Computational strategies for regression model selection

C. Gatu[1], M. Hofmann[2], E. Kontoghiorghes[3,4]

[1] "Alexandru Ioan Cuza" University of Iași, Romania

[2] University of Oviedo, Spain

[3] Cyprus University of Technology, Cyprus

[4] Birkbeck University of London, UK

Limassol, 5th of April 2018

Content

Linear regression model

$$y \quad = \quad a_1 \times \beta_1 \quad + \quad a_2 \times \beta_2 \quad + \quad \dots \quad + \quad a_n \times \beta_n \quad + \quad \varepsilon$$

y, $a_1, \dots, a_n$: variables (data)

$\beta_1, \dots, \beta_n$: parameters (unknown)

ε: error

Linear regression model

$$y = a_1 \times \beta_1 + a_2 \times \beta_2 + \ldots + a_n \times \beta_n + \varepsilon$$

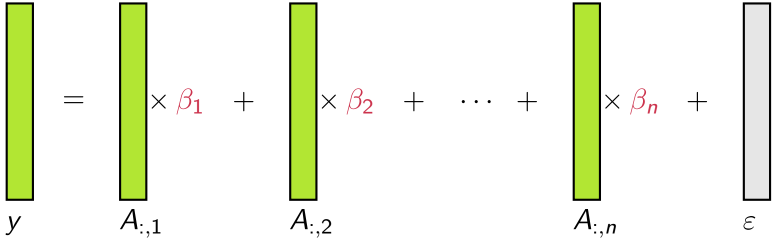
$y, a_1, \ldots, a_n$: variables (data)

$\beta_1, \ldots, \beta_n$: parameters (unknown)

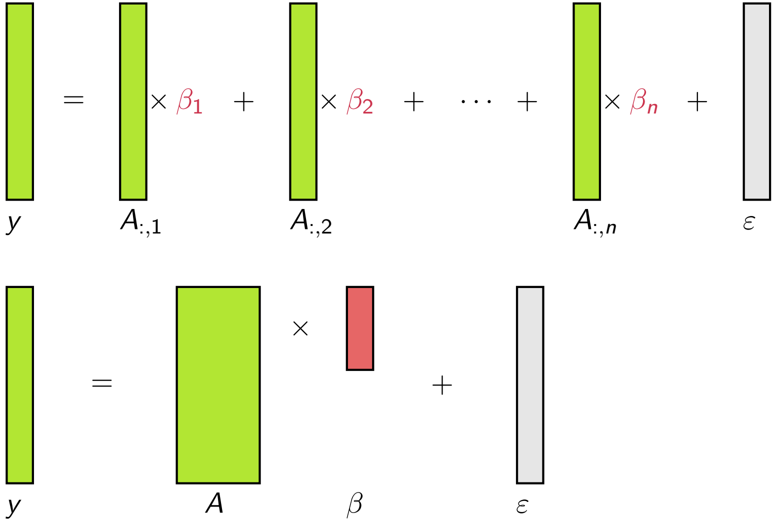
ε: error

$$\begin{array}{ccccccccccc}
y_1 & = & a_{1,1}\,\beta_1 & + & a_{1,2}\,\beta_2 & + & \ldots & + & a_{1,n}\,\beta_n & + & \varepsilon_1 \\
y_2 & = & a_{2,1}\,\beta_1 & + & a_{2,2}\,\beta_2 & + & \ldots & + & a_{2,n}\,\beta_n & + & \varepsilon_2 \\
\vdots & & \vdots & & \vdots & & & & \vdots & & \vdots \\
y_m & = & a_{m,1}\,\beta_1 & + & a_{m,2}\,\beta_2 & + & \ldots & + & a_{m,n}\,\beta_n & + & \varepsilon_m
\end{array}$$

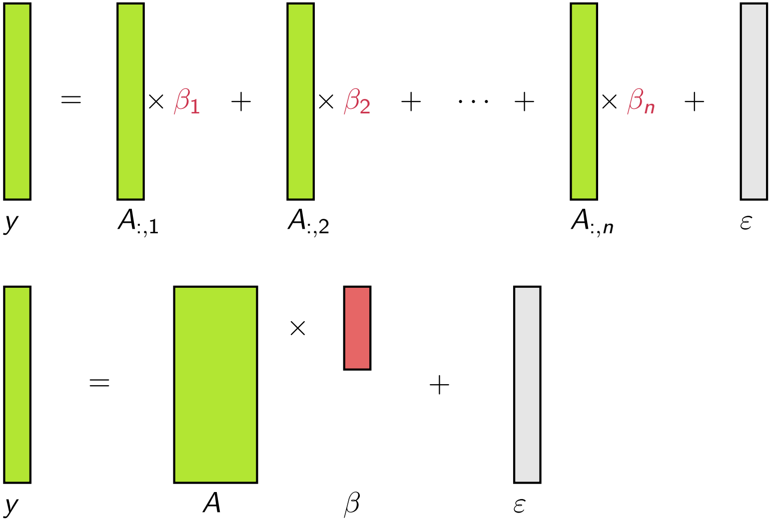
Matrix notation



Matrix notation



Matrix notation



$$y = A\beta + \varepsilon, \qquad e \sim (0, \sigma^2\Omega)$$

Least squares estimation

- Ordinary Linear Model (OLM)

$$y = A\beta + \varepsilon, \quad \varepsilon \sim (0, \sigma^2 I_m),$$

where $y \in \mathbb{R}^m$, $A \in \mathbb{R}^{m\times n}$ and $\varepsilon \in \mathbb{R}^m$.

- Ordinary least squares (OLS):

$$\hat{\beta} = \operatorname*{argmin}_{\beta} \|y - A\beta\|^2 \quad = (A^T A)^{-1} A^T y.$$

- Residual sum of squares (RSS):

$$\text{RSS}(\hat{\beta}) = \|y - A\hat{\beta}\|^2$$

OLS and QR Decomposition

- QR decomposition:

$$Q^T \begin{bmatrix} A & y \end{bmatrix} = \begin{bmatrix} R & z \\ 0 & \rho \\ 0 & 0 \end{bmatrix} \begin{matrix} n \\ 1 \\ m-n-1 \end{matrix},$$

  where R is upper-triangular.

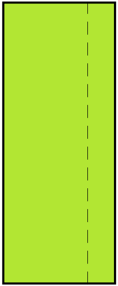
- OLS estimator:

$$\hat{\beta} = R^{-1} z.$$

- Residual sum of squares:

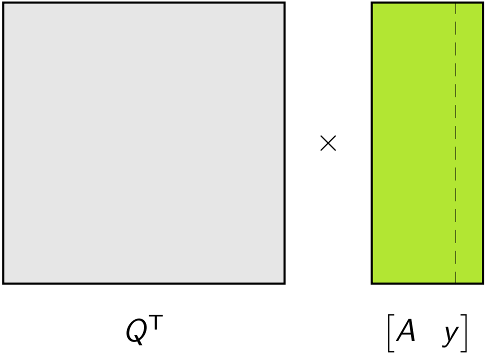
$$\text{RSS}(\hat{\beta}) = \rho^2.$$

QR decomposition

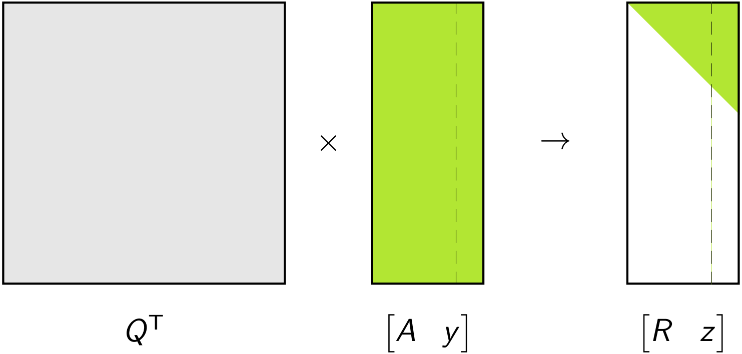


$$\begin{bmatrix} A & y \end{bmatrix}$$

QR decomposition



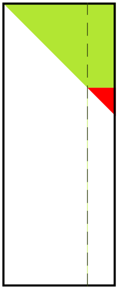
QR decomposition



QR decomposition

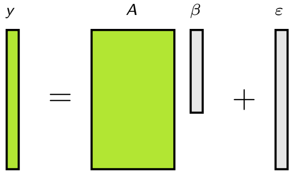
$$\hat{\beta} = R^{-1}z$$

$$\text{RSS}(\hat{\beta}) = \rho^2$$



$$\begin{bmatrix} R & z \end{bmatrix}$$

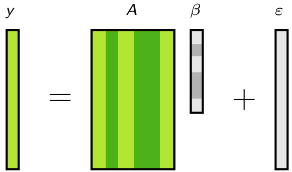
Subset model



Full model F:

OLS: $\hat{\beta}$, RSS(F)

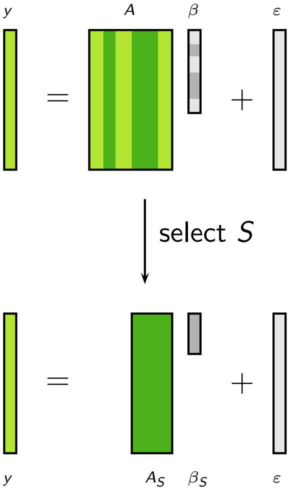
Subset model



Full model F:

OLS: $\hat{\beta}$, RSS(F)

Subset model



<u>Full model F:</u>
OLS: $\hat{\beta}$, RSS(F)

<u>Subset model S:</u>
OLS: $\hat{\beta}_S$, RSS(S) $\geq$ RSS(F)

LS estimation of a subset linear model

- Let $A_S = AS$, $\beta_S = S^T\beta$, $S_{n\times k}$ – selection matrix.

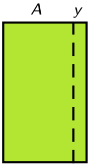
- $y = A_S\beta_S + \varepsilon \quad \longrightarrow \quad z = RS\beta_S + \zeta; \quad \zeta \backsim (0, \sigma^2 I_n)$

- QR decomposition:

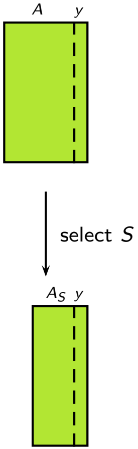
$$Q_S^T \begin{bmatrix} RS & y \end{bmatrix} = \begin{bmatrix} R_S & z_S \\ 0 & \rho_S \\ 0 & 0 \end{bmatrix} \begin{matrix} k \\ 1 \\ n-k-1 \end{matrix},$$

- $\hat{\beta}_S = R_S^{-1} z_S$ and $\mathrm{RSS}(\hat{\beta}_S) = \mathrm{RSS}(\hat{\beta}) + \rho_S^2$.

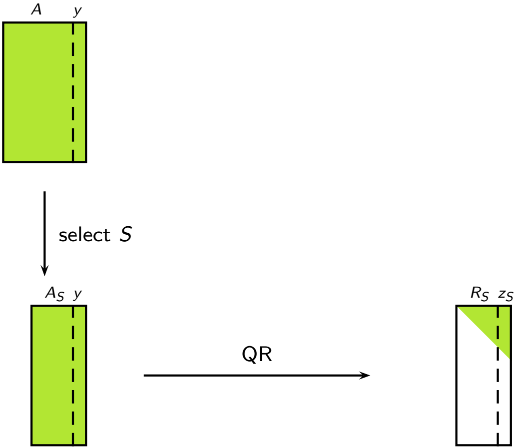
QR downdating



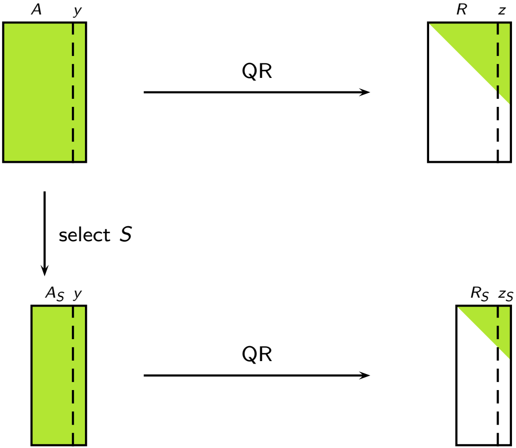
QR downdating



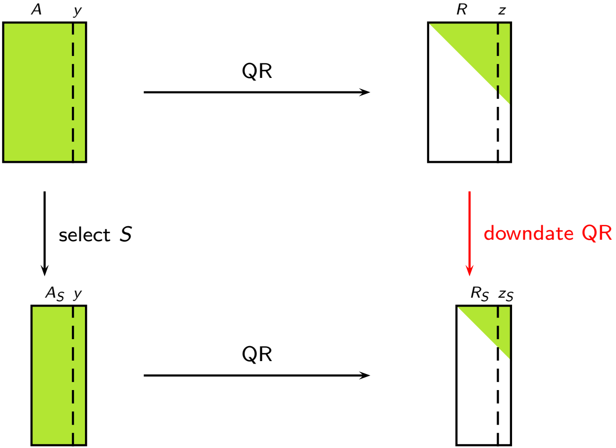
QR downdating



QR downdating



QR downdating



Content

All subset regressions

- Ordinary linear model with n variables:

$$F = \{1, 2, \ldots, n\}$$

- Problem statement:

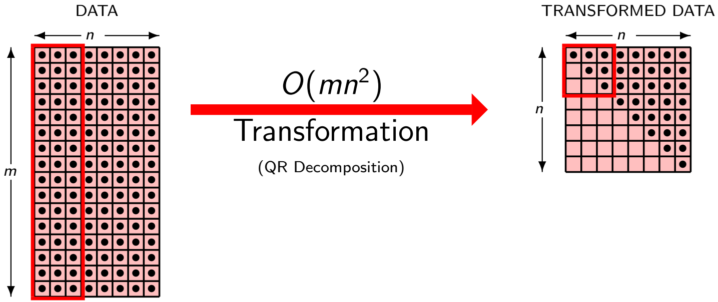
For all subset sizes k, find the *best* subset model S_k^* of size k.

$(k = 1, 2, \ldots, n,\ S_k^* \subset F,\ |S_k^*| = k)$

- *Best*? Smallest RSS.

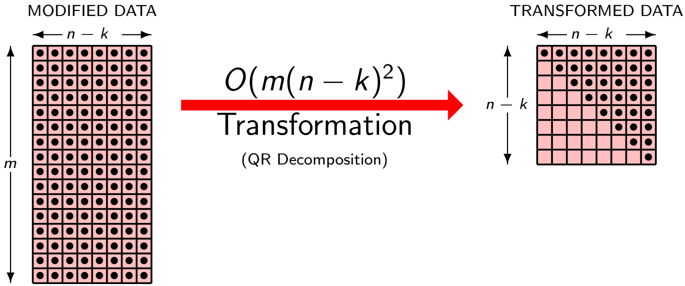
- Computational cost: $2^n - 1$ possible models

Proper subset models



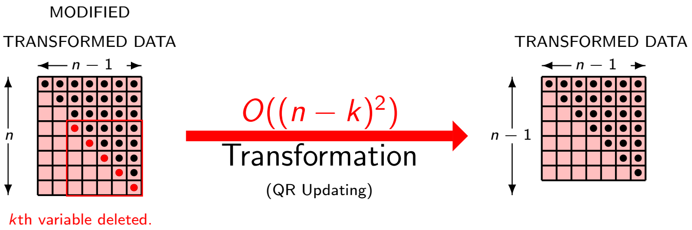
- The transformed data provides the n sub-models with variables: $\{1\}, \{1, 2\}, \{1, 2, 3\}, \ldots, \{1, 2, 3, \ldots, n\}$.
- All $2^n - 1$ sub-models need to be computed.

All subsets: inefficient Approach



- Delete k $(k = 1, \ldots, n-1)$ variables from the original data.
- Repeat for all 2^{n-1} possible combinations of k.
- Inefficient because it does not utilise previous computations.

Utilising previous computation

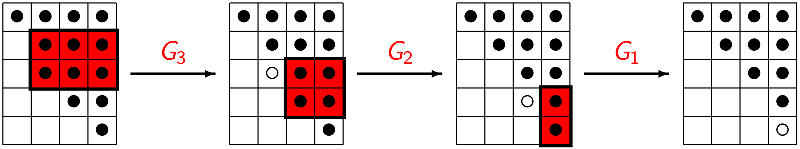


- Delete a variable from the transformed data.
- Re-transformation of the modified data.
- Obtain new sub-models.

- Re-transform the data after deleting a single variable.

Re-estimate after dropping a single column

2nd column dropped



Zero element — Non-zero element — Annihilated element

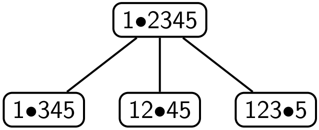
$$C_{\text{Drop}}(\lambda) = t\sum_{j=1}^{\lambda}(j+1) = t(\lambda^2+3\lambda)/2.$$

Tree strategy

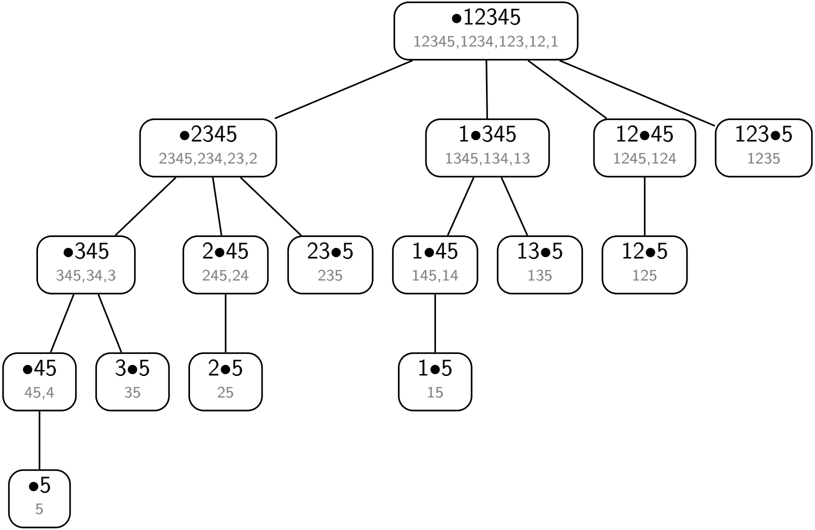
- Use a Tree strategy to generate the $2^n - 1$ sub-sets.
- A node of the Tree corresponds to a set of variables.
- A child node is obtained from its parent by dropping one variable.

Notation

- The variables right of the bullet • will be deleted from the set one at a time. The last element of a set is never deleted.



Regression tree: 2^{n-1} nodes and $2^n - 1$ models



Dropping Column Algorithm (DCA): complexity

- Drop(V, i):

$$C_{\text{Drop}}(n,i) = \sum_{j=1}^{n-i}(j+1) = (n-i)(n-i+3)/2.$$

- Generating the RT having as root $\{v_1 \cdots v_i \bullet v_{i+1} \cdots v_n\}$:

$$C(n-i) = \sum_{j=1}^{n-i-1}\left(C_{\text{Drop}}(n-1, n-i-j) + C(n-i-j)\right)$$
$$= 3 \cdot 2^{(n-i)} - (n-i+2)(n-i+3)/2.$$

- DCA:

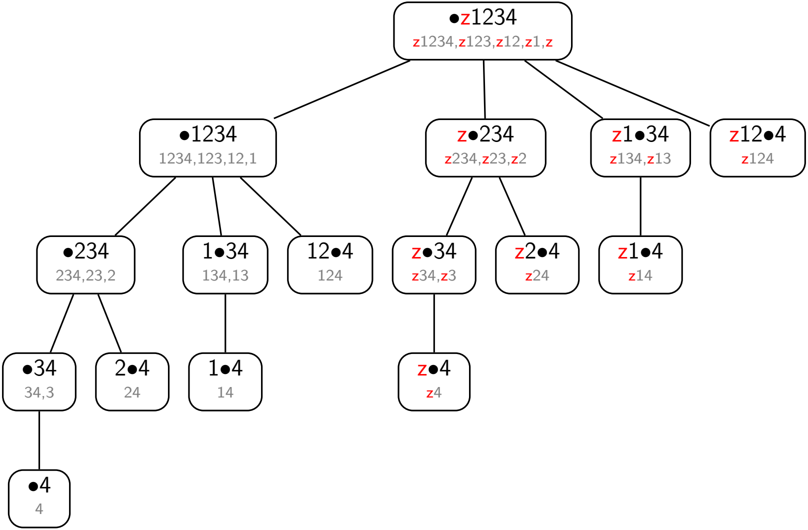
$$C_{\text{DCA}}(n) = C(n) = 3 \cdot 2^n - (n+2)(n+3)/2 = O(2^n).$$

Content

Variable updating

- Consider the updated set $V = \{z, 1, 2, \cdots, n\}$.
- Compute all $2^{n+1} - 1$ subsets.
- Half of the models have been already generated, e.g. the models corresponding to all combinations of the elements $1, 2, \ldots, n$.
- Compute only the models that include the new element.

Variable updating



Content

Subrange selection

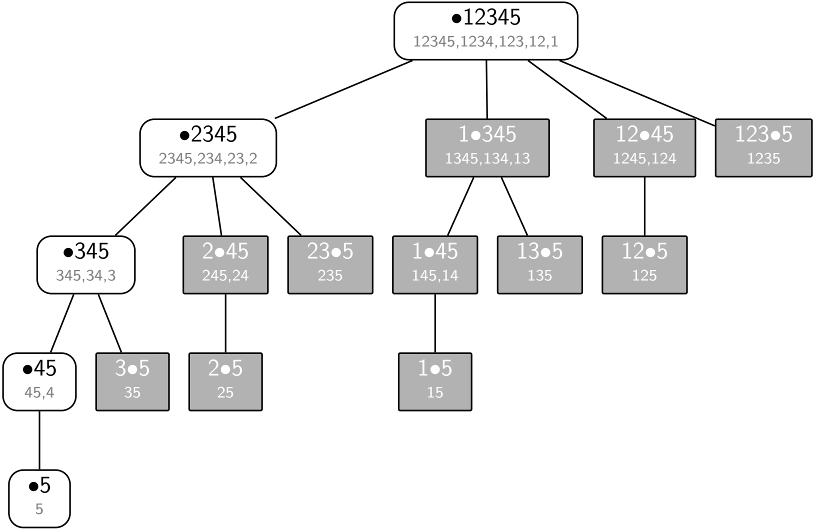
- Given a and b $(1 \leq a \leq b \leq n)$ generate the minimal subtree to find all subset models of size k, $a \leq k \leq b$.

- Δ_k the minimal subtree which generates the submodels of size k.

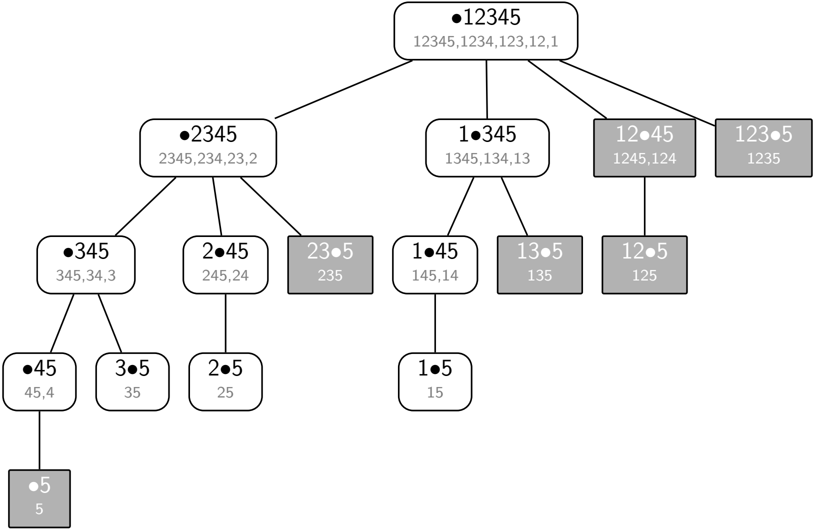
$$\Delta_k(V,i) = \begin{cases} (V,i) & \text{if} \quad k = n - i, \\ \big((V,i), \Delta_k(\text{Drop}(V,i+1),i), \cdots \\ \qquad \cdots, \Delta_1(\text{Drop}(V,i+k), i+k-1)\big) & \text{if} \quad k < n - i. \end{cases}$$

- $\Delta_{a-b}(V,i) = \bigcup_{k=a}^{b} \Delta_k(V,i)$.

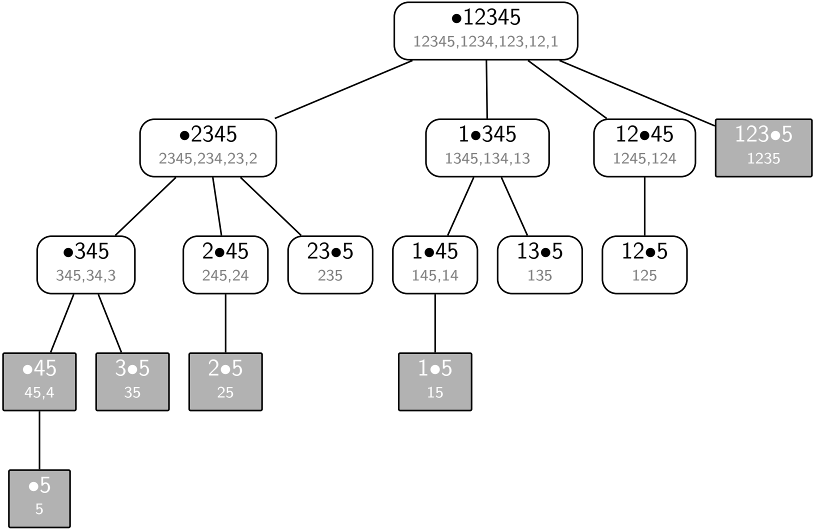
$\Delta_1(\{12345\}, 0)$



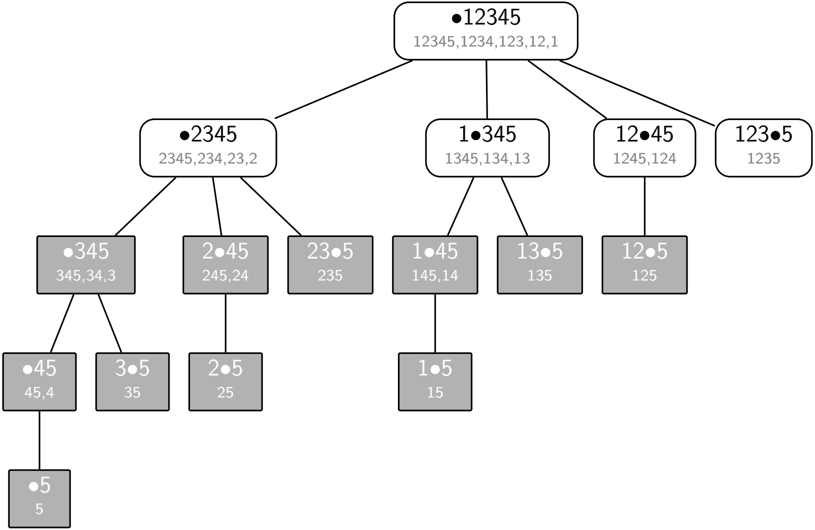
$\Delta_2(\{12345\}, 0)$



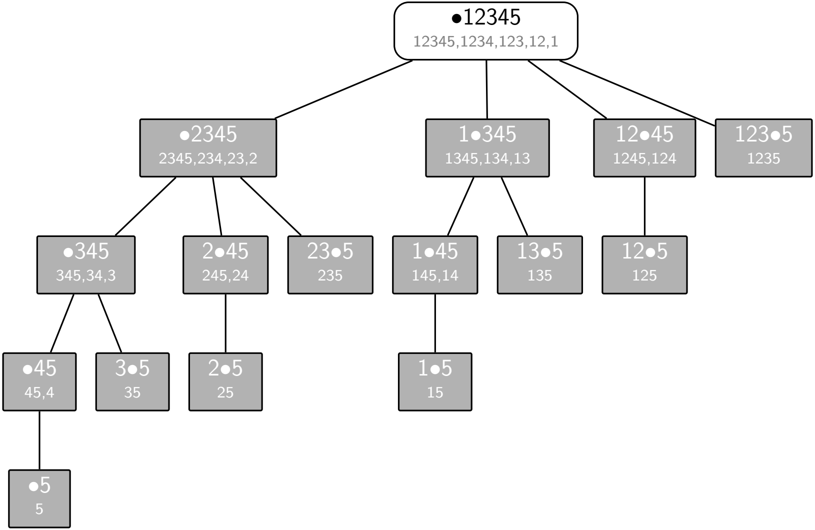
$\Delta_3(\{12345\}, 0)$



$\Delta_4(\{12345\},0)$



$\Delta_5(\{12345\}, 0)$



Computational cost

- nodes

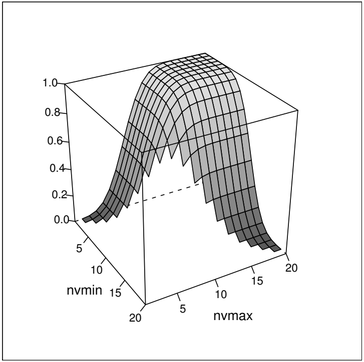
$$N_{a,b}(n) = \sum_{t=a}^{b-1} \binom{n-1}{t-1} + \binom{n}{b};$$

- operations

$$T_{a,b}(n) = \sum_{t=a}^{b-1} \sum_{j=0}^{t-1} \binom{n-t+j-1}{j} T_{\text{drop}}(t-j) + \sum_{j=0}^{b-1} \sum_{i=j}^{n-b+j-1} \binom{i}{j} T_{\text{drop}}(n-i-1).$$

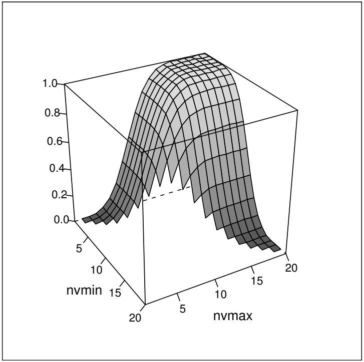
Subrange DCA

number of operations



theoretical, normalized

execution time

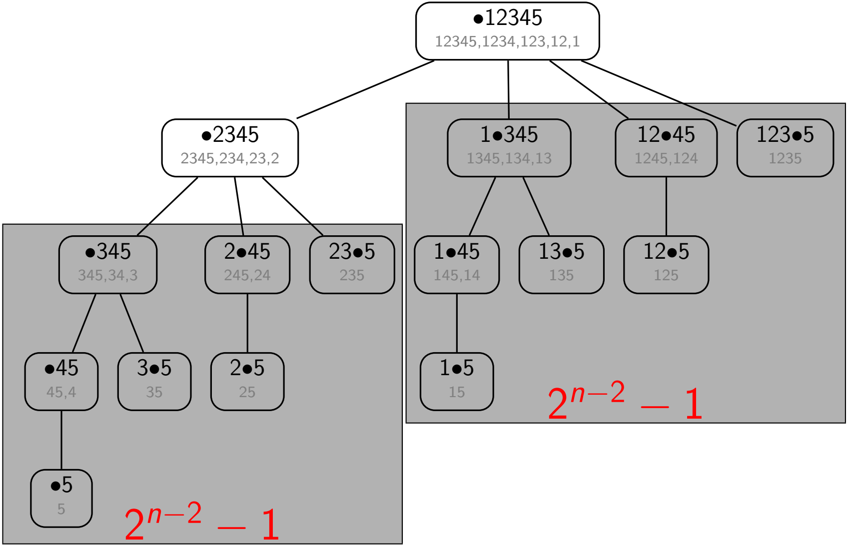


experimental, normalized

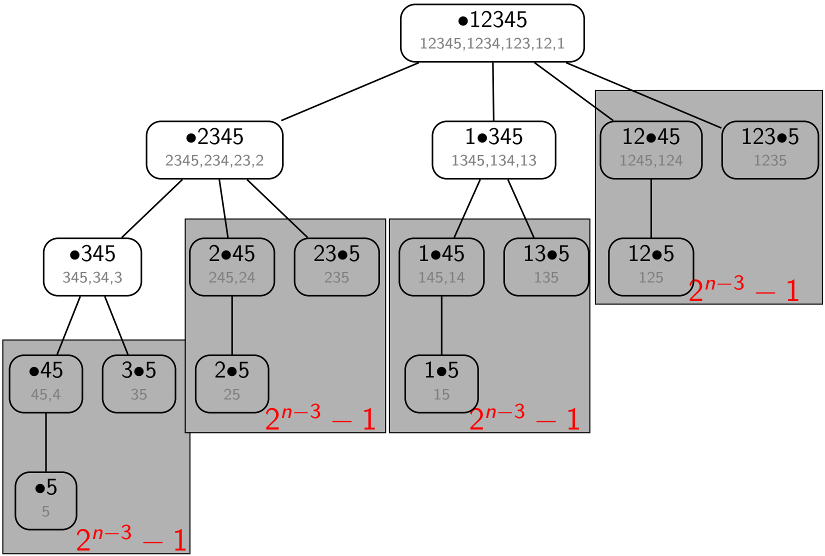
$(n = 20)$

Content

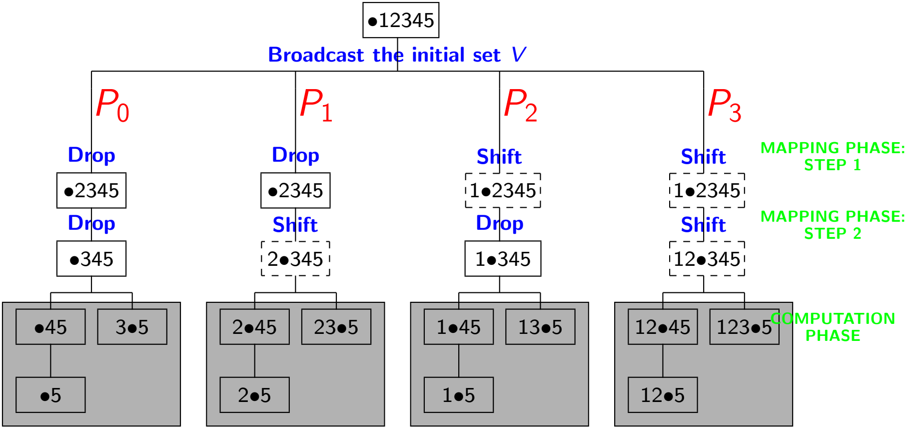
Complexity property of the regression tree



Complexity property of the regression tree



The Parallel Dropping Columns Algorithm (PDCA)



$$C_{\text{PDCA}}(n, 2^\rho) \approx 3 \cdot 2^{n-\rho}$$

$$\text{Speedup}(n, 2^\rho) = C_{\text{DCA}}(n) / C_{\text{PDCA}}(n, 2^\rho) \approx 2^\rho$$

The PDCA using $p = 2^{\varrho}$ processors

initially do:

- Transform the initial data $\longrightarrow V \equiv \{v_1 \cdots v_n\}$
- Obtain the subsets $\{v_1\}, \ldots, \{v_1 \cdots v_n\}$
- $k \longleftarrow 0$
- Broadcast (V, k) to the processors $P_0, \ldots P_{p-1}$

```
r ⟵ rank of the processor
each processor do:
for v = 1, ..., ϱ do
    if ((r div 2^(ϱ−s)) mod 2) = 0 then
        (V, k) ⟵ Drop(V, k)
        Extract the new subsets {v_1 ··· v_{k+1}}, ..., {v_1 ··· v_{k+1} ··· v_{|V|}}
    else
        (V, k) ⟵ Shift(V, k)     // i.e. k ⟵ k − 1
    end if
end for
call Subtree(V, k)
end do
```

The complexity of the PDCA using 2^ρ processors

- Complexity of the mapping phase

$$C_{\text{map}}(n,\rho) = \sum_{j=1}^{\rho} C_{\text{Drop}}(n,j) = \rho\big(3n^2 + 6n - 4 - \rho(3n - \rho + 3)\big)/6$$

- Complexity of the computation phase

$$C(n-\rho) = 3 \cdot 2^{n-\rho} - (n-\rho+2)(n-\rho+3)/2$$

- Complexity of the PDCA

$$\begin{aligned} C_{\text{PDCA}}(n,\rho) &= C(n-\rho) + C_{\text{map}}(n,\rho) \\ &= 3 \cdot 2^{n-\rho} - (n+2)(n+3)/2 + \\ &\quad \rho(3 \cdot n^2 + \rho^2 + 12n - 3n\rho - 6\rho + 11)/6 \\ &\approx 3 \cdot 2^{n-\rho} \end{aligned}$$

- $\text{Speedup}(n, 2^\rho) = C_{\text{DCA}}(n) / C_{\text{PDCA}}(n,\rho) \approx 2^\rho$

Execution times in seconds of the PDCA and PDCA–2

# of Variables	# of Proc.	DCA Serial	PDCA Time	PDCA Effic.	PDCA–2 Time	PDCA–2 Effic.	Theoretical Effic.
15	1	0.600	0.603	0.99	0.610	0.98	1.00
15	2		0.313	0.97	0.310	0.98	0.99
15	4		0.160	0.94	0.157	0.96	0.99
15	8		0.080	0.94	0.080	0.94	0.98
20	1	21.52	21.52	1.00	21.53	0.99	1.00
20	2		11.03	0.98	10.81	0.99	0.99
20	4		5.63	0.96	5.46	0.98	0.99
20	8		2.88	0.93	2.78	0.97	0.99
21	1	43.49	43.59	0.99	43.55	0.99	1.00
21	2		22.54	0.96	22.03	0.98	0.99
21	4		11.41	0.95	11.16	0.96	0.99
21	8		5.83	0.93	5.47	0.98	0.99
25	1	757.28	759.89	0.99	773.71	0.98	1.00
25	2		389.56	0.97	381.54	0.99	0.99
25	4		201.12	0.94	190.97	0.99	0.99
25	8		102.89	0.92	98.55	0.97	0.99

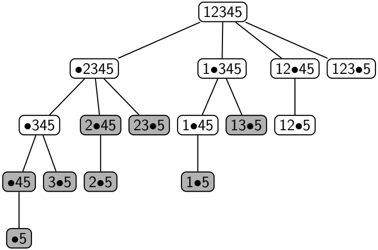
SUN Enterprise 10000

Content

Content

Branch and bound algorithm

- Exhaustive algorithm.
- Exploit statistical information (RSS) to cut subtrees.



Deriving the best sub-models

- For each $V = \{v_1, \cdots, v_n\}$ consider a criterion $f(V)$.
- Problem:

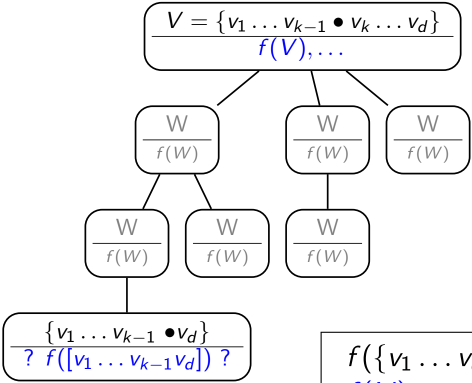
$$\text{for all} \quad p = 1, \ldots, n \quad \text{find} \qquad V_p^* = \underset{|V|=p}{\operatorname{argmin}} f(V)$$

- E.g. $\min(f([1]), f([2]), \ldots, f([n]))$;
  $\min(f([1,2]), f([1,3]), \ldots, f([1,n]), f([2,3]), \ldots)$;
  $\min(f([1,2,3]), f([1,2,4]), \ldots)$; ...
- Objective: Prune non-optimal subtrees.
- Properties:

– if $V_1 \subseteq V_2$ then $\mathrm{RSS}(V_1) \geq \mathrm{RSS}(V_2)$,

– if $|V_1| = |V_2|$ and $\mathrm{RSS}(V_1) \geq \mathrm{RSS}(V_2)$ then $f(V_1) \geq f(V_2)$.

Branch-and-bound: prunning



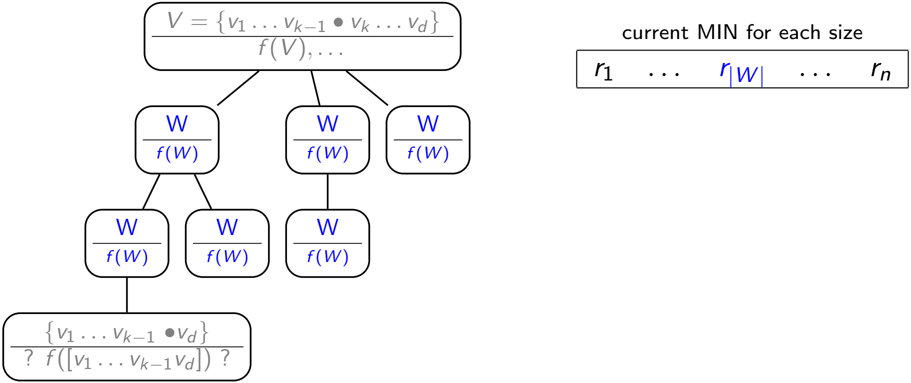
current MIN for each size

r_1	$\dots$	r_k	$\dots$	r_n

$$\left.\begin{array}{l} f(\{v_1 \dots v_d\}) \geq f(V) \\ f(V) \geq r_k \end{array}\right\} \Rightarrow f([v_1 \dots v_d]) \geq r_k$$

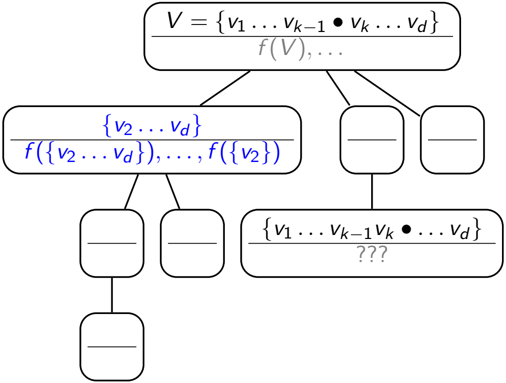
Lemma 1. $r_j^{(g)} \geq r_{j+1}^{(g)}$, *where* $g = 1, \dots, 2^{n-1}$ *and* $j = 1, \dots, n-1$.

Branch-and-bound: prunning



Lemma 2. $r_{|W|} \leq f(W)$, *where* W *is any set obtained from a node of* $T(V, k-1)$, $f(V) \leq r_k$ *and* $1 \leq k < d$.

Branch-and-bound: prunning fails



current MIN for each size

r_1	$\ldots$	r_k	$\ldots$	r_n

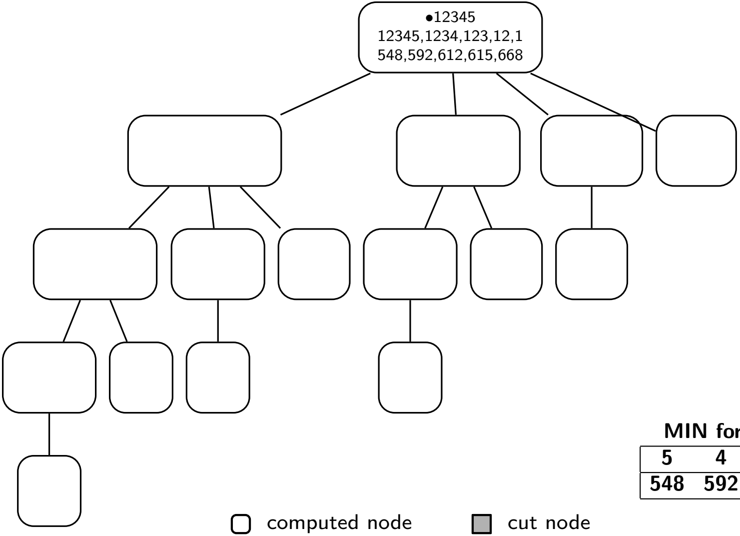
$$f(V) < r_k$$

Branch-and-bound Algorithm

1: Compute the QRD of $A \in \Re^{m\times n}$: $Q^T A = \binom{R}{0}\begin{smallmatrix} n \\ m-n \end{smallmatrix}$ and $Q^T y = \tilde{y}$
2: Let $V = \{1, 2, \ldots, n\}$, $k = 0$ and
$r_j = \text{RSS}(\{1, 2, \ldots, j\}) \equiv \sum_{i=j+1}^{m} \tilde{y}_i^2$ where $j = 1, \ldots, n$
3: **call** ProcessSubtree(V, k)

4: **def** ProcessSubtree$(V, k) =$ **do**
5: **for** $i = k + 1, \ldots, |V| - 1$ **do**
6: **if** $(r_i > \text{RSS}(V))$ **then**
7: $V^{(i)} \leftarrow \text{Drop}(V, i)$
8: $r_j = \min(r_j, \text{RSS}(\{v_1^{(i)}, v_2^{(i)}, \ldots v_j^{(i)}\}))$, where
$j = i, \ldots, |V| - 1$
9: **else**
10: Cut the remaining sub-trees and go to step 13
11: **end if**
12: **end for**
13: **call** ProcessSubtree$(V^{(j)}, j - 1)$, where
$j = k + 1, \ldots, \min(i, |V| - 2)$
14: **end def**

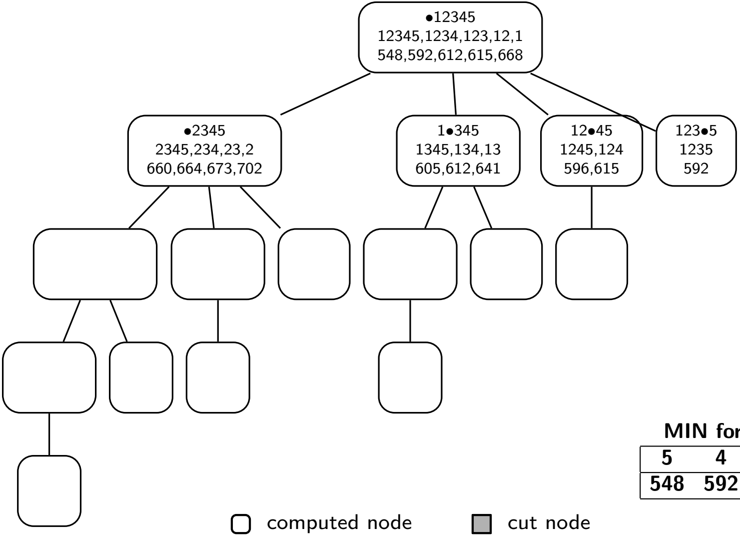
Branch-and-bound algorithm



MIN for # of variables

5	4	3	2	1
548	592	612	615	668

Branch-and-bound algorithm

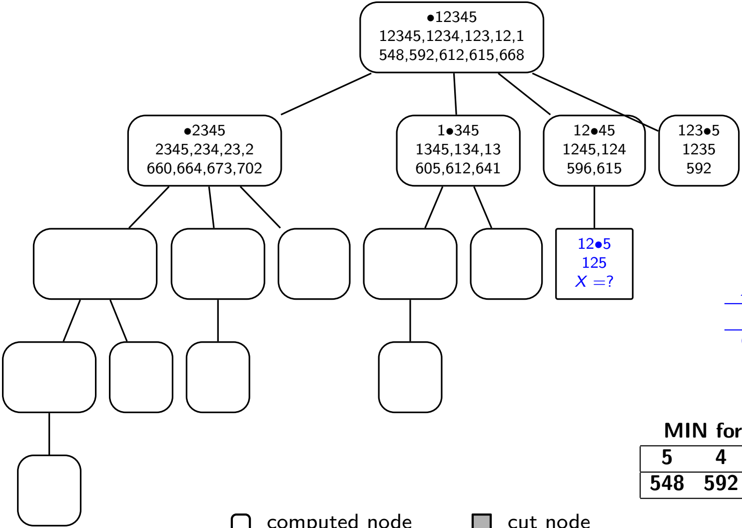


MIN for # of variables

5	4	3	2	1
548	592	612	615	668

computed node cut node

Branch-and-bound algorithm



$f(1245) = 596$
$\text{MIN}_3 = 612$
$X > f(1245)$

$f(1245) < \text{MIN}_3$

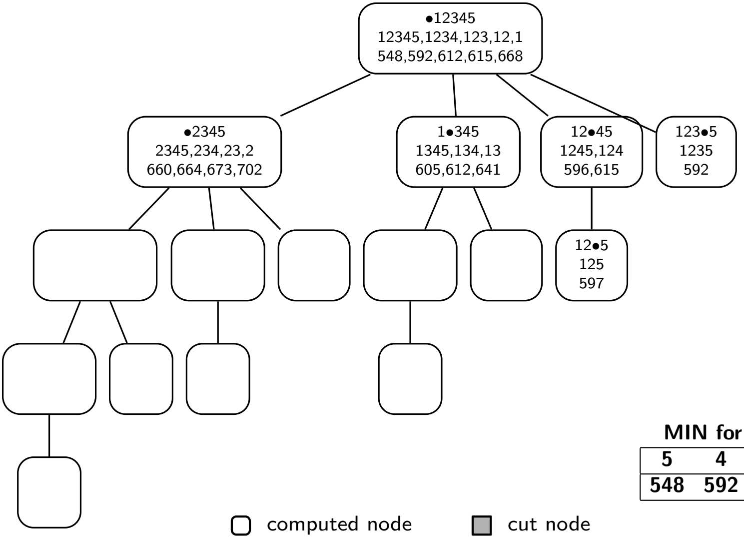
COMPUTE

MIN for # of variables

5	4	3	2	1
548	592	612	615	668

computed node   cut node

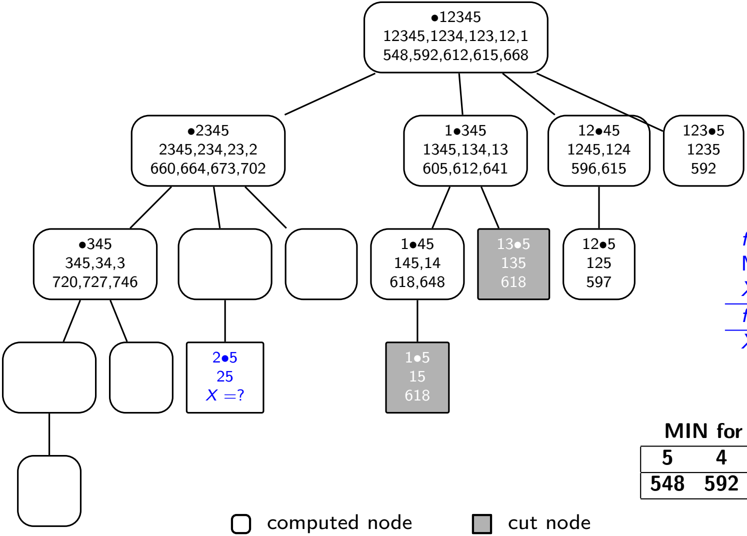
Branch-and-bound algorithm



MIN for # of variables

5	4	3	2	1
548	592	597	615	668

Branch-and-bound algorithm



$f(2345) = 660$
$\text{MIN}_2 = 615$
$X > f(2345)$

$f(2345) > \text{MIN}_2$

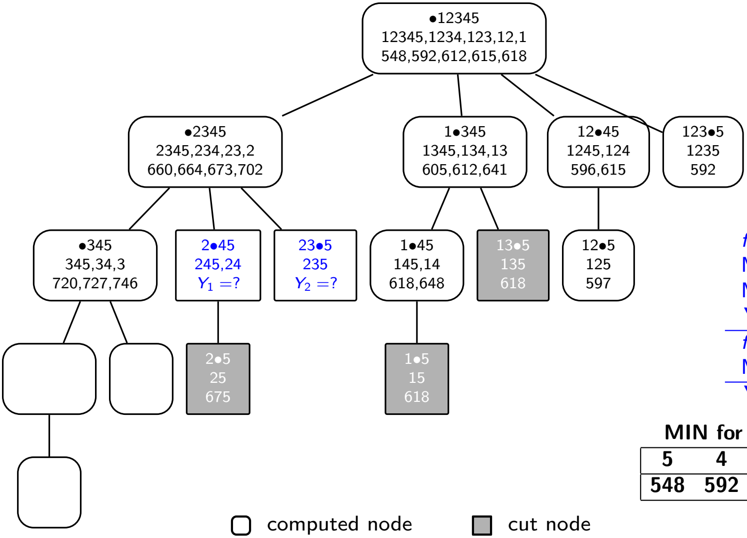
$X > \text{MIN}_2$

MIN for # of variables

5	4	3	2	1
548	**592**	**597**	**615**	**668**

▢ computed node ■ cut node

Branch-and-bound algorithm



$f(2345) = 660$
$\mathrm{MIN}_2 = 615$
$\mathrm{MIN}_3 = 597$

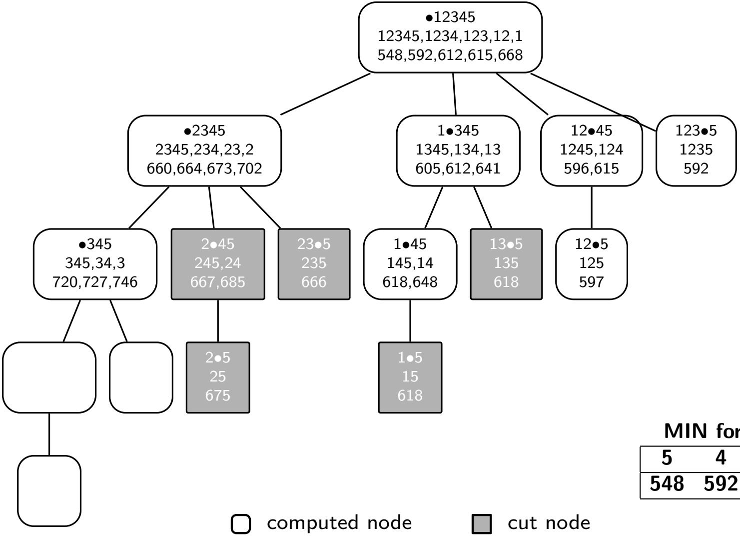
$$\frac{Y_1, Y_2 > f(2345)}{\begin{array}{c} f(2345) > \mathrm{MIN}_2 \\ \mathrm{MIN}_2 > \mathrm{MIN}_3 \end{array}}$$

$$\overline{Y_1, Y_2 > \mathrm{MIN}_3}$$

MIN for # of variables

5	4	3	2	1
548	592	597	615	668

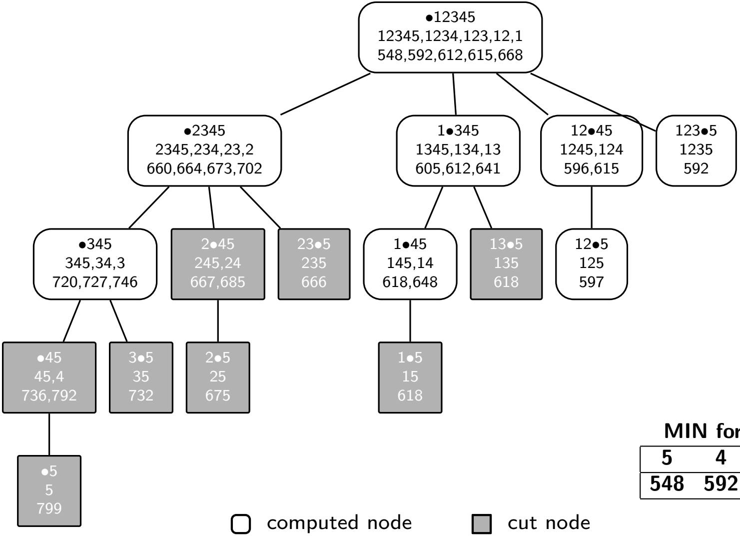
Branch-and-bound algorithm



MIN for # of variables

5	4	3	2	1
548	592	597	615	668

Branch-and-bound algorithm



MIN for # of variables

5	4	3	2	1
548	592	597	615	668

Execution times in seconds of the LBA and BBA

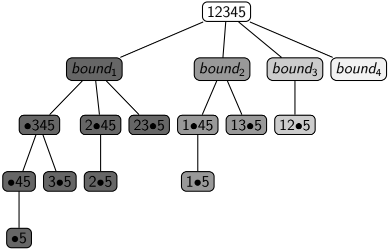
	Generating all models			Cutting sub-trees			
	Time			LBA		BBA	
# of Var	LBA	BBA	LBA / BBA	Time	Nodes	Time	Nodes
15	2.17	0.15	14	0.17	1958	0.02	969
16	4.55	0.28	16	0.15	1556	0.03	760
17	9.49	0.54	18	0.58	5932	0.08	2966
18	19.93	1.07	19	0.94	8870	0.13	4383
19	43.64	2.17	20	2.55	23316	0.29	11655
20	88.73	4.48	20	11.35	108806	1.07	54403
21	184.51	8.65	21	7.77	64348	0.80	32174
22	387.50	17.02	23	7.13	49994	0.72	24988
23	811.99	34.16	24	8.18	57060	0.81	28511
24	1739.70	68.20	26	62.76	454332	5.74	227159
25	3617.78	136.00	27	53.66	358008	5.60	178997
25	1h	2 min		1 min		6 sec	

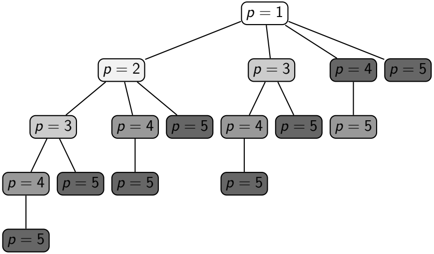
$$C_{\text{LBA}}(n)/C_{\text{BBA}}(n) \approx 0.0058(2n^3 + 9n^2 - 5n - 6) \equiv O(n^3)$$

Content

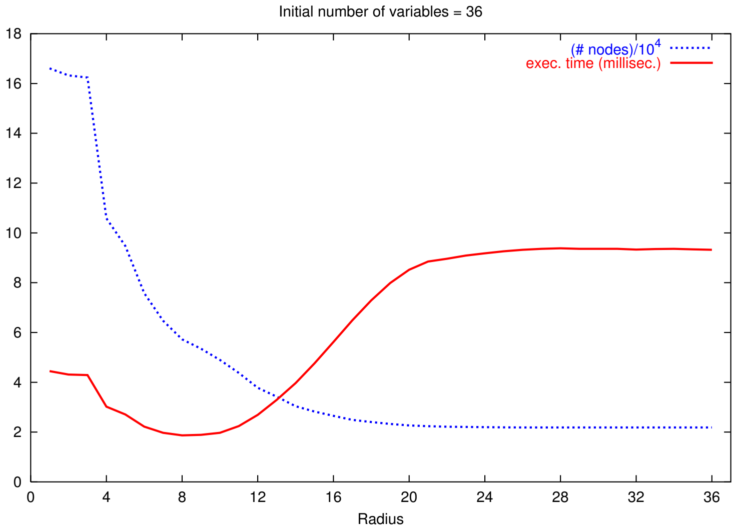
Variable preordering

- Pre-order variables into the root node such that $bound_1 \geq bound_2 \geq bound_3 \geq bound_4$.
- preorder variables in all nodes within a given radius p.

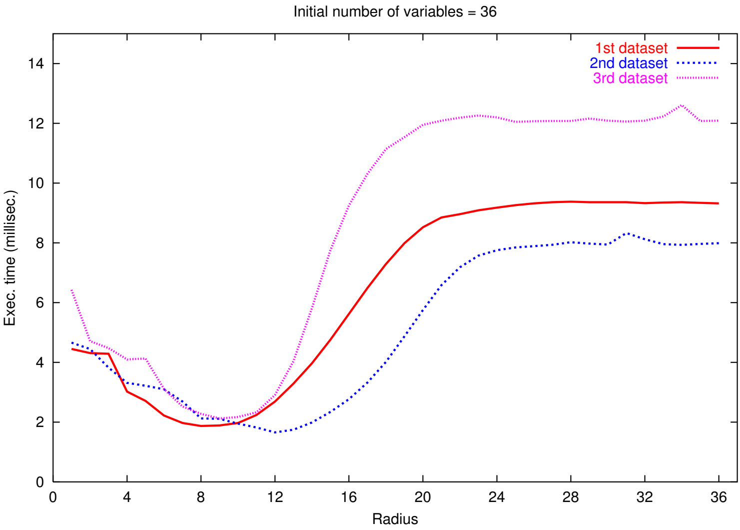




Execution times of PBBA using various radious levels



Execution times of PBBA using various radious levels



Heuristic branch-and-bound (HBBA)

- Finds good, but not necessarily optimal, subset models.
- VERY FAST!
- Tolerance parameter τ.
- Relative residual error (RRE):

$$RRE(W_j) = \frac{RSS(W_j) - RSS(V_j^*)}{RSS(W_j)}$$

where W_j is a subset model of size j, V_j^* is the optimal subset model of the same size.

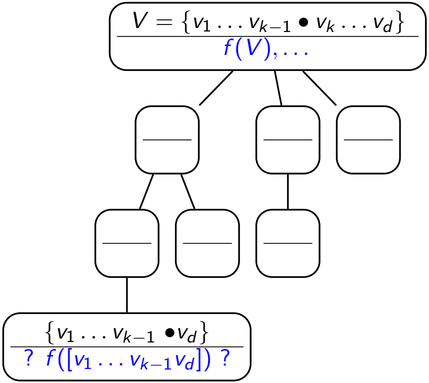
- It has been shown that

$$\boxed{RRE(W_j) < \tau,}$$

where W_j has been computed by HBBA_τ.

Content

Heuristic Branch and bound

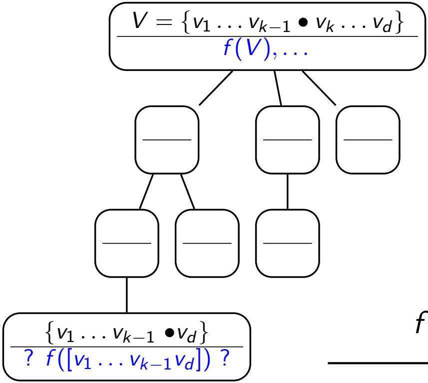


current MIN for each size

r_1	$\ldots$	r_k	$\ldots$	r_n

$$f(V) < r_k$$

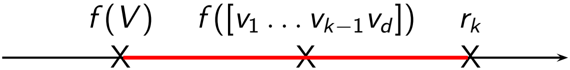
Heuristic Branch and bound



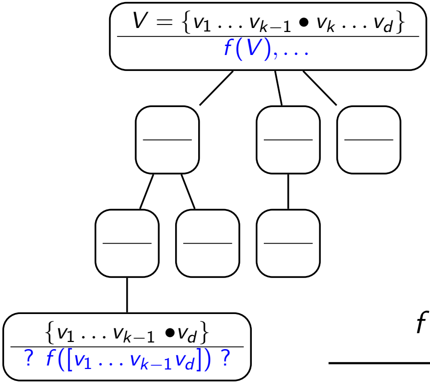
current MIN for each size

r_1	$\ldots$	r_k	$\ldots$	r_n

$$f(V) < r_k$$



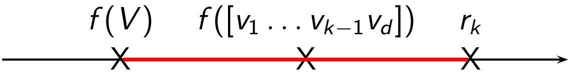
Heuristic Branch and bound



current MIN for each size

r_1	$\ldots$	r_k	$\ldots$	r_n

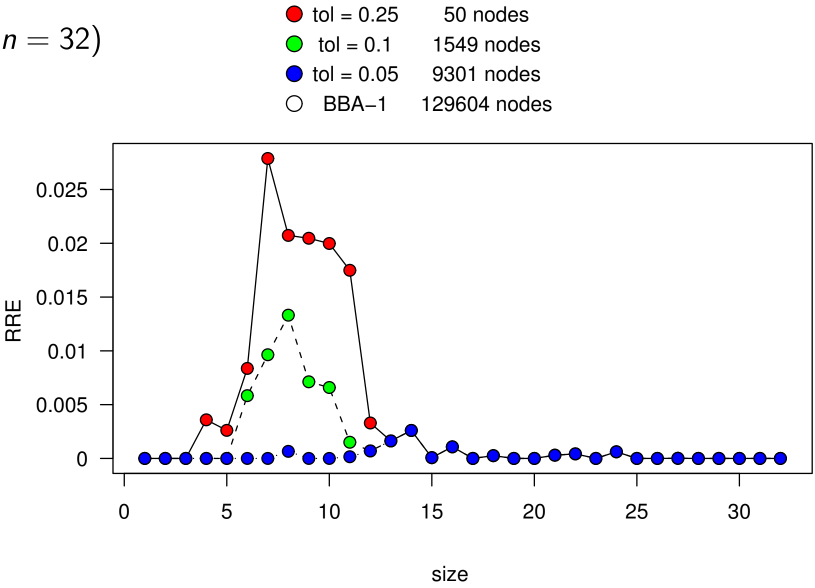
$$f(V) < r_k$$



$$\frac{r_k - f(V)}{f(V)} \leq \tau \Leftrightarrow (1+\tau) f(V) \geq r_k$$

HBBA

$(n = 32)$



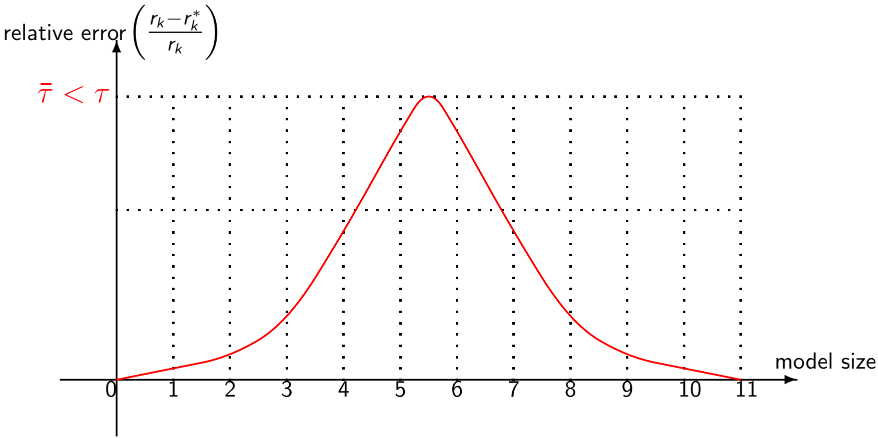
tol = 0.25 50 nodes
tol = 0.1 1549 nodes
tol = 0.05 9301 nodes
BBA−1 129604 nodes
0.025
0.02
0.015
0.01
0.005
0
RRE
0
5
10
15
20
25
30
size

Exec. times in sec. for various versions of the BBA

	Exhaustive methods		Heuristics with $\tau = 0.1$		Heuristics with $\tau = 0.25$	
# of Var	BBA	BBA–1	HBBA	HBBA–1	HBBA	HBBA–1
15	0.02	0.01	0.01	0.01	0.009	0.003
20	1.14	0.05	0.48	0.02	0.16	0.009
25	5.60	0.32	1.78	0.05	0.20	0.010
30	22.54	1.09	3.46	0.04	1.46	0.005
35	171.64	3.01	42.47	0.32	4.77	0.040
40	10049.32	45.09	168.17	1.31	12.27	0.030
41	3197.91	63.22	80.94	1.12	0.89	0.070
42	28176.72	76.09	4949.46	3.15	255.20	0.090
43	31567.22	289.52	1353.42	4.50	115.70	0.093
44	3806.57	89.07	266.99	2.78	11.93	0.086
45	47342.35	149.80	2105.87	1.74	17.25	0.042
45	13 h	2.5 min	36 min	2 sec	17 sec	0.042 sec

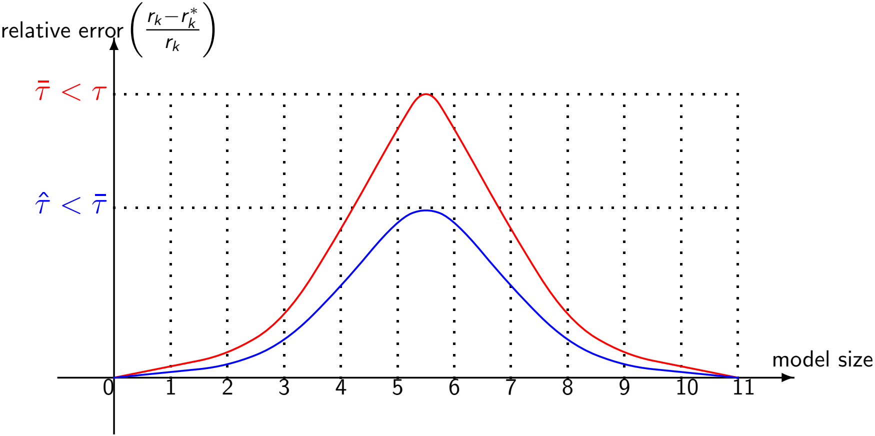
Content

Size Heuristic Branch and bound



Split the tolerance τ (scalar) in $(\tau_1, \ldots, \tau_n)$ (vector).

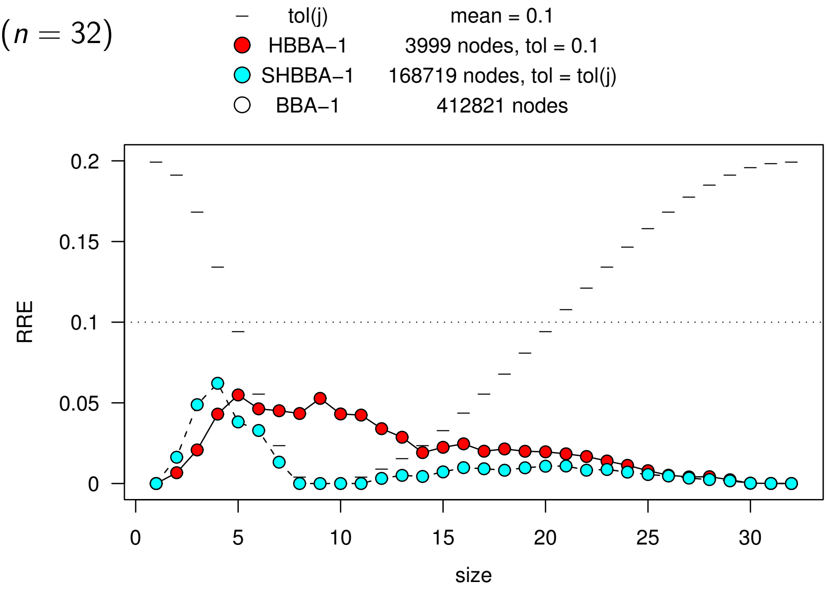
Size Heuristic Branch and bound



Split the tolerance τ (scalar) in $(\tau_1, \ldots, \tau_n)$ (vector).

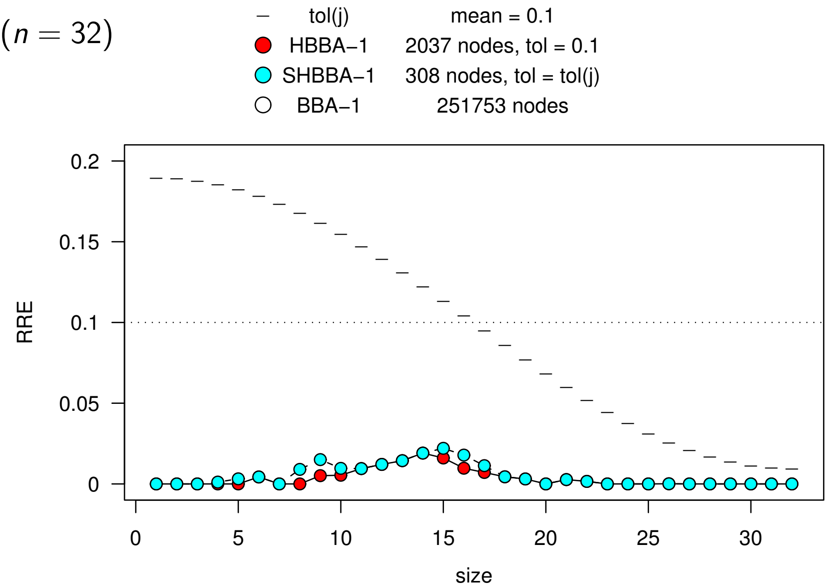
Size HBBA

$(n = 32)$



Size HBBA

$(n = 32)$



Content

Least squares estimation

- **Ordinary Linear Model (OLM)**:
  - $y = A\beta + \varepsilon, \qquad \varepsilon \sim (0, \sigma^2 I)$

- **Ordinary Least Squares (OLS)**:
  - $\widehat{\beta} = \underset{\beta}{\operatorname{argmin}} \|y - A\beta\|^2 \quad \equiv \quad (A^T A)^{-1} A^T y$
  - $\mathrm{RSS} = \|y - A\widehat{\beta}\|^2$

- **Non-Negative Least Squares (NNLS)**:
  - $\widetilde{\beta} = \underset{\beta}{\operatorname{argmin}} \|y - A\beta\|^2 \quad \text{subject to} \quad \beta \geq 0$
  - $\text{NN-RSS} = \|y - A\widetilde{\beta}\|^2$

Least squares estimation

- **Ordinary Linear Model (OLM)**:
  - $y = A\beta + \varepsilon, \qquad \varepsilon \sim (0, \sigma^2 I)$
- **Ordinary Least Squares (OLS)**:
  - $\widehat{\beta} = \underset{\beta}{\operatorname{argmin}} \|y - A\beta\|^2 \quad \equiv \quad (A^T A)^{-1} A^T y$
  - $\text{RSS} = \|y - A\widehat{\beta}\|^2$
- **Non-Negative Least Squares (NNLS)**:
  - $\widetilde{\beta} = \underset{\beta}{\operatorname{argmin}} \|y - A\beta\|^2 \quad \text{subject to} \quad \beta \geq 0$
  - $\text{NN-RSS} = \|y - A\widetilde{\beta}\|^2$

Non-Negative Model Selection (NNMS)

- Given

$$\text{a set of } n \text{ variables } V = \{1, 2, \cdots, n\}$$

- For all $p = 1, \cdots, n$ find

$$W_p = \underset{W \subseteq V}{\text{argmin}} \ \text{NN-RSS}(W) \quad \text{subject to} \quad |W| = p$$

- Complexity: solve $2^n - 1$ NNLS problems

NNMS: naive approach

procedure naiveNNMS(n, V) $\{ V = \{1, \cdots, n\} \}$

$W_p \leftarrow \emptyset$; $r_p \leftarrow +\infty$, for $p = 1, \cdots, n$

for $p = 1, \cdots, n$ **do**

for all $W \subseteq V$ with $|W| = p$ **do**

solve NNLS(W)

if (NN-RSS(W) $< r_p$) **then**

$W_p \leftarrow W$; $r_p \leftarrow$ NN-RSS(W)

end procedure

Non-Negative Least Squares (NNLS)

- Quadratic programming (Simplex)
- Alternative

Compute unconstraint $\widehat{\beta}$
if ($\widehat{\beta} \geq 0$) **then**
  $\widehat{\beta}_c \equiv \widehat{\beta}$
else
  { $\widehat{\beta}_c$ *boundary in* $([0, \infty))^n$ }
  Enumerate all unconstraint submodels
end if

- Example: n=3; 123, 12, 13, 23, 1, 2, 3

Non-Negative Least Squares (NNLS)

- Quadratic programming (Simplex)
- Alternative (Waterman 1974; Armstrong et al. 1976; Cutler 1993)

Compute unconstraint $\widehat{\beta}$
if ($\widehat{\beta} \geq 0$) **then**
  $\widehat{\beta}_c \equiv \widehat{\beta}$
else
  { $\widehat{\beta}_c$ *boundary in* $([0, \infty))^n$ }
  Enumerate all unconstraint submodels
end if

- Example: n=3; 123, 12, 13, 23, 1, 2, 3

Non-Negative Least Squares (NNLS)

- Quadratic programming (Simplex)
- Alternative

  Compute unconstraint $\widehat{\beta}$
  if ($\widehat{\beta} \geq 0$) **then**
  $\widehat{\beta}_c \equiv \widehat{\beta}$
  else
  { $\widehat{\beta}_c$ *boundary in* $([0, \infty))^n$ }
  Enumerate all unconstraint submodels
  end if

- Example: n=3; 123, 12, 13, 23, 1, 2, 3

NNLS: alternative solution

LEMMA 1.

The NNLS problem is equivalent to

$$S^* = \underset{S \subseteq V}{argmin}\ RSS(S) \quad subject\ to \quad \beta_S \geq 0,$$

where the $\beta_S \in \Re^{|S|}$ *is the coefficient vector corresponding to the variables selected in* S.

NNMS: new formulation

LEMMA 2.

The problem of regression subset selection with non-negative coefficients constraints is equivalent to

$$S_p^* = \underset{S \subseteq V}{\text{argmin}}\ RSS(S) \quad \text{subject to}$$
$$\beta_S \geq 0 \quad \text{and} \quad |S| \leq p, \qquad \text{for} \quad p = 1, \cdots, n.$$

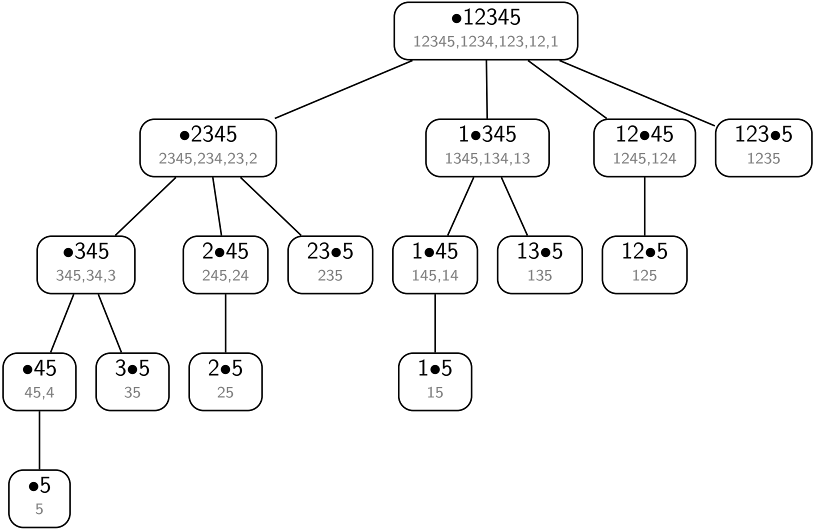
For all $\quad p = 1, \cdots, n \quad$ find

$$W_p = \underset{W \subseteq V}{\text{argmin}}\ \text{NN-RSS}(W) \quad \text{subject to} \quad |W| = p$$

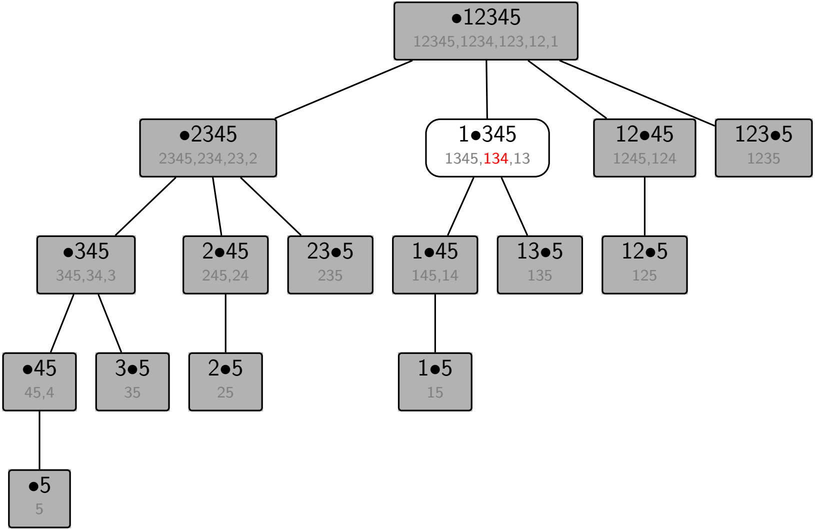
NNMS: enumerative algorithm

procedure NNMS(n, V) $\{\ V = \{1, \cdots, n\}\ \}$
    $W_p \leftarrow \emptyset$; $r_p \leftarrow +\infty$, for $p = 1, \cdots, n$
    for $p = 1, \cdots, n$ **do**
        for all $S \subseteq V$ with $|S| = p$ **do**
            solve OLS(S)
            if $(\widehat{\beta}_S \geq 0)$ **then**
                for $j = p, \cdots, n$ **do**
                    if $(\mathrm{RSS}(S) < r_j)$ **then**
                        $W_j \leftarrow S$; $r_j \leftarrow \mathrm{RSS}(S)$
end procedure

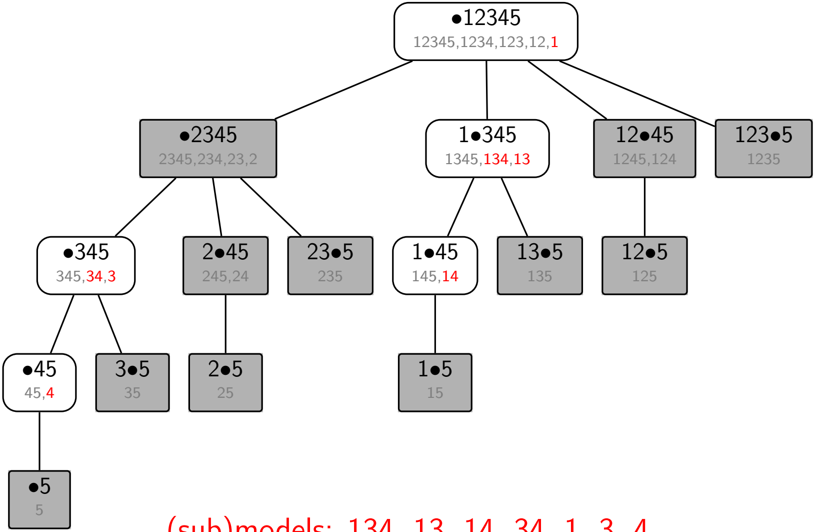
Enumeration scheme: regression tree ($n = 5$)



134 – unconstraint solution



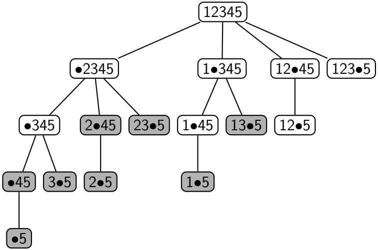
134 – constraint solution



(sub)models: 134, 13, 14, 34, 1, 3, 4

Branch and bound algorithms

- Exhaustive algorithm.
- Exploit statistical information (RSS) to cut subtrees.



Generated nodes & times (seconds)

n	naiveNNMS		NNMS		BB-NNMS		BB-NNMS§	
	nodes	time	nodes	time	nodes	time	nodes	time
15	16'384	25	16'384	0.06	489	0.004	800	0.004
16	32'768	57	32'768	0.12	885	0.004	891	0.004
17	65'536	139	65'536	0.23	1'120	0.005	1'667	0.008
18	131'072	331	131'072	0.48	1'270	0.008	3'254	0.015
19	262'144	723	262'144	0.95	1'867	0.008	3'686	0.017
20	524'288	1'678	524'288	1.93	8'830	0.040	6'983	0.033
21	1'048'576	3'775	1'048'576	3.90	15'451	0.068	13'289	0.062
22	2'097'152	8'876	2'097'152	7.90	8'128	0.036	14'522	0.070
23	4'194'304	19'707	4'194'304	15.80	17'292	0.084	26'706	0.128
24	8'388'608	43'818	8'388'608	32.30	56'894	0.276	54'159	0.258
25	***	***	16'777'216	64.10	35'798	0.176	61'549	0.301

*** – canceled after 24 hours; §– mean values over 1000 runs on different data.

Content

lmSubsets

- INPUT:
  - an object of class `'lm'` or
  - a formula or
  - a matrix.
- OUTPUT:
  - an object of class `'lmSubsets'`;
  - `$rss`: residual sum of squares;
  - `$which`: selected variables.
- FUNCTIONS: `print`, `plot`, `summary`

lmSubsets

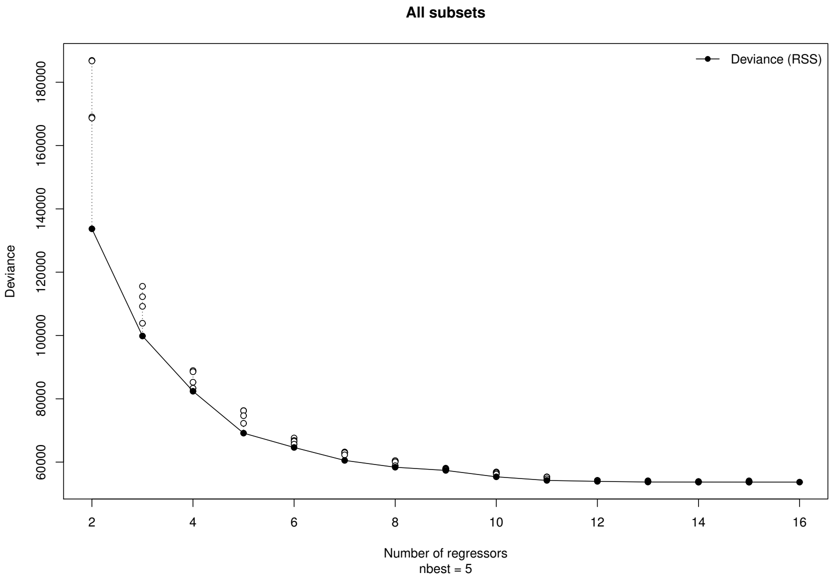
```
lmSubsets              package:lmSubsets              R Documentation

All-Subsets Regression

Description:

     All-subsets regression for ordinary linear models.

Usage:

     lmSubsets(formula, data, subset, weights, na.action,
     model = TRUE, x = FALSE, y = FALSE, contrasts = NULL, offset, ...)

     lmSubsets.fit(x, y, weights = NULL, offset = NULL,
     include = NULL, exclude = NULL, nmin = NULL, nmax = NULL,
     tolerance = 0, pradius = NULL, nbest = 1, ..., .algo = "hpbba")
```

Example:

```
library(lmSubsets)

## load data
data("AirPollution", package = "lmSubsets")

## canonical example: fit all subsets
all.AirPoll <- lmSubsets(mortality ~ ., data = AirPollution, nbest = 5)

## visualize RSS
plot(all.AirPoll)

## summarize
summary(all.AirPoll)

## plot summary
plot(summary(all.AirPoll))
```

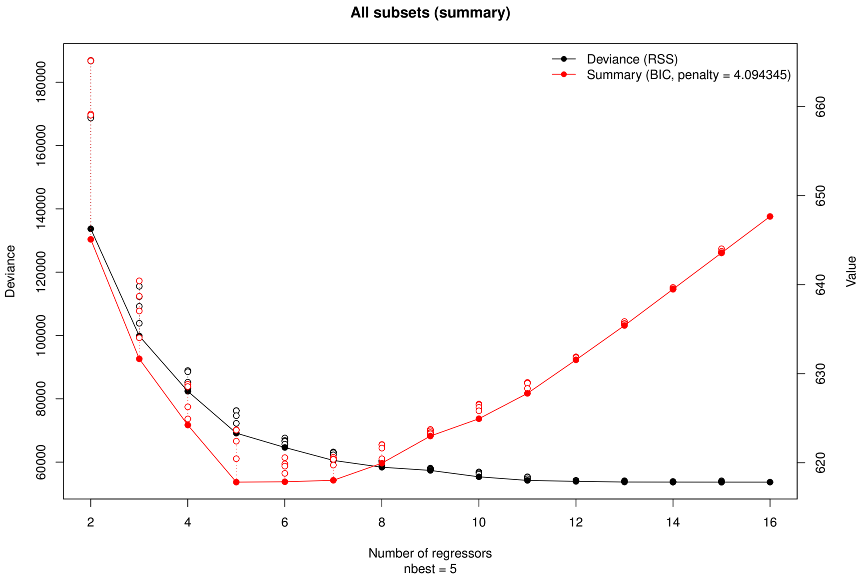
```
> print(all.AirPoll)
[...]

Model fit (deviance):
  best x size
     2         3          4          5          6          7          8
  1. 133694.54  99841.07  82388.53  69154.11  64633.79  60538.76  58385.72
  2. 168695.53 103859.31  83335.14  72250.33  65659.86  62288.70  58870.48
  3. 169041.38 109202.60  85241.98  74666.42  66554.64  62953.77  60057.48
  4. 186715.91 112259.15  88542.69  76230.34  66837.27  63007.12  60422.51
  5. 186896.19 115541.19  88919.66  76276.41  67621.51  63205.56  60465.10
     9         10         11         12         13         14         15
  1.  57379.21  55358.05  54221.58  53921.82  53712.66  53696.00  53683.31
  2.  57617.43  56185.55  54718.93  54146.37  53874.74  53696.65  53690.20
  3.  57748.66  56550.95  55260.67  54186.59  53900.78  53709.86  53695.48
  4.  57948.25  56818.31  55298.82  54217.60  53917.84  53846.53  53845.54
  5.  58093.85  56896.70  55343.71  54219.26  54112.97  53872.20  54097.09
     16
  1.  53680.02
  2.
  3.
  4.
  5.
```

```
> plot(all.AirPoll)
```



```
> summary(all.AirPoll)
Call:
lmSubsets(formula = mortality ~ ., data = AirPollution, nbest = 3)
Summary:
  Value: BIC (penalty = 4.094345)

Model fit:
  best x size
    DEVIANCE
    Summary
      2           3           4           5            6            7
  1st 133694.5375  99841.0707  82388.5289  69154.1114   64633.7871   60538.7565
         645.0938    631.6694    624.2358    617.8236*    617.8619     618.0290
  2nd 168695.5325 103859.3092  83335.1406  72250.3324   65659.8646   62288.6991
         659.0460    634.0369    624.9212    620.4516     618.8069*    619.7388
  3rd 169041.3808 109202.5995  85241.9793  74666.4172   66554.6389   62953.7731
         659.1689    637.0469    626.2786    622.4252     619.6191*    620.3761
      8           9           10          11          12          13
  1st  58385.7150  57379.2090  55358.0499  54221.5787  53921.8188  53712.6644
         619.9506    623.0016    624.9444    627.7941    631.5559    635.4170
  2nd  58870.4775  57617.4285  56185.5482  54718.9259  54146.3720  53874.7417
         620.4468    623.2502    625.8346    628.3420    631.8052    635.5978
  3rd  60057.4817  57748.6552  56550.9524  55260.6675  54186.5889  53900.7791
         621.6445    623.3867    626.2236    628.9331    631.8498    635.6268
      14          15          16
  1st  53696.0048  53683.3135  53680.0215
         639.4928    643.5729    647.6636
  2nd  53696.6464  53690.1979
         639.4935    643.5806
  3rd  53709.8603  53695.4814
         639.5082    643.5865
```

```
> plot(summary(all.AirPoll))
```



lmSelect: problem reformulation

- Ordinary linear model with n variables:

$$F = \{1, 2, \ldots, n\}$$

- Problem statement:

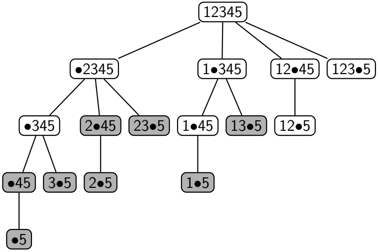
  find the *best* subset model S^*, $S^* \subset F$.

- *Best*? Smallest ~~RSS~~ AIC.

- Computational cost: $2^n - 1$ possible models

Branch and bound algorithm

- Exhaustive algorithm.
- Exploit statistical information (~~RSS~~ AIC) to cut subtrees.



lmSelect

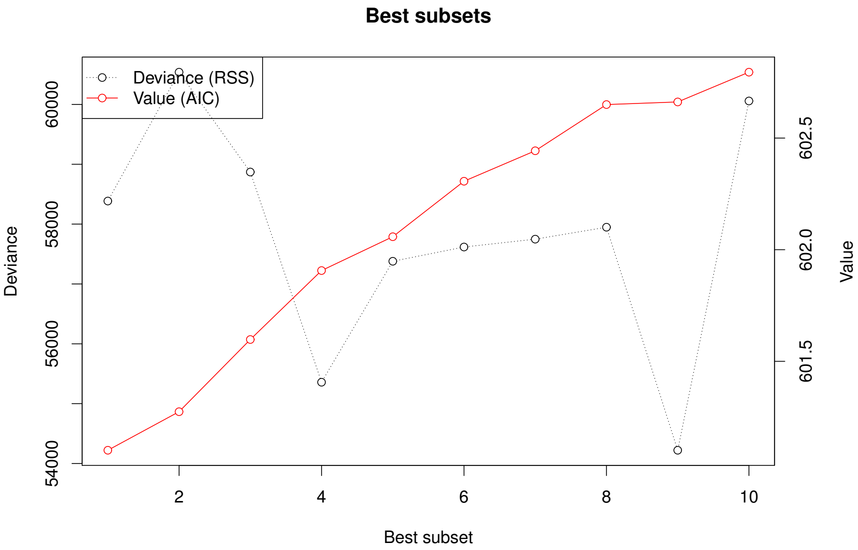
```
lmSelect               package:lmSubsets               R Documentation

All-Subsets Regression

Description:  Best-subsets regression for ordinary linear models.

Usage:

     lmSelect(formula, data, subset, weights, na.action,
     model = TRUE, x = FALSE, y = FALSE, contrasts = NULL, offset, ...)

     lmSelect.fit(x, y, weights = NULL, offset = NULL,
     include = NULL, exclude = NULL, penalty = "BIC", tolerance = 0,
     pradius = NULL, nbest = 1, ..., .algo = "hpbba")

     lmSubsets.select(object, penalty = "BIC", ...)
```

Examples:

```
library(lmSubsets)

## load data
data("AirPollution", package = "lmSubsets")

## canonical example: fit best subsets
best.AirPoll <- lmSelect(mortality ~ ., data = AirPollution, nbest = 10)

## equivalent to:
## Not run:

all.AirPoll <- lmSubsets(mortality ~ ., data = AirPollution, nbest = 10)
best.AirPoll <- lmSelect(all.AirPoll)
## End(Not run)


## visualize RSS
plot(best.AirPoll)

## summarize
summary(best.AirPoll)
```

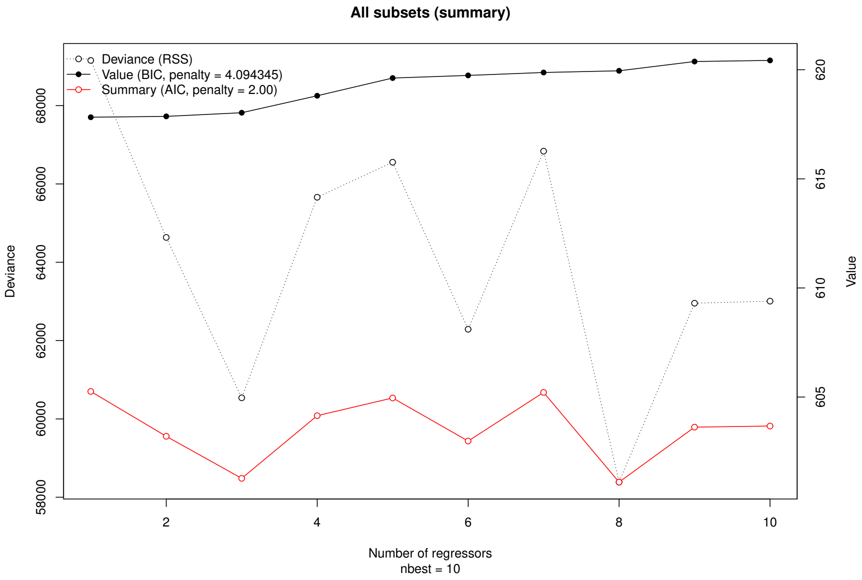
```
> print(best.AirPoll)
[...]
Model fit:
          1st        2nd        3rd        4th        5th        6th
  df          6.0000     7.0000     8.0000     7.0000     7.0000     8.0000
  Deviance 69154.1114 64633.7871 60538.7565 65659.8646 66554.6389 62288.6991
  VALUE      617.8236*  617.8619   618.0290   618.8069   619.6191   619.7388
          7th        8th         9th        10th
  df          7.0000     9.0000      8.0000     8.0000
  Deviance 66837.2687 58385.7150* 62953.7731 63007.1215
  VALUE      619.8733   619.9506    620.3761   620.4269
```

```
> plot(best.AirPoll)
```



```
> summary(best.AirPoll)

Call:
  lmSelect(formula = mortality ~ ., data = AirPollution, nbest = 10)

Summary:
  Value: AIC (penalty = 2.00)

Model fit:
           1st          2nd         3rd         4th         5th         6th
  df          6.0000       7.0000      8.0000      7.0000      7.0000      8.0000
  Deviance 69154.1114 64633.7871 60538.7565 65659.8646 66554.6389 62288.6991
  VALUE      617.8236*    617.8619    618.0290    618.8069    619.6191    619.7388
  Summary    605.2575     603.2015    601.2743    604.1465    604.9587    602.9841
           7th         8th          9th         10th
  df          7.0000      9.0000       8.0000      8.0000
  Deviance 66837.2687 58385.7150* 62953.7731 63007.1215
  VALUE      619.8733    619.9506    620.3761    620.4269
  Summary    605.2129    601.1015*   603.6213    603.6721
```

```
> plot(summary(best.AirPoll))
```



lmSubsets vs lmSelect

- lmSubsets finds the best models of each possible size.
- lmSelect finds the best model (by default, in terms of AIC).
- Both methods are equivalent (in terms of AIC).
- lmSelect is faster while lmSubsets is more informative.

```
> all.AirPoll <- lmSubsets(mortality ~ ., data = AirPollution, nbest = 1)
> all.AirPoll$.nodes
[1] 172
> best.AirPoll <- lmSelect(mortality ~ ., data = AirPollution, nbest = 1)
> best.AirPoll$.nodes
[1] 112
```

leaps vs lmSubsets

Leaps & BBA: Execution times in seconds for datasets of different sizes, without and with variable preordering.

# Var.	36	37	38	39	40	41	42	43	44	45	46	47	48
Leaps	8	29	44	30	203	57	108	319	135	316	685	2697	6023
BBA	2	5	12	8	35	14	9	55	27	37	97	380	1722
Leaps-1	3	16	28	9	82	33	22	203	79	86	306	1326	1910
BBA-1	1	4	13	2	20	11	4	47	18	15	51	216	529

Genetic algorithms for variables selection in **R**

- gaselect: A Genetic Algorithm (GA) for Variable Selection from High-Dimensional Data
- kofnGA: A Genetic Algorithm for Fixed-Size Subset Selection
- glmulti: Model selection and multimodel inference made easy

Content

Settings

- gaSelect vs. lmSubsets.
- Average over $T = 100$ artificial datasets.
- "True" submodel comprises $n/3$ variables.
- 4 core Intel(R) Core(TM) i7-3612QM CPU @ 2.10GHz, running under Ubuntu 15.10.

I. Problem size

- gaSelect():

T	n	mean time (sec.)	min.	max.
100	205	3333	2114	5272

- lmSelect(): $(\tau = 0)$

T	n	mean time	min	max	average AIC
100	125	4841	1455	54883	1371

- gaSelect():

T	n	mean time	min	max	average AIC
100	125	323	237	454	1408 (0.027)

II. Relative errors of GA solutions

	Time (sec.)		AIC		rel. AIC	
n	GA	$HBBA_0$	GA	$HBBA_0$	GA	$HBBA_0$
50	14	0.06	557	549	0.015	0
70	61	4	812	771	0.05	0
80	80	56	917	881	0.04	0
90	91	390	1024	992	0.03	0
n	GA	$HBBA_{0.04}$	GA	$HBBA_{0.04}$	GA	$HBBA_{0.04}$
90	91	69	1024	992	0.03	0.0001

Note: Average values over $T = 100$ runs (datasets).

III. gaSelect vs. lmSubsets (tolerance $\tau = 0.04$)

# of Var	Time (sec.) GA	Time (sec.) HBBA	Submodel size GA	Submodel size HBBA	AIC GA	AIC HBBA	relative AIC
50	14	0.05	25	23	553	549	0.007
60	22	0.34	30	28	668	662	0.009
70	61	2.47	61	33	811	771	0.05
80	80	11.6	63	38	917	881	0.04
90	91	69	65	43	1024	992	0.03
100	93	416	53	48	1123	1103	0.02
110	118	1251	59	52	1232	1210	0.02

lmSubsets():

- faster (less execution time) – for small data sets;
- more parsimonious models (less selected variables);
- better models (smaller AIC);
- *controlled relative error (bounded by τ).*

Conclusions

- Regression tree based algorithms for model selection.
- Branch-and-bound strategy prunes non-optimal subtrees.
- Heuristic strategies find solution close to the optimum.
- Novel approach to non-negative subset regression selection.
  - Enumerate unconstrained subproblems.
- An R package for regression subset selection.
- Versatile tool that allows subset model investigation.
- The proposed alorithms makes the selection procedure feasible for larger-scale models.

References

C. Gatu and E. J. Kontoghiorghes.
Parallel algorithms for computing all possible subset regression models using the QR decomposition.
Parallel Computing, 29(2003), pp. 505–521.

C. Gatu and E. J. Kontoghiorghes.
Branch-and-bound algorithms for computing the best subset regression models.
Journal of Computational and Graphical Statistics, 15, 139–156, 2006.

M. Hofmann, C. Gatu and E. J. Kontoghiorghes.
Efficient algorithms for computing the best subset regression models for large-scale problems.
Computational Statistics and Data Analysis, 52(1), 16–29, 2007.

C. Gatu, P. Yanev and E. J. Kontoghiorghes.
A graph approach to generate all possible regression submodels.
Computational Statistics & Data Analysis, 52 (2007), pp. 799–815.

C. Gatu and E. J. Kontoghiorghes.
A fast algorithm for non-negativity model selection.
Statistics and Computing, 23 (2013), pp. 403–411.

M. Hofmann, C. Gatu, E. J. Kontoghiorghes and A. Zeileis.
`lmSubsets`: Variable Subset Selection in Linear Regressions.
R package version 0.1-0, `https://CRAN.R-project.org/package=lmSubsets`, 2018.

M. Hofmann, C. Gatu, E. J. Kontoghiorghes, A. Zeileis and A. Colubi.
`lmSubsets`: Efficient Computation of Best Subset Linear Regressions in R.
Journal of Statistical Software, 2016, (to be submitted).