrm{NP}}$-complete. Therefore, both problems are polynomially reducible to each other. In this section we show how these reductions can be done and point out practical issues.

7.1 Preferred minimal diagnosis reduced to minimum weighted diagnosis

Let φ_{h} and $\varphi_{\mathrm{s}} = \{c_1, \ldots, c_m\}$ be the clause sets with a strict total order s.th. $c_1 < \ldots < c_m$. Again, we assume φ_{h} to be a set of clauses (cf. Remark 1). We can reduce the preferred minimal diagnosis (A-preferred MCS) problem to the minimum weighted diagnosis (MinUNSAT) problem by building the following clause sets φ'_{h} and φ'_{s}:

$$\begin{aligned} \varphi'_{\mathrm{h}} &:= \varphi_{\mathrm{h}} \\ \varphi'_{\mathrm{s}} &:= \varphi_{\mathrm{s}} \end{aligned}$$

with weight w_i defined recursively:

$$\begin{aligned} w_m &:= 1 \\ w_i &:= \left(\sum_{j=i+1}^{m} w_j \right) + 1 \qquad \text{for } 1 \leq i < m \end{aligned}$$

With these weights assigned we achieve two important properties: (i) the descending order of the constraints, and, more importantly, (ii) the lexicographical order, because constraint c_i with weight $\left(\sum_{j=i+1}^{m} w_j\right) + 1 = w_{i+1} + \ldots + w_m + 1$ has a greater weight than the sum of weights of all following and less preferred constraints $c_{i+1}, \ldots, c_m$.

Each step requires polynomial time. However, the downside of the above reduction is the growth of the weights. It can be shown by induction that $\left(\sum_{j=i+1}^{m} w_j\right) + 1 = 2^{m-i}$. The most preferred clause has weight 2^{m-1}. Data types `int` and `long` with a typically length not longer than 64 Bit get easily exceeded because each clause requires one bit. Thus, arbitrary-precision arithmetic would be necessary which performs slower.

Useful for comparison, Argelich et al. (2009) propose an encoding of hierarchically dependent optimization problems to MaxSAT. Our encoding can be interpreted as a special case of the encoding in Argelich et al. (2009), where each optimization problem consists of one constraint only.

7.2 Minimum weighted diagnosis reduced to preferred minimal diagnosis

First, we show how we can reduce the minimum weighted diagnosis (MinUNSAT) problem easily to the preferred minimal diagnosis (A-preferred MCS) problem if the weights comply with the following property:

Proposition 4 *Let φ_{h} and $\varphi_{\mathrm{s}} = \{c_1, \ldots, c_m\}$ be clause sets and $w_1, \ldots, w_m \in \mathbb{N}_{\geq 1}$ be weights. If there exists a permutation π of the indices $\{1, \ldots, m\}$ such that:*

$$w_{\pi(i)} > \sum_{j=i+1}^{m} w_{\pi(j)}$$

then the strict total order $<_\pi$ with $c_{\pi(1)} < \cdots < c_{\pi(m)}$ with clause sets φ_{h} and φ_{s} is a reduction to the preferred minimal diagnosis problem.

Proof The reduction is correct since (i) it preserves the order of the soft clauses w.r.t. their weights, and (ii) the weights are in a relation such that for each clause $c_{\pi(i)}$ the minimum weighted diagnosis solution will try to satisfy $c_{\pi(i)}$ before satisfying all clauses $c_{\pi(i+1)}, \ldots, c_{\pi(m)}$ and so will a solution of the preferred minimal diagnosis due to the strict total clause ordering. □

We can check whether such a permutation can be found by: (i) Sorting soft clauses $c_1, \ldots, c_m$ in ascending order w.r.t. their weights for complexity $\mathcal{O}(m \log m)$, and (ii) iterating through the sorted list and checking each clause c_i whether the inequality holds for complexity $\mathcal{O}(m)$. The property of Proposition 4 is *sufficient* but *not necessary*. There are other classes of minimum weighted diagnosis instances without this property which are reducible in polynomial time, too.

Example 1 $\varphi_s = \{x_1, \ldots, x_m\}$ with weights $w_1 = m, \ldots, w_m = 1$, i.e., a descending weight for each clause. We assume $\text{atMost}(x_1, \ldots, x_m) \subseteq \varphi_h$, i.e., at most one soft clause is allowed to be true. The minimum weighted diagnosis solution will try to satisfy the clause with the highest weight. The preferred minimal diagnosis problem with the strict total ordering $x_1 < \cdots < x_m$ will be a diagnosis which contains clauses except for the most preferred one in the ordering which can be satisfied under φ_h. Since at most one of the clauses can be true, the result is the same.

An exact encoding of the minimum weighted diagnosis problem as an preferred minimal diagnosis instance can be done by building an adder network with φ'_h and φ'_s as follows:

1. Add all hard clauses φ_h to φ'_h, i.e., $\varphi'_h := \varphi_h$.
2. Encode each soft clause $c_i \in \varphi_s$ by an unit clause with a fresh variable s_i (similar to Remark 1) and add the implication $s_i \rightarrow c_i$ to the hard clauses:
$$\varphi'_h := \varphi'_h \cup \{(\neg s_i \vee c_i) \mid c_i \in \varphi_s\}$$
3. Build the binary representation of the sum of weights:
$$\sum_{i=1}^{m} w_i \cdot s_i = a_l \cdot 2^l + \cdots + a_0 \cdot 2^0$$
Encode this sum as an adder network with input variables $s_1, \ldots, s_m$ and output variables $a_l, \ldots, a_0$. The output variables are the new soft unit clauses and the strict total order is given by the order of coefficients of the binary representation from the most significant bit a_l to the least significant bit a_0. The encoding $\texttt{Add}(s_1, \ldots, s_m; a_0, \ldots, a_l)$ of the adder network is added to the hard clauses.
$$\begin{aligned} \varphi'_h &:= \varphi'_h \cup \texttt{tseitin}(\texttt{Add}(s_1, \ldots, s_m; a_0, \ldots, a_l)) \\ \varphi'_s &:= \{\{a_j\} \mid j \in \{0, \ldots, l\}\} \\ < &:= a_l < \cdots < a_0 \end{aligned}$$
The encoding of the adder network can be done with a polynomial number of clauses and auxiliary variables compared to the input length, see for example (Warners 1998).

8 Greedy approaches

We show a greedy approach of the preferred minimal diagnosis problem (resp. minimum weighted diagnosis problem) on the basis of the minimum weighted diagnosis problem (resp. preferred minimal diagnosis problem). Even though both problem can be reduced to each other without loss in theory, see Section 7, we may get better running times for the greedy approaches. Additionally, as we have described in Section 7, there are practical issues with an exact reduction. The following greedy approaches can be easily implemented. Afterwards, in Section 9, we evaluate the time and quality tradeoff of the greedy approaches.

8.1 Greedy approach for the preferred minimal diagnosis

Let φ_{h} and $\varphi_{\mathrm{s}} = \{c_1, \ldots, c_m\}$ be the clause sets with a strict total order s.t. $c_1 < \ldots < c_m$. We assume φ_{h} to be a set of clauses without loss of generality, see Remark 1. We can develop a greedy approach for the preferred minimal diagnosis problem by solving the minimum weighted diagnosis problem with the following clause sets φ'_{h} and φ'_{s}:

$$\begin{aligned} \varphi'_{\mathrm{h}} &:= \varphi_{\mathrm{h}} \\ \varphi'_{\mathrm{s}} &:= \varphi_{\mathrm{s}} \\ w_i &:= m - (i-1) \end{aligned}$$

With this weight assignment, the most preferred clause is assigned to weight m, the next one to weight $m-1$ and so on. The lexicographical order will be imitated somewhat but not sufficiently to be exact. The greater the distances of the weights of consecutive constraints c_i and c_{i+1} the better the solution quality becomes. Above we set the distance to 1. In Section 7.1 we have seen how to determine distances to reach an exact reduction.

Example 2 Consider $\varphi_{\mathrm{h}} = \emptyset$ and $\varphi_{\mathrm{s}} = \{x \vee y, \neg x, \neg y, x, z\}$ with the strict total order $x \vee y < \neg x < \neg y < x < z$. The preferred minimal diagnosis is $\Delta = \{\neg y, x\}$, but the solution of the greedy approach for the minimum weighted diagnosis problem with a weight distance of 1 leads to diagnosis $\{\neg x\}$, because $\neg x$ with weight 4 is less than clauses x and $\neg y$ with weights $2 + 3 = 5$.

8.2 Greedy approach for the minimum weighted diagnosis

Solvers for the preferred minimal diagnosis problem can be used for a greedy approach for the minimum weighted diagnosis problem. Let φ_{h} and $\varphi_{\mathrm{s}} = \{c_1, \ldots, c_m\}$ be clause sets and $w_1, \ldots, w_m \in \mathbb{N}_{\geq 1}$ be weights. Let π be a permutation of the indices $1, \ldots, m$ such that the weights are sorted, i.e., if $i < j$ then $w_{\pi(i)} < w_{\pi(j)}$. We define:

$$\begin{aligned} \varphi'_{\mathrm{h}} &:= \varphi_{\mathrm{h}} \\ \varphi'_{\mathrm{s}} &:= \varphi_{\mathrm{s}} \\ < &:= c_{\pi(1)}, \ldots, c_{\pi(m)} \end{aligned}$$

The preferred minimal diagnosis will prefer to satisfy a clause with a high weight value over satisfying multiple clauses with low weight values. Whereas the minimum weighted diagnosis solution will minimize the sum of the weights of unsatisfied clauses in total. The transformation can be done in polynomial time: We sort the clauses w.r.t. their weights, which can be done in $\mathcal{O}(m \log m)$ where m is the number of soft clauses.

Example 3 Consider $\varphi_{\mathrm{h}} = \emptyset$ and $\varphi_{\mathrm{s}} = \{x, \neg x \vee y, \neg y, \neg x\}$ with weights $w_1 = 6$, $w_2 = 5$, $w_3 = 4$ and $w_4 = 3$. The greedy approach for the minimum weighted diagnosis instance with a strict total order relies on the weights of the soft clauses: $x < \neg x \vee y < \neg y < \neg x$. The preferred minimal diagnosis is $\Delta = \{\neg y, \neg x\}$ with a cost of 7 in terms of the original weights. But the minimum weighted diagnosis solution is diagnosis $\{x\}$ with a cost of 6.

9 Experimental evaluation

We evaluate both problems, the preferred minimal diagnosis problem and the minimum weighted diagnosis problem, with real industrial datasets from the automotive domain. Next we describe the test environment, the used instances and the considered use cases in detail.

9.1 Setup and test instances

For our benchmarks we use test instances based on real automotive configuration data from three different major German car manufacturers. The configuration model of constructable cars is given as a product overview formula φ_{POF} in Propositional Logic (Küchlin and Sinz 2000; Sinz et al. 2003) (see Introduction to Section 4). A φ_{POF} is satisfiable and each satisfying assignment represents a valid configurable car. If the φ_{POF} is over-constrained (unsatisfiable), then no cars are constructable. We denote a φ_{POF} by `Mx-y`, where each `x` is a different car manufacturer and each `y` is a different type series or model type.

Table 4 shows statistics about the used φ_{POF} instance for our evaluations. Column "φ_{POF}" shows the number of constraints (arbitrary Boolean formulas) and the number of variables of each φ_{POF}. Column "$\texttt{tseitin}(\varphi_{\text{POF}})$" shows the number of clauses and the number of variables the CNF transformed formula. We used an improved Plaisted-Greenbaum transformation which tries to avoid introducing auxiliary variables if possible. The instances `M1-1` and `M-2` have the same number of in both columns because the formula φ_{POF} is already in CNF and no further transformation is needed.

For our evaluation we consider three use cases from the automotive domain:

1. **Re-Configuration of User Requirements:** The hard constraints consist of the product overview formula φ_{POF}. We randomly choose 50 % of the features $\texttt{vars}(\varphi_{\text{POF}})$ which represent the user requirements (soft constraints). Since not all user requirements are consistent w.r.t. the φ_{POF} in general, such a random selection easily gets inconsistent. The goal is to optimize the user requirements (cf. Section 4, Use Case 1(a) and Walter and Küchlin (2014)).
2. **Re-Configuration of Constraints:** The constraints of the φ_{POF} are considered as soft constraints. We randomly choose 50 % of the features $\texttt{vars}(\varphi_{\text{POF}})$ which represent the user requirements (hard constraints). Similarly to the previous use case, such a random user selection easily leads to inconsistency. But in contrast to the previous use case, we want to optimize the φ_{POF} constraints instead of the user requirements. By this use case we try to realistically imitate, for example, an engineering situation where new hard requirements are given and the corresponding engineer wants to be guided by an optimized repair suggestion to adjust the rules such that a non-constructible car becomes constructible (cf. Section 4, Use Case 1(b)).
3. **Computation of the Optimal Example Configuration** The hard constraints consist of the product overview formula φ_{POF}. We randomly create 10,000 clauses of length 2 (soft constraints). By this use case we try to imitate overlap errors of a BOM position

Table 4 Statistics about the used POFs

	φ_{POF}		$\texttt{tseitin}(\varphi_{\text{POF}})$	
Instance	Constraints	Variables	Clauses	Variables
`M1-1`	11 593	996	11 593	996
`M1-2`	4 274	612	4 274	612
`M2-1`	495	483	5 557	488
`M3-1`	2 750	2 352	81 267	5 054
`M3-2`	864	607	30 885	4 447
`M3-3`	2 485	2 245	83 979	6 187

and want to find an optimal example configuration covering the most important overlap errors (cf. Section 4, Use Case 2(b)). Also, we wanted to create more difficult instances by increasing the number of soft constraints.

For each of the three use cases we evaluated four scenarios:

1. **Preferred Minimal Diagnosis**: The order of the soft constraints is chosen by random.
2. **Minimum Weighted Diagnosis**: The weights are chosen between 1 and 10 by random.
3. **Greedy Approach of the Preferred Minimal Diagnosis**: Greedy approach for the preferred minimal diagnosis as described in Section 8.1.
4. **Greedy Approach of the Minimum Weighted Diagnosis**: Greedy approach for the minimum weighted diagnosis as described in Section 8.2.

For each scenario we created 5 instances and calculated the average time in order to get a reasonably distribution. We set a timeout of 600 seconds for each instance.

We used different solver settings for the computation of the preferred minimal diagnosis:

– **Linear Search**: Linear search (**LS**), Algorithm 1, with backbone improvement (**LSB**) plus the usage of the in-/decremental SAT-Solving interface (**LSBOpt**), see Remark 2.
– **FastDiag**: Basic FastDiag (**FD**), Algorithm 2, with backbone improvement (**FDB**) plus the usage of the in-/decremental SAT-Solving interface (**FDBOpt**), see Remark 2.

For a reasonable comparison between the computation of the preferred minimal diagnosis and the minimum weighted diagnosis we implemented the linear search and the FastDiag algorithms. We implemented the linear search and the FastDiag algorithms on top of the public available SAT solver MiniSAT 2.2 (Eén and Sörensson 2004)[1] in order to make the test setup similar to the minimum weighted diagnosis solvers which are based on MiniSAT 2.2. or PicoSAT (Biere 2008),[2] except for the CPLEX-based approach. The inc-/decremental interface is implemented by adding an additional fresh blocking variable to the clauses which is used to block the clause after a decremental operation.

On the MinUNSAT side we used the following external solvers for our evaluation:

– **OpenWBO**: The open source framework OpenWBO of Martins et al. (2014) includes a whole set of different MinUNSAT solvers, all based on top of the MiniSAT-like solver (Eén and Sörensson 2006). We used the version 1.3.0 release (January 2, 2015)[3] and MiniSAT 2.2 (Eén and Sörensson 2004) as underlying SAT solver for a reasonable comparison of our implementations for the preferred minimal diagnosis computation.
– **Eva500a**: Eva500a[4] (Narodytska and Bacchus 2014) is a MinUNSAT solver based on iteratively calling an underlying SAT solver and exploiting the unsatisfiable core if the formula is unsatisfiable. But instead of restricting the blocking variables as done in the WPM1 algorithm (Ansótegui et al. 2009) a compact version of MinUNSAT resolution is used. This approach was the best overall solver in the industrial category of the MaxSAT competition in 2014[5] and is based on Glucose (Audemard et al. 2013), a variant of MiniSAT (Eén and Sörensson 2004).

[1] MiniSAT homepage: www.minisat.se

[2] PicoSAT homepage: http://fmv.jku.at/picosat

[3] OpenWBO is available at: http://sat.inesc-id.pt/open-wbo

[4] Eva500a is available at: http://www.maxsat.udl.cat/14/solvers/

[5] MaxSAT Competition 2014: http://www.maxsat.udl.cat/14/results/index.html

Table 5 Results for the preferred minimal diagnosis for use case "Re-Configuration of User Requirements" (in seconds)

	M1-1	M1-2	M2-1	M3-1	M3-2	M3-3
$\frac{\lvert\Delta\rvert}{\lvert\varphi_S\rvert} \times 100$	16	1	63	90	70	72
LS	0.23	0.02	0.11	9.38	1.06	7.56
LSOpt	0.05	0.02	0.01	0.19	0.06	0.30
LSBOpt	0.05	0.02	0.01	0.15	0.06	0.27
FD	0.87	0.04	0.27	18.51	2.22	16.62
FDOpt	0.03	0.01	0.01	0.13	0.05	0.25
FDBOpt	0.03	0.01	0.01	0.10	0.05	0.24
LS #S	420	302	89	116	90	311
LS #U	79	5	153	1,061	214	812
FD #S	162	27	70	109	79	264
FD #U	238	19	339	2,165	461	1,739

– **msuncore**:.[6] An unsatisfiable core-guided approach with iterative SAT calls using a reduced number of blocking variables. The solver suite includes different solver versions and is based on PICOSAT (Biere 2008). We tested releases 1.0, 1.1 and 1.2.
– **CPLEX**: Translation of a MinUNSAT instance to an ILP instance and using the commercial MIP solver CPLEX[7] from IBM (cf. Ansótegui and Gabàs (2013)).

We also tested other MinUNSAT solvers but first pre-evaluations have shown that the solver list above were the most competitive one for our benchmark data.

Our experiments were run on the following settings: Intel(R) Core(TM) i3-2100 CPU with 3.1GHz and 2 GB main memory running 64-bit Linux Ubuntu 12.04.5.

9.2 Evaluation: preferred minimal diagnosis

Tables 5, 6 and 7 show the results for the computation of the preferred minimal diagnosis for each use case, respectively. The first row shows the percentage of the preferred minimal diagnosis compared to the number of soft clauses. The rows "LS", "LSOpt" and "LSBOpt" show the average running times in seconds for the linear search solver settings. The rows "FD", "FDBOpt" and "FDOpt" show the average running times in seconds for the FASTDIAG solver settings. The fastest running times are highlighted for each instance. The rows "LS #S" and "LS #U" (resp. "FD #S" and "FD #U") show the number of satisfiable and unsatisfiable SAT calls for the linear search solvers (resp. FASTDIAG solvers).

All three tables show that the size of the preferred minimal diagnosis is quite small for the first car manufacturer and quite high for the second and third car manufacturer. The only exception is POF M3-1 in Table 7 where the percentage is also small with 8 %. Consequentially, the number of unsatisfiable SAT calls is very high for the second and third car manufacturer for both, the linear search and the FASTDIAG algorithm.

[6] msuncore is available at: http://logos.ucd.ie/web/doku.php?id=msuncore

[7] CPLEX is available at: http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/

Table 6 Results for the preferred minimal diagnosis for use case "Re-Configuration of Constraints" (in seconds)

	M1-1	M1-2	M2-1	M3-1	M3-2	M3-3
$\frac{\lvert\Delta\rvert}{\lvert\varphi_s\rvert} \times 100$	1	1	22	39	29	33
LS	7.04	1.20	0.15	11.72	1.50	9.21
LSOpt	6.31	1.07	0.03	0.80	0.21	0.84
LSBOpt	6.42	1.07	0.03	0.80	0.21	0.83
FD	4.05	0.10	0.49	34.47	4.51	28.68
FDOpt	0.69	0.02	0.02	0.53	0.13	0.50
FDBOpt	0.66	0.02	0.02	0.52	0.14	0.49
LS #S	11,500	4,458	383	1,667	608	1,668
LS #U	128	8	113	1,084	257	818
FD #S	733	72	181	1,096	340	987
FD #U	596	52	310	2,601	656	2,055

The use case "Re-Configuration of User Requirements" turns out to be the easiest one and can be solved nearly equally fast by all solver settings within one second. Two exceptions arise for car manufacturer three and POFs `M3-1` and `M3-3`, where the usage the inc-/decremental SAT solver interface has huge impact on the performance with running times up to 100 times faster. The usage of backbones leads to nearly the same running times.

The results use cases "Re-Configuration of Constraints" and "Computation of the Optimal Example Configuration" also show that the usage of the inc-/decremental SAT solver leads to running times up to 100 faster. Even though the linear search and the FASTDIAG approach are equally fast on some instance it turns out that in total the FASTDIAG algorithm with backbones and the usage of the inc-/decremental SAT solver interface is dominating all other solvers on all instances for all three use cases.

Table 7 Results for the preferred minimal diagnosis for use case "Computation of the Optimal Example Configuration" (in seconds)

	M1-1	M1-2	M2-1	M3-1	M3-2	M3-3
$\frac{\lvert\Delta\rvert}{\lvert\varphi_s\rvert} \times 100$	2	1	45	8	54	47
LS	5.87	5.29	8.80	100.30	33.65	76.19
LSOpt	5.75	5.00	3.62	3.83	5.48	7.68
LSBOpt	5.66	4.99	3.04	1.91	3.92	5.46
FD	6.87	0.19	22.26	190.69	79.09	172.77
FDOpt	1.02	0.04	4.38	3.28	6.21	8.14
FDBOpt	0.92	0.04	2.48	1.29	3.07	4.44
LS #S	9,799	9,995	5,454	1,886	4,504	5,244
LS #U	202	6	4,547	8,115	5,497	4,757
FD #S	973	55	3,829	1,724	3,470	3,720
FD #U	841	35	10,672	16,940	12,399	11,041

Table 8 Results for MinUNSAT for use case "Re-Configuration of User Requirements" (in seconds)

	M1-1	M1-2	M2-1	M3-1	M3-2	M3-3
$\frac{\text{optimum}}{\sum_{i=1}^{m} w_i} \times 100$	11	2	57	86	67	62
OpenWBO (default)	0.20	0.01	2.20	t/o	t/o	t/o
OpenWBO (LS-SU)	16.63	0.02	t/o	t/o	t/o	t/o
OpenWBO (WMSU3-iterative)	13.66	0.01	t/o	t/o	t/o	t/o
msuncore1.0	4.92	0.01	t/o	t/o	t/o	t/o
msuncore1.1	0.80	0.01	t/o	t/o	t/o	t/o
msuncore1.2	t/o	0.01	t/o	t/o	t/o	t/o
Eva500a	0.53	0.08	0.64	16.16	4.58	14.58
CPLEX	0.72	0.02	0.06	5.29	7.05	4.30

9.3 Evaluation: Minimum weighted diagnosis

Tables 8, 9 and 10 show the results for the computation of the minimum weighted diagnosis for each use case, respectively. The first row shows the percentage of the sum of weights of the minimum weighted diagnosis (optimum) compared to the total sum oft weights of the soft clauses ($\sum_{i=1}^{m} w_i$).

For use case "Re-Configuration of User Requirements", Table 8, solvers Eva500a and CPLEX are the only solvers who were able to solve all instances within the timeout. CPLEX performs better than Eva500a, except for POF M3-2. The instances M1-1 and M1-2 of the first car manufacturer could be solved by all solvers, except msuncore1.2 for POF M1-1.

For use case "Re-Configuration of Constraints", Table 9, all solvers were able to solve all instances within the given timeout, except OpenWBO (LS-SU) for POF M1-1. CPLEX has the best overall running times, except for instance M1-2, where the running time is only slightly slower by 0.1 seconds.

Only CPLEX solved all instances within the timeout for use case "Computation of the Optimal Example Configuration", Table 10. POF M1-2 could be solved by almost all other solvers faster than CPLEX.

Table 9 Results for MinUNSAT for use case "Re-Configuration of Constraints" (in seconds)

	M1-1	M1-2	M2-1	M3-1	M3-2	M3-3
$\frac{\text{optimum}}{\sum_{i=1}^{m} w_i} \times 100$	1	1	22	39	29	33
OpenWBO (default)	0.30	0.02	0.14	13.33	1.46	10.09
OpenWBO (LS-SU)	t/o	0.41	0.41	30.95	1.40	4.34
OpenWBO (WMSU3-iterative)	25.36	0.01	0.46	106.77	2.67	41.28
msuncore1.0	0.97	0.01	0.22	26.77	87.29	21.68
msuncore1.1	1.01	0.01	0.20	26.66	3.92	21.72
msuncore1.2	1.01	0.01	0.22	26.81	3.72	21.83
Eva500a	0.80	0.09	0.12	0.41	0.42	0.34
CPLEX	0.06	0.02	0.03	0.19	0.24	0.14

Table 10 Results for MinUNSAT for use case "Computation of the Optimal Example Configuration" (in seconds)

	M1-1	M1-2	M2-1	M3-1	M3-2	M3-3
$\frac{\text{optimum}}{\sum_{i=1}^{m} w_i} \times 100$	1	1	40	77	52	42
OpenWBO (default)	t/o	0.04	t/o	t/o	t/o	t/o
OpenWBO (LS-SU)	t/o	1.75	t/o	t/o	t/o	t/o
OpenWBO (WMSU3-iterative)	t/o	0.01	t/o	t/o	t/o	t/o
msuncore1.0	t/o	0.04	t/o	t/o	t/o	t/o
msuncore1.1	t/o	0.04	t/o	t/o	t/o	t/o
msuncore1.2	t/o	0.04	t/o	t/o	t/o	t/o
Eva500a	3.26	0.16	t/o	t/o	t/o	t/o
CPLEX	3.23	0.27	1.13	9.32	38.45	9.45

For all three use cases only CPLEX solved all instances within the timeout, which makes this solver very robust. Also, the running times are most often among the fastest ones. This comes as a bit of a surprise since we expected that CDCL-based MinUNSAT solvers would perform better on Boolean instances. One explanation may be that automotive configuration instances are small compared to other instances, e.g., instances from the MaxSAT evaluations.[8] Thus, the full potential of CDCL-based MinuNSAT solvers may not be exhausted.

9.4 Evaluation: greedy approach of the preferred minimal diagnosis

Tables 11, 12 and 13 show the results for the greedy approach for the minimum weighted diagnosis for each use case, respectively. We measure the greedy approach quality by minimality and accuracy. The first row "$\texttt{minimiality}(\Delta)$" shows the minimality which is the size of constraints contained in the diagnosis compared to the size of the preferred minimal diagnosis. The greedy based diagnosis does not have to be a superset of the preferred minimal diagnosis. Thus, a minimality < 1 can occur.

$$\texttt{minimality}(\Delta) := \frac{|\Delta_{\text{min}}|}{|\Delta|}$$

The second row "$\texttt{accuracy}(\Delta)$" shows the accuracy in terms of how much of the constraints are shared between the greedy based diagnosis and the preferred minimal diagnosis. The higher the accuracy the more constraints from the preferred minimal diagnosis are contained within the greedy based diagnosis.

$$\texttt{accuracy}(\Delta) := \frac{|\Delta \cap \Delta_{\text{min}}|}{|\Delta_{\text{min}}|}$$

For all results in this category, Tables 11, 12 and 13, we calculated the $\texttt{minimality}(\Delta)$ and $\texttt{accuracy}(\Delta)$ only for CPLEX. Similar to Section 9.3, only CPLEX solved all instances for each use case within the timeout limit. The minimality and accuracy result depends on the greedy based model of the MinUNSAT solver.

[8] MaxSAT Evaluations: http://www.maxsat.udl.cat

Table 11 Results for greedy based preferred minimal diagnosis for use case "Re-Configuration of User Requirements" (in seconds)

	M1-1	M1-2	M2-1	M3-1	M3-2	M3-3
minimality(Δ)	1.42	0.95	1.04	1.04	1.02	1.14
accuracy(Δ)	0.56	0.56	0.93	0.95	0.93	0.87
OpenWBO (default)	t/o	0.01	t/o	t/o	t/o	t/o
OpenWBO (LS-SU)	t/o	0.35	t/o	t/o	t/o	t/o
OpenWBO (WMSU3-iterative)	t/o	0.52	t/o	t/o	t/o	t/o
msuncore1.0	t/o	0.02	t/o	t/o	t/o	t/o
msuncore1.1	t/o	0.02	t/o	t/o	t/o	t/o
msuncore1.2	t/o	0.02	t/o	t/o	t/o	t/o
Eva500a	t/o	0.10	27.68	t/o	t/o	t/o
CPLEX	0.75	0.02	0.06	5.29	6.72	4.05

For use case "Re-Configuration of User Requirements", Table 11, all solvers exceeded the timeout except for CPLEX. The size of the greedy based preferred minimal diagnosis computed by CPLEX is most often smaller than the size of the preferred minimal diagnosis, resulting in a minimality(Δ) value greater than 1. The accuracy is quite good for the second and third car manufacturers.

For use case "Re-Configuration of Constraints", Table 12, almost every solver could solve each instance. CPLEX has the best running times. The minimality and accuracy is quite good and almost equal to 1 in the most cases.

Only CPLEX could solve all instances for use case "Computation of the Optimal Example Configuration", Table 13. The minimality and accuracy is quite good for the third car manufacturer and a bit worse for hte first and second car manufacturer.

9.5 Evaluation: greedy approach of the minimum weighted diagnosis

Tables 14, 15 and 16 show the results for the computation of the greedy based minimum weighted diagnosis for each use case, respectively. The first row shows the quality by

Table 12 Results for greedy based preferred minimal diagnosis for use case "Re-Configuration of Constraints" (in seconds)

	M1-1	M1-2	M2-1	M3-1	M3-2	M3-3
minimality(Δ)	1.09	0.92	1.02	1.00	1.01	1.00
accuracy(Δ)	0.83	0.91	0.96	1.00	0.97	1.00
OpenWBO (default)	7.76	0.90	0.15	13.47	t/o	10.19
OpenWBO (LS-SU)	t/o	t/o	17.40	t/o	t/o	t/o
OpenWBO (WMSU3-iterative)	t/o	31.96	t/o	t/o	t/o	t/o
msuncore1.0	1.11	0.01	0.22	26.88	t/o	21.83
msuncore1.1	1.09	0.01	0.21	26.95	t/o	21.43
msuncore1.2	1.09	0.01	0.21	27.05	t/o	21.93
Eva500a	t/o	0.59	0.18	0.84	0.96	0.75
CPLEX	0.07	0.02	0.04	0.17	0.28	0.14

Table 13 Results for greedy based preferred minimal diagnosis for use case "Computation of the Optimal Example Configuration" (in seconds)

	M1-1	M1-2	M2-1	M3-1	M3-2	M3-3
minimality(Δ)	1.84	1.24	1.11	1.05	1.05	1.13
accuracy(Δ)	0.33	0.48	0.75	0.91	0.80	0.84
OpenWBO (default)	t/o	5.19	t/o	t/o	t/o	t/o
OpenWBO (LS-SU)	t/o	t/o	t/o	t/o	t/o	t/o
OpenWBO (WMSU3-iterative)	t/o	t/o	t/o	t/o	t/o	t/o
msuncore1.0	t/o	0.04	t/o	t/o	t/o	t/o
msuncore1.1	t/o	0.04	t/o	t/o	t/o	t/o
msuncore1.2	t/o	0.04	t/o	t/o	t/o	t/o
Eva500a	t/o	15.21	t/o	t/o	t/o	t/o
CPLEX	3.14	0.27	1.05	9.54	44.41	9.67

percentage p which calculated as follows:

$$p = \frac{(\sum_{i=1}^{m} w_i) - \text{greedyOpt}}{(\sum_{i=1}^{m} w_i) - \text{exactOpt}} \times 100$$

Where greedyOpt is the sum of weights of the result of the preferred minimal diagnosis solver and exactOpt is the optimum of the minimum weighted diagnosis solver.

In terms of runnings times the results of the greedy based minimum weighted diagnosis are quite the same as the results of the preferred minimal diagnosis computation in Section 9.2. FASTDIAG improved by backbones and the inc-/decremental SAT solver interface dominates the evaluation on all instances and all use cases, except for a few instances where the running time is just slightly worse. Without the usage of the inc-/decremental SAT solver interface the running times are up to 100 times slower for both, linear search and FASTDIAG.

The greedy approach qualities are quite good of the first and second car manufacturer with percentages beginning from 89 % up to 99 % (except for one instance with 83 %). Whereas the greedy approach qualities of the third car manufacturer is worse with percentages beginning from 78 % up to 99 %. For the use case "Re-Configuration of Constraints" the greedy approach quality is very good with 99 % for each POF.

Table 14 Results for greedy based MinUNSAT for use case "Re-Configuration of User Requirements" (in seconds)

	M1-1	M1-2	M2-1	M3-1	M3-2	M3-3
p (%)	89	97	94	88	95	82
LS	0.29	0.02	0.11	9.00	1.03	7.33
LSOpt	0.05	0.02	0.01	0.19	0.06	0.30
LSBOpt	0.04	0.02	0.01	0.14	0.06	0.27
FD	1.03	0.07	0.26	17.79	2.13	15.98
FDOpt	0.03	0.01	0.01	0.12	0.06	0.25
FDBOpt	0.03	0.01	0.01	0.10	0.06	0.24

Table 15 Results for greedy based MinUNSAT for use case "Re-Configuration of Constraints" (in seconds)

	M1-1	M1-2	M2-1	M3-1	M3-2	M3-3
p (%)	99	99	99	99	99	99
LS	7.10	1.04	0.14	11.80	1.48	9.27
LSOpt	6.27	0.95	0.03	0.81	0.22	0.83
LSBOpt	6.20	0.96	0.03	0.80	0.22	0.83
FD	3.84	0.08	0.45	34.78	4.48	29.03
FDOpt	0.62	0.02	0.02	0.54	0.13	0.51
FDBOpt	0.62	0.02	0.02	0.53	0.13	0.50

10 Related work

Benavides et al. (2010) give a good survey about automated analysis of configuration models based on feature models. The authors show how the encoding of feature models to Propositional Logic and CSP can be used for automated verification, optimization with subject to an objective function, explanations of conflicts and corrective explanations (diagnoses).

Marques-Silva et al. (2013) improve the computation of an MCS by newly introduced techniques, i.e., usage of backbone literals, disjoint unsatisfiable cores and satisfied clauses. Not all techniques can be applied to the computation of an A-preferred MCS, i.e., only the usage of backbone literals can be adopted. Hence the proposed enhanced version of FastDiag (Felfernig et al. 2012; O'Callaghan et al. 2005) of Marques-Silva et al. (2013) cannot be adopted for A-preferred MCS computation. The newly proposed MCS algorithm, called ClauseD, exploits the fact that a falsified clause does not contain complementary literals, but it cannot be adopted either.

FastDiag (Felfernig et al. 2012; O'Callaghan et al. 2005) is based on the idea of divide-and-conquer which has been successfully exploited in the QuickXplain algorithm (Junker 2004). Whereas QuickXplain computes the preferred explanation (a minimal unsatisfiable subset (MUS) in the context of Propositional Logic), the FastDiag algorithm computes the preferred diagnosis (an MCS in the context of Propositional Logic), which can be interpreted as the *inverse* of QuickXplain.

Table 16 Results for greedy based MinUNSAT for use case "Computation of the Optimal Example Configuration" (in seconds)

	M1-1	M1-2	M2-1	M3-1	M3-2	M3-3
p (%)	99	99	94	80	94	78
LS	6.03	5.37	8.89	100.29	33.19	85.76
LSOpt	5.82	5.05	3.75	4.01	5.75	6.99
LSBOpt	5.65	5.12	3.15	2.24	4.07	4.83
FD	8.29	0.16	22.33	186.27	79.01	179.96
FDOpt	1.10	0.03	4.43	3.11	6.39	7.61
FDBOpt	0.98	0.04	2.56	1.24	3.16	3.95

The computational complexity in terms of the number of calls to an NP-oracle for computing preferred MCSes and MSSes sets is studied by Marques-Silva and Previti in (2014), where they also consider the computation of preferred conflicts and give an overview of algorithms and techniques which can or cannot be adopted to involve preferences.

Mencía and Marques-Silva (2014) introduce improvements on computing an MCS in the context of CSP, i.e., they adopt the CLAUSED algorithm from Marques-Silva et al. (2013). The adopted variant is also not applicable for the computation of the A-preferred MCS.

11 Conclusion and future work

In this work, we compared the problem of finding the minimal preferred diagnosis (A-preferred MCS in the context of Propositional Logic) with the problem of finding the minimum weighted diagnosis (partial weighted MinUNSAT in the context of Propositional Logic). We presented detailed practical use cases from automotive configuration.

We proved the FP^{NP}-hardness of the A-preferred MCS problem, which was an open question before. By this result we pointed out that both problems are FP^{NP}-complete and therefore are equally hard to solve in terms of the number of NP-oracle calls.

We evaluated the performance of both problems with benchmarks based on real automotive configuration data. The experimental evaluations have shown that the computation of the preferred minimal diagnosis is more robust, since all solvers could solve every instance. Linear search and FASTDIAG could substantially be improved by using the inc-/decremental SAT solver interface. In contrast, all solvers for the computation of the minimum weighted diagnosis could not solve every instance. The only exception was the CPLEX-based approach which remained robust by solving all instances of every use case reasonable fast. To fully understand this observation a detailed analysis of both solver concepts is required.

There are several more research directions for future work. To the best of our knowledge the exact lower bound for the computation of a minimal diagnosis (minimal correction subset) in the context of Propositional Logic is still an open issue.

FASTDIAG and partial weighted MinUNSAT solver in their original form compute only a single preferred minimal diagnosis (resp. minimum weighted diagnosis). Algorithms for both problems can be extended to compute the set of all diagnoses. It would be interesting to compare the enumeration of all diagnoses for both approaches and evaluate them on real instances from industrial applications like the automotive domain.

Another interesting evaluation could be the reduction of the partial weighted MinUNSAT problem to the A-preferred MCS problem by using an adder-network encoding as described in Section 7.2. With such an encoding, we could use every A-preferred MCS solver, like FASTDIAG, to solve partial weighted MinUNSAT problem instances.

References

Ansótegui, C., Bonet, M.L., & Levy, J. (2009). Solving (weighted) partial MaxSAT through satisfiability testing, In Kullmann, O. (Ed.) *SAT 2009, LNCS, vol. 5584, pp. 427–440. Springer Berlin Heidelberg*.

Ansótegui, C., & Gabàs, J. (2013). Solving (weighted) partial MaxSAT with ILP, In Gomes, C.P., & Sellmann, M. (Eds.) *CPAIOR 2013, LNCS, vol. 7874, pp. 403–409. Springer*.

Argelich, J., Lynce, I., & Marques-Silva, J. (2009). On solving boolean multilevel optimization problems, In Boutilier, C. (Ed.) *IJCAI 2009, pp. 393–398*.

Argelich, J., & Manyà, F. (2006). Exact Max-SAT solvers for over-constrained problems. *J Heuristics*, *12*(4–5), 375–392.

Audemard, G., Lagniez, J., & Simon, L. (2013). Improving glucose for incremental SAT solving with assumptions: Application to MUS extraction, In Järvisalo, M., & Gelder, A.V. (Eds.) *SAT 2013, LNCS, vol. 7962, pp. 309–317. Springer*.

Benavides, D., Segura, S., & Ruiz-Cortés, A. (2010). Automated analysis of feature models 20 years later: A literature review. *Information Systems*, *35*(6), 615–636.

Biere, A. (2008). PicoSAT essentials. *JSAT*, *4*(2–4), 75–97.

Chen, Z., & Toda, S. (1995). The complexity of selecting maximal solutions. *Inform. Comput*, *119*(2), 231–239.

Cook, S.A. (1971). The complexity of theorem-proving procedures, In Harrison, M.A., Banerji, R.B., & Ullman, J.D. (Eds.) *STOC, pp. 151–158. ACM*.

Eén, N., & Sörensson, N. (2004). An extensible SAT-solver, In Giunchiglia, E., & Tacchella, A. (Eds.) *SAT 2003, LNCS, vol. 2919, pp. 502–518. Springer Berlin Heidelberg*.

Eén, N., & Sörensson, N. (2006). Translating pseudo-boolean constraints into SAT. *JSAT*, *2*, 1–26.

Felfernig, A., Schubert, M., & Zehentner, C. (2012). An efficient diagnosis algorithm for inconsistent constraint sets. *AIEDAM*, *26*(1), 53–62.

Franco, J., & Martin, J. (2009). A history of satisfiability, In Biere, A., Heule, M., van Maaren, H., & Walsh, T. (Eds.) *Handproceedings of Satisfiability, FAIA, vol. 185, chap. 1, pp. 3–74. IOS Press*.

Fu, Z., & Malik, S. (2006). On solving the partial MAX-SAT problem, In Biere, A., & Gomes, C.P. (Eds.) *SAT 2006, LNCS, vol. 4121, pp. 252–265. Springer*.

Gottlob, G., & Fermüller, C.G. (1993). Removing redundancy from a clause. *Artif. Intell*, *61*(2), 263–289.

Heras, F., Morgado, A., & Marques-Silva, J. (2011). Core-guided binary search algorithms for maximum satisfiability, In Burgard, W., & Roth, D. (Eds.) *AAAI, pp. 36–41. AAAI Press*.

Heras, F., Morgado, A., & Marques-Silva, J. (2012). An empirical study of encodings for group MaxSAT, In Kosseim, L., & Inkpen, D. (Eds.) *Canadian Conf. on AI, LNCS, vol. 7310, pp. 85–96. Springer*.

Heras, F., Morgado, A., & Marques-Silva, J. (2012). Lower bounds and upper bounds for MaxSAT, In Hamadi, Y., & Schoenauer, M. (Eds.) *LION 6, LNCS, vol. 7219, pp. 402–407. Springer Berlin Heidelberg*.

Jenner, B., & Torán, J. (1995). Computing functions with parallel queries to NP. *Theoretical Computer Science*, *141*(1–2), 175–193.

Junker, U. (2004). QUICKXPLAIN: Preferred explanations and relaxations for over-constrained problems. In *AAAI, pp. 167–172. AAAI Press / The MIT Press*.

Krentel, M.W. (1988). The complexity of optimization problems. *J. Comput. System Sci*, *36*(3), 490–509.

Küchlin, W., & Sinz, C. (2000). Proving consistency assertions for automotive product data management. *J. Automat. Reason*, *24*(1–2), 145–163.

Kügel, A. (2012). Improved exact solver for the weighted MAX-SAT problem, In Berre, D.L. (Ed.) *POS-10. Pragmatics of SAT, EasyChair Proceedings in Computing, vol. 8, pp. 15–27. EasyChair*.

Le Berre, D., & Parrain, A. (2010). The Sat4j library, release 2.2. *JSAT*, *7*(2–3), 59–6.

Li, C.M., & Manyà, F. (2009). MaxSAT, hard and soft constraints, In Biere, A., Heule, M., van Maaren, H., & Walsh, T. (Eds.) *Handproceedings of Satisfiability, FAIA, vol. 185, chap. 19, pp. 613–631. IOS Press*.

Liffiton, M.H., & Sakallah, K.A. (2008). Algorithms for computing minimal unsatisfiable subsets of constraints. *J. Autom. Reasoning*, *40*(1), 1–33.

Marques-Silva, J., Heras, F., Janota, M., Previti, A., & Belov, A. (2013). On computing minimal correction subsets, In Rossi, F. (Ed.) *IJCAI, pp. 615–622, IJCAI/AAAI*.

Marques-Silva, J., & Previti, A. (2014). On computing preferred MUSes and MCSes, In Sinz, C., & Egly, U. (Eds.) *SAT 2014, LNCS, vol. 8561, pp. 58–74. Springer*.

Martins, R., Manquinho, V.M., & Lynce, I. (2014). Open-WBO: A modular MaxSAT solver, In Sinz, C., & Egly, U. (Eds.) *SAT 2014, LNCS, vol. 8561, pp. 438–445. Springer Int. Publishing*.

Mencía, C., & Marques-Silva, J. (2014). Efficient relaxations of over-constrained CSPs. In *ICTAI 2014, pp. 725–732. IEEE*.

Morgado, A., Heras, F., Liffiton, M.H., Planes, J., & Marques-Silva, J. (2013). Iterative and core-guided MaxSAT solving: A survey and assessment. *Constraints*, *18*(4), 478–534.

Narodytska, N., & Bacchus, F. (2014). Maximum satisfiability using core-guided maxsat resolution, In Brodley, C.E., & Stone, P. (Eds.) *AAAI, pp. 2717–2723. AAAI Press*.

O'Callaghan, B., O'Sullivan, B., & Freuder, E.C. (2005). Generating corrective explanations for interactive constraint satisfaction, In van Beek, P. (Ed.) *CP 2005, LNCS, vol. 3709, pp. 445–459. Springer*.

Papadimitriou, C.M. (1994). Computational complexity. Addison-Wesley. Massachusetts: Reading.

Plaisted, D.A., & Greenbaum, S. (1986). A structure-preserving clause form translation. *J. Symbolic Comput*, *2*(3), 293–304.

Schrijver, A. (1998). *Theory of linear and integer programming*: Wiley-Interscience.

Selman, A.L. (1994). A taxonomy of complexity classes of functions. *J. Comput. Syst. Sci*, *48*(2), 357–381.

Sinz, C. (2005). Towards an optimal CNF encoding of boolean cardinality constraints, In van Beek, P. (Ed.) *CP 2005, LNCS, vol. 3709, pp. 827–831. Springer*.

Sinz, C., Kaiser, A., & Küchlin, W. (2003). Formal methods for the validation of automotive product configuration data. *AIEDAM*, *17*(1), 75–97.

Tseitin, G.S. (1970). On the complexity of derivations in the propositional calculus. *Studies in Constructive Mathematics and Mathematical Logic Part II*, 115–125.

Walter, R., Felfernig, A., & Küchlin, W. (2015). Inverse QuickXPlain vs. MaxSAT — a comparison in theory and practice, In Tiihonen, J., Falkner, A., & Axling, T. (Eds.) *Proc. of the 17th Int. Config. Workshop, pp. 97–104. Vienna, Austria*.

Walter, R., & Küchlin, W. (2014). ReMax – a MaxSAT aided product configurator, In Felfernig, A., Forza, C., & Haag, A. (Eds.) *Proc. of the 16th Int. Config. Workshop, pp. 59–66. Novi Sad, Serbia*.

Walter, R., Zengler, C., & Küchlin, W. (2013). Applications of MaxSAT in automotive configuration, In Aldanondo, M., & Falkner, A. (Eds.) *Proc. of the 15th Int. Config. Workshop, pp. 21–28. Vienna, Austria*.

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