A Nontangential Cutting Plane Algorithm

by

Siriphong Lawphongpanich

June 2001

Approved for public release; distribution is unlimited.

Prepared for: Naval Postgraduate School
Monterey, California 93943
This report was prepared for and funded by the Naval Postgraduate School. Reproduction of all or part of this report is authorized.

This report was prepared by:

Siriphong Lawphongpanich
Associate Professor of Operations Research

Reviewed by:

R. KEVIN WOOD
Associate Chairman for Research
Department of Operations Research

Released by:

JAMES M. EAGLE
Chairman
Department of Operations Research

DAVID W. NETZER
Associate Provost and Dean of Research
A Nontangential Cutting Plane Algorithm

Siriphong Lawphongpanich

Naval Postgraduate School
Monterey, CA 93943

N/A

NPS-OR-01-008

N/A

Approved for public release; distribution is unlimited.

The cutting plane algorithm typically generates cuts that are tangential, or nearly so, to the Lagrangian dual function of the underlying optimization problem. This paper demonstrates that the algorithm still converges to an optimal solution when cuts are nontangential. These cuts are generated by not solving the subproblems to optimality or nearly so. Computational results from randomly generated linear and quadratic programming problems indicate that nontangential cuts can lead to a more efficient algorithm.

Cutting Plane Algorithm, Decomposition, Large Scale Systems

Unclassified

Unclassified

Unclassified

Unclassified

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std 239-18
A Nontangential Cutting Plane Algorithm

Siriphong Lawphongpanich
Operations Research Department
Naval Postgraduate School
Monterey, California 93943

May 2001

Abstract:

A cutting plane algorithm for continuous optimization problems typically generates cuts that are tangential, or nearly so, to the Lagrangian dual function of the underlying optimization problem. This paper demonstrates that the algorithm still converges to an optimal solution when cuts are nontangential. These cuts are generated by not solving the subproblems to optimality or nearly so. Computational results from randomly generated linear and quadratic programming problems indicate that nontangential cuts can lead to a more efficient algorithm.

Keywords: Cutting Plane Algorithm, Decomposition, Large-Scale Systems

1. Introduction

Consider the following optimization problem:

\[
P: \quad f^* = \min f(x) \\
\text{s.t.} \quad g(x) \leq 0 \\
x \in X,
\]

where \( g(x) = [g_1(x), \ldots, g_m(x)]^T \). In addition, \( f(x) \) and \( g_p(x), p = 1, \ldots, m \), are convex functions, and \( X \) is a nonempty compact subset of \( \mathbb{R}^n \). For convenience, assume that Slater's constraint qualification (see, e.g., Bazaraa et al. [1993]) holds, i.e., there exists a point \( x_0 \in X \) such that \( g(x_0) < 0 \).

A dual of problem \( P \) is

\[
D: \quad L^* = \max L(u) \\
\text{s.t.} \quad u \geq 0 \text{ and } u \in \mathbb{R}^m,
\]

* This research was partially supported by the Naval Postgraduate School Institutionally Funded Research Program.
where \( L(u) = \min_{x \in X} \{ f(x) + u g(x) \} \) and \( x y \) denotes the usual dot product between two vectors \( x \) and \( y \). As defined, \( L(u) \) is the Lagrangian dual function associated with problem \( P \). One method for solving problem \( D \) is the cutting plane algorithm (CPA) and below is a typical version (e.g., Bazaraa et al. [1993]).

The Cutting Plane Algorithm

**Step 0:** Find a point \( x_0 \in X \) such that \( g(x_0) < 0 \). Set \( k = 1 \) and go to Step 1.

**Step 1:** Solve the following (master) problem:

\[
\begin{align*}
\text{M}[k]: \quad & \max \quad w \\
\text{s.t.} \quad & w \leq f(x_i) + u g(x_i), \ \forall i = 0, \ldots, (k-1), \\
& u \geq 0.
\end{align*}
\]

Let \((w_k, u_k)\) denote an optimal solution and go to Step 2.

**Step 2:** Solve the following (sub)problem:

\[
\begin{align*}
\text{S}[u_k]: \quad & L(u_k) = \min_{x \in X} \{ f(x) + u g(x) \}
\end{align*}
\]

If \( w_k = L(u_k) \), stop and \( u_k \) is an optimal solution to \( D \). Otherwise, let \( x_k \) denote an optimal solution to \( S[u_k] \), replace \( k \) with \( k + 1 \), and go to Step 1.

The master problem in Step 1 is a linear program for which there exists a finite algorithm, e.g., the simplex algorithm (e.g., Dantzig and Thapa [1997]). For the subproblem in Step 2, a typical convergence proof for CPA requires an optimal solution. In practice, many would employ CPA only when the subproblem has a closed form solution or is easy to solve, for example, with an algorithm that terminates after performing only a small number of iterations. When a finite algorithm does not exist, several articles (e.g., Zakeri et al. [2000] and references cited therein) indicate that CPA would generate an approximate solution to problem \( D \) in a finite number of iterations if the subproblem is solved to near optimality, i.e., \( \varepsilon \)-optimality. In some cases, it may be necessary for \( \varepsilon_k \to 0 \), as \( k \to \infty \).

The approach in this paper is different, in that it does not attempt to obtain an optimal or near optimal solution to the subproblem. Instead, the algorithm applied to the subproblem is terminated or truncated after a predetermined number of iterations, \( r \geq 1 \).
When \( r \) is small, the resulting solution is far from being optimal to the subproblem. Moreover, truncating the algorithm before it reaches an optimal subproblem solution results in cuts, i.e., hyperplanes defined by the master problem constraints (1), that are not necessarily tangential to \( L(u) \).

For the remainder, Section 2 describes a nontangential cutting plane algorithm and proves its convergence, and Section 3 presents results from a computational study to illustrate the advantage of nontangential cuts.

2. A Nontangential Cutting Plane Algorithm

The nontangential cutting plane algorithm (NCPA) stated below uses an algorithmic map to solve the subproblem. As in Zangwill [1969], let \( I(x, u) \) denote a mapping that maps a point \((x, u) \in X \times U \) to a subset of \( X \), where, in our context, \( X \) is as defined previously and \( U = \{ u : u \geq 0 \text{ and } u \in R^m \} \). Then, an algorithm for the subproblem is an iterative process that begins with a feasible point, \( x_0 \), and generates a sequence of points \( \{x_k\} \) recursively using the recursion \( x_k \in I(x_{k-1}, u) \).

A nontangential cutting plane algorithm

**Step 0**: Find a point \( x_0 \in X \) such that \( g(x_0) < 0 \). Set \( k = 1 \) and go to Step 1.

**Step 1**: Solve the master problem, \( M[k] \). Let \((w_k, u_k, \pi^k)\) denote its optimal primal and dual solutions and go to Step 2.

**Step 2**: Let \[ y_k = \sum_{i=0}^{k-1} \pi_i^k x_i \] and \( x_k \in I(y_k, u_k) \). If \( w_k = f(x_k) + u_k g(x_k) \), stop and \( u_k \) is an optimal solution to \( D \). Otherwise, replace \( k \) with \( k + 1 \), and go to Step 1.

With the exception of requiring an optimal dual solution, \( \pi^k \), to the master problem in Step 1, the first two steps of NCPA are the same as those in CPA. Instead of solving the subproblem optimally or nearly so, \( x_k \) in Step 2 is the result of applying an algorithmic map to \((y_k, u_k)\) once. In practice, it may be more efficient to apply the algorithm map recursively several times. However, once is enough to establish convergence.
In Step 2, the initial solution, \( y_k \), for the algorithmic map is a convex combination of \( x_k \), \( k = 0, \ldots, (k - 1) \), and feasible to problem P. The former is true because \( \mathcal{X} \) is optimal to the dual of the master problem stated below.

\[
\text{DM}[k]: \quad \min \sum_{i=0}^{k-1} \pi_i f(x_i) \\
\text{s.t.} \quad \sum_{i=0}^{k-1} \pi_i g(x_i) \leq 0, \\
\sum_{i=0}^{k-1} \pi_i = 1, \\
\pi_i \geq 0, \forall \ i = 0, \ldots, (k-1).
\]

The feasibility follows from the convexity assumption for each component of \( g(x) \). In particular, the following holds because \( g_p(x) \) is convex.

\[
g_p(y_k) = g_p \left( \sum_{i=0}^{k-1} \pi_i^k x_i \right) \leq \sum_{i=0}^{k-1} \pi_i^k g_p(x_i) \leq 0, \forall \ p = 1, \ldots, m.
\]

To establish convergence for NCPA, assume that the algorithmic map \( \mathcal{I}(x,u) \) satisfies the following convergence conditions similar to those in Zangwill [1969]:

a) \( \mathcal{I}(x,u) \) is closed for any point \( (x, u) \) such that \( x \) is not a solution to the subproblem \( S[u] \), i.e., \( \min_{x \in X} \{ f(x) + u g(x) \} \).

b) If \( y \in X \) is not a solution of problem \( S[u] \), then \( f(x) + u g(x) < f(y) + u g(y) \) for every \( x \in \mathcal{I}(y,u) \). When \( y \in X \) is a solution, \( f(x) + u g(x) = f(y) + u g(y) \) \( \forall \ x \in \mathcal{I}(y,u) \).

The first part of condition (b) ensures that the new cut eliminates \( (w_k, u_k) \) from the feasible region of the next master problem, \( M[k + 1] \). Under these two conditions, the following theorem justifies the stopping criterion in Step 2.

**Theorem 1:** If \( w_k = f(x_k) + u_k g(x_k) \), then \( u_k \) solves problem D and \( y_k \) solves problem P.

**Proof:** Recall from the above discussion that \( y_k \) is feasible to problem P. Because \( u_k \) is feasible to problem \( M[k] \), it must be nonnegative, thereby feasible to problem D.
The complementary slackness conditions in linear programming ensure that \( w_k = f(x_i) + u_k g(x_i) \) for all \( i \) such that \( \pi_i^k > 0 \) and \( i = 0, \ldots, (k-1) \). Combining this fact with the convexity of \( f(x) \) and \( g_p(x) \) and the convergence condition (b) yields the following:

\[
\begin{align*}
    w_k &= \sum_{i=0}^{k-1} \pi_i^k \left( f(x_i) + u_k g(x_i) \right), \\
    &\geq f\left( \sum_{i=0}^{k-1} \pi_i^k x_i \right) + u_k \left( \sum_{i=0}^{k-1} \pi_i^k x_i \right), \\
    &= f(y_k) + u_k g(y_k), \\
    &\geq f(x_k) + u_k g(x_k).
\end{align*}
\]

Since the theorem assumes that \( w_k = f(x_k) + u_k g(x_k) \), it follows from the above sequence of equations that \( w_k = f(x_k) + u_k g(x_k) = f(y_k) + u_k g(y_k) \). However, the convergence condition (b) further guarantees that \( y_k \) solves \( S[u_k] \), i.e.,

\[ L(u_k) = f(y_k) + u_k g(y_k) = w_k. \tag{2} \]

Because \( M[k] \) and \( DM[k] \) must have the same objective value at optimality, the following must hold:

\[ w_k = \sum_{i=0}^{k-1} \pi_i^k f(x_i) \geq f(y_k), \tag{3} \]

where the inequality follows from our convexity assumption for \( f(x) \). Combining (2) and (3) yields that \( L(u_k) \geq f(y_k) \). On the other hand, the weak duality theorem (e.g., Bazaraa et al. [1993]) ensures that \( L(u_k) \leq f(y_k) \). So, \( L(u_k) = f(y_k) \), i.e., the primal, \( y_k \), and dual, \( u_k \), solutions have the same objective value, and the strong duality theorem (e.g., Bazaraa et al. [1993]) guarantees that both solutions must be optimal to their respective problems.\[ \]

From Theorem 1, \( y_k \) and \( u_k \) are optimal to their respective problems when NCPA terminates after a finite number of iterations. When it does not, NCPA generates sequences \( \{u_k\}, \{w_k\}, \{x_k\} \) and \( \{y_k\} \) with the following properties:

\begin{enumerate}
    \item[c)] \( w_{k-1} \geq w_k \geq L^* \),
    \item[d)] \( f(x_k) + u_k g(x_k) < f(y_k) + u_k g(y_k) \).
\end{enumerate}
The first follows from the fact that $M[k]$ contains more cuts than $M[k-1]$ and every master problem is a relaxation of problem $D$. The second is due to the convergence condition (b).

The theorem below addresses the convergence of $\{y_k\}$ and $\{u_k\}$.

**Theorem 2:** Assume that there exists a point $x_0 \in X$ such that $g(x_0) < 0$ and $\Gamma(x,u)$ satisfies the two convergence conditions. If NCPA does not terminate after a finite number of iterations, then there exists an index set $\Omega \subseteq \{0, 1, 2, \ldots\}$ such that the sequences $\{y_k\}_{k \in \Omega}$ and $\{u_k\}_{k \in \Omega}$ converge to optimal solutions for problems $P$ and $D$, respectively.

**Proof:** Zangwill [1969] shows that every $u_k$ lies in a compact set under the first assumption. Therefore, there exists an index set $\Omega$ such that the subsequence $\{u_k\}_{k \in \Omega}$ converges to $u_\infty$.

Because $(w_k, u_k)$ solves $M[k]$, the following holds:

$$f(x_i) + u_k g(x_i) \geq w_k, \quad \forall \ i = 0, \ldots, (k-1).$$

From property (c), $\{w_k\}$ is a monotonically nonincreasing sequence and bounded below. Thus, $\{w_k\}$ must converge to $w_\infty$. Taking the limit in (4) for $k \in \Omega$ yields

$$f(x_i) + u_\infty g(x_i) \geq w_\infty, \quad \forall \ i \geq 0.$$ (5)

Since $X$ is compact and $x_i \in X$, there must exist a subsequence $\Omega_1 \subseteq \Omega$ for which $\{x_i\}_{i \in \Omega_1}$ converges to $x_\infty$. Now, taking the limit in (5) for $i \in \Omega_1$ gives

$$f(x_\infty) + u_\infty g(x_\infty) \geq w_\infty.$$ (6)

From the proof of Theorem 1, $w_k \geq f(y_k) + u_k g(y_k)$. Using a similar argument, there must exist a subsequence $\Omega_2 \subseteq \Omega_1$ that leads to the following:

$$w_\infty \geq f(y_\infty) + u_\infty g(y_\infty).$$ (7)

Combining (6) and (7) produces

$$f(x_\infty) + u_\infty g(x_\infty) \geq f(y_\infty) + u_\infty g(y_\infty).$$ (8)

If $y_\infty$ is not optimal to $S[u_\infty]$, then convergence condition (a) ensures that $x_\infty \in \Gamma(y_\infty, u_\infty)$ and $f(x_\infty) + u_\infty g(x_\infty) < f(y_\infty) + u_\infty g(y_\infty)$ which contradicts (8). Therefore, $y_\infty$ must be a
solution of $S[u]$, i.e., $L(u) = f(y) + u_s g(y_s) = f(x) + u_s g(x_s)$. From (6) and (7), it follows that $L(u) = w$, which, as in Theorem 1, implies that $y$ and $u_s$ are optimal to problems P and D, respectively.[]

3. Computational Results

This section summarizes computational results on two sets of randomly generated problems. One set is quadratic and the other is linear. We implement CPA and NCPA

using GAMS version 2.50 (Brooke et al. [1998]). With one exception (described below), we use CPLEX version 6.5 (ILOG [1999]) with default settings to solve linear problems and MINOS version 5.04 (Murtagh and Saunders [1995]), also with default settings, to solve nonlinear ones. All CPU times reported here are from a 500 MHz Pentium III computer with 384 MB of RAM and Windows NT version 4.0 (see, e.g., Solomon [1998]) operating system.

Quadratic Problems

In this set of problems, the functions in problem P are of the form $f(x) = (Q_0 x + Q_0 x) + c x, g(x) = (Q_p x + Q_p x) + c_p x + d_p, p = 1, \ldots, m$, and the set $X = \{x: x_j \geq 0\}$.

We use a procedure similar to the one described in Rosen and Suzuki [1965] to generate data for these functions. Letting $U[a, b]$ denote uniform random numbers between $a$ and $b$, the procedure can be stated as follows:

**Step 1:** Let elements of matrix $Q_p$, $p = 0, \ldots, m$, and vector $c_p, p = 1, \ldots, m$, be $U[-5, 5]$ and $U[-3, -1]$, respectively.

**Step 2:** Let elements of optimal primal, $x^*$, and dual, $(u^*, v^*)$, solutions to be $U[0, 2]$ and $U[0, 5]$, respectively. Then, adjust $v^*$ so that $x^* v^*_j = 0, j = 1, \ldots, n$, and choose $d_p, p = 1, \ldots, m$, to satisfy the complementary slackness conditions: $g_p(x^*) u^*_p = 0, p = 1, \ldots, m$.

**Step 3:** Set $c_0 = v^* - 2Q_0^T Q_0 x^* - \sum_{p=1}^m u^*_p (2Q^T_p Q_p x^* + c_p)$. 

7
The expression for \( c_0 \) in Step 3 ensures that \((x^*, u^*, v^*)\) satisfies the Karush-Kuhn-Tucker conditions (e.g., Bazaraa et al. [1993]) for the convex quadratic program defined by matrices \( Q_p \), vectors \( c_p \), and constants \( d_p \).

Table 1 compares iterates from CPA and NCPA when solving a quadratic problem generated using the above procedure. The problem has 20 variables and 10 constraints. For CPA, the table lists the following information at the end of iteration \( k \):

- **Master problem**: The optimal objective value, \( w_k \), the associated gap value, which is the difference between \( w_k \) and \( L^* \) as a percentage of the latter (i.e., gap = \( 100 \times (w_k - L^*)/L^* \)), and the number of iterations (see the column labeled ‘iter’ in the table) and CPU seconds (see the column labeled ‘sec’ in the table) required to solve each master problem to optimality using CPLEX.

- **Subproblem**: The number of iterations (see the column labeled ‘iter’ in the table) and CPU seconds (see the column labeled ‘sec’ in the table) required to solve each subproblem by MINOS until the default optimality tolerance (set at 1.0E-6) is satisfied.

- **Total time** (see the column labeled ‘Total sec’ in the table) spent solving the master and subproblem.

Except for the gap value column under the subproblem heading, Table 1 also provides the same information for NCPA. For this quadratic problem, we allow MINOS to perform at most five iterations when ‘solving’ the subproblem in NCPA. The subproblem gap value is the percent difference between the optimal subproblem objective value and the one obtained after five MINOS iterations. Observe that the subproblem gap values for NCPA decrease (not necessarily in a monotonic fashion) from 79.80% to nearly zero as the iterations progress. As the sequences \( \{y_k\} \) and \( \{u_k\} \) converge to optimal primal and dual solutions, \( y_k \), for \( k \) sufficiently large, must be nearly optimal to the subproblem at iteration \( k \), i.e., \( \min_{x \in \mathcal{X}} \{ f(x) + u g(x) \} \). So, regardless of the number of iterations performed, convergence condition (b) ensures that \( x_k \) is also nearly optimal to the subproblem for sufficiently large \( k \). Thus, the condition automatically controls the quality of the subproblem solutions without using a sequence \( \{ \varepsilon_k \} \) that converges to zero.
<table>
<thead>
<tr>
<th>$k$</th>
<th>$w_k$</th>
<th>gap</th>
<th>iter</th>
<th>sec</th>
<th>it</th>
<th>sec</th>
<th>Total</th>
<th>$w_k$</th>
<th>gap</th>
<th>iter</th>
<th>sec</th>
<th>it</th>
<th>sec</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-114.90</td>
<td>99.90</td>
<td>2</td>
<td>0.0470</td>
<td>28</td>
<td>0.0293</td>
<td>0.0763</td>
<td>-1037.52</td>
<td>99.06</td>
<td>2</td>
<td>0.0470</td>
<td>79.80</td>
<td>5</td>
<td>0.0195</td>
</tr>
<tr>
<td>2</td>
<td>-446.19</td>
<td>99.60</td>
<td>2</td>
<td>0.0470</td>
<td>22</td>
<td>0.0391</td>
<td>0.0861</td>
<td>-2424.31</td>
<td>97.81</td>
<td>2</td>
<td>0.0470</td>
<td>58.06</td>
<td>5</td>
<td>0.0293</td>
</tr>
<tr>
<td>3</td>
<td>-982.24</td>
<td>99.11</td>
<td>2</td>
<td>0.0310</td>
<td>18</td>
<td>0.0508</td>
<td>0.0818</td>
<td>-5081.23</td>
<td>95.40</td>
<td>2</td>
<td>0.0470</td>
<td>47.36</td>
<td>5</td>
<td>0.0293</td>
</tr>
<tr>
<td>4</td>
<td>-1323.94</td>
<td>98.80</td>
<td>2</td>
<td>0.0320</td>
<td>22</td>
<td>0.0488</td>
<td>0.0808</td>
<td>-6013.53</td>
<td>94.56</td>
<td>2</td>
<td>0.0470</td>
<td>49.20</td>
<td>5</td>
<td>0.0410</td>
</tr>
<tr>
<td>5</td>
<td>-2842.32</td>
<td>97.43</td>
<td>3</td>
<td>0.0470</td>
<td>23</td>
<td>0.0703</td>
<td>0.1173</td>
<td>-9836.21</td>
<td>91.09</td>
<td>5</td>
<td>0.0470</td>
<td>32.69</td>
<td>5</td>
<td>0.0312</td>
</tr>
<tr>
<td>6</td>
<td>-2961.76</td>
<td>97.32</td>
<td>3</td>
<td>0.0470</td>
<td>14</td>
<td>0.0488</td>
<td>0.0958</td>
<td>-13612.48</td>
<td>87.68</td>
<td>2</td>
<td>0.0470</td>
<td>43.60</td>
<td>5</td>
<td>0.0293</td>
</tr>
<tr>
<td>7</td>
<td>-3316.43</td>
<td>97.00</td>
<td>3</td>
<td>0.1250</td>
<td>22</td>
<td>0.0605</td>
<td>0.1855</td>
<td>-21985.81</td>
<td>80.09</td>
<td>10</td>
<td>0.0310</td>
<td>59.65</td>
<td>5</td>
<td>0.0312</td>
</tr>
<tr>
<td>8</td>
<td>-5425.06</td>
<td>95.09</td>
<td>3</td>
<td>0.0310</td>
<td>21</td>
<td>0.0605</td>
<td>0.0915</td>
<td>-24482.28</td>
<td>77.83</td>
<td>13</td>
<td>0.0470</td>
<td>47.37</td>
<td>5</td>
<td>0.0293</td>
</tr>
<tr>
<td>9</td>
<td>-7614.05</td>
<td>93.11</td>
<td>2</td>
<td>0.0470</td>
<td>21</td>
<td>0.0605</td>
<td>0.1075</td>
<td>-27065.69</td>
<td>75.50</td>
<td>5</td>
<td>0.0470</td>
<td>30.61</td>
<td>5</td>
<td>0.0488</td>
</tr>
<tr>
<td>10</td>
<td>-10088.74</td>
<td>90.87</td>
<td>5</td>
<td>0.0470</td>
<td>20</td>
<td>0.0586</td>
<td>0.1056</td>
<td>-45729.55</td>
<td>58.60</td>
<td>5</td>
<td>0.0630</td>
<td>23.15</td>
<td>5</td>
<td>0.0312</td>
</tr>
<tr>
<td>20</td>
<td>-39127.93</td>
<td>64.57</td>
<td>18</td>
<td>0.0470</td>
<td>24</td>
<td>0.0801</td>
<td>0.1271</td>
<td>-102357.00</td>
<td>7.33</td>
<td>19</td>
<td>0.0310</td>
<td>0.05</td>
<td>5</td>
<td>0.0312</td>
</tr>
<tr>
<td>28</td>
<td>-100367.00</td>
<td>9.13</td>
<td>19</td>
<td>0.0470</td>
<td>22</td>
<td>0.0605</td>
<td>0.1075</td>
<td>-109452.00</td>
<td>0.91</td>
<td>25</td>
<td>0.0460</td>
<td>0.23</td>
<td>5</td>
<td>0.1504</td>
</tr>
</tbody>
</table>
Total | 798 | 2.3900 | 1059 | 4.1895 | 6.5795 | 391 | 1.311 | 140 | 1.1855 | 2.4965 |

Table 1: Computational results for a quadratic problem with 20 variables and 10 constraints.
In Table 1, the values of \( w_k \) from NCPA also converges to \( L^* \) faster than those from CPA. Without the requiring the cuts to be tangential to the Lagrangian dual function, nontangential cutting planes can make deeper cuts as Figure 1 illustrates. In the figure, the master objective value due to the tangential cuts is \( w_1 \) and the one for the nontangential cuts is smaller at \( w_2 \). Overall, NCPA requires fewer iterations and less CPU time to achieve a solution with a 1% gap or less. In Table 1, the total time required to solve the master and subproblems for CPA (≈ 6.58 sec.) is more than 2.5 times the one for NCPA (≈ 2.50 sec.).

![Figure 1: Tangential and nontangential cuts](image)

Table 2 summarizes results from solving 25 random quadratic problems of various sizes. For each problem size (identified by the number of variables and constraints), we generate five random problems and solve them by the two methods until the gap is less than or equal to 1%. As in the above problem with 20 variables and 10 constraints, the maximum number of iterations, \( r \), allowed for the subproblem in NCPA is five. For each method, Table 2 reports the average gap value achieved, number of iterations, and CPU times spent solving the master and subproblems.
<table>
<thead>
<tr>
<th>var</th>
<th>con</th>
<th>Cutting Plane Algorithm</th>
<th>Nontangential Cutting Plane Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Master Problem</td>
<td>Subproblem</td>
</tr>
<tr>
<td></td>
<td></td>
<td>gap</td>
<td>iter</td>
</tr>
<tr>
<td>20</td>
<td>5</td>
<td>0.76</td>
<td>219</td>
</tr>
<tr>
<td>40</td>
<td>10</td>
<td>0.87</td>
<td>1106</td>
</tr>
<tr>
<td>60</td>
<td>15</td>
<td>0.97</td>
<td>2876</td>
</tr>
<tr>
<td>80</td>
<td>20</td>
<td>0.91</td>
<td>4821</td>
</tr>
<tr>
<td>100</td>
<td>25</td>
<td>0.98</td>
<td>8040</td>
</tr>
</tbody>
</table>

Table 2: Computational results for randomly generated quadratic problems.

<table>
<thead>
<tr>
<th>mc</th>
<th>sc</th>
<th>sv</th>
<th>Cutting Plane Algorithm</th>
<th>Nontangential Cutting Plane Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Master Problem</td>
<td>Subproblem 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>gap</td>
<td>iter</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>40</td>
<td>0.91</td>
<td>49</td>
</tr>
<tr>
<td>20</td>
<td>40</td>
<td>80</td>
<td>0.89</td>
<td>132</td>
</tr>
<tr>
<td>30</td>
<td>60</td>
<td>120</td>
<td>0.90</td>
<td>246</td>
</tr>
<tr>
<td>40</td>
<td>80</td>
<td>160</td>
<td>0.94</td>
<td>355</td>
</tr>
<tr>
<td>50</td>
<td>100</td>
<td>200</td>
<td>0.92</td>
<td>621</td>
</tr>
<tr>
<td>200</td>
<td>400</td>
<td>800</td>
<td>0.98</td>
<td>9362</td>
</tr>
</tbody>
</table>

Note: mc = number of constraints in the master problem, sc = number of constraints in each subproblem, sv = number of variables in each subproblem. From these values, the numbers of constraints and variables in the monolithic problem are (mc + 2×sc) and (2×sv), respectively.

Table 3: Computational results for randomly generated linear problems.
In general, NCPA requires fewer iterations and less CPU time for both the master and subproblems. Because we restricted the number of iterations for the subproblem to be no more than five for NCPA, it is no surprise that NCPA uses fewer iterations and less CPU time on the subproblem. On the other hand, the results for the master problems in Table 2 suggest that those with nontangential cuts are easier to solve as well. Finally, the last column in the same table gives ratios of the two total CPU times, those for NCPA over those for CPA, and they range from 0.55 to 0.09. In other words, the saving due to the nontangential cuts ranges from 45% for small quadratic problems to 91% for large ones.

**Linear Problems**

Problems in this set are random linear programs of the form \( \min \{ cx : Ax \leq b, x \geq 0 \} \), where

\[
A = \begin{bmatrix}
A_{11} & A_{12} \\
A_{21} & 0 \\
0 & A_{32}
\end{bmatrix}.
\]

With respect to problem \( P, f(x) = cx \), \( g(x) = [A_{11}: A_{12}]x - b_1 \), and \( X = \{ x : [A_{21}; 0]x \leq b_2, [0: A_{32}]x \leq b_3, \text{ and } x \geq 0 \} \). For our experiments, elements of \( A_{11} \) and \( A_{12} \) are \( U[-1, 5] \), those for \( A_{21} \) and \( A_{32} \) are \( U[-1, 10] \), and the optimal primal and dual solutions are \( U[0, 5] \). The remaining data were chosen so that the primal and dual solutions satisfy the Karush-Kuhn-Tucker conditions in a manner similar to the procedure described above. (Note that the optimal primal and dual solutions generated in this manner are usually not basic.)

We also replaced problem \( M[k] \) with problem \( DM[k] \) in Step 1. Doing so reduces CPA to Dantzig-Wolfe decomposition [1960]. Moreover, the structure of the set \( X \) allows the subproblem, \( S[k] \), to separate into two independent linear programs.

In Step 2 of NCPA, \( y_k \) is not necessarily an extreme point. This makes it difficult to warm start CPLEX with a basic feasible solution. One simple way to resolve this is to treat each subproblem in NCPA as a nonlinear problem and let MINOS perform at most \( r \)
iterations. Unlike quadratic problems, setting $r$ to five results in ‘shallow’ cuts and, as a consequence, NCPA requires too many master iterations to arrive at a solution with 1% gap. For the results reported below, we first solve the problems by CPA. For NCPA, we set $r$ to be approximately 50% of the minimum number of iterations required to solve each subproblem in CPA.

Table 3 reports the results for linear problems. These results are similar to those for the quadratic problems in that NCPA requires fewer iterations and less CPU time to arrive a solution with no more than 1% gap. As in the quadratic case, the saving due to the nontangential cuts ranges from 53% for small linear problems to 81% for large ones.

References:


INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center ................................................................. 2
   8725 John J. Kingman Rd., STE 0944
   Ft. Belvoir, VA  22060-6218

2. Research Office (Code 09) ................................................................. 1
   Naval Postgraduate School
   Monterey, CA  93943-5000

3. Dudley Knox Library (Code 013) .......................................................... 2
   Naval Postgraduate School
   Monterey, CA  93943-5002

4. Prof. Sirimuth Lawphongpanich (Code OR/Gv) ........................................ 5
   Dept of Operations Research
   Naval Postgraduate School
   Monterey, CA  93943-5000

5. Richard Mastowski (Editorial Assistant) ............................................... 2
   Dept of Operations Research
   Naval Postgraduate School
   Monterey, CA  93943-5000