Submitted to *INFORMS Journal on Computing*
manuscript JOC-2017-12-OA-200

Online Supplement for "A Criterion Space Method for Biobjective Mixed Integer Programming: the Boxed Line Method"

Tyler Perini, Natashia Boland, Diego Pecin, Martin Savelsbergh
H. Milton Stewart School of Industrial and Systems Engineering
Georgia Institute of Technology, Atlanta

1. Pseudocode of basic variant of the Boxed Line Method

Algorithm 1 BasicOuterLoop

Input: Objective functions $z_1(x), z_2(x)$ and feasible set X are *fixed and global*
Output: N the entire nondominated frontier as a set of line segments, Q any unexplored regions of the frontier space (for partial run-time)

$z^L \leftarrow \text{lexmin}\{(z_1, z_2), IP\}$ is the UL nondominated point
$z^R \leftarrow \text{lexmin}\{(z_2, z_1), IP, ifs = z^L.solution\}$ is the BR nondominated point
$N \leftarrow point(z^L) \cup point(z^R)$
if $\min_{i=1,2}\{|z_i^L - z_i^R|\} < \epsilon$ **then**
  $Q \leftarrow \emptyset$ no region to explore
  return (N, Q)
else
  $Q \leftarrow B(z^L, z^R)$ the unexplored region defined by box with corner points z^L, z^R
end if
while $Q \neq \emptyset$ **do**
  $B(z^L, z^R) \leftarrow element(Q)$
  $Q \leftarrow setminus(Q, B(z^L, z^R))$
  $\mu \in (z_2^R, z_2^L)$ arbitrary horizontal dividing line between z^L, z^R
  $z^* \leftarrow \text{lexmin}\{(z_1, z_2) : z_2(x) \leq \mu, IP, ifs = z^R.solution\}$ find NDP in lower half
  if $\mu - z_2^* > \epsilon$ **then**
    if $\min_{i=1,2}\{|z_i^* - z_i^R|\} < \epsilon$ **then**
      $N \leftarrow N \cup point(z^*)$ if epsilon-close, add to ND frontier, do not add region to queue
    else
      $Q \leftarrow Q \cup B(z^*, z^R)$ otherwise, add a new unexplored region to the queue
    end if
    $\hat{z} \leftarrow \text{lexmin}\{(z_2, z_1) : z_1(x) \leq z_1^* - \epsilon, IP, ifs = z^L.solution\}$
    if $\min_{i=1,2}\{|\hat{z}_i - z_i^L|\} < \epsilon$ **then**
      $N \leftarrow N \cup point(\hat{z})$
    else
      $Q \leftarrow Q \cup B(z^L, \hat{z})$
    end if
  else $\{|z_2^* - \mu| < \epsilon\}$
    $(z^1, z^2, z1\_open, z2\_open, M) \leftarrow$ BasicInnerLoop(z^*, z^L, z^R)
    if $\min_{i=1,2}\{|z_i^1 - z_i^2|\} < \epsilon$ **then**
      $N \leftarrow N \cup M \cup point(z^1) \cup point(z^2)$ M *only* includes NDPs that dominate z^1 or z^2
    else
      $N \leftarrow N \cup M \cup line(z^1, z^2)$
    end if
    if $z1\_open$ **then**
      $\hat{z}^1 \leftarrow \text{argmin}\{m_2 : m \in M \cup z^L, m_2 \geq z_2^1\}$ NDP found from Inner Loop that dominates z^1
    else $\{z^1$ is closed$\}$
      $\hat{z}^1 \leftarrow z^1$ closed endpoint as LR boundary of unexplored region
    end if
    if $\min_{i=1,2}\{|z_i^L - \hat{z}_i^1|\} < \epsilon$ **then**
      $N \leftarrow N \cup point(z^L) \cup point(\hat{z}^1)$ add to frontier
    else
      $Q \leftarrow Q \cup B\,(z^L, \hat{z}^1)$ add to queue
    end if
    if $z2\_open$ **then**
      $\hat{z}^2 \leftarrow \text{argmin}\{m_1 : m \in M \cup z^R, m_1 \geq z_1^2\}$ NDP found from Inner Loop that dominates z^2
    else $\{z^2$ is closed$\}$
      $\hat{z}^2 \leftarrow z^2$ closed endpoint as UL boundary of unexplored region
    end if
    if $\min_{i=1,2}\{|\hat{z}_i^2 - z_i^R|\} < \epsilon$ **then**
      $N \leftarrow N \cup point(\hat{z}^2) \cup point(z^R)$ add to frontier
    else
      $Q \leftarrow Q \cup B\,(\hat{z}^2, z^R)$ add to queue
    end if
  end if
end while
return (N, Q)

Algorithm 2 BasicInnerLoop

Input: nondominated point $z^* = \text{lexmin}\{(z_1, z_2) : z_2(x) \leq \mu\}$ such that z^* on the split, i.e. $z_2^* = \mu$
Input: z^L, z^R upper-left and lower-right boundaries of the rectangle to explore
Output: $(z^1, z^2, z1\_open, z2\_open, M)$ the UL and BR nondominated points endpoints defining the line segment of the nondominated frontier containing z^*; markers to tell whether each endpoint is open (TRUE) or closed (FALSE); and the nondominated points that dominate open endpoints in M

$(z^1, z^2, \vec{w}, w\_known) \leftarrow LineGen(z^*, z^L, z^R)$ generate line segment of frontier that contains z^* for integer vector x_I^* via Line Generation subroutine (line segment defined by two endpoints) and return slope of the line segment to be used for scalarization
if $|z_2^1 - z_2^L| < \epsilon$ and $|z_1^1 - z_1^L| > \epsilon$ **then**
  $z1\_open \leftarrow TRUE$ because z^L dominates z^1
else
  $z1\_open \leftarrow FALSE$ temporary status with best of known information
end if
if $|z_1^2 - z_1^R| < \epsilon$ and $|z_2^2 - z_2^R| > \epsilon$ **then**
  $z2\_open \leftarrow TRUE$
else
  $z2\_open \leftarrow FALSE$
end if
$M \leftarrow \emptyset$
if $\min_{i=1,2}\{|z_i^1 - z_i^2|\} < \epsilon$ **or** $\neg w\_known$ **then**
  return $(z^*, z^*, z1\_open, z2\_open, M)$ ND segment is just the isolated nondominated point z^*
end if
$y^* \leftarrow solution\,(\min\{\vec{w}^T z(x) : z_1(x) \leq z_1^2 - \epsilon(z2\_open),\ \ z_2(x) \leq z_2^1 - \epsilon(z1\_open),\ \ IP, ifs = z^*.solution\})$ (epsilon adjustments only for open endpoints)
while $\vec{w}^T z(y^*) < \vec{w}^T z^* - \epsilon$ (z^* is suboptimal to $z(y^*)$ w.r.t. $\vec{w}$) **do**
  if $\min_{i=1,2}\{|z_i^* - z_i(y^*)|\} < \epsilon$ and $max_{i=1,2}\{|z_i^* - z_i(y^*)|\} > \epsilon$ **then**
    ERROR message
  end if
  if $z_1(y^*) \leq z_1^* - \epsilon$ **then**
    $\hat{v}.solution \leftarrow \min\{z_2(x) : \vec{w}^T z(x) \leq \vec{w}^T z^*,\ \ x_I = y_I^*,\ \ LP,\ ifs = y_I^*\}$ explore the slice of y_I^*
    $\hat{v} \leftarrow z(\hat{v}.solution)$
    if $\hat{v} \in line(z^1, z^2)$ **then**
      $\hat{v}.solution \leftarrow \min\{z_1(x) : z_2(x) \leq \hat{v}_2 + \epsilon,\ \ x_I = y_I^*,\ \ LP,\ ifs = y_I^*\}$ correct $\hat{v}$
      $\hat{v} \leftarrow z(\hat{v}.solution)$
    end if
    if $\hat{v} \in line(z^1, z^2)$ **then**
      $z^1 \leftarrow \hat{v}$
      $z1\_open \leftarrow FALSE$
    else $\{\hat{v} \notin line(z^1, z^2)\}$
      $v^1 \leftarrow \hat{v}$ keep track of most recent $\hat{v}$ that dominates z^1
      $z^1 \leftarrow point(line(z^1, z^2), z_2 = \hat{v}_2)$
      $z1\_open \leftarrow TRUE$
    end if
  end if
  if $z_1(y^*) \geq z_1^* + \epsilon$ **then**
    $\hat{v}.solution \leftarrow \min\{z_1(x) : \vec{w}^T z(x) \leq \vec{w}^T z^*,\ \ x_I = y_I^*,\ \ LP,\ ifs = y_I^*\}$ explore the slice of y_I^*
    $\hat{v} \leftarrow z(\hat{v}.solution)$
    if $\hat{v} \in line(z^1, z^2)$ **then**
      $z^2 \leftarrow \hat{v}$
      $z2\_open \leftarrow FALSE$
    else $\{\hat{v} \notin line(z^1, z^2)\}$
      $v^2 \leftarrow \hat{v}$ keep track of most recent $\hat{v}$ that dominates z^2
      $z^2 \leftarrow point(line(z^1, z^2), z_1 = \hat{v}_1)$
      $z2\_open \leftarrow TRUE$
    end if
  end if
  $y^* \leftarrow solution\,(\min\{\vec{w}^T z(x) : z_1(x) \leq z_1^2 - \epsilon(z2\_open),\ \ z_2(x) \leq z_2^1 - \epsilon(z1\_open),\ \ IP,\ ifs = z^*\})$
end while
if $z1\_open$ and $|z_2^1 - z_2^L| > \epsilon$ **then**
  $\hat{z}^1 \leftarrow point(\min\{z_1(x) : z_2(x) \leq z_2^1,\ IP,\ ifs = v^1.solution\}, z_2^1)$ NDP that dominates z^1
  $M \leftarrow M \cup \hat{z}^1$
end if
if $z2\_open$ and $|z_1^2 - z_1^R| > \epsilon$ **then**
  $\hat{z}^2 \leftarrow point(z_1^2, \min\{z_2(x) : z_1(x) \leq z_1^2,\ IP,\ ifs = v^2.solution\})$ NDP that dominates z^2
  $M \leftarrow M \cup \hat{z}^2$
end if
return $(z^1, z^2, z1\_closed, z2\_closed, M)$

Algorithm 3 RecursiveOuterLoop

Input: Objective functions $z_1(x), z_2(x)$ and feasible set X are *fixed and global*
Output: N the entire nondominated frontier as a set of line segments, Q any unexplored regions of the frontier space (for partial run-time)

$z^L \leftarrow \text{lexmin}\{(z_1, z_2), IP\}$ is the UL nondominated point
$z^R \leftarrow \text{lexmin}\{(z_2, z_1), IP, ifs = z^L.solution\}$ is the BR nondominated point
$N \leftarrow point(z^L) \cup point(z^R)$
if $\min_{i=1,2}\{|z_i^L - z_i^R|\} < \epsilon$ **then**
  $Q \leftarrow \emptyset$ no region to explore
  return (N, Q)
else
  $Q \leftarrow B(z^L, z^R)$ the unexplored region defined by box with corner points z^L, z^R
end if
while $Q \neq \emptyset$ **do**
  $B(z^L, z^R) \leftarrow element(Q)$
  $Q \leftarrow setminus(Q, B(z^L, z^R))$
  $\mu \in (z_2^R, z_2^L)$ arbitrary horizontal dividing line between z^L, z^R
  $z^* \leftarrow \text{lexmin}\{(z_1, z_2) : z_2(x) \leq \mu, IP, ifs = z^R.solution\}$ find NDP in lower half
  if $\mu - z_2^* > \epsilon$ **then**
    if $\min_{i=1,2}\{|z_i^* - z_i^R|\} < \epsilon$ **then**
      $N \leftarrow N \cup point(z^*)$ if epsilon-close, add to ND frontier, do not add region to queue
    else
      $Q \leftarrow Q \cup B(z^*, z^R)$ otherwise, add a new unexplored region to the queue
    end if
    $\hat{z} \leftarrow \text{lexmin}\{(z_2, z_1) : z_1(x) \leq z_1^* - \epsilon, IP, ifs = z^L.solution\}$
    if $\min_{i=1,2}\{|\hat{z}_i - z_i^L|\} < \epsilon$ **then**
      $N \leftarrow N \cup point(\hat{z})$
    else
      $Q \leftarrow Q \cup B(z^L, \hat{z})$
    end if
  else $\{|z_2^* - \mu| < \epsilon\}$
    $L \leftarrow \emptyset$ no known line segments so far
    $M \leftarrow$ RecursiveInnerLoop(z^*, z^L, z^R, L) M includes all found ND points and line segments
    $M_< \leftarrow Orderbyz1(\{z^L, z^R\} \cup M)$ giving ordered set $(z^L, m^1, ..., m^k, z^R)$
    for $\forall$ consecutive pairs $(a, b) \subset M_<$ **do**
      if $line(a, b) \in M$ **then**
        $N \leftarrow N \cup line(a, b)$ add line segment to frontier
      else
        $N \leftarrow N \cup point(a) \cup point(b)$ add individual points to frontier
        if $\min_{i=1,2}\{|a_i - b_i|\} > \epsilon$ **then**
          $Q \leftarrow Q \cup B(a, b)$ add unexplored region to queue
        end if
      end if
    end for
  end if
end while
return (N, Q)

Algorithm 4 RecursiveInnerLoop

Input: nondominated point z^*
Input: z^L, z^R known NDPs providing the upper-left and lower-right corner points of the box
Input: L the set of all inherited line segments so far (for line trimming); note that $L = \emptyset$ on first call
Output: M all found nondominated points/line segments

$(z^1, z^2, \vec{w}, w\_known) \leftarrow LineGen(z^*, z^L, z^R)$
if w_known **then**
  $(z^1, z^2) \leftarrow LineTrim(z^*, z^1, z^2, \vec{w}, L)$
end if
if $|z_2^1 - z_2^L| < \epsilon$ and $|z_1^1 - z_1^L| > \epsilon$ **then**
  $z1\_open \leftarrow TRUE$ because z^L dominates z^1
else
  $z1\_open \leftarrow FALSE$ temporary status with best of known information
end if
if $|z_1^2 - z_1^R| < \epsilon$ and $|z_2^2 - z_2^R| > \epsilon$ **then**
  $z2\_open \leftarrow TRUE$
else
  $z2\_open \leftarrow FALSE$
end if
$M \leftarrow \emptyset$
if $\min_{i=1,2}\{|z_i^1 - z_i^2|\} < \epsilon$ or $\neg w\_known$ **then**
  $M \leftarrow M \cup point(z^*)$ frontier is just the isolated nondominated point z^*
  return M
else
  $L \leftarrow L \cup line(z^1, z^2)$
end if
$y^* \leftarrow solution\,(\min\{\vec{w}^T z(x) : z_1(x) \le z_1^2 - \epsilon(z2\_open),\ \ z_2(x) \le z_2^1 - \epsilon(z1\_open),\ \ IP,\ ifs = z^*.solution\})$ (epsilon adjustment only for open endpoints)
while $\vec{w}^T z(y^*) < \vec{w}^T z^* - \epsilon$ (z^* is suboptimal to $z(y^*)$ w.r.t. $\vec{w}$) **do**
  if $\min_{i=1,2}\{|z_i^* - z_i(y^*)|\} < \epsilon$ and $max_{i=1,2}\{|z_i^* - z_i(y^*)|\} > \epsilon$ **then**
    ERROR message
  end if
  $M \leftarrow M \cup z(y^*)$
  if $z_1(y^*) \le z_1^* - \epsilon$ **then**
    $\alpha \leftarrow \text{argmin}\{p_2 : p \in z^L \cup M, p_1 < z_1^*\}$ bound by the nearest (on left) found NDP to z^*
    $M' \leftarrow InnerLoop(z(y^*), \alpha, z^*, L)$ recurse within $B(\alpha, z^*)$ with inherited line segments in L
    $M \leftarrow M \cup M'$
    $L \leftarrow L \setminus line(z^1, z^2)$ remove "old" line segment
    $\hat{p} \leftarrow \text{argmin}\{p_2 : p \in M, p_1 < z_1^*\}$ choose the nearest (on left) found NDP to z^*
    if $\hat{p} \in line(z^1, z^2)$ **then**
      $z^1 \leftarrow \hat{p}$
      $z1\_open \leftarrow FALSE$
    else $\{\hat{p} \notin line(z^1, z^2)\}$
      $z^1 \leftarrow point(line(z^1, z^2), z_2 = \hat{p}_2)$
      $z1\_open \leftarrow TRUE$
    end if
  end if
  if $z_1(y^*) \ge z_1^* + \epsilon$ **then**
    $\beta \leftarrow \text{argmin}\{p_1 : p \in z^R \cup M, p_2 < z_2^*\}$
    $M' \leftarrow InnerLoop(z(y^*), z^*, \beta, L)$ recurse within $B(z^*, \beta)$
    $M \leftarrow M \cup M'$
    $\hat{p} \leftarrow \text{argmin}\{p_1 : p \in M, p_2 < z_2^*\}$ choose the nearest (on right) found NDP to z^*
    if $\hat{p} \in line(z^1, z^2)$ **then**
      $z^2 \leftarrow \hat{p}$
      $z2\_open \leftarrow FALSE$
    else $\{\hat{p} \notin line(z^1, z^2)\}$
      $z^2 \leftarrow point(line(z^1, z^2), z_1 = \hat{p}_1)$
      $z2\_open \leftarrow TRUE$
    end if
  end if
  $L \leftarrow L \cup line(z^1, z^2)$ add updated line segment
  $y^* \leftarrow solution\,(\min\{\vec{w}^T z : z_1(x) \le z_1^2 - \epsilon(z2\_open),\ \ z_2(x) \le z_2^1 - \epsilon(z1\_open),\ \ IP,\ ifs = z^*.solution\})$ (epsilon adjustment only for open endpoints)
end while
$M \leftarrow M \cup line(z^1, z^2)$
return M

2. Line Segment Generation Subroutine

The purpose of the line segment generation subroutine is to provide the inner loop with a single line segment that can be reduced to an NLS by simply updating the endpoints. Finding the maximal line segment, $L(z^1, z^2)$, of the slice problem for given integer solution x_I^* with $z^* = z(x^*) \in L(z^1, z^2)$ can be done, in theory, by accessing the values of the dual variables at the optimal branch-and-bound node of the IP that found z^* and applying LP duality theory. However, we are not confident that, in practice, the IP solver will provide the LP dual variables needed, since it may have found z^* with a primal heuristic, eliminated variables and/or constraints or modified coefficients with preprocessing, or added constraints. Therefore, we simply use a variant of dichotomic search (Cohon 1978, Aneja and Nair 1979), restricted to a box around z^* of size sufficiently large to ensure that the gradient of any line segment that contains z^* and lies within the box can be calculated accurately.

The Boxed Line Method requires the line generation subroutine to produce three interrelated forms of output. Primarily, it seeks the two endpoints of the line segment, z^1 and z^2. In addition, as long as $z^1 \neq z^2$, it seeks the gradient vector, $\vec{w}$, of the line segment $L(z^1, z^2)$. When $z^1 = z^2$, we say that $\vec{w}$ does not exist; we note that when the gradient vector $\vec{w}$ exists, it is normalized before it is returned. Obviously, if we have z^1 and $z^2 \neq z^1$, then it is trivial to compute $\vec{w}$. However, if $\vec{w}$ is discovered first, it is also possible to find each endpoint by solving the following LPs:

$$x^1 = \arg\min\{z_1(x) : \vec{w}^T z(x) \leq \vec{w}^T z^*, z_2(x) \leq z_2^L, x_I = x_I^*, x \in \mathcal{X}\}, \text{ and} \tag{1}$$

$$x^2 = \arg\min\{z_2(x) : \vec{w}^T z(x) \leq \vec{w}^T z^*, z_1(x) \leq z_1^R, x_I = x_I^*, x \in \mathcal{X}\}. \tag{2}$$

Then $z^1 = z(x^1)$ and $z^2 = z(x^2)$. (By construction, z^1 or z^2 should not be outside the box $B(z^L, z^R)$, since the box's corner points are NDPs.) Therefore, the subroutine either finds *both* endpoints *or* the gradient of the line segment, and then the remaining output can be computed accordingly. Because of this, Algorithm 5, giving details of the subroutine, is flexible in the order in which these outputs are discovered. The algorithm generally searches for z^2 first, and in the case that neither z^2 nor $\vec{w}$ are found, then it assumes $z^2 = z^*$ and searches for z^1.

Algorithm 5 Line Segment Generation

Input: z^* nondominated point whose integer vector is fixed

Input: z^L, z^R upper-left and lower-right boundaries of the rectangle to explore

Output: $(z^1, z^2, \vec{w}, w\_known)$ the UL and BR endpoints for the line segment of frontier including z^* and the slope of the line segment (if found)

$z1\_known, z2\_known \leftarrow FALSE$

$w\_known \leftarrow FALSE$ weight vector for which z^* is optimal (if found, can be used to directly compute z^1, z^2)

$\delta_1, \delta_2 \leftarrow 100\epsilon$ starting horizontal/vertical distance

(Explore for z^2 in lower-right:)

$i \leftarrow 0$

$t^i \leftarrow \text{lexmin}\{(z_2, z_1) : z_1(x) \leq z_1^* + \delta_1,\ LP,\ ifs = z^*.solution\}$ is the BR nondominated point

while $w\_known = FALSE$ **and** $z2\_known = FALSE$ **do**

  if $\min(|z_1^* - t_1^i|, |z_2^* - t_2^i|) < \epsilon$ **then**

    $z^2 \leftarrow z^*$

    $z2\_known \leftarrow TRUE$

    BREAK

  end if

  $\vec{w}^i \leftarrow slope(z^*, t^i)$

  $t^{i+1} \leftarrow \min\{(\vec{w}^i)^T z(x),\ LP,\ ifs = z^*.solution\}$ (scalarized by $\vec{w}^i$)

  if $(\vec{w}^i)^T t^{i+1} = (\vec{w}^i)^T z^*$ and $|z_1^* - t_1^i| > \epsilon$ and $|z_2^* - t_2^i| > \epsilon$ **then**

    $\vec{w} \leftarrow \vec{w}^i$

    $w\_known \leftarrow TRUE$

    if $i \geq 1$ and $t^i \in B(z^L, z^R)$ **then**

      $z^2 \leftarrow t^i$

      $z2\_known \leftarrow TRUE$

    end if

    BREAK

  end if

  $i \leftarrow i + 1$

end while

(continued on next page)

(Line Generation)
if $w\_known = FALSE$ **then**
  (Explore for z^1 in upper-left:)
  $i \leftarrow 0$
  $t^i \leftarrow \text{lexmin}\{(z_1, z_2) : z_2(x) \leq z_2^* + \delta_2,\ LP,\ ifs = z^*.solution\}$ is the UL nondominated point
end if
while $w\_known = FALSE$ **and** $z1\_known = FALSE$ **do**
  if $\min(|z_1^* - t_1^i|, |z_2^* - t_2^i|) < \epsilon$ **then**
    $z^1 \leftarrow z^*$
    $z1\_known \leftarrow TRUE$
    BREAK
  end if
  $\vec{w}^i \leftarrow slope(z^*, t^i)$
  $t^{i+1} \leftarrow \min\{(\vec{w}^i)^T z(x),\ LP,\ ifs = z^*.solution\}$
  if $(\vec{w}^i)^T t^{i+1} = (\vec{w}^i)^T z^*$ **then**
    $\vec{w} \leftarrow \vec{w}^i$
    $w\_known \leftarrow TRUE$
    if $i \geq 1$ **then**
      $z^1 \leftarrow t^i$
      $z1\_known \leftarrow TRUE$
    end if
    BREAK
  end if
  $i \leftarrow i + 1$
end while
(By the time the code has reached this point, either $\vec{w}$ is known OR *both* z^1 *and* z^2 are known)
if $z1\_known = FALSE$ **then**
  $z^1 \leftarrow \min\{z_1(x) : \vec{w}^T z(x) \leq \vec{w} z^* + \epsilon, z_2(x) \leq z_2^L,\ LP,\ ifs = z^*.solution\}$
end if
if $z2\_known = FALSE$ **then**
  $z^2 \leftarrow \min\{z_2(x) : \vec{w}^T z(x) \leq \vec{w} z^* + \epsilon, z_1(x) \leq z_1^R,\ LP,\ ifs = z^*.solution\}$
end if
if $w\_known = FALSE$ **then**
  $\vec{w} \leftarrow [-1, -1]$ clearly impossible gradient vector
end if
$\vec{w} \leftarrow \vec{w} / ||\vec{w}||_1$
return $(z^1, z^2, \vec{w}, w\_known)$

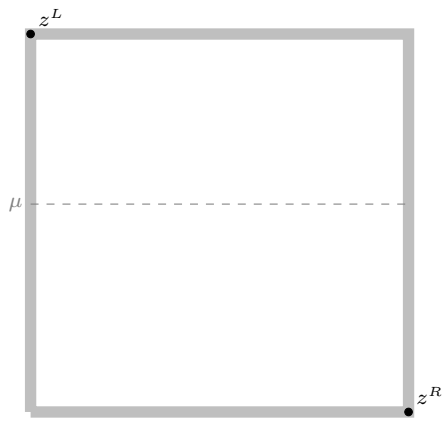

Figure 1 **A region Y with $n=0$ nondominated line segments in its interior and $g=1$ vertical gaps. An arbitrary horizontal split line $z_2=\mu$ is shown as a dashed line.**

3. Complexity of the Basic Method

EXAMPLE 1. For $n=0$ and $g=1$ (see Figure 1), $\ell(0,1)=2$ and $s(0,1)=0$.

Proof of Example 1 For any split line $z_2=\mu$ where $z_2^R<\mu<z_2^L$, we have that the first lexicographic optimization IP (5) returns z^R. Since we have $z_2^R<\mu$, the algorithm continues by solving a second lexicographic optimization IP (6), which yields z^L. Note that no new regions are added to the queue (because $B(z^L,z^L)$ and $B(z^R,z^R)$ are trivially small), so we have that the iteration of the outer loop terminates having solved $\ell(0,1)=2$ lexicographic IPs and $s(0,1)=0$ scalarized IPs. □

EXAMPLE 2. For $n=1$ and $g=0$ (see Figure 2), $\ell(1,0)=1$ and $s(1,0)=1$.

Proof of Example 2 Since a single NLS traverses Y, solving the first lexicographic minimization IP (5) with any split line $z_2=\mu$ where $z_2^R<\mu<z_2^L$ will result in z^* on the split line, i.e., $z_2^*=\mu$. The line generation subroutine will generate the entire line segment within Y, and because there are no vertical gaps, we must have that $z_2^1=z_2^L$ and $z_2^2=z_2^R$. Since z^R is nondominated, the latter implies $z^2=z^R$, and so z^2 is *closed*. However, z^1 may be open *or* closed; namely, if $z_1^L<z_1^1$, z^1 will be flagged as open. Regardless of whether it is open or closed, the inner loop will solve the single-objective IP (8) once to confirm that the line segment is indeed nondominated (recall that the IP formulation includes a conditionally strict inequality for the open endpoint which makes z^L infeasible). By our assumption that $n=1$, an optimal solution must exist on the line segment. At the end of the inner loop, since $z_2^1=z_2^L$, no other single-objective IPs will be solved by the inner loop since it is known that z^L is the NDP that dominates z^1. After the inner loop concludes,

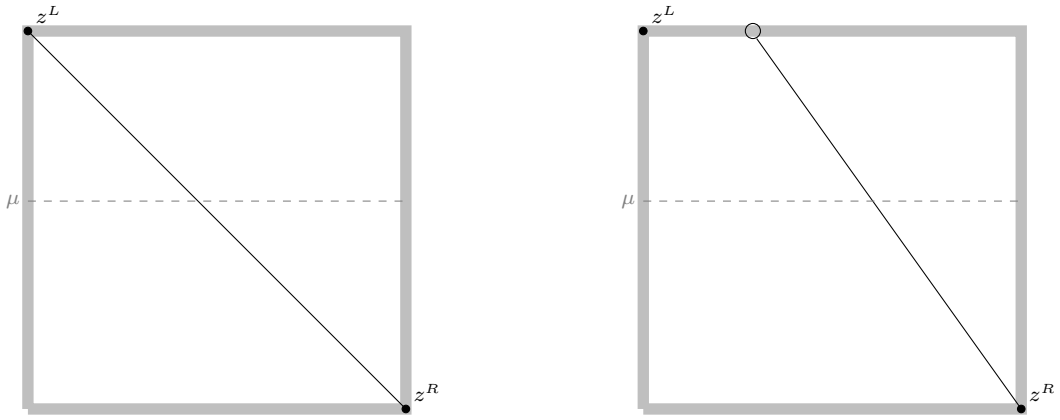

Figure 2 **The only two cases for $n=1$, $g=0$: The nondominated line segment must be incident to z^R to ensure no vertical gap, but the endpoint z^1 may be either closed (left) or open (right).**

the outer loop would not add any box to the queue, since they would both be trivially small. Thus, $\ell(1,0)=1$ and $s(1,0)=1$. $\square$

LEMMA 1. *For all $g \in \{0,1,2\}$, $s(1,g)=1$.*

Proof of Lemma 1 Note that we have already shown $s(1,0)=1$ by Example 2. Therefore, suppose $g=1$ or $g=2$.

First, consider when the split line $z_2=\mu$ intersects the NLS. Then solving one lexicographic IP (5) finds z^* such that $z_2^*=\mu$ (we do not count lexicographic IPs for s). Once the line segment containing z^* is generated, the single-objective IP (8) is solved to determine if the line segment is nondominated. As mentioned previously, the algorithm checks if $z_2^1=z_2^L$ and $z_1^L<z_1^1$, in which case endpoint z^1 is flagged as open, and it is known that z^L dominates z^1 (and similarly for z^2 and z^R). Note that by assumption, $n=1$ implies there are no other NDPs in the interior of $B(z^1,z^2)$ that dominate the line segment, and by construction we have the conditionally strict inequality that prevents finding z^L (or z^R) in the case that z^L dominates z^1 (or z^R dominates z^2). Hence, an optimal solution to the single-objective IP (8) will certainly map to a point on the line segment, which will terminate the while loop within the inner loop. Since the NDPs that dominate the open endpoints are known, no additional single-objective IPs will be solved to discover what points dominate z^1 or z^2, and so the inner loop terminates.

Once the inner loop has generated the NLS within the given region, the outer loop will add at most two nontrivial boxes, whose interiors contain $n'=n''=0$ NLSs, to the queue. So we have for some nonnegative integers $(g',g'') \in \mathbb{Z}_+^2$ where $g'+g''=g$ but $g',g'' \leq 1$

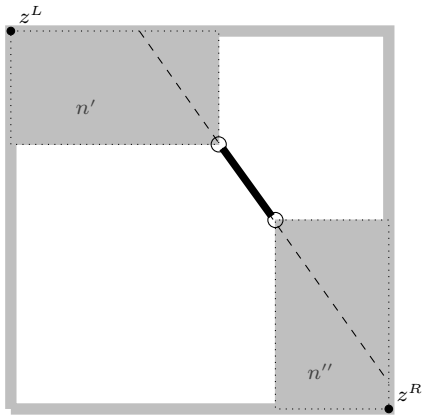

Figure 3 **General NDF where there are $n \geq 1$ nondominated line segments. In the worst case for counting single-objective IP solves, both endpoints are open.**

(this follows from $g' \leq n' + 1$ and $g'' \leq n'' + 1$ where $n' = n'' = 0$). Therefore, $s(1, g) = 1 + s(0, g') + s(0, g'') = 1$ because $s(0, 0) = 0$ by definition, and $s(0, 1) = 0$ by Example 1. Thus, $s(1, g) = 1$ for $g \in \{0, 1, 2\}$.

Second, consider when the split line crosses at a vertical gap in the NDF. Note that a single-objective IP is not solved until the inner loop is initiated, which respectively does not occur until a NDP is found on the split line. Therefore there can be at most g iterations of the outer loop in which $z_2^* < \mu$, wherein each iteration is only performing lexicographic IP solves as the split line identifies a vertical gap in the NDF which is then removed by initiating new boxes to the queue (note we do not count any of these lexicographic IP solves). Therefore, in at most g iterations the lexicographic optimization IP will return z^* such that $z_2^* = \mu$. At such point, the process would follow the process argued above, so again we have $s(1, g) = 1$ for $g \in \{0, 1, 2\}$. □

THEOREM 1. *For all $n \geq 1$, $\hat{s}(n) = \frac{n(n+1)}{2} + 2(n-1)$.*

Proof of Theorem 1 We use induction. The case $n = 1$ follows since $\hat{s}(1) = 1 = \frac{1(1+1)}{2} + 2(1-1)$. Now assume that, for some $n \geq 1$, $\hat{s}(n) = \frac{n(n+1)}{2} + 2(n-1)$, and consider $\hat{s}(n+1)$. By Lemma 2 and the inductive assumption,

$$\begin{aligned}\hat{s}(n+1) &= n + 3 + \hat{s}(n) = n + 3 + \frac{n(n+1)}{2} + 2(n-1) = \frac{1}{2}n^2 + \frac{7}{2}n + 1 \\ &= \frac{1}{2}n^2 + \frac{3}{2}n + 1 + 2n = \frac{(n+1)(n+2)}{2} + 2n,\end{aligned}$$

as required. □

4. Line Segment Trimming Subroutine

Let $L_\alpha := L(\alpha^1, \alpha^2)$, where $\alpha_1^1 < \alpha_1^2$ and $\alpha_2^1 > \alpha_2^2$, denote the current (or "child") line segment, which contains NDP z^α, and has gradient vector $\vec{w}^\alpha$. Suppose that a line segment from a parent call, $L_\beta := L(\beta^1, \beta^2)$, where $\beta_1^1 < \beta_1^2$ and $\beta_2^1 > \beta_2^2$, contains the NDP z^β and has gradient $\vec{w}^\beta$. The two line segments L_α and L_β can be extended to lines that intersect provided $\vec{w}^\alpha \neq \vec{w}^\beta$ (recall the gradient vectors are normalized). Their point of intersection, $\gamma \in \mathbb{R}^2$, is easily computed: it is the solution of the system of two linear equations given by $\vec{w}^\alpha \gamma = \vec{w}^\alpha z^\alpha$ and $\vec{w}^\beta \gamma = \vec{w}^\beta z^\beta$.

The resulting intersection point is contained in both line segments if and only if

$$\gamma_1 \in [\alpha_1^1, \alpha_1^2] \cap [\beta_1^1, \beta_1^2] \quad \text{and} \quad \gamma_2 \in [\alpha_2^2, \alpha_2^1] \cap [\beta_2^2, \beta_2^1]. \tag{3}$$

If γ satisfies (3), then the current line segment, L_α, is trimmed. Exactly one of the two subsets of L_α, $L(\alpha^1, \gamma)$ or $L(\gamma, \alpha^2)$, must be dominated by points from L_β. Since $z^\alpha \in L_\alpha$ is an NDP, L_α is trimmed so as to retain the portion of it that contains z^α: if $\gamma_1 < z_1^\alpha$, then update $\alpha^1 = \gamma$ and flag α^1 to be closed; otherwise, if $\gamma_1 > z_1^\alpha$, then update $\alpha^2 = \gamma$ and flag α^2 to be closed. In the unlikely case that $\gamma_1 = z_1^\alpha$, L_α is trimmed as follows: if $\gamma_1 < z_1^\beta$, then update $\alpha^2 = \gamma$ and flag α^2 to be closed; otherwise, if $\gamma_1 > z_1^\beta$, then update $\alpha^1 = \gamma$ and flag α^1 to be closed. Note that updating the appropriate endpoint of L_α to be the intersection point γ and flagging it as closed prevents rediscovery of any point in L_β that dominates any part of L_α. See Algorithm 6 for details.

Algorithm 6 LineTrim: Segment Trimming

Input: z^* nondominated point

Input: $z^1, z^2, \vec{w}$ upper-left and lower-right points of the line segment and the gradient vector

Input: $L = \{L_1, L_2, ..., L_n\}$ finite (possibly empty) set of line segments to test for intersection, where $L_i = L(a^i, b^i)$

Output: $(z^1, z^2, \vec{w}, w\_known)$ the endpoints for the line segment of frontier including z^* and the gradient vector (if found)

if $L = \emptyset$ **then**
  return (z^1, z^2)
end if
for i=1,2,...,n **do**
  $\vec{w}^i \leftarrow (b^i_1 - a^i_1, a^i_2 - b^i_2)$
  $\vec{w}^i \leftarrow \vec{w}/||\vec{w}||_1$
  if $\vec{w} \neq \vec{w}^i$ **then**
    solve SLE for γ:

$$\begin{bmatrix} \vec{w}_1 & \vec{w}_2 \\ \vec{w}^i_1 & \vec{w}^i_2 \end{bmatrix} \begin{bmatrix} \gamma_1 \\ \gamma_2 \end{bmatrix} = \begin{bmatrix} \vec{w}_1 z^*_1 + \vec{w}_2 z^*_2 \\ \vec{w}^i_1 a^i_1 + \vec{w}^i_2 a^i_2 \end{bmatrix}$$

    if $\gamma_1 \in [z^1_1, z^2_1] \cap [a^i_1, b^i_1]$ **and** $\gamma_2 \in [z^2_2, z^1_2] \cap [b^i_2, a^i_2]$ **then**
      if $\gamma_1 < z^*_1$ **then**
        $z^1 \leftarrow \gamma$
      else $\{\gamma_1 > z^*_1\}$
        $z^2 \leftarrow \gamma$
      end if
    end if
  end if
end for
return (z^1, z^2)

5. Instance Generation

The generated instances have the objective functions $z_1(x) := x_1$ and $z_2(x) := x_2$, where $x \in \mathbb{R}^2$ is a vector of two continuous decision variables. Each instance's NDF includes some sections of the line segment $L_k = \{(x_1, x_2) \in \mathbb{R}^2 : x_1 + x_2 = 0, -k \leq x_i \leq k \text{ for } i = 1, 2\}$ where $k \in (0, \infty)$ is a parameter. This line segment will be one slice in the instance. The instance has π other slices, where π is a parameter, all of which have their image in criterion space given by a pointed cone. The vertices of each cone lie on a line segment parallel to L_k but shifted vertically down by $d \in (0, \frac{2k}{\pi})$ units: they lie on

$$L_d = \{(x_1, x_2) : x_1 + x_2 = -d, -k \leq x_i \leq k - d \text{ for } i = 1, 2\},$$

where d is a parameter. The slice with vertex $(a, b) \in L_d$ is generated by the slice problem with feasible set

$$P_{(a,b)}(\theta_1, \theta_2) = \{x \in \mathbb{R}^2 \ : \ \theta_1 x_1 + (1 - \theta_1) x_2 \geq \theta_1 a + (1 - \theta_1) b \tag{4}$$

$$\theta_2 x_1 + (1 - \theta_2) x_2 \geq \theta_2 a + (1 - \theta_2) b\}, \tag{5}$$

where $\theta_1 \in [\frac{1}{2}, 1]$ and $\theta_2 \in [0, \frac{1}{2}]$ are parameters that control the width of the cone. These ranges for θ_1 and θ_2 ensure that the resulting cone contains $(a, b) + \mathbb{R}^2_+$ and is contained in the half-space that contains $L_d + \mathbb{R}^2_+$. Figure 4 illustrates this structure, showing the image of the feasible set in criterion space for 4 slices: L_k and three pointed cones with vertices lying on L_d. Note that $\theta_1 = 1$ and $\theta_2 = 0$ imply that the cone is precisely $(a, b) + \mathbb{R}^2_+$, and hence that (a, b) is an isolated NDP. Otherwise, if $\theta_1 < 1$, the left-hand boundary of the cone, from (a, b) to its intersection with L, is an NLS, while if $\theta_2 > 0$, the right-hand boundary of the cone, from (a, b) to its intersection with L_k, is an NLS. All instances are constructed to have the property that no two cones overlap within the band between L_k and L_d. This means that L_k alternates between a section that is part of the NDF and a section dominated by one cone.

Given any set of π polyhedra, say $P^i = \{x \in \mathbb{R}^m \ : \ A^i x \geq c^i\}$, for $i = 1, \ldots, \pi$, the MIP feasible set $\{(x, y) \in \mathbb{R}^m \times \{0, 1\}^\pi \ : \ A^i x \geq c^i - M_i(1 - y_i), \ \forall i = 1, \ldots, \pi, \ \sum_{i=1}^{\pi} y_i = 1\}$ has (for appropriately chosen big-M values, $(M_i)_{i=1}^{\pi}$) slice problems with feasible sets $\{P^i\}_{i=1}^{\pi}$. We take the objective function vector to be $z(x) = (x_1, ...x_m)$ so that any polyehdron in criterion space is easily reverse-engineered to give a polyhedron in decision space (they have precisely the same description).

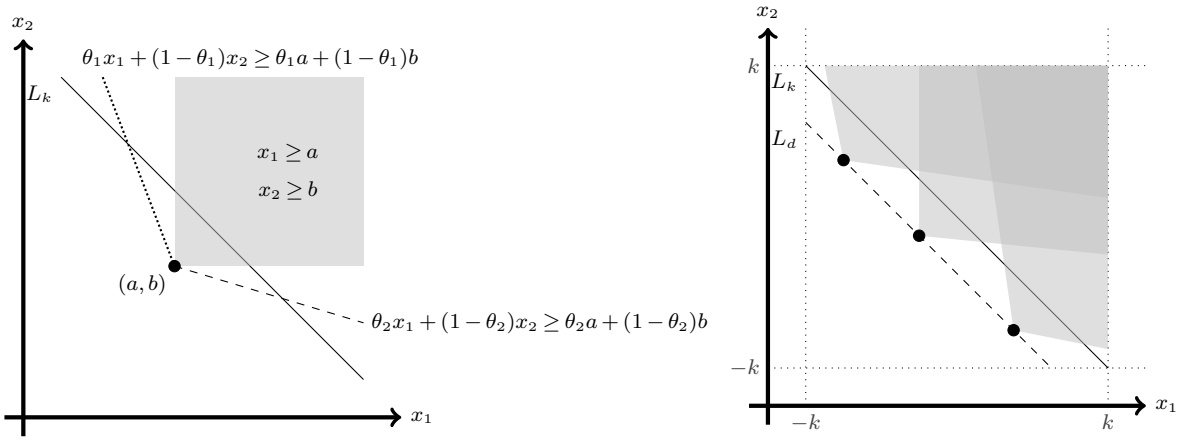

Figure 4 **(Left) The orthogonal pointed cone with vertex (a, b) is shaded. The generalized pointed cone is defined between the dotted and dashed lines. Different slices in an instance may have different values of θ_1 and θ_2. (Right) An example with 4 slices, including L_k and the boundary of three cones.**

For the $\pi + 1$ polyhedra consisting of L_k and π pointed cones $P_{(a_i,b_i)}(\theta_1^i, \theta_2^i)$, for $i = 1, \ldots, \pi$, where (a_i, b_i) denotes the vertex of the ith cone, this gives the following BOMIP:

$$\text{minimize} \qquad (x_1, x_2) \tag{6}$$

$$\text{s.t.} \qquad x_1 + x_2 \geq -2k(1 - y_0) \tag{7}$$

$$\theta_1^i x_1 + (1-\theta_1^i)x_2 \geq \theta_1^i a_i + (1-\theta_1^i)b_i - 2k(1-y_i) \quad \forall i = 1, 2, ..., \pi \tag{8}$$

$$\theta_2^i x_1 + (1-\theta_2^i)x_2 \geq \theta_2^i a_i + (1-\theta_2^i)b_i - 2k(1-y_i) \quad \forall i = 1, 2, ..., \pi \tag{9}$$

$$\sum_{i=0}^{\pi} y_i = 1 \tag{10}$$

$$-k \leq x_i \leq k \qquad \forall i = 1, 2 \tag{11}$$

$$y \in \{0, 1\}^{\pi+1}. \tag{12}$$

This BOMIP has a number of variables and a number of constraints that is linear in π: it has $\pi + 3$ variables and $4 + 2\pi$ constraints.

We generate two different classes of instances having the structure described above. One class has the π cone points distributed randomly along L_d, but has $\theta_1 = 1$ and $\theta_2 = 0$ for all cones. Since this means that all cones are orthogonal pointed cones, we refer to these as *fixed cone-width instances*. An illustration of these instances is given in Figure 5. The other class has the cone points equally spaced along L_d, but has randomized values of θ_1 and θ_2; we call these *randomized cone-width instances*. These are illustrated by the right-hand side of Figure 4.

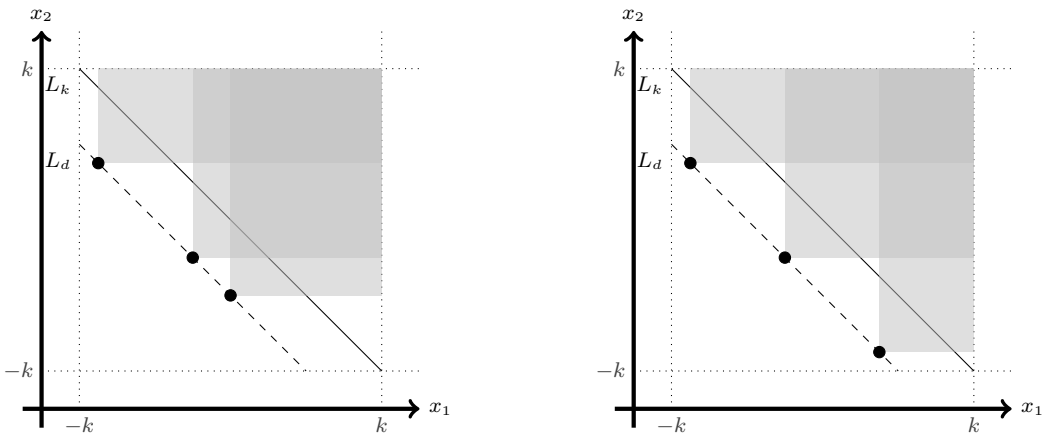

Figure 5 **NDPs and associated cones generated by the fixed cone-width class of instances. (Left) If the pointed cones from NDPs overlap on L_k, then there are fewer than $\pi+1$ nondominated segments of L_k. (Right) Choosing the NDPs such that $-b_i < a_{i+1}$ prevents such overlapping, thus guaranteeing exactly $\pi+1$ line segments of L_k are nondominated. All instances we generate are of the latter form.**

All instances are carefully designed to ensure that each NLS, which is not a point, has length at least ϵ. In the case of the fixed cone-width instances, the only such NLSs are sections of L_k, and these can be guaranteed to have length at least ϵ by a careful choice of the randomized spacing between cone points, as a function of d. For fixed cone-width instances we take $d = k/4$, and we expect the choice of k to ensure that $d \geq \epsilon$. The latter condition also ensures that the horizontal and vertical gaps in the NDF are of length at least ϵ. In the case of random cone-width instances, an NLS may be either a section of L_k or a section of a cone boundary. The length of an NLS from L_k is ensured to be at least ϵ by restricting the values of θ_1 and θ_2 to lie within $[\frac{3}{4}, 1]$ and $[0, \frac{1}{2}]$, respectively, and choosing $d = k/(\pi+1) - \frac{1}{2}$. The length of an NLS generated by a cone boundary is ensured to be at least ϵ by requiring $d \geq \epsilon$. Again, we expect the choice of k to guarantee that $d \geq \epsilon$, ensuring that the horizontal and vertical gaps in the NDF are also of length at least ϵ. We force a proportion of the cones in a randomized cone-width instance to be orthogonal[1].

Specifics of our procedure for generating fixed cone-width instances are given in Algorithm 7. By construction, the NDF associated with such an instance has exactly π isolated NDPs and $\pi+1$ line segments (2 half-open at the ends of L_k and $\pi-1$ open segments in between isolated NDPs). Even large instances of this class of BOMIPs are solved quite

[1] A cone is chosen to be orthogonal with probability ϕ, a parameter. In all our instances, we used $\phi = 0.05$.

quickly by most methods, so they are not ideal for testing computational efficiency. However, since their NDFs are known exactly, these instances are useful for validating the correctness of an algorithm.

Specifics of our procedure for generating random cone-width instances are given in Algorithm 8 and Algorithm 9, which generate the cone width parameters $\{(\theta_1^i, \theta_2^i)\}_{i=1}^{\pi}$ and cone vertices, $\{(a^i, b^i)\}_{i=1}^{\pi}$, respectively. The NDFs associated with the resulting BOMIP will have no more than $3\pi + 1$ line segments ($\pi + 1$ from L_k and at most 2 per cone), including open endpoints induced by any orthogonal cones and closed endpoints induced by intersecting slices. The random cone-width class of instances is more difficult to solve, in practice, than the fixed cone-width, because of the high frequency of intersecting slices in the NDF.

Together, these two structured sets of instances provide a useful way to study the accuracy and robustness of a BOMIP algorithm.

Algorithm 7 Fixed Cone-Width NDP Generation

$d = k/4$ (distance by which L_d is shifted down)
$w = (2k - d)/n$ (the width of subintervals)
$a_1 = U(-k, -k + w)$
$b_1 = -a_1 - d$
for $i = 2, 3, ..., n$ **do**
  $a_i = U(\max\{-k + (i-1)w, -b_{i-1} + \epsilon\}, -k + iw)$
  $b_i = -a_i - d$
end for

Algorithm 8 Randomized Theta Generation

thetalist= $\emptyset$
for $i = 1, 2, ..., n$ **do**
  if $U(0,1) \leq \pi$ **then**
    $\theta_1 = 1$
    $\theta_2 = 0$
  else
    $\theta_1 = U(\frac{3}{4}, 1)$
    $\theta_2 = U(0, \frac{1}{4})$
  end if
  thetalist.append((θ_1, θ_2))
end for

Algorithm 9 Randomized Cone-Width NDP Generation

$d = k/(n+1) - 0.5$
$a_1 = -k + 0.5d$
$b_1 = -a_1 - d$
for $i = 2, ..., n$ **do**
  $a_i = a_{i-1} + 2d + 1$
  $b_i = -a_i - d$
end for

6. Comparison between BLM variants

Algorithm	Ins	nNDP	nIPF	TT	IPT	LPT	nIP	nLex	nMin	nScal	nGood	nLP	nBox	nSIS	nZL
Basic	16	2810	115	680.0	522.1	157.0	8591	2848	14	2881	-	22687	2839	2302	2830
	17	2986	110	865.4	668.1	196.4	9058	3003	7	3045	-	24920	2995	2442	2987
	18	2775	101	739.1	571.8	166.7	8399	2787	12	2813	-	22478	2778	2253	2769
	19	6204	181	2060.2	1650.6	407.2	18721	6217	18	6269	-	53756	6205	5145	6193
	20	3145	100	849.7	646.1	202.6	9596	3182	34	3198	-	26839	3168	2704	3154
Avg.		3584	121.4	1038.9	811.7	226.0	10873	3607.4	17	3641.2	-	30136	3597	2969.2	3586.6
SIS	16	2764	115	206.0	148.6	57.1	1934	517	14	551	335	8287	843	335	500
	17	2952	110	257.7	185.6	71.7	1960	521	7	562	349	8915	862	349	507
	18	2738	101	232.8	170.5	61.9	1909	506	12	535	350	8172	847	350	492
	19	6166	181	577.1	433.1	143.1	3805	1018	16	1080	673	18212	1681	673	1000
	20	3122	100	216.1	148.8	66.9	1748	454	33	471	336	8698	777	336	427
Avg.		3548.4	121.4	297.9	217.3	80.1	2271.2	603.2	16.4	639.8	408.6	10456.8	1002	408.6	585.2
Recursive	16	2810	115	639.7	485.7	153.2	8212	2586	-	3040	-	22340	2577	2220	2568
	17	2987	110	835.3	639.2	195.3	8757	2799	-	3159	-	24674	2791	2416	2783
	18	2775	101	691.6	527.2	163.6	8015	2512	-	2991	-	22161	2503	2138	2494
	19	6198	181	2006.9	1594.1	410.4	18023	5718	-	6587	-	53148	5706	4964	5694
	20	3144	100	824.9	618.7	205.2	9113	2828	-	3457	-	26537	2814	2458	2800
Avg.		3582.8	121.4	999.7	773	225.5	10424	3288.6	-	3846.8	-	29772	3278.2	2839.2	3267.8

Table 1 Comparison between the different algorithms for historical instances, class C160. Times are reported in seconds.

Algorithm	Ins	nNDP	nIPF	TT	IPT	LPT	nIP	nLex	nMin	nScal	nGood	nLP	nBox	nSIS	nZL
Basic	21	16850	294	37665.5	33307.6	4339.7	53514	17790	40	17894	-	171370	17803	15926	17766
	22	19778	410	42045.3	36927.4	5095.7	61766	20540	23	20663	-	191611	20510	18074	20476
	23	17319	343	38995.1	34570.3	4409.4	53859	17895	26	18043	-	171514	17884	15776	17871
	24	19898	460	46967.1	41632.8	5312.7	62338	20706	34	20892	-	200619	20696	17867	20682
	25	13682	337	24450.1	21268.9	3171.0	42196	14024	27	14121	-	130934	13994	12130	13964
Avg.		17505.4	368.8	38024.6	33541.4	4465.7	54734.6	18191	30	18322.6	-	173209.6	18177.4	15954.6	18151.8
SIS	21	15699	294	6106.2	4890.5	1211.5	6598	1741	40	1849	1227	43443	2956	1227	1722
	22	18840	410	7788.7	6297.5	1485.3	8613	2287	23	2421	1595	52662	3850	1595	2233
	23	16449	343	6977.4	5607.1	1366.8	7459	1982	26	2131	1338	46627	3309	1338	1961
	24	18546	460	9535.6	7899.2	1630.3	10070	2672	34	2884	1808	56219	4468	1808	2679
	25	13239	337	5329.1	4296.6	1029.2	6578	1753	26	1853	1193	37911	2916	1193	1700
Avg.		16554.6	368.8	7147.4	5798.2	1344.6	7863.6	2087	29.8	2227.6	1432.2	47372.4	3499.8	1432.2	2059
Recursive	21	16831	294	37201.4	32767.3	4417.0	52297	16979	-	18339	-	170040	16971	15611	16955
	22	19763	410	41650	36417.7	5210.1	60312	19600	-	21112	-	189879	19568	17830	19536
	23	17315	343	38684.6	34149.4	4521.7	52585	17042	-	18501	-	170082	17030	15546	17018
	24	19890	460	46196.8	40725.8	5449.5	60571	19576	-	21419	-	198351	19566	17509	19552
	25	13667	337	24635	21314.5	3310.4	40899	13143	-	14613	-	129295	13113	11834	13083
Avg.		17493.2	368.8	37673.6	33074.9	4581.7	53332.8	17268	-	18796.8	-	171529.4	17249.6	15666	17228.8

Table 2 Comparison between the different algorithms for historical instances, class C320. Times are reported in seconds.

Algorithm	Ins	nNDP	nIPF	TT	IPT	LPT	nIP	nLex	nMin	nScal	nGood	nLP	nBox	nSIS	nZL
Basic	A	15002	5001	3996.1	1803.6	1875.2	52010	14653	398	22306	-	104487	14613	3259	14567
	B	15002	5001	3853.1	1687.7	1849.7	51738	14561	493	22123	-	103984	14519	3154	14475
	C	15002	5001	3981.6	1850.5	1819.9	51695	14544	518	22089	-	103755	14495	3193	14446
	D	15002	5001	3842.0	1699.0	1824.9	51744	14601	449	22093	-	104201	14563	3153	14521
	E	15002	5001	3948.9	1767.4	1864.2	51745	14574	474	22123	-	104228	14542	3116	14504
Avg.		15002	5001	3924.3	1761.6	1846.8	51786.4	14586.6	466.4	22146.8	-	104131	14546.4	3175	14502.6
SIS	A	15002	5001	4454.9	2278.3	1852.4	53523	16157	326	20820	63	113195	16218	63	16064
	B	15002	5001	4909.8	2623.9	1976.3	53881	16322	386	20749	102	113185	16397	102	16191
	C	15002	5001	4267.5	1969.3	1973.5	53239	16081	422	20592	63	112113	16126	63	15954
	D	15002	5001	4334.7	2025.1	1985.4	53682	16209	373	20798	93	113018	16277	93	16092
	E	15002	5001	4642.9	2328.5	1982.3	53552	16187	391	20704	83	112541	16239	83	16065
Avg.		15002	5001	4522.0	2245.0	1954.0	53575.4	16191.2	379.6	20732.6	80.8	112810.4	16251.4	80.8	16073.2
Recursive	A	15002	5001	2861.8	1383.9	1184.6	29711	3638	-	22435	-	72937	3616	1	3594
	B	15002	5001	2788.4	1301.3	1192.8	29587	3614	-	22359	-	72826	3594	1	3574
	C	15004	5001	2827.3	1350.4	1184.6	29538	3597	-	22344	-	72824	3578	1	3559
	D	15002	5001	2788.5	1306.2	1187.8	29664	3630	-	22404	-	72986	3608	1	3586
	E	15002	5001	2851.8	1367.3	1189.4	29641	3689	-	22263	-	72909	3664	1	3639
Avg.		15002.4	5001	2823.6	1341.8	1187.8	29628.2	3633.6	-	22361	-	72896.4	3612	1	3590.4

Table 3 Comparison between the different algorithms for generated instances, $n = 5000$. Times are reported in seconds.

Algorithm	Ins	nNDP	nIPF	TT	IPT	LPT	nIP	nLex	nMin	nScal	nGood	nLP	nBox	nSIS	nZL
Basic	A	22502	7501	9268.1	4379.1	4008.0	81466	21893	669	37011	-	157416	21833	8149	21765
	B	22502	7501	9093.0	4228.6	3981.7	81333	21847	705	36934	-	157264	21796	8101	21741
	C	22502	7501	8912.3	4065.1	3984.0	81446	21847	704	37048	-	157339	21797	8192	21747
	D	22502	7501	8460.9	3818.5	3814.2	81301	21807	746	36941	-	157198	21757	8058	21705
	E	22502	7501	8672.5	4058.3	3786.0	81444	21840	708	37056	-	157230	21794	8240	21742
Avg.		22502	7501	8881.4	4109.9	3914.8	81398	21846.8	706.4	36998	-	157289.4	21795.4	8148	21740
SIS	A	22502	7501	11346.9	6662.2	3791.6	79742	24039	595	30955	114	168338	24123	114	23916
	B	22502	7501	12163.8	7132.0	4131.0	79662	23994	591	30960	123	167968	24073	123	23854
	C	22501	7501	12520.1	7754.7	3879.3	79925	24106	595	31013	105	168628	24178	105	23968
	D	22502	7501	12395.5	7320.7	4155.7	79593	23959	628	30944	103	168028	24034	103	23817
	E	22503	7501	11874.0	6862.9	4129.8	79772	24007	621	31027	110	168178	24086	110	23889
Avg.		22502	7501	12060.1	7146.5	4017.5	79738.8	24021	606	30979.8	111	168228	24098.8	111	23888.8
Recursive	A	22502	7501	6544.4	3209.7	2535.2	44466	5305	-	33856	-	109695	5278	1	5251
	B	22502	7501	6329.9	3016.2	2517.2	44485	5382	-	33721	-	109677	5345	1	5308
	C	22502	7501	6341.6	2964.6	2562.0	44445	5257	-	33931	-	109532	5228	1	5199
	D	22502	7501	6172.0	2822.3	2538.8	44415	5136	-	34143	-	109698	5099	1	5062
	E	22505	7501	6345.6	2997.1	2538.9	44451	5284	-	33883	-	109541	5252	2	5220
Avg.		22502.6	7501	6346.7	3002.0	2538.4	44452.4	5272.8	-	33906.8	-	109628.6	5240.4	1.2	5208

Table 4 Comparison between the different algorithms for generated instances, $n = 7500$. Times are reported in seconds.

References

Y. P. Aneja and K. P. Nair. Bicriteria transportation problem. *Management Science*, 25(1):73–78, 1979.

J. L. Cohon. *Multiobjective programming and planning*. Academic Press, 1978.