# Cognitive Radio Resource Management based on Machine Learning

Raviraj S. Adve

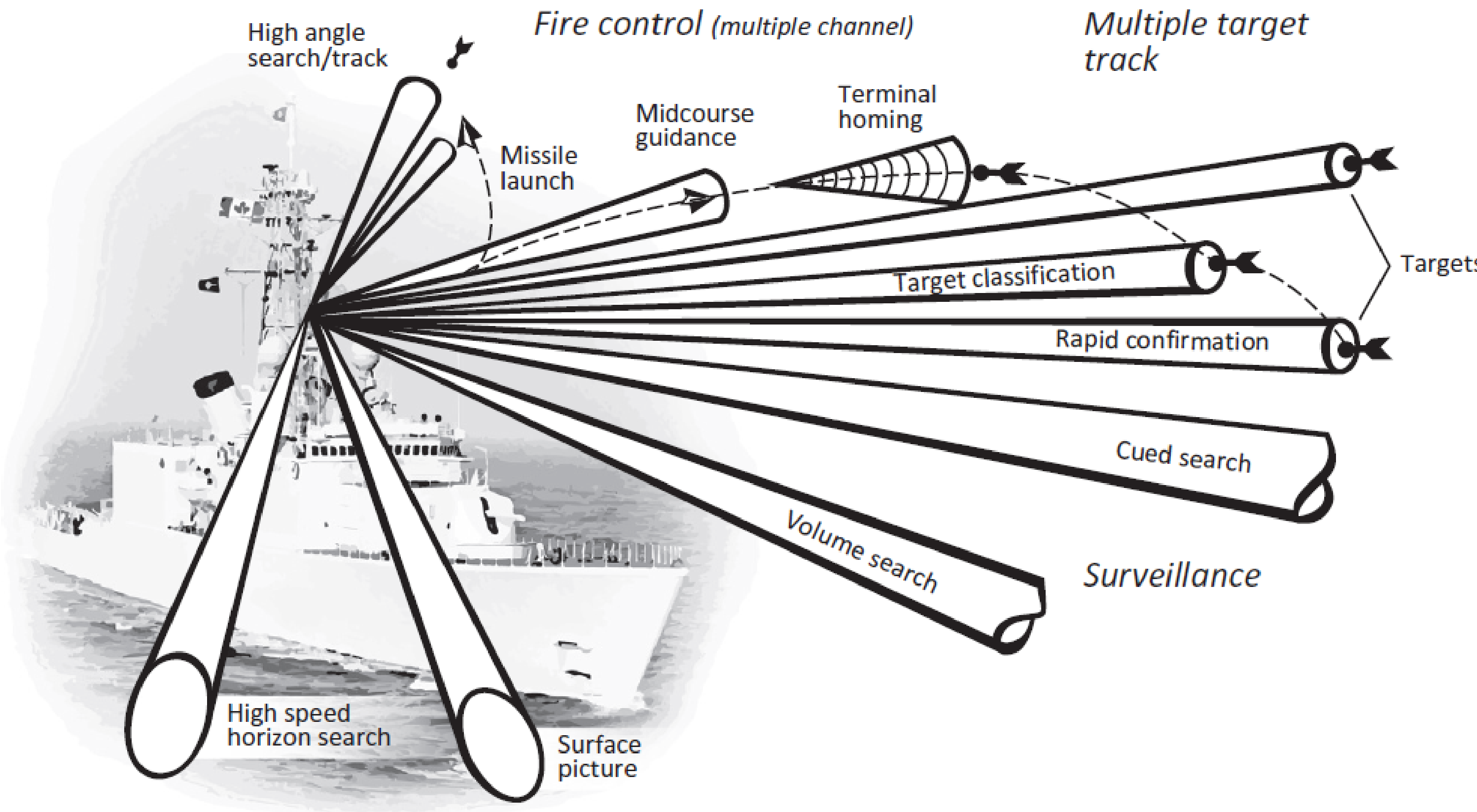
Department of Electrical and Computer Engineering

e-mail: rsadve@comm.utoronto.ca

UNIVERSITY OF TORONTO

NATO SET-241, 2017

# Multifunction Radar




[Image taken from “Adaptive Radar Resource Management” book by P. Moo and Z. Ding, 2015]

# Radar Resource Management

- Radar resource management (RRM) considers
  - Parameter selection (e.g. pulse duration, PRF, number of pulses, task priorities, …)
  - Scheduling the tasks on the radar timeline(s).

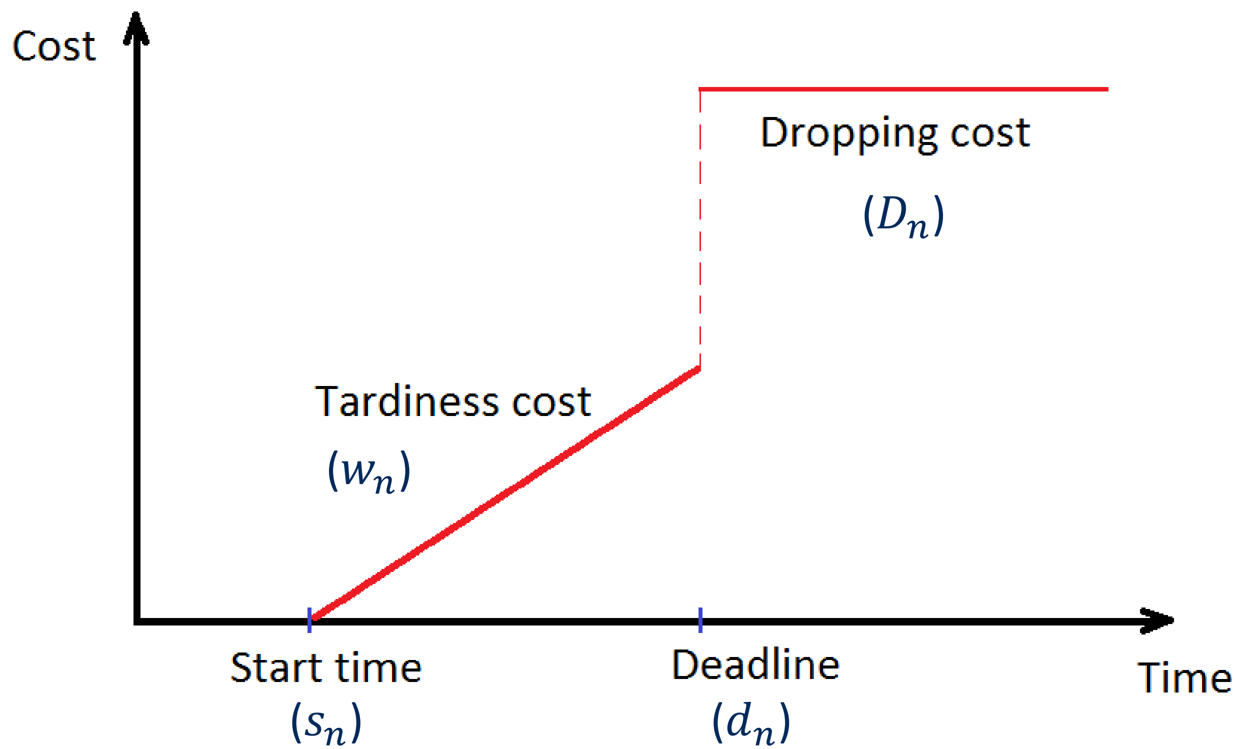
- As a common theme in most of the previous works, RRM can be split into three stages
  1) Task parameter selection
  2) Task down-selection
     - In case of an overloaded system, some tasks may need to be **dropped**.
  3) Task scheduling
     - The time that each task starts to execute and also the channel on which the task is performed are determined.

# Task Parameters

- For each task, there is a starting time ($s_n$) after which the task can be scheduled, and there is also a deadline ($d_n$) after which the task must be dropped.
- Each task has a dropping cost ($D_n$).
- Each task has a length ($\ell_n$), (dwell time).
- For each task (if executed), there is a tardiness cost which is assumed to be linearly proportional ($w_n$) to the difference between the execution time ($e_n$) and the starting time ($s_n$) ($s_n \leq e_n \leq d_n$).

# Task Cost Function

- If task $n$ is dropped $x_n = 0$, else $x_n = 1$.
- $c_n = x_n w_n(e_n - s_n) + (1 - x_n)D_n$
  - Model is for illustration; any reasonable cost function can be used
- $s_n \leq e_n \leq d_n$ if $x_n = 1$



# Problem Formulation

- There are $N$ tasks with given operational parameters.
- There are $K$ channels.
- Objective is to minimize the total cost

$$\min\left\{\sum_{n=1}^{N} x_n w_n (e_n - s_n) + (1 - x_n) D_n\right\}$$

such that $x_n \in \{0, 1\}$

$s_n \leq e_n \leq d_n$ if $x_n = 1$

No tasks overlap in time

- This is an NP-hard problem.

# Optimal Solution: Branch and Bound (B&B)

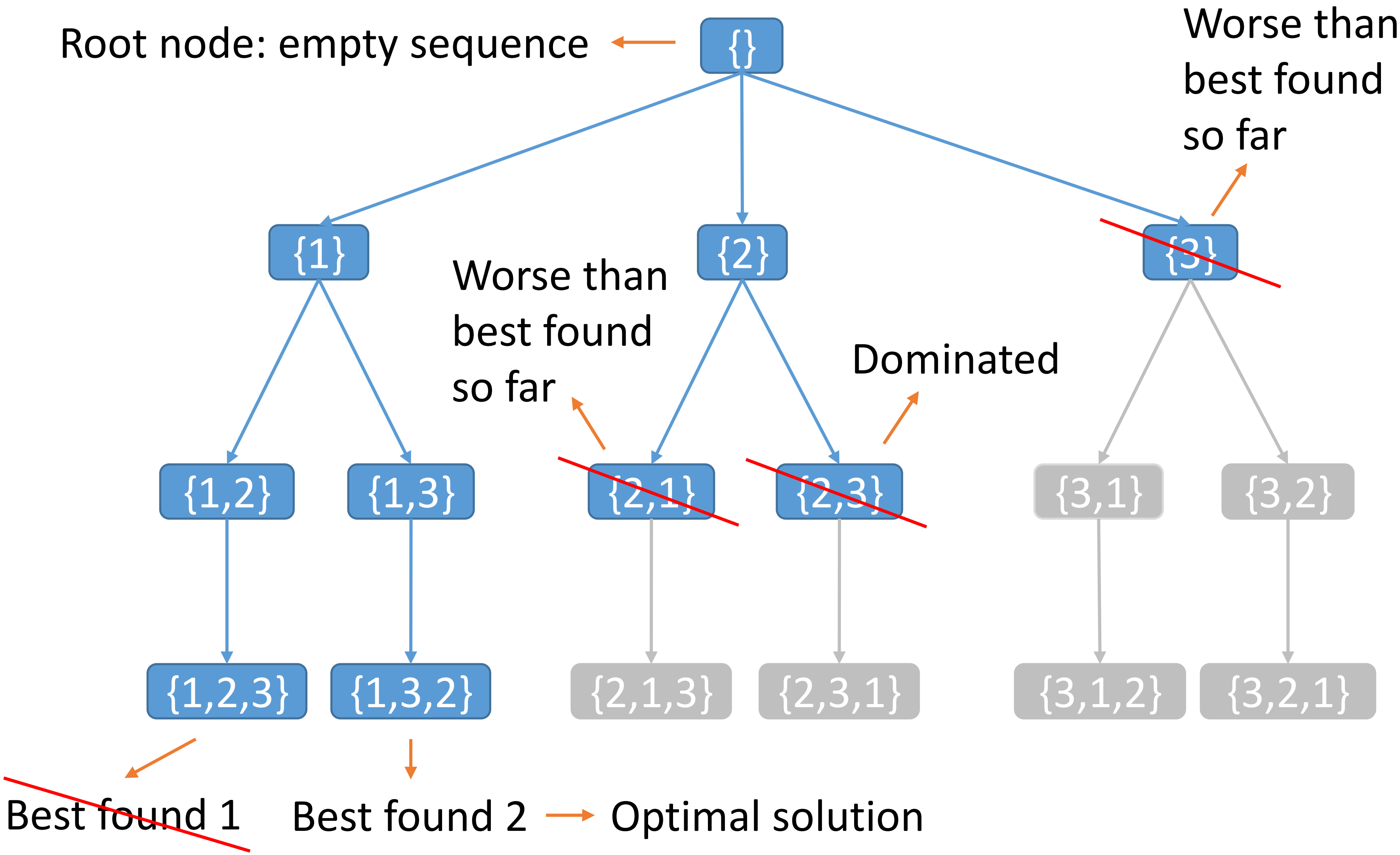
- This procedure implicitly enumerates all possible solutions on a search tree.
  - Each node of the tree is a partial schedule

- Rules (e.g. bounds, dominance rules, …) are used to prune off nodes that are provably suboptimal (i.e., bounding).
  - Example of bound: track cost of “best- known-solution-so-far”
  - Example of dominance rule: check whether an unscheduled task can be fit in a time gap of the partial schedule

- Once the entire tree has been explored, the best solution found in the search is returned.

  - This is the optimal solution

# Branch and Bound: Search Tree



# Simulation parameters (B&B v/s heuristics)

- Setup

  Number of channels (timelines): 4

  Timeline window: 100 sec

  Task starting time: $\mathcal{U}(0, 100)$ sec

  Task interval (deadline – starting time): $\mathcal{U}(2, 12)$ sec

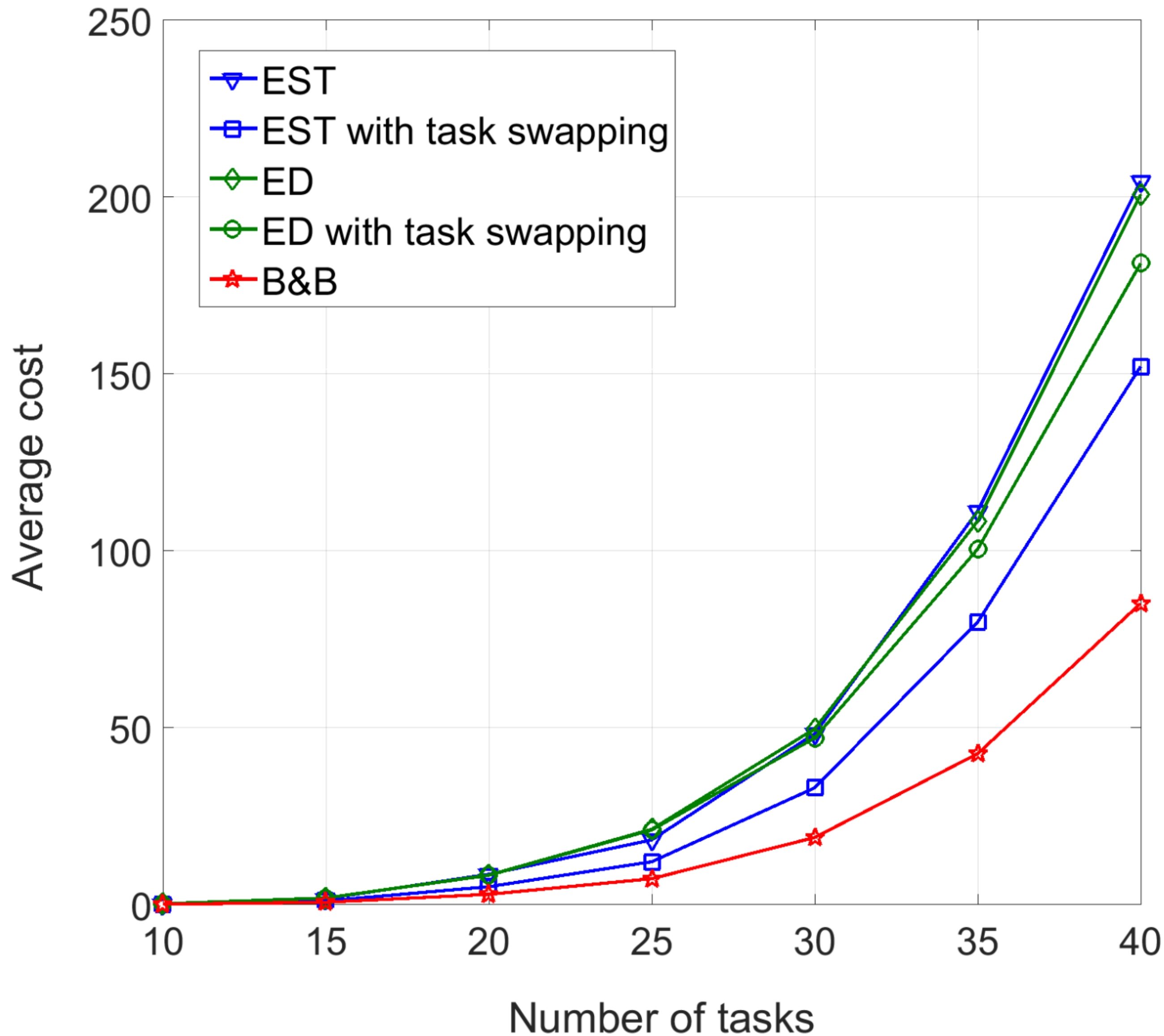
  Task length: $\mathcal{U}(2, 11)$ sec

  Dropping cost: $\mathcal{U}(100, 500)$

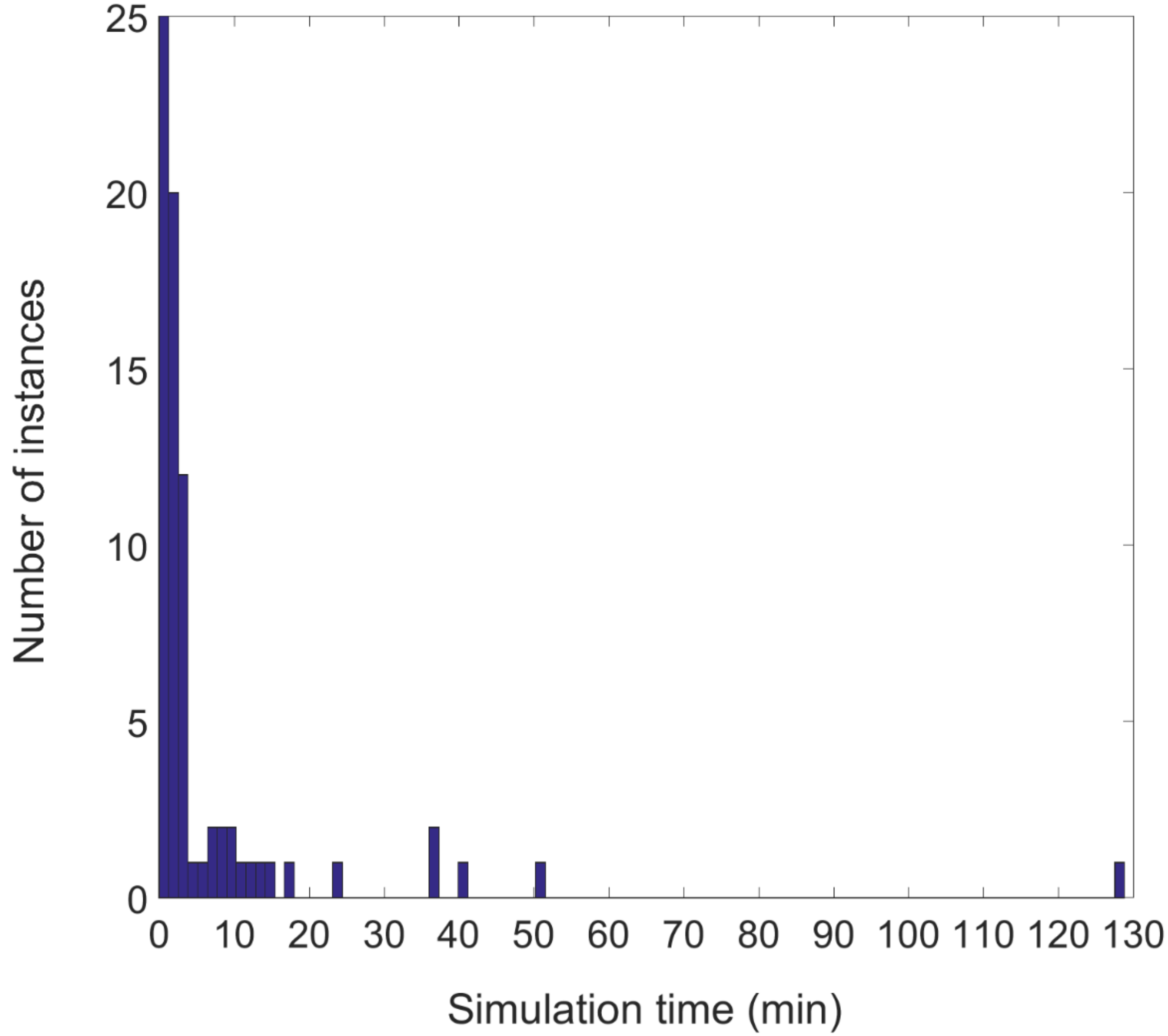
  Tardiness cost slope: $\mathcal{U}(1, 5)$

  Number of Monte Carlo trials: 1000

# B&B v/s heuristics: Average Cost - Number of Tasks

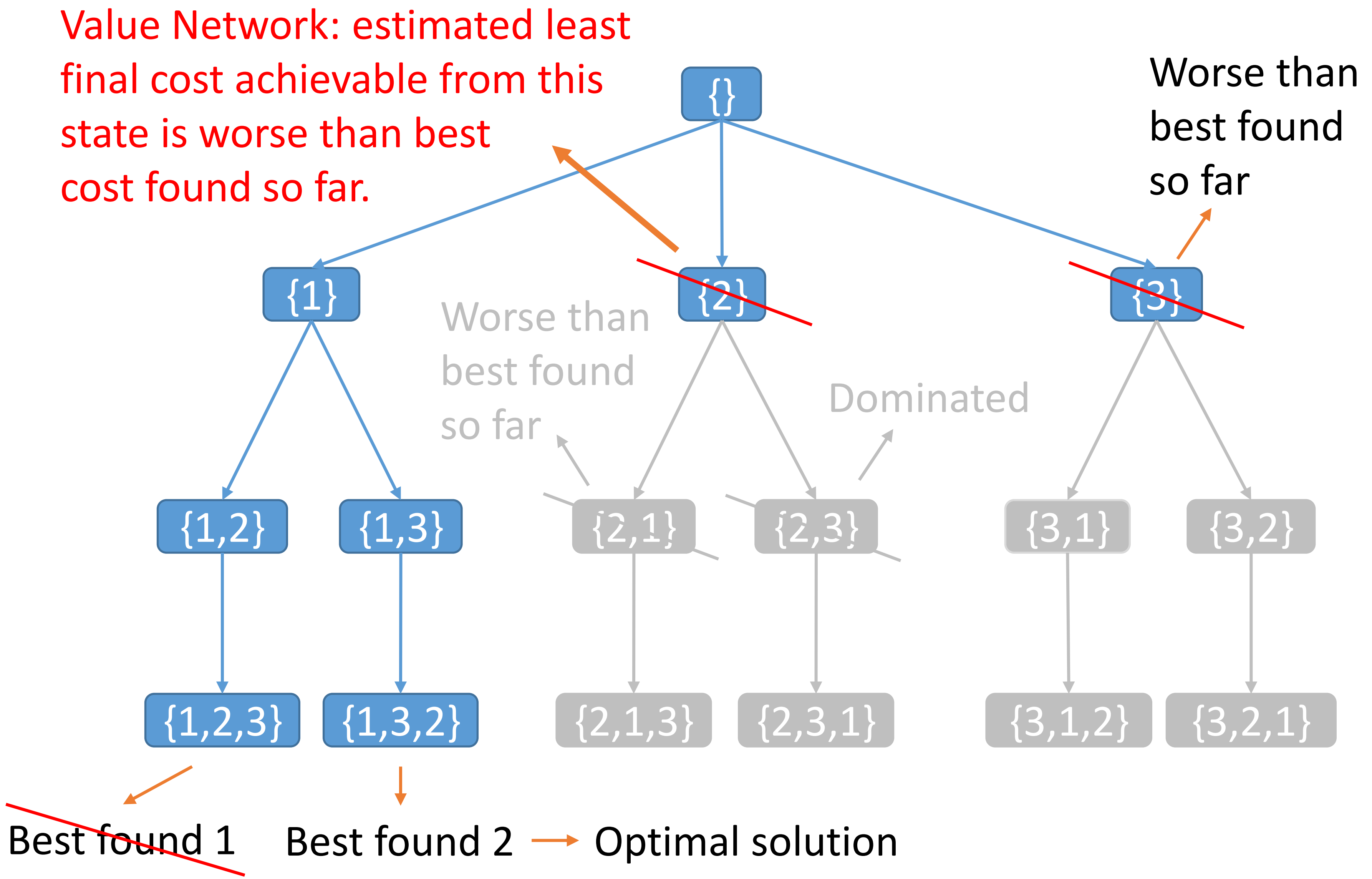


# Simulation Time is Heavy-Tailed




- 949 of 1000 realizations in first bin

# Branch and Bound Method Enhanced with Neural Networks



# Neural Network Architecture: Value Network

- The optimal value function $v^*(s)$ determines the least overall cost (of a complete schedule) that can be obtained starting from state $s$ (a partial schedule).
  - The depth of the search may be reduced by truncating the tree at state $s$ and replacing the subtree below $s$ by an approximate value function $v(s) \approx v^*(s)$.
  - A value network is used to produce the approximate value function.
  - The  weights of the network are obtained by regression on the state-outcome pairs $(s, v^*(s))$ obtained from training data.

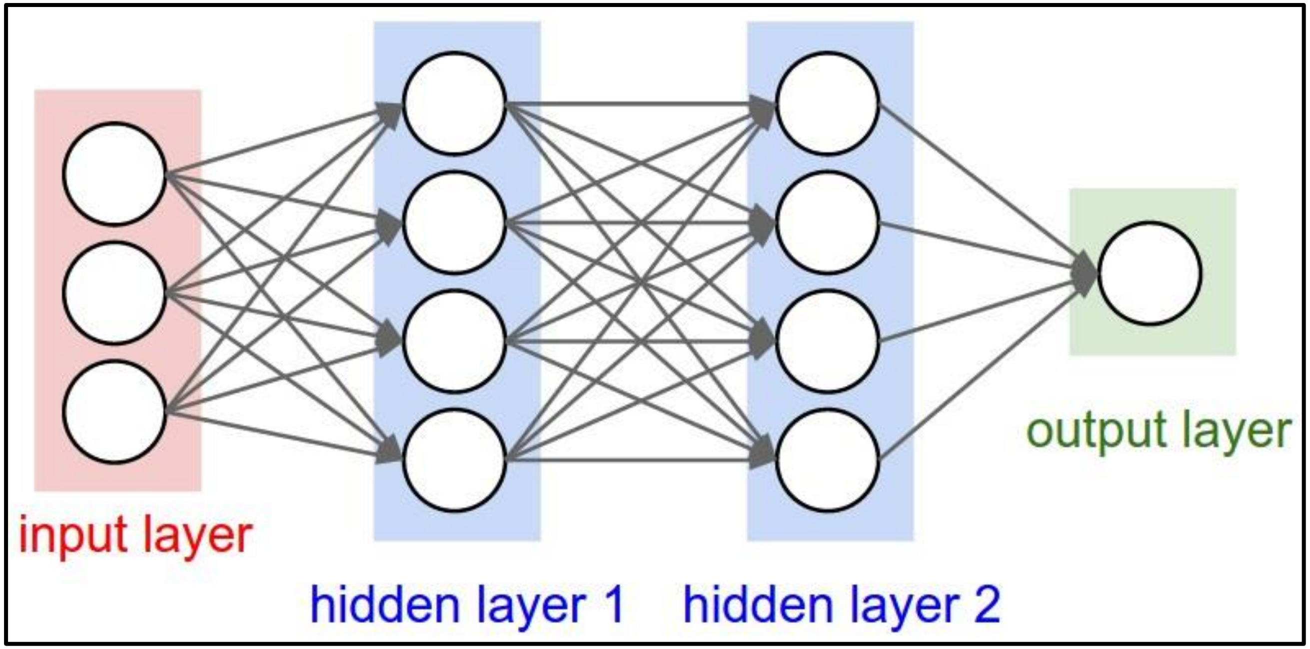
# Neural Network Architecture: State Definition

- A state, $s$, is a representation of a partial schedule (node). It includes the initial parameters of the tasks as well as their state (scheduled, dropped, or not scheduled) in the corresponding partial schedule.

| Input feature | Description |
|---|---|
| 1, 2, 3 | status (1 0 0: scheduled, 0 1 0: dropped, 0 0 1: unscheduled) |
| 4 | start time |
| 5 | end time |
| 6 | task length |
| 7 | execution time |
| 8 | tardiness coefficient |
| 9 | dropping cost coefficient |
| 10 | tardiness cost or drop cost (if scheduled or dropped) |
| 11, 12, 13, 14 | assigned timeline (one-hot encoded) |

# Value Network Implementation

- The final output of the network is a scalar number representing the estimated least final cost of the partial schedule input.




- In order to increase the robustness of the algorithm to estimation errors, the output of the network is divided by a fixed scalar $\beta \geq 1$.

# Simulation Results (same setup as before)

| | EST | EST+SW | ED | ED+SW | B&B | $\beta$ = 1 | $\beta$ = 1.5 | $\beta$ = 2 |
|---|---|---|---|---|---|---|---|---|
| Average Cost | 93.2 | 68.3 | 115.8 | 101.5 | 38.6 | 45.7 | 44.5 | 42.9 |
| Average # of visited nodes | - | - | - | - | 13134 | 448 | 1466 | 2460 |

# Simulation Results (same setup as before)

- (Cost, Number of visited nodes) for some random instances

| | B&B | $\beta$ = 1 | $\beta$ = 1.5 | $\beta$ = 2 | EST+SW |
|---|---|---|---|---|---|
| **Sample 1** | (54.9, 1670) | (63.1, 36) | (63.1, 52) | (63.1, 211) | 63.1 |
| **Sample 2** | (73.6, 554298) | (96.3, 110) | (96.2, 1512) | (78.9, 54027) | 100.0 |
| **Sample 3** | (221.9, 54514) | (221.9, 23610) | (221.9, 46626) | (221.9, 52705) | 328.4 |
| **Sample 4** | (47.5, 6998) | (49.4, 173) | (49.4, 196) | (47.5, 206) | 49.7 |
| **Sample 5** | (30.1, 2017) | (39.1, 105) | (39.1, 131) | (39.1, 153) | 39.1 |

## Concluding Remarks

- Heuristic methods have low complexity, but unsatisfactory performance.
- B&B algorithm finds the optimal solution, but has high complexity.
- Neural networks can be used to evaluate the importance of each node in the search tree, and eliminate nodes which are unlikely to result in the optimal solution.
- Solutions found offline by the B&B method can be used as labeled data for supervised training of the neural networks.

# Questions

# Value Network Implementation Details

- Convolutional filters have a width of 7 (looking at the features of 7 consecutive tasks at each stride).
- 64 filters are used at each layer (the output of each convolutional layer has 64 features).
- The first fully connected layer has 512 hidden units, and the second fully connected layer has 128 hidden units. The last layer has one scalar output.
- The network is trained using 90000 samples obtained from the branch-and-bound method.
- The weights are obtained by minimizing the L2-loss using the Adaptive Moment Estimation (Adam) optimization method.
- We use 100000 steps of the Random Reshuffling (RR) method with mini-batches of size 100.

# Optimal Solution: Branch and Bound (B&B)

**Initialization**

$UB \leftarrow \infty$

Let $\mathbf{T}$ be an empty sequence

$\mathbf{PF} \leftarrow$ {all tasks}

$\mathbf{NS} \leftarrow \{\}$; $\mathbf{DR} \leftarrow \{\}$

Push $(\mathbf{T}, \mathbf{PF}, \mathbf{NS}, \mathbf{DR})$ tuple on STACK.

**while** STACK is not empty

Let $(\mathbf{T}, \mathbf{PF}, \mathbf{NS}, \mathbf{DR})$ be the tuple on top of STACK.

**if** $\mathbf{PF} \neq \{\}$

Let $j \in \mathbf{PF}$

$\mathbf{PF} \leftarrow \mathbf{PF} \setminus j$

$\mathbf{T}' \leftarrow \mathbf{T}|j$; $\mathbf{PF}' \leftarrow \mathbf{PF} \cup \mathbf{NS}$; $\mathbf{NS}' \leftarrow \{\}$; $\mathbf{DR}' \leftarrow \mathbf{DR}$ → Set up a new node

$\mathbf{NS} \leftarrow \mathbf{NS} \cup j$

**if** $\mathbf{T}'$ follows the start-times dominance rule → Dominance rule

Move any task whose deadline has passed on all timelines from $\mathbf{PF}'$ to $\mathbf{DR}'$. → Delete tasks with expired deadlines

$C' \leftarrow \text{TardinessCost}(\mathbf{T}') + \text{DroppingCost}(\mathbf{DR}')$

**if** ($\mathbf{T}'$ is active)

and ($\mathbf{T}'$ is LOWS-active) → Dominance rules

and $(C' < UB)$

Push $(\mathbf{T}', \mathbf{PF}', \mathbf{NS}', \mathbf{DR}')$ tuple on STACK. → Add new node to tree for further investigation

**else**

$C \leftarrow \text{TardinessCost}(\mathbf{T}) + \text{DroppingCost}(\mathbf{DR})$

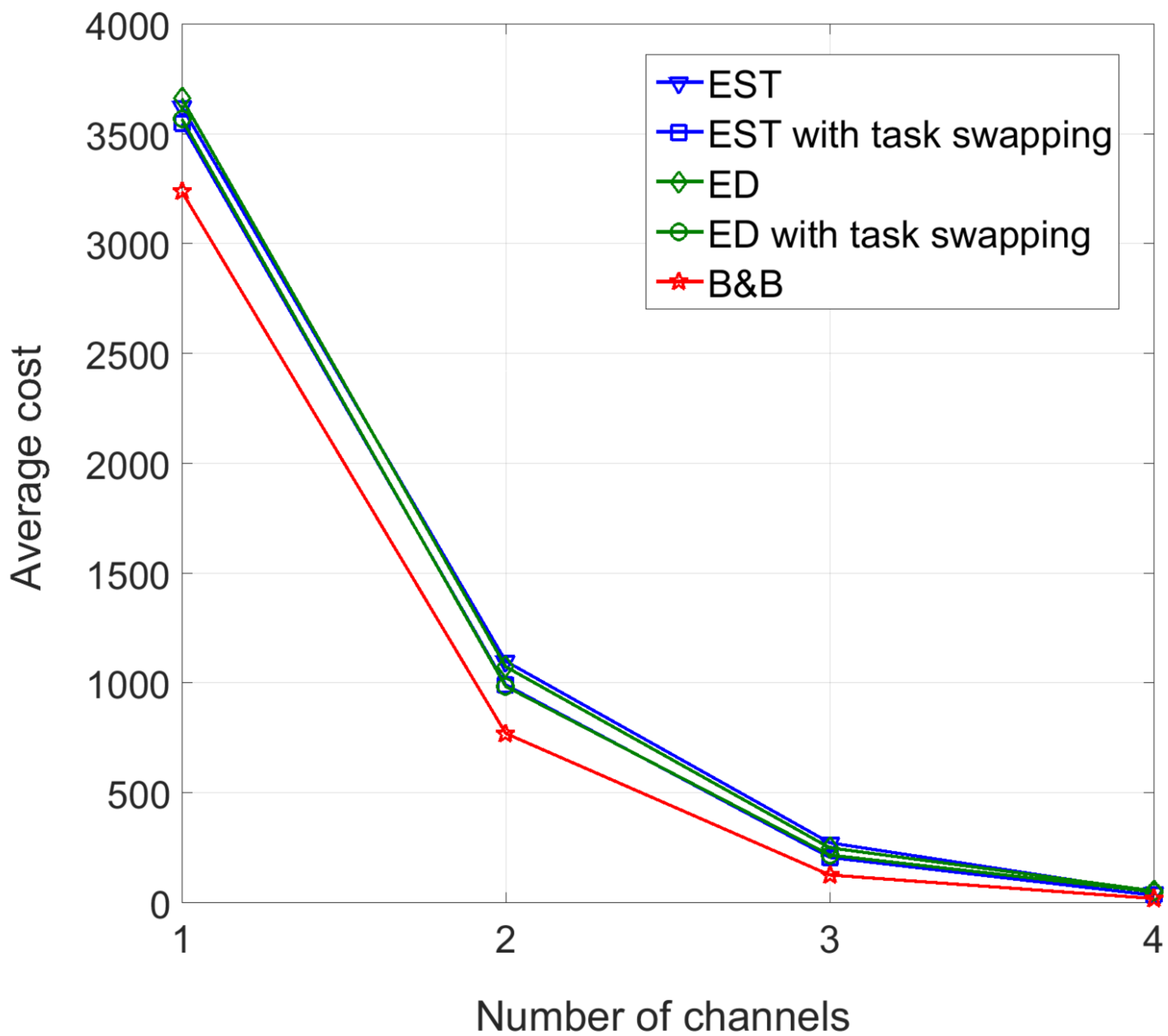
**if** $(\mathbf{NS} = \{\})$ and $(C < UB)$

$UB \leftarrow C$

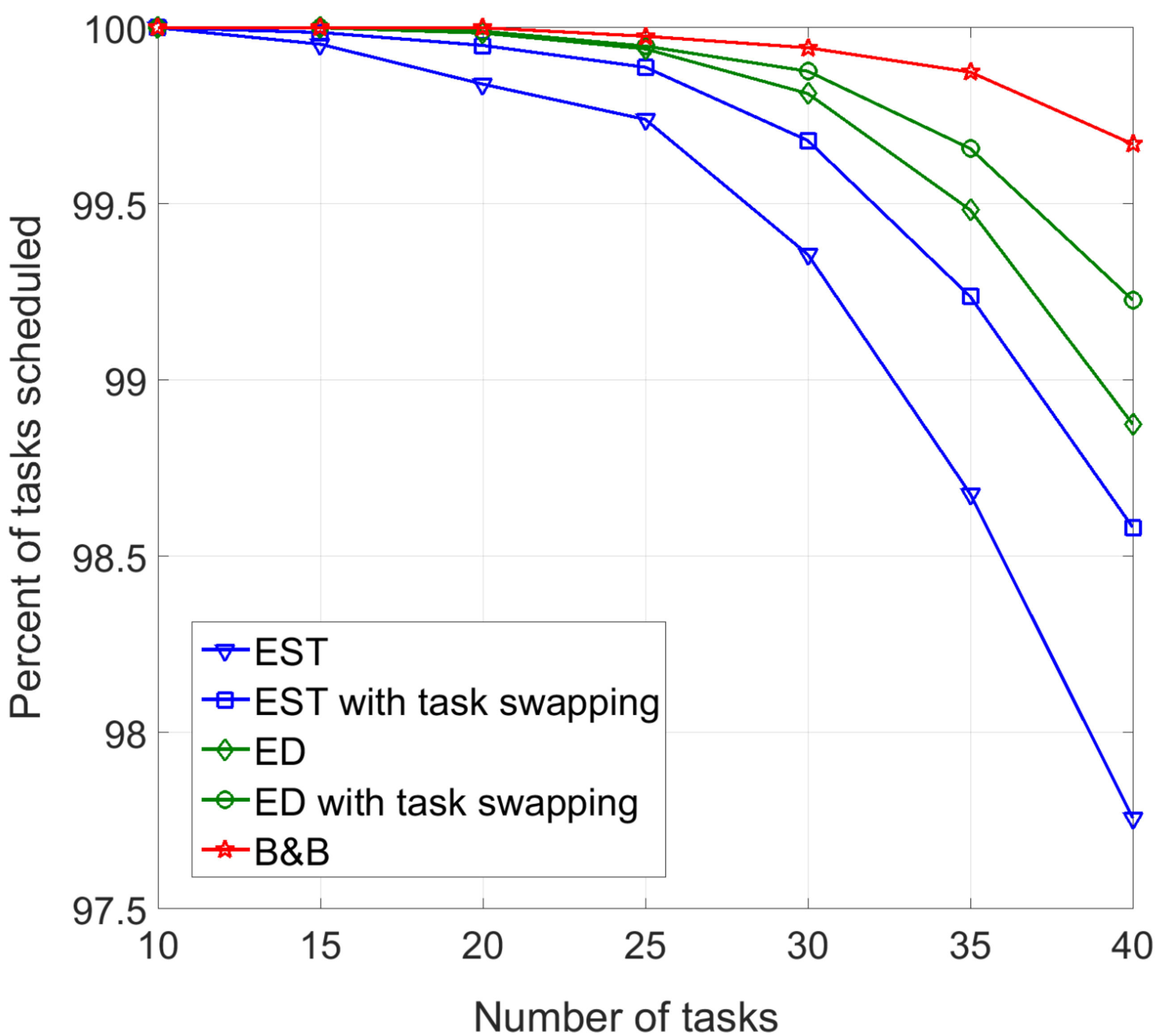
$T^* \leftarrow T$

Remove $(\mathbf{T}, \mathbf{PF}, \mathbf{NS}, \mathbf{DR})$ tuple from STACK. → Remove completely investigated node

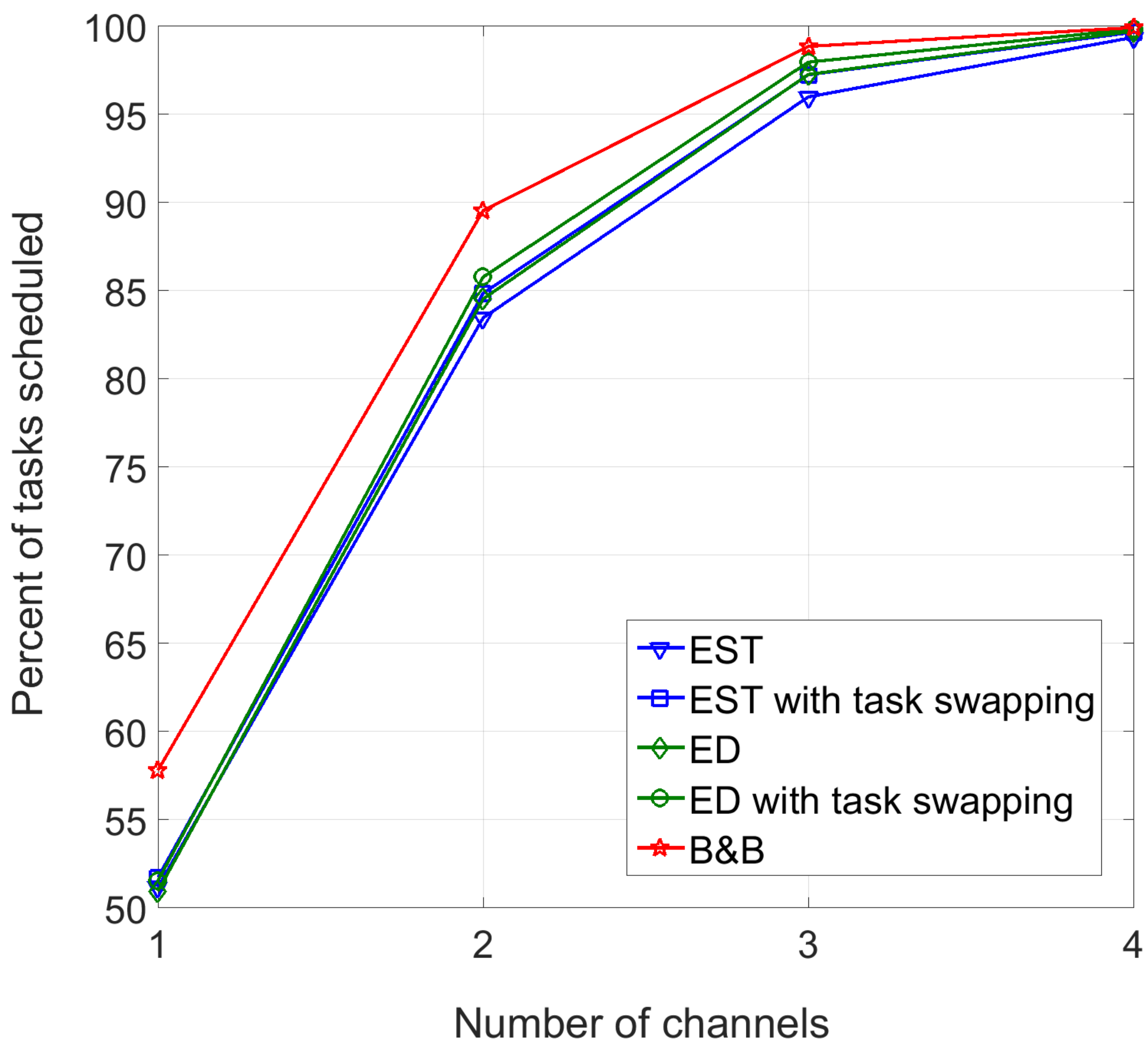
# B&B v/s heuristics: Average Cost - Number of Channels



# B&B v/s heuristics: % Tasks Scheduled - Number of Tasks

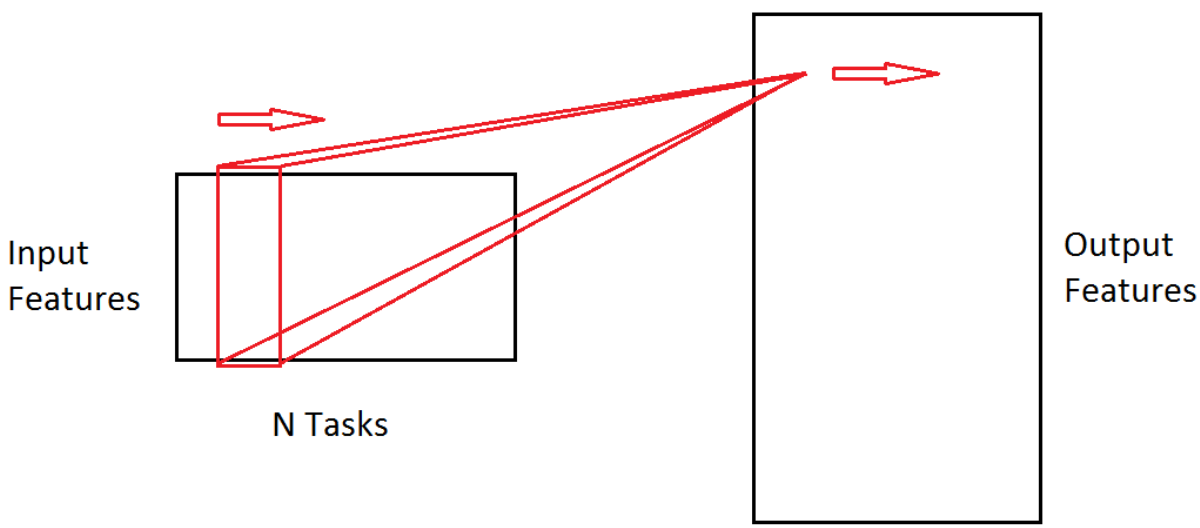


# B&B v/s heuristics: % Tasks Scheduled - Number of Channels



# Value Network Implementation

- The value network is implemented using three convolutional layers and three fully connected layers.
- The input to the network is a state (representing a partial schedule) formatted as a matrix with each column corresponding to a task and each row representing a feature.
- The coefficients of the filters are obtained using supervised training.




Convolutional Layer