Comparison of pruning tests for interval B&B global optimization methods

Boglárka G.-Tóth, José Fernández and Leocadio González Casado

30 September, 2008

SCAN 2008

Outline

Outline

1. Interval B&B method
2. Pruning Method with Support Functions
3. Baumann Tent Pruning-Dividing Method
4. Comparison of the methods
5. Some ways to combine the methods

Notation

- $\boldsymbol{x} = [\underline{x}, \overline{x}] \subseteq \mathbb{R}$ is a real interval, $\mathbb{I}$ is the set of real intervals.
- $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in \mathbb{I}^n = \mathbb{I} \times \ldots \times \mathbb{I}$ is called a box.
- $\mathrm{w}(\boldsymbol{x}) = \overline{x} - \underline{x}$ is the width of the box
- $f(\boldsymbol{x}) = \{f(x) : x \in \boldsymbol{x}\}$ denotes the range of f over $\boldsymbol{x}$.

Arithmetic operations

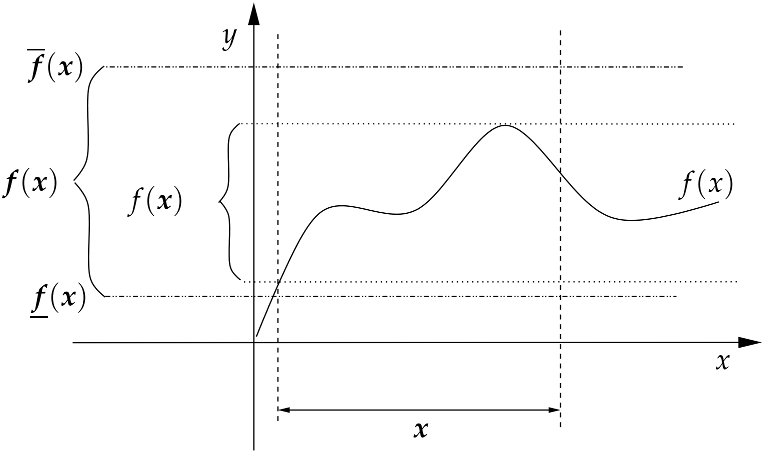
Interval arithmetic operations are defined by

$$\boldsymbol{x} * \boldsymbol{y} = \{x * y : x \in \boldsymbol{x}, y \in \boldsymbol{y}\} \text{ for } \boldsymbol{x}, \boldsymbol{y} \in \mathbb{I},$$

where $* \in \{+, -, \cdot, /\}$, and $\boldsymbol{x}/\boldsymbol{y}$ is only defined if $0 \notin \boldsymbol{y}$.
For example

$$\boldsymbol{x} + \boldsymbol{y} = [\underline{x} + \underline{y}, \overline{x} + \overline{y}]$$

Inclusion function



Definition

A function $\mathbf{f} : \mathbb{I}^n \to \mathbb{I}$ is said to be an *inclusion function* for $f : \mathbb{R}^n \to \mathbb{R}$ provided that $f(\mathbf{x}) \subseteq \mathbf{f}(\mathbf{x})$ for all boxes $\mathbf{x} \subset \mathbb{I}^n$ within the domain of f.

Interval B&B Method: The prototype algorithm

$\mathcal{L}_{\mathcal{W}} \leftarrow \S$, $\mathcal{L}_{\mathcal{S}} \leftarrow \emptyset$

while ($\mathcal{L}_{\mathcal{W}} \neq \emptyset$) **do**

  Select an interval $\boldsymbol{x}$ from $\mathcal{L}_{\mathcal{W}}$ — *Selection Rule*

  Compute $f(\boldsymbol{x})$ — *Bounding Rule*

  if ($\boldsymbol{x}$ cannot be discarded) — *Elimination Rule*

    Divide $\boldsymbol{x}$ into subintervals $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^p$ — *Division Rule*

    for $i = 1, \ldots, p$ **do**

      if ($\boldsymbol{x}^i$ satisfies the termination criterion) — *Termination Rule*

        Store $\boldsymbol{x}^i$ in $\mathcal{L}_{\mathcal{S}}$

      else

        Store $\boldsymbol{x}^i$ in $\mathcal{L}_{\mathcal{W}}$

return $\mathcal{L}_{\mathcal{S}}$

Interval B&B Method: The prototype algorithm

$\mathcal{L}_\mathcal{W} \leftarrow \S$, $\mathcal{L}_\mathcal{S} \leftarrow \emptyset$

while ($\mathcal{L}_\mathcal{W} \neq \emptyset$) **do**

Select an interval $\boldsymbol{x}$ from $\mathcal{L}_\mathcal{W}$ — *Selection Rule*

Compute $\boldsymbol{f}(\boldsymbol{x})$ — *Bounding Rule*

if (**$\boldsymbol{x}$ cannot be discarded**) — *Elimination Rule*

Divide $\boldsymbol{x}$ into subintervals $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^p$ — *Division Rule*

for $i = 1, \ldots, p$ **do**

if ($\boldsymbol{x}^i$ satisfies the termination criterion) — *Termination Rule*

Store $\boldsymbol{x}^i$ in $\mathcal{L}_\mathcal{S}$

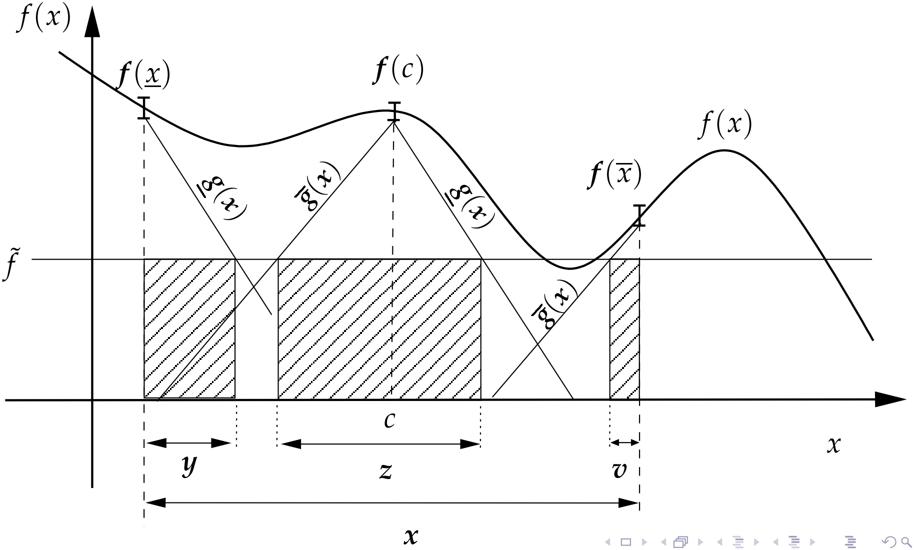
else

Store $\boldsymbol{x}^i$ in $\mathcal{L}_\mathcal{W}$

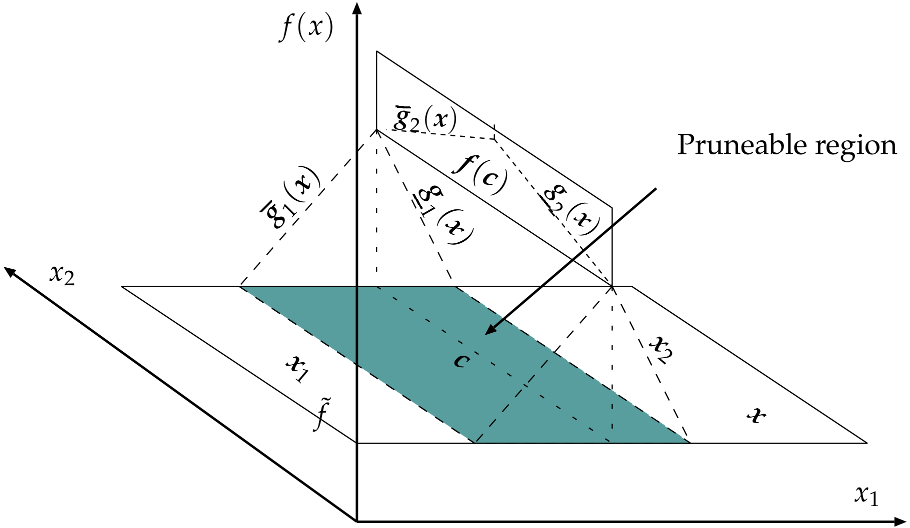
return $\mathcal{L}_\mathcal{S}$

Outline

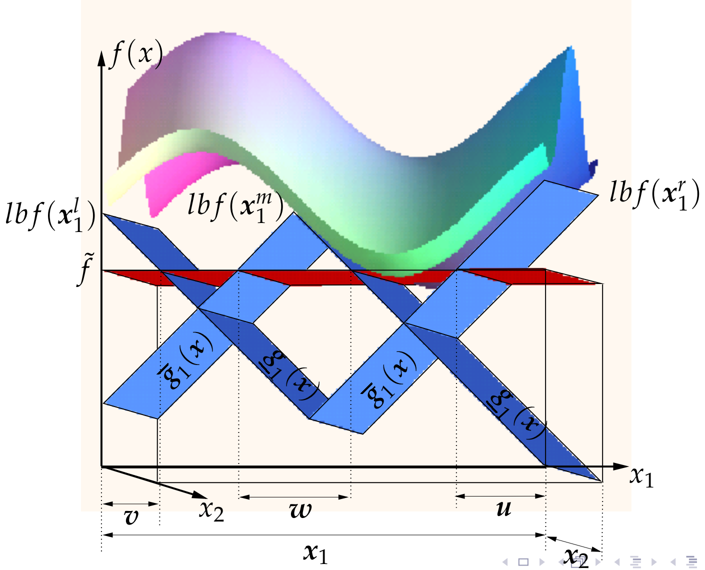
Pruning Method with Support Functions: One-dimensional case



Pruning Method with Support Functions: Multi-dimensional case



Pruning Method with Support Functions: Multi-dimensional case



Pruning Method with Support Functions: Pseudo-code

Calculate $RejectArea[i],\ i = 1 \dots n$ using the centered form to obtain $lbf(X_i^m)$.

if (pruning is possible)

Prune (and divide) the coordinate which maximize the rejected area.

else

Evaluate the natural inclusion if it was better until now for the highest lower bound $lbf(X_i^m)$.

if (pruning is not possible using the new lower bound)

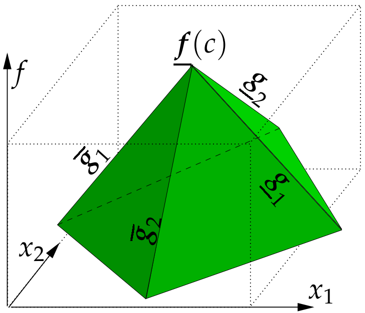
Divide using Ratz and Csendes' Rule C:

$$\max_i \mathrm{w}(\boldsymbol{g}_i(\boldsymbol{x})(\boldsymbol{x}_i - \mathrm{m}(\boldsymbol{x}_i))).$$

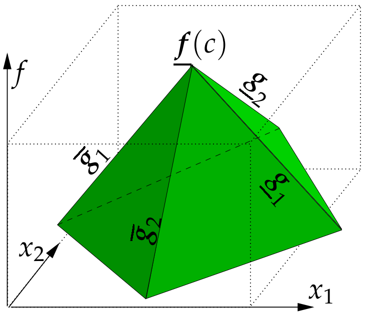
return the generated subintervals with their Support Vectors.

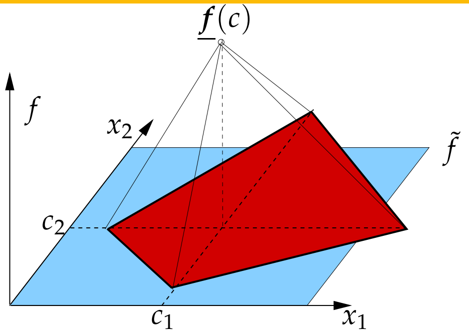
Outline

Baumann Tent Pruning-Dividing Method

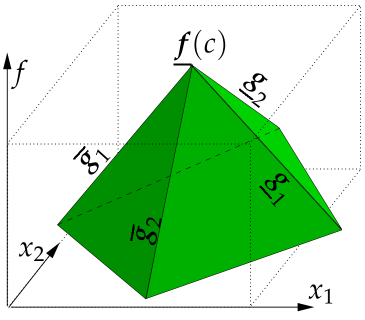


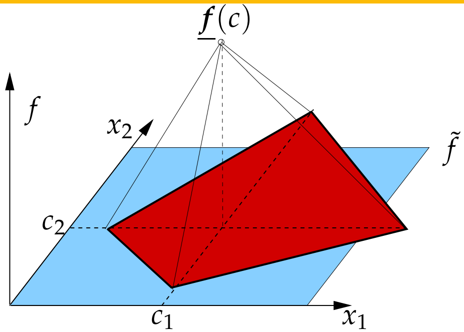
Baumann Tent Pruning-Dividing Method

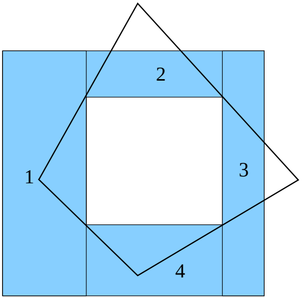




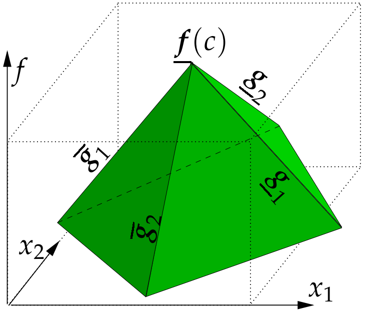
Baumann Tent Pruning-Dividing Method

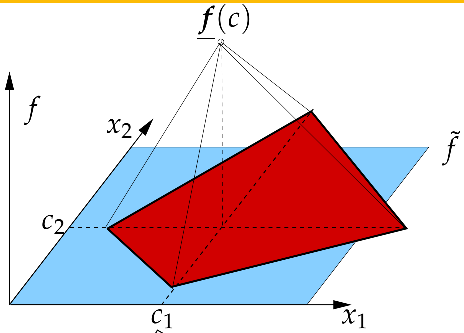


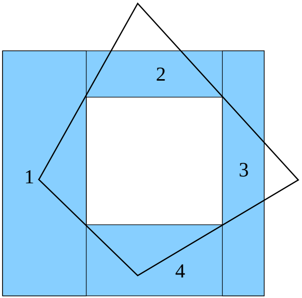


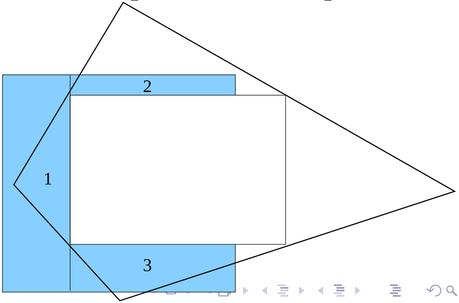


Baumann Tent Pruning-Dividing Method

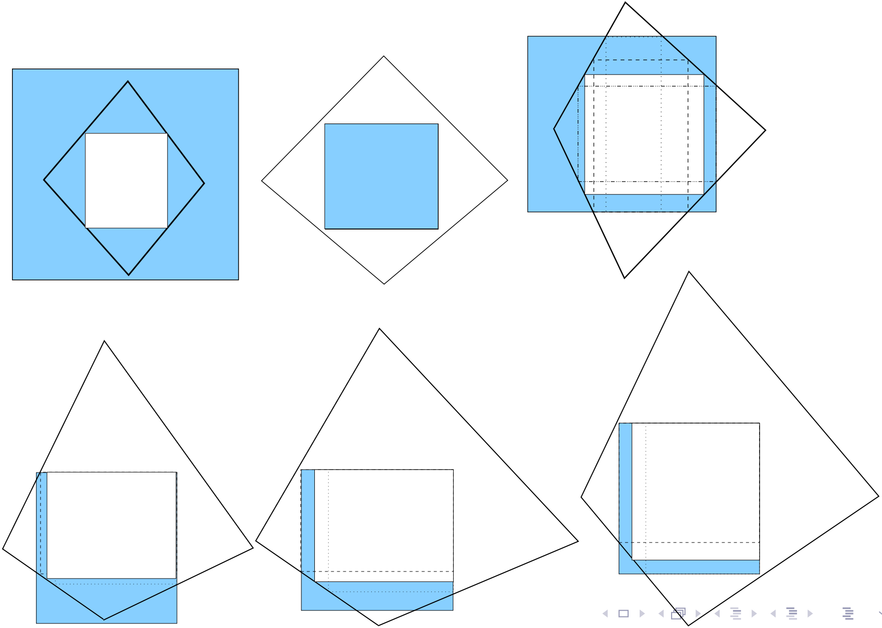




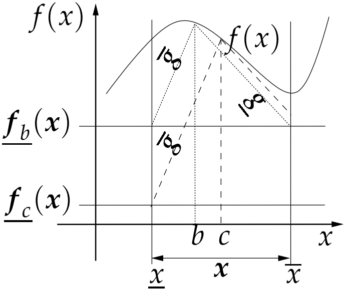




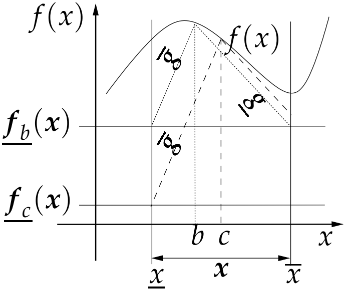
Baumann Tent Pruning-Dividing Method

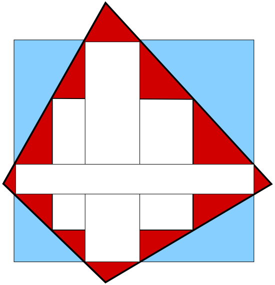


Baumann Tent Pruning-Dividing Method

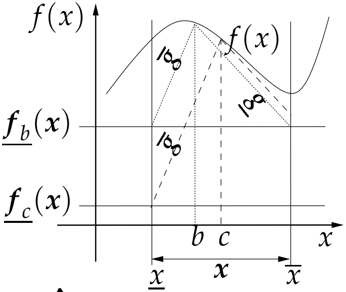


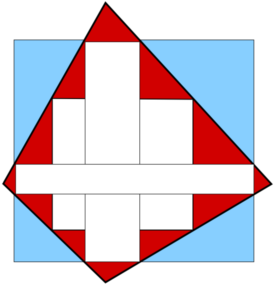
Baumann Tent Pruning-Dividing Method

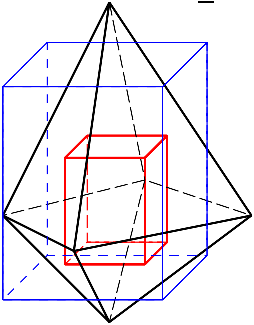




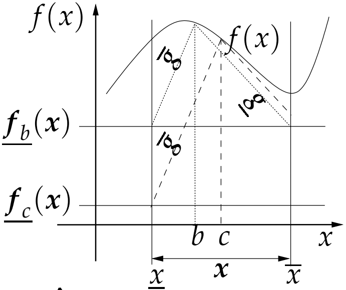
Baumann Tent Pruning-Dividing Method

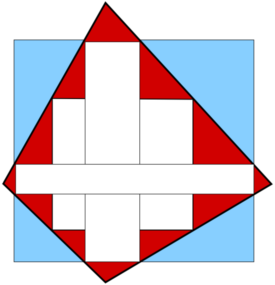


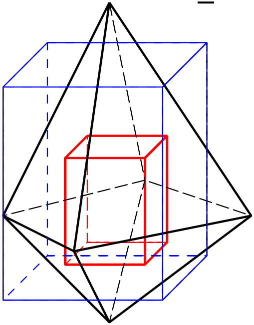


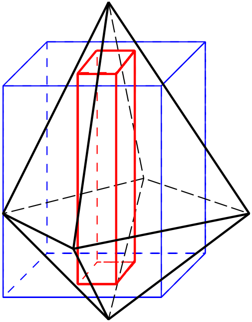


Baumann Tent Pruning-Dividing Method

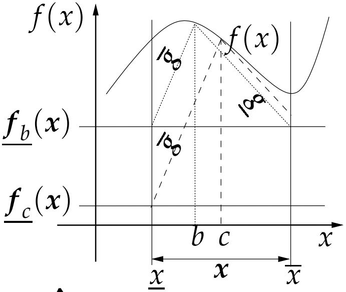


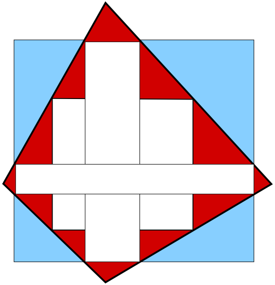


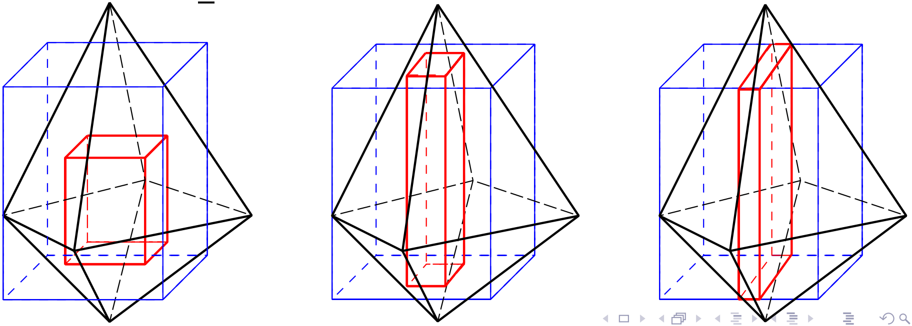




Baumann Tent Pruning-Dividing Method







Baumann Tent Pruning-Dividing Method: Pseudo-code

Compute the shifting factor vector sf, and sort it to ssf

out = # directions where the middlebox is out, and $pref = \frac{vol(SPB_{out})}{2n-2out}$;

for ($k = out + 1; k < n; k++$) **do**

  if (it is possible to shift more with ssf_k)

    Compute $vol(SPB_k)$ from $vol(SPB_{k-1})$ shifting with ssf_k;

    if $\left(\frac{vol(SPB_k)}{2n-2k} > pref\right)$

      $pref = \frac{vol(SPB_k)}{2n-2k}$;

      $prefshift = k$;

  else break;

if ($pref > PruningIndex$)

  Prune the box $SPB_{prefshift}$, generating the remaining boxes;

else

  Divide the box into subintervals;

return the generated subintervals;

Outline

Comparison of the methods

Componentwise Pruning

Advantages

- Higher speedup (2.37)
- Always two box generated
- Natural or centered form use

Disadvantages

- High storing necessity
- More evaluation for easy problems

Baumann Tent Pruning

Advantages

- Low storing necessity
- No additional function evaluation
- Adaptive division

Disadvantages

- Lower speedup (1.84)
- Bad shaped and too many new boxes

OUTLINE

How to combine the methods

1. We can do first the componentwise pruning, and when the centered form becomes better, change to the Baumann Tent pruning.

How to combine the methods

1. We can do first the componentwise pruning, and when the centered form becomes better, change to the Baumann Tent pruning.

How to combine the methods

1. We can do first the componentwise pruning, and when the centered form becomes better, change to the Baumann Tent pruning.
2. When the Baumann point is nearer to the border than the center (so when we would get bad shaped boxes), do the componentwise pruning.

How to combine the methods

1. We can do first the componentwise pruning, and when the centered form becomes better, change to the Baumann Tent pruning.

2. When the Baumann point is nearer to the border than the center (so when we would get bad shaped boxes), do the componentwise pruning. It is necessary to compute support functions for the new boxes from Baumann Tent pruning. See here.

How to combine the methods

1. We can do first the componentwise pruning, and when the centered form becomes better, change to the Baumann Tent pruning.
2. When the Baumann point is nearer to the border than the center (so when we would get bad shaped boxes), do the componentwise pruning. It is necessary to compute support functions for the new boxes from Baumann Tent pruning. See here.
3. Decide on which to use by computing the pruneable region by both method and choose the better $\frac{vol}{\#new\ boxes}$.

How to combine the methods

1. We can do first the componentwise pruning, and when the centered form becomes better, change to the Baumann Tent pruning.
2. When the Baumann point is nearer to the border than the center (so when we would get bad shaped boxes), do the componentwise pruning. It is necessary to compute support functions for the new boxes from Baumann Tent pruning. See here.
3. Decide on which to use by computing the pruneable region by both method and choose the better $\frac{vol}{\#new\ boxes}$. It can be inefficient with so much computation. The support functions have to be computed for the new boxes of the Baumann Tent pruning.

How to combine the methods

1. We can do first the componentwise pruning, and when the centered form becomes better, change to the Baumann Tent pruning.
2. When the Baumann point is nearer to the border than the center (so when we would get bad shaped boxes), do the componentwise pruning. It is necessary to compute support functions for the new boxes from Baumann Tent pruning. See here.
3. Decide on which to use by computing the pruneable region by both method and choose the better $\frac{vol}{\#new\ boxes}$. It can be inefficient with so much computation. The support functions have to be computed for the new boxes of the Baumann Tent pruning.
4. Do the componentwise pruning only from the middle, so no storing of the support vectors are necessary, and we can use the 2nd or 3rd idea.

REFERENCES

J. A. Martínez, L. G. Casado, I. García, and B. Tóth.
AMIGO: Advanced multidimensional interval analysis global optimization algorithm.
In *Frontiers in global optimization*, volume 74 of *Nonconvex Optimization and Its Applications*, pages 313–326. Kluwer, Dordrecht, 2004.

B. Tóth and L. G. Casado.
Multi-dimensional pruning from the Baumann point in an interval global optimization algorithm.
Journal of Global Optimization, 38:215–236, 2007.