The Inequality-Satisfiability problem

Dorit S. Hochbaum *
Department of Industrial Engineering and Operations Research and
Walter A. Haas School of Business,
University of California, Berkeley
email: hochbaum@ieor.berkeley.edu
Erick Moreno-Centeno †
Department of Industrial Engineering and Operations Research
University of California, Berkeley
email: emc@ieor.berkeley.edu

May 30, 2007

Abstract

We define a generalized variant of the satisfiability problem (SAT) where each "clause" is an *or*-list of inequalities in n variables. The inequality satisfiability problem (I-SAT) is to find whether there exists a feasible point in $\Re^n$ that satisfies at least one inequality in each "clause". We show that I-SAT is harder than SAT in that I-SAT is NP-complete even when restricted to contain only two inequalities per "clause". We provide here an algorithm for solving an I-SAT on n variables and m "clauses" each containing up to k inequalities, with complexity dominated by $O((km)^{\min(n,m)-1}m(kn + \log m))$. In fact the complexity of the algorithm is polynomial when either the number of variables or the number of "clauses" is fixed. A problem of major interest in manufacturing called the *mold casting problem*, is shown to be a special case of an I-SAT on two variables and at most 9 inequalities per "clause".

Keywords: Satisfiability (SAT); Inequality Satisfiability; or-inequalities

1 Introduction

This paper introduces a new generalization of the satisfiability problem (SAT) with inequalities instead of literals. Our study of this problem, which we call the *inequality satisfiability* problem (I-SAT), has been motivated by a problem arising in the molding and casting manufacturing process.

The inequality satisfiability problem (I-SAT) is defined for given m *or*-lists of inequalities (to which we will refer as inequality-clauses or i-clauses), each of which contains up to k n-variable inequalities; the problem is to find an n-dimensional vector which satisfies at least one inequality per i-clause. A general instance of I-SAT is shown in Figure 1.

Another way to interpret I-SAT is to regard each i-clause as the union (OR) of up to k n-dimensional half-spaces. The problem is then to find a point in $\Re^n$ that lies in the intersection of all the half-space-unions defined by each i-clause, or to determine that no such point exists. In other words, each i-clause defines a (non-convex) set ($\subseteq \Re^n$) containing all the points $\in \Re^n$

*Research supported in part by NSF awards No. DMI-0085690 and DMI-0084857.
†PhD studies supported by CONACyT, Mexico grant No. 160619.

$$\left(\begin{array}{cccc} & \alpha_{1,1}^{(1)}x_1+\cdots+\alpha_{1,n}^{(1)}x_n & \leq & \beta_1^{(1)} \\ \bigcup & \vdots & \vdots & \vdots \\ \bigcup & \alpha_{k,1}^{(1)}x_1+\cdots+\alpha_{k,n}^{(1)}x_n & \leq & \beta_k^{(1)} \end{array}\right) \bigcap \cdots \bigcap \left(\begin{array}{cccc} & \alpha_{1,1}^{(m)}x_1+\cdots+\alpha_{1,n}^{(m)}x_n & \leq & \beta_1^{(m)} \\ \bigcup & \vdots & \vdots & \vdots \\ \bigcup & \alpha_{k,1}^{(m)}x_1+\cdots+\alpha_{k,n}^{(m)}x_n & \leq & \beta_k^{(m)} \end{array}\right)$$

Figure 1: A generic I-SAT instance with each i-clause containing up to k inequalities. The union $\bigcup$ expresses the "or" operation, and the intersection $\bigcap$ expresses the "and" operation.

satisfying at least one of its inequalities (i.e. lie in the union of the half-spaces defined by the i-clause's inequalities); and the problem is to find a point $\in \Re^n$ that belongs to the intersection of all such sets, or to determine that such intersection is empty.

For $k = 1$, I-SAT is a linear programming (LP) feasibility problem on n variables and m inequalities. However for $k \geq 2$, I-SAT is a generalization of SAT.

Our main result here is an algorithm for solving I-SAT, which we call the *projection algorithm*, with running time $T(m, n)$:

$$T(m,n) = \begin{cases} O\left((km)^m n\right) & \text{if } m \leq n \\ \\ O\left((km)^{n-1} m(kn + \log m)\right) & \text{if } n < m \end{cases} \tag{1}$$

This complexity demonstrates the polynomial time solvability of I-SAT for either a fixed number of variables or a fixed number of i-clauses. We further show here that I-SAT is NP-hard even for $k = 2$. We devise an algorithm for I-SAT for $n = 1$, and then establish a complexity lower bound for the problem. As the lower bound is equal to the upper bound - the complexity of the algorithm - this proves that this algorithm is the best possible for the problem. In other words, the *concrete complexity* of I-SAT for $n = 1$ is $\Theta(m(k + \log m))$. Finally we present a new algorithm for solving a generalization of the mold casting problem.

The paper is organized as follows: Section 2 includes an analysis of the computational complexity of I-SAT, and the result that even when restricted to contain at most 2 inequalities per i-clause, I-SAT remains NP-complete. Section 3 describes an algorithm for I-SAT with $n = 1$, a proof that this algorithm has best possible complexity, and the description of the projection algorithm for the general I-SAT problem. Finally section 4 provides details on the mold casting problem that motivated our study.

Throughout we assume that the coefficients and right-hand sides of the inequalities are rational or integral. Rational data is transformed to integer data by multiplying it by a suitably large number (e.g. the least common multiple of their denominators). With rational data the running time of solving a system of n equations in n variables is $O(n^3)$, [8]. Finite precision rationals are also important in implementing the "$<$" operation, which is used here.

2 The complexity of I-SAT

Let 2I-SAT refer to instances of I-SAT where each i-clause has at most 2 inequalities, and 3I-SAT refer to instances of I-SAT where each i-clause has at most 3 inequalities. We prove that 3I-SAT is NP-complete, implying that I-SAT is NP-complete. We then reduce 3I-SAT to 2I-SAT thus proving that 2I-SAT is NP-complete as well.

Lemma 2.1 3I-SAT $\in$ **NP** $-$ **complete**

Proof: We first show that I-SAT $\in$ **NP**. For an I-SAT instance to be feasible it must contain m inequalities (one from each i-clause) that are satisfied simultaneously. There always exists a solution to a system of m inequalities that is of polynomial-size in the length of the input. This is true since LP feasibility $\in$ **NP**. For such a certificate vector of polynomial length, $\mathbf{x}$, we can check in $O(kmn)$ time that $\mathbf{x}$ satisfies at least one inequality per i-clause.

To complete the NP-completeness proof we demonstrate that 3SAT is polynomial-time reducible to 3I-SAT. Consider any formula Φ consisting of m clauses $C_1, ..., C_m$, and n variables $a_1, ..., a_n$, and let a 3I-SAT instance be constructed as follows: For each variable $a_i \in \Phi$ there is a variable $b_i \in \Psi$; for each clause $C_j \in \Phi$ there is an i-clause $D_j \in \Psi$, and for the q^{th} literal of the j^{th} clause $l_q^j \in C_j$ there is an inequality $e_q^j \in D_j$. If $l_q^j := a_i$ then $e_q^j := b_i \geq 1$ and if $l_q^j := \neg\, a_i$ then $e_q^j := b_i \leq -1$. An example of this mapping between Φ and Ψ is given in equations (2) and (3) respectively.

$$\begin{aligned}
\Phi &= \bigcap_{j=1}^{m} C_j = \bigcap_{j=1}^{m} (l_1^j \cup l_2^j \cup l_3^j) \\
&= (\neg a_{1_1} \cup a_{1_2} \cup \neg a_{1_3}) \cap \cdots \cap (a_{j_1} \cup \neg a_{j_2} \cup a_{j_3}) \cap \cdots \cap (\neg a_{m_1} \cup a_{m_2} \cup a_{m_3}) \qquad (2) \\
\Psi &= \bigcap_{j=1}^{m} D_j = \bigcap_{j=1}^{m} (e_1^j \cup e_2^j \cup e_3^j) \\
&= \begin{pmatrix} & b_{1_1} & \leq & -1 \\ \bigcup & b_{1_2} & \geq & 1 \\ \bigcup & b_{1_3} & \leq & -1 \end{pmatrix} \bigcap \cdots \bigcap \begin{pmatrix} & b_{j_1} & \geq & 1 \\ \bigcup & b_{j_2} & \leq & -1 \\ \bigcup & b_{j_3} & \geq & 1 \end{pmatrix} \bigcap \cdots \bigcap \begin{pmatrix} & b_{m_1} & \leq & -1 \\ \bigcup & b_{m_2} & \geq & 1 \\ \bigcup & b_{m_3} & \geq & 1 \end{pmatrix} \qquad (3)
\end{aligned}$$

It is easy to see that Ψ is feasible if and only if Φ is satisfiable. ■

Although I-SAT is analogous to SAT, while 2SAT (SAT with at most two literals per clause) is solvable in polynomial-time [3], 2I-SAT is NP-complete when the number of variables is not fixed.

Theorem 2.2 2I-SAT $\in$ **NP** $-$ **complete**

Proof: As before 2I-SAT $\in$ **NP**. To prove the NP-completeness we reduce 3I-SAT to 2I-SAT in a manner analogous to the reduction from SAT to 3SAT.

Consider any formula Ψ consisting of m i-clauses $E_1, ..., E_m$, and n variables $a_1, ..., a_n$. We will construct a 2I-SAT instance Γ in at most $n + 2m$ variables such that Γ is feasible if and only if Ψ is feasible.

We examine the i-clauses $E_i \in \Psi$ one by one: If E_i has 2 or fewer inequalities we do nothing. Therefore we can restrict our attention to instances with exactly three inequalities in each i-clause. For E_i with three inequalities, we create 3 i-clauses F_i^j, $j = 1, 2, 3$ of Γ, each containing one of the three inequalities, say inequality r, $r = 1, 2, 3$, from E_i and inequality r of the following set:

$$\begin{aligned}
x_i &\leq 0 \\
y_i &\leq 0 \qquad (4) \\
x_i + y_i &\geq 1
\end{aligned}$$

An important property of the set of inequalities (4) is that it is infeasible, but any subset of two inequalities of the set of three is feasible. To prove that Ψ is feasible if and only if Γ is feasible it is sufficient to show that the 3 i-clauses F_i^j and E_i are equivalent.

First assume that the vector $\mathbf{p} \in \Re^n$ is feasible for E_i. This means that it satisfies at least one of its inequalities, say the 3rd inequality. This inequality appears in F_i^3 together with the inequality $x_i + y_i \geq 1$. Let an augmented vector $\mathbf{p}' \in \Re^{n+2m}$ that is equal to $\mathbf{p}$ on the first n variables and has $x_i = 0$ and $y_i = 0$. By the property of (4) $\mathbf{p}'$ satisfies F_i^1 and F_i^2 because the assignment of $x_i = y_i = 0$ satisfies the first two inequalities. Therefore, $\mathbf{p}'$ satisfies the three i-clauses corresponding to E_i.

For the converse we assume that a vector $\mathbf{q} \in \Re^{n+2m}$ is feasible for the three i-clauses F_i^j, $j = 1, 2, 3$, corresponding to E_i. Since the three inequalities (4) are not satisfied simultaneously, then $\mathbf{q}$ satisfies at least one of the inequalities in E_i. Thus projecting the vector $\mathbf{q}$ to the first n values will be a vector in $\Re^n$ that satisfies E_i (and all the other i-clauses). Finally, we note that this reduction is polynomial as we add at most $2m$ i-clauses and $2m$ variables. ■

3 The projection algorithm

We define a vector $\mathbf{x} \in \Re^n$ to be I-SAT *feasible* if it satisfies at least one inequality in each i-clause.

We first comment that there is an obvious LP-based algorithm for I-SAT as follows. Enumerate all possible systems of m inequalities, one from each i-clause, and check each such system for linear programming feasibility. The complexity of this algorithm is $O(k^m LP(m, n))$, where $LP(m, n)$ is the complexity of solving an LP feasibility problem in n variables and m inequalities. Since linear programming is polynomially solvable, then for a fixed number of i-clauses m, this is a polynomial time algorithm.

We first define and solve the *single-variable* I-SAT problem. This special case of I-SAT is such that all inequalities are defined on a unique variable (i.e. $n = 1$). Therefore this problem is defined on m clauses, each containing at most k upper/lower bound inequalities.

For a single-variable I-SAT it is easy to identify the set of all *infeasible* solutions for each i-clause. Since all inequalities in a single-variable i-clause are of the form of upper or lower bounds, e.g. $x_2 \leq u$ or $x_2 \geq \ell$, then their complements form *open* intervals, or strict inequality upper and lower bound constraints. Each such open interval is the set of all the infeasible real values which don't satisfy the corresponding inequality. Therefore, for each i-clause, the intersection of all these open intervals is the set of real values which do not satisfy any of its inequalities, and thus do not satisfy this i-clause. We call such interval for i-clause D_j the *i-infeasible interval* I_j.

This discussion shows that the single-variable I-SAT problem is equivalent to finding a real value that does *not* belong to a union of m intervals. Procedure 1-I-SAT shown next solves a single interval I-SAT problem by identifying such real value, or stating that none exists and therefore the instance of I-SAT is infeasible. Note that although the intervals may be rays, (a, ∞), we still refer to them as *intervals*.

procedure 1-I-SAT $(D_1, \ldots D_{m'})$

Step 1: For $j = 1, \ldots, m'$ find the i-infeasible interval of D_j, I_j, by comparing upper and lower bounds.

Step 2: For $I_j = (a_j, b_j)$ let $I_1, \ldots, I_{m'}$ be a sorted sequence according to the left endpoints, $a_1 \leq a_2 \leq \ldots \leq a_{m'}$. If $a_1 > -\infty$, then return *feasible*.

Step 3: Set $I(1) = I_1$.
For $q = 1, \ldots, m' - 1$, do
For $I(q) = (-\infty, b(q))$ and $I_{q+1} = (a_{q+1}, b_{q+1})$,
if $a_{q+1} < b(q)$, then $I(q + 1) = (-\infty, \max\{b(q), b_{q+1}\})$, else return *feasible*.

If $b_{q+1} = \infty$, then return *infeasible*.
enddo

Step 4: If $b(m') < \infty$, then return *feasible*.

Lemma 3.1 *The complexity of* procedure 1-I-SAT *for a single-variable* I-SAT *on m i-clauses is $O(m(k + \log m))$.*

Proof: Procedure 1-I-SAT requires $O(k)$ steps to find each i-infeasible interval, for a total of $O(km)$ time. The sorting of the lower bounds in Step 2 takes $O(m \log m)$ time. Finding the union $I(q)$ in Step 3 takes $O(m)$ operations in total. The complexity of the one variable I-SAT is therefore $O(m(k + \log m))$. ■

Next we give a lower bound on the number of steps required to solve a single-variable I-SAT problem on m i-clauses and at most k inequalities per i-clause. This lower bound is obtained under the *decision tree with linear tests model*. The reader is referred to [4] for a full description of this model.

Theorem 3.2 *Under the decision tree with linear tests model, a lower bound on the complexity of solving a single-variable* I-SAT *is $\Theta(m(k + \log m))$.*

Proof: Reading the input to the single-variable I-SAT problem requires $\Theta(km)$ steps. We use a result from [4] to show that also $\Theta(m \log m)$ is a lower bound on the complexity of the single-variable I-SAT. This implies, with Lemma 3.1, that $\Theta(m(k + \log m))$ steps are required to solve a single-variable I-SAT problem.

The *integer interval cover* problem is to decide whether the union of m intervals with integer endpoints covers an interval with integer endpoints. This problem is easily shown to be equivalent to solving a single-variable I-SAT problem on $m+2$ i-clauses each containing at most 2 inequalities.

Fredman and Weide [4] studied the evaluation version of the *interval cover* problem which requires computing the measure of the union of m intervals. They showed that under the decision tree with linear tests model the evaluation problem, even when restricting the interval endpoints to be integer, requires $\Theta(m \log m)$ steps to solve. Their argument applies directly to the decision version of the *integer interval cover* problem. The only adaptation needed is that each leaf node is labeled with "yes" or "no" rather than with a value m or less than m. Therefore this argument applies also for the single-variable I-SAT problem. ■

As the complexity of the algorithm established in Lemma 3.1 matches the lower bound proved in Theorem 3.2, it follows that this is the concrete complexity of the problem I-SAT for $n = 1$:

Corollary 3.3 *The concrete complexity of the single-variable I-SAT problem is $\Theta(m(k + \log m))$.*

Next we show how to solve the general I-SAT problem. For this purpose let us define a vector $\mathbf{x} \in \Re^n$ to be *binding* for an inequality if $\mathbf{x}$ satisfies it at equality.

Remark 3.4 *An I-SAT feasible solution exists if and only if there is an I-SAT feasible solution binding for at least one inequality.*

Proof: Obviously if $\mathbf{p}$ is an I-SAT feasible solution binding for some inequality, then $\mathbf{p}$ is also a feasible solution to the I-SAT problem. On the other hand if $\mathbf{p} \in \Re^n$ is an I-SAT feasible solution, then the system of m linear inequalities satisfied by $\mathbf{p}$, one from each i-clause, forms a convex

feasible set in $\Re^n$. Therefore there exists a feasible vector that is binding for at least one inequality. ■

This remark implies that it is sufficient to restrict the search for I-SAT feasible solutions to solutions that satisfy any of the $O(km)$ inequalities set as equalities.

The *projection algorithm* entails a recursive reduction of an I-SAT instance on n variables and m clauses to a collection of km I-SAT instances each on at most $n-1$ variables, and at most $m-1$ i-clauses. This reduction in number of variables and i-clauses is achieved by selecting a single inequality at a time, out of km inequalities, setting it as an equality. From this equation, one of the variables is selected arbitrarily, say x_i, and written as a linear function in the remaining variables. Any solution to this equation obviously satisfies this inequality, and thus the i-clause from which it was selected. We then substitute for x_i in all inequalities of the other i-clauses yielding a collection of up to $k(m-1)$ inequalities on the remaining variables in at most $m-1$ i-clauses. If the substitution in an inequality results in a tautology (e.g. $0 \leq 2$), then the entire i-clause can be removed (as there is always an inequality satisfied in that i-clause). And if the substitution results in a contradiction (e.g. $2 \leq 0$), then the contradictory inequality is removed from (the reduced) I-SAT instance. If all inequalities in a specific i-clause are removed, then this reduced I-SAT instance is infeasible.

Corollary 3.5 *An* I-SAT *problem on m clauses each containing at most k inequalities on n variables is reducible to $O(km)$* I-SAT *problems each in at most $m-1$ clauses each containing at most k inequalities on $n-1$ variables.*

At the n^{th} recursion level of the *projection algorithm* we have a collection of at most $(km)^{n-1}$ single-variable I-SAT problems each on at most $m-n$ i-clauses. If a feasible value is found in any of the single-variable I-SAT problems, then substituting the equation that generated this I-SAT problem we find an I-SAT feasible vector for the for the previous recursion level (with one more variable). Otherwise, if all the generated single-variable I-SAT problems are infeasible, then from Remark 3.4 the I-SAT problem is also infeasible. It might also be possible that some branch of this recursive procedure leads to an I-SAT problem containing a single i-clause. In this case, if any of the inequalities of such i-clause is not contradictory, then any of its feasible solutions is feasible for the original I-SAT problem. Note that when $m \leq n$ then all branches lead to I-SAT problems with a single i-clause containing inequalities on at most $n-m$ variables.

We next give the Projection Algorithm. For this purpose let $\mathcal{C}$ be the set of i-clauses of an I-SAT instance, where the j^{th} i-clause consists of k_j inequalities $e_1^j, \ldots, e_{k_j}^j$. For a vector $\mathbf{x} \in \Re^n$, denote by $\mathbf{x}^{-p}$ an $(n-1)$-dimensional vector restricting $\mathbf{x}$ to a projected subspace with the p^th dimension missing.

Projection Algorithm $(n, m, \mathcal{C})$
01: If $m = 1$, then return *feasible.*
02: If $n = 1$, then return procedure 1-I-SAT $(\mathcal{C})$.
03: For $j = 1, \ldots, m$ do
04: For $i = 1, \ldots, k_j$ do
05: Set $e_i^j \in D_i$ as equality, and let $x_{p_i} = f_i(\mathbf{x}^{-p_i})$ be the implied linear function.
06: Let $n' = n$, $m' = m$ and $\mathcal{C}' = \mathcal{C}$, and $\text{flag}_{e_i^j} = 1$
07: For every inequality $e \in D_q$, for $q \neq i$ do
08: Substitute $x_{p_i} = f_i(\mathbf{x}^{-p_i})$ in e
09: If inequality e becomes a tautology, then $\mathcal{C}' \leftarrow \mathcal{C}' \setminus \{D_q\}$, $m' \leftarrow m' - 1$.
10: If inequality e becomes a contradiction, then $D'_q \leftarrow D'_q \setminus \{e\}$.

11: If $D'_q = \emptyset$ then set $\text{flag}_{e_i^j} = 0$, {the substituted I-SAT is infeasible.}
12: enddo
13: If $\text{flag}_{e_i^j} = 1$, then
14: If Projection Algorithm $(n', m', \mathcal{C}') =$ *feasible*[1], then return *feasible*
15: enddo
16: enddo
17: return *infeasible*

For simplicity of presentation, the Projection Algorithm does not output a specific feasible solution. If the I-SAT instance is feasible, then an I-SAT feasible vector can be easily obtained from the inequalities (e_i^j) that lead to feasible reduced I-SAT instances along the recursive iterations (in lines 13 and 14 of the code).

The correctness of the Projection Algorithm follows directly from Remark 3.4.

Theorem 3.6 *The running time $T(m,n)$ of the* Projection Algorithm *is given by equation (1).*

Proof: At any given recursive iteration, the Projection Algorithm reduces the current I-SAT instance to $O(km)$ I-SAT instances with at least one less variable and one less i-clause. To construct each of the reduced I-SAT instances takes $O(kmn)$ steps. Therefore the running time is given by the recurrence: $T(m,n) \leq km(kmn + T(m-1, n-1))$, with base cases $T(m,1) = m(k + \log m)$ and $T(1,n) = O(1)$, the solution of which is given by equation (1). Note that this is dominated by the complexity expression $O((km)^{\min(m,n)-1} m(kn + \log m))$ given in the abstract. ■

Corollary 3.7 *I-SAT is solved in polynomial time when either the number of clauses, m, is fixed or the number of variables, n, is fixed.*

One might think that it is sufficient to check all the $O(k)$ inequalities of *one* i-clause in an I-SAT instance. However the following example shows that this is not the case.

Example 3.1 *Consider the following I-SAT instance:*

$$\bigcap_{i=1}^{r} \left(\bigcup \begin{array}{rcl} y-(x-i) & \leq & 0 \\ y+(x-i) & \geq & 0 \end{array} \right) \bigcap \left(\; y-(x-(r+1)) \;\; \leq \;\; 0 \; \right) \bigcap \left(\; y+(x-(r+1)) \;\; \geq \;\; 0 \; \right)$$

The reader may verify that none of the lines defined by any of the inequalities in the first r i-clauses have an I-SAT feasible point, but the line defined by the inequality of the $(r+1)^{th}$ i-clause is feasible for $y \geq 0$ and line defined by the inequality of the $(r+2)^{th}$ i-clause is feasible for $y \leq 0$.

4 The mold casting problem

Casting is the process by which an object is reproduced through the use of a mold. A molten substance, such as metal or plastic, is poured or forced into a mold and allowed to harden, and finally the parts of the mold are removed. For today's mass production needs, reutilization of the mold is important to maintain the productions costs as low as possible; therefore a major concern in the casting and molding manufacturing process is the need to determine whether a given 3D

[1] where n' is the number of variables in the substituted inequalities.

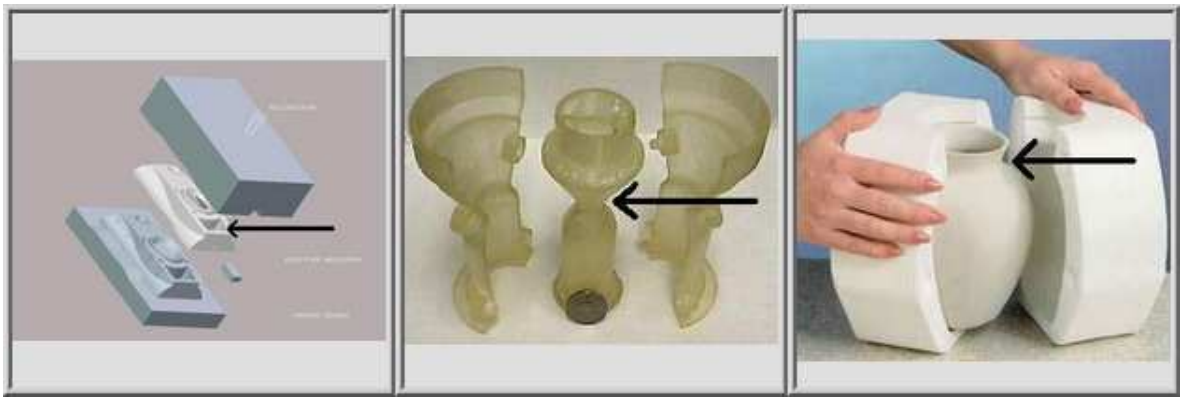

Figure 2: The casting direction is the direction of the cast removal. All of the objects have undercut features (pointed by the arrows) that prevent the direction perpendicular to the cast removal direction to be castable direction.

figure can be manufactured using a reusable two-part mold. For earlier work on this area, and a more extensive introduction see references [9, 5].

To guarantee that a given object can be mass produced using a two-part mold, there must exist a parting direction such that the mold parts can be removed from the object without destroying either the mold or the object. Reusable molds consist on two main halves, which are separated in opposite directions (called *casting directions*) to remove the object [6]. For a parting direction to be feasible, it is necessary that the object being reproduced is free from undercut features relative to the direction (i.e. depressions or protuberances on the boundary of the object), which make impossible the removal of the mold parts into opposite directions without colliding with the object built [1]. In Figure 2 we show three examples of objects manufactured by two-part molds.

Khardekar et al. [7] represent the mold casting problem by enumerating pairs of conflicting faces that are such that directions that go through both of them are infeasible mold parting directions. The conflicting faces are then approximated by triangles. The convex region constructed from the nine vectors connecting the vertices of both triangles, contains all infeasible directions generated by this pair. So the feasible directions must be in the complement of this convex region and must satisfy at least one of the nine inequalities that define the complement of the convex region. Khardekar et al. showed that the problem of finding a feasible direction can be solved by finding values of two variables that satisfy, for each set of nine inequalities, at least one inequality. In other words if there is a nonempty intersection of the complements of the region, then there exists at least one direction that satisfies at least one of the 9 inequalities per set of inequalities. The mold casting problem is thus reduced to an I-SAT problem with $k = 9$ inequalities per i-clause and $n = 2$. Therefore, it follows from Theorem 3.6 that the projection algorithm solves the casting problem in $O(m^2 \log m)$ time.

The most efficient algorithm for the mold casting problem is by Ahn et. al. [1] who used a computational geometry approach to achieve running time of $O(\eta^4)$ for determining whether an object is castable, where η is the number of faces of the object. In terms of the I-SAT problem, the i-clauses correspond to pairs of interacting facets of the object [7]. Therefore $m = O(\eta^2)$ and the running time of the projection algorithm for the mold casting problem is $O(\eta^4 \log \eta)$. This running time is worse by a factor of $\log \eta$ than the algorithm of Ahn et. al. but the I-SAT problem for $n = 2$ and $k = 9$ is more general than the mold casting problem.

Acknowledgements

Special thanks to Professor McMains of the Mechanical Engineering Department at the University of California at Berkeley for taking to our attention the mold casting problem an its relation to the I-SAT problem. We also thank two anonymous referees who helped us to improve the presentation of this paper. The first author was supported in part by the National Science Foundation under award No. DMI-0620677. The second author is grateful to Consejo Nacional de Ciencia y Tecnología (CONACyT), México, for supporting his doctoral studies under grant No. 160619.

References

[1] H. K. Ahn, M. de Berg, P. Bose, S. W. Cheng, D. Halperin, J. Matousek, and O. Schwarzkopf. Separating an object from its cast. *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, 1997.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms.* MIT Press, Cambridge, Mass., 2nd edition, 2001.

[3] S. Even, A. Itai, and A. Shamir. On the complexity of timetable and multicommodity flow problems. *SIAM Journal on Computing*, 5(4):691–703, 1976.

[4] M. L. Fredman and B. Weide. On the complexity of computing the measure of $\bigcup[a_i, b_i]$. *Commun. ACM*, 21(7):540–544, 1978.

[5] M. A. Ganter and P. A Skoglund. Feature extraction for casting core development. *17th Design Automation Conference presented at the 1991 ASME Design Technical Conferences*, 1991.

[6] K. C. Hui and S. T. Tan. Mould design with sweep operations–a heuristic search approach. *Computer-Aided Design*, 24(3):81–91, 1992.

[7] R. Khardekar, G. Burton, and S. McMains. Finding feasible model parting directions using graphics hardware. *ACM symposium on Solid and Physical modeling*, 2005.

[8] A. Schrijver. *Theory of linear and integer programming.* Wiley-Interscience series in discrete mathematics and optimization. Wiley, New-York, 1998.

[9] M. N. Srinivasan and B. Ravi. Decision criteria for computer-aided parting surface design. *Computer-Aided Design*, 22:11–18, 1990.