

Inference Duality as a Basis for Sensitivity Analysis *

J. N. Hooker

Graduate School of Industrial Administration, Carnegie Mellon University,
Pittsburgh, PA 15213 USA,

Abstract. The constraint programming community has recently begun to address certain types of optimization problems. These problems tend to be discrete or to have discrete elements. Although sensitivity analysis is well developed for continuous problems, progress in this area for discrete problems has been limited. This paper proposes a general approach to sensitivity analysis that applies to both continuous and discrete problems. In the continuous case, particularly in linear programming, sensitivity analysis can be obtained by solving a dual problem. One way to broaden this result is to generalize the classical idea of a dual to that of an "inference dual," which can be defined for any optimization problem. To solve the inference dual is to obtain a proof of the optimal value of the problem. Sensitivity analysis can be interpreted as an analysis of the role of each constraint in this proof. This paper shows that traditional sensitivity analysis for linear programming is a special case of this approach. It also illustrates how the approach can work out in a discrete problem by applying it to 0-1 linear programming (linear pseudo-boolean optimization).

1 Introduction

Sensitivity analysis addresses the issue of how much the solution of an optimization problem responds to perturbations in the problem data. It is an indispensable element of applied modeling, perhaps as important as obtaining the solution itself. It is needed not only to anticipate the effect of changes in the problem, but to deal with the fact that in applied work, obtaining the information necessary to formulate an accurate model is often the hardest part of the task. Sensitivity analysis typically reveals that the solution depends primarily on a few key data, whereas the rest of the problem can be altered somewhat without appreciable effect. This allows one to focus time and resources on collecting and verifying the information that really matters. More generally, it directs the decision maker's attention to those aspects of the problem that should be closely watched.

By far the most widely used optimization tool is linear programming, for which sensitivity analysis is highly developed. One of the basic results is that the solution of the linear programming *dual* indicates the sensitivity of the optimal

* Partially supported by U.S. Office of Naval Research grant N00014-95-1-0517.

value to perturbations in the right-hand sides of the inequality constraints. These results have been extended to certain discrete optimization problems. There are duality theories for integer programming, for instance, that can serve as a basis for sensitivity analysis; a brief survey may be found in Section 23.7 of [13]. Integer dual solutions become very complex as the problem grows, however, and are rarely used in practice. These and related approaches (e.g., [12, 14]) are based on an investigation of how the optimal value depends on the right-hand sides of inequality and equality constraints, and it is unclear how the ideas would generalize to problems with other types of constraints.

The approach taken here is to define the *inference dual* of an optimization problem and use it as the basis for sensitivity analysis. The inference dual is the problem of inferring from the constraints a best possible bound on the optimal value. The solution of the dual is a proof, using an inference method that is appropriate to the problem. Sensitivity analysis can be viewed as an analysis of the role of each constraint in this proof. For example, a constraint may not even appear as a premise in the proof, in which case it is redundant, or if it does, the proof may yet go through if the constraint is weakened by a determinable amount. This type of analysis can in principle be carried out for any type of constraint.

There may be several proofs of an optimal bound, and if so sensitivity analysis differs somewhat in each. This phenomenon is known as “degeneracy” in classical mathematical programming, where it tends to be regarded as a technical nuisance. Here it is seen to be a natural outcome of the fact that more than one rationale can be given for an optimal solution.

To solve the inference dual, one must

- a) identify inference rules that are complete for the type of constraints in the problem;
- b) use the rules to prove optimality.

In the linear programming dual, for example, one infers inequalities from other inequalities. The inference rule is simple: all inequalities implied by a constraint set can be obtained by taking nonnegative linear combinations of the constraints. This is essentially the content of the classical “separation lemma” for linear programming, which is therefore a completeness theorem for linear inference. To find the particular linear combination that solves the dual, one can use information obtained in solving the original problem (the “primal”). The multipliers in this combination indicate the sensitivity of the optimal value to perturbations in the right-hand sides of the corresponding constraints.

The question here is how to address points (a) and (b) for discrete optimization problems. One answer lies in the idea of deriving a dual solution from information gathered while solving the primal problem. A discrete problem can be solved enumeratively by building a search tree that branches on values of the variables. The structure of this tree reflects the structure of a proof of optimality, in the following way. First, a constraint is added to require the objective function value to be less than the true minimum, so that the tree now establishes

infeasibility of the augmented constraint set. At each leaf node of the tree, a certain type of proposition (a “multivalent clause”) that is violated at that node is inferred from one of the constraints. The tree now indicates the structure of a resolution proof that the multivalent clauses are unsatisfiable, using a “multivalent” resolution method that generalizes classical resolution. The inference rules required in (a) are therefore those of multivalent resolution, plus those needed to infer multivalent clauses from constraints. The proof required in (b) is given by the structure of the search tree. Sensitivity analysis consists generally in noting the role played by each constraint in the proof, and specifically in checking how much the constraints can be changed so that they still imply the multivalent clauses that are used as premises in the proof.

Inference duality also permits a generalization of Benders decomposition to any optimization problem. This technique allows one to generate “Benders cuts” that are analogous to nogoods but that exploit problem structure in a way that nogoods do not. This idea is developed in [7]. Other connections between logical inference and optimization are surveyed in [4, 6].

The paper begins with an elementary treatment of the inference dual and its role in linear programming sensitivity analysis. It then describes multivalent resolution and shows how an optimality proof of this kind can be recovered from an enumeration tree that solves the primal problem. It concludes with an application to linear 0-1 programming and some practical observations.

2 The Inference Dual

Consider a general optimization problem,

$$\begin{aligned} \min f(x) & \qquad (1) \\ \text{s.t. } x \in S \\ & x \in D. \end{aligned}$$

The *domain* D is distinguished from the *feasible set* S . In most applications x is a vector (x_1, \dots, x_n) , in which case D_{x_j} denotes the domain of x_j .

To state the inference dual it is necessary to define the notion of implication with respect to a domain D . Let P and Q be two propositions about x ; that is, their truth or falsehood is determined by the value of x . P *implies* Q with respect to D (notated $P \xrightarrow{D} Q$) if Q is true for any $x \in D$ for which P is true.

The *inference dual* of (1) is

$$\begin{aligned} \max z & \qquad (2) \\ \text{s.t. } x \in S \xrightarrow{D} f(x) \geq z \end{aligned}$$

So the dual seeks the largest z for which $f(x) \geq z$ can be inferred from the constraint set. A strong duality theorem is true almost by definition. To state it, it is convenient to say that the optimal value of a minimization problem is respectively ∞ or $-\infty$ when the problem is infeasible or unbounded, and vice-versa for a maximization problem.

Theorem 1 Strong Inference Duality. *The optimization problem (1) has the same optimal value as its inference dual (2).*

Proof. If z^* is the optimal value of (1), then clearly $x \in S$ implies $f(x) \geq z^*$, which shows that the optimal value of the dual is at least z^* . The dual cannot have an optimal value larger than z^* , because this would mean that $f(x) = z^*$ cannot be achieved in (1) for any feasible x . If (1) is infeasible, then any z is feasible in (2), which therefore has optimal value ∞ . If (1) is unbounded, then (2) is infeasible with optimal value $-\infty$. \square

Because strong duality is a trivial affair for inference duality, interesting duality theorems must deal with some other aspect. A natural task for a duality theorem is to provide a complete method for deriving inferences in the dual, as explained earlier.

A dual solution provides sensitivity analysis because *it specifies the role played by each constraint in a deduction of the optimal value*. This is illustrated below in the cases of linear and 0-1 programming.

3 Linear Programming Duality and Sensitivity Analysis

A linear programming problem

$$\begin{aligned} \min \quad & cx \\ \text{s.t.} \quad & Ax \geq a \\ & x \geq 0, \end{aligned} \tag{3}$$

where matrix A is $m \times n$, has the following inference dual.

$$\begin{aligned} \max \quad & z \\ \text{s.t.} \quad & (Ax \geq a, x \geq 0) \xrightarrow{R^n} cx \geq z. \end{aligned} \tag{4}$$

The dual looks for a *linear implication* $cx \geq z$ of the constraints that maximizes z . Linear implication is characterized by the following, which is equivalent to a classical separation lemma for linear programming.

Theorem 2 Linear implication.. *$Ax \geq a$ linearly implies $cx \geq z$ (i.e., $Ax \geq a \xrightarrow{R^n} cx \geq z$) if and only if $Ax \geq a$ is infeasible or there is a real vector $u \geq 0$ for which $uA \leq c$ and $ua \geq z$.*

This means that the dual (4) seeks a nonnegative linear combination $uAx \geq ua$ of $Ax \geq a$ that dominates $cx \geq z$ (i.e., $uA \leq c$ and $ua \geq z$) and that maximizes z . So the dual can be written in the classical way,

$$\begin{aligned} \max \quad & ua \\ \text{s.t.} \quad & uA \leq c \\ & u \geq 0 \end{aligned} \tag{5}$$

The vector $u \in R^m$ in effect encodes a proof, because it gives instructions for deducing the optimal value z . The duality theorem for linear programming follows immediately from Theorem 2.

Corollary 3. *The optimal value of a linear programming problem (3) is the same as that of its classical dual (5), except when both are infeasible.*

Note that unlike inference duality, the classical theorem requires a regularity condition to the effect that either a problem or its dual must be feasible.

The optimal dual solution u^* provides sensitivity analysis because it indicates the role of each constraint in a proof of optimality. For instance, if $u_i^* = 0$, then constraint i has no role in the proof, and the constraint can be omitted without invalidating the proof and therefore without changing the optimal value.

More generally, one can reason as follows. If the vector a of right-hand sides is changed to $a + \Delta a$, then u^* remains a feasible solution of the resulting dual problem (5) with value $u^*(a + \Delta a)$. So the optimal dual value is at least $u^*(a + \Delta a)$, and by Corollary 3 the same is true of the optimal solution of the primal problem.

This means that if constraint i is strengthened by raising its right-hand side a_i by $\Delta a_i \geq 0$, the optimal value will rise at least $u_i^* \Delta a_i$. (In particular, it will rise to ∞ if the problem becomes infeasible.) If a_i is reduced by Δa_i , the optimal value can fall no more than $u_i^* \Delta a_i$. The increase or decrease is exactly $u_i^* \Delta a_i$ if Δa_i lies within easily computable bounds.

The dual problem can have several optimal extreme point solutions (i.e., optimal solutions that are not convex combinations of each other). This can occur when the primal solution is "degenerate." Each solution gives rise to a different sensitivity analysis.

A more complete exposition of linear programming sensitivity analysis may be found in [2].

4 A Multivalent Resolution Method

It is well known that the resolution method originally developed by Quine [9, 10] (and later extended to first order logic by Robinson [11]) provides a complete refutation method for propositional logic in conjunctive normal form.²

The resolution method is readily generalized to problems in which the variable domains contain more than two discrete values. The method that results is related to Cooper's algorithm for achieving k -consistency [3], but it is convenient here to cast it as an inference method. Consider a set S of *multivalent clauses* of the form,

$$(x_1 \in T_{i1}) \vee \dots \vee (x_n \in T_{in}),$$

where each $x_j \in T_{ij}$ is a *literal*, and each $T_{ij} \subset D_{x_j}$. If $T_{ij} = \emptyset$, then the literal $x_j \in T_{ij}$ can be omitted from the disjunction, and it is convenient to say that the clause does not contain x_j . The *resolvent on x_j* of the clauses in S is

$$(x_j \in \bigcap_i T_{ij}) \vee \bigvee_{k \neq j} (x_k \in \bigcup_i T_{ik}).$$

² Quine's method, sometimes called consensus, was actually for formulas in disjunctive normal form, but it is easily dualized to treat conjunctive normal form.

For example, the first three clauses below resolve on x_1 to produce the fourth. Here each x_j has domain $\{1, 2, 3, 4\}$.

$$\begin{aligned} &(x_1 \in \{1, 4\}) \vee (x_2 \in \{1\}) \\ &(x_1 \in \{2, 4\}) \vee (x_2 \in \{2, 3\}) \\ &(x_1 \in \{3, 4\}) \vee (x_2 \in \{1\}) \\ &(x_1 \in \{4\}) \vee (x_2 \in \{1, 2, 3\}) \end{aligned}$$

To check a multivalent clause set S for satisfiability, identify a subset of clauses whose resolvent does not already belong to S . If there is no such subset, stop and conclude that S is satisfiable. If the resolvent is the empty clause (i.e., each $T_j = \emptyset$), stop and conclude that S is unsatisfiable. Otherwise add the resolvent to S and repeat. The proof that multivalent resolution is a sound and complete refutation method is parallel to that for ordinary resolution. A weaker result (Theorem 4, below) will suffice for present purposes.

5 Solving the Dual via the Primal

It is possible in general to characterize a *primal method* for solving a problem as one that examines possible values of the variables, and a *dual method* a one that examines possible proofs of optimality without interpreting the variables. In classical optimization, a branch-and-bound method is a primal method, whereas a pure cutting plane method is essentially a dual method (see [8] for background). Primal methods have generally proved more effective for optimization, although primal and dual methods are often combined, as in branch-and-cut and dual ascent algorithms.

Fortunately, the inference dual of a discrete optimization problem can be solved by examining the results of a primal method, in particular an enumeration tree that is generated to solve the primal problem. In fact, this approach yields both a complete refutation method for the type of inference used in the dual and an algorithm for constructing a proof of optimality. It will be seen that a refutation method, as opposed to a full inference method, suffices for sensitivity analysis.

The most straightforward way to solve (1) by enumeration is to branch on values of the variables until all feasible solutions have been found. The search backtracks whenever a feasible solution is found or some constraint is violated. The best feasible solution is optimal. A generic algorithm for generating the search tree appears in Fig. 1. The development below is readily modified to accommodate search algorithms that use bounding and other devices for pruning the tree.

Let z^* be the optimal value found by enumeration. To solve the dual, first modify the enumeration tree so that it refutes the claim that $f(x) < z^*$. This is done simply by adding the constraint $f(x) < z_t$ to the constraint set \mathcal{C} for each node t at which a feasible solution is found, where z_t is the value of that solution. Then $f(x) < z_t$ can be regarded as the constraint violated at node t . There is now at least one violated constraint at every leaf node.

```

Let  $C$  be the set of constraints that define the feasible set  $S$  in (1).
Let  $L$  be a list of active nodes, initially containing the root node,
    which is associated with an empty set of assignments (labels).
Let  $\bar{z}$  be an upper bound on the optimal value; initially  $\bar{z} = \infty$ .
While  $L$  is nonempty:
    Remove a node from  $L$ , and let  $A$  be the set of assignments
        associated with the node.
    If the assignments in  $A$  violate no constraint in  $C$  then
        If  $A$  assigns values  $v_1, \dots, v_n$  to every variable  $x_1, \dots, x_n$  so as
            to satisfy every constraint in  $C$  (or the assignments in  $A$ 
            can be readily extended to all variables so as to satisfy
            every constraint) then
                Let  $\bar{z} = \min\{\bar{z}, f(v_1, \dots, v_n)\}$ .
            Else
                Choose a variable  $x_j$  that  $A$  does not assign a value.
                For each  $v \in D_{x_j}$ :
                    Add a node to  $L$  and associate it with  $A \cup \{x_j = v\}$ .
    If  $\bar{z} < \infty$  then  $\bar{z}$  is the optimal value of (1).
    Else (1) is infeasible.

```

Fig. 1. A generic enumeration algorithm for solving the primal problem.

For any leaf node t let $(x_{j_1}, \dots, x_{j_d}) = (v_1, \dots, v_d)$ be the assignments made along the path from node t to the root. As just noted, these assignments violate some $C_t \in C$. Because C_t is equivalent to the conjunction of all multivalent clauses it implies, $(x_{j_1}, \dots, x_{j_d}) = (v_1, \dots, v_d)$ violates some multivalent clause M_t implied by C_t . Without loss of generality M_t can be assumed to contain only variables in $\{x_{j_1}, \dots, x_{j_d}\}$. The enumeration tree is now a refutation of $\bigwedge_t M_t$. Due to the following theorem, the tree's structure indicates how to construct a resolution proof of $\neg \bigwedge_t M_t$.

Theorem 4. *Consider an enumeration tree in which a multivalent clause M_t is associated with each node t . Let the clause associated with any nonleaf node be the resolvent on x_j of those associated with the node's children, where x_j is the variable on which the tree branches at node i . Then if M_t is falsified at each leaf node t , the clause associated with the root node is empty.*

Proof. It suffices to show that the clause associated with the root node is falsified, because only the empty clause is falsified when no variables are fixed. This can be proved by induction. It is given that M_t is falsified at any leaf node t . Now consider any nonleaf node k , and let nodes $1, \dots, p$ be its children. By the induction hypothesis, M_1, \dots, M_p are falsified. Suppose that the search branches on variable x_j at node k and that $x_j = v_s$ is the assignment that generates child node s . Then M_s cannot contain a literal $x_j \in T_{s_j}$ with $v_s \in T_{s_j}$. If it did, M_s would be true at node s . Therefore $\bigcap_{s=1}^p T_{s_j} = \emptyset$, which means that the resolvent M_k of $\{M_1, \dots, M_p\}$ does not contain x_j . So M_k is falsified at node k . It follows by induction that the root clause is falsified. \square

A refutation of $f(x) < z^*$ can therefore be constructed from two ingredients: multivalent resolution, and an algorithm for checking whether a constraint implies a multivalent clause that is violated by a given variable assignment $(x_{j_1}, \dots, x_{j_m}) = (v_1, \dots, v_m)$. The construction proceeds as follows. For each leaf node t of the enumeration tree, select any constraint C_t violated at that node; if the node represents a feasible solution with value z_t , let C_t be $f(x) < z_t$. For each t identify a multivalent clause M_t that a) is implied by C_t , b) contains only variables among those that are fixed along the path from node t to the root, and c) is violated at node t . Then the desired refutation infers each M_t from C_t and then refutes $\bigwedge_t M_t$ using multivalent resolution, as indicated in Theorem 4.

Sensitivity analysis now consists of observing the role of the constraints in the refutation just constructed. Some specific results can be obtained by applying the three principles below. Let a node be *feasible* if no constraints in \mathcal{C} are violated at the node, and *infeasible* otherwise. The principles do not require that the resolution refutation actually be constructed; only that the clauses M_t at infeasible nodes and the values z_t at feasible nodes be saved. Because the search tree may have a very large number of leaf nodes, it is practical to save only the *distinct* M_t 's associated with a given constraint. So for any constraint $C \in \mathcal{C}$ let $\{M_t \mid t \in M(C)\}$ be the set of distinct clauses M_t for which $C = C_t$. The number of feasible nodes is likely to be small, however, and it is practical to keep a list of z_t for each feasible node t as well as which variables are fixed (and to what values they are fixed) at t .

- (S1) If a constraint in \mathcal{C} is not associated with any leaf nodes, then it is redundant and can be dropped without affecting the optimal value.
- (S2) More generally, if a constraint $C \in \mathcal{C}$ is replaced with a constraint C' that still implies M_t for all $t \in M(C)$, then:
- i) the optimal value will not decrease;
 - ii) the optimal value will be at least $\min_{t \in I} \{z_t\}$, where I is the set of feasible nodes at which C' is not violated.
- (S3) Still more generally, if a constraint $C \in \mathcal{C}$ is replaced with C' , then the optimal value will be at least

$$\min \left\{ \min_{t \in I} z_t, \min_{t \in I'} \underline{z}_t \right\},$$

where I is as above, and I' is the set of nodes $t \in M(C)$ at which C' does not imply M_t . Also \underline{z}_t is the minimum value of the objective function $f(x)$ subject only to $\neg M_t$. That is,

$$\underline{z}_t = \min \{ f(x) \mid x_j \in D_{x_j} \setminus T_{tj} \text{ for all } x_j \text{ in } M_t \}.$$

These principles can clearly be adapted to analyze the effect of changing two or more constraints simultaneously.

6 0-1 Duality and Sensitivity Analysis

A 0-1 linear programming problem may be stated,

$$\begin{aligned} \min \quad & cx \\ \text{s.t.} \quad & Ax \geq a \\ & x \in \{0, 1\}^n. \end{aligned} \tag{6}$$

The inference dual is,

$$\begin{aligned} \max \quad & z \\ \text{s.t.} \quad & Ax \geq a \xrightarrow{\{0,1\}^n} cx \geq z. \end{aligned} \tag{7}$$

A separation lemma for this dual would consist of a complete inference method for linear 0-1 inequalities. Such a method was presented in [5]. Only a complete refutation method is required for present purposes, however, and it can be obtained as indicated in the previous section.

A refutation method is obtained by combining multivalent resolution with a method for checking whether a 0-1 inequality implies a multivalent clause that is falsified by a given variable assignment. Because the variables in this case are bivalent, multivalent clauses become ordinary clauses. Multivalent resolution therefore reduces to ordinary resolution. It is straightforward to check whether $ax \geq \alpha$ implies a clause that is falsified by $(x_{j_1}, \dots, x_{j_d}) = (v_{j_1}, \dots, v_{j_d})$. Let J be the set of indices j in $\{j_1, \dots, j_d\}$ for which $a_j > 0$ if $v_j = 1$ and $a_j < 0$ if $v_j = 0$. Let $x_j(v)$ be x_j if $v = 1$ and $\neg x_j$ if $v = 0$. Then $\bigvee_{j \in J} x_j(1 - v_j)$ is the desired clause if $\sum_{j \in J} a_j v_j + \sum_{j \notin J} \max\{0, a_j\} < \alpha$, and otherwise there is no such clause.

Consider for example the problem,

$$\begin{aligned} \min \quad & 7x_1 + 5x_2 + 3x_3 \\ \text{s.t.} \quad & 2x_1 + 5x_2 - x_3 \geq 3 \quad (a) \\ & -x_1 + x_2 + 4x_3 \geq 4 \quad (b) \\ & x_1 + x_2 + x_3 \geq 2 \quad (c) \\ & x \in \{0, 1\}^3. \end{aligned} \tag{8}$$

The enumeration tree of Fig. 2 solves the problem. The optimal solution value is 8.

A resolution proof that $z \geq 8$ can be reconstructed in a way very similar to that presented above for satisfiability problems. At each leaf node, one of the violated inequalities is chosen as a premise; in this case, constraints (a) and (b) suffice. If the leaf node represents a feasible solution, the premise consists of the constraint $cx \leq z - 1$. These combined premises must lead to a contradiction, thus proving that $z \geq 8$.

The contradiction can be demonstrated by resolution, as depicted in Fig. 3. The inequalities at nodes 8 and 9, for instance, respectively imply clauses that resolve to obtain $\neg x_1 \vee \neg x_2$ at node 4. The latter resolves with x_2 , implied by the inequality at node 5, to obtain $\neg x_1$ at node 2.

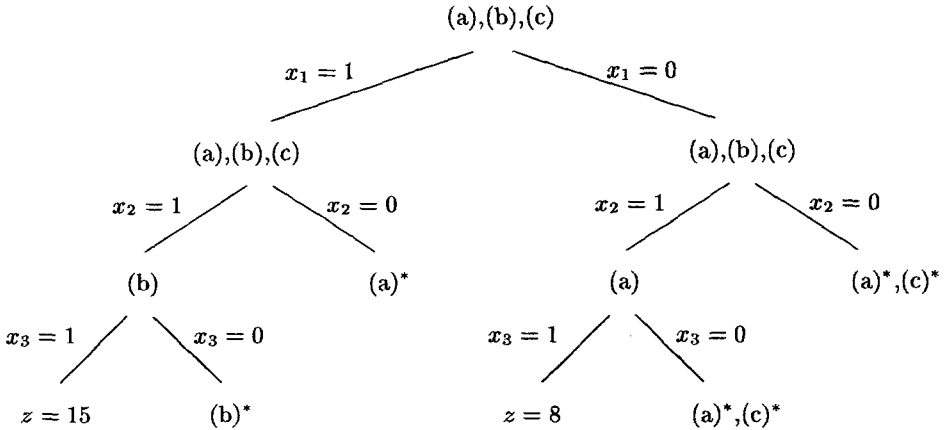


Fig. 2. A solution of the subproblem by branching. The constraints remaining (i.e., not yet satisfied) at each nonleaf node are indicated. The constraints violated at each leaf node, if any, are indicated with an asterisk; if no constraints are violated, the objective function value z is shown.

The inequality at node 10 implies a second clause $\neg x_1 \vee \neg x_3$ (not shown in Fig. 3) that resolves with x_3 at node 11. But this clause can be neglected because it is not falsified by variables fixed between nodes 10 and the root.

Because the clauses inferred at the leaf nodes are falsified by the fixed variables, the enumeration tree proves they are unsatisfiable. So there is a resolution proof of unsatisfiability. In this case, the empty clause is generated below the root, at node 3.

The three principles (S1)-(S3) cited earlier for carrying out sensitivity analysis are readily applied to this 0-1 programming example.

(S1) Because constraint (c) is associated with no leaf node in Fig. 3, it is redundant and can be dropped without changing the optimal solution.

(S2) Constraint (a) is associated with leaf nodes 5 and 7. Both M_5 and M_7 are the singleton clause x_2 , and so one can set $M(C) = \{5\}$.

i) Constraint (a) can be altered in any fashion such that it continues to imply $x_2 = 1$, without reducing the optimal value. For instance, the right-hand side can be reduced to any number greater than 2 and increased arbitrarily. The coefficient of x_2 can be changed arbitrarily (if the inequality becomes infeasible, it implies x_2 as well as any other clause). The coefficient of x_1 can be increased to any number less than 3 and reduced arbitrarily, and so forth. Constraint (b) can be similarly analyzed. It is not hard to write general procedures for this.

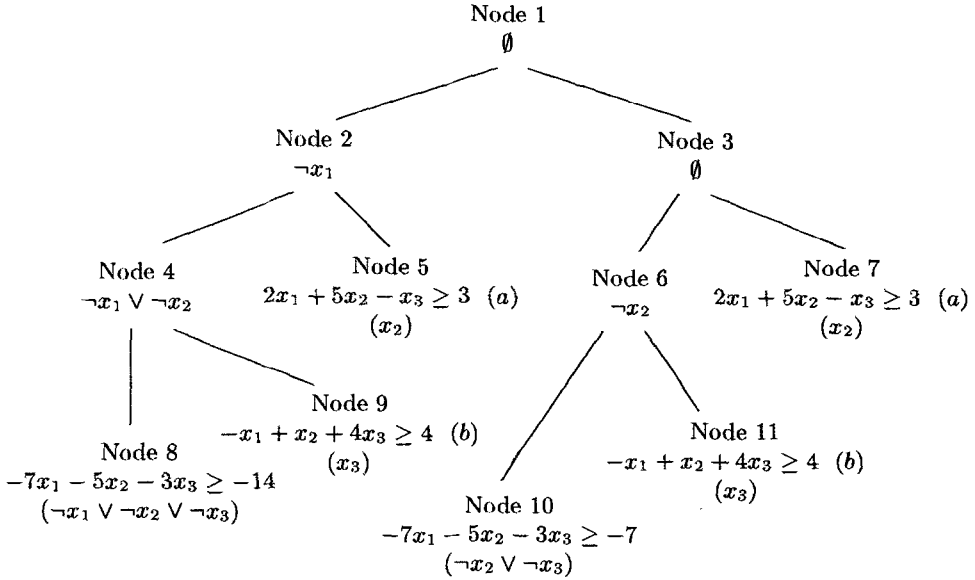


Fig. 3. Construction of a proof of $z \geq 8$. The violated constraint at each leaf node is shown, along with a falsified clause it implies. Resolvents are shown at the nonleaf nodes.

ii) Suppose the right-hand side of constraint (a) is raised to 5. Of the two feasible nodes 8 and 10, (a) is now violated at 10. The new optimal value is therefore at least $z_8 = 15$.

(S3) Suppose constraint (a) is weakened by changing it to $2x_1 + 5x_2 - x_3 \geq 2$, so that it no longer implies x_2 ($I' = \{5\}$). Constraint (a) of course remains unviolated at the feasible nodes 8 and 10 ($I = \{8, 10\}$). So the optimal value is at least

$$\min \{ \min \{ z_8, z_{10} \}, \underline{z}_5 \} = 0,$$

where $\underline{z}_5 = \min \{ 7x_1 + 5x_2 + 3x_3 \mid \neg x_2 \} = 0$. In this case no useful bound is obtained. Constraint (b) can be similarly analyzed.

7 Practical Considerations

The style of sensitivity analysis proposed here must be adapted to the problem context. The interests of practitioners should guide which questions are asked, and these questions should guide the type of perturbations that are studied. Even in classical linear programming, sensitivity analysis can provide information that is too complex and voluminous to be assimilated. One must take care to highlight

the results that are intelligible and relevant. At this point it is impossible to predict the problem classes in which inference duality can yield useful sensitivity results. Only practical trials can resolve this issue.

One possible impediment to the interpretation of sensitivity analysis is “massive degeneracy.” There may be a large number of dual solutions, each giving rise to a different sensitivity analysis. Fortunately, a single dual solution can show a constraint to be *unimportant*. For instance, if a constraint is redundant in one dual solution, then it is redundant *simpliciter*, in the sense that it can be dropped without changing the solution. Or if one dual solution yields an upper bound on the effect of a problem alteration, this bound is valid in general. But to establish categorically that a constraint is *important*, one must show that it is important in all dual solutions.

The effects of degeneracy can be ameliorated somewhat. Degeneracy has two sources: many different search trees arrive at the same optimal value, and a given search tree can be analyzed in many different ways. The first source must be accepted, because it is usually impractical to solve a problem more than once. However, if the role of a few particular constraints are of interest, the search tree can be analyzed with this in mind. These constraints should be avoided, whenever possible, when associating violated constraints with leaf nodes. More generally, the multivalent clauses derived from violated constraints should be as weak as possible. Ultimately, however, degeneracy must be viewed as inherent in the nature of things rather than an artifact of the analysis. It is often an unavoidable fact that different rationales can be given for an optimal solution, each drawing on different information in the constraint set.

Although the results presented here apply only to problems that are solved by tree search, it is rare that a (verified) optimal solution is found by other means in discrete problems. As noted earlier, the analysis can be modified to accommodate tree searches that use such devices as bounding and relaxations to prune the tree.

It is also common for applications to require both discrete and continuous variables. It is an interesting research issue as to how the classical methods for continuous sensitivity analysis may be combined with the discrete methods presented here.

References

1. Barth, P., *Logic-Based 0-1 Constraint Solving in Constraint Logic Programming*, Kluwer (Dordrecht, 1995).
2. Chvátal, V., *Linear Programming*, W. H. Freeman (New York, 1983).
3. Cooper, M. C., An optimal k -consistency algorithm, *Artificial Intelligence* **41** (1989) 89-95.
4. Hooker, J. N., A quantitative approach to logical inference, *Decision Support Systems* **4** (1988) 45-69.
5. Hooker, J. N., Generalized resolution for 0-1 linear inequalities, *Annals of Mathematics and Artificial Intelligence* **6** (1992) 271-286.

6. Hooker, J. N., Logic-based methods for optimization, in A. Borning, ed., *Principles and Practice of Constraint Programming, Lecture Notes in Computer Science* **874** (1994) 336-349.
7. Hooker, J. N., Logic-based Benders decomposition (1996). Available on <http://www.gsia.cmu.edu/afs/andrew/afs/jh38/jnh.html>.
8. Nemhauser, G. L., and L. A. Wolsey, *Integer and Combinatorial Optimization*, Wiley (New York, 1988).
9. Quine, W. V., The problem of simplifying truth functions, *American Mathematical Monthly* **59** (1952) 521-531.
10. Quine, W. V., A way to simplify truth functions, *American Mathematical Monthly* **62** (1955) 627-631.
11. Robinson, J. A., A machine-oriented logic based on the resolution principle, *Journal of the ACM* **12** (1965) 23-41.
12. Schrage, L., and L. Wolsey, Sensitivity analysis for branch and bound integer programming, *Operations Research* **33** (1985) 1008-1023.
13. Schrijver, A., *Theory of Linear and Integer Programming*, Wiley (Chichester, 1986).
14. Skorin-Kapov, J., and F. Granot, Nonlinear integer programming: Sensitivity analysis for branch and bound, *Operations Research Letters* **6** (1987) 269-274.