Multiple Hypothesis Tracking (MHT)

- MHT is the name given to a type of TT algorithms that keep at each time step multiple hypotheses about the past and current association uncertainties.
- The first structured MHT (which we will call “conceptual MHT”) was described by D. B. Reid in 1979.
- These algorithms do not use a separate track initialization procedure and hence track initiation is integrated into the algorithm.
- Between consecutive time instants, if the implemented MHT algorithm keeps hypotheses tracks
  the corresponding MHT implementation is called hypothesis based track based respectively.

Conceptual MHT

- It was first described by D. B. Reid in 1979.
- It is an hypothesis based brute force implementation i.e., between consecutive time instants, different hypotheses \( \Theta_{k-1}^{i} \) about the past are kept in the memory.
- The idea is to generate all possible hypotheses and then to depend on pruning of these hypotheses, otherwise, it has a combinatoric explosion in the number of hypotheses.
- Uses techniques such as
  - Clustering;
  - Pruning of low probability hypotheses;
  - N-scan pruning;
  - Combining similar hypotheses
  to reduce the number of hypotheses.
Each of the hypotheses \( \{ \Theta_{k-1}^i \}_{i=1}^{N_h} \) kept about the past are characterized by their assumed number of targets (tracks) and their corresponding sufficient statistics.

\[
\Theta_{k-1}^1, P(\Theta_{k-1}^1) \quad \Theta_{k-1}^2, P(\Theta_{k-1}^2)
\]

**Hypothesis Generation:** Form \( \Theta_k^i \equiv \{ \theta_k, \Theta_{k-1}^i \} \)

<table>
<thead>
<tr>
<th>Hypothesis</th>
<th>( y_k^1 )</th>
<th>( y_k^2 )</th>
<th>( y_k^3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T_1 )</td>
<td>FA</td>
<td>FA</td>
<td>FA</td>
</tr>
<tr>
<td>( T_2 )</td>
<td>FA</td>
<td>NT</td>
<td>FA</td>
</tr>
<tr>
<td>( T_3 )</td>
<td>NT</td>
<td>FA</td>
<td>FA</td>
</tr>
</tbody>
</table>

**Hypothesis Probability:** Let \( \Theta_k^i \equiv \{ \theta_k, \Theta_{k-1}^i \} \)

\[
P(\Theta_k^i|y_{0:k}) \propto P(y_k|\Theta_k^i, y_{0:k-1})P(\theta_k|\Theta_{k-1}^i, y_{0:k-1})P(\Theta_{k-1}^i|y_{0:k-1})
\]

\[
\times \beta_{FA}^m \beta_{NT}^m \prod_{j \in \mathcal{J}_D^i} P_{D,k|k-1}(y_k^{\Theta_{k-1}^i(j)}) \prod_{j \in \mathcal{J}_{ND}^i} (1 - P_{D,k|k-1}^i) P_{G}^i P(\Theta_{k-1}^i|y_{0:k-1})
\]

where we have used the “Fundamental Theorem of TT” introduced at the last lecture.

**Important remark:** Note that the sets \( \mathcal{J}_D^i \) and \( \mathcal{J}_{ND}^i \) are dependent on the previous hypothesis \( \Theta_{k-1}^i \) because the number of targets and estimates of the targets can be different for each different previous hypothesis.
**Conceptual MHT**

**New Set of Measurements** \( \{ y_i^k \}^{m_k}_{i=1} \)

**Set of Hypotheses** \( \{ \Theta_i^{k-1} \}^{N_h}_{i=1} \)

**Generate New Hypotheses** \( \{ \Theta_i^{k} \}^{N_h}_{i=1} \)

**Calculate Hyp. Probabilities** \( \{ P(\Theta_i^{k}) \}^{N_h}_{i=1} \)

**Reduce Number of Hypotheses** \( \Theta_i^{k-z-1} \)

**User Presentation Logic**

**Computation and hypothesis number reduction techniques**
- Clustering.
- Pruning of low probability hypotheses.
- N-scan pruning
- Combining similar hypotheses

---

**Clustering** is the processing of hypotheses about the groups of targets (tracks) that do not share measurements (in the gates) separately.

\( \Theta_1^{k-1} \)

\( \Theta_2^{k-1} \)

**Cluster-1**

**Cluster-2**

---

**Clustering** is the processing of hypotheses about the groups of targets (tracks) that do not share measurements (in the gates) separately.

\( \Theta_1^{k-1} \)

\( \Theta_2^{k-1} \)

**Cluster-1**

**Cluster-2**
Cluster Management:
- When targets get closer
  - If at any time instant a measurement falls inside the gates of two tracks that have different clusters, the corresponding two clusters must be merged into a super-cluster.
  - The hypotheses for each cluster are combined into super-hypotheses.
- When targets separate
  - If a group of tracks in a cluster did not share measurements (inside their gates) with the rest of the tracks in the cluster for some specified time period, the cluster can be divided into two smaller clusters.
  - Hypotheses for the cluster are also divided into smaller hypotheses corresponding to two smaller clusters.

Process Each Cluster Separately: Form $\Theta^\ell_k \triangleq \{\theta_k, \Theta^{\ell-1}_k\}$ for each cluster as if the other clusters do not exist.

Before clustering $\Theta^1_{k-1}, P(\Theta^1_{k-1})$

Cluster-1 $\Theta^2_{k-1}, P(\Theta^2_{k-1})$

Cluster-2 $\Theta^1_{k-1}, P(\Theta^1_{k-1})$

Pruning low probability hypotheses: For each cluster
- One can delete hypotheses that has probability less than a threshold (e.g., 0.001).
  
Deletion Condition: $P(\Theta^i_k) < \gamma_p$
- Another idea is to sum the probabilities of the hypotheses in descending order and discard the ones over a predetermined probability mass (e.g., 0.99).
  
Deletion Condition: $\sum_{\ell=1}^{i} P(\Theta^\ell_k) > \gamma_c$

where the ordering $\ell = 1, \ldots, N_k$ is descending in probabilities.
**N-scan Pruning:**
- This scheme assumes that any uncertainty at time \( k - N \) is perfectly resolved by the time \( k \) for all \( k \).
- It is a general commonsense to choose \( N \geq 5 \).
- For this purpose, \( N \) last ancestors of each created hypothesis is kept in memory.
Conceptual MHT

**N-scan Pruning:**
- This scheme assumes that any uncertainty at time \( k - N \) is perfectly resolved by the time \( k \) for all \( k \).
- It is a general commonsense to choose \( N \geq 5 \).
- For this purpose, \( N \) last ancestors of each created hypothesis is kept in memory.

**Hypothesis Merging:** It is also suggested in Reid’s original paper to check for every hypothesis pair that
- two hypotheses have same number of targets (tracks)
- the estimates of the tracks are close to the corresponding ones in the other hypothesis.
If these conditions are satisfied
- the two hypotheses are merged into a single hypothesis
- the resulting single hypothesis is assigned the probability that is obtained by summing the probabilities of the individual hypotheses.

Conceptual MHT is attractive in the sense that each hypothesis saved between time instants is an alternative representation of reality and can be interpreted easily.
- However, except for toy examples, generating all possible hypotheses and then discarding most of them was deemed inefficient because we are basically spending computation for hypotheses that we, at the end, throw away.
- Moreover, some hypotheses keep different combinations of exactly the same tracks. Hence the number of actual tracks we are considering is much less than the number of hypotheses.
- For these reasons, an alternative track based implementation was adopted until an efficient way to implement a hypothesis based MHT was found in [Cox (1996)] in 1996.
Reminder of Assignment Problem

Let \( \Theta_{k}^{i} \triangleq \{ \theta_{k}, \Theta_{k-1}^{i} \} \).

\[
P(\Theta_{k}^{i}|y_{0:k}) \propto p(y_{k}|\Theta_{k}^{i}, y_{0:k-1})P(\theta_{k}|\Theta_{k-1}^{i}, y_{0:k-1})P(\Theta_{k-1}^{i}|y_{0:k-1})
\]

\[
\propto \beta_{FA}^{mFA} \beta_{NT}^{mNT} \prod_{j \in J_{D}} P_{D}^{j} P_{G}^{j} \prod_{k \in J_{N}} \frac{P_{D}^{j} P_{G}^{j}}{1 - P_{D}^{j} P_{G}^{j}} \prod_{j \in J_{N}} P(\Theta_{k-1}^{i}|y_{0:k-1})
\]

Divide and multiply the right hand side by

\[
C_{i} \triangleq \prod_{j=1}^{n_{T}} (1 - P_{D}^{j} P_{G}^{j}) = \prod_{j \in J_{D}} (1 - P_{D}^{j} P_{G}^{j}) \prod_{j \in J_{N}} (1 - P_{D}^{j} P_{G}^{j})
\]

Find the N-best Solutions to an Assignment Problem

- Given an assignment matrix \( A_{i} \), we can find the best solution with Auction or similar algorithms in polynomial time.
- People had considered the generalization of this problem to N-best solutions.
- The key point is to express finding N-best solutions problem into a number of best solution assignment problems.
- Then for each of the best solution assignment problems Auction algorithm can be used.
Hypothesis Based Implementation

Find the N-best Solutions to an Assignment Problem

Murty’s Algorithm

Given the assignment matrix \( A_i \),
- Find the best solution using Auction algorithm.
- 2nd best solution:
  - Express the 2nd best solution as the solution of a number of best solution assignment problems.
  - Find the solution to each of these problems by Auction.
  - The solution giving the maximum reward (minimum cost) is the second best solution.
- Repeat the procedure if further solutions are required.

Murty’s Algorithm


Hypothesis Based Implementation

- **Our aim:** Given the previous hypotheses \( \{ \Theta_{k-1}^i \}_{i=1}^{N_h} \) and current measurements \( \{ y_{i}^k \}_{i=1}^{m_k} \), we would like to find the best \( N_h \) current hypotheses \( \{ \Theta_{k}^j \}_{j=1}^{N_h} \) without generating all the hypotheses.
- **Reminder of Hypothesis Probability**

\[
P(\Theta_{k}^j | y_{0:k}) \propto \beta_{F}^{P} A^m_{NT} \prod_{j \in J_k^f} \frac{P_{D}^j p_{D}^j p_{k-1}^j (y_{j}^{-1}(j))}{1 - P_{D}^j p_{G}^j} C_{t} p(\Theta_{k-1}^i | y_{0:k-1})
\]

Maximized by Assignment Problem

We would like to find \( \{ \Theta_{k}^j \}_{j=1}^{N_h} \) that maximizes \( P(\Theta_{k}^j | y_{0:k}) \).

This can be obtained in two steps:
- Obtain the solution from the assignment (Murty’s algorithm).
- Multiply the obtained quantity by previous hypothesis dependent terms.

Generating \( N_h \)-best Hypotheses

Given previous hypotheses \( \{ \Theta_{k-1}^i \}_{i=1}^{N_h} \), the corresponding hypothesis probabilities \( P(\Theta_{k-1}^i | y_{0:k-1}) \) \( \{ i=1 \}^{N_h} \) and current measurements \( \{ y_{i}^k \}_{i=1}^{m_k} \),

- Find assignment matrices \( \{ A_i \}_{i=1}^{N_h} \) for all previous hypotheses.
- Obtain the best hypotheses shown as \( \{ \Theta_{k}^i \}_{i=1}^{N_h} \) for each assignment matrix.
- Calculate the corresponding probabilities \( P(\Theta_{k}^i | y_{0:k}) \) \( \{ i=1 \}^{N_h} \).
- Order the obtained hypotheses according to their probabilities. Call the resulting ordered list as HYP-LIST and the corresponding list of probabilities as PROB-LIST.
- ...
Hypothesis Based Implementation

Generating $N_h$-best Hypotheses

- For $j = 2 : N_h$
  - Loop through the assignment matrices and find the $j$th best solutions $\{\Theta_i^j\}_{i=1}^{N_h}$ for each of them.
  - Calculate the probabilities $P(\Theta_i^j)$ corresponding to $\{\Theta_i^j\}_{i=1}^{N_h}$.
  - If $P(\Theta_i^j)$ is higher than the lowest probability in PROB-LIST, add $\Theta_i^j$ to HYP-LIST and the corresponding probability to PROB-LIST.
  - Discard the lowest probability hypothesis from HYP-LIST and its corresponding probability from PROB-LIST.
  - If $P(\Theta_i^j)$ is lower than the lowest probability in PROB-LIST discard $\Theta_i^j$ and never use $A_i$ again in subsequent recursions.
- The hypotheses in the HYP-LIST are the $N_h$ best hypotheses.

Track Based Implementation

Most of the time, hypotheses are composed of combinations of exactly same tracks. The number of tracks might be significantly lower than the number of hypotheses.

$$\Theta_{k-1}^1, P(\Theta_{k-1}^1) \quad \Theta_{k-1}^2, P(\Theta_{k-1}^2)$$

Instead of keeping $\Theta_{k-1}^1$ and $\Theta_{k-1}^2$ which would duplicate information of $T_1$, we can save $T_1$ and $T_2$ for the next time instant.

Track Based Implementation

- Tracks at time $k$ are shown by $\{T_i^k\}_{i=1}^{N_t}$
- Each track is kept with its Score shown as $Sc(T_i^k)$.
- Instead of a hypotheses tree, form a track tree.
- Delete low score tracks.

For reducing the number of tracks further and for user presentation, generation of hypotheses is still necessary. One advantage this time is that one can only use high score tracks for hypothesis generation.

For generating hypotheses, keeping the track compatibility information is necessary. One can keep a binary matrix as below.

<table>
<thead>
<tr>
<th>$T_1^k$</th>
<th>$T_2^k$</th>
<th>$T_3^k$</th>
<th>$T_4^k$</th>
<th>$T_5^k$</th>
<th>$T_6^k$</th>
<th>$T_7^k$</th>
<th>$T_8^k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_1^k$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$T_2^k$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$T_3^k$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$T_4^k$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$T_5^k$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$T_6^k$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$T_7^k$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$T_8^k$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Hypothesis Generation: An hypothesis is basically a collection of compatible tracks.

Examples: $\Theta^1_k = \{T^1_k, T^5_k, T^8_k\}$, $\Theta^2_k = \{T^2_k, T^3_k, T^7_k, T^8_k\}$

Score of an Hypothesis
$Sc(\Theta^i_k) = \sum_{T^j_k \in \Theta^i_k} Sc(T^j_k)$

Probability of an Hypothesis
$P(\Theta^i_k) = \frac{\exp(Sc(\Theta^i_k))}{1 + \sum_{j=1}^{N_h} \exp(Sc(\Theta^j_k))}$

Probability of a Track
$P(T^i_k) = \sum_{\Theta^i_k \ni T^i_k} P(\Theta^i_k)$

For keeping the number of tracks and hypothesis generation computations under control
- Clustering incompatible tracks into clusters can facilitate hypothesis generation
- N-scan pruning can be applied to track trees (instead of hypothesis trees in the previous case) by keeping histories of the tracks in memory.
- Merging the tracks that have the same recent measurement history is another idea to reduce the number of tracks.

The simplest method is to show the user the maximum probability hypothesis.
However, this can be a little jumpy because the maximum probability hypothesis can change quite erratically.
Another method is to show track clusters with their overall (weighted) mean, covariance and expected number of targets in them.
Another idea is to keep a separate track list which, at each step, is updated with a selection of tracks from different hypotheses.
Chapter 16 of the textbook gives extensive details about track based implementation of MHT and user presentation logic. Consult it for further details.
**General: Which Multi TT Method to Use?**

<table>
<thead>
<tr>
<th>Computation</th>
<th>SNR</th>
<th>Low</th>
<th>Normal</th>
<th>High</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>Low</td>
<td>Group TT or PHD</td>
<td>GNN</td>
<td>GNN</td>
</tr>
<tr>
<td>Normal</td>
<td>Normal</td>
<td>MHT</td>
<td>GNN or JPDA</td>
<td>GNN</td>
</tr>
<tr>
<td>High</td>
<td>High</td>
<td>TBD or MHT</td>
<td>MHT</td>
<td>Any</td>
</tr>
</tbody>
</table>

- GNN and JPDA are very bad in low SNR.
- When using GNN, one generally has to enlarge the overconfident covariances to account for neglected data association uncertainty.
- JPDA has track coalescence and should not be used with closely spaced targets, see the “coalescence avoiding” versions.
- MHT requires significantly higher computational load but it is said to be able to work reasonably under 10-100 times worse SNR.

---

**References**