Algorithms for Wireless Communications II

Andreas Burg

Institute for Integrated Signal Processing Systems, RWTH Aachen

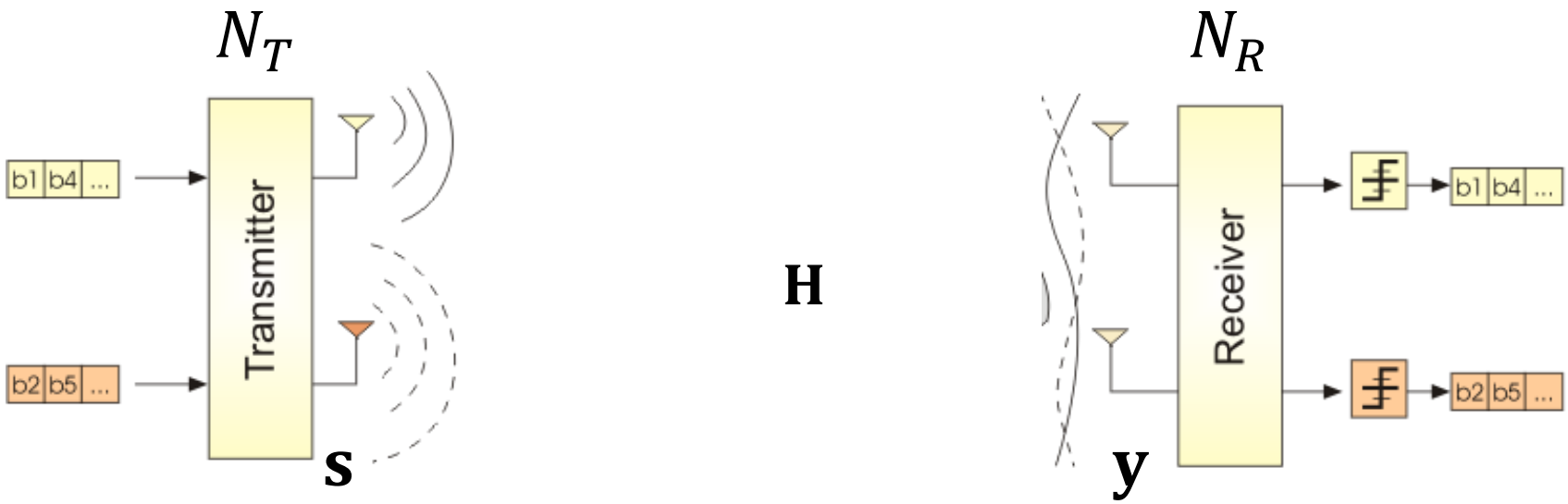
Telecommunications Circuits Laboratory, EPFL

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

MIMO ML Receivers

- Repetition: System Model and ML Detection
- Tree Search Algorithms for MIMO
  - *Sphere decoding*

System Model for Spatial Multiplexing



$$\mathbf{y} = \mathbf{H}\mathbf{s} + \mathbf{n}$$

- $\mathbf{s} = [s_1 \quad \cdots \quad s_{N_T}]^T$: Transmitted vector-symbol with $s_i \in \mathcal{O}$ ➔ $\boldsymbol{s} \in \mathcal{O}^{N_T}$
- $\mathcal{O}$: Set of constellation points
- $\mathbf{H} = [\mathbf{h_1} \quad \cdots \quad \mathbf{h}_{N_T}]$: Channel matrix
- $\mathbf{y} = [y_1 \quad \cdots \quad y_{N_T}]^T$: Received vector
- $\mathbf{n} = [n_1 \quad \cdots \quad n_{N_T}]^T$: Gaussian noise with i.i.d. entries $\mathbf{n} \sim \mathcal{CN}(0, \sigma^2)$

Optimum MIMO Detection

Task of the MIMO detector: estimate the transmitted symbol vector $\mathbf{s}$ based on the received vector $\mathbf{y}$

- Assume we know the channel matrix $\mathbf{H}$

- Best possible estimate if obtained using Maximum Likelihood criterion

$$\hat{\boldsymbol{s}}_{\mathrm{ML}} = \arg\max_{\hat{\boldsymbol{s}} \in \mathcal{O}^{N_T}} Pr(\boldsymbol{y}|\boldsymbol{s} = \hat{\boldsymbol{s}})$$

- The noise is i.i.d. Gaussian, hence

$$Pr(\boldsymbol{y}|\boldsymbol{s} = \hat{\boldsymbol{s}}) \propto \boldsymbol{e}^{-\|\boldsymbol{y}-\boldsymbol{H}\hat{\boldsymbol{s}}\|^2}$$

  - Remember: taking the logarithm yields a sufficient statistics

- The ML detection rule for MIMO spatial multiplexing becomes

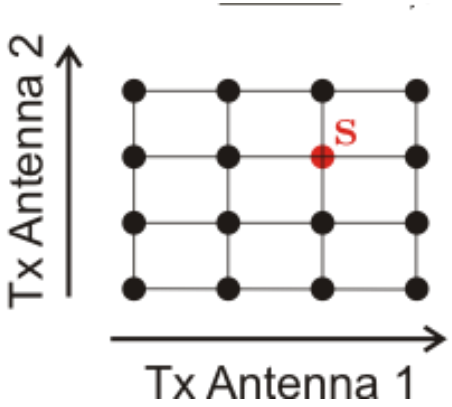
$$\hat{\boldsymbol{s}}_{\mathrm{ML}} = \arg\min_{\hat{\boldsymbol{s}} \in \mathcal{O}^{N_T}} \|\boldsymbol{y} - \boldsymbol{H}\hat{\boldsymbol{s}}\|^2$$

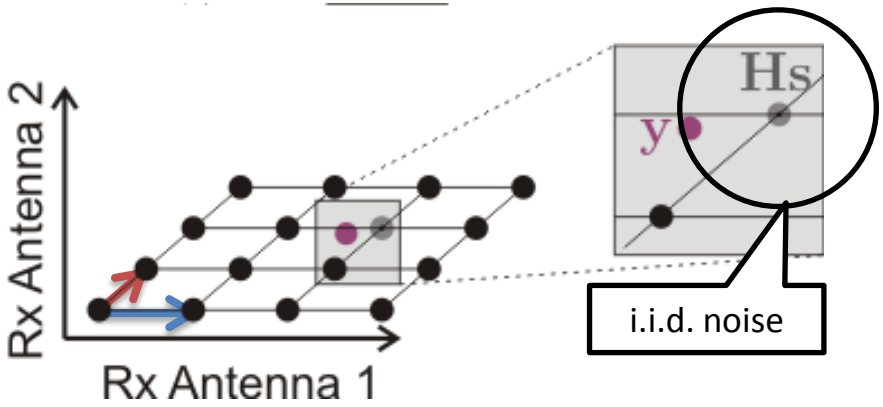
Graphical Interpretation

- Consider a system with $N_T = N_R = 2$ (2x2) with real-valued 4-PAM modulation on each antenna/stream

- Transmitter: streams lie along the orthogonal axis in the signal space
- Receiver: Signal components lie in the direction of the columns of $\mathbf{H}$
  ➔**Skewed constellation points**

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & 0.3 \\ 0 & 0.5 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} + \begin{bmatrix} n_1 \\ n_2 \end{bmatrix}$$





- The noise translates the received point $\mathbf{y}$ away from $\mathbf{Hs}$

ML: find the point among all $\mathbf{Hs}$ that lies closest to $\mathbf{y}$

Complexity of ML Detection

- Straightforward approach to solving the ML detection problem: Evaluate $\|\boldsymbol{y} - \boldsymbol{H}\hat{\boldsymbol{s}}\|^2$ for all possible candidate symbols and find the minimum

- Number of possible candidates
  - For M-QAM: M possible candidates for each spatial stream
  - For N_T spatial streams: M^{N_T} candidates

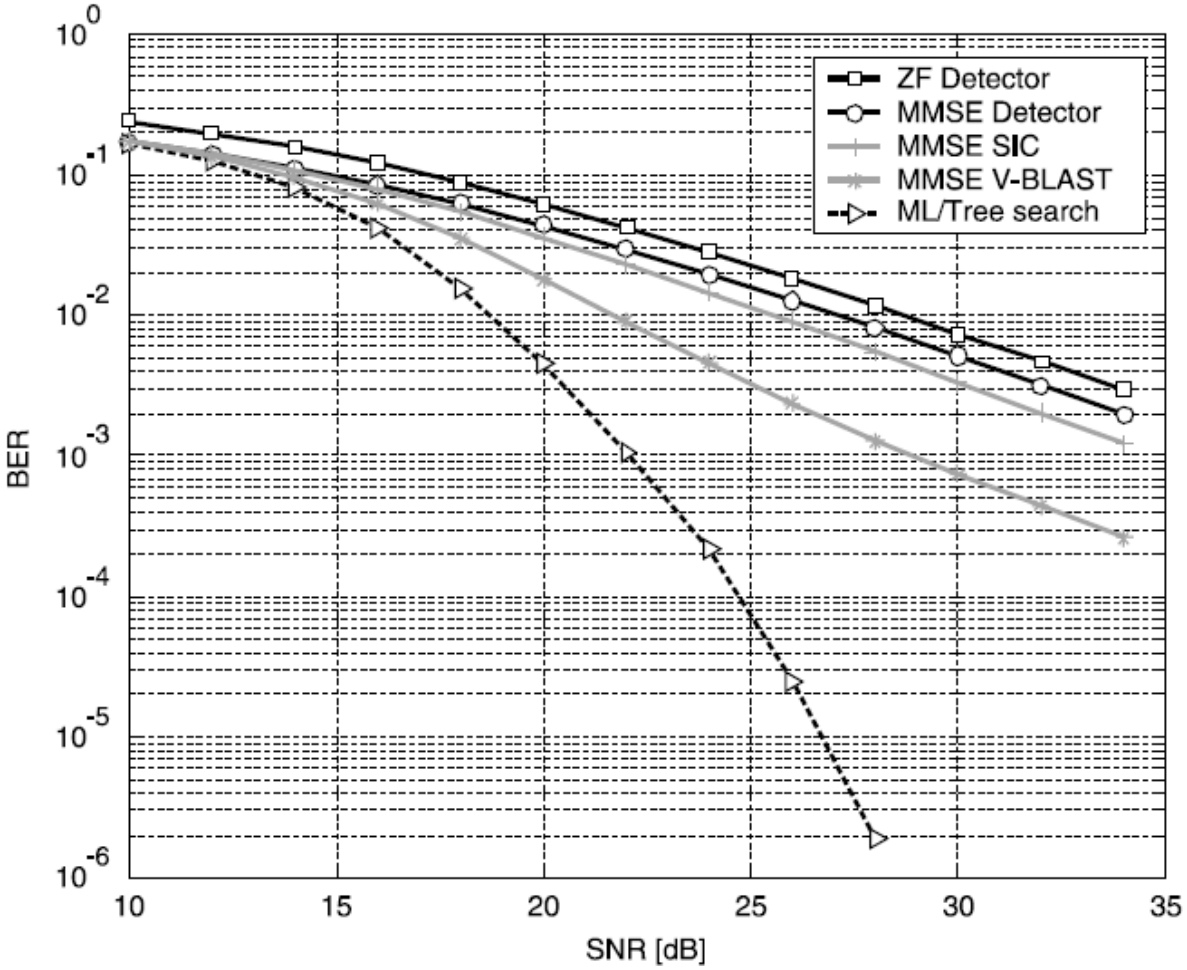
- Assume $M = 2^q$: transmit q bits on each stream
- Number of symbols: 2^{qN_T}
- Spectral efficiency: qN_T bits per vector-symbol (bits/s/Hz)

Number of candidate symbols to check grows exponentially with the spectral efficiency

- Example: 4 streams, 64-QAM
  - Spectral efficiency is 24 bits/s/Hz
  - For each received symbol ML detection checks 16 million candidates

Performance of Linear and SIC Detectors Compared to ML

Linear and SIC detectors do not achieve the full diversity



Triangularization

- Start from the original ML problem

$$\hat{\boldsymbol{s}}_{\mathrm{ML}} = \arg\min_{\mathbf{s}\in\mathcal{O}^{N_T}} \|\boldsymbol{y} - \mathbf{H}\mathbf{s}\|^2$$

- Apply the QR decomposition to obtain a triangular system

$$\mathbf{H} = \mathbf{Q}\mathbf{R}$$

$$\tilde{\mathbf{y}} = \mathbf{Q}^{\mathrm{H}}\mathbf{y} = \mathbf{R}\mathbf{s} + \tilde{\boldsymbol{n}}$$

- Since $\mathbf{Q}$ is unitary, the the properties of the noise do not change

- Consider the *triangularized* system instead of the original system
- Minimizes the *Euclidean Distance* (ED) $\mathrm{d}(\mathbf{s})$ between $\tilde{\mathbf{y}}$ and $\mathbf{R}\boldsymbol{s}$

$$\hat{\boldsymbol{s}}_{\mathrm{ML}} = \arg\min_{\mathbf{s}\in\mathcal{O}^{N_T}} \mathrm{d}(\mathbf{s}) \qquad \textbf{with} \qquad \mathrm{d}(\mathbf{s}) = \|\tilde{\mathbf{y}} - \mathbf{R}\boldsymbol{s}\|^2$$

Recursive Distance Computation

- Define the error-vector $\mathbf{e} = \tilde{\mathbf{y}} - \mathbf{R}\boldsymbol{s}$

- The ED can be written as a sum of *Distance Increments* $|e_i|^2$

$$d(\mathbf{s}) = \|\mathbf{e}(\hat{\mathbf{s}})\|^2 = \sum_{i=N_T}^{1} |e_i|^2$$

that can be computed recursively with *partial Euclidean distances* (PEDs) $d_i(\mathbf{s})$ as intermediate results and $d(\mathbf{s}) = d_1(\mathbf{s})$

$$d_i(\mathbf{s}) = d_{i+1}(\mathbf{s}) + |e_i|^2$$

- Initialize with $d_{N_T+1}(\mathbf{s}) = 0$

Computing Distance Increments

- In the triangular system

$$\mathbf{e} = \begin{bmatrix} e_1 \\ \vdots \\ \vdots \\ e_{N_T} \end{bmatrix} = \begin{bmatrix} \tilde{y}_1 \\ \vdots \\ \vdots \\ \tilde{y}_{N_T} \end{bmatrix} - \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1N_T} \\ 0 & r_{22} & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & r_{N_T N_T} \end{bmatrix} \begin{bmatrix} s_1 \\ \vdots \\ \vdots \\ s_{N_T} \end{bmatrix}$$

each distance increment $|e_i|^2$ depends only on $\mathbf{s}^{(i)} = [s_i \quad \cdots \quad s_{N_T}]$

$$|e_i|^2 = \left| \tilde{y}_i - \sum_{j=i}^{N_T} r_{ij} s_j \right|^2$$

and therefore, also the PEDs $d_i(\mathbf{s}^{(i)})$ depend only on $\mathbf{s}^{(i)} = [s_i \quad \cdots \quad s_{N_T}]$

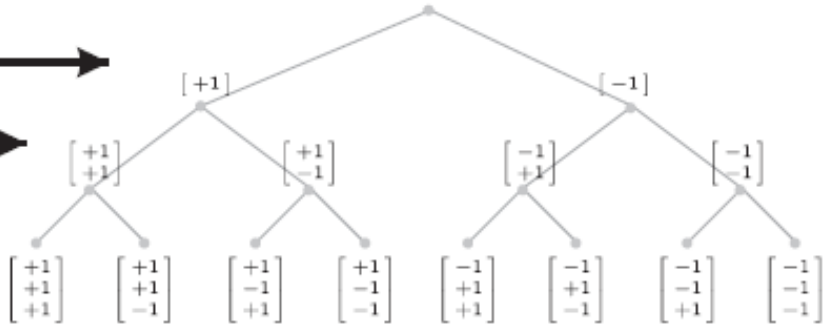
- $\mathbf{s}^{(i)} = [s_i \quad \cdots \quad s_{N_T}]$: Partial vector symbol

Tree Representation

We can associate partial vector symbols with nodes in a tree

- $\mathbf{s}^{(i)} = [s_i \quad \cdots \quad s_{N_T}]$ labels the nodes on level i
- The leaves correspond to the set of complete vector-symbols $\mathcal{O}^{N_T}$

$$\begin{matrix} |e_1|^2 \leftarrow \\ |e_2|^2 \leftarrow \\ |e_3|^2 \leftarrow \end{matrix} \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \hat{y}_3 \end{bmatrix} - \begin{bmatrix} R_{11} & R_{21} & R_{31} \\ 0 & R_{22} & R_{32} \\ 0 & 0 & R_{33} \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix}$$



$$\mathbf{s}^{(3)} = \begin{bmatrix} s_3 \end{bmatrix}^T \qquad d_3(\mathbf{s}^{(3)}) = |e_3|^2$$

$$\mathbf{s}^{(2)} = \begin{bmatrix} s_2 & s_3 \end{bmatrix}^T \qquad d_2(\mathbf{s}^{(2)}) = d_3 + |e_2|^2$$

$$\mathbf{s}^{(1)} = \begin{bmatrix} s_1 & s_2 & s_3 \end{bmatrix}^T \qquad d_1(\mathbf{s}^{(1)}) = d_2 + |e_1|^2$$

Partial Euclidean distance

- Branches are associated with distance increments $|e_i|^2$
- Nodes are associated with PEDs $d_i(\mathbf{s}^{(i)})$

- **ML detection**: traverse the tree to find the leaf with the smallest PED $d_1(\mathbf{s}^{(1)})$

PED Computation During Tree Traversal

- PEDs can be computed sequentially when proceeding from a node on level $i+1$ to one of its children on level i :

$$d_i(\mathbf{s}) = d_{i+1}(\mathbf{s}) + |e_i|^2$$

$$|e_i|^2 = \left| \tilde{y}_i - \sum_{j=i+1}^{N_T} r_{ij} s_j - r_{ii} s_i \right|^2$$

where $\sum_{j=i+1}^{N_T} r_{ij} s_j$ is defined by the partial symbol parent node on level $i+1$ and $r_{ii} s_i$ is determined by the selected child

- Note: PEDs increase monotonically since $d_i(\mathbf{s}) = d_{i+1}\left(\mathbf{s}^{(i+1)}\right) + |e_i|^2 \geq d_{i+1}(\mathbf{s})$ when proceeding from a node to one of its children
- $d_1\left(\mathbf{s}^{(1)}\right)$

Complexity

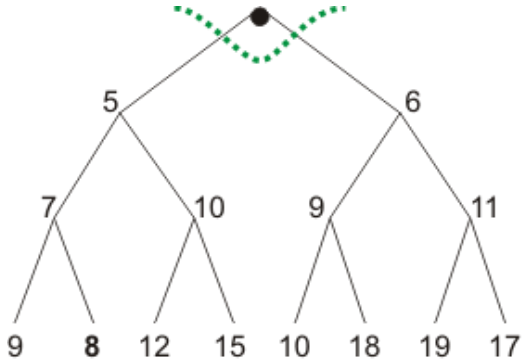
- Recursive computation of PEDs reduces complexity by reusing intermediate results
- Unfortunately, the number of leaves is not reduced

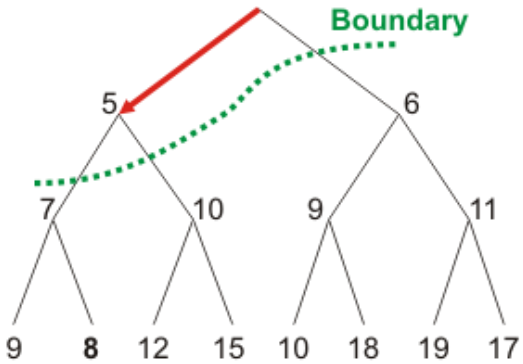
Solution

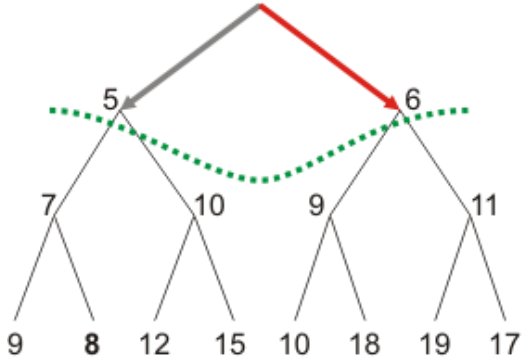
- Different well known techniques from computer science

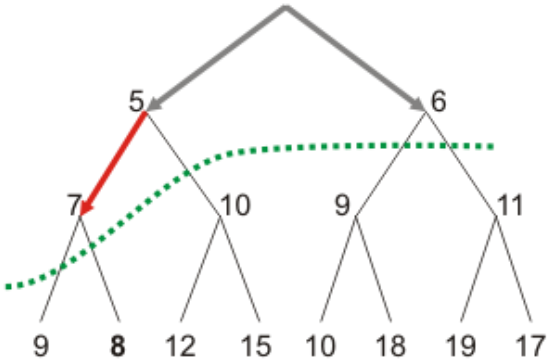
Dijkstra Search

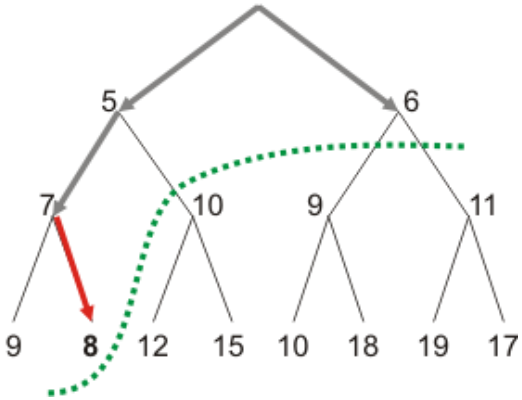
- The visited nodes define the boundary of the tree
- In each step, examine the children of the nodes in the boundary of the tree
- Across all nodes along the boundary: identify the child with the smallest PED and add it to the boundary
- The ML solution is found when a leaf is reached











- ✓ **Minimum number of visited nodes**
- **Boundary can grow large => significant memory requirements**
- **Finding the next node to expand requires complex sorting**

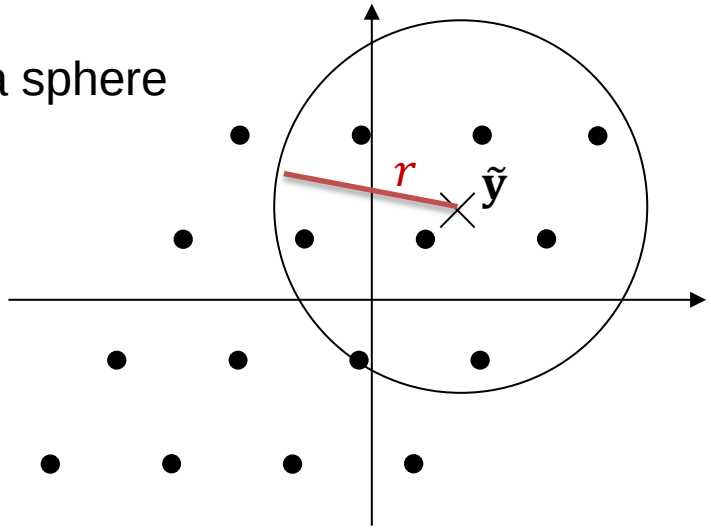
Sphere Decoding: Limiting the Search Space

Objective: reduce the complexity of the ML search

The Sphere Constraint

- Idea: restrict the search to points $\mathbf{Rs}$ inside a sphere with radius r around $\tilde{\mathbf{y}}$

Sphere constraint:
$\|\tilde{\mathbf{y}} - \mathbf{Rs}\|^2 < r^2$



but we need an efficient way to check this constraint

- Note: PEDs are monotonically increasing since $d_i(\mathbf{s}) = d_{i+1}(\mathbf{s}) + |e_i|^2 \geq d_{i+1}(\mathbf{s})$ when proceeding from a node to one of its children

Apply the sphere constraint to also to PEDs (all nodes in the tree)

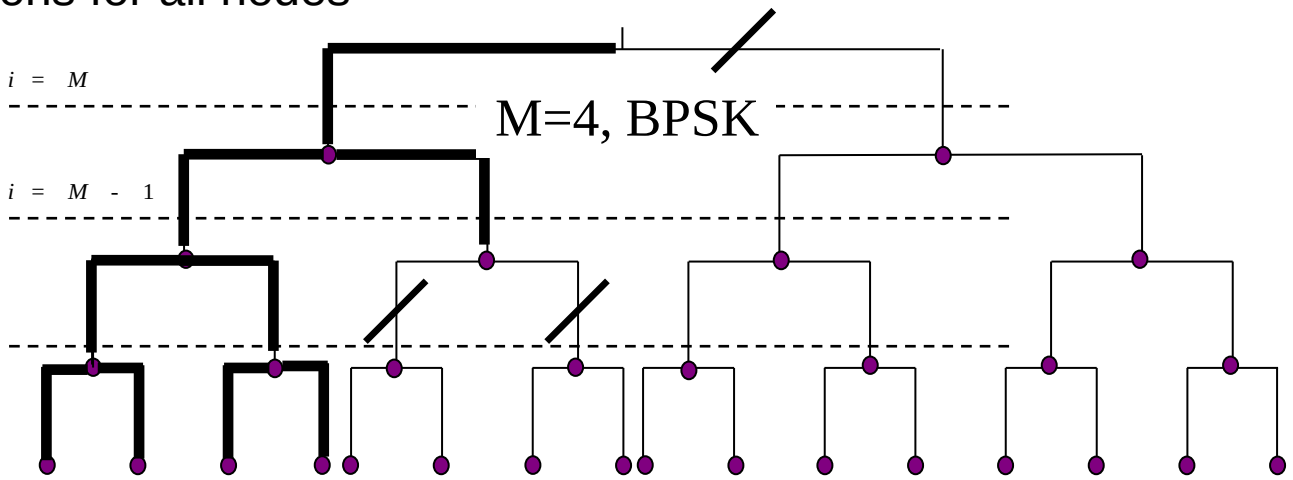
Tree Pruning

Sphere constraint: $d_i(\mathbf{s}) < r^2$

- If the PED of a node violates the sphere constraint, also all its children will violate the constraint

Tree pruning leads to complexity reduction

- When traversing the tree, prune (i.e., do not follow) all branches that lead to a node that violates the sphere constraint
- Avoids PED computations for all nodes below a branch that has been pruned



Example:

- Only 4 out of 16 leafs survive!!

Tree Traversal Strategy

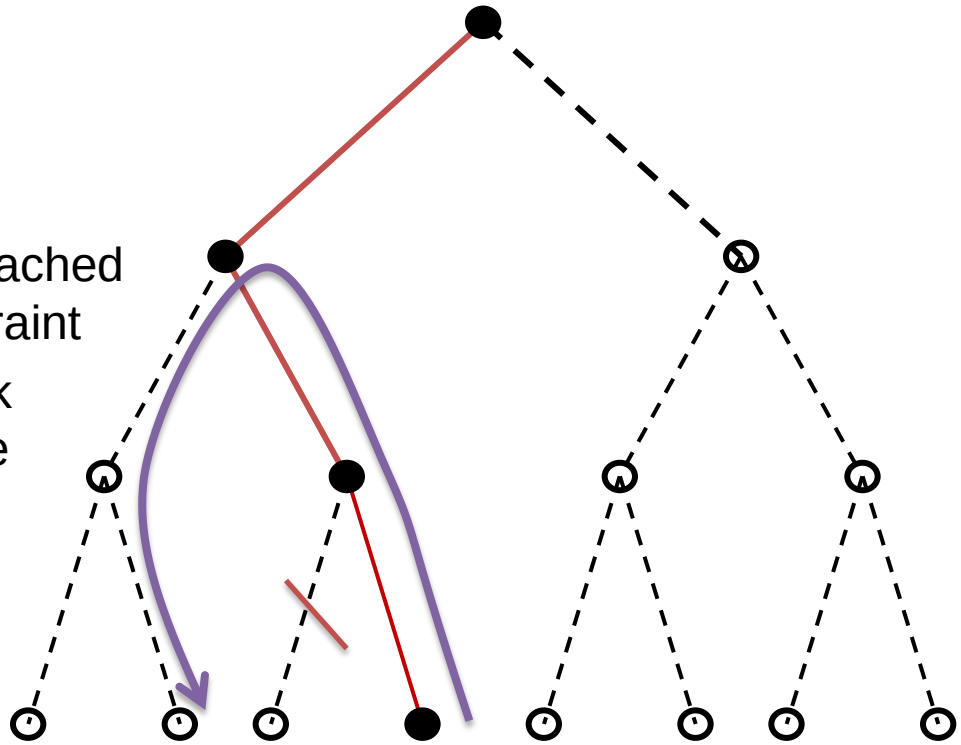
In what order should we traverse the tree?

Goals

- Always proceed to a new leaf as quickly as possible to reduce the radius
- Avoid large lists of intermediate candidates (partially expanded nodes) that are still needed

Depth first tree traversal

- Start from the root
- Forward: proceed down until a leaf is reached or until no child meets the sphere constraint
- When forward iteration ends, move back up and continue with the next child node along the current path that meets the sphere constraint



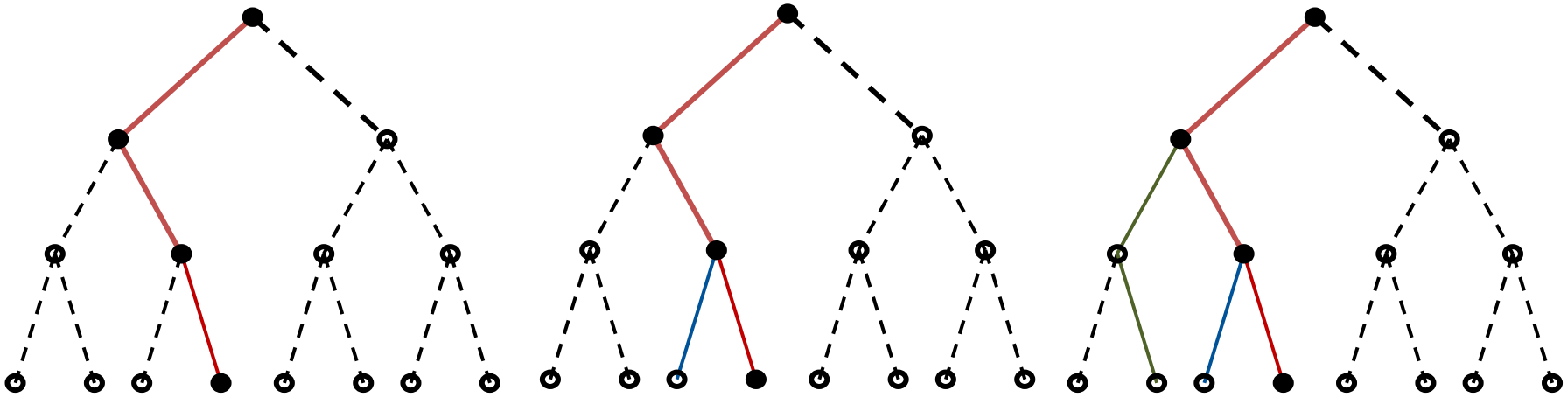
Depth-First Tree Traversal Strategy

Avoid large lists of intermediate candidates (partially expanded nodes) that are still needed

Algorithm

- Start from the root
- Forward: proceed down until a leaf is reached or until no child meets the sphere constraint
- When forward iteration ends, move back up and continue with the next child node along the current path that meets the sphere constraint

Example (without pruning)

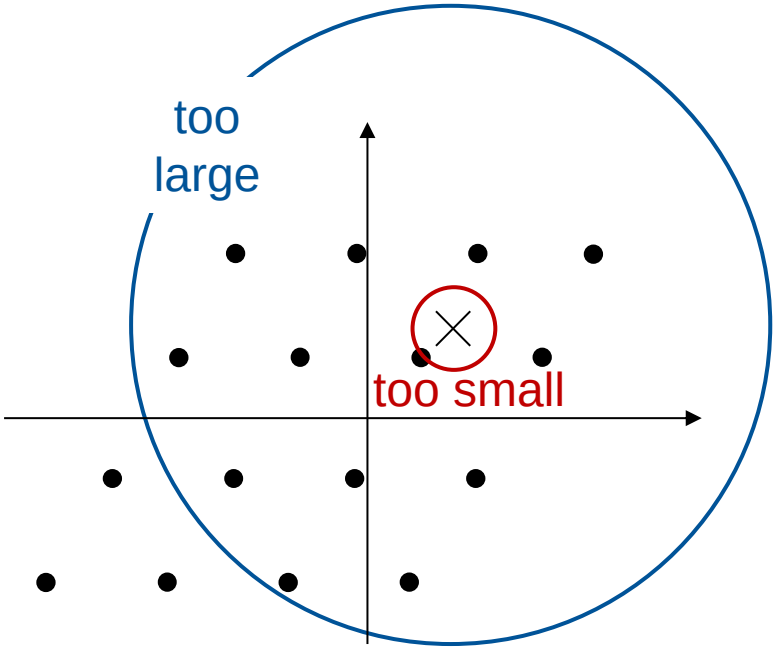


Choosing the Radius for the Sphere Constraint

The choice of the radius is important

- A small radius reduces the complexity
- However, if the radius is chosen too small, it may not even contain the ML solution
  ⇨ search has to be restarted with larger radius

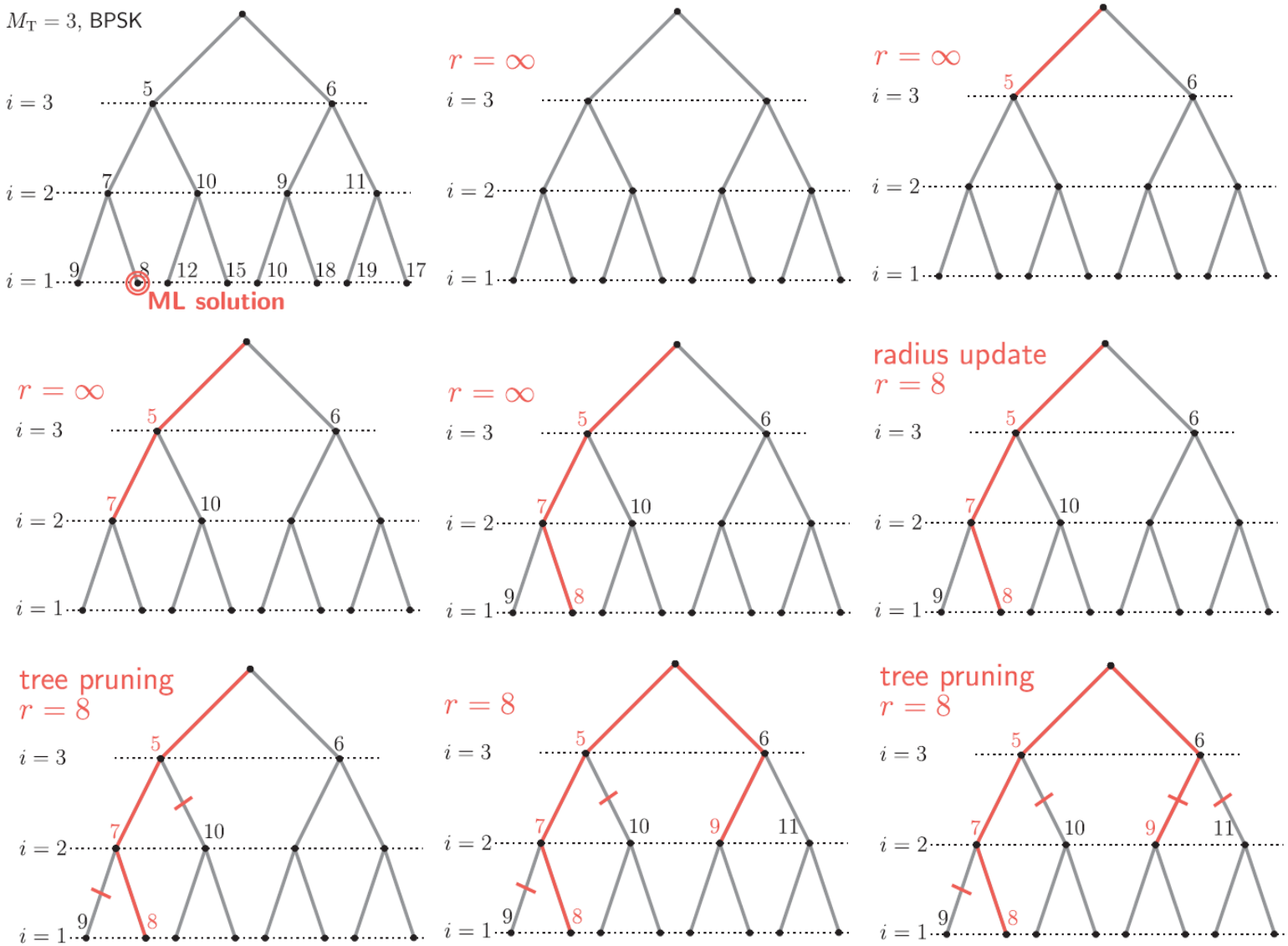
- Different proposals exist to select a suitable radius



Most popular proposition: RADIUS REDUCTION

- Start with an infinite initial radius ($r = \infty$)
- Update the radius whenever a leaf is reached ($r \leftarrow d_i(\mathbf{s})$)
  - The ML solution is always found
  - The radius shrinks monotonically
- Often referred to as “branch-and-bound” strategy

Depth First Tree Traversal with Radius Reduction



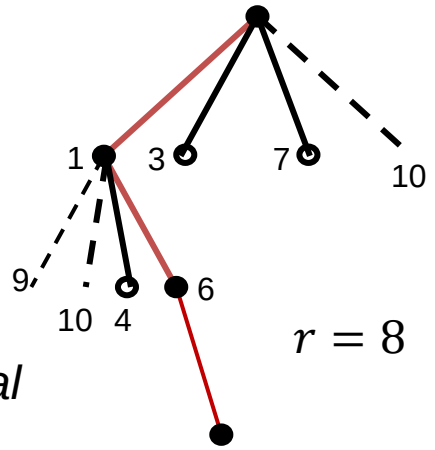
Checking the Sphere Constraint during Tree Traversal

How can we identify which children of a node s_i meet the sphere constraint?

- Start from the sphere constraint

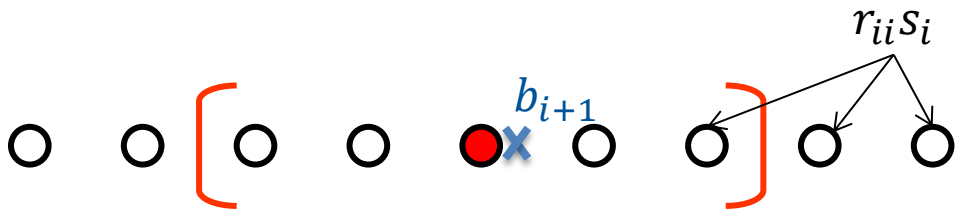
$$d_{i+1}\left(\mathbf{s}^{(i+1)}\right) + \left|b_{i+1}\left(\mathbf{s}^{(i+1)}\right) - r_{ii}s_i\right|^2 < r^2$$

$$b_{i+1}\left(\mathbf{s}^{(i+1)}\right) = \tilde{y}_i - \sum_{j=i+1}^{N_T} r_{ij}s_j$$



- Assume real-valued PAM constellations s_i

- Simply solving the inequality for s_i yields an *admissible interval*



Identify the (best) child s_i with the smallest PED (distance increment)

- Solve $b_{i+1}\left(\mathbf{s}^{(i+1)}\right) = r_{ii}s_i$ and quantize to the nearest constellation point

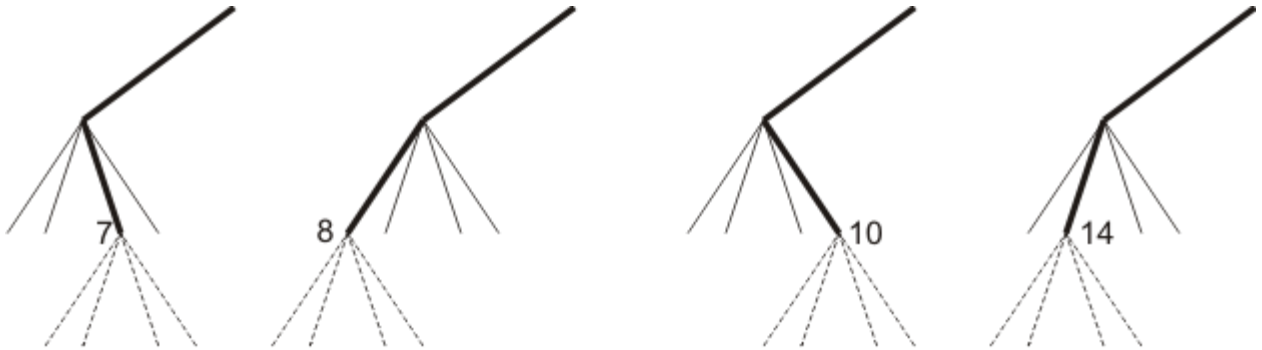
$$s_i = \mathrm{Q}\left(b_{i+1}\left(\mathbf{s}^{(i+1)}\right)/r_{ii}\right)$$

Enumeration Strategies

- Without radius updating, the **order** in which the children of a node are examined is **irrelevant**

- With radius updating, it is desirable to find the best possible solution **as early as possible**

Schnorr-Euchner enumeration: examine branches (children of a node) in ascending order of their PEDs (distance increments $|e_i|^2$)



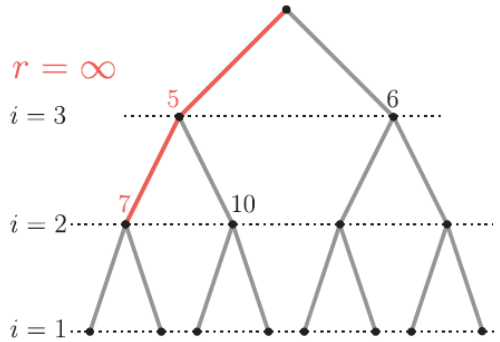
- The first leaf corresponds to the z**ero forcing SIC solution**
- Faster shrinkage of the radius ⇨ more efficient tree pruning
- **Requires ordering** of candidates in the admissible set

Schnorr-Euchner (SE) Enumeration

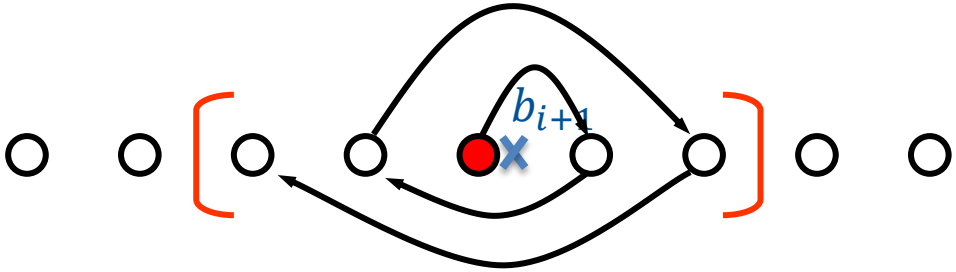
SE in the real-valued case

- Starting point (first child to be visited): best child (center of admissible interval)

$$s_i = \mathrm{Q}\left(b_{i+1}\left(\mathbf{s}^{(i+1)}\right)/r_{ii}\right)$$



- When later visiting the remaining children: enumerate children (symbols) in the admissible interval in zig-zag order



Enumeration stops when

- the computed boundary of the admissible interval is reached
- OR equivalently, when the sphere constraint check for an enumerated child fails
  ➔ no need to explicitly compute the boundary of the admissible interval!!

Schnorr-Euchner (SE) Enumeration

SE in the complex-valued case (QAM)

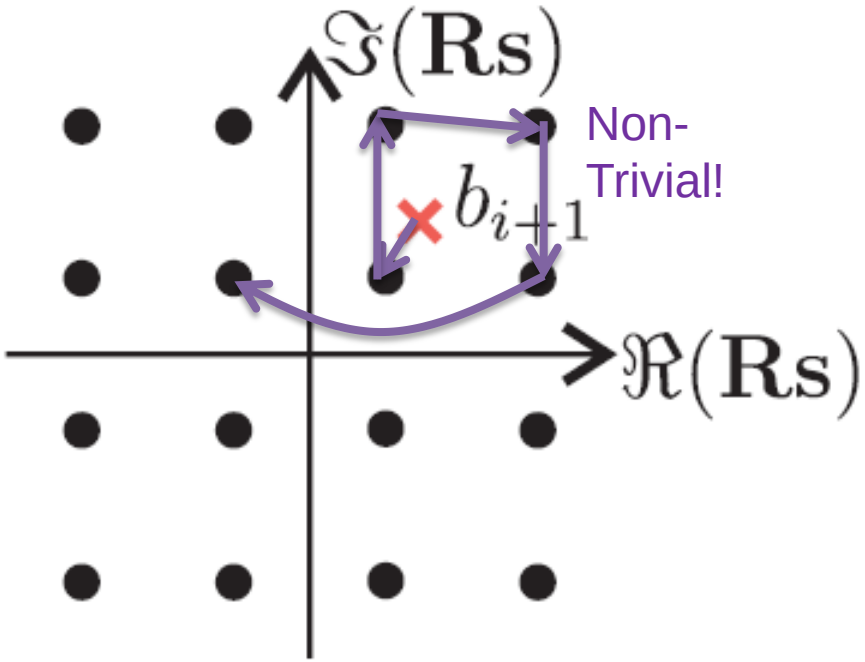
- Enumeration starting point can be found as in the real-valued case, by solving

$$s_i = \mathrm{Q}\left(b_{i+1}\left(\mathbf{s}^{(i+1)}\right)/r_{ii}\right)$$

However, in the complex plane

- Admissible intervals can not be defined
- no pre-defined order exists for enumeration!!

So what can we do?

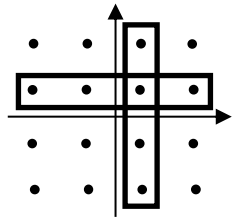


Schnorr-Euchner (SE) Enumeration with QAM

Real-valued decomposition

- Idea: write a complex-valued system as a real-valued system

$$\begin{bmatrix}\Re\{\mathbf{y}\}\\ \Im\{\mathbf{y}\}\end{bmatrix} = \begin{bmatrix}\Re\{\mathbf{H}\} & -\Im\{\mathbf{H}\}\\ \Im\{\mathbf{H}\} & \Re\{\mathbf{H}\}\end{bmatrix}\begin{bmatrix}\Re\{\mathbf{s}\}\\ \Im\{\mathbf{s}\}\end{bmatrix} + \begin{bmatrix}\Re\{\mathbf{n}\}\\ \Im\{\mathbf{n}\}\end{bmatrix}$$



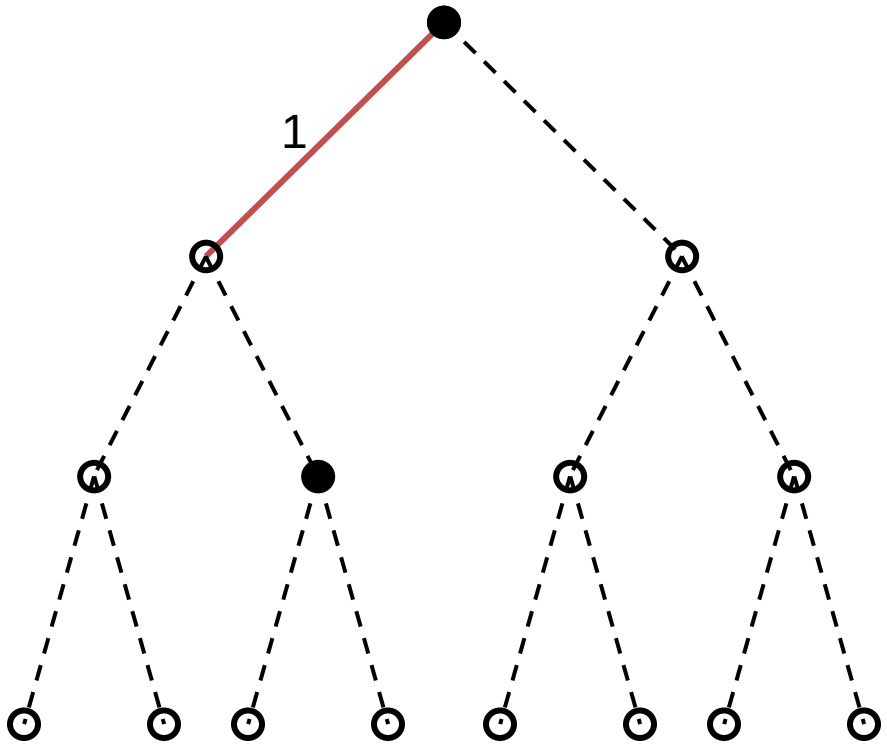
- M-QAM system (complex valued) with N_T streams is transformed into an $\sqrt{M}$-PAM system (real-valued) with $2N_T$ streams

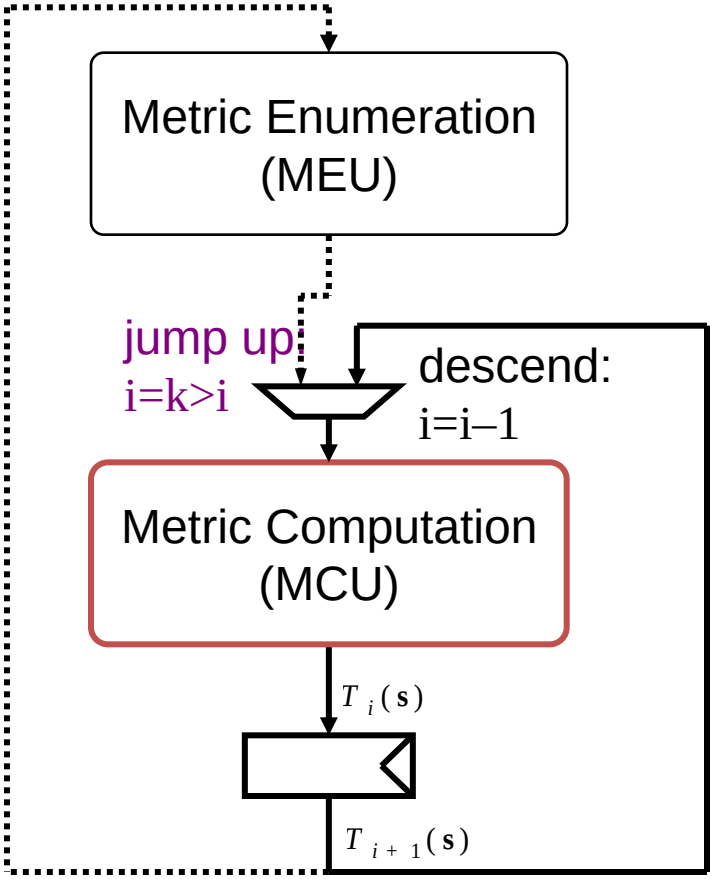
Exhaustive search enumeration

- Compute PEDs for all children of a node
- Enumerate by explicitly sorting children according to their PEDs
- Enumeration stops by

One cycle-per-node VLSI architecture

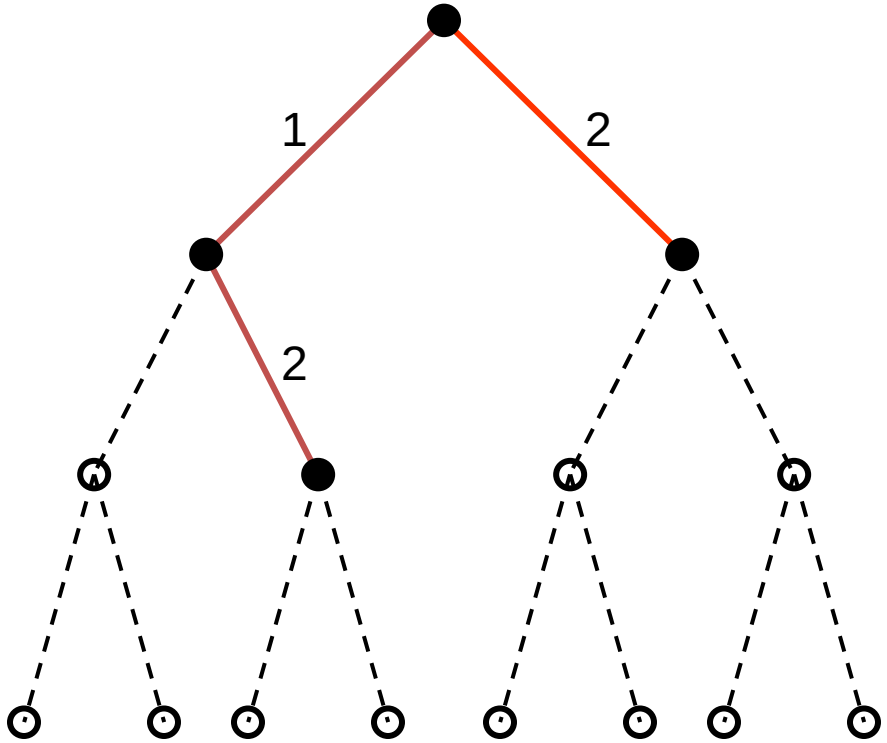
- MCU finds the starting point for Schnorr-Euchner enumeration on the current node and computes the corresponding PED
- MCU advances to next level: $i=i-1$

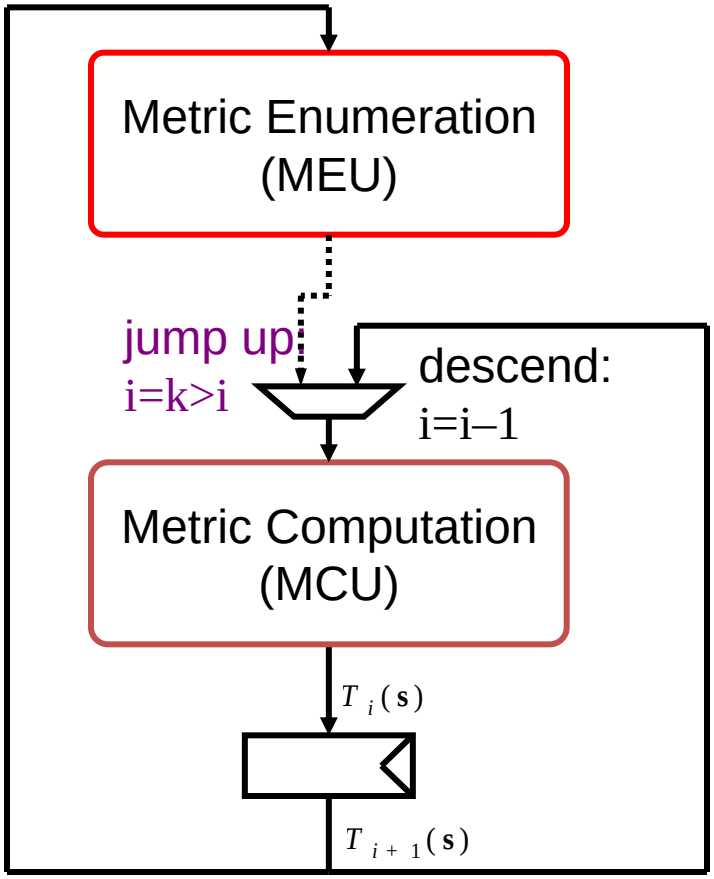




One cycle-per-node VLSI architecture

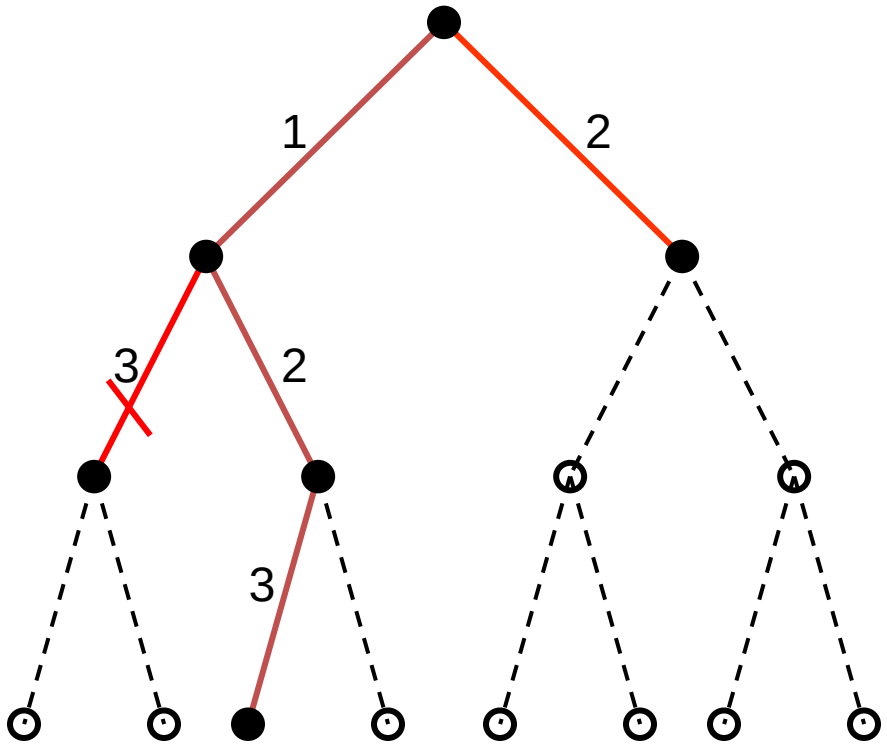
- The MEU follows the MCU with one cycle delay and performs the Schnorr-Euchner enumeration
- MCU advances to next level: $i=i-1$

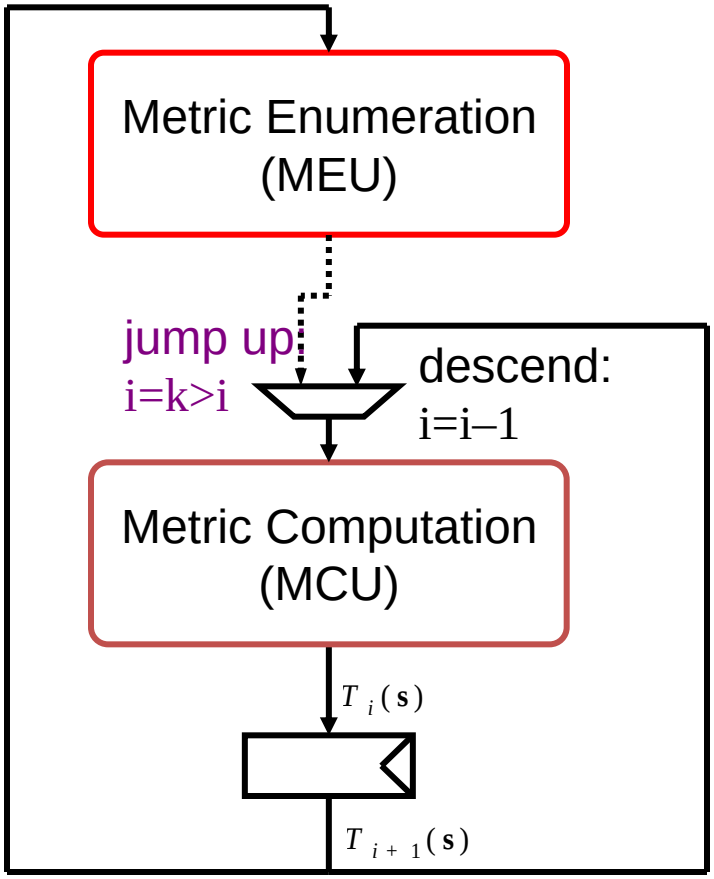




One cycle-per-node VLSI architecture

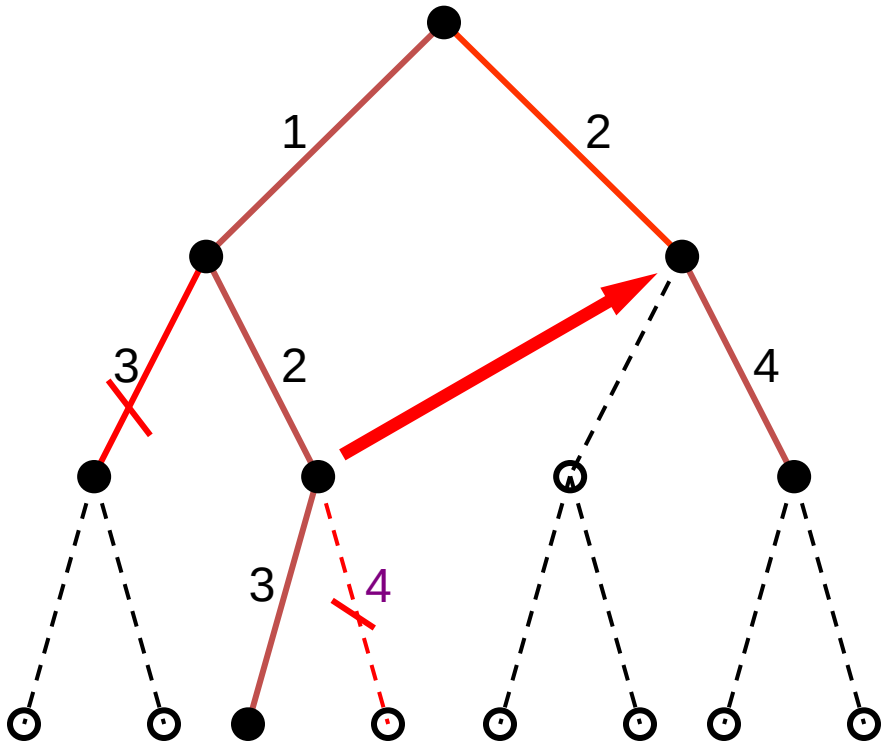
- MEU keeps a list of the preferred (i.e., lowest PED) children along the current path
- The list is ordered depth-first

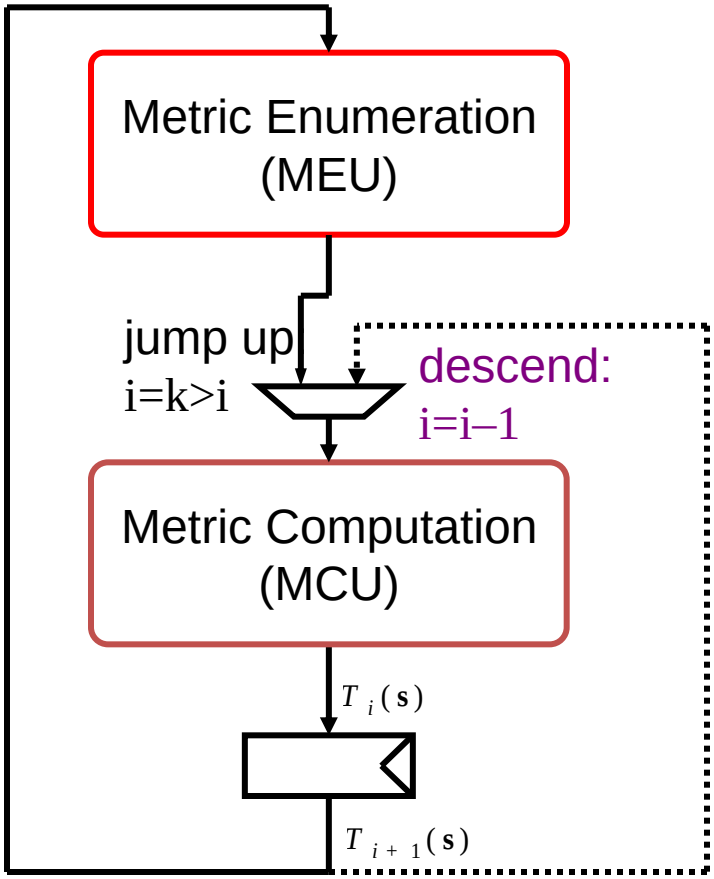




One cycle-per-node VLSI architecture

- MEU provides the PED of a new node from its list
- MCU updates the radius and jumps immediately to a new node

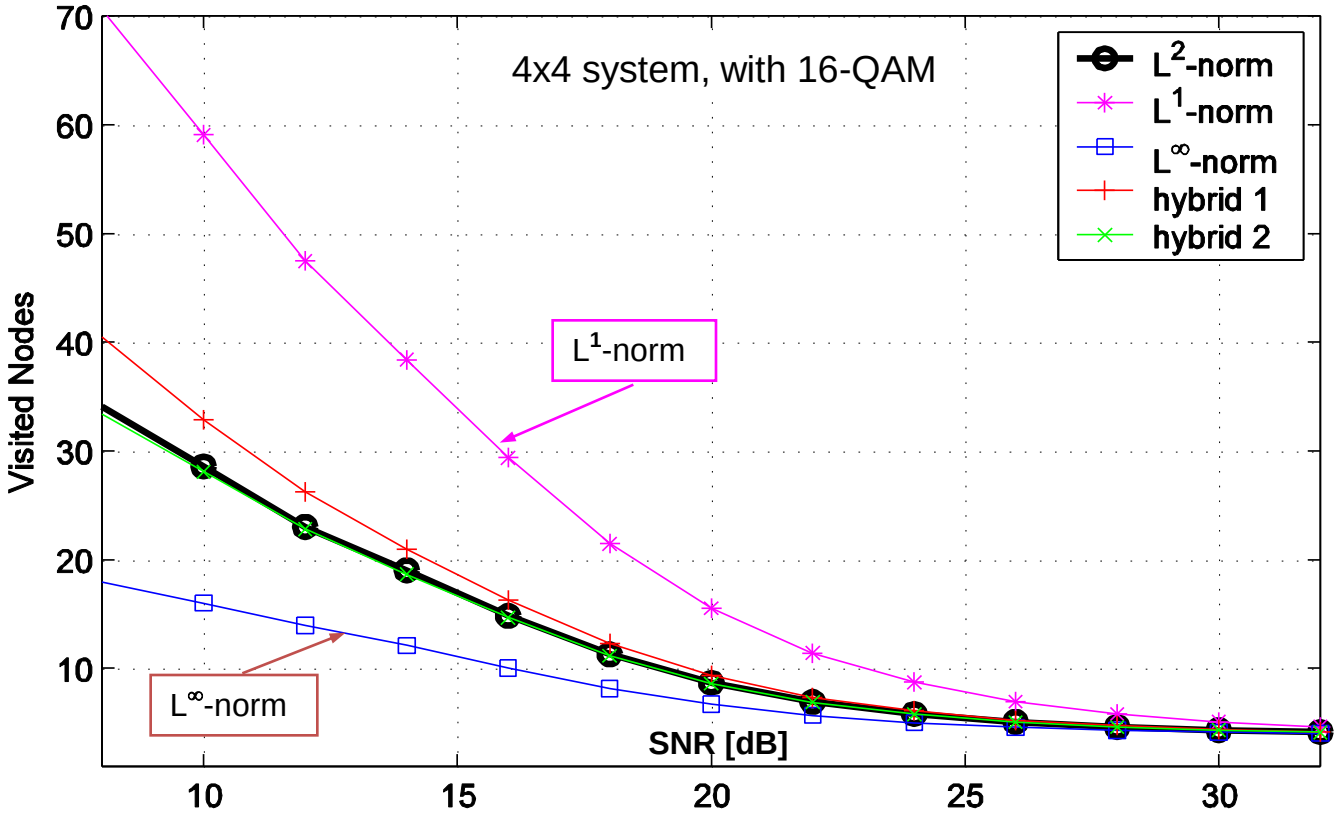




Complexity of Sphere Decoding

Unfortunately, the effort for traversing the tree is not fixed.

- The instantaneous effort depends on the individual symbol and noise realization
- The average effort depends on the SNR

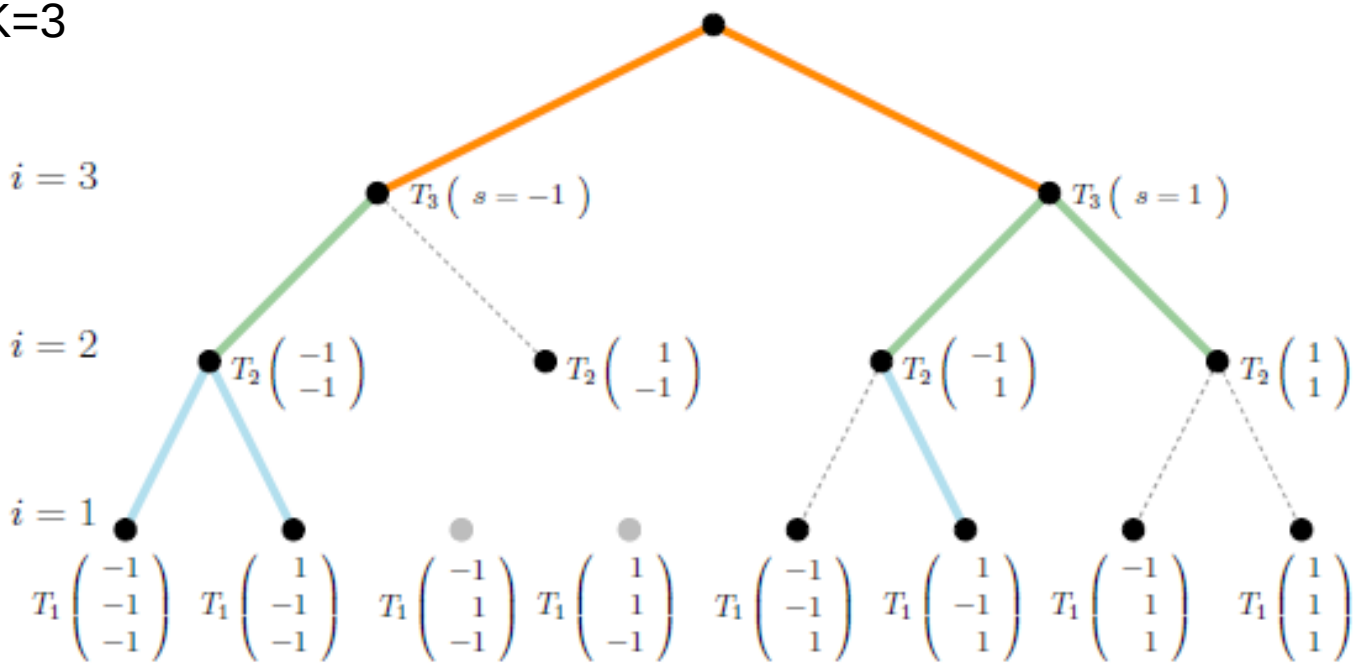


K-Best Algorithm

Tree search algorithm with a complexity constraint

- Traverse the tree “breadth first”: proceeds only in forward direction
- On each level, keep only the K nodes with the smallest PEDs and continue with their children

Example: K=3

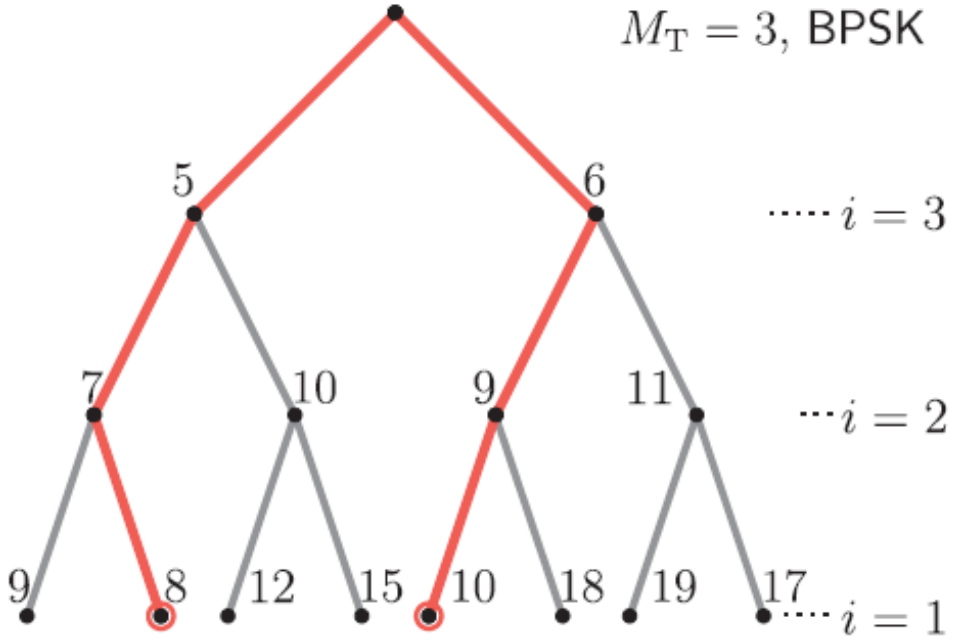


- Can not achieve ML performance

FSD Algorithm

Another fixed complexity scheme

- Perform a special type of sorting (worst layer first)
- Expand all children on the first level(s)
- On the subsequent levels expand (follow) only the best child (=SIC)



- Also does not necessarily find the ML solution, BUT …