On Exact Selection of Minimally Unsatisfiable Subformulae

Renato Bruni (bruni@dis.uniroma1.it)
Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", Via Buonarroti 12 - 00185 Roma, Italy

Abstract. A minimally unsatisfiable subformula (MUS) is a subset of clauses of a given CNF formula which is unsatisfiable but becomes satisfiable as soon as any of its clauses is removed. The selection of a MUS is of great relevance in many practical applications. This expecially holds when the propositional formula encoding the application is required to have a well-defined satisfiability property (either to be satisfiable or to be unsatisfiable). While selection of a MUS is a hard problem in general, we show classes of formulae where this problem can be solved efficiently. This is done by using a variant of Farkas' lemma and solving a linear programming problem. Successful results on real-world contradiction detection problems are presented.

Keywords: Infeasibility analysis, MUS selection, (Un)Satisfiability

1. Introduction

Several problems arising from different fields are usually encoded into propositional logic formulae. A propositional formula $\mathcal{F}$ in *conjunctive normal form* (CNF) is a conjunction of clauses C_j, each clause being a disjunction of literals, each literal being either a positive (α_i) or a negative ($\neg\alpha_i$) propositional variable, with $j \in \{1,\ldots,m\},\ i \in \{1,\ldots,n\}$. By denoting with I_j the set of variables of C_j, and with $[\neg]$ the possible presence of $\neg$, this is

$$\bigwedge_{j=1\ldots m} \Big(\bigvee_{i\in I_j} [\neg]\ \alpha_i\Big)$$

The *satisfiability* problem (SAT) consists in determining whether there exists a truth assignment $\{True,False\}$ for the variables such that $\mathcal{F}$ evaluates to *True*. Extensive references can be found in (Chandru and Hooker, 1999; Gu et al., 1997; Kleine Büning and Lettman, 1999; Truemper, 1998).

Generally, when an instance $\mathcal{F}$ encodes a system or a structure one must design, $\mathcal{F}$ should have a well-defined solution property (either to be satisfiable or to be unsatisfiable). When $\mathcal{F}$ is unsatisfiable, but it should be satisfiable, we would like to modify the underlying system in order to make $\mathcal{F}$ satisfiable. Conversely, when $\mathcal{F}$ is unsatisfiable and it should be so, if the underlying system needs to be re-designed, we would like to keep $\mathcal{F}$ unsatisfiable. The first problem can sometimes be

approached by solving the *maximum satisfiability* problem (Max-SAT), see e.g. (Battiti and Protasi, 1998). This consists in finding a truth assignment for the variables maximizing the number of clauses C_j which evaluates to *True*. So far, satisfiability can be restored by removing from the underlying system all elements corresponding to clauses which could not be satisfied. However, such approach is not desirable in many practical cases. Very often, in fact, we cannot just delete a part of our system, because we need the functionalities contained in that part. Instead, we would like to locate and understand the problem, and, basing on this information, re-design only the small part of the system causing the problem. As for the second problem, we typically would like to know which part of the underlying system should not be changed, and which one can be modified (or possibly removed). Both of the above problems can be approached by looking for a *minimally unsatisfiable subset* of clauses (MUS) within an unsatisfiable $\mathcal{F}$ (see Sect. 2).

An algorithm for selecting an approximation of a MUS is proposed in (Bruni, 2002). The problem of deciding whether a CNF formula contains a *minimally unsatisfiable* (MU) subformula of fixed deficiency δ, for all δ, is proved NP-complete in (Kleine Büning and Zhao, 2002). Related works in the field of propositional formulae are those on decomposition of a CNF with maximal deficiency $\delta^\star \leq k$ (where k is a constant) into the union of all MUS and the intersection of all *maximally satisfiable subformulae* in polynomial time (Kullmann, 2000), and on recognition of MU formulae with fixed deficiency δ in polynomial time (Fleischner et al., 2002).

The problem of the selection of an *irreducible infeasible subsystem* (IIS), which is the analogous of a MUS in the case of systems of linear inequalities, has been studied with regard to infeasibility analysis (Amaldi et al., 1999; Chinneck, 2001; Tamiz et al., 1996). In the case of systems of linear inequalities having real variables, the problem has been approached both by means of heuristics (Chinneck, 2001) and exact algorithms (Gleeson and Ryan, 1990). In the case of systems of linear inequalities having integer variables (more computationally demanding), the problem has been approached by means of additive or subtractive heuristics (Guieu and Chinneck, 1999).

A procedure for the exact selection of a MUS is here presented (Sect. 4). It is based on Farkas' lemma (Sect. 3) adapted from the linear to the binary case. While selection of a MUS is a hard problem in general, we show (Sect. 5) classes of formulae for which this can be done efficiently by solving a linear programming problem. This depends on the structure of the polytope defined by the linear relaxation of SAT. A compendium of studies on such structure is for instance in (Chandru and Hooker, 1999). This procedure is applied to real-world data mining

problems (Sect. 6), where logical *rules* are encoded into clauses (Bruni and Sassano, 2001). A contradiction in the set of rules corresponds to a set of clauses jointly unsatisfiable. Checking the rules for inconsistencies produces a series of MUS selection problems, where all the conflicting rules should be located, and it would not help deleting some of them.

2. Minimally Unsatisfiable Subformulae

Definition 1. A *minimally unsatisfiable subformula* (MUS) of a CNF formula $\mathcal{F}$ is a set $\mathcal{M}$ of clauses having the following properties:

- $\mathcal{M} \subseteq \mathcal{F}$ (in the sense of clause-subset, i.e. $C_j \in \mathcal{M} \Rightarrow C_j \in \mathcal{F}$).
- $\mathcal{M}$ is unsatisfiable.
- Every proper clause-subset of $\mathcal{M}$ is satisfiable.

Clearly, $\mathcal{F}$ contains a MUS if and only if $\mathcal{F}$ is unsatisfiable. Therefore, the problem of deciding whether a formula $\mathcal{F}$ contains a MUS is co-NP-complete, since SAT is well-known NP-complete. A MUS may be a proper subformula of $\mathcal{F}$ or coincide with $\mathcal{F}$. Sometimes in literature, adverb "minimally" is replaced, with same meaning, by adjective "minimal".

Definition 2. Given a CNF formula $\mathcal{F}$, the *MUS selection* problem consists in finding a minimally unsatisfiable subformula $\mathcal{M} \subseteq \mathcal{F}$ or proving that such a subformula does not exist.

In the general case, more than one MUS can be contained in the same $\mathcal{F}$. Some of them can overlap, in the sense that they can share some clauses, but they cannot be fully contained one in another. Formally, the collection of all MUS of $\mathcal{F}$ is a *clutter*. Relations between the structures of Max-SAT solution and MUS are investigated in (Bruni, 2002). Note that the MUS selection problem is different, although closely related, from the *minimal unsatisfiable subformula* problem of deciding whether a CNF formula contains a subformula in $MU(k)$, see (Kleine Büning and Zhao, 2002). Besides, empirical experience suggests that finding a MUS typically requires much more time than just solving the SAT problem, just like finding an IIS requires much more time than just solving the feasibility of a system of linear inequalities (Guieu and Chinneck, 1999).

The problems depicted in Sect. 1 correspond to the problem of selecting a MUS, as follows. In the case we want to restore satisfiability

by locating only the small part of the underlying system causing the problem, this actually can be solved by locating a MUS. Re-design of that part is another issue, and, typically, requires again the work of the original (human or not) designer of the system. *Postinfeasibility analysis*, in fact, often needs "cooperation of *algorithmic engine* and *human intelligence*" (Chinneck and Dravnieks, 1991). The process could need to be repeated until all MUS are removed from the formula.

Conversely, in the case when unsatisfiability should be kept while modifying the underlying system, this again can be solved by locating a MUS. That is the part of the system that should not be changed.

3. The Linear Case

In the case of systems of linear inequalities, when we are interested in real-valued solutions, the following result on infeasibility holds:

THEOREM 1. (Farkas' lemma, 1894) *Let A be an $h \times k$ real matrix and let b be a real h-vector. Then there exists a real k-vector $x \geq \mathbf{0}$ with $Ax = b$ if and only if $y^T b \geq 0$ for each real h-vector y with $y^T A \geq \mathbf{0}$.*

A proof is for instance in (Schrijver, 1986). Geometrically, this means that if an h-vector b does not belong to the cone generated by the h-vectors $a_1, \ldots, a_k$ (columns of A), there exists a linear hyperplane separating b from $a_1, \ldots, a_k$.

There are several other equivalent forms of Farkas' lemma, and we now convert it in one which is more suitable to our purposes. It can be put as the alternative feasibility of two linear systems by requiring $y^T b < 0$ instead of $y^T b \geq 0$. Moreover, removing the limitation $x \geq \mathbf{0}$, we have for the alternative system $y^T A = \mathbf{0}$. Finally, asking for $Ax \leq b$, we have for the alternative system $y \geq \mathbf{0}$ (for details, see e.g. Bertsimas and Tsitsiklis, 1997). The following variant is therefore obtained: given a matrix $A \in \mathbb{R}^{\mathrm{h} \times \mathrm{k}}$ and a vector $b \in \mathbb{R}^{\mathrm{h}}$, consider the system of linear inequalities:

$$\begin{cases} Ax \leq b \\ x \in \mathbb{R}^{\mathrm{k}} \end{cases} \tag{1}$$

and the new system of linear inequalities obtained from the former one:

$$\begin{cases} y^T A = \mathbf{0} \\ y^T b < 0 \\ y \geq \mathbf{0} \\ y \in \mathbb{R}^{\mathrm{h}} \end{cases} \tag{2}$$

We have that exactly one of the two following possibilities holds:

– (1) is feasible, i.e. there exists $x \in \mathbb{R}^{\mathrm{k}}$ verifying all its inequalities.

– (2) is feasible, i.e. there exists $y \in \mathbb{R}^{\mathrm{h}}$ verifying all its inequalities.

An *irreducible infeasible subsystem* (IIS) is a subset of the inequalities of an infeasible system that is itself infeasible, but for which any proper subset is feasible. Clearly, it constitutes the analogous of a MUS in the case of systems of linear inequalities. An IIS can be selected within (1) by solving the following new system:

$$\begin{cases} y^T A &= \mathbf{0} \\ y^T b &\leq -1 \\ y &\geq \mathbf{0} \\ y &\in \mathbb{R}^{\mathrm{h}} \end{cases} \tag{3}$$

The *support* of a vertex denotes the indices of its non-zero components; $\mathbf{0}$ and $\mathbf{1}$ respectively denote vectors of zeroes and ones of appropriate dimension.

THEOREM 2. (Gleeson and Ryan, 1990) *Consider two systems of linear inequalities in the form (1) and (3). If (3) is infeasible, (1) is feasible. On the contrary, if (3) is feasible, (1) is infeasible, and, moreover, each IIS of (1) is given by the support of each vertex of the polyhedron (3).*

The proof is based on polyhedral arguments using properties of extreme rays, see (Gleeson and Ryan, 1990). Therefore, checking the feasibility of (1), and, if infeasible, identifying one of its IIS, becomes the problem of finding a vertex of a polyhedron.

4. Propositional Formulae: the General Case

In the case of propositional formulae, it is well-known that a clause

$$(\bigvee_{i \in \pi} \alpha_i \vee \bigvee_{i \in \nu} \neg \alpha_i)$$

can be expressed as a linear inequality by using n binary variables $x_i \in \{0, 1\}$ corresponding to the propositional variables $\alpha_i \in \{\mathit{False}, \mathit{True}\}$, and by defining the incidence m-vectors of the set of its positive literals b^π and of that one of its negative literals b^ν

$$\sum_{i=1}^{n} b_i^\pi x_i + \sum_{i=1}^{n} b_i^\nu (1 - x_i) \geq 1 \tag{4}$$

Equivalently, by denoting with $|\nu|$ the number of negative literals in the clause, this can be rewritten as

$$\sum_{i=1}^{n} b_i^{\nu} x_i - \sum_{i=1}^{n} b_i^{\pi} x_i \leq |\nu| - 1 \tag{5}$$

Denote now with B the $\{0, \pm 1\}^{m \times n}$ matrix whose rows correspond to clauses as shown above (each element is -1 if the corresponding propositional variables is positive, 1 if it is negative, 0 otherwise). Denote also with $\nu(B)$ the m-vector of all the $|\nu_j|$. The following system of linear inequalities with binary variables represents a CNF propositional formula.

$$\begin{cases} Bx & \leq & \nu(B) - \mathbf{1} \\ x & \in & \{0,1\}^n \end{cases} \tag{6}$$

In order to use the results given for the linear case, let us consider the linear relaxation of such system.

$$\begin{cases} Bx & \leq & \nu(B) - \mathbf{1} \\ x & \leq & \mathbf{1} \\ -x & \leq & \mathbf{0} \\ x & \in & \mathbb{R}^{\mathrm{n}} \end{cases} \tag{7}$$

We will suppose that our formula always contains at least a unit clause, i.e. a clause containing a single literal, since in the absence of that the linear relaxation (7) is always feasible. Note that, in practical applications, such assumption is generally verified. The above system (7) is now in the form of (1). The first group of inequalities of type (5) are called *clausal inequalities*. In particular, the matrix $A \in \{0, \pm 1\}^{(m+2n) \times n}$ and the vector $b \in \mathbb{Z}^{m+2n}$ are composed as follows.

$$A = \begin{bmatrix} B \\ I \\ -I \end{bmatrix} \begin{matrix} m \\ n \\ n \end{matrix} \qquad b = \begin{bmatrix} \nu(B) - \mathbf{1} \\ \mathbf{1} \\ \mathbf{0} \end{bmatrix} \begin{matrix} m \\ n \\ n \end{matrix}$$

Therefore, a system which plays the role of (3) can now be written.

$$\begin{cases} y^T \begin{bmatrix} B \\ I \\ -I \end{bmatrix} & = & \mathbf{0} \\ y^T \begin{bmatrix} \nu(B) - \mathbf{1} \\ \mathbf{1} \\ \mathbf{0} \end{bmatrix} & \leq & -1 \\ y_1, \ldots, y_h & \geq & 0 \\ y & \in & \mathbb{R}^{\mathrm{h}} \end{cases} \tag{8}$$

So far, the following result on the couple of systems (6) and (8) holds. The *restriction* of the support of a vertex to clausal inequalities will denote the indices of its non-zero components among those corresponding to clausal inequalities.

THEOREM 3. *Consider two systems of linear inequalities in the form (6) and (8). In this case, if (8) is feasible, (6) is infeasible, and the restriction of the support of each vertex of the polyhedron (8) to clausal inequalities contains a MUS of (6). On the contrary, if (8) is infeasible, (7) is feasible, but it cannot be decided whether (6) is feasible or not.*

Proof. We first prove that the restriction of the support of a vertex of (8) to clausal inequalities contains a MUS of (6). Assume (8) is feasible, and let v_1 be the vertex found. Therefore, (7) is infeasible (from Theorem 2), and an IIS in (7), called here IIS_1, is given by the support of v_1. Such IIS_1 is in general composed by a set CI_1 of *clausal inequalities* (inequalities of the type (5)) and a set BC_1 (possibly empty) of *box constraints* (the ones imposing $0 \leq x_i \leq 1$). The set of inequalities CI_1 has no $\{0,1\}$ solutions, since removing the BC_1 from IIS_1, while imposing the more strict *integer constraints* IC_1 (the ones imposing $x_i \in \{0,1\}$), keeps IIS_1 unsatisfiable. Therefore, a MUS is contained into the clauses corresponding to CI_1. Such MUS can still be a subset of the clauses corresponding to CI_1, because, though $IIS_1 = CI_1 \cup BC_1$ is minimally infeasible, imposing the more strict integer constraints can make $CI_1 \cup IC_1$ not minimal.

On the other hand, not all MUS in (6) can be obtained by such procedure. This because, if (8) is infeasible, (7) is feasible (by Theorem 2). When imposing the more strict integer constraints instead of the box constraints, however, nothing can be said on the feasibility of (6).

Example. Consider $\mathcal{F}_1$ composed by the four following clauses.

$$C_1 = (\alpha_1 \vee \alpha_2),\ C_2 = (\neg\alpha_1 \vee \alpha_2),\ C_3 = (\neg\alpha_2),\ C_4 = (\alpha_1 \vee \neg\alpha_2)$$

A and b can easily be obtained, in the following manner.

$$A = \left[\begin{array}{cc} -1 & -1 \\ 1 & -1 \\ 0 & 1 \\ -1 & 1 \\ \hline 1 & 0 \\ 0 & 1 \\ \hline -1 & 0 \\ 0 & -1 \end{array}\right] \qquad b = \left[\begin{array}{c} -1 \\ 0 \\ 0 \\ 0 \\ \hline 1 \\ 1 \\ \hline 0 \\ 0 \end{array}\right]$$

Therefore, the system to be solved, in the form of (8), is the following.

$$\begin{cases} -y_1 + y_2 - y_4 + y_5 - y_7 = 0 \\ -y_1 - y_2 + y_3 + y_4 + y_6 - y_8 = 0 \\ -y_1 + y_5 + y_6 \leq -1 \\ y_1,\, y_2,\, y_3,\, y_4,\, y_5,\, y_6,\, y_7,\, y_8 \geq 0 \\ y \in \mathbb{R}^8 \end{cases}$$

Solving such system yields the vertex (1, 1, 2, 0, 0, 0, 0, 0). Therefore, a MUS in $\mathcal{F}_1$ is given by the set of clauses $\{C_1, C_2, C_3\}$ (and $\mathcal{F}_1$ is proved unsatisfiable).

From the practical point of view, for the motivations reported above, we are interested in MUS composed by a small number of clauses. Moreover, it may happen that not all clauses are equally preferable for the composition of the MUS that we are selecting. When this can be evaluated, a cost c_j for taking each clause C_j into the MUS that is being selected can be assigned. Such costs c_j for the clauses corresponds to costs for the variables of system (8). Therefore, a cost h-vector is computed. Its first m components will correspond to clausal inequalities, while the last $2n$ components will correspond to box constraints. So far, the solution of the following linear program produces a MUS having the desired clause composition.

$$\begin{cases} \min \sum_{j=1}^{h} c_j y_j \\ \text{s.t.} \quad y^T \begin{bmatrix} B \\ I \\ -I \end{bmatrix} = \mathbf{0} \\ y^T \begin{bmatrix} \nu(B) - \mathbf{1} \\ \mathbf{1} \\ \mathbf{0} \end{bmatrix} \leq -1 \\ y \geq \mathbf{0}, \quad y \in \mathbb{R}^{\mathrm{h}} \end{cases} \tag{9}$$

The result of Theorem 3 is not completely analogous to the linear case. In order to obtain more analogy, let us define the following property. An *integral point* will denote a point having all integer components.

Integral-point property: A class of polyhedra which, if non-empty, contain at least one integral point, has the integral-point (IP) property.

THEOREM 4. *If the polyhedron (7), which is the linear relaxation of (6), has the integral-point property, the following holds. If (8) is infeasible, (6) is feasible. On the contrary, if (8) is feasible, (6) is infeasible*

and each MUS is given by the restriction of the support of each vertex of polyhedron (8) to clausal inequalities.

Proof. We first prove that (8) infeasible ⇒ (6) feasible.
When (8) is infeasible, (7) is feasible (by Theorem 3). Since we assumed that the IP-property holds for (7), it contains at least one integral point. Since the box constraints hold for (7), such integer point must have {0,1}-components, hence (6) is feasible.

We now prove that (8) feasible ⇒ a MUS in (6) is given by the restriction of the support of a vertex in (8) to clausal inequalities. Denote such set of clausal inequalities by CI_1, and denote also by BC_1 and IC_1 respectively the box constraints and the integer constraints on the variables appearing in CI_1. It was already proved that, if (8) is feasible, CI_1 corresponds to an unsatisfiable subset of clauses. We prove, by contradiction, that the set of clauses corresponding to CI_1 is minimal. Assuming $CI_1 \cup IC_1$ not minimal, there is a smaller set of clausal inequalities ${CI_1}' \subset CI_1$ such that ${CI_1}' \cup IC_1$ is infeasible. On the other hand, $CI_1 \cup BC_1$ is minimal (by Theorem 2), so the set of inequalities ${CI_1}' \cup BC_1$ must be feasible. However, we assumed that the IP-property holds for (7), so ${CI_1}' \cup BC_1$ contains at least one integral point having {0,1}-components, which is the contradiction.

Therefore, CI_1 corresponds to a MUS.

So far, when the IP property holds, solving a linear programming problem solves the MUS selection problem. There are several classes of formulae for which the linear relaxation (7) defines a polyhedron having the integral-point property. This is discussed in next section. The following Table I reports, for a number of known classes of propositional CNF, whether such property holds or not.

Table I. When does the Integral-point property hold.

Class of Propositional Formulae	IP Property
Horn (Dowling and Gallier, 1984; Scutellà, 1990)	yes
Renamable Horn (Lewis, 1978; Aspvall, 1980)	yes
Q-Horn (Boros et al., 1990)	no
Extended Horn (Chandru and Hooker, 1991)	yes
Balanced (Conforti and Cornuéjols, 1995; Truemper, 1998)	yes
Quadratic (Aspvall et al., 1979)	no
(Extended) Nested (Knuth, 1990; Hansen et. al., 1993)	no
SLUR solvable (Schlipf et al., 1995)	no
Matched (Franco and Van Gelder, 2002)	yes
Linear Autarkies (van Maaren, 2000; Kullmann, 2000)	no

Classes considered above are already known to be easy cases for the satisfiability problem. Not surprisingly, no easy classes for the MUS selection problem which were not known to be easy classes for the Satisfiability problem appear here. This because, since MUS selection gives the answer to SAT, that would be an easy class for SAT as well. However, considerable research effort has been spent on searching easy classes for SAT for at last two decades, hence discovering entirely new ones does not seem an easy task.

5. Special Classes of Propositional Formulae

Two interesting classes of propositional CNF formulae verify the integral-point property: formulae which are *extended Horn*, which include *Horn* and *renamable Horn*, and formulae which are *balanced*. Moreover, two other classes of formulae share the IP property, and are incomparable with the above ones. The first is the class of *matched* formulae (Franco and Van Gelder, 2002). The second is the class of *satisfiable quadratic* CNF (and consequently satisfiable Q-Horn, etc.), since quadratic formulae are renamable Horn if and only if they are satisfiable. However, the last two classes are always satisfiable, so they are not of practical interest with respect to the MUS selection problem.

Extended Horn formulae are characterizable as follows:

THEOREM 5. (Rounding theorem, Chandrasekaran, 1990) *Given a system $Ax \geq b$, $x \geq \mathbf{0}$, with A being a $h \times k$ integral matrix and b an integral h-vector. If there exist a $k \times k$ matrix T such that:*

– *T and T^{-1} are integral;*

– *each row of T^{-1} has at most one negative entry, and it is -1;*

– *each row of AT^{-1} has at most one negative entry, and it is -1;*

then, if x is a (fractional) solution to the above system, so is $T^{-1}\lceil Tx \rceil$.

The proof, in (Chandrasekaran, 1990), is based on properties of polyhedra having an integral largest element. In the case of a CNF, the above can be particularized as follows (Chandru and Hooker, 1991). Consider again the Satisfiability problem, written using inequalities in the form (4). Denote with D the $\{0, \pm 1\}^{m \times n}$ matrix whose rows correspond to clauses (its elements are 1 if the corresponding propositional variables is positive, -1 if it is negative, 0 otherwise), and with d the opportune right-hand-side m-vector.

$$\begin{cases} Dx & \geq & d \\ -x & \geq & -\mathbf{1} \\ x & \geq & \mathbf{0} \\ x & \in & \mathbb{Z}^n \end{cases} \tag{10}$$

Its linear relaxation, when put in the form of Theorem 5, becomes:

$$\begin{cases} \begin{bmatrix} D \\ -I \end{bmatrix} x & \geq & \begin{bmatrix} d \\ -\mathbf{1} \end{bmatrix} \\ x & \geq & \mathbf{0} \\ x & \in & \mathbb{R}^n \end{cases} \tag{11}$$

Conditions required in Theorem 5, in the case of CNF formulae, translate as follows. Each row of T^{-1} should have at most one $+1$ and at most one -1. By adding a new $\{0, \pm 1\}$ column vector r to T^{-1}, whose values are such that the $\{0, \pm 1\}^{(n+1) \times n}$ matrix $[T^{-1}|r]$ has exactly one $+1$ and one -1 per row, we obtain the arcs-nodes incidence matrix of a direct graph with $n + 1$ nodes (nodes correspond to the columns of $[T^{-1}|r]$). Such digraph should be a *rooted arborescence*, i.e. a rooted directed tree in which all arcs point away from the root r. Arcs of such digraph corresponds to variables of the formula. The rows of D (i.e. the clauses) can be interpreted as flows on the defined digraph: a positive [negative] literal is a unit flow going along [opposite to] the arc corresponding to that variable. If the flow has the so-called *extended star-chain property* (Chandru and Hooker, 1991), that is it can be partitioned into a (possibly empty) set of unit flows going into the root on an extended star, and a (possibly empty) unit flow on one chain, all the conditions of Theorem 5 are verified, and the CNF is called extended Horn.

Therefore, if (11) has a fractionary solution, (10) has an integral solution, so extended Horn formulae verify the IP property. Recognition of extended Horn formulae is not known to be solvable in polynomial time. The problem of the *arborescence realization* arise (Swaminathan and Wagner, 1986). However, formulae can be build in order to be extended Horn, by checking (in linear time) if each new clause defines an acceptable flow in the arborescence (Chandru and Hooker, 1991). Therefore, the entire procedure of MUS selection can be done in polynomial time, if our application supports to be designed by testing each new clause, and accepting it only when extended-Horn.

The second class is composed by balanced formulae, defined below.

Definition 3. A $\{0, \pm 1\}$ matrix is *balanced* if every square submatrix with exactly two nonzeroes entries in each row and column sums to a multiple of four.

The class of balanced matrices contains the classes of *totally unimodular matrices* and *network matrices* (see Truemper, 1998). Note that the property of balancedness is a general property, in the sense that it does not requires the matrix to represent a CNF formula. When such property is verified by a matrix representing a CNF formula (in the form (10), or, equivalently, in the form (6)), the following result holds.

THEOREM 6. (Conforti and Cornuéjols, 1995) *If the matrix representing the CNF instance in form (10) is balanced, the polytope is integral, i.e. has all integral extreme points.*

The proof, not immediate, is developed using polyhedral theory. The problem of recognition of balanced matrix has polynomial-time complexity. The algorithm which solves it is based on decomposition results, and proceeds by decomposing the given matrix, and checking if all the obtained submatrices are balanced (Conforti et al., 1994). Therefore, the entire procedure for MUS selection requires polynomial time on balanced matrices.

One can observe that another procedure to find a MUS could be based on *additive* or *subtractive filters*. Such methods would at first test if formula $\mathcal{F}$ is unsatisfiable. If so, additive methods would generate a new formula $\mathcal{F}'$ by adding clauses form $\mathcal{F}$ until $\mathcal{F}'$ is unsatisfiable. So far, $\mathcal{F}'$ is the selected IIS. The algorithm proposed in (Bruni, 2002) is an evolution on additive methods. Subtractive methods would instead remove from $\mathcal{F}$ each clause C_j not needed for unsatisfiability by testing if the formula remains unsatisfiable when removing C_j. What remains of $\mathcal{F}$ is the selected IIS. Such methods can therefore guarantee to find a MUS when it exists. However, they require to solve (at most) m

times the *base* problem (satisfiability for MUS selection, feasibility for IIS selection). When this can be done in polynomial time, the whole procedure would be theoretically polynomial. However, similar "brute force" approach are recognized to be quite computationally demanding (Chinneck and Dravnieks, 1991; Gleeson and Ryan, 1990). Moreover, the proposed approach allows a more powerful manner of choosing the clause composition of selected MUS, by using the cost function in (9).

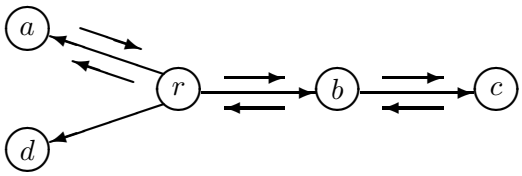
Example. Consider the following CNF formula $\mathcal{F}_2$.

$$(\neg\alpha_1 \vee \alpha_2 \vee \alpha_3) \wedge (\alpha_1 \vee \neg\alpha_2 \vee \neg\alpha_3) \wedge (\alpha_2 \vee \neg\alpha_4) \wedge (\alpha_4)$$

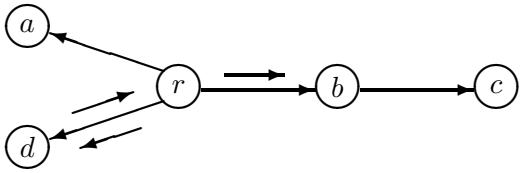
Suppose also that the following T^{-1} matrix could be obtained.

$$T^{-1} = \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \left(\begin{array}{cccc} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{array}\right) \end{array} \begin{array}{c} r \\ 1 \\ 1 \\ 0 \\ 1 \end{array}$$

The rooted arborescence corresponding to T^{-1} is now as follows. The flow corresponding to the first [second] clause is given by the shorter arrows drown over [under] the arcs.



The flow corresponding to the third [forth] clause is given by the shorter arrows drown over [under] the arcs.



Such flows have the extended star-chain property. Therefore, although neither Horn nor renamable Horn, $\mathcal{F}_2$ is extended Horn, so we are in the conditions of Theorem 4.

Suppose now that we are unable to obtain a suitable T^{-1} matrix. In this case, the following matrix, corresponding to $\mathcal{F}_2$ in form (10), must

be tested for balancedness.

$$\begin{pmatrix} -1 & 1 & 1 & 0 \\ 1 & -1 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

It's easy to see that every square submatrix with two nonzeroes per row and per column sums to 0, hence it is balanced, so we are again assured to be in the conditions of Theorem 4. So far, A and b are built as follows.

$$A = \left[\begin{array}{rrrr} 1 & -1 & -1 & 0 \\ -1 & 1 & 1 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & -1 \\ \hline 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \hline -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{array}\right] \qquad b = \left[\begin{array}{r} 0 \\ 1 \\ 0 \\ -1 \\ \hline 1 \\ 1 \\ 1 \\ 1 \\ \hline 0 \\ 0 \\ 0 \\ 0 \end{array}\right]$$

Since the IP property is verified, the MUS selection problem is completely solved by solving the following system.

$$\begin{cases} y_1 - y_2 + y_5 - y_9 = 0 \\ -y_1 + y_2 - y_3 + y_6 - y_{10} = 0 \\ -y_1 + y_2 + y_7 - y_{11} = 0 \\ y_3 - y_4 + y_8 - y_{12} = 0 \\ y_2 - y_4 + y_5 + y_6 + y_7 + y_8 \leq -1 \\ y_1, y_2, y_3, y_4, y_5, y_6, y_7, y_8, y_9, y_{10}, y_{11}, y_{12} \geq 0 \\ y \in \mathbb{R}^{12} \end{cases}$$

This system is infeasible. Hence, we are guaranteed that no MUS is present in $\mathcal{F}_2$, which is therefore satisfiable.

6. Implementation and Computational Experience

Many problems of error detection or classification into large data-sets are solved by using *rules* (Fellegi and Holt, 1976). By encoding the rules into clauses, an *inconsistency*, or *contradiction*, in the set of rules

corresponds to a set of clauses jointly unsatisfiable (Bruni and Sassano, 2001). Since rules should be free from contradiction, this is one of the cases when the resulting logic formula should be satisfiable. Therefore, checking the rules for inconsistencies produces MUS selection problems. Note that all the conflicting rules should be located, and it would not help deleting some of them. In particular, detection of partial inconsistencies (which are inconsistencies having effect only for some values of the data) produces a series of instances which are very similar. CNF formulae generated for the validation of rules used for a real-world census are here considered.

The proposed algorithm is implemented in C++ and runs on a Pentium IV PC. After the initial routine converting rules into clauses and generating the CNF formulae, the procedure composes the above shown matrix A and vector b as in (8). By adding an opportune objective function, a problem in the form (9) is then passed to a standard routine for solving linear programming (ILOG Cplex [1]) implementing the simplex algorithm. When a vertex is found, its support is used to produce the MUS which is given in output. Although the simplex method has exponential-time complexity, and there are polynomial-time complexity methods for solving the same problem (e.g. barrier's method) and then finding a vertex, the former alternative is generally reckoned to be faster in practice (see for instance Bertsimas and Tsitsiklis, 1997 for a description). When the problem in form (9) is infeasible, either we check that we are in one of the described special classes (such check depending on the problem), and in such case no MUS exist, or we need to solve the satisfiability problem for the original CNF formula. However, for all considered real problems, when (9) is infeasible, the original formula turn out to be satisfiable.

Table II reports number of variables (n) and number of clauses (m) both for the original formula and for the selected MUS, in addition to computational times (in seconds). We report only instances corresponding to partial inconsistencies, hence formulae containing a MUS. Those results are intended to give an example of application, rather than exploring all the computational possibilities of the proposed procedure, since the latter is not the focus of present paper.

[1] More information available at `www.cplex.com`.

Table II. Real-world contradiction detection problems.

Original formula			Selected MUS		
Name	n	m	n	m	time
EditPartIncons0	3,000	15,003	2	3	1.0
EditPartIncons1	3,000	15,002	21	22	4.1
EditPartIncons2	3,000	15,003	4	5	8.3
EditPartIncons3	3,000	15,003	3	4	6.5
EditPartIncons4	3,000	15,015	30	31	4.2
EditPartIncons5	3,000	15,002	1	2	0.9
EditPartIncons6	3,000	15,005	16	17	4.1
EditPartIncons7	3,000	15,003	6	7	6.5
EditPartIncons8	3,000	14,998	2	3	3.2
EditPartIncons9	3,000	15,003	2	3	1.6

In the majority of the cases, only one MUS was present in the CNF, that means only one inconsistency was present in the set of rules. If, however, after repairing the found inconsistency, the new CNF still contains a MUS, that is another inconsistency, and should as well be repaired independently. The whole procedure, according to human experts having the charge of writing the rules, turn out to be a very satisfactory tool for the design of a contradiction-free set of rules.

7. Conclusions

The problem of *MUS selection* is formally defined. Such problem is computationally hard and arises in several applicative fields. This typically happens when the application is encoded into a propositional formula which should have a well-defined satisfiability property (either to be satisfiable or to be unsatisfiable).

A procedure for solving the MUS selection problem is here presented. Under special conditions, the proposed procedure is able to exactly solve such problem by simply solving a linear programming problem, which can be done with polynomial-time or simplex based algorithms. Known classes of CNF formulae are studied with respect to verification of such special conditions. The largest classes of CNF formulae verifying the above conditions result to be *extended Horn* and *balanced formulae*. Computational experience on real-world data mining problems is very encouraging.

Acknowledgements

The author whish to thank Carlo Mannino for helpful discussions.

References

Amaldi, E., M.E. Pfetsch, and L. Trotter, Jr. 1999. Some structural and algorithmic properties of the maximum feasible subsystem problem. *in proc. of 10th Integer Programming and Combinatorial Optimization conference.*, Lecture Notes in Computer Science 1610, Springer-Verlag, 45–59.

Aspvall, B. 1980. Recognizing disguised NR(1) instance of the satisfiability problem. *Journal of Algorithms* 1, 97-103.

Aspvall, B., M.F. Plass, and R.E. Tarjan. 1979. A linear time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters* 8, 121-123.

Battiti, R. and M. Protasi. 1998. Approximate Algorithms and Heuristics for MAX-SAT. In D.Z. Du and P.M. Pardalos eds. *Handbook of Combinatorial Optimization*, Kluwer Academic Publishers, 1, 77-148.

Bertsimas, D. and J.N. Tsitsiklis. 1997. *Introduction to Linear Optimization*. Athena Scientific, Belmont, Massachusetts.

Boros, E., Y. Crama, and P.L. Hammer. 1990. Polynomial time inference of all valid implications for Horn and related formulae. *Annals of Mathematics and Artificial Intelligence* 1, 21-32.

Bruni, R. 2002. Approximating Minimal Unsatisfiable Subformulae by means of Adaptive Core Search. To appear in *Discrete Applied Mathematics*.

Bruni, R. and A. Sassano. 2001. Error Detection and Correction in Large Scale Data Collecting. In *Advances in Intelligent Data Analysis*, Lecture Notes in Computer Science 2189, Springer, 84-94.

Chandrasekaran, R. 1984. Integer programming problems for which a simple rounding type of algorithm works. In W.R. Pulleyblank, ed. *Progress in Combinatorial Optimization*, Academic Press Canada, 101-106.

Chandru, V. and J.N. Hooker. 1991. Extend Horn clauses in propositional logic. *Journal of the ACM* 38, 203-221.

Chandru, V. and J.N. Hooker. 1999. *Optimization Methods for Logical Inference*. Wiley, New York.

Chinneck, J.W. 2001. Fast Heuristics for the Maximum Feasible Subsystem Problem. *INFORMS Journal on Computing* 13(3), 210-223.

Chinneck, J.W. and E.W. Dravnieks. 1991. Locating Minimal Infeasible Constraint Sets in Linear Programs. *ORSA Journal on Computing* 3, 157-168.

Conforti, M. and G. Cornuéjols. 1995. A class of logical inference problems soluble by linear programming. *Journal of the ACM* 42(5), 1107-1113.

Conforti, M., G. Cornuéjols, A. Kapoor, and K. Vuskovic. 1994. Recognizing balanced 0, + or - 1 matrices. In *Proceedings 5th annual SIAM/ACM Symposium on Discrete Algorithms*, 103-111.

Dowling, W.F. and J.H. Gallier. 1984. Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *Journal of Logic Programming* 1, 267-284.

Fellegi, P. and D. Holt. 1976. A Systematic Approach to Automatic edit and Imputation. *Journal of the American Statistical Association*, 71(353), 17-35.

Fleischner, H., O. Kullmann, and S. Szeider. 2002. Polynomial-Time Recognition of Minimal Unsatisfiable Formulas with Fixed Clause-Variable Difference. To appear in *Theoretical Computer Science*.

Franco, J. and A. Van Gelder. 2002. A Perspective on Certain Polynomial Time Solvable Classes of Satisfiability. To appear in *Discrete Applied Mathematics*.

Garey, M.R. and D.S. Johnson. 1979. *Computers and Intractability*. Freeman, New York.

Gleeson, J. and J. Ryan. 1990. Identifying Minimally Infeasible Subsystems of Inequalities. *ORSA Journal on Computing* 2(1), 61-63.

Gu, J., P.W. Purdom, J. Franco, and B.W. Wah. 1997. Algorithms for the Satisfiability (SAT) Problem: A Survey. *DIMACS Series in Discrete Mathematics* 35, 19-151, American Mathematical Society.

Guieu, O. and J.W. Chinneck. 1999. Analyzing Infeasible Mixed-Integer and Integer Linear Programs *INFORMS Journal on Computing* 11 (1).

Hansen, P., B. Jaumard, and G. Plateau. 1993. An extension of nested satisfiability. *RUTCOR Technical Report* 29-93, Rutgers University, New Brunswick, NJ.

Kleine Büning, H. and T. Lettman. *Propositional logic: deduction and algorithms*. Cambridge University Press, Cambridge, 1999.

Kleine Büning, H. and X. Zhao. 2002. The Complexity of Read-Once Resolution. *Annals of Mathematics and Artificial Intelligence* 36, 419-435.

Knuth, D.E. 1990. Nested satisfiability. *Acta Informatica* 28, 1-6.

Kullmann, O. 2000. An application of matroid theory to the SAT Problem. *ECCC* TR00-018.

Lewis, H.R. 1978. Renaming a set of clauses as a Horn set. *Journal of the ACM* 25(1), 134-135.

Van Maaren, H. 2000. A Short Note on Some Tractable Cases of the Satisfiability Problem. *Information and Computation* 158, 125-130.

Schlipf, J.S., F.S. Annexstein, J.V. Franco, and R.P. Swaminathan. 1995. On Finding Solutions for Extended Horn Formulas. *Information Processing Letters* 54(3), 133-137.

Schrijver, A. 1986. *Theory of Linear and Integer Programming*. Wiley, New York.

Swaminathan, R.P., and D.K. Wagner. 1995. The arborescence-realization problem. *Discrete Applied Mathematics* 59, 267-283.

Scutellà, M.G. 1990. A note on Dowling and Gallier's top-down algorithm for propositional Horn satisfiability. *Journal of Logic Programming* 8, 265-273.

Tamiz, M., S.J. Mardle, and D.F. Jones. 1996. Detecting IIS in Infeasible Linear Programs using Techniques from Goal Programming. *Computers and Operations Research* 23, 113-191.

Truemper, K. 1998. *Effective Logic Computation*. Wiley, New York.