EFFICIENT SEQUENTIAL DECISION-MAKING ALGORITHMS FOR CONTAINER INSPECTION OPERATIONS

Sushil Mittal and Fred Roberts

Rutgers University & DIMACS

David Madigan

Columbia University & DIMACS

Port of Entry Inspection Algorithms

- Goal: Find ways to intercept illicit nuclear materials and weapons destined for the U.S. via the maritime transportation system
- Currently inspecting only small % of containers arriving at ports

Port of Entry Inspection Algorithms

Aim: Develop decision support algorithms that will help us to “optimally” intercept illicit materials and weapons subject to limits on delays, manpower, and equipment

Find inspection schemes that minimize total cost including cost of false positives and false negatives

Mobile VACIS: truck-mounted gamma ray imaging system

Sequential Decision Making Problem

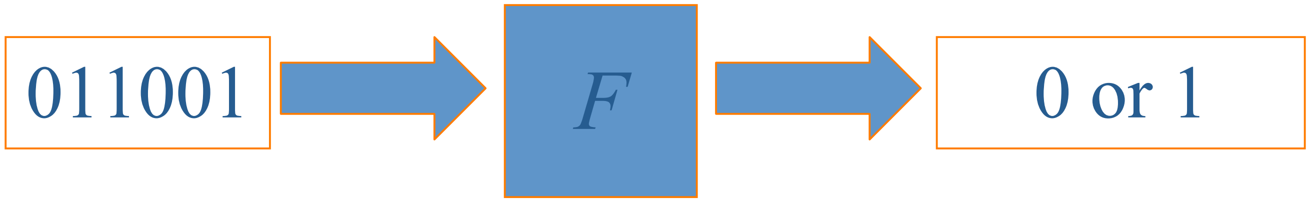
- Containers arriving are classified into categories
- Simple case: 0 = “ok”, 1 = “suspicious”
- Containers have attributes, either in state 0 or 1
- **Sample attributes**:
  - Does the ship’s manifest set off an alarm?
  - Is the neutron or Gamma emission count above certain threshold?
  - Does a radiograph image return a positive result?
  - Does an induced fission test return a positive result?
- **Inspection scheme:**
  - *specifies which inspections are to be made based on previous observations*
- Different “sensors” detect presence or absence of various attributes

Sequential Decision Making Problem

- Simplest Case: Attributes are in state 0 or 1

- Then: Container is a *binary string* like 011001

- So: Classification is a *decision function* F that assigns each binary string to a category.



If attributes 2, 3, and 6 are present, assign container to category $F(011001)$.

Sequential Decision Making Problem

- If there are two categories, 0 and 1, decision function F is a *Boolean function*.

- Example:

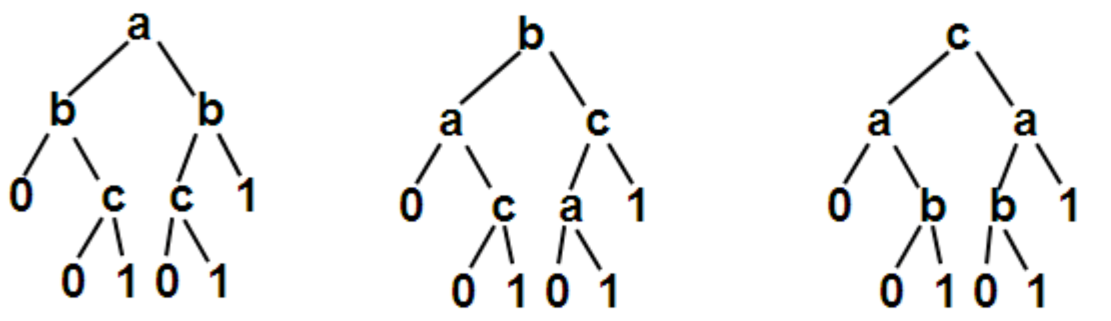
a	b	c	F(abc)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

- This function classifies a container as positive iff it has at least two of the attributes.

Binary Decision Tree Approach

- Binary Decision Tree:
  - Nodes are sensors or categories (0 or 1)
  - Two arcs exit from each sensor node, labeled left and right.
  - Take the right arc when sensor says the attribute is present, left arc otherwise

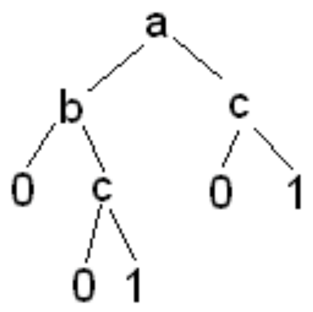
a	b	c	F(abc)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



Cost of a BDT

- Cost of a BDT comprises of:
  - Cost of utilization of the tree and
  - Cost of misclassification

A BDT, τ
with n = 3

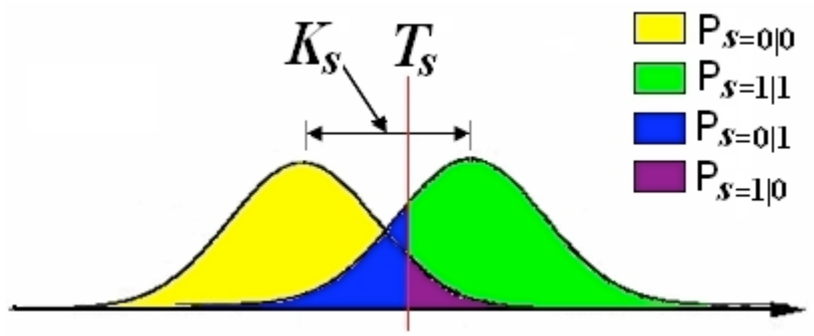


$$\begin{aligned}f(\tau) = {} & P_0(C_a + P_{a0|0}C_b + P_{a0|0}P_{b1|0}C_c + P_{a1|0}C_c) \\ & + P_1(C_a + P_{a0|1}C_b + P_{a0|1}P_{b1|1}C_c + P_{a1|1}C_c) \\ & + P_0(P_{a0|0}P_{b1|0}P_{c1|0} + P_{a1|0}P_{c1|0})C_{FP} \\ & + P_1(P_{a0|1}P_{b0|1} + P_{a0|1}P_{b1|1}P_{c0|1} + P_{a1|1}P_{c0|1})C_{FN}\end{aligned}$$

P_1 is prior probability of occurrence of a bad container

$P_{i|j}$ is the conditional probability that given the container was in state j, it was classified as i

Sensor Thresholds



$$P_{s=0|0} + P_{s=1|0} = 1$$
$$P_{s=1|1} + P_{s=0|1} = 1$$

- T_s can be adjusted for minimum cost
- Anand et. al. reported the cheapest trees obtained from an extensive search over a range of sensor thresholds. For example: for n=4, 194,481 tests were performed with thresholds varying between [-4,4] with a step size of 0.4

Previous work: A quick overview

- **Approach**:
  - Builds on ideas of Stroud and Saeger[1] at Los Alamos National Laboratory
  - Inspection schemes are implemented as Binary Decision Trees which are obtained from various Boolean functions of different attributes
  - Only "Complete" and "Monotonic" Boolean functions give potentially acceptable Binary decision trees
  - n=4

1 Stroud, P. D. and Saeger K. J., "*Enumeration of Increasing Boolean Expressions and Alternative Digraph Implementations for Diagnostic Applications", Proceedings Volume IV, Computer, Communication and Control Technologies, (2003), 328-333*

Optimum Threshold Computation

- Extensive search over a range of thresholds has some practical drawbacks:
  - Large number of threshold values for every sensor
  - Large step size
  - Grows exponentially with the number of sensors (computationally infeasible for $n > 4$)
- Therefore, we utilize non-linear optimization techniques like:
  - Gradient descent method
  - Newton's method

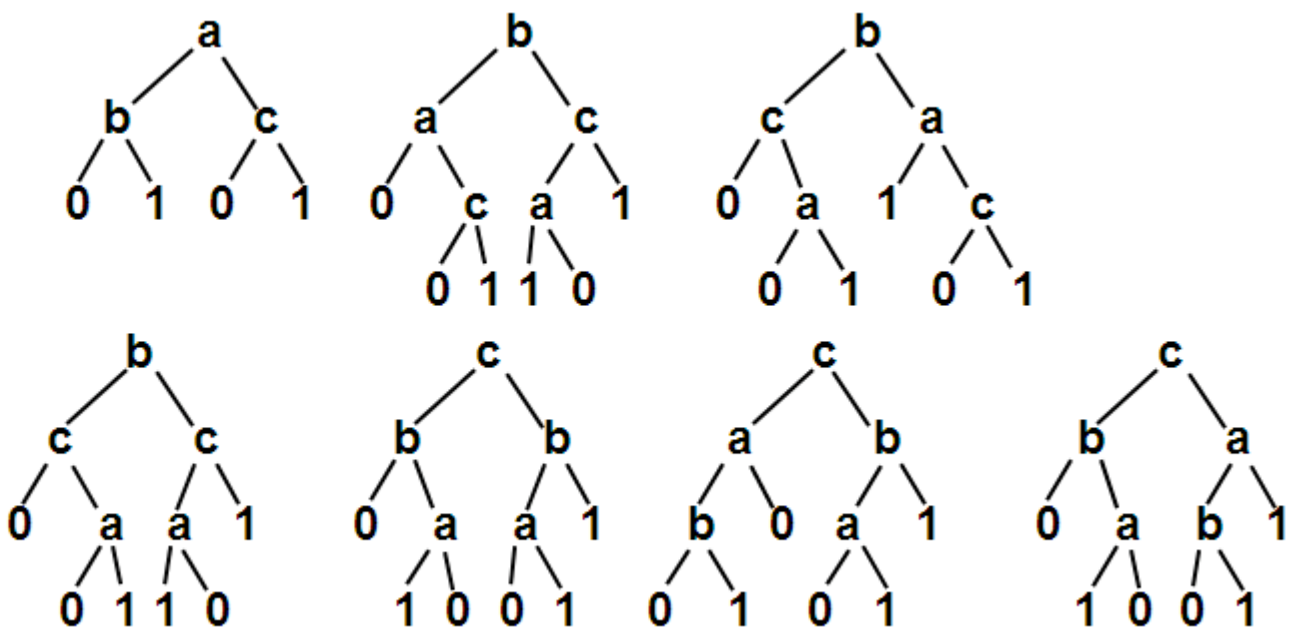
Searching through a Generalized Tree Space

- We expand the space of trees from Stroud and Saeger's "Complete" and "Monotonic" Boolean Functions to Complete and Monotonic BDTs, because…
- Unlike Boolean functions, BDTs may not consider all sensor outputs to give a final decision
- Advantages:
  - Allows more, potentially useful trees to participate in the analysis
  - Helps defining an irreducible tree space for search operations
  - Moves focus from Boolean Functions to Binary Decision Trees

Revisiting Monotonicity

- **Monotonic Decision Trees**
  - A binary decision tree will be called monotonic if all the left leafs are class “0” and all the right leafs are class “1”.
- Example:

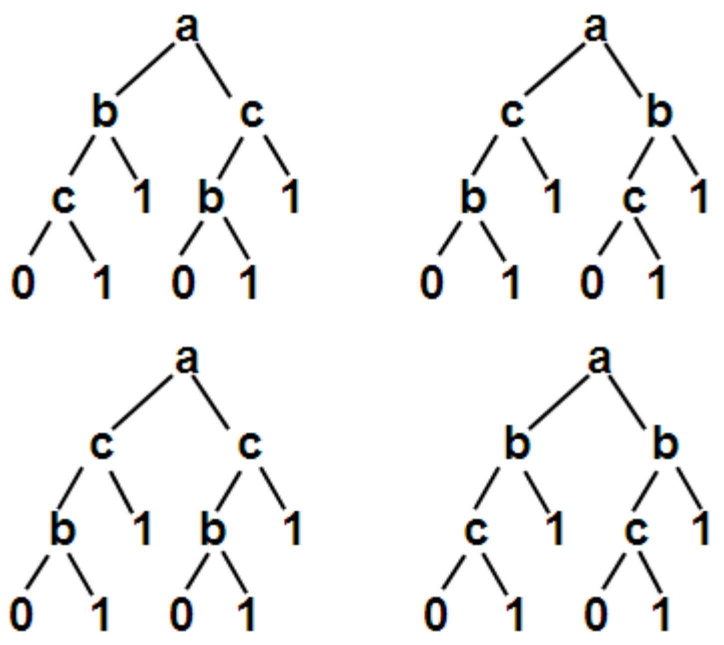
a	b	c	F(abc)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1



Revisiting Completeness

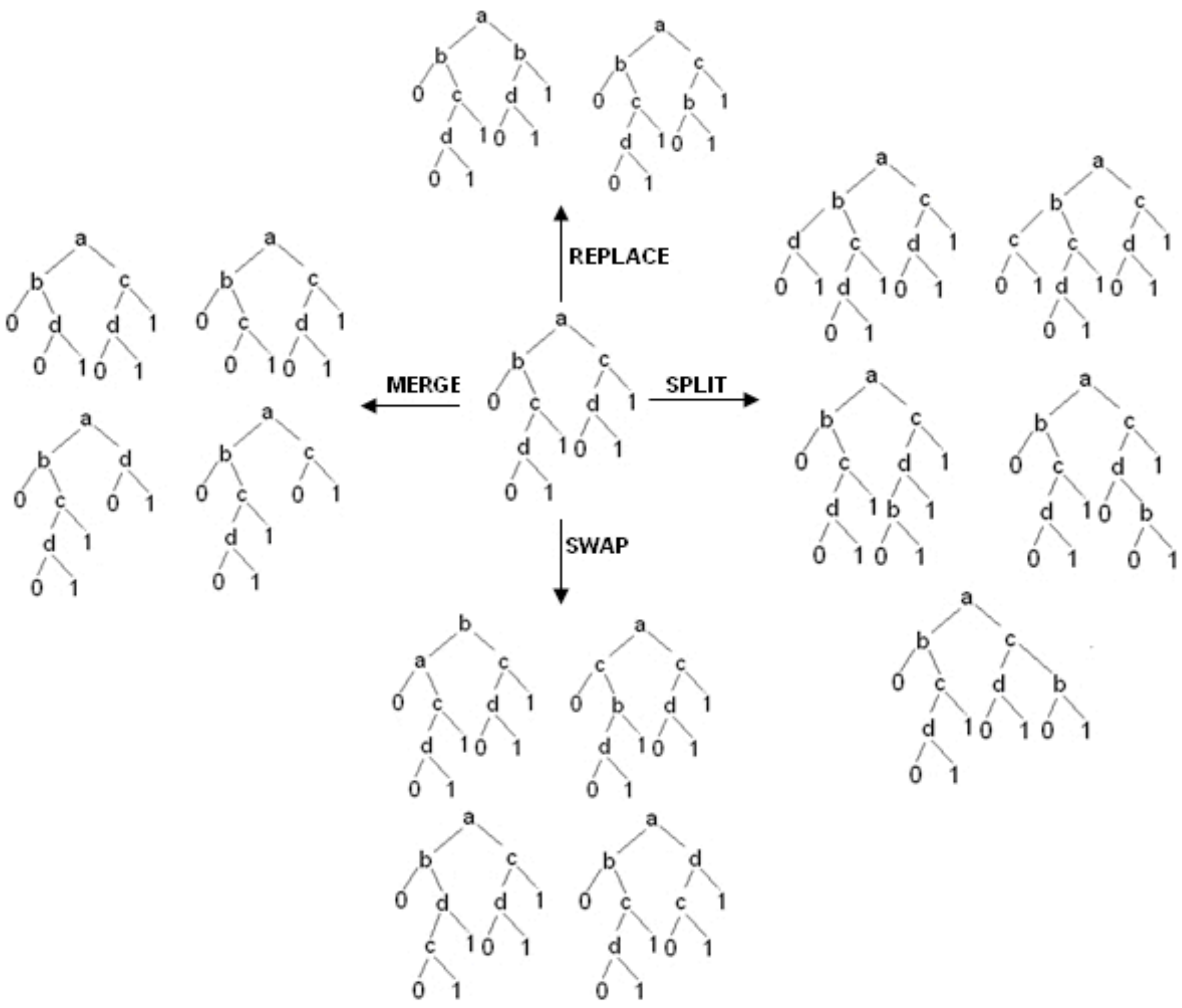
- **Complete Decision Trees**
  - A binary decision tree will be called complete if every sensor occurs at least once in the tree and at any non-leaf node in the tree, its left and right sub-trees are not identical.
- Example:

a	b	c	F(abc)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



The CM Tree Space

No. of attributes	*Distinct BDTs*	*Trees From CM Boolean Functions*	*Complete and Monotonic BDTs*
2	*74*	*4*	*4*
3	*16,430*	*60*	*114*
4	*1,079,779,602*	*11,808*	*66,000*



REPLACE
MERGE
SPLIT
SWAP

Tree Space Traversal

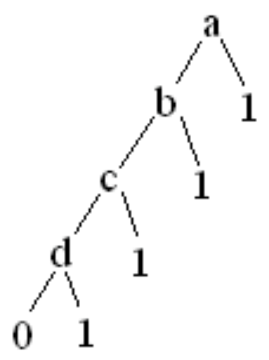
- Greedy Search
  1. Randomly start at any tree in the CM tree space
  2. Find its neighboring trees using neighborhood operations
  3. Move to the neighbor with the lowest cost
  4. Iterate till the solution converges

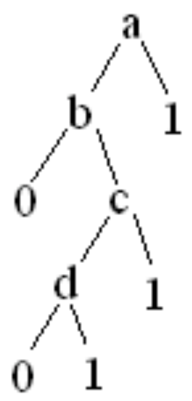
  – The CM Tree space has a lot of local minima. For example: 9 in the space of 114 trees for 3 sensors and 193 in the space of 66,000 trees for 4 sensors.

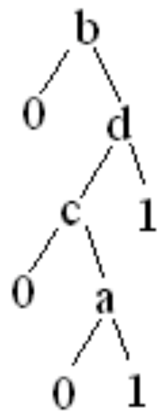
- Proposed Solutions
  - Stochastic Search Method with Simulated Annealing
  - Genetic Algorithms based Search Method

Tree Space Irreducibility

- We have proved that the CM tree space is irreducible under the neighborhood operations
- Simple Tree:
  - A simple tree is defined as a CM tree in which every sensor occurs exactly once in such a way that there is exactly one path in the tree with all sensors in it.



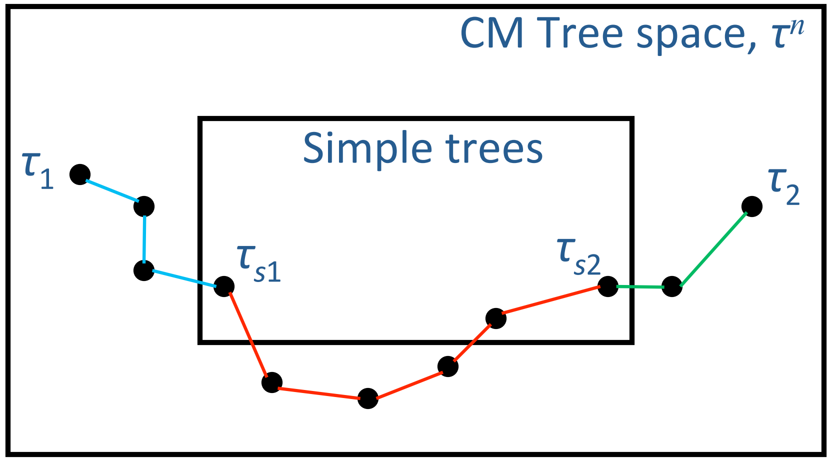




To Prove: Given any two trees τ_1, τ_2 in CM tree space, τ^n, τ_2 can be reached from τ_1 by an arbitrary sequence of neighborhood operations

We prove this in three different steps:

1. Any tree τ_1 can be converted to a simple tree τ_{s1}
2. Any simple tree τ_{s1} can be converted to any other simple tree τ_{s2}
3. Any simple tree τ_{s2} can be converted to any tree τ_2



Results

- Significant computational savings over previous methods
- Have run experiments with up to 10 sensors
- Genetic algorithms especially useful for larger scale problems

Current Work

- Tree equivalence
- Tree reduction and irreducible trees
- Canonical form representation of the equivalence class of trees
- Revisiting completeness and monotonicity

Thank You!

Previous work: A quick overview

Monotonic Boolean Functions:

•Given two strings $x_1x_2\ldots x_n$, $y_1y_2\ldots y_n$

•F is monotonic iff $x_i \geq y_i$ for all i implies that

$$F(x_1x_2\ldots x_n) \geq F(y_1y_2\ldots y_n).$$

Complete Boolean Functions:

•Boolean function F is incomplete if F can be calculated by finding at most n-1 attributes and knowing the value of the input string on those attributes

•In other words, F is complete if all the attributes contribute towards the output

Previous work: A quick overview

- Stroud and Saeger: "brute force" algorithm for enumerating binary decision trees implementing complete, monotonic Boolean functions and choosing least cost BDT.

No. of attributes	*Distinct BDTs*	*CM Boolean Expressions*	*BDTs from CM Boolean Functions*
2	*74*	*2*	*4*
3	*16,430*	*9*	*60*
4	*1,079,779,602*	*114*	*11,808*
5	*5×10^{18}*	*6894*	*263,515,920*

Infeasible beyond $n > 4$!

Problems with Standard Approaches

- Gradient Descent Method: Setting the value of the step size heuristically, since:
  - Too small step size: long time to converge
  - Too big step size: might skip the minimum
- Newton's Method:
  - The convergence depends largely on the starting point
  - Occasionally drifts in the wrong direction and hence fails to converge.
- Solution: *combination of gradient descent and Newton's methods*

The Combined Method

1. Initialize $\mathbf{T}$ as vector of random sensor threshold values
2. Compute $\partial\mathbf{f}$, $\mathbf{H}f(\tau)$
3. If $\mathbf{H}f(\tau)$ is not positive definite, then find a close approximation
4. If $\mathbf{H}f(\tau)$ is not well-conditioned, then take a few steps using gradient descent until it becomes well-conditioned
5. Take a step using Newton's method
6. Repeat steps 1-5 until the solution converges
7. Repeat steps 1-6 a few times and choose the overall

Tree Neighborhood and Tree Space

- Structure based methods
- Classification based methods
- We choose structure based neighborhood methods because :
  - Small changes in tree structure do not effect the cost significantly , and…
  - All BDTs with same Boolean function may differ a lot in cost

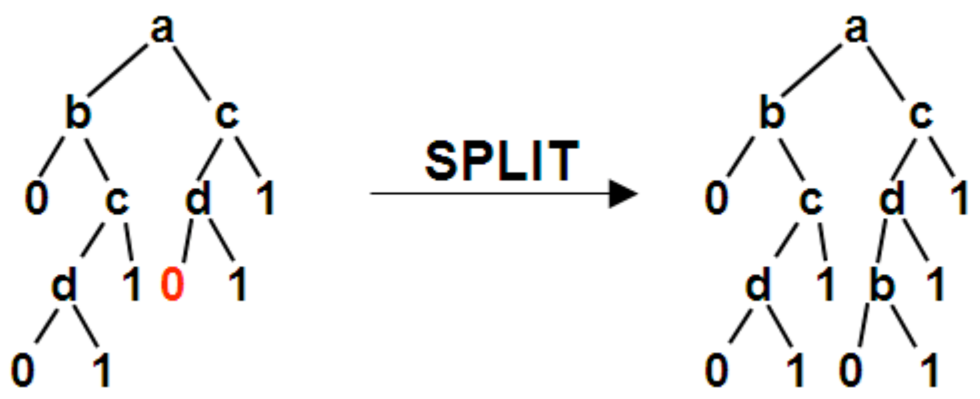
Tree Neighborhood and Tree Space

- Define tree neighborhood such that the Complete and Monotonic (CM) tree space is irreducible
- **Irreducibility**
  - Any tree in the CM tree space can be reached from any other tree by using the neighborhood operations repetitively
  - An irreducible CM tree space helps “search” for the cheapest trees using neighborhood operations

Search Operations

- *Split*

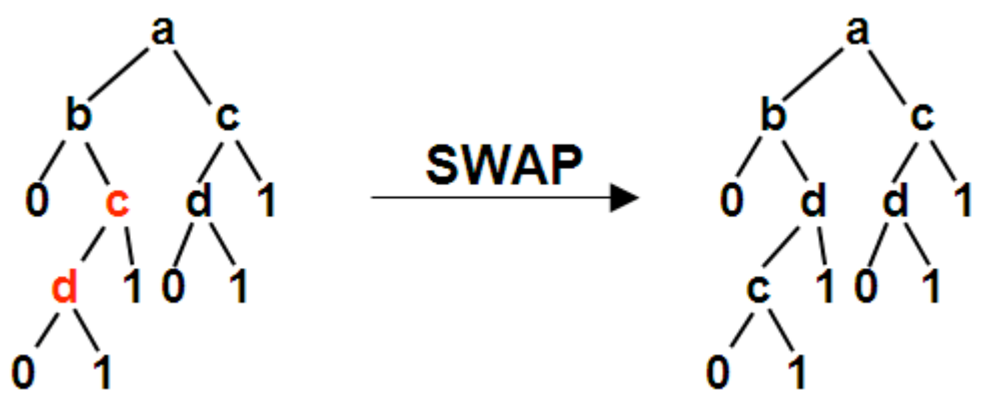
  Pick a leaf node and replace it with a sensor that is not already present in that branch, and then insert arcs from that sensor to 0 and to 1.



Search Operations

- *Swap*

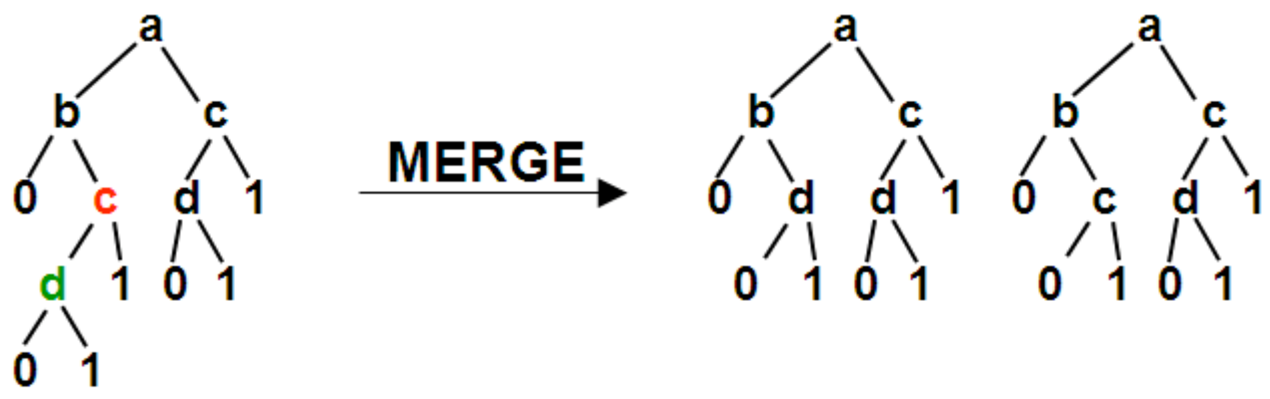
  Pick a non-leaf node in the tree and swap it with its parent node such that the new tree is still monotonic and complete and no sensor occurs more than once in any branch.



Search Operations

- *Merge*

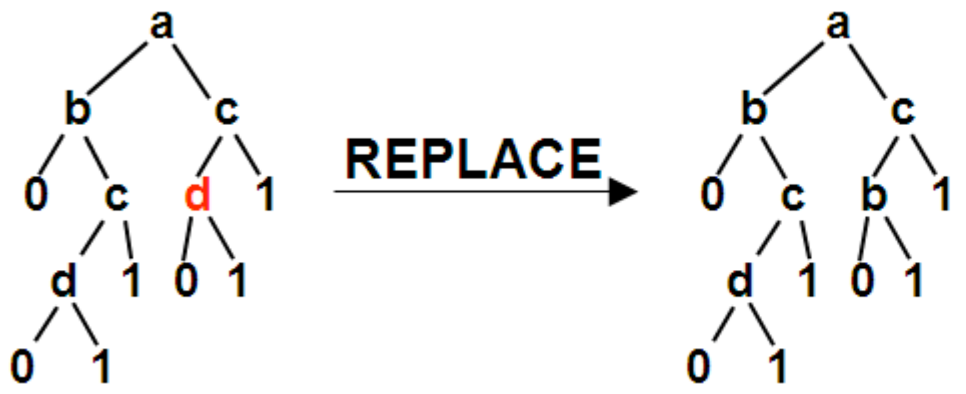
  Pick a parent node of two leaf nodes and make it a leaf node by collapsing the two leaf nodes below it, or pick a parent node with one leaf node, collapse both of them and shift the sub-tree up in the tree by one level.



Search Operations

- *Replace*

  Pick a node with a sensor occurring more than once in the tree and replace it with any other sensor such that no sensor occurs more than once in any branch.



Stochastic Search Method

1. Randomly start at any tree in CM space
2. Find its neighboring trees, and find their optimum costs
3. Select move according to the following probability. If we are at the ith tree τ_i, then the probability of going to its kth neighbor τ_{ik}, is given by

$$P_{ki} = \frac{\left(f(\tau_i)/f(\tau_{ik})\right)^{1/t}}{\sum_{j=1}^{n_i}\left(f(\tau_i)/f(\tau_{ij})\right)^{1/t}}$$

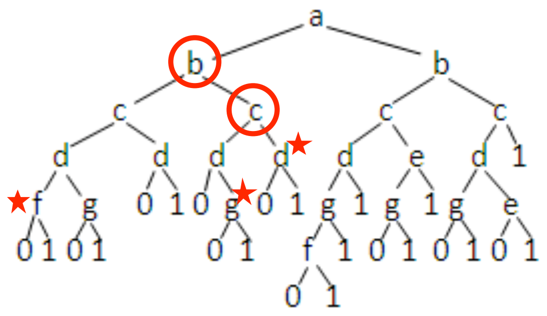
   where n_i is the number of neighboring trees of τ_i

4. Initialize the temperature $t = 1$, and lower it in discrete unequal steps after every m hops until the solution converges
5. Repeat steps 1-4 a few times and choose the overall minimum

Tree Space Irreducibility

1. $\tau_1 \longrightarrow \tau_{s1}$:

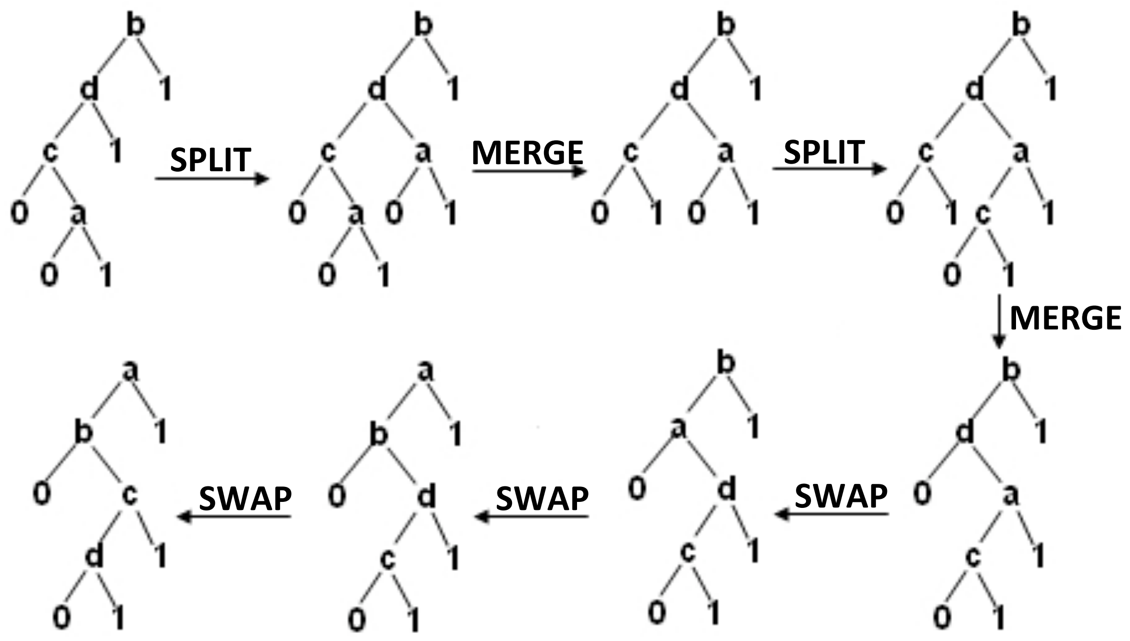
•Repeated subtree merger
•To remove a node at depth k, at most k-2 need to be checked for completeness
•We prove that there is at least one node in a subtree at any time, that can be merged without disturbing the overall completeness constraint



Tree Space Irreducibility

2. $\tau_{s1} \longrightarrow \tau_{s2}$:

- First convert τ_{s1} to have similar "skeleton" as τ_{s2}
- Then use repeated *Swap* operations



Tree Space Irreducibility

3. $\tau_{s2} \longrightarrow \tau_2$:

- The process of going from a tree to a simple tree is entirely reversible. For example:
  - any split operation can be reversed using a merge operation and vice-versa
  - swap and replace operations can be reversed by opposite swap and replace operations, respectively

- Therefore, $\tau_2 \longrightarrow \tau_{s2}$ implies $\tau_{s2} \longrightarrow \tau_2$

Genetic Algorithms based Search

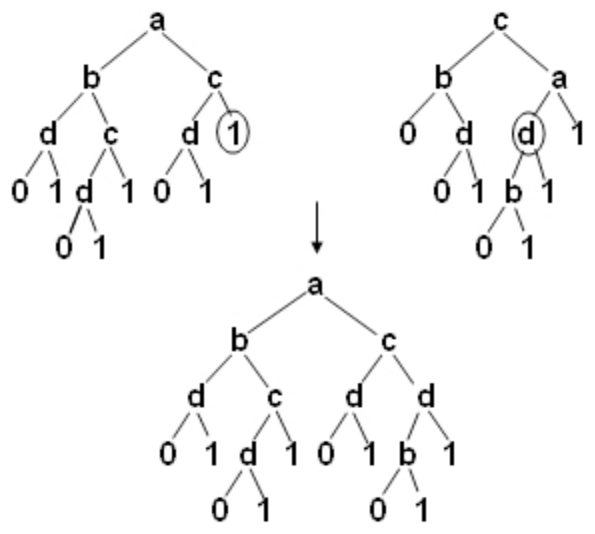
- The underlying idea is to get a population of "better" trees from a current population of "good" trees by using the basic operations:
  - Selection
  - Crossover
  - Mutation
- "better" decision trees correspond to the ones cheaper than the current ones ("good")

Genetic Algorithms based Search

- *Selection:*
  - Select a random, initial population of N trees from CM tree space
- *Crossover:*
  - Performed k times between every pair of trees in the current best population, $bestPop$

Genetic Algorithms based Search

– For each crossover operation between two trees τ_i and τ_j, we randomly select a node in each tree and exchange their subtrees

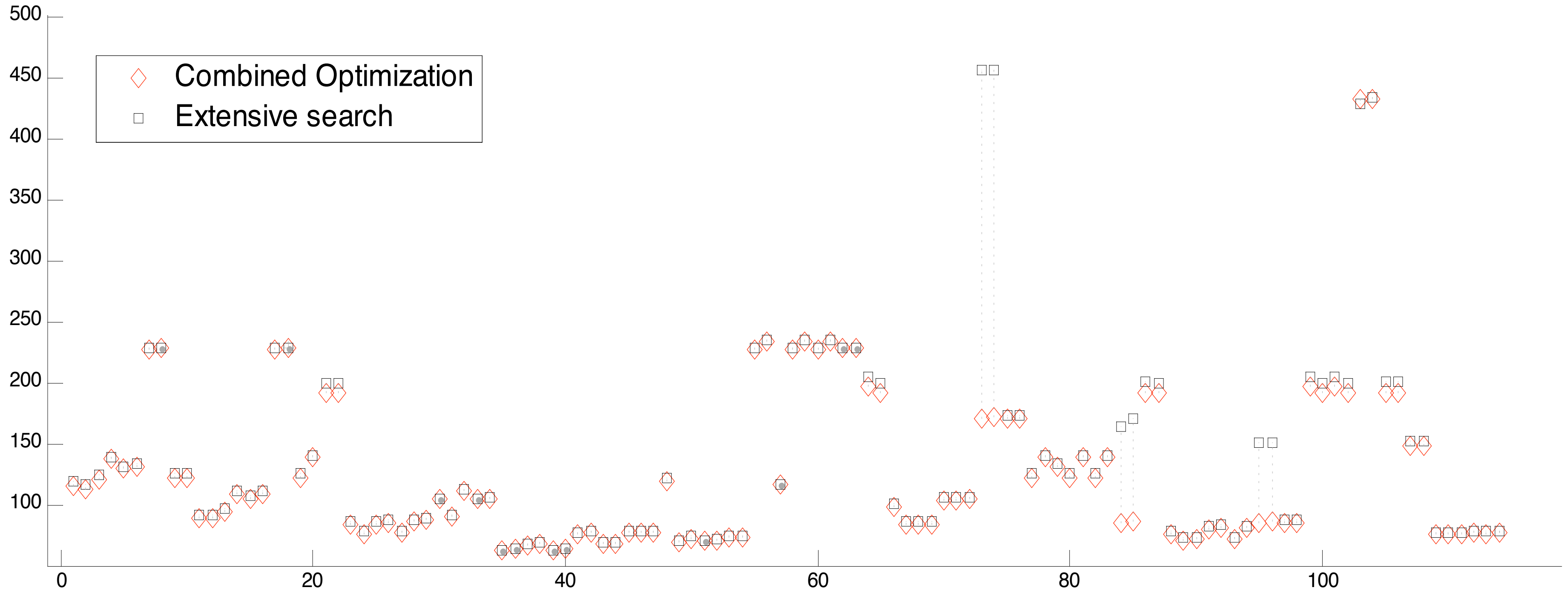


– However, we impose certain restriction on the selection of nodes, so that the resultant trees still lie in CM tree space

Genetic Algorithms based Search

- *Mutation:*
  - Performed after every m generations of the algorithm
  - We do two types of mutations:
    1. Generate all neighbors of the current best population and put them into the gene pool
    2. Replace a fraction of the trees of $bestPop$ with random trees from the CM tree space

Results I - Threshold Optimization



- Many times the minimum obtained using the optimization method was considerably less than the one from the extensive search technique.

Results II - Searching CM Tree Space

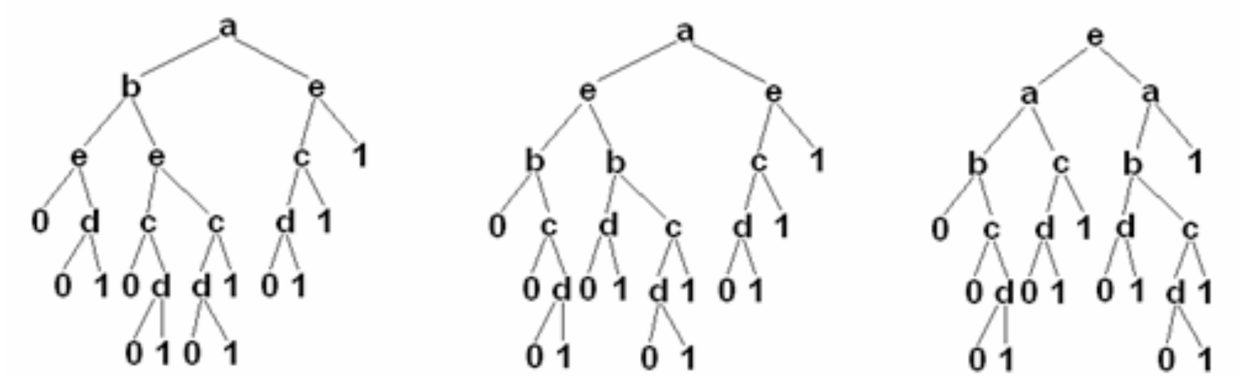
- Stochastic Search Method:
  - Successfully performed experiments for up to $n = 5$
  - For example, for 4 sensors (66,000 trees)
    - 100 different experiments were performed
    - Each experiment was started **10** times randomly at some tree and chains were formed by making stochastic moves in the neighborhood, until convergence
    - Only **4890** trees were examined on average for every experiment
    - Global minimum was found **82** times while the second best tree was found **10** times

Results II - Searching CM Tree Space

- Genetic Algorithms based Method:
  - Successfully performed experiments for up to $n = 10$
  - For 4 sensors (66,000 trees)
    - 100 different experiments were performed
    - Each experiment was started with a random population of **20** trees and was continued for **27** generations each; the mutations are performed after every **3** generations
    - Only **1440** trees were examined on average for every experiment
    - Global minimum was found all **100** times
    - The algorithm returns a whole population of good trees most of which belong to 50 best trees

Results II - Searching CM Tree Space

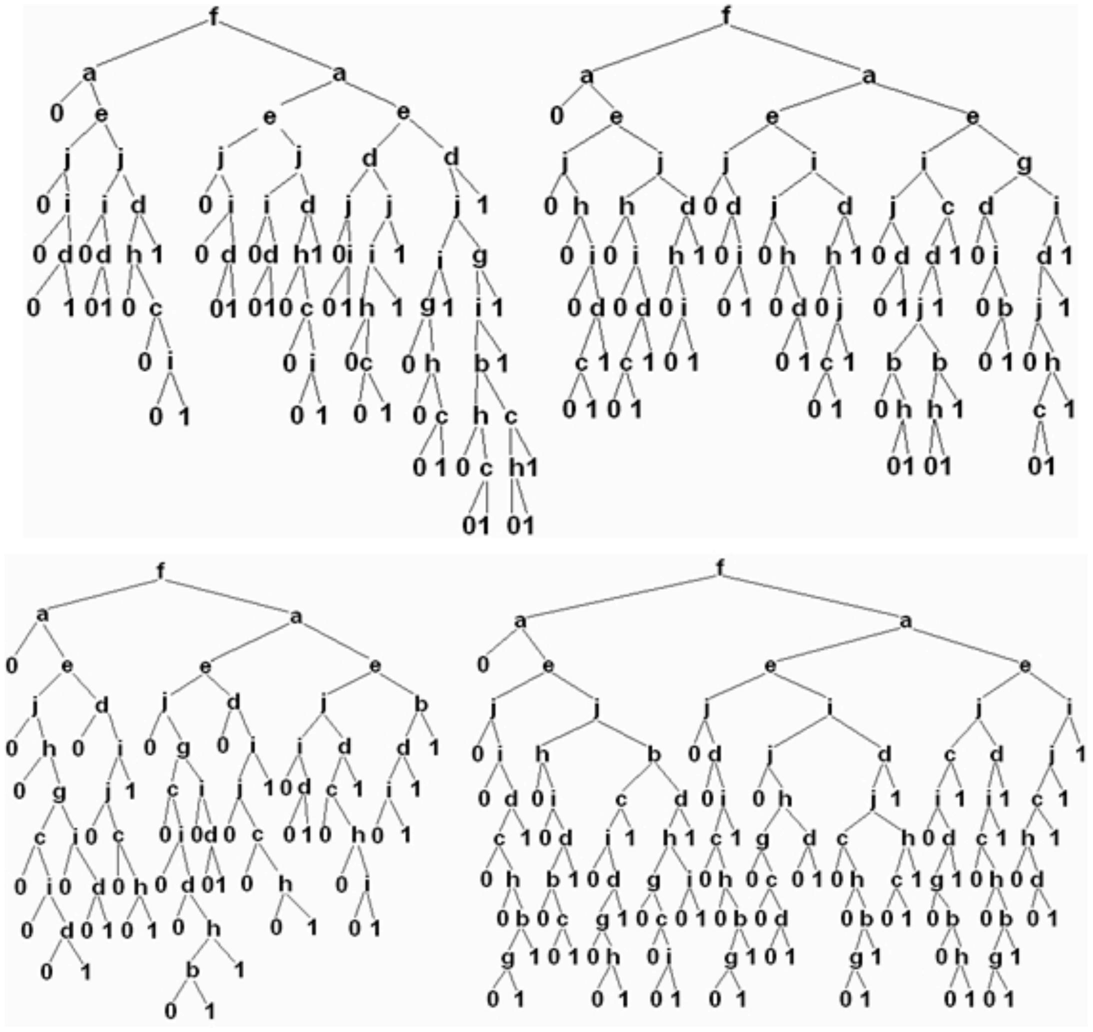
- Similarly, for n = 5, the tree space consists of more than 22.5 billion trees, we always obtained one of the following best trees:



- Each of these trees costs 41.4668

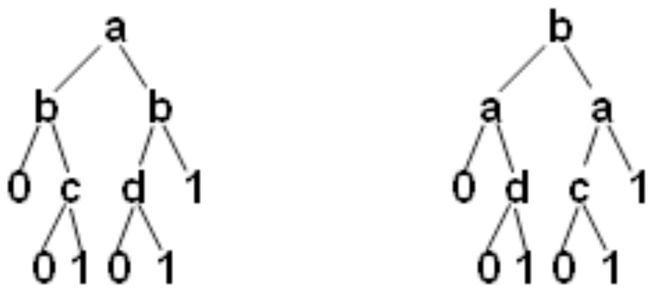
Results II - Searching CM Tree Space

- For n = 10, following were the best trees over a few runs:



Current Work

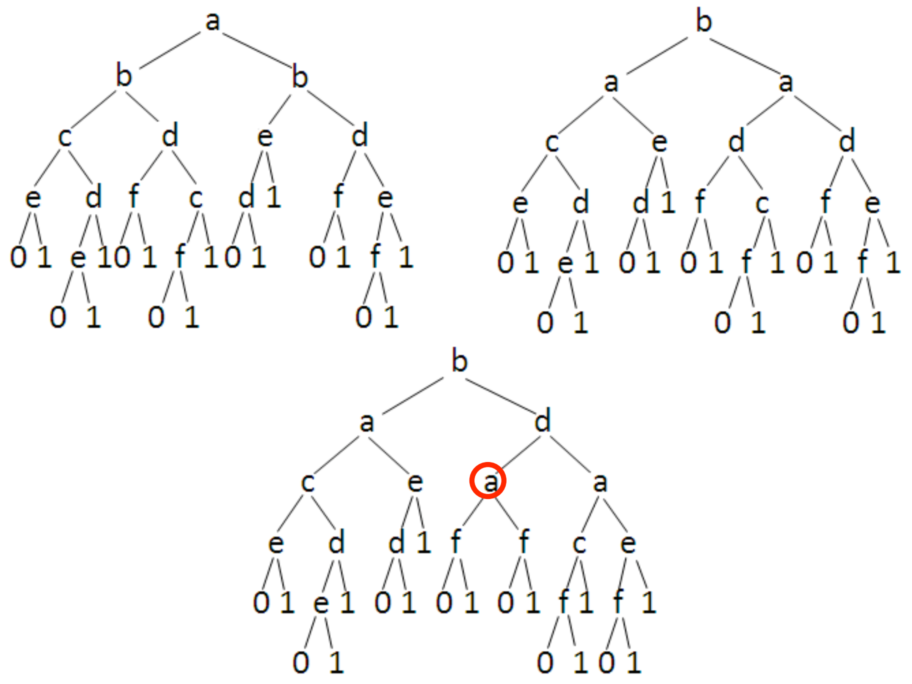
- Tree Equivalence
  - –*Decision Equivalence*: Two or more decision trees are called decision equivalent if their underlying Boolean function is same
  - –*Cost Equivalence:* Two trees are called cost equivalent iff they are "transposes" of each other. For example:



  - –The size of largest equivalence class also increases more than double exponentially with n
  - –Therefore, we define a space of equivalence classes of decision trees, with a unique, canonical representation of each class

Current Work

- Tree Reduction and Irreducible Trees
  - –A transpose of a complete tree can be incomplete. For example:



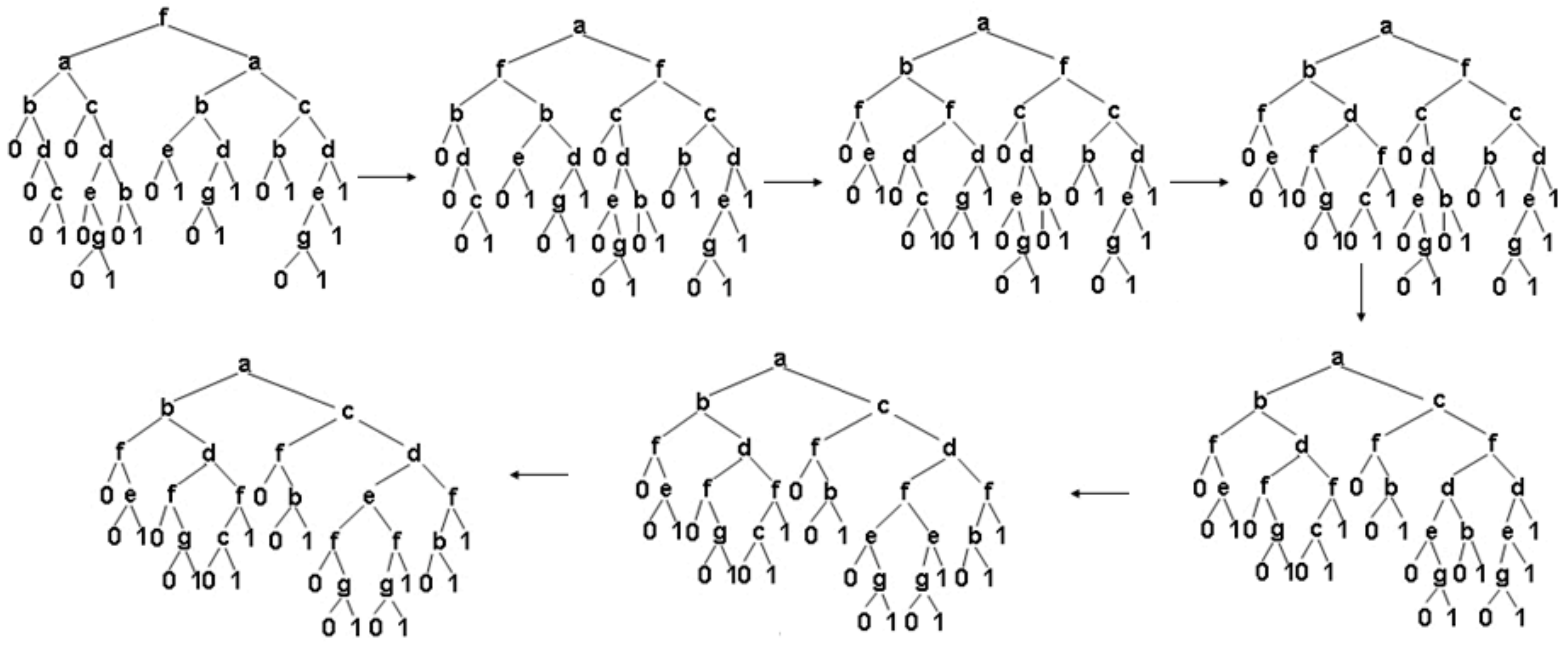
  - –*Irreducible Trees:* A tree will be called irreducible, if all the trees belonging to its equivalence class are complete

Current Work

- Canonical Form Representation
  - We chose a lexicographic representation of the equivalence class
    - "Pull-up" the lexicographically smallest sensor as the root node and recursively repeat the procedure in the left and right subtrees
  - A canonical form representation of an equivalence class enables us to "shrink" the tree space
  - Every tree is first converted to its canonical form, before checking for its cost, therefore checking the cost of only one tree from an equivalence class is sufficient

Current Work

- Canonical Form Representation: Example

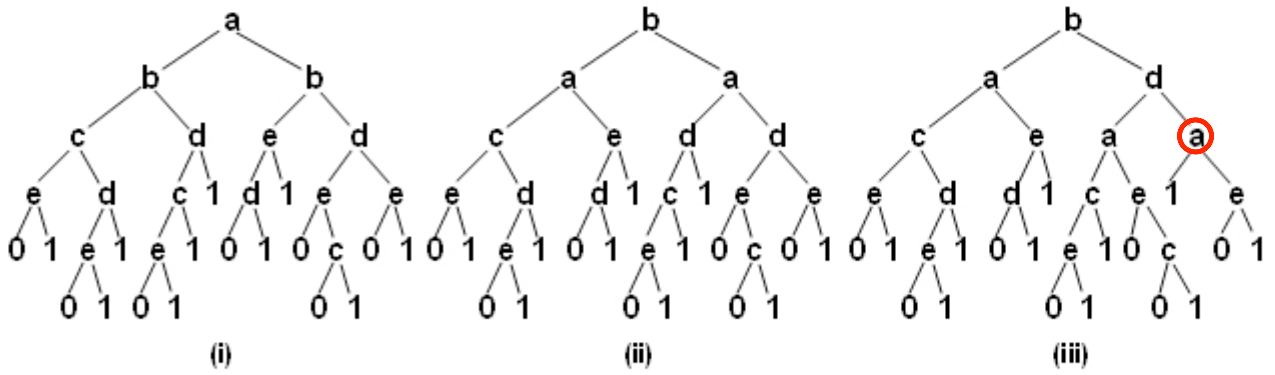


Current Work

- Revisiting Completeness:
  1. At any node in a tree, the left and right subtrees should not be cost-equivalent
  2. At any node in a tree, the left and right subtrees should not have identical Boolean function
- 2 covers 1, therefore…
- *Equi-complete BDT:* A binary decision tree will be called equi-complete if every sensor occurs at least once in the tree and, at any non-leaf node, the left and right subtrees do not correspond to same Boolean function.

Current Work

- Revisiting Monotonicity:
  1. A cost-equivalent tree of a monotonic tree can be non-monotonic ('0' as right leaf, '1' as left leaf or both).
- *Equi-monotonic BDT:* A binary decision tree will be called equi-monotonic, if all the trees belonging to its equivalence class are monotonic.



Discussion

1. The exhaustive search method, for finding the optimum thresholds for a given tree, become practically infeasible beyond a very small number of sensors.
2. The threshold optimization technique discussed in our work provide faster and better ways to calculate the optimal total cost of a tree.
3. The exhaustive search method, for finding the cheapest tree in the entire space of trees is also hard to extend beyond a very small number of sensors.
4. We described a couple of efficient search methods to find the best trees in the CM tree space

Discussion

5. Expanding the ideas of monotonicity and completeness from BDFs to BDTs is reasonable because:
   - certain trees obtained from incomplete/ non-monotonic BDFs are potentially valid BDTs and,
   - it facilitates tree search algorithms
6. We proved that the proposed CM tree space is irreducible under the defined neighborhood operations.
7. We discussed the ideas of tree equivalence and tree reduction that help us “shrink” the tree space
8. We describe way to represent an equivalence class with a unique, canonical form.

Future Work

- A more basic and rigorous analysis of monotonicity is required
- Different instances of a sensor in a tree can be set to different thresholds for optimum cost
- Sensor models, other than the one we use could be tried

Acknowledgements

- Dr. Fred Roberts
- DIMACS, NSF and ONR
- Dr. Peter Meer and Oncel Tuzel
- Dr. Endre Boros