Consistency for 0–1 Programming

Danial Davarnia[1] and J. N. Hooker[2]

[1] Iowa state University
davarnia@iastate.edu
[2] Carnegie Mellon University
jh38@andrew.cmu.edu

Abstract. Concepts of consistency have long played a key role in constraint programming but never developed in integer programming (IP). Consistency nonetheless plays a role in IP as well. For example, cutting planes can reduce backtracking by achieving various forms of consistency as well as by tightening the linear programming (LP) relaxation. We introduce a type of consistency that is particularly suited for 0-1 programming and develop the associated theory. We define a 0-1 constraint set as LP-consistent when any partial assignment that is consistent with its linear programming relaxation is consistent with the original 0-1 constraint set. We prove basic properties of LP-consistency, including its relationship with Chvátal-Gomory cuts and the integer hull. We show that a weak form of LP-consistency can reduce or eliminate backtracking in a way analogous to k-consistency but is easier to achieve. In so doing, we identify a class of valid inequalities that can be more effective than traditional cutting planes at cutting off infeasible 0-1 partial assignments.

Keywords: Consistency · Resolution · Constraint satisfaction · Integer programming · Backtracking · Cutting planes

1 Introduction

Consistency is a fundamental concept of constraint programming (CP) and an essential tool for the reduction of backtracking during search [1]. Curiously, the concept never explicitly developed in mathematical programming, even though solvers rely on a similar type of branching search. In fact, the cutting planes of integer programming can reduce backtracking by achieving various forms of consistency as well as by tightening the linear programming (LP) relaxation.

This suggests that it may be useful to investigate the potential role of consistency concepts in mathematical programming. We do so for 0–1 integer programming in particular. We study how consistency relates to such integer programming ideas as the LP relaxation, Chvátal-Gomory cutting planes [4], and the integer hull, as well as how consistency can be achieved for 0–1 inequalities. Our main contribution is to introduce a type of consistency, *LP-consistency*, that seems particularly relevant to 0–1 programming, and to develop the underlying theory. We show that achieving a form of partial LP-consistency can reduce backtracking in ways that traditional cutting planes cannot.

One way to reduce backtracking is to identify partial assignments to the variables that are *inconsistent* with the constraint set, meaning that they cannot occur in a feasible solution of the constraints. Branching decisions that result in such partial assignments can then be avoided, thus removing infeasible subtrees from the search. Unfortunately, it is generally hard to identify inconsistent partial assignments in advance.

The essence of consistency is that it makes it easier to identify inconsistent partial assignments. Full consistency allows one to recognize an inconsistent partial assignment by the fact that it violates a constraint that contains only the variables in the partial assignment. Because full consistency is very hard to achieve, CP solvers rely on *domain consistency* (generalized arc consistency) [1,5,11,12], which reduces variable domains to the point that every value in them occurs in some feasible solution. If domain consistency is obtained at the current node of the search tree, branching on any value in a variable's domain can lead to a feasible solution. Domain consistency is itself hard to achieve for the entire constraint set, but can often be achieved, or partially achieved, for individual global constraints in the CP model, and this reduces backtracking significantly [15].

Our approach is based on the idea that consistency can be defined with respect to a *relaxation* of the constraint set. Specifically, we interpret consistency as making it possible to identify inconsistent partial assignments by checking whether they are consistent with a certain type of relaxation. This perspective allows us to propose alternative types of consistency by using various types of relaxation. For traditional consistency, the relaxation is obtained simply by dropping constraints that contain variables that are not in the partial assignment. We define LP-consistency by replacing this relaxation with the LP relaxation. Thus LP-consistency ensures that any partial assignment that is consistent with the LP relaxation is inconsistent with the original constraint set. Fortunately, one can easily check consistency with an LP relaxation simply by solving the LP problem that results from adding the partial assignment to the LP relaxation.

This poses the question of whether it is practical to achieve LP-consistency for a 0–1 problem. There is no known practical method for achieving full LP-consistency, but we take a cue from the concept of k-consistency in CP [6,16,17], which is weaker than full consistency but sufficient to avoid backtracking if the constraints are not too tightly coupled by common variables. While achieving traditional k-consistency is impractical, we define a similar property, *sequential LP k-consistency*, that can be computed for small k. This in turn can avoid some backtracking that traditional cutting planes may permit, because it focuses on identifying inconsistent partial assignments rather than cutting off fractional solutions of the LP relaxation.

A method for obtaining sequential LP k-consistency is suggested by our practice of defining all consistency concepts in terms of projection, as proposed in [10]. One can define sequential LP k-consistency, in particular, in terms of the results of lifting a problem from $k-1$ dimensions to k dimensions, and then projecting it back into $k-1$ dimensions. This same lift-and-project operation is carried out by a special case of the widely-used lift-and-project technique [2], and we show that this procedure obtains sequential LP k-consistency.

We begin below by defining and illustrating basic consistency concepts and showing how they can be cast in terms of projection. We also indicate how consistency can eliminate or reduce backtracking. We review some prior work showing that an inference method of propositional logic, resolution, can achieve consistency for 0–1 problems, and that a weak form of resolution, input resolution, can generate all Chvátal-Gomory cuts for a set of logical clauses.

At this point we introduce LP-consistency and show some elementary properties, namely that consistency implies LP-consistency, and a constraint set that describes the integer hull is necessarily LP-consistent. Yet LP-consistency is a concept that does not occur in polyhedral theory, and an LP-consistent constraint set need not describe the integer hull. While the facet-defining inequalities that describe the integer hull are generally regarded as the strongest valid inequalities, we show that they can be weaker than a non-facet-defining inequality that achieves LP-consistency, in the sense that they exclude fewer inconsistent 0–1 (partial) assignments. We further elaborate on connections with cutting plane theory by showing that a 0–1 partial assignment is consistent with the LP relaxation if and only if it violates no logical clause that is a Chvátal-Gomory (C-G) cut, and a 0–1 problem is LP-consistent if and only if all of its implied logical clauses are C-G cuts. We also note that while input resolution derives C-G cuts, it does not achieve LP-consistency.

The remainder of the paper defines and develops the concept of sequential LP k-consistency. It shows that achieving sequential LP k-consistency for $k = 1, \ldots, n$ (where n is the number of variables) avoids backtracking altogether for branching order $x_1, x_2, \ldots, x_n$. In practice, one would achieve sequential LP k-consistency for a few small values of k. We then prove that a restricted version of the well-known lift-and-project procedure [2] achieves sequential k-consistency for a given k. Finally, we illustrate how achieving sequential LP k-consistency even for $k = 2$ can avoid backtracking that is permitted by traditional separating cuts.

2 Consistency and Projection

To define consistency, it is convenient to adopt basic terminology as follows. The *domain* D_j of a variable x_j is the set of values that can be assigned to x_j. A *constraint* C is an object that *contains* some set $\{x_1, \ldots, x_k\}$ of variables, such that any given assignment of values to $(x_1, \ldots, x_k)$ either *satisfies* or *violates* C. Thus a constraint is satisfied or violated only when all of its variables have been assigned values. An assignment to x satisfies a constraint set $\mathcal{S}$ when it satisfies all the constraints in $\mathcal{S}$. A list of symbols defined hereafter appears in Table 1.

Let x_J be the tuple containing the variables in $\{x_j \mid j \in J\}$ for $J \subseteq N = \{1, \ldots, n\}$. A *partial assignment* to x is an assignment of values to x_J for some $J \subseteq N$. We can now define a consistent partial assignment and a consistent constraint set.

Definition 1. *Given a constraint set $\mathcal{S}$, a partial assignment $x_J = v_J$ is* consistent with $\mathcal{S}$ *if $\mathcal{S} \cup \{x_J = v_J\}$ is feasible.*

Since it is hard in general to determine whether $\mathcal{S} \cup \{x_J = v_J\}$ is feasible, it is hard to identify which partial assignments are consistent with $\mathcal{S}$. Consistent constraint sets are defined so that it is easy to identify which partial assignments are consistent with them.

Definition 2. *A constraint set $\mathcal{S}$ is* consistent *if every partial assignment to x that violates no constraint in $\mathcal{S}$ is consistent with $\mathcal{S}$.*

The contrapositive is perhaps more intuitive: $\mathcal{S}$ is consistent when every partial assignment that is inconsistent with $\mathcal{S}$ violates some individual constraint in $\mathcal{S}$. Thus a consistent constraint set can

Table 1: List of symbols.

x_J	tuple of variables x_j for $j \in J$
$D(\mathcal{S})$	satisfaction set of constraint set $\mathcal{S}$
$D_J(\mathcal{S})$	set of assignments to x_J that are consistent with $\mathcal{S}$
D_j	domain of x_j
$D(\mathcal{S})\|_J$	projection of $D(\mathcal{S})$ onto x_J
$\mathcal{S}_J$	set of constraints in $\mathcal{S}$ that contain only variables in x_J
S_{LP}	LP relaxation of 0–1 constraint set $\mathcal{S}$
$D_J(\mathcal{S}_{\text{LP}})$	set of 0–1 assignments to x_J that are consistent with $\mathcal{S}_{\text{LP}}$
$\mathcal{S}_{\text{C}}$	set of clausal inequalities implied by individual constraints of $\mathcal{S}$
J_k	$\{1, \ldots, k\}$

be viewed as one in which implied constraints are made explicit, in the sense that every inconsistent partial assignment is explicitly ruled out by some constraint in the set.

Since full consistency is generally hard to achieve, the constraint programming community has found various weaker forms of consistency to be more useful. By far the most popular is domain consistency, also known as generalized arc consistency [1,5,11,12].

Definition 3. *A constraint set $\mathcal{S}$ is* domain consistent *if $x_j = v_j$ is consistent with $\mathcal{S}$ for all $v_j \in D_j$ and all variables x_j.*

That is, every value in the domain of a variable x_j is assigned to x_j in some feasible solution of $\mathcal{S}$. A consistent constraint set is necessarily domain consistent.

Example 1. Suppose that $\mathcal{S}$ is the constraint set

$$\begin{array}{l} x_1 + x_2 \qquad\; + x_4 \geq 1 \\ x_1 - x_2 + x_3 \qquad \geq 0 \\ x_1 \qquad\qquad\quad - x_4 \geq 0 \\ x_j \in \{0,1\}, \text{ all } j \end{array}$$

The feasible solutions $(x_1, \ldots, x_4)$ of $\mathcal{S}$ are listed below:

$$\begin{array}{ccc} (0,1,1,0) & (1,0,1,0) & (1,1,0,1) \\ (1,0,0,0) & (1,0,1,1) & (1,1,1,0) \\ (1,0,0,1) & (1,1,0,0) & (1,1,1,1) \end{array}$$

Set $\mathcal{S}$ is not consistent because, for instance, the partial assignment $(x_1, x_2) = (0,0)$ violates no constraint in $\mathcal{S}$ but is inconsistent with $\mathcal{S}$ due to the fact that $(x_1, x_2) = (0,0)$ in none of the feasible solutions. On the other hand, $\mathcal{S}$ is domain consistent because $x_j = 0$ and $x_j = 1$ occur in some feasible solution for each j.

The various consistency concepts are more easily defined in terms of projection, as proposed in [10]. Let $D(\mathcal{S})$ be the satisfaction set of $\mathcal{S}$; that is, the set of assignments to x that satisfy $\mathcal{S}$. Also let $D_J(\mathcal{S})$ be the set of partial assignments $x_j = v_j$ that are consistent with $\mathcal{S}$, so that $D_J(\mathcal{S}) = \{v_J \mid \mathcal{S} \cup \{x_J = v_J\} \text{ is feasible}\}$. The *projection* of $D(\mathcal{S})$ onto x_J, which we may write $D(\mathcal{S})|_J$, is $\{x_J \mid x \in D(\mathcal{S})\}$. Note that the projection is identical to the set of assignments to x_J that are consistent with $\mathcal{S}$, so that $D(S)|_J = D_J(\mathcal{S})$.

This last observation allows us to define consistency in terms of projection. Let $\mathcal{S}_J$ be the set of constraints in $\mathcal{S}$ whose variables belong to x_J. Then $D_J(\mathcal{S}_J)$ is the set of assignments to x_J that violate no constraints in $\mathcal{S}$. We assume that $\mathcal{S}$ contains the *in-domain constraints* $x_j \in D_j$ for all $j \in N$.

Proposition 1. *A constraint set $\mathcal{S}$ is consistent if and only if $D_J(\mathcal{S}_J) = D(\mathcal{S})|_J$ for all $J \subseteq N$. Equivalently, $\mathcal{S}$ is consistent if and only if all 0–1 partial assignments $x_J = v_J$ that are consistent with $\mathcal{S}_J$ are consistent with $\mathcal{S}$. In addition, $\mathcal{S}$ is domain consistent if and only if $D_j = D(\mathcal{S})|_{\{j\}}$ for all $j \in N$.*

Example 2. If $\mathcal{S}$ is as in Example 1, $\mathcal{S}$ is not consistent because, for instance, the satisfaction set $D_{\{1,2\}}(\mathcal{S}_{\{1,2\}}) = \{(0,0),(0,1),(1,0),(1,1)\}$ of $\mathcal{S}_{\{1,2\}} = \{x_1 \in \{0,1\},\ x_2 \in \{0,1\}\}$ is different from the projection onto (x_1, x_2) of $D(\mathcal{S})$, which is $D(\mathcal{S})|_{\{1,2\}} = \{(0,1),(1,0),(1,1)\}$. However, $\mathcal{S}$ is domain consistent because $D_j = \{0,1\} = D(\mathcal{S})_{\{j\}}$ for all j.

Consistency can be understood as defined with respect to a relaxation of $\mathcal{S}$. For classical consistency, the relaxation is $\mathcal{S}_J$, obtained by omitting constraints from $\mathcal{S}$. We will later define consistency with respect to the linear programming relaxation of $\mathcal{S}$.

3 Consistency and Backtracking

It is well known that consistency is closely related to backtracking. We note first that branching can find a feasible solution for a fully consistent constraint set without backtracking, assuming of course that the constraints have a solution. Suppose we branch on variables $x_1, \ldots, x_n$ in that order. Each node in level j of the branching tree corresponds to a partial assignment $(x_1, \ldots, x_{j-1}) = (v_1, \ldots, v_{j-1})$. We branch on x_j at the node by assigning to x_j each value $v_j \in D_j$ for which the partial assignment $(x_1, \ldots, x_j) = (v_1, \ldots, v_j)$ violates no constraint in $\mathcal{S}$. Due to the consistency of $\mathcal{S}$, this partial assignment is consistent with $\mathcal{S}$ for at least one value $v_j \in D_j$. Thus branching can continue to the bottom of the tree with no need to backtrack.

A weaker form of consistency avoids backtracking if there is limited coupling of variables. Let the directed *dependency graph* G of $\mathcal{S}$ for the ordering $1, \ldots, n$ consist of vertices corresponding to variables x_j and directed edges (x_i, x_j) whenever $i < j$, and x_i and x_j occur in a common constraint of $\mathcal{S}$. The *width* of G for the ordering $1, \ldots, n$ is the maximum out-degree of the nodes of G.

Definition 4. *A constraint set $\mathcal{S}$ is* k-consistent *if $D_J(\mathcal{S}_J) = D_{J\cup\{j\}}(\mathcal{S}_{J\cup\{j]})|_J$ for all $J \subseteq N$ with $|J| = k - 1$ and all $j \in N \setminus J$. $\mathcal{S}$ is* strongly k-consistent *if it is j-consistent for $j = 1, \ldots, k$.*

The following is proved in [7].

Proposition 2. *Let G be the directed dependency graph of constraint set $\mathcal{S}$ for the ordering $1, \ldots, n$. Then if the branching order is $x_1, \ldots, x_n$, $\mathcal{S}$ can be solved without backtracking if $\mathcal{S}$ is strongly k-consistent and G has width less than k.*

A still weaker form of consistency avoids backtracking if the branching order is given. It is not necessary to consider all sets J and all indices $j \notin J$, but only variables on which we have branched. We therefore define a form of k-consistency that assumes the branching order is $x_1, \ldots, x_n$. Let $J_k = \{1, \ldots, k\}$.

Definition 5. *A 0–1 constraint set $\mathcal{S}$ is* sequentially k-consistent if $D_{J_{k-1}}(\mathcal{S}_{J_{k-1}}) = D_{J_k}(\mathcal{S}_{J_k})|_{J_{k-1}}$.

Thus $\mathcal{S}$ is sequentially k-consistent if for every partial assignment $(x_1, \ldots, x_{k-1}) = (v_1, \ldots, v_{k-1})$ that violates no constraint in S, there is a value v_k in D_k such that $(x_1, \ldots, x_k) = (v_1, \ldots, v_k)$ violates no constraint in $\mathcal{S}$. The following is easy to show.

Proposition 3. *If the branching order is $x_1, \ldots, x_n$, constraint set $\mathcal{S}$ can be solved without backtracking if $\mathcal{S}$ is sequentially k-consistent for $k = 1, \ldots, n$.*

Example 3. Let $\mathcal{S} = \{3x_1 + 2x_2 \geq 1,\ -x_1 + 2x_2 \geq 0,\ x \in \{0,1\}^2\}$. Proposition 3 implies that we can avoid backtracking by branching in the order x_1, x_2, because $\mathcal{S}$ is sequentially 1-consistent and sequentially 2-consistent. The lack of backtracking does not follow from Proposition 2, however, because $\mathcal{S}$ is not 2-consistent, and its dependency graph has width 1 for the ordering 1,2. $\mathcal{S}$ is not 2-consistent because the partial assignment $x_2 = 0$ violates no constraints and has no extension to a consistent assignment $(x_1, x_2) = (v_1, 0)$.

Even domain consistency suffices to avoid backtracking if it is achieved at every node. Supposing that a given node corresponds to a partial assignment as above, let $\mathcal{S}' = \mathcal{S} \cup \{(x_1, \ldots, x_{j-1}) = (v_1, \ldots, v_{j-1})\}$. Then if $\mathcal{S}'$ is domain consistent, x_j can be assigned any value in its domain to obtain an assignment $(x_1, \ldots, x_j) = (v_1, \ldots, v_j)$ that is consistent with $\mathcal{S}'$. The process can continue without backtracking if domain consistency is similarly achieved at subsequent nodes.

4 Consistency and Resolution

Previous research has shown that the resolution procedure of propositional logic achieves consistency for a 0–1 constraint set.

First, some definitions. A *literal* ℓ_j is a proposition of the form x_j or $\neg x_j$. A logical *clause* is a disjunction $\bigvee_{j\in J} \ell_j$ of literals, which we denote by $\ell(J)$, where J is possibly empty. Given clauses $\ell(J)$ and $\ell(J')$, the former *absorbs* the latter if $J \subseteq J'$. For example, $x_1 \vee \neg x_2$ absorbs $x_1 \vee \neg x_2 \vee x_3$. One clause logically implies another if and only if the one absorbs the other.

Given clauses $\ell(J_1) \vee x_k$ and $\ell(J_2) \vee \neg x_k$, where $k \notin J_1 \cup J_2$, the *resolvent* of the clauses is $\ell(J_1 \cup J_2)$. For example, the resolvent of $x_1 \vee x_2 \vee x_4$ and $x_1 \vee \neg x_3 \vee \neg x_4$ is $x_1 \vee x_2 \vee \neg x_3$, while the clauses $x_1 \vee \neg x_2$ and $\neg x_1 \vee x_2$ do not have a resolvent. A resolvent is logically implied by the conjunction of its two parents but is absorbed by neither.

Given a clause set $\mathcal{C}$, a *resolution proof* of clause C_m from $\mathcal{C}$ is a sequence of clauses $C_1, \ldots, C_m$ such that C_m is the resolvent of two previous clauses in the sequence, and each C_i for $i = 1, \ldots, m-1$ either belongs to $\mathcal{C}$ or is the resolvent of two earlier clauses in the sequence. It can be assumed that a resolvent is not generated when it is absorbed by a previous clause in the sequence. An *input proof* of C_m is a resolution proof in which at least one of the parents of each resolvent belongs to $\mathcal{C}$ [3]. There is a resolution proof of any clause that is logically implied by $\mathcal{C}$ [13,14], but not necessarily an input proof.

A 0–1 constraint set $\mathcal{S}$ *logically implies* 0–1 constraint set $\mathcal{S}'$ when all 0–1 points that satisfy $\mathcal{S}$ also satisfy $\mathcal{S}'$. $\mathcal{S}$ and $\mathcal{S}'$ are *logically equivalent* when they logically imply each other. A logical clause

$$\bigvee_{j\in J^+} x_j \vee \bigvee_{j\in J^-} \neg x_j$$

is *represented* by the 0–1 inequality

$$\sum_{j\in J^+} x_j + \sum_{j\in J^-} (1 - x_j) \geq 1$$

A 0–1 inequality is *clausal* when it represents a clause. It is clear that a 0–1 inequality is logically equivalent to the set of clausal inequalities it implies. Thus if we let $\mathcal{S}_\mathrm{C}$ be the set of clausal inequalities that are implied by some inequality in $\mathcal{S}$, then $\mathcal{S}$ is logically equivalent to $\mathcal{S}_\mathrm{C}$. It is convenient to say that a clausal inequality has a resolution proof from $\mathcal{S}_\mathrm{C}$ if the clause it represents has a resolution proof from the clauses represented by $\mathcal{S}_\mathrm{C}$. It is shown in [9] that resolution on clausal inequalities achieves consistency.

Proposition 4. *If 0–1 constraint set $\mathcal{S}$ is augmented with all clausal inequalities that have resolution proofs from $\mathcal{S}_\mathrm{C}$, the resulting constraint set is consistent.*

Example 4. If $\mathcal{S}$ is the constraint set of Example 3, $\mathcal{S}_\mathrm{C}$ contains the clausal inequalities $x_1 + x_2 \geq 1$ and $-x_1 + x_2 \geq 0$. One clausal inequality, namely $x_2 \geq 1$, has a resolution proof from $\mathcal{S}_\mathrm{C}$, and adding this inequality to $\mathcal{S}$ yields a consistent constraint set.

A second example illustrates how a traditional cutting plane can serve the dual purpose of tightening the linear programming (LP) relaxation and achieving consistency. Let the LP relaxation of $\mathcal{S} = \{Ax \geq b,\ x \in \{0,1\}^n\}$ be $\mathcal{S}_{\mathrm{LP}} = \{Ax \geq b,\ x \in [0,1]^n\}$.

Example 5. Suppose that $\mathcal{S}$ is the constraint set of Example 1. In this case, $\mathcal{S}$ and $\mathcal{S}_{\mathrm{C}}$ are identical. Resolution yields two additional clausal inequalities, $x_1+x_2 \geq 1$ and $x_1+x_3 \geq 1$. By Proposition 4, adding these inequalities to $\mathcal{S}$ achieves consistency. These inequalities are also traditional cutting planes for $\mathcal{S}$, in particular Chvátal-Gomory (C-G) cuts. The first cuts off two fractional vertices $(x_1, \ldots, x_4) = (\frac{1}{3}, \frac{1}{3}, 0, \frac{1}{3}), (\frac{1}{2}, 0, 0, \frac{1}{2})$ of the polytope described by $\mathcal{S}_{\mathrm{LP}}$, and the second cuts off $(\frac{1}{2}, \frac{1}{2}, 0, 0)$ as well. The inequalities therefore serve the dual purpose of achieving consistency and tightening the LP relaxation. As it happens, adding both resolvents yields an integral polytope, but we will see that a consistent constraint does not in general describe an integral polytope.

Input proofs from $\mathcal{S}_{\mathrm{C}}$ do not necessarily achieve consistency, but they derive all clausal C–G cuts for $\mathcal{S}_{\mathrm{C}}$. The following is proved in [8].

Proposition 5. *Given a 0–1 constraint set $\mathcal{S}$, a clausal inequality is a C-G cut for $\mathcal{S}_{\mathrm{C}} \cup \{x \in [0,1]\}$ if and only if it has an input proof from $\mathcal{S}_{\mathrm{C}}$.*

5 LP-consistency

While resolution can always achieve consistency, it is not a practical method for the reduction of backtracking. Resolution proofs tend to explode rapidly in length and complexity. However, the LP relaxation of $\mathcal{S}$ provides an additional tool for this purpose. Specifically, it provides a more useful test for consistency than whether a partial assignment violates a constraint.

Consistency of $\mathcal{S}$ implies that any partial assignment $x_J = v_J$ that is consistent with $\mathcal{S}_J$ (i.e., violates no constraint in $\mathcal{S}$) is consistent with $\mathcal{S}$. We want a type of consistency that ensures that any partial assignment consistent with $\mathcal{S}_{\mathrm{LP}}$ is consistent with $\mathcal{S}$. We can achieve this by defining consistency with respect to the LP relaxation $\mathcal{S}_{\mathrm{LP}}$ rather than the relaxation $\mathcal{S}_J$. Recall that classical consistency is defined so that $D_J(\mathcal{S}_J) = D(S)|_J$. We therefore define *LP-consistency* as follows.

Definition 6. *A 0–1 constraint set $\mathcal{S}$ is* LP-consistent *if $D_J(\mathcal{S}_{\mathrm{LP}}) \cap \{0,1\}^{|J|} = D(\mathcal{S})|_J$ for all $J \subseteq N$.*

Here, $D_J(\mathcal{S}_{\mathrm{LP}})$ refers to the set of *0–1 assignments* to x_J that are consistent with $\mathcal{S}_{\mathrm{LP}}$. Thus $\mathcal{S}$ is *LP-consistent* if $\mathcal{S}_{\mathrm{LP}} \cup \{x_J = v_J\}$ is infeasible for any 0–1 partial assignment $x_J = v_J$ that is inconsistent with $\mathcal{S}$.

Example 6. Consider the 0–1 constraint set $\mathcal{S} = \{2x_1 + 4x_2 \geq -1,\ 2x_1 - 4x_2 \geq -3,\ x \in \{0,1\}^2\}$ (Fig. 1). The partial assignment $x_1 = 0$ is consistent with $\mathcal{S}_{\mathrm{LP}}$ but not with $\mathcal{S}$, because both $(x_1, x_2) = (0,0)$ and $(x_1, x_2) = (0,1)$ violate $\mathcal{S}$. So $\mathcal{S}$ is not LP-consistent.

Two elementary properties of LP-consistency follow.

Proposition 6. *A consistent 0–1 constraint set is LP-consistent.*

Proof. Consider any 0–1 partial assignment $x_J = v_J$ that is consistent with S_{LP}. We claim that $x_J = v_J$ is consistent with $\mathcal{S}$, which suffices to show that $\mathcal{S}$ is LP-consistent. Since $S_{\mathrm{LP}} \cup \{x_J = v_J\}$ is feasible, $x_J = v_J$ violates no constraints in $\mathcal{S}$. Now since $\mathcal{S}$ is consistent, this means that $x_J = v_J$ is consistent with $\mathcal{S}$, as claimed. □

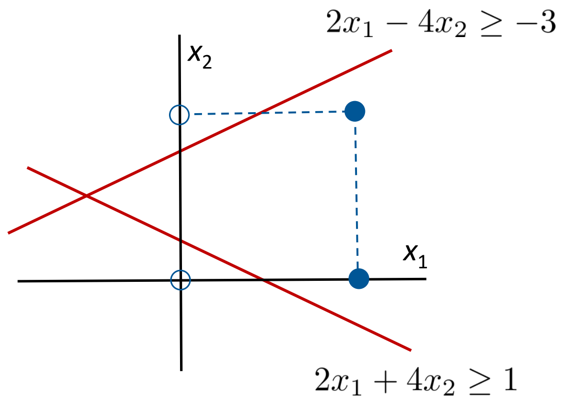

Fig. 1: Illustration of Example 6.

In addition, a 0–1 constraint set that describes the integer hull (the convex hull of feasible 0–1 points) is LP-consistent.

Proposition 7. *Given 0–1 constraint set $\mathcal{S}$, if $\mathcal{S}_{\text{LP}}$ describes the integer hull of $D(\mathcal{S})$, then $\mathcal{S}$ is LP-consistent.*

Proof. Suppose that $\mathcal{S} \cup \{x_J = v_J\}$ is infeasible for a given 0–1 partial assignment $x_J = v_J$. Then $x_J = v_J$ describes a face of the unit hypercube that is disjoint from $D(\mathcal{S})$. This implies that the face is disjoint from the convex hull of $D(\mathcal{S})$, which is described by $\mathcal{S}_{\text{LP}}$. Thus $\mathcal{S}_{\text{LP}} \cup \{x_J = v_J\}$ is infeasible, and it follows that $\mathcal{S}$ is LP-consistent. $\square$

It is essential to observe that a convex hull model is not necessary to achieve LP-consistency, a fact that will be exploited in later sections. This can be seen in an example.

Example 7. Consider the following two constraint sets (Fig. 2), which have the same feasible set:

$$\mathcal{S}^1 = \{x_1 + x_2 \leq 1,\ x_2 + x_3 \leq 1,\ x \in \{0,1\}^3\}$$
$$\mathcal{S}^2 = \{x_1 + 2x_2 + x_3 \leq 2,\ x \in \{0,1\}^3\}$$

The LP relaxation $\mathcal{S}^1_{\text{LP}}$ describes the integer hull of $D(\mathcal{S}^1) = D(\mathcal{S}^2)$, and so $\mathcal{S}^1$ is LP-consistent by Proposition 7. Yet the constraint set $\mathcal{S}^2$ is also LP-consistent, even though $\mathcal{S}^2_{\text{LP}}$ does not describe the convex hull, but describes a polytope with fractional extreme points $(x_1, x_2, x_3) = (0, \frac{1}{2}, 1), (1, \frac{1}{2}, 0)$. Interestingly, the inequality $x_1 + 2x_2 + x_3 \geq 2$ in $\mathcal{S}^2$ is the sum of the two nontrivial facet-defining inequalities in $\mathcal{S}^1$ and is therefore weaker than either of them from a polyhedral point of view. Yet it cuts off more infeasible 0–1 points than either of the facet-defining inequalities and is therefore stronger in this sense. Indeed, the purpose of achieving LP-consistency is to cut off infeasible 0–1 (partial) assignments, not to cut off fractional vertices of the LP relaxation.

6 Characterizing LP-Consistency

The following result gives a necessary condition for consistency based on clausal inequalities.

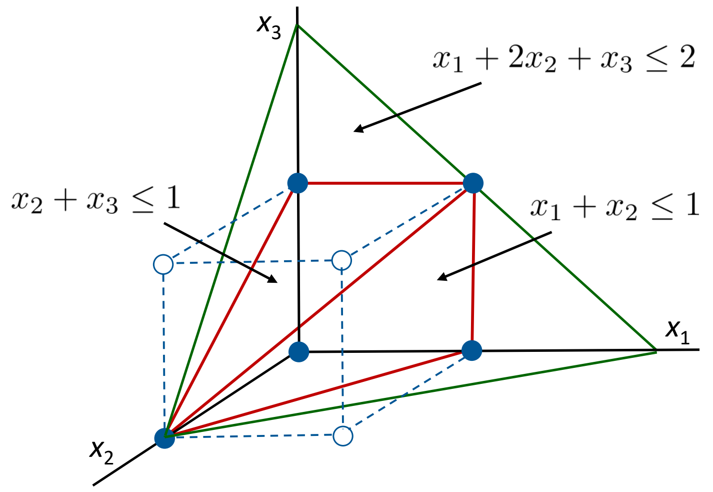

Fig. 2: Illustration of Example 7

Proposition 8. *If a constraint set $\mathcal{S}$ is consistent, then all of its implied clausal inequalities are in $\mathcal{S}_{\mathrm{C}}$.*

Proof. Suppose that $\mathcal{S}$ is consistent, and let C be any clausal inequality implied by $\mathcal{S}$. Then the assignment $x_J = v_J$ violates C, where x_J are the variables in C and v_j is 1 when x_j is negated in C and 0 otherwise. This means $x_J = v_J$ is inconsistent with $\mathcal{S}$, which implies by the consistency of $\mathcal{S}$ that $x_J = v_J$ violates an inequality $\alpha x \geq \beta$ in $\mathcal{S}$. As a result, C must be implied by $\alpha x \geq \beta$, showing that $C \in \mathcal{S}_{\mathrm{C}}$. □

LP-consistency allows us to derive a stronger argument on the relation between an LP-consistent set and its implied clausal inequalities, as it provides both necessary and sufficient conditions. In particular, a 0–1 constraint set $\mathcal{S}$ is LP-consistent if and only if all of its implied clauses are C-G cuts for $\mathcal{S}_{\mathrm{LP}}$. This is due to the following fact.

Proposition 9. *Given a 0–1 constraint set $\mathcal{S}$, a 0–1 partial assignment is consistent with $\mathcal{S}_{\mathrm{LP}}$ if and only if the assignment violates no clausal C-G cut for $\mathcal{S}_{\mathrm{LP}}$.*

Proof. It suffices to show that a given 0–1 partial assignment $x_J = v_J$ violates a clausal C-G for $\mathcal{S}_{\mathrm{LP}}$ if and only if $\mathcal{S}_{\mathrm{LP}} \cup \{x_J = v_J\}$ is infeasible. Suppose first that $x_J = v_J$ violates a clausal inequality $ax \geq \beta$ that is a C-G cut for $\mathcal{S}_{\mathrm{LP}}$, where S_{LP} is the system $Ax \geq b$. Since $x_J = v_J$ violates $ax \geq \beta$, we can write the inequality as $a_J x_J \geq \beta$, where $a_J v_J \leq \beta - 1$. Now since $ax \geq \beta$ is a C–G cut, there is a tuple $u \geq 0$ of multipliers such that $uA = a$ and $\beta - 1 < ub \leq \beta$. We therefore have $(uA)_J v_J = a_J v_J \leq \beta - 1 < ub$. This implies that $x_J = v_J$ violates $uAx \geq ub$, and so $\mathcal{S}_{\mathrm{LP}} \cup \{x_J = v_J\}$ must be infeasible.

For the converse, suppose that $\mathcal{S}_{\mathrm{LP}} \cup \{x_J = v_J\}$ is infeasible, which means that the face of the unit hypercube defined by $x_J = v_J$ lies outside the polytope defined by S_{LP}. Let $J^+ = \{j \in J \mid v_j = 0\}$ and $J^- = \{j \in J \mid v_j = 1\}$. Then some inequality of the form $\sum_{j \in J^+} x_j + \sum_{j \in J^-} (1 - x_j) \geq \bar{\pi}$ for some $\bar{\pi} > 0$ separates the face just mentioned from the polytope; i.e., $x_J = v_J$ violates this inequality. Since this inequality is valid for $\mathcal{S}_{\text{LP}}$, it is dominated by some surrogate of $Ax \ge b$. That is there exists a tuple $u \ge 0$ of multipliers such that $uA \ge ub$ is of the form

$$\sum_{j \in J^+} x_j + \sum_{j \in J^-} (1 - x_j) \ge \pi \tag{1}$$

where $\pi \ge \bar{\pi}$, and $\pi \le |J|$ because $\mathcal{S}$ is feasible. Now pick any subset $\hat{J} \subseteq J$ with $|\hat{J}| = \lceil \pi \rceil - 1$, let $\hat{J}^+ = J^+ \cap \hat{J}$, and let $\hat{J}^- = J^- \cap \hat{J}$. Take the sum of (1) with $-x_j \ge -1$ for $j \in \hat{J}^+$ and $x_j \ge 0$ for $j \in \hat{J}^-$. This yields a clausal inequality that is a surrogate of $Ax \ge b$:

$$\sum_{j \in J^+ \setminus \hat{J}^+} x_j + \sum_{j \in J^- \setminus \hat{J}^-} (1 - x_j) \ge 1 + \pi - \lceil \pi \rceil$$

Rounding up the right-hand side (if necessary) yields a clausal C–G cut violated by $x_J = v_J$. Thus $x_J = v_J$ violates a clausal C-G cut for $\mathcal{S}_{\text{LP}}$, as claimed. □

Example 8. Consider again the constraint set $\mathcal{S}$ of Example 5. The partial assignment $(x_1, x_3) = (0, 0)$ is inconsistent with $\mathcal{S}_{\text{LP}}$ and violates a clausal C-G cut, namely $x_1 + x_3 \ge 1$. The cut is obtained by assigning multipliers $\frac{1}{4}, \frac{1}{2}, \frac{1}{4}, \frac{1}{4}, \frac{1}{2}$ to the three constraints of $\mathcal{S}$, $x_2 \ge 0$, and $x_3 \ge 0$, respectively. The partial assignment $(x_1, x_3) = (0, 1)$ is consistent with $\mathcal{S}_{\text{LP}}$ and therefore violates no clausal C-G cut.

Corollary 1. *A constraint set $\mathcal{S}$ is LP-consistent if and only if all of its implied clausal inequalities are C-G cuts for $\mathcal{S}_{\text{LP}}$.*

Proof. Suppose first that $\mathcal{S}$ is LP-consistent, and let C be any clausal inequality implied by $\mathcal{S}$. Then the assignment $x_J = v_J$ violates C, where x_J are the variables in C and v_j is 1 when x_j is negated in C and 0 otherwise. This means $x_J = v_J$ is inconsistent with $\mathcal{S}$, which implies by the LP-consistency of $\mathcal{S}$ that $x_J = v_J$ is inconsistent with $\mathcal{S}_{\text{LP}}$. By Proposition 9, $x_J = v_J$ violates some clausal C-G cut C' of $\mathcal{S}_{\text{LP}}$. Then C' must absorb C, which means C is likewise a C-G cut of $\mathcal{S}_{\text{LP}}$.

Conversely, suppose all clausal inequalities implied by $\mathcal{S}$ are C-G cuts for $\mathcal{S}_{\text{LP}}$, and consider any partial assignment $x_J = v_J$ that is consistent with $\mathcal{S}_{\text{LP}}$. By Proposition 9, $x_J = v_J$ violates no clausal C-G cut of $\mathcal{S}_{\text{LP}}$. This means that it violates no clause implied by $\mathcal{S}$, which implies that $x_J = v_J$ is consistent with $\mathcal{S}$, as desired. □

Example 9. The constraint set $\mathcal{S}$ of Example 1 is LP-consistent because its implied clausal inequalities are all absorbed by the inequalities in $\mathcal{S} \cup \{x_1 + x_2 \ge 1,\ x_1 + x_3 \ge 1\}$, and these are all C-G cuts for $\mathcal{S}_{\text{LP}}$.

7 LP-consistency and Resolution

We have seen that full resolution achieves consistency for a 0–1 constraint set $\mathcal{S}$. That is, $\mathcal{S}$ is consistent if it contains all clausal inequalities that have resolution proofs from $\mathcal{S}_{\text{C}}$. Full resolution therefore achieves LP-consistency, since any consistent constraint set is LP-consistent (Proposition 6). However, it is generally unnecessary to apply full resolution to achieve LP-consistency, and it is unclear what kind of resolution is necessary and sufficient for this purpose.

Yet one can go some distance toward achieving LP-consistency by generating all clausal inequalities that have input proofs from $\mathcal{S}_{\text{C}}$.

Example 10. Let $\mathcal{S}$ consist of the following constraints, in addition to $x \in \{0,1\}$:

$$\begin{array}{ll} x_1+x_2+x_3+x_4 \geq 1 & x_1-x_2+x_3+x_4 \geq 0 \\ x_1+x_2+x_3-x_4 \geq 0 & x_1-x_2+x_3-x_4 \geq -1 \\ x_1+x_2-x_3+x_4 \geq 0 & x_1-x_2-x_3+x_4 \geq -1 \\ x_1+x_2-x_3-x_4 \geq -1 & x_1-x_2-x_3-x_4 \geq -2 \end{array}$$

In this case, $\mathcal{S}_\text{C} = \mathcal{S}$. Input proofs from $\mathcal{S}_\text{C}$ yield the inequalities

$$\begin{array}{lll} x_1+x_2+x_3 \geq 1 & x_1+x_2+x_4 \geq 1 & x_1+x_3+x_4 \geq 1 \\ x_1+x_2-x_3 \geq 0 & x_1+x_2-x_4 \geq 0 & x_1+x_3-x_4 \geq 0 \\ x_1-x_2+x_3 \geq 0 & x_1-x_2+x_4 \geq 0 & x_1-x_3+x_4 \geq 0 \\ x_1-x_2-x_3 \geq -1 & x_1-x_2-x_4 \geq -1 & x_1-x_3-x_4 \geq -1 \end{array}$$

Adding these inequalities to $\mathcal{S}_\text{C}$ to obtain $\mathcal{S}'$ does not achieve LP-consistency. The partial assignment $x_1 = 0$ is inconsistent with $\mathcal{S}$ but consistent with $\mathcal{S}'_\text{LP}$, where the latter is due to the fact that $(x_1, \ldots, x_4) = (0, \frac{1}{2}, \frac{1}{2}, \frac{1}{2})$ satisfies $\mathcal{S}'_\text{LP}$. Yet the input proofs are not useless because they eliminate some partial assignments that are inconsistent with $\mathcal{S}$, such as $(x_1, x_2) = (0, 0)$, which is consistent with $\mathcal{S}_\text{C}$ but not with $\mathcal{S}'_\text{LP}$.

On the other hand, it may be possible to achieve LP-consistency without adding all clauses that have input proofs from $\mathcal{S}_\text{C}$, and even without adding all the clauses in $\mathcal{S}_\text{C}$.

Example 11. Let $\mathcal{S}$ be the constraint set of Example 6, where $\mathcal{S}_\text{C} = \{x_1+x_2 \geq 1,\ x_1-x_2 \geq 0\}$. The clausal inequality $x_1 \geq 1$ has an input proof from $\mathcal{S}_\text{C}$, but $\mathcal{S}_\text{C}$ is LP-consistent without adding this inequality. In fact, we can achieve LP-consistency for $\mathcal{S}$ even without adding to it all the clausal inequalities in $\mathcal{S}_\text{C}$. For example, if we add only $x_1 + x_2 \geq 1$ to $\mathcal{S}$, the resulting constraint set is LP-consistent (Fig. 3).

These examples suggest that resolution is not a natural technique for achieving LP-consistency or even partial LP-consistency. However, we will see in Section 9 that a technique borrowed from integer programming can achieve partial LP-consistency in a reasonably practical way.

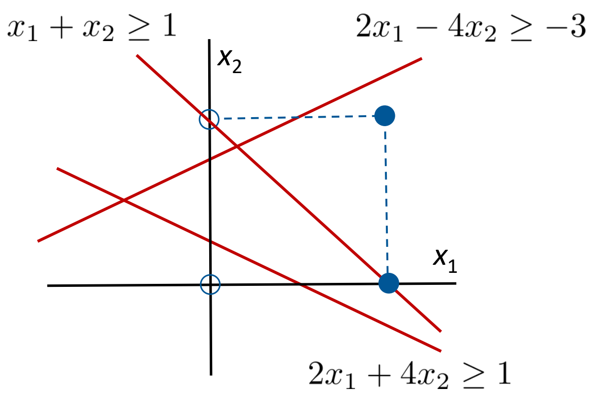

Fig. 3: Illustration of Example 11.

8 LP-Consistency and Backtracking

Like full consistency in CP, full LP-consistency is difficult to achieve. We therefore follow the lead of the CP community and consider weaker forms of consistency. One that seems appropriate to 0–1 programming is inspired by k-consistency. While even k-consistency is hard to achieve in practice, and the CP community focuses on domain consistency instead, a form of LP-consistency analogous to sequential k-consistency may be practical for 0–1 programming.

Recall that $\mathcal{S}$ is sequentially k-consistent if $D_{J_{k-1}}(\mathcal{S}_{J_{k-1}}) = D_{J_k}(\mathcal{S}_{J_k})|_{J_{k-1}}$, and that sequential k-consistency for $k = 1, \ldots, n$ suffices to avoid backtracking when the branching order is $x_1, \ldots, x_n$. A parallel definition that relates to linear programming is as follows.

Definition 7. *A 0–1 constraint set $\mathcal{S}$ is* sequentially LP k-consistent *if $D_{J_{k-1}}(\mathcal{S}_{\text{LP}}) = D_{J_k}(\mathcal{S}_{\text{LP}})|_{J_{k-1}}$.*

Equivalently, we can say that $\mathcal{S}$ is sequentially LP k-consistent if for every 0–1 partial assignment $x_{J_{k-1}} = v_{J_{k-1}}$ that is consistent with $\mathcal{S}_{\text{LP}}$, there is a 0–1 assignment $x_k = v_k$ for which $x_{J_k} = x_{J_k}$ is consistent with $\mathcal{S}_{\text{LP}}$. Thus sequential LP k-consistency is analogous to sequential k-consistency but based on the $\mathcal{S}_{\text{LP}}$ relaxation rather than the $S_{J_{k-1}}$ relaxation.

This form of consistency can also allow us to avoid backtracking, if we are willing to solve appropriate LP problems. Specifically, suppose that at a given node in the branching tree, prior branching has fixed $(x_1, \ldots, x_{k-1}) = (v_1, \ldots, v_{k-1})$. For the next branch, we select a value $v_k \in \{0, 1\}$ for which the partial assignment $(x_1, \ldots, x_k) = (v_1, \ldots, v_k)$ is consistent with $\mathcal{S}_{\text{LP}}$; that is, for which the LP problem $S_{\text{LP}} \cup \{(x_1, \ldots, x_k) = (v_1, \ldots, v_k)\}$ is feasible. We then set $x_k = v_k$ and continue to the next level of the tree. The following theorem guarantees that the LP problem will be feasible for at least one value of v_k, and that this process avoids backtracking.

Proposition 10. *If $\mathcal{S}$ is a feasible 0–1 constraint set over x and the branching order is $x_1, \ldots, x_n$, achieving sequential LP k-consistency for $k = 1, \ldots, n$ suffices to solve $\mathcal{S}$ without backtracking.*

Proof. Since $\mathcal{S}$ is feasible, S_{LP} is feasible at the root node of the branching tree, and so the empty assignment is consistent with $\mathcal{S}_{\text{LP}}$. Arguing inductively, suppose the partial assignment $(x_1, \ldots, x_{k-1}) = (v_1, \ldots, v_{k-1})$ that reflects the branching decisions down to the node at level k is consistent with $\mathcal{S}_{\text{LP}}$. Since $\mathcal{S}$ is sequentially LP k-consistent, there exists a 0–1 value v_k of x_k for which the partial assignment $(x_1, \ldots, x_k) = (v_1, \ldots, v_k)$ is consistent with $\mathcal{S}_{\text{LP}}$. By induction, $\mathcal{S}_{\text{LP}} \cup \{(x_1, \ldots, x_n) = (v_1, \ldots, v_n)\}$ is feasible at the terminal node of the tree for some tuple $(v_1, \ldots, v_n)$ of 0–1 values. But in this case, $(x_1, \ldots, x_n) = (v_1, \ldots, v_n)$ satisfies $\mathcal{S}$, and we have solved the problem without backtracking. $\square$

Example 12. Consider the constraint set $\mathcal{S}$ of Example 6. $\mathcal{S}$ is not sequentially LP 2-consistent because $x_1 = 0$ is consistent with $\mathcal{S}_{\text{LP}}$, but neither $(x_1, x_2) = (0, 0)$ nor $(x_1, x_2) = (0, 1)$ is consistent with $\mathcal{S}_{\text{LP}}$. Also, backtracking is possible, because if we set $x_1 = 0$ at the root node because $x_1 = 0$ is consistent with $\mathcal{S}_{\text{LP}}$, we cannot find a consistent value for x_2 at the child node and must backtrack. Now suppose we add the clause $x_1 + x_2 \geq 1$ to $\mathcal{S}$ to obtain a constraint set $\mathcal{S}'$ that is sequentially LP 2-consistent (Fig. 3). At the root node we must branch on $x_1 = 1$, because $x_1 = 0$ is not consistent with $\mathcal{S}'_{\text{LP}}$. At the child node, branching on $x_2 = 1$ yields an assignment $(x_1, x_2) = (1, 1)$ that is consistent with $\mathcal{S}_{\text{LP}}$ and, in fact, solves $\mathcal{S}$ without backtracking.

9 Achieving LP Consistency

The definition of sequential LP k-consistency, Definition 7, already suggests a method for achieving it. Namely, we wish to obtain the results of lifting the problem from $k-1$ dimensions to k dimensions, and then projecting back onto $k-1$ dimensions. This operation can be achieved by using basics of the lift-and-project method of [2] as defined next.

To achieve sequential LP k-consistency, we proceed as follows. Given $\mathcal{S} = \{Ax \geq b,\ x \in \{0,1\}^n\}$ where $0 \leq x_i \leq 1$ is included in $Ax \geq b$, we generate the nonlinear system

$$\begin{aligned}(Ax-b)x_k &\geq 0\\ (Ax-b)(1-x_k) &\geq 0\end{aligned}$$

We next linearize the system by replacing each x_k^2 with x_k, and each product $x_i x_k$ with $y_{\{i,k\}}$. Let the resulting system be $R_k(\mathcal{S}_{\text{LP}})$. Finally, project this system onto x_J to obtain the system $R_k(\mathcal{S}_{\text{LP}})|_{J_{k-1}}$.

Proposition 11. *Given a 0–1 constraint set $\mathcal{S}$, applying the above algorithm and augmenting $\mathcal{S}$ with the constraints in $R_k(\mathcal{S}_{\text{LP}})$ yields a constraint set that is sequentially LP k-consistent.*

Proof. For a given 0–1 partial assignment $x_J = v_J$, suppose that $\mathcal{S}_{\text{LP}} \cup \{(x_{J_k}) = (v_{J_k})\}$ is infeasible for $v_k = 0,1$. It suffices to show that $R_k(\mathcal{S}_{\text{LP}})|_{J_{k-1}} \cup \{x_{J_{k-1}} = v_{J_{k-1}}\}$ is infeasible. It follows from [2] that $R_k(\mathcal{S}_{\text{LP}})$ describes the convex hull of the union of $D(\mathcal{S}_{\text{LP}} \cup \{x_k = v_k\})$ over $v_k = 0,1$. We claim that $x_{J_{k-1}} = v_{J_{k-1}}$ does not satisfy $R_k(\mathcal{S}_{\text{LP}})|_{J_{k-1}}$. Assume to the contrary. Then there exists a point $w = (v_{J_{k-1}}, \tilde{v}_k, \tilde{v}_K, \tilde{y})$ that satisfies $R_k(\mathcal{S}_{\text{LP}})$, where $K = N \setminus J_k$. This point must be representable as a convex combination of two points of the form $(v_{J_{k-1}}, 0, \dot{v}_K, \dot{y})$ and $(v_{J_{k-1}}, 1, \ddot{v}_K, \ddot{y})$, since the components of $v_{J_{k-1}}$ are integral and cannot be represented as the convex combination of other points. However, by assumption such points do not exist because $\mathcal{S}_{\text{LP}} \cup \{x_{J_k} = v_{J_k}\}$ is infeasible for $v_k = 0,1$. This yields the desired contradiction. □

Example 13. Consider again Example 6, in which

$$\mathcal{S} = \{2x_1 - 4x_2 \leq -1,\ -2x_1 + 4x_2 \leq 3,\ x_1, x_2 \in \{0,1\}\}$$

Recall that $\mathcal{S}$ is not LP 2-consistent because $x_1 = 0$ is consistent with $\mathcal{S}_{\text{LP}}$ and $(x_1, x_2) = (0, v_2)$ is inconsistent with $\mathcal{S}_{\text{LP}}$ for $v_2 = 0,1$. We wish to achieve sequential LP 2-consistency by applying the modified lift-and-project procedure. First generate the constraints

$$\begin{array}{ll} (2x_1 - 4x_2 + 1)x_2 \leq 0 & x_1 x_2 \geq 0\\ (2x_1 - 4x_2 + 1)(1 - x_2) \leq 0 & x_1(1 - x_2) \geq 0\\ (-2x_1 + 4x_2 - 3)x_2 \leq 0 & (1 - x_1)x_2 \geq 0\\ (-2x_1 + 4x_2 - 3)(1 - x_2) \leq 0 & (1 - x_1)(1 - x_2) \geq 0 \end{array}$$

After linearizing and writing $y_{\{1,2\}}$ simply as y, we obtain the system $R_2(\mathcal{S}_{\text{LP}})$:

$$\begin{array}{ll} -3x_2 + 2y \leq 0 & y \geq 0\\ 2x_1 - x_2 - 2y + 1 \leq 0 & x_1 - y \geq 0\\ x_2 - 2y \leq 0 & x_2 - y \geq 0\\ -2x_1 + 3x_2 + 2y - 3 \leq 0 & -x_1 - x_2 + y + 1 \geq 0 \end{array}$$

Finally, projecting onto x_1 yields $R_2(\mathcal{S})|_{\{1\}} = \{\frac{1}{2} \leq x_1 \leq 1\}$. Adding this constraint to $\mathcal{S}_{\mathrm{LP}}$, as illustrated in Fig. 4, achieves sequential LP 2-consistency because $x_1 = 0$ is inconsistent with the resulting constraint set.

One could, in principle, apply lift-and-project repeatedly to achieve sequential LP k-consistency for $k = 1, \ldots, n$, which would allow one to avoid backtracking altogether. This is impractical, however, because the lift-and-project process quickly explodes in complexity as k increases. However, it may be practical to achieve sequential LP k-consistency for a few small k, a strategy that has three advantages. First, it is computationally manageable for sufficiently small k. Second, it generates sparse cuts (cuts with at most $k-1$ terms), which are generally conceived as the most effective type of cuts in branch-and-bound.

The third advantage is that achieving sequential LP-consistency can avoid branching that traditional cutting planes do not avoid, because it focuses on excluding inconsistent partial assignments, rather than on tightening the LP relaxation by cutting off fractional points. This can be illustrated in a very simple context as follows.

Example 14. Suppose we wish to maximize $3x_2 - x_1$ subject to the constraint set $\mathcal{S}$ in the previous example. Suppose further that we apply a traditional branch-and-cut procedure that generates separating disjunctive cuts at the root node (Fig. 5(a)). The solution of the LP relaxation at the root node is $(x_1, x_2) = (\frac{1}{2}, 1)$. The two disjunctive cuts at this node are $-x_1 + 4x_2 \geq -3$ (corresponding to the disjunction $x_1 = 0 \vee x_1 = 1$) and $x_1 \geq \frac{1}{2}$ (corresponding to $x_2 = 0 \vee x_2 = 1$). Only the first cut is generated, because only it cuts off the fractional solution $(\frac{1}{2}, 1)$. This results in a new LP solution $(x_1, x_2) = (0, \frac{3}{4})$. The procedure then branches on x_2. The $x_2 = 0$ branch yields the fractional LP solution $(x_1, x_2) = (\frac{1}{2}, 0)$, and it is necessary to branch on x_1. The $x_2 = 1$ branch yields the integer LP solution $(x_1, x_2) = (1, 1)$, which solves the problem. The resulting search tree has 5 nodes.

Suppose now that we achieve sequential LP 2-consistency as described in Example 13 by generating the inequality $x_1 \geq \frac{1}{2}$, even though it does not cut off the fractional LP solution (Fig. 5(b)). Since the partial assignment $x_1 = 0$ is inconsistent with the LP relaxation, we immediately branch on $x_1 = 1$, which yields the integer LP solution $(x_1, x_2) = (1, 1)$. The problem is solved with only 2 nodes in the search tree, even though we used no traditional separating cuts at all.

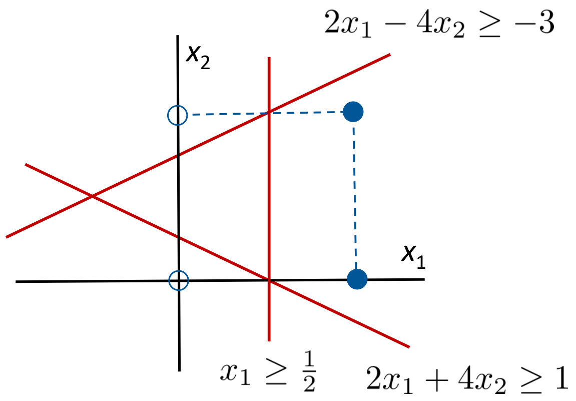

Fig. 4: Illustration of Example 13.

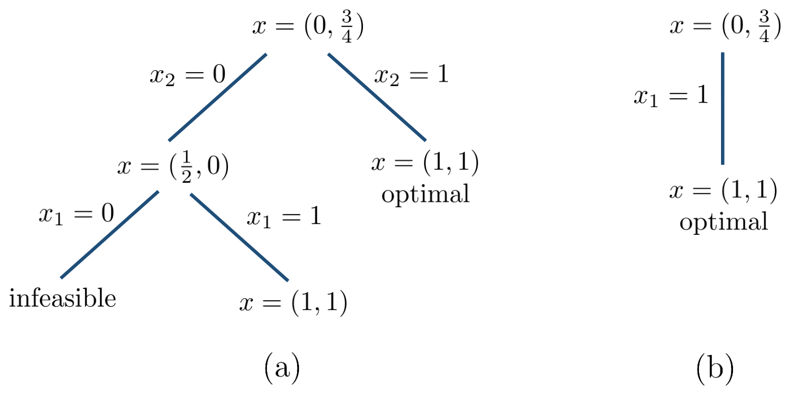

Fig. 5: Illustration of Example 14

.

10 Conclusion

We provided a theoretical foundation for a new type of consistency, LP-consistency, that is particularly suited to 0–1 programming. It is based on the idea that consistency can, in general, be defined with respect to a type of relaxation. LP-consistency is obtained by replacing the relaxation used for traditional consistency concepts with the LP relaxation. We also defined sequential LP k-consistency, a weaker form of LP-consistency that is easier to achieve but nonetheless reduces backtracking. In fact, sequential k-consistency can be obtained by a restricted form of the well-known lift-and-project process of integer programming. LP-consistency maintenance brings a new approach to 0–1 programming because it focuses on eliminating inconsistent 0–1 partial assignments rather than fractional solutions of the LP relaxation. We showed that achieving even sequential 2-consistency can avoid backtracking that traditional cutting planes allow.

This work points to at least three further research programs. One is to extend the concepts introduced here to general mixed integer/linear programming (MILP), which appears to be straight-foward. A second is to investigate the computational usefulness of sequential LP k-consistency for MILP solvers, in particular by achieving sequential LP k-consistency for small k near the top of the search tree. A third is to conduct a systematic study of the ability of traditional cutting planes to achieve consistency, both traditional forms and LP-consistency, in an MILP problem. This could allow one to make better use of known cutting planes by generating cuts that do not separate fractional solutions but enhance the consistency properties of the constraint set.

References

1. Apt, K.R.: Principles of Constraint Programming. Cambridge University Press, Cambridge, UK (2003)
2. Balas, E.: Disjunctive programming and a hierarchy of relaxations for discrete optimization problems. SIAM Journal on Algebraic and Discrete Methods **6**, 466–485 (1985)
3. Chang, C.L.: The unit proof and the input proof in theorem proving. Journal of the ACM **14**, 698–707 (1970)
4. Chvátal, V.: Edmonds polytopes and a hierarchy of combinatorial problems. Discrete Mathematics **4**, 305–337 (1973)

5. Davis, E.: Constraint propagation with intervals labels. Artificial Intelligence **32**, 281–331 (1987)
6. Freuder, E.C.: Synthesizing constraint expressions. Communications of the ACM **21**, 958–966 (1978)
7. Freuder, E.C.: A sufficient condition for backtrack-free search. Communications of the ACM **29**, 24–32 (1982)
8. Hooker, J.N.: Input proofs and rank one cutting planes. ORSA Journal on Computing **1**, 137–145 (1989)
9. Hooker, J.N.: Integrated Methods for Optimization, 2nd ed. Springer (2012)
10. Hooker, J.N.: Projection, consistency, and George Boole. Constraints **21**, 59–76 (2016)
11. Mackworth, A.: Consistency in networks of relations. Artificial Intelligence **8**, 99–118 (1977)
12. Montanari, U.: Networks of constraints: Fundamental properties and applications to picture processing. Information Science **7**, 95–132 (1974)
13. Quine, W.V.: The problem of simplifying truth functions. American Mathematical Monthly **59**, 521–531 (1952)
14. Quine, W.V.: A way to simplify truth functions. American Mathematical Monthly **62**, 627–631 (1955)
15. Régin, J.C.: Global constraints: A survey. In: Milano, M., Van Hentenryck, P. (eds.) Hybrid Optimization: The Ten Years of CPAIOR, pp. 63–134. Springer, New York (2010)
16. Tsang, E.: Foundations of Constraint Satisfaction. Academic Press, London (1983)
17. Van Hentenryck, P.: Constraint Satisfaction in Logic Programming. MIT Press, Cambridge, MA (1989)