A Subgradient Algorithm for Nonlinear Integer Programming[*]

Zhijun Wu[†]

Abstract. This paper describes a subgradient approach to nonlinear integer programming and, in particular, nonlinear 0-1 integer programming. In this approach, the objective function for a nonlinear integer program is considered as a nonsmooth function over the integer points. The subgradient and the supporting plane for the function are defined, and a necessary and sufficient condition for the optimal solution is established, based on the theory of nonsmooth analysis. A new algorithm, called the subgradient algorithm, is developed. The algorithm is in some sense an extension of Newton's method to discrete problems: The algorithm searches for a solution iteratively among the integer points. In each iteration, it generates the next point by solving the problem for a local piecewise linear model. Each local model is constructed using the supporting planes for the objective function at a set of previously generated integer points. A solution is found when either the optimality condition is satisfied or an iterate is repeated. In either case, the algorithm terminates in finite steps. The theory and the algorithm are presented. The methods for computing the supporting planes and solving the linear subproblems are described. Test results for a small set of problems are given.

Abbreviated title: Nonlinear Integer Programming

Key words: Nonlinear integer programming, subgradient methods, nonlinear least squares, nonlinear constrained optimization, linear integer programming, branch-and-bound methods

AMS (MOS) subject classification: 65K05, 65K10, 90C10, 90C27, 90C30

[*]This work is part of author's Ph.D. thesis directed by Professors John E. Dennis and Robert E. Bixby at the Department of Mathematical Sciences, Rice University.

[†]Current Address: Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439.

1 Introduction

We are interested in solving a class of nonlinear integer programming problems

$$\begin{aligned} \mathbf{min} \quad & f(x) \\ & x \in \quad B^n = \{0,1\}^n \end{aligned} \tag{1}$$

or its natural extension

$$\begin{aligned} \mathbf{min} \quad & f(x) \\ & x \in R^n \qquad integral, \end{aligned} \tag{2}$$

where $f : R^n \longrightarrow R$ is a general nonlinear function.

This class of problems contains many $\mathcal{NP}$-hard problems and has important theoretical and practical applications. For example, consider the problem that for any norm $\| \ \|$,

$$\begin{aligned} \mathbf{min} \quad & \| b - Ax \| \\ & x \in R^n \qquad integral, \end{aligned} \tag{3}$$

where $b \in R^m$, and A is an $m \times n$ matrix with integer elements. This problem, called the closest vector problem in integer programming, has been proven to be $\mathcal{NP}$-complete even for simple norms such as l_2 and l_∞ [11, 24, 25].

Another example is related to the solution of a class of more general problems: mixed-integer nonlinear programming problems. A mixed-integer nonlinear program

$$\begin{aligned} \mathbf{min} \quad & g(x, y) \\ & y \in R^m \\ & x \in R^n \qquad integral \end{aligned} \tag{4}$$

can be formulated, under appropriate assumptions, as a nonlinear integer program

$$\begin{aligned} \mathbf{min} \quad & f(x) \\ & x \in R^n \qquad integral, \end{aligned} \tag{5}$$

where

$$f(x) = \mathbf{min}\ \{g(x, y) : \ y \in R^n\}. \tag{6}$$

If x is bounded, Problem (2) can be transformed into Problem (1). Therefore, we focus only on Problem (1) in this work.

Several approaches to the solution of Problem (1) have been studied. The main ones are enumeration, algebraic, and linearization approaches [1, 2, 7, 16, 17, 20, 21]. For a general review, readers are referred to [8, 13, 19, 22, 26]. Most of these approaches consider problems with special structures. For problems with general objective functions, such as Problem (3) and Problem (4), they usually do not apply, owing to their special requirements for the form of the objective function.

In this work, Problem (1) is considered for general cases. A subgradient approach to the problem is proposed. In this approach, the objective function for a nonlinear integer program is considered as a nonsmooth function over the integer points. The subgradient and the supporting plane for the function are defined, and a necessary and sufficient condition for the optimal solution is established, based on the theory of nonsmooth analysis [6, 28, 29]. A new algorithm, called the subgradient algorithm, is developed. The algorithm is in some sense an extension of Newton's method to discrete problems: The algorithm searches for a solution iteratively among the integer points. In each iteration, it generates the next point by solving the problem for a local piecewise linear model. Each local model is constructed using the supporting planes for the objective function at a set of previously generated integer points. A solution is found when either the optimality condition is satisfied or an iterate is repeated. In either case, the algorithm terminates in finite steps.

This paper presents the theory and the algorithm for the subgradient approach to nonlinear integer programming. The methods for computing the supporting planes and solving the linear subproblems are described. Test results for a small set of problems also are given.

The paper is organized as follows: Section 2 introduces the definitions of subgradient and supporting plane and presents the necessary and sufficient optimality condition. Section 3 describes the subgradient algorithm and discusses the stopping criteria, the complexity issues, and the solution properties. The issues on computing supporting planes and solving piecewise linear subproblems are addressed in Sections 4 and 5, respectively. The mathematical formulations are derived, and the methods for solving the subproblems are given. Section 6 describes the numerical test. Section 7 contains concluding remarks.

2 The Nonsmooth Theory

Given a nonlinear objective function $f : R^n \longrightarrow R$, consider the restriction of the function $f : B^n \longrightarrow R$. Let this function be denoted by f^r. It is a function over the discrete set of all 0-1 integer points and is nondifferentiable, or, in other words, nonsmooth.

Problem (1) is equivalent to the following nonsmooth problem:

$$\begin{aligned} \min \quad & f^r(x) \\ x \in \quad & B^n = \{0,1\}^n. \end{aligned} \tag{7}$$

We call f and f^r the continuous and discrete objective functions for Problem (1), respectively.

Definition 1 *A subgradient of f^r at $\bar{x} \in B^n$ is a vector $s \in R^n$ such that*

$$s^T(x - \bar{x}) \le f^r(x) - f^r(\bar{x}) \quad \forall x \in B^n. \tag{8}$$

Definition 2 *The subdifferential of f^r at $\bar{x} \in B^n$ is the set of all subgradients of f^r at $\bar{x}$ defined by the following equation:*

$$\partial f^r(\bar{x}) = \{s \in R^n : \; s^T(x - \bar{x}) \le f^r(x) - f^r(\bar{x}) \quad \forall x \in B^n\}. \tag{9}$$

Definition 3 *A supporting plane of f^r at $\bar{x} \in B^n$ is a hyperplane defined by the following equation:*

$$g(x) = f^r(\bar{x}) + s^T(x - \bar{x}), \qquad s \in \partial f^r(\bar{x}). \tag{10}$$

A supporting plane is said to be "good" if it is tight as a bounding function. For example, in Figure 1, B is better than A, and C is the best. Given a subgradient, we can define a supporting plane, and vice versa. Therefore, the notions of subgradient and supporting plane are correlated.

Theorem 1 *Let f be convex and differentiable, and $\nabla f(\bar{x})$ be the gradient of f at $\bar{x}$. Then, $\nabla f(\bar{x}) \in \partial f^r(\bar{x}) \;\; \forall \bar{x} \in B^n$.*

Proof: It suffices to show that for any $\bar{x} \in B^n$,

$$\nabla f(\bar{x})^T(x - \bar{x}) \le f(x) - f(\bar{x}) \qquad \forall x \in B^n. \tag{11}$$

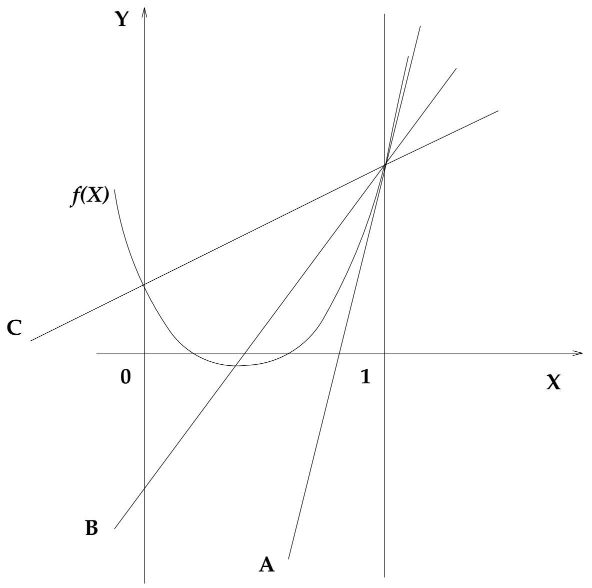

Figure 1: Simple examples for supporting planes

For $x = \bar{x}$, the inequality (11) holds obviously. So, we need to consider only $x \neq \bar{x}$, $x \in B^n$. Since f is differentiable, the directional derivative of f at $\bar{x}$ in the direction of $(x - \bar{x})$, defined as

$$\lim_{\lambda \to 0} \frac{f(\bar{x} + \lambda(x - \bar{x})) - f(\bar{x})}{\lambda}, \tag{12}$$

exists and is equal to $\nabla f(\bar{x})^T (x - \bar{x})$.

Since f is convex, for $\lambda \in (0, 1]$,

$$\begin{aligned} f(x) - f(\bar{x}) &= \frac{\lambda f(x) + (1-\lambda) f(\bar{x}) - f(\bar{x})}{\lambda} \\ &\geq \frac{f(\lambda x + (1-\lambda)\bar{x}) - f(\bar{x})}{\lambda} \\ &= \frac{f(\bar{x} + \lambda(x - \bar{x})) - f(\bar{x})}{\lambda}, \end{aligned} \tag{13}$$

which implies

$$\begin{aligned} f(x) - f(\bar{x}) &\geq \lim_{\lambda \to 0} \frac{f(\bar{x} + \lambda(x - \bar{x})) - f(\bar{x})}{\lambda} \\ &= \nabla f(\bar{x})^T (x - \bar{x}). \end{aligned} \tag{14}$$

□

Theorem 2 *The subdifferential $\partial f^r(\bar{x})$ of f^r at $\bar{x} \in B^n$ is a convex set.*

Proof: For any $\bar{x}$, let $s_1,\ s_2 \in \partial f^r(\bar{x})$. We show that

$$\lambda s_1 + (1-\lambda)s_2 \in \partial f^r(\bar{x}) \qquad \text{for any } \lambda \in [0,1]. \tag{15}$$

Since $s_1,\ s_2 \in \partial f^r(\bar{x})$,

$$\begin{aligned} s_1(x-\bar{x}) &\leq f^r(x) - f^r(\bar{x}) \qquad \forall x \in B^n, \quad \text{and} \qquad (16)\\ s_2(x-\bar{x}) &\leq f^r(x) - f^r(\bar{x}) \qquad \forall x \in B^n. \qquad (17)\end{aligned}$$

So, for any $x \in B^n$ and $\lambda \in [0,1]$,

$$\begin{aligned} & (\lambda s_1 + (1-\lambda)s_2)(x-\bar{x}) \\ =\ & \lambda s_1(x-\bar{x}) + (1-\lambda)s_2(x-\bar{x}) \\ \leq\ & \lambda(f(x)-f(\bar{x})) + (1-\lambda)(f(x)-f(\bar{x})) \\ =\ & f(x) - f(\bar{x}), \qquad (18)\end{aligned}$$

which, by the definition of a subgradient (8), implies

$$\lambda s_1 + (1-\lambda)s_2 \in \partial f^r(\bar{x}) \qquad \text{for any } \lambda \in [0,1]. \tag{19}$$

□

Theorem 3 *A necessary and sufficient condition for $x^* \in B^n$ to be a minimizer of f^r (and also f) over B^n is $0 \in \partial f^r(x^*)$.*

Proof: By the definition of a subgradient (8), $0 \in \partial f^r(x^*)$ for $x^* \in B^n$ if and only if

$$0(x-x^*) \leq f^r(x) - f^r(x^*) \qquad \forall x \in B^n, \tag{20}$$

which just means that

$$f^r(x^*) \leq f^r(x) \qquad \forall x \in B^n. \tag{21}$$

□

Note that the subgradient of a function at a given point may not be unique. Usually, there are infinitely many. No general methods can be used to compute all the subgradients, especially for nonlinear nonsmooth functions. However, as we will see in the following sections, we can determine a zero subgradient without computing the whole set of the subdifferential.

3 The Subgradient Algorithm

A subgradient algorithm, as outlined in Figure 2, solves a nonlinear integer programming problem with the following iterative procedure. The algorithm assumes a starting point $x^{(0)} \in B^n$. At the ith iteration (i starts from 0 to m, an arbitrarily large number), if f^r has a zero subgradient at $x^{(i)}$, or $x^{(i)} = x^{(j)}$ for some $j < i$, then $x^{(i)}$ is an optimal solution, and the algorithm stops. Otherwise, a supporting plane $g_{x^{(i)}}$ for f^r at $x^{(i)}$ is generated. The supporting planes $g_{x^{(j)}}$ for all $j \leq i$ define a piecewise linear function $p(x) = \max_{j \leq i}\{g_{x^{(j)}}(x)\}$. An linear integer subproblem, $\min_{x \in B^n} p(x)$, is then solved, and the solution is used by the algorithm as $x^{(i+1)}$ for next iteration.

The algorithm has two stopping criteria. One is the optimality condition stated in Theorem 3. The other is to test whether an iterate is repeated. We will show in the following that if an iterate is repeated, it must be an optimal solution. This criterion prevents cycling in the algorithm and guarantees that the algorithm will terminate in a finite number of steps.

Theorem 4 *Let $p^{(i)}$ be the piecewise linear function constructed in the ith iteration of Algorithm 1. Then for any i, $p^{(i)}(x) \leq f^r(x) \ \ \forall x \in B^n$.*

Proof: As presented in Algorithm 1,

$$\begin{aligned} p^{(i)}(x) &= \max\{g(x): \ g \in H\} \\ &= \max_{0 \leq j \leq i}\{g^{(j)}(x): \ g^{(j)} \in H\}, \end{aligned} \tag{22}$$

where $g^{(j)} \in H$ is a supporting plane for f^r generated in the jth iteration of the algorithm. By the definition of a supporting plane (10),

$$g^{(j)}(x) = f^r(x^{(j)}) + (s^{(j)})^T(x - x^{(j)}), \tag{23}$$

where $x^{(j)} \in B^n$ is the jth iterate, and $s^{(j)} \in \partial f^r(x^{(j)})$. By the definition of a subgradient (8),

$$(s^{(j)})^T(x - x^{(j)}) \leq f^r(x) - f^r(x^{(j)}) \qquad \forall x \in B^n. \tag{24}$$

So, $g^{(j)}(x) \leq f^r(x) \ \ \forall x \in B^n$. Since this is true for all j, $0 \leq j \leq i$,

$$\max_{0 \leq j \leq i}\{g^{(j)}(x): \ g^{(j)} \in H\} \leq f^r(x) \qquad \forall x \in B^n, \tag{25}$$

which means that $p^{(i)}(x) \leq f^r(x) \ \ \forall x \in B^n$. □

Algorithm 1 $\{$*A subgradient algorithm*$\}$

```
0 {Initialization}
  $T = \phi$, $H = \phi$, $i = 0$
  pick up $x^{(i)} \in B^n$
1 {Iteration}
  do while $i \leq m$
    1.1 {Optimality testing}
        if $x^{(i)} \in T$ or $0 \in \partial f^r(x^{(i)})$ is known then
          $x^{(i)}$ is an optimal solution, stop
        end if
    1.2 {Generating supporting planes}
        $T = T \cup \{x^{(i)}\}$
        $H = H \cup \{g_{x^{(i)}} : \; g_{x^{(i)}}(x) = f^r(x^{(i)}) + s^T_{x^{(i)}}(x - x^{(i)}), \;\; s_{x^{(i)}} \in \partial f^r(x^{(i)})\}$
    1.3 {Solving a linear integer minimax problem}
        find a solution $x^{(*)}$ for
        $\mathbf{min}_{x \in B^n} \{p(x) = \mathbf{max} \{g(x) : \; g \in H\}\}$
    1.4 {Updating}
        $i = i + 1$
        $x^{(i)} = x^{(*)}$
  end do
```

Figure 2: Subgradient algorithm

Theorem 5 *Let $z^{(i)}$ and $z^{(i+1)}$ be the optimal values of the linear integer minimax subproblems in the ith and $(i+1)$th iterations in Algorithm 1, respectively. Then, for any i, $z^{(i)} \leq z^{(i+1)}$. If, in addition, $x^{(i+1)}$ is unique and $x^{(i+1)} \neq x^{(i+2)}$, then $z^{(i)} < z^{(i+1)}$.*

Proof: First we prove $z^{(i)} \leq z^{(i+1)}$.

As defined in Algorithm 1,

$$\begin{aligned} z^{(i)} &= \min_{x \in B^n} p^{(i)}(x), \quad \text{and} & (26) \\ z^{(i+1)} &= \min_{x \in B^n} p^{(i+1)}(x). & (27) \end{aligned}$$

So, it suffices to show that $p^{(i)}(x) \leq p^{(i+1)} \quad \forall x \in B^n$. By the definition of $p^{(i)}$, for any $x \in B^n$,

$$\begin{aligned} p^{(i)}(x) &= \max_{0 \leq j \leq i} \{g^{(j)}(x) : \ g^{(j)} \in H\} \\ &\leq \max_{0 \leq j \leq i+1} \{g^{(j)}(x) : \ g^{(j)} \in H\} \\ &= p^{(i+1)}(x). & (28) \end{aligned}$$

So, $p^{(i)}(x) \leq p^{(i+1)}(x) \quad \forall x \in B^n$, and $z^{(i)} \leq z^{(i+1)}$.

Now we show that $z^{(i)} < z^{(i+1)}$ if $x^{(i+1)}$, the solution to the subproblem

$$\min_{x \in B^n} p^{(i)}(x), \tag{29}$$

is unique, and $x^{(i+1)} \neq x^{(i+2)}$. The proof is by contradiction.

Suppose that Problem (29) has an unique solution $x^{(i+1)}$ and that $z^{(i)} = z^{(i+1)}$. Since $z^{(i)} = z^{(i+1)}$, $p^{(i)}(x^{(i+1)}) = p^{(i+1)}(x^{(i+2)})$. But,

$$\begin{aligned} p^{(i+1)}(x^{(i+2)}) &= \max_{0 \leq j \leq i+1} \{g^{(j)}(x^{(i+2)}) : \ g^{(j)} \in H\} \\ &= \max \{p^{(i)}(x^{(i+2)}), \ g^{(i+1)}(x^{(i+2)})\}. & (30) \end{aligned}$$

Then, $p^{(i)}(x^{(i+1)}) \geq p^{(i)}(x^{(i+2)})$.

However, $p^{(i)}(x^{(i+1)}) < p^{(i)}(x^{(i+2)})$ by the uniqueness of $x^{(i+1)}$ and the fact that $x^{(i+1)} \neq x^{(i+2)}$. Hence, we have a contradiction. Thus, $z^{(i)} \neq z^{(i+1)}$, and $z^{(i)}$ can only be strictly less than $z^{(i+1)}$ by the first argument of the theorem. □

Theorem 6 *Let $T = \{x^{(j)} \in B^n, \ j < i\}$ be the sequence of integer points generated by Algorithm 1 up to the ith iteration. If $\exists j < i$ such that $x^{(j)} = x^{(i)}$, then $x^{(i)}$ must be an optimal solution.*

Proof: Let j be the integer such that $j < i$ and $x^{(j)} = x^{(i)}$.

As defined in the algorithm, $x^{(i)}$ is a solution to the linear integer subproblem

$$\min_{x \in B^n} p^{(i-1)}(x). \tag{31}$$

So,

$$p^{(i-1)}(x^{(i)}) \leq p^{(i-1)}(x) \qquad \forall x \in B^n. \tag{32}$$

By Theorem 4,

$$p^{(i-1)}(x) \leq f^r(x) \qquad \forall x \in B^n. \tag{33}$$

Thus,

$$p^{(i-1)}(x^{(i)}) \leq f^r(x) \qquad \forall x \in B^n \tag{34}$$

and, in particular,

$$p^{(i-1)}(x^{(i)}) \leq f^r(x^{(i)}). \tag{35}$$

But,

$$\begin{aligned} p^{(i-1)}(x^{(i)}) &= \max_{0 \leq k \leq i-1} \{g^{(k)}(x^{(i)}) : \; g^{(k)} \in H\} \\ &\geq g^{(j)}(x^{(i)}) \\ &= g^{(j)}(x^{(j)}) \\ &= f^r(x^{(j)}) \\ &= f^r(x^{(i)}). \end{aligned} \tag{36}$$

Then, $f^r(x^{(i)}) = p^{(i-1)}(x^{(i)})$, and

$$f^r(x^{(i)}) \leq f^r(x) \qquad \forall x \in B^n. \tag{37}$$

□

Theorem 7 *Algorithm 1 is finite.*

Proof: It follows immediately from Theorem 6 and the fact that there are only finitely many distinct points $x \in B^n$. □

Corollary 1 *Let $T = \{x^{(j)} \in B^n,\ j < i\}$ be the sequence of integer points generated by Algorithm 1 up to the ith iteration. Let $z^{(j_i)} = f^r(x^{(j_i)})$ be the minimal of f^r in T. Then,*

$$z^{(i-1)} \leq z^* \leq z^{(j_i)}, \tag{38}$$

and also

$$|z^{(j_i)} - z^{(i-1)}| = 0 \qquad \text{for } i \text{ sufficiently large,} \tag{39}$$

where $z^{(i-1)}$ is defined as in Theorem 5, and z^* is the optimal value of f^r in B^n.

Proof: First we prove the inequalities in (38).

By Theorem 4,

$$p^{(i-1)}(x) \leq f^r(x) \qquad \forall x \in B^n. \tag{40}$$

Therefore,

$$z^{(i-1)} = \min_{x \in B^n} p^{(i-1)}(x) \leq \min_{x \in B^n} f^r(x) = z^*. \tag{41}$$

The second inequality follows from the fact that any feasible point $x \in B^n$ yields an upper bound $f^r(x)$ for the optimal value of f^r.

Now we show the statement (39).

By Theorem 6 and Theorem 7, the algorithm stops at the ith iteration if $x^{(i)} = x^{(j)}$ for some $j < i$. Then as in the proof for Theorem 6,

$$z^{(i-1)} = p^{(i-1)}(x^{(i)}) = f^r(x^{(i)}) = z^*. \tag{42}$$

Since now $z^{(j_i)} = z^*$, $z^{(i-1)} = z^{(j_i)}$. □

Corollary 1 simply implies that at the ith iteration, the algorithm finds the best solution $x^{(j_i)}$ among all iterates. The difference between the objective value at this solution and the optimal value of f^r is bounded by $|z^{(j_i)} - z^{(i-1)}|$, and the bound also decreases with increasing i.

Finally, since it is not straightforward to test the optimality condition in Theorem 3, we state a more constructive, but equivalent, necessary and sufficient condition in the following theorem.

Theorem 8 *A necessary and sufficient condition for $x^* \in B^n$ to be a minimizer of f^r (and also f) over B^n is that $\exists s \in \partial f^r(x^*)$ such that*

$$s_i \leq 0 \qquad \forall i \text{ such that } x_i^* = 1, \tag{43}$$

and

$$s_i \geq 0 \qquad \forall i \text{ such that } x_i^* = 0. \tag{44}$$

Proof: Necessity follows directly from Theorem 3 and the fact that $s = 0$ satisfies conditions (43) and (44). For sufficiency, suppose that $\exists s \in \partial f^r(x^*)$ satisfying conditions (43) and (44). Then,

$$s^T(x - x^*) \leq f^r(x) - f^r(x^*) \qquad \forall x \in B^n, \tag{45}$$

and

$$0 \leq s^T(x - x^*) \qquad \forall x \in B^n. \tag{46}$$

Therefore,

$$\begin{aligned} 0(x - x^*) &\leq s^T(x - x^*) \\ &\leq f^r(x) - f^r(x^*) \qquad \forall x \in B^n, \end{aligned} \tag{47}$$

and then $0 \in \partial f^r(x^*)$. By Theorem 3, x^* is a minimizer of f^r. □

4 Computing the Supporting Planes

In this section, we describe the methods for computing the supporting planes. Related subproblems are derived. The solution methods are given.

4.1 Optimizing a Supporting Plane: A Lifting Process

In the subgradient algorithm, a supporting plane for the discrete objective function at a given integer point is required at each iteration. In general, a supporting plane g for a discrete objective function f^r at a given integer point $\bar{x} \in B^n$ is a linear function, and

$$g(x) = f^r(\bar{x}) + s^T(x - \bar{x}) \qquad s \in \partial f^r(\bar{x}), \qquad x \in R^n. \tag{48}$$

To obtain this function, we need to compute a subgradient $s \in \partial f^r(\bar{x})$ and, in particular, a subgradient such that the supporting plane supports the objective function as tightly as possible. However, ∂f^r usually is not given explicitly, and therefore it is not easy to obtain an arbitrary subgradient.

Note that in principle, given an arbitrary objective function f for Problem (1), we can always replace it with a strictly convex function $\tilde{f}$ without changing the solution of the problem. For example, we can define a function $q : R^n \longrightarrow R$ such that

$$q(x) = \sum_{i=1}^{n} (x_i - 1/2)^{2k} - n/4^k, \tag{49}$$

where $k > 0$ is an integer, and let

$$\tilde{f}(x) = f(x) + \rho q(x), \tag{50}$$

where $\rho > 0$ is a sufficiently large number. Since $q(x) = 0$, $\forall x \in B^n$, $\tilde{f}$ and f agree on all $x \in B^n$. Therefore, they correspond to the same discrete objective function f^r and have the same solution set. However, since q is strictly convex, $\tilde{f}$ is strictly convex for ρ sufficiently large. Therefore, without loss of generality, in the following discussion, we always assume that the objective function f for Problem (1) is strictly convex.

By Theorem 1, if f is convex and differentiable, $\nabla f(\bar{x})$, the gradient of f at $\bar{x}$, is a subgradient of f^r at $\bar{x}$. A trivial way to choose s, therefore, is to set s to $\nabla f(\bar{x})$. Unfortunately, with this subgradient, g usually is too "steep" to be a preferred supporting plane;

To obtain a "better" subgradient, we introduce a method called the lifting process. The process starts with the subgradient $s = \nabla f(\bar{x})$ and then updates it such that the corresponding supporting plane g is "lifted," in other words, made "flatter" or "closer" to f^r. The updated s remains a subgradient as long as g still supports f^r at $\bar{x}$:

$$g(x) \leq f^r(x) \qquad \forall x \in B^n. \tag{51}$$

The lifting process continues until the best possible supporting plane is obtained. However, for every update, the condition (51) must be verified. For a given subgradient s, if S is defined such that $x \in S$ if $f(x) \leq g(x)$, the condition (51) is equivalent to that the interior of S does not contain 0-1 integer points.

Figure 3 illustrates with a simple example how the lifting process is conducted and the condition (51) is guaranteed. In this example, the lifting process is applied to find a subgradient of f^r at $\bar{x}$. First, s is set to $\nabla f(\bar{x})$.

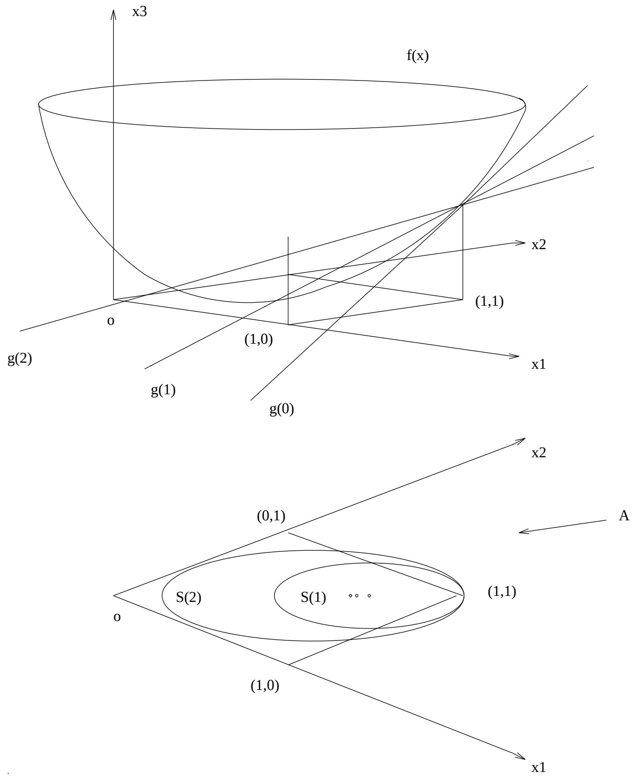

Figure 3: The lifting process for computing subgradients

The supporting plane defined by this subgradient is $g_{(0)}$. Then, s is updated to "lift" $g_{(0)}$ a little bit, and $g_{(1)}$ and $S_{(1)}$ are obtained. Define

$$A = \{x \in R^n : \ x_i \geq 0, \text{ if } \bar{x}_i = 1, \text{ and } x_i \leq 1 \text{ if } \bar{x}_i = 0, \ i = 1, \ldots, n\}. \quad (52)$$

Geometrically, A is a region that contains B^n, and its boundary is formed by hyperplanes $x_i = 1 - \bar{x}_i$, $i = 1, \ldots, n$. Once it has been observed that the interior of A contains no points in B^n other than $\bar{x}$, the condition (51) holds for $g_{(1)}$ if $S_{(1)}$ is inside of A. Therefore, in order to obtain better subgradients, s can further be updated until the corresponding S hits the boundary of A (e.g., $g_{(2)}$ and $S_{(2)}$ in Figure 3).

Let $g_{(0)}$ be the supporting plane of f^r at $\bar{x}$ such that

$$g_{(0)}(x) = f^r(\bar{x}) + \nabla f(\bar{x})^T (x - \bar{x}). \quad (53)$$

Then, we have the following formal definitions and results.

Definition 4 *For any $i > 0$, let $g_{(i-1)}$ be a supporting plane of f^r at $\bar{x}$ and $g_{(i)}$ the supporting plane of f^r obtained by updating the gradient of $g_{(i-1)}$. Then, $g_{(i)}$ is said to be lifted from $g_{(i-1)}$ if $g_{(i)}(x) \geq g_{(i-1)}(x) \ \ \forall x \in B^n$ and there exists at least one point $x \in B^n$ such that $g_{(i)}(x) > g_{(i-1)}(x)$.*

Definition 5 *For any $s \in R^n$, the following set*

$$S = \{x \in R^n : \ f(x) \leq g(x)\} \quad (54)$$

is called the projection set of s on R^n with respect to the function f at $\bar{x}$, where g is defined such that $g(x) = f^r(\bar{x}) + s^T(x - \bar{x})$, $\forall x \in R^n$.

Theorem 9 *For any $s \in R^n$ and convex function f, the projection set S of s with respect to f at $\bar{x}$ is convex.*

Proof: Let $x, \ x' \in S$. We show that

$$\lambda x + (1 - \lambda) x' \in S \qquad \forall \lambda \in [0, 1]. \quad (55)$$

This follows immediately from the fact that

$$\begin{aligned} f(\lambda x + (1-\lambda)x') &\leq \lambda f(x) + (1-\lambda) f(x') \\ &\leq \lambda g(x) + (1-\lambda) g(x') \\ &= g(\lambda x + (1-\lambda)x'), \end{aligned} \quad (56)$$

because that f is convex, $x, \ x' \in S$, and g is linear. □

Theorem 10 *For any $s \in R^n$, $s \in \partial f^r(\bar{x})$ if and only if*

$$x \notin S^\circ \qquad \forall x \in B^n, \tag{57}$$

where S° is the interior of the projection set S of s with respect to f at $\bar{x}$.

Proof: First we prove sufficiency:

If $x \notin S^\circ \; \forall x \in B^n$, $f(x) \geq g(x) \;\; \forall x \in B^n$, where $g(x) = f^r(\bar{x}) + s^T(x - \bar{x})$. This is equivalent to

$$f^r(x) \geq f^r(\bar{x}) + s^T(x - \bar{x}) \qquad \forall x \in B^n. \tag{58}$$

So, $s \in \partial f^r(\bar{x})$ by the definition of a subgradient (8).

Now we prove necessity:

If $s \in \partial f^r(\bar{x})$,

$$s^T(x - \bar{x}) \leq f^r(x) - f^r(\bar{x}) \;\; \forall x \in B^n. \tag{59}$$

Then,

$$g(x) \leq f^r(x) = f(x) \;\; \forall x \in B^n, \tag{60}$$

implying that $x \notin S^\circ \;\; \forall x \in B^n$. □

Theorem 10 provides a necessary and sufficient condition to check whether $s \in \partial f^r(\bar{x})$. While this condition cannot be tested directly, a sufficient condition in Theorem 11 can be used instead in practice.

Theorem 11 *Let A and S be defined as in (52) and (54), respectively. Then, for any $s \in R^n$, $s \in \partial f^r(\bar{x})$ if $S \subseteq A$.*

Proof: It is easy to see that $\bar{x}$ is the only point in B^n contained in A°, the interior of A. Since $S \subseteq A$, $\bar{x}$ is also the only possible point in B^n that can be contained in S°, the interior of S. But, $\bar{x}$ is a boundary point of S. Therefore, $x \notin S^\circ \;\; \forall x \in B^n$, and $s \in \partial f^r(\bar{x})$ by Theorem 9. □

4.2 A Nonlinear Least Squares Formulation

Now consider the updated subgradient s and its corresponding projection set S. Let d_i be the distance between S and the ith boundary of A, $x = \bar{c}_i$, where $\bar{c}_i = 1 - \bar{x}_i$ by the definition of A in (52). Then, $d = (d_1, \ldots, d_n)$ is a function of s, and the lifting process described in the preceding section can be formulated mathematically as a special nonlinear optimization problem:

$$\begin{aligned} \mathbf{min} \quad & \| \, d(s) \, \| \\ \mathbf{st.} \quad & d_i(s) \geq 0 \qquad i = 1, \ldots, n. \end{aligned} \tag{61}$$

Problem (61) does not need to be solved exactly. A feasible solution will suffice, since only a "good-enough" subgradient is sought. The problem can be solved by using the following two relaxation rules:

1. Keep the feasibility while making the objective function value as small as possible but not necessarily optimal.

2. Minimize the objective function while keeping the amount of infeasibility as small as possible but not necessarily zero.

Algorithm 2 (Figure 4) is designed to obtain an approximate solution to Problem (61). First, s is set to an initial value. Then, a "better" s is obtained by updating back and forth each component of s. If $d(s) \geq 0$, s is updated so that the corresponding supporting plane can be "lifted". Otherwise, some components of s are updated to "lower" the supporting plane. The algorithm stops when a good enough subgradient is obtained.

While Algorithm 2 usually requires many updates, another approach is to solve the problem without considering the constraints:

$$\mathbf{min} \quad \| \, d(s) \, \| \, . \tag{62}$$

The solution to this problem, for the case of l_2 norm, can be obtained by a standard nonlinear least squares method [10, 27]. Let s^* be the solution, and $\epsilon = \| \, d(s^*) \, \|$ be the optimal value. Then, we say s^* is an ϵ-approximation to the solution for Problem (61) in the sense that it solves exactly the following problem:

$$\begin{aligned} \mathbf{min} \quad & \| \, d(s) \, \| \\ \mathbf{st.} \quad & d_i(s) + \epsilon \geq 0 \qquad i = 1, \ldots, n. \end{aligned} \tag{63}$$

With this approximation, the total amount of infeasibility caused by s^* is always bounded by a quantity in the order of $O(\epsilon)$. The smaller the ϵ is, the better the solution s^* will be.

```
Algorithm 2 {An update method}

  0 {Initialization}
    for $i = 1, \ldots, n$ do
        set initial values for $s_i$, $\underline{s}_i$, and $\overline{s}_i$
    end do
  1 {Updating}
    if $d_i(s) \geq 0 \;\; \forall i$ then
      if $\| d(s) \|$ small enough, stop
      for $i = 1, \ldots, n$ do
          $\underline{s}_i = s_i$
          $s_i = s_i + (\overline{s}_i - s_i)/2$
      end do
    else
      for $\forall i$ such that $d_i(s) < 0$ do
          $\overline{s}_i = s_i$
          $s_i = s_i - (s_i - \underline{s}_i)/2$
    end if
  2 {Backtracking}
    goto 1
```

Figure 4: An update method

4.3 Computing Extreme Points of a Convex Body

No matter what methods are used to solve Problem (61), the major computation will be the evaluation of the function $d(s)$ for all s. For $i = 1, 2, \ldots, n$, $d_i(s)$ is computed by finding the extreme point of S along the x_i direction and then calculating the distance between the extreme point and the ith boundary of A. The extreme point of S along the x_i direction can be found by solving a constrained optimization problem:

$$\begin{aligned} \textbf{min} \quad & x_i - 2\bar{c}_i x_i \\ \textbf{st.} \quad & x \in S, \end{aligned} \tag{64}$$

or equivalently,

$$\begin{aligned} \textbf{min} \quad & x_i - 2\bar{c}_i x_i \\ \textbf{st.} \quad & f(x) - g(x) \leq 0, \end{aligned} \tag{65}$$

where f, g, and S are defined as before, and $\bar{c}_i = 1 - \bar{x}_i$. This problem is not very hard to solve. Its objective function is linear, and there is only one nonlinear constraint. As we will show below, the solution of this problem is unique, and the first-order necessary condition is also sufficient. Therefore, the solution can be obtained by solving a system of nonlinear equations. In the rest of this section, we discuss mathematical properties related to Problem (64). We describe how to solve the problem in greater detail in next section.

For simplicity, we assume in the following statements that $\bar{x}_i = 1 \ \forall i$. Then, Problem (64) is reduced to

$$\begin{aligned} \textbf{min} \quad & x_i \\ \textbf{st.} \quad & f(x) - g(x) \leq 0. \end{aligned} \tag{66}$$

Lemma 1 *Given $s \in \partial f^r(\bar{x})$ and its corresponding projection set S, if S is closed and bounded, the solution of Problem (66) exists and is unique.*

Proof: Existence can be proved by the fact that the objective function is continuous, and S is closed and bounded.

Now we prove the uniqueness. The proof is by contradiction.

Suppose that x^i and $\hat{x}^i$ are both solutions of Problem (66). Then, $z^i = \lambda x^i + (1-\lambda)\hat{x}^i$ for any $\lambda \in (0,1)$ is also a solution, because $\hat{x}^i_i = x^i_i$, and

$$\begin{aligned} z^i_i &= \lambda x^i_i + (1-\lambda)\hat{x}^i_i \\ &= \lambda x^i_i + (1-\lambda)x^i_i \\ &= x^i_i. \end{aligned} \tag{67}$$

However, since f is strictly convex and x^i, $\hat{x}^i \in S$,

$$\begin{aligned} f(z^i) &< \lambda f(x^i) + (1-\lambda) f(\hat{x}^i) \\ &\leq \lambda g(x^i) + (1-\lambda) g(\hat{x}^i) \\ &= g(z^i), \end{aligned} \tag{68}$$

which implies that z^i is an interior point of S. This is a contradiction to the fact that any solution of Problem (66) is an extreme point of S. □

Lemma 2 *For Problem (66), define a Lagrangian function*

$$l^i(x; u^i) = x_i + u^i(f(x) - g(x)), \tag{69}$$

where u^i is a scalar. Then, for any i, $0 \leq i \leq n$, a necessary and sufficient condition for x^i to be an optimal solution to Problem (66) is that $\exists u^i \geq 0$ such that

$$\begin{aligned} \nabla_x l^i(x^i; u^i) &= 0 \\ u^i(f(x^i) - f(\bar{x}) - s^T(x^i - \bar{x})) &= 0 \\ f(x^i) - f(\bar{x}) - s^T(x^i - \bar{x}) &\leq 0, \end{aligned} \tag{70}$$

which, with $\nabla_x l^i(x^i; u^i)$ written explicitly, is equivalent to

$$\begin{aligned} u^i(f'_{x_1}(x^i) - s_1) &= 0 \\ u^i(f'_{x_2}(x^i) - s_2) &= 0 \\ &\vdots \\ u^i(f'_{x_{i-1}}(x^i) - s_{i-1}) &= 0 \\ 1 + u^i(f'_{x_i}(x^i) - s_i) &= 0 \\ u^i(f'_{x_{i+1}}(x^i) - s_{i+1}) &= 0 \\ &\vdots \\ u^i(f'_{x_n}(x^i) - s_n) &= 0 \\ u^i(f(x^i) - f(\bar{x}) - s^T(x^i - \bar{x})) &= 0 \\ (f(x^i) - f(\bar{x}) - s^T(x^i - \bar{x})) &\leq 0. \end{aligned} \tag{71}$$

Proof: Necessity follows directly from the first-order necessary condition for a nonlinear constrained optimization problem [12, 15, 31].

Note that the ith equation of (71) implies that $u^i > 0$, and $\nabla_x^2 l^i(x^i; u^i) = u^i \nabla^2 f(x^i)$ is positive definite. So, the necessary condition is also sufficient by the second-order sufficiency condition for a nonlinear constrained optimization problem [12, 15, 31]. $\square$

Lemma 3 *Given $\hat{s} \in R^n$ and its corresponding projection set $\hat{S}$, let $\hat{x}^i$ be an extreme point of $\hat{S}$ along the x_i direction, then there is a neighborhood $N(\hat{s},\ \epsilon)$ of $\hat{s}$ and a function $x^i:\ R^n \longrightarrow R^n$ continuous and differentiable in $N(\hat{s},\ \epsilon)$ such that*

$$\hat{x}^i = x^i(\hat{s}). \tag{72}$$

Proof: Rewrite (71) in the following way:

$$\begin{array}{rcl} F_1(x^i; u^i; s) & = & u^i(f'_{x_1}(x^i) - s_1) = 0 \\ F_2(x^i; u^i; s) & = & u^i(f'_{x_2}(x^i) - s_2) = 0 \\ & \vdots & \\ F_{i-1}(x^i; u^i; s) & = & u^i(f'_{x_{i-1}}(x^i) - s_{i-1}) = 0 \\ F_i(x^i; u^i; s) & = & 1 + u^i(f'_{x_i}(x^i) - s_i) = 0 \\ F_{i+1}(x^i; u^i; s) & = & u^i(f'_{x_{i+1}}(x^i) - s_{i+1}) = 0 \\ & \vdots & \\ F_n(x^i; u^i; s) & = & u^i(f'_{x_n}(x^i) - s_n) = 0 \\ H(x^i; u^i; s) & = & (f(x^i) - f(\bar{x}) - s^T(x^i - \bar{x})) = 0, \end{array} \tag{73}$$

where the last equality holds because $u^i > 0$ from the ith equality and

$$u^i(f(x^i) - f(\hat{x}) - s^T(x^i - \hat{x})) = 0. \tag{74}$$

So, the inequality in (71) is removed.

Let $F = (F_1,\ F_2,\ \ldots,\ F_n, H)$. Then (73) is equivalent to

$$F(x^i; u^i; s) = 0. \tag{75}$$

By Lemmas 1 and 2, given $\hat{s}$, $\exists \hat{x}^i$ and $\hat{u}^i$ such that $F(\hat{x}^i; \hat{u}^i; \hat{s}) = 0$. And $\hat{x}^i$, $\hat{u}^i$ are also unique.

Differentiate F with respect to x^i and u^i,

$$\begin{aligned}
\nabla_{(x^i;u^i)} F^T &= \begin{pmatrix} \nabla_{(x^i;u^i)} F_1 \\ \nabla_{(x^i;u^i)} F_2 \\ \vdots \\ \nabla_{(x^i;u^i)} F_n \\ \nabla_{(x^i;u^i)} H \end{pmatrix} \\
&= \begin{pmatrix}
\frac{\partial^2 f}{\partial x_1 \partial x_1} u^i & \dots & \frac{\partial^2 f}{\partial x_1 \partial x_i} u^i & \dots & \frac{\partial^2 f}{\partial x_1 \partial x_n} u^i & 0 \\
\vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\
\frac{\partial^2 f}{\partial x_i \partial x_1} u^i & \dots & \frac{\partial^2 f}{\partial x_i \partial x_i} u^i & \dots & \frac{\partial^2 f}{\partial x_i \partial x_n} u^i & f'_{x_i} - s_i \\
\vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\
\frac{\partial^2 f}{\partial x_n \partial x_1} u^i & \dots & \frac{\partial^2 f}{\partial x_n \partial x_i} u^i & \dots & \frac{\partial^2 f}{\partial x_n \partial x_n} u^i & 0 \\
 & & & & & \\
0 & \dots & f'_{x_i} - s_i & \dots & 0 & 0
\end{pmatrix} \\
&= \begin{pmatrix} u^i \nabla^2 f & \nabla h \\ \nabla h^T & 0 \end{pmatrix}, \qquad (76)
\end{aligned}$$

where $\nabla h = (0,\ \dots,\ f'_{x_i} - s_i,\ 0,\ \dots,\ 0)^T$.

Using the fact that $\nabla^2 f$ is positive definite and $\nabla h \neq 0$, since $f'_{x_i} - s_i = -1/u^i$, it is not difficult to verify that $\nabla_{(x^i;u^i)} F$ is nonsingular at $(\hat{x}^i; \hat{u}^i; \hat{s})$. Therefore, by the implicit function theorem, the lemma is proved. □

From Lemmas 1, 2, and 3, we obtain the following theorem:

Theorem 12 *Given $\hat{s} \in R^n$ and its corresponding projection set $\hat{S}$, let $\hat{x}^i$ be an extreme point of $\hat{S}$ along the x_i direction. Let $d = (d_1,\ d_2,\ \dots,\ d_n)^T$ and $\hat{d} = (\hat{x}_1^1 - \bar{c}_1, \hat{x}_2^2 - \bar{c}_2,\ \dots,\ \hat{x}_n^n - \bar{c}_n)^T$. Then, there is a neighborhood $N(\hat{s},\ \epsilon)$ of $\hat{s}$ and a function $d :\ R^n \longrightarrow R^n$ continuous and differentiable in $N(\hat{s},\ \epsilon)$ such that $\hat{d} = d(\hat{s})$.*

Proof: Simply set $d_i(s) = x_i^i(s) - \bar{c}_i$ and then apply Lemma 3. □

4.4 Solving the Constrained Subproblems

In this section, we discuss the techniques for solving the constrained optimization subproblem (64). The solution to such a problem is required to evaluate $d_i,\ i = 1, \dots, n$ in the least squares problem (61). We will again only consider the formulation (66). The results can be extended to general cases.

A constrained optimization problem can be expensive to solve [12, 15]. However, the problem (66) has a special structure. It is a problem with a linear objective function and a nonlinear convex constraint. Also, as shown in the preceding section, the solution to the problem is unique and can be obtained by solving a system of nonlinear equations:

$$F(x;u) = \begin{pmatrix} \nabla_x l(x;u) \\ h(x) \end{pmatrix} = 0, \tag{77}$$

where $l(x;u)$ is the Lagrangian function for the problem (66), u is the Lagrangian multiplier, and $h(x) = f(x) - g(x)$.

Note that the Jacobian of $F(x;u)$ is

$$\nabla F(x;u)^T = \begin{pmatrix} \nabla_x^2 l(x;u) & \nabla h(x) \\ \nabla^T h(x) & 0 \end{pmatrix}, \tag{78}$$

and $\nabla_x^2 l(x,u)$ is symmetric positive definite (the proof for Lemma 2). Therefore, to solve the system (77), we can use the quasi-Newton method with a structural BFGS secant update by [30, 9]. Also, in computing the Newton step, we can take advantage of this special property to solve each linear system efficiently.

By a secant method for solving a system of nonlinear equations

$$F(x) = 0, \tag{79}$$

where $F : R^n \longrightarrow R^n$, we mean the iterative procedure

$$\begin{aligned} Bs &= -F(x) && (80) \\ x_+ &= x + s && (81) \\ B_+ &= \mathcal{B}(x, s, y, B), && (82) \end{aligned}$$

where s is the quasi-Newton step, $y = F(x_+) - F(x)$, and B_+ is required to satisfy the secant equation

$$B_+ s = y, \tag{83}$$

where B_+ is an approximate to the first-order information for $F(x_+)$ and is obtained by updating B with a process called the secant update. Among various kinds of secant updates, the BFGS update is in some sense the most effective one. However, it requires the Jacobian to be symmetric positive definite.

Often, in practice, a part of the first order information is available, and we need only to approximate the rest. This kind of secant approximation is referred to as the structural secant update, for the special structure of the problem is taken into account. The structural BFGS update approximates the unknown part of the first order information using the BFGS update, but computes the available part exactly.

Now consider the system (77). Part of its first order information, $\nabla h(x)$, can be computed exactly, while $\nabla_x^2 l(x;u)$ needs to be approximated. Since $\nabla_x^2 l(x;u)$ is symmetric positive definite, we can apply the structural BFGS update. Therefore, a secant method for the system (79) can be formulated as the following iterative procedure:

$$B = \begin{pmatrix} B_l & \nabla h(x) \\ \nabla h(x)^T & 0 \end{pmatrix} \tag{84}$$

$$Bs = -F(x;u) \tag{85}$$

$$(x_+;u_+) = (x;u) + s \tag{86}$$

$$B_{l+} = B_l + \frac{yy^T}{y^Ts} - \frac{B_l s s^T B_l}{s^T B_l s} \tag{87}$$

$$B_+ = \begin{pmatrix} B_{l+} & \nabla h(x_+) \\ \nabla h(x_+)^T & 0 \end{pmatrix}. \tag{88}$$

Note that in the above procedure, a linear system

$$Bs = -F \tag{89}$$

needs to be solved for each update, where

$$B = \begin{pmatrix} B_l & \nabla h \\ \nabla h^T & 0 \end{pmatrix} \tag{90}$$

with B_l being symmetric positive definite. We can solve this system as follows:

First, let $s = (x;\alpha)^T$ and $-F = (y;\beta)^T$, where $x, y \in R^n$, and α and β are scalars. Rewrite the system as

$$B_l x + \nabla h \alpha = y \tag{91}$$

$$\nabla h^T x = \beta. \tag{92}$$

Solve this system for x and α to obtain

$$
\begin{aligned}
x &= B_l^{-1}(y - \nabla h \alpha) && (93) \\
\alpha &= \frac{\nabla h^T B_l^{-1} y - \beta}{\nabla h^T B_l^{-1} \nabla h}. && (94)
\end{aligned}
$$

This is equivalent to the following steps:

$$
\begin{aligned}
B_l a &= \nabla h && (95) \\
B_l b &= y && (96) \\
x &= b - \nabla h \alpha && (97) \\
\alpha &= \frac{\nabla h^T b - \beta}{\nabla h^T a}, && (98)
\end{aligned}
$$

where, since B_l is symmetric positive definite, the equations (95) and (96) can be solved by using the Cholesky factorization [18].

5 Solving the Linear Integer Minimax Subproblems

We now discuss how to solve the linear integer minimax subproblems in the subgradient algorithm. The problems are formulated in the following form:

$$
\mathbf{min}_{x \in B^n} \quad \{p^{(i)}(x) = \mathbf{max}\ \{g_{x(j)}(x),\ j = 0,\ \ldots,\ i\}\}, \qquad (99)
$$

or, equivalently,

$$
\begin{aligned}
\mathbf{min} \quad & \eta && (100) \\
\mathbf{st.} \quad & \eta \geq g_{x(j)}(x) \qquad j = 0,\ \ldots,\ i && (101) \\
& 1 \geq x \geq 0, \quad x \ \text{ integral}, && (102)
\end{aligned}
$$

where $g_{x(j)}$ is the jth linear supporting function generated by the algorithm, and i indicates that the problem is the one in the ith iteration.

The above problem is a linear integer programming problem with one continuous variable and can be solved with an enumeration method [1, 2]. Also, observe that $p^{(i+1)}$ is generated by adding one more supporting plane to $p^{(i)}$, which implies that problems in the ith and $(i+1)$th iterations are almost the same except that the problem in the $(i+1)$th iteration has one more constraint. Therefore, to solve the $(i+1)$th problem, we can use the result from solving the ith problem to reduce the total computation.

We present a branch-and-bound procedure for solving the problem (100). A branch-and-bound procedure, as illustrated in Algorithm 3 (Figure 5), solves the problem (100) as follows: First, a relaxed problem, the problem without integrality constraints, is solved. If a 0-1 integer solution x is obtained, the algorithm terminates, and x is an optimal solution. Otherwise, x_j for some j is set to 1 or 0, and two corresponding subproblems are generated. Recursively, for each subproblem, again, a relaxed problem is solved. If the optimal value of the relaxed problem is larger than a known upper bound for the optimal value of the original problem, the subproblem is eliminated, and is not considered any more. If a 0-1 integer solution is obtained, the solution is locally optimal to the original problem. An upper bound for the optimal value of the original problem is obtained. Otherwise, the subproblem is divided, and two more subproblems are generated. The process continues until all subproblems are either eliminated or solved. Among all 0-1 solutions obtained for the subproblems, the one that yields the smallest objective value is the optimal solution to the original problem [3, 13, 26].

Let the relaxed problem for the problem (100) be the following:

$$\textbf{min} \quad \eta \tag{103}$$

$$\textbf{st.} \quad \eta \geq g_{x^{(j)}}(x) \qquad j = 0, \ldots, i \tag{104}$$

$$1 \geq x \geq 0. \tag{105}$$

In Algorithm 3, $p = (p_1, p_2, \ldots, p_n)$ represents the problem that is the same as the relaxed problem (103) except that some x_j's are set to 1 or 0, where

$$p_j = 1 \quad \text{if and only if} \quad x_j \quad \text{is set to} \quad 1 \tag{106}$$

$$p_j = 0 \quad \text{if and only if} \quad x_j \quad \text{is set to} \quad 0 \tag{107}$$

$$p_j = \wedge \quad \text{otherwise.} \tag{108}$$

Also, P is a stack, and **push** and **pop** are two standard stack operations.

In Algorithm 3, when a locally optimal solution is found, an upper bound for the optimal value is obtained. Let the upper bound be denoted by $\overline{z}^{(i)}$. Then, since a solution that provides an objective value better than $\overline{z}^{(i)}$ is always desired, the strategy to choose a branching variable is to try the variable that may reduce the current objective value if it is set to 1. Write a problem p in the following form:

$$\textbf{min} \quad \eta \tag{109}$$

Algorithm 3 {*Solving integer minimax problems*}

```
0 {Initialization}
  $p = (\wedge, \wedge, \ldots, \wedge)$, $P = \phi$, push$(p, P)$
  $\underline{z}_p^{(i)} = -\infty$,  $\overline{z}^{(i)} = \min_{0 \le j \le i} \{f(x^{(j)})\}$
1 {Iteration}
  do while $P \neq \phi$
    1.1 {Problem selection and relaxation}
        solve $p =$ pop$(P)$
        let $\underline{z}_p^{(i)}$ be the optimal value
        let $x_p^{(i)}$ be the optimal solution
    1.2 {Prunning}
        if $\underline{z}_p^{(i)} \ge \overline{z}^{(i)}$, go to next loop
        if $x_p^{(i)}$ is integral, $\overline{z}^{(i)} = \min(\overline{z}^{(i)}, \underline{z}_p^{(i)})$ and go to next loop
    1.3 {Branching}
        pick up $x_j$ for some $j$ with $p_j = \wedge$
        set $p_j = 0$, push$(p, P)$
        set $p_j = 1$, push$(p, P)$
  end do
2 {Termination}
  the solution $x_p^{(i)}$ for some $p$ that yields $\overline{z}^{(i)}$ is optimal
```

Figure 5: Solving integer minimax problems

$$\textbf{st.} \quad \eta \geq b_j + a_{j1}x_1 + \ldots + a_{jm}x_m \tag{110}$$

$$j = 0, \ \ldots, \ i \tag{111}$$

$$1 \geq x \geq 0, \tag{112}$$

assuming $p_k = \wedge \ \ \forall k = 1, \ \ldots, \ m$. Then, a branching variable x_k for this problem is selected if k solves the problem:

$$\min_{1 \leq k \leq m} \{ \max_{0 \leq j \leq i} \{ b_j + a_{jk} \} \}. \tag{113}$$

The relaxed problems can be solved by using a standard dual simplex method [4, 5]. Let $p^{(i)}$ and $p^{(i+1)}$ be the relaxed problems for the ith and $(i+1)$th linear integer minimax subproblems, respectively. As we have mentioned before, $p^{(i+1)}$ is the same as $p^{(i)}$ except that it has one more constraint. The dual optimal basis for $p^{(i)}$ is dual feasible for $p^{(i+1)}$. Therefore, $p^{(i+1)}$ can be solved with the dual optimal basis for $p^{(i)}$ as its initial basis. Also, if p is a problem, and s is its subproblem, s and p have a similar relationship as $p^{(i+1)}$ and $p^{(i)}$: s is the same as p except that s contains one more constraint $x_j = 1/0$ for some j. So, the dual optimal basis for p can be used as the initial basis for s.

6 Preliminary Test Results

A program has been written to test the subgradient algorithm with the following problems:

#1:
Objective function:
$\sum_{i=1}^{n} x_i^2 - 1.8 \sum_{i=1}^{n} x_i + 0.81n$
Optimal solution: $x_i = 1, \ i = 1, \ldots, n$

#2:
Objective function:
$2 \sum_{i=1}^{n} x_i^2 + \sum_{i=1}^{n-1} x_i x_{i+1} - 2 \sum_{i=1}^{n/2} (1.9x_{2i-1} + 1.1x_{2i}) + 1.205n$
Optimal solution: $x_{2i-1} = 1, \ x_{2i} = 0, \ i = 1, \ldots, n/2$

#3:
Objective function:
$\sum_{i=1}^{n} (x_i - 0.9)^{8/3}$

Optimal solution: $x_i = 1,\ i = 1, \ldots, n$

#4:
Objective function:
$\sum_{i=1}^{n} x_i^2 - 0.8 \sum_{i=1}^{n} x_i + 0.16n$
Optimal solution: $x_i = 0,\ i = 1, \ldots, n$

These problems are constructed so that the problem dimensions are scalable and the optimal solutions are easy to verify. Problem #1 is a simple problem with a separable objective function. Problem #2 represents an unseparable case. The objective function for Problem #3 is more complicated in a certain way. Problem #4 is a geometrically very symmetric problem. It exemplifies the case when the subgradient algorithm sometimes may require many steps to converge.

The test was conducted on a parallel machine nCUBE located at Caltech. However, the parallel implementation of the algorithm will not be addressed in this paper. Interested readers are referred to [32] for more details. The test was not a complete one in terms of both problem types and sizes. Many implementation details still need to be worked out before more numerical studies can be made. Nevertheless, the results obtained through this simple implementation have demonstrated that the subgradient algorithm is very effective for solving most of the test problems.

Listed in Table 1 to 7 are samples of the test results. The tables contain the problem dimensions, the initial guesses, and the numbers of iterations required to find an optimal solution. For a problem of dimension n, there are 2^n number of 0-1 feasible points. Therefore, in the worst case, the algorithm may need to search through all these points for an optimal solution. However, the algorithm actually solved the test problems in only a few iterations. The major reason is because that the algorithm can determine the optimal solution as soon as it is generated, using the necessary and sufficient condition. This fact is not true for all traditional combinatorial methods, for example, the branch-and-bound method, which cannot determine an optimal solution until all necessary feasible points are enumerated. The initial guesses (or, in other words, the starting points) do make differences in required numbers of iterations: Some lead to an optimal solution in a few steps, while others may need more steps. However, the mathematics for why this is the case and how one can choose good starting points has not yet been fully understood.

Table 1 contains the results for Problem #1 of dimension 8 with different

Table 1: Test problem: #1

n	Initial Guess	Iteration
8	(0,0,0,0,0,0,0,0)	2
8	(0,0,0,0,0,0,0,1)	7
8	(0,0,0,0,0,0,1,1)	3
8	(0,0,0,0,0,1,1,1)	3
8	(0,0,0,0,1,1,1,1)	2
8	(0,0,0,1,1,1,1,1)	3
8	(0,0,1,1,1,1,1,1)	2
8	(0,1,1,1,1,1,1,1)	7
8	(1,1,1,1,1,1,1,1)	1
8	(0,1,0,1,0,1,0,1)	2
8	(0,0,1,1,0,0,1,1)	2
8	(0,1,1,1,0,1,1,1)	2
8	(0,0,0,1,0,0,0,1)	3

Table 2: Test problem: #1

n	Initial Guess	Iteration
8	(0,. . .,0)	2
16	(0,. . .,0)	2
32	(0,. . .,0)	2

Table 3: Test problem: #2

n	Initial Guess	Iteration
8	(0,0,...,0,0)	3
16	(0,1,...,0,1)	2
32	(0,0,...,0,0)	3

Table 4: Test problem: #3

n	Initial Guess	Iteration
8	(0,0,0,0,0,0,0,0)	2
8	(0,0,0,0,0,0,0,1)	2
8	(0,0,0,0,0,0,1,1)	2
8	(0,0,0,0,0,1,1,1)	2

Table 5: Test problem: #3

n	Initial Guess	Iteration
16	(0,...,0,0,0,0,0,0)	2
16	(0,...,0,0,0,0,0,1)	2
16	(0,...,0,0,0,0,1,1)	2
16	(0,...,0,0,0,1,1,1)	3
16	(0,...,0,0,1,1,1,1)	2
16	(0,...,0,1,1,1,1,1)	2

Table 6: Test problem: #3

n	Initial Guess	Iteration
32	(0,. . .,0,1)	3
32	(0,1,. . .,1)	2
64	(0,. . .,0)	2
64	(1,. . .,1)	1

Table 7: Test problem: #4

n	Initial Guess	Iteration
4	(0,0,0,0)	13
4	(0,0,0,1)	11
4	(0,0,1,0)	11
4	(0,0,1,1)	13
4	(0,1,0,0)	11
4	(0,1,0,1)	13
4	(0,1,1,0)	13
4	(0,1,1,1)	11
4	(1,0,0,0)	11
4	(1,0,0,1)	13
4	(1,0,1,0)	13
4	(1,0,1,1)	11
4	(1,1,0,0)	13
4	(1,1,0,1)	11
4	(1,1,1,0)	11
4	(1,1,1,1)	13

starting points. Only up to 7 iterations were used by the algorithm to find an optimal solution. When the optimal solution was used as the starting point, the algorithm was able to determine the solution immediately by finding a zero subgradient for the objective function at this solution, and therefore, it only took one iteration.

Tables 2 and 3 contain the results for Problem #1 and #2 with different dimensions. We see that with the same staring point, the numbers of iterations did not change very much as problem dimensions increased.

Tables 4, 5, and 6 contain the results for Problem 3 with different starting points and problem dimensions. We did not enumerate all possible starting points (there are too many). In any case, we can conclude from these results that with a reasonable starting point, the algorithm can find an optimal solution for this problem in only a few iterations. The biggest problem for this case is of dimension 64. The total number of integer points is 2^{64}, a huge number. However, the subgradient algorithm did not need to enumerate many of these points, but stopped immediately after an optimal solution was reached.

We understand that the nonlinear integer programming problem is $\mathcal{NP}$-hard, and therefore, it is impossible to find both an efficient and a general solution [14, 23]. Hence we do not expect the subgradient algorithm to be able to solve all the problems efficiently, either. Problem #4 shows that the subgradient algorithm cannot find an optimal solution in a small number of steps. From Table 7, we see that for most starting points, the algorithm enumerated almost all the integer points. The reason is that the problem is geometrically very symmetric. The "lifting" process failed to find good-enough supporting planes, and hence the zero subgradient for the optimal solution. Therefore, the algorithm was not able to stop even if the optimal solution was found. The algorithm continued until the second optimality condition was satisfied (i.e., an iterate was repeated). However, because of the symmetry, the second condition also required many steps.

Finally, we point out that we also tested Problem #1 ($n = 8$) with the subgradient in each iteration simply set to the gradient of the continuous objective function. With this choice of subgradient, the algorithm was not able to find an optimal solution even in more than 25 iterations. This result shows the significance of computing good subgradients and hence supporting planes by the lifting process.

7 Concluding Remarks

We have presented a new algorithm for nonlinear integer programming. Our approach is based on considering the problem as a nonsmooth problem and using the subgradient information to linearize the nonlinear objective function. The subgradient algorithm searches for an optimal solution iteratively through integer points, and in each iteration, it generates the next point by solving the problem for a local piecewise linear model, constructed with the supporting planes for the objective function at a set of integer points already generated. In order to determine whether an iterate is optimal, two optimality testing criteria have been established and are employed in the algorithm. One is the necessary and sufficient condition for the optimal solution. It is established with standard nonsmooth function theories. The other one is based on a combinatorial property: an iterate is optimal if it is repeated. This one, in particular, is useful to keep the algorithm finite.

We have discussed how to compute the supporting planes using special continuous optimization techniques. Problem formulations and related mathematical results are presented, and numerical methods are given.

The piecewise linear subproblem can be solved by using standard linear integer programming methods. However, a special branch-and-bound procedure is designed that exploits the problem structure and hence can solve the subproblem efficiently.

A program has been written to test the subgradient algorithm. Preliminary results show that the algorithm solves most of the test problems effectively.

Several aspects of this work may be extended. The most important one is a more complete numerical test. This will not be a trivial job because a good implementation of the algorithm requires considerable work. In addition, few test problems are available, while artificial problems usually are too arbitrary to make any specific conclusions. However, a relatively complete test on a class of quadratic 0-1 integer programs is feasible and will be very useful for understanding basic numerical properties of the algorithm. A performance comparison between the algorithm and a traditional method, say, a branch-and-bound method, can also be conducted.

We have mentioned that any problem can be formulated so that its continuous objective function is strictly convex. However, we have not yet given a truly practical method to convert a given function, for example, how to choose an appropriate ρ in (50). More work needs to be done on this subject.

The subgradient algorithm can be extended to mixed-integer nonlinear programming. However, we have not studied possible successes and limitations of applying the algorithm to this class of problems.

The technique for computing the supporting planes described in this paper can be improved by replacing the lifting process with a procedure that can compute integer lattice-free convex bodies. If such an "oracle" exists, even better supporting planes can be obtained, and the algorithm will be more effective. Finding such an "oracle" itself is an important research issue in integer programming [24, 25].

Acknowledgments

The author is grateful to Professors John E. Dennis and Robert E. Bixby for many discussions relating to this work, and for their support, encouragement, and guidance during his Ph.D. study.

References

[1] E. Balas and J. B. Mazzola [1984a]. *Nonlinear 0-1 Programming: I. Linearization Techniques.* Mathematical Programming 30, 1–21.

[2] E. Balas and J. B. Mazzola [1984b]. *Nonlinear 0-1 Programming: II. Dominance Relations and Algorithms.* Mathematical Programming 30, 22–45.

[3] R. E. Bixby [1987]. *Notes on Combinatorial Optimization.* Technical Report TR87-21, Dept. of Math. Sci., Rice Univ., Houston.

[4] R. E. Bixby [1990]. *Implementing the Simplex Method: The Initial Basis.* Technical Report TR90-32, Dept. of Math. Sci., Rice Univ., Houston.

[5] V. Chvátal [1980]. *Linear Programming.* W. H. Freeman and Company, New York.

[6] F. H. Clarke [1983]. *Optimization and Nonsmooth Analysis.* John Wiley, New York.

[7] Y. Crama, P. Hansen and B. Jaumard [1990]. *The Basic Algorithm for Pseudo-Boolean Programming Revisited.* Discrete Applied Mathematics.

[8] G. B. Dantzig [1960]. *On the Significance of Solving Linear Programming Problems with Some Integer Variables.* The Rand Corporation, Document P1486.

[9] J. E. Dennis, Jr., H. J. Martinez and R. A. Tapia [1989]. *A Convergence Theory for the Structured BFGS Secant Method with an Application to Nonlinear Least Squares.* Journal of Optimization Theory and Applications 61, 159–176.

[10] J. E. Dennis, Jr., and R. B. Schnabel [1983]. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Prentice-Hall, Englewood Cliffs, N.J.

[11] P. van Emde-Boas [1981]. *Another NP-Complete Partition Problem and the Complexity of Computing Short Vectors in a Lattice.* Report 81-04, Mathematical Institute, Univ. of Amsterdam, Amsterdam.

[12] R. Fletcher [1987]. *Practical Methods of Optimization.* John Wiley & Sons, New York.

[13] R. S. Garfinkel and G. L. Nemhauser [1972]. *Integer Programming.* John Wiley & Sons, Inc., New York.

[14] M. R. Garey and D. S. Johnson [1979]. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, New York.

[15] P. E. Gill, W. Murray, and M. H. Wright [1981]. *Practical Optimization.* Academic Press, New York.

[16] F. Glover and E. Woolsey [1973]. *Further Reduction of Zero-One Polynomial Programs to Zero-One Linear Programming Problems.* Operations Research 21(1), 156–161.

[17] F. Glover and E. Woolsey [1974]. *Converting the 0-1 Polynomial Programming Problems to a 0-1 Linear Program.* Operations Research 22, 180–182.

[18] G. H. Golub and C. F. Van Loan [1989]. *Matrix Computations.* The Johns Hopkins Univ. Press, Baltimore.

[19] M. Grötschel, L. Lovász, and A. Schrijver [1987]. *Geometric Algorithm and Combinatorial Optimization.* Springer, New York.

[20] P. L. Hammer, I. Rosenberg and S. Rudeanu [1963]. *On the Determination of the Minima of Pseudo-Boolean Functions.* (In Romanian) Studii si Cercetari Matematice 14, 359–364.

[21] P. L. Hammer and S. Rudeanu [1968]. *Boolean Methods in Operations Research and Related Areas.* Springer, New York.

[22] P. Hansen, B. Jaumard, V. Mathon [1989]. *Constrained Nonlinear 0-1 Programming.* RRR #47-89, RUTCOR, Rutgers Univ., New Brunswick, N.J.

[23] J. E. Hopcroft and J. D. Ullman [1979]. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley Publishing Company, Reading, MA.

[24] L. Lovász [1986]. *An Algorithmic Theory of Numbers, Graphs and Convexity.* CBMS-NSF Regional Conference Series in Applied Mathematics 50. SIAM, Philadelphia.

[25] L. Lovász [1989]. *Geometry of Numbers and Integer Programming.* M. Iri and K. Tanabe (eds.), *Mathematical Programming,* 177–201, KTK Scientific Publisher, Tokyo.

[26] G. L. Nemhauser and L. A. Wolsey [1988]. *Integer and Combinatorial Optimization.* John Wiley & Sons, New York.

[27] J. M. Ortega and W. C. Rheinboldt [1970]. *Iterative Solution of Nonlinear Equations in Several Variables.* Academic Press, New York.

[28] R. T. Rockafellar [1970]. *Convex Analysis.* Princeton University Press, Princeton, N.J.

[29] N. Z. Shor [1985]. *Minimization Methods for Non-Differentiable Functions.* Springer-Verlag, New York.

[30] R. Tapia [1988]. *On Secant Updates for Use in General Constrained Optimization.* Mathematics of Computation 51, 181–202.

[31] R. Tapia [1990]. *An Introduction to the Algorithms and Theory of Constrained Optimization.* Lecture Notes, Dept. of Math. Sci., Rice Univ., Houston.

[32] Z. Wu [1991]. *A Subgradient Algorithm for Nonlinear Integer Programming and Its Parallel Implementation.* Ph.D. thesis, Dept. of Math. Sci., Rice Univ., Houston.