An Extension of an Algorithm for Multiple Hypotheses Tracking

L.V.Bojilov

1. Introduction. The paper presents an extension of Nagarajan (et.al.) algorithm [1] for dealing with multitarget tracking. The algorithm in [1] is intended to overcome combinatorial problems, arising when multiple targets are to be tracked simultaneously in track-while-scan radars. We suggest in this paper additional rules in the algorithm processing, which lead to considerable reduction of computational load even in comparison with Nagarajan's algorithm.

2. The problem formulation. In reference [2] the authors present new approach for calculating probability of each hypothesis. They suggest to utilize information from signal processor of the radar for improving the tracking process. As a result of this, in the algorithm of [1] the authors consider only two possibilities for any measurement, receiving at scan k: a) to be originated from one of the tracking targets; or b) to be from a new target.

Following the notation in [3] the authors assume at scan k N targets $T_1, T_2, \ldots, T_N$, their predicted track measurements $\hat{z}_1(k), \hat{z}_2(k), \ldots, \hat{z}_N(k)$ and associated covariance matrices $S_1(k), S_2(k), \ldots, S_N(k)$, respectively, according to hypothesis, say, Ω_g^{k-1}, retained after scan $k-1$. They assume also the class conditional density of measurement $z_i(k)$ $(i = 1,2,\ldots, M)$ to be given by normal distribution

/1/ $$p\big(z_i(k)/T_j\big) = N\big(z_i(k); \hat{z}_j(k), S_j(k)\big), \quad j = 1,2,\ldots, N .$$

Using the assumption, mentioned above, and following Bayes theorem, they derive for probability of the event ψ_{ij}, that the i -th measurement is from j -th target

/2/ $$P\big(T_j/z_i(k)\big) = \frac{p\big(z_i(k)/T_j\big)}{\sum_j p\big(z_i(k)/T_j\big)} .$$

Consider all hypotheses retained at the end of scan $k-1$ the authors derive recursive formula for calculating probability of every new hypothesis at scan k according to every one hypothesis at scan $k-1$

/3/ $$P\big(\Omega_h^k\big) = \frac{1}{C}\left[P\big(\Omega_g^{k-1}\big)\prod_{i=1}^{M_k}\beta\big(i, j_h, \Omega_g^{k-1}\big)\right].$$

Here C is normalized constant and $\beta\big(i, j_h, \Omega_g^{k-1}\big)$ is probability calculated in eqn. /2/.

3. Nagarajan's algorithm. The most important feature of this formula is that the probability of any new hypothesis is proportional of certain factors already evaluated. The advantage of this feature can be seen in the algorithm, stated below, and proposed in reference [1].

Hereafter, we will assume one hypothesis retaining after $k-1$ scan, taking into account that presented part of the algorithm can be repeated for any additional hypothesis at scan $k-1$. For simplifying the notation let to represent factors β from eqn. /3/ as $\beta(m,t)$, where $m = 1,2,\ldots, M_k$ denotes measurements indices and $t = T_1, T_2, \ldots, T_N, T_{new}$ denotes target's indices. The values of β, as it has been mentioned above, can be previously evaluated. The Table 1 contains such kind of values from the example cited in [1].

The score of any feasible hypothesis will contain eight terms in the product as to the Table 1. A hypothesis is said to be feasible if not more than one measurement is associated with any known target, but multiple measurements can be associated with new targets (the last row in Table 1). We can see, however, that if we convert Table 1 dividing every column's element by the last element of the column (from T_{new} -row) the arrangement of the hypothesis according to their scores will not be changed as it is seen from Table 2.

Table 1

	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
T_1	**0.37**	**-**	**0.35**	**0.61**	**0.72**	**0.43**	**-**	**0.15**
T_2	**0.23**	**0.45**	**0.33**	**0.15**	**0.2**	**0.37**	**0.72**	**0.6**
T_3	**-**	**0.35**	**0.25**	**0.21**	**-**	**0.16**	**0.27**	**0.15**
T_4	**0.35**	**0.17**	**0.05**	**-**	**0.07**	**-**	**-**	**0.08**
T_{new}	**0.05**	**0.03**	**0.02**	**0.03**	**0.01**	**0.04**	**0.01**	**0.02**

Table 2

	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
T_1	**7.4**	**-**	**17.5**	**20.3**	**72**	**10.8**	**-**	**7.5**
T_2	**4.6**	**15**	**16.5**	**5**	**20**	**9.2**	**72**	**30**
T_3	**-**	**11.7**	**12.5**	**7**	**-**	**4**	**27**	**7.5**
T_4	**7.0**	**5.7**	**2.5**	**-**	**7**	**-**	**-**	**4**
T_{new}	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

Table 3

	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
I	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
T_1	**5**	**4**	**3**	**6**	**8**	**1**	**-**	**-**
T_2	**7**	**8**	**5**	**3**	**2**	**6**	**4**	**1**
T_3	**7**	**3**	**2**	**8**	**4**	**6**	**-**	**-**
T_4	**1**	**5**	**2**	**8**	**3**	**-**	**-**	**-**

And the last step before algorithm representation is to construct the **preferred measurements matrix** - Table 3. In the row T_1 of this table the value 5 means that M_5 is the most preferable measurement for the first target, the next value of 4 - that measurement M_4 is the next preferable and so on. For example one possible hypothesis is (5,7,3,1). Another way of expressing this hypothesis is by using **preference index** from first row of table 3 - (0,0,1,0). We can notice that the less is the index, the more preferable is corresponding measurement. Before starting algorithm' s steps it will be useful to discuss the next lemma. Let $P(\psi_i)$ represent the probability of the hypothesis ψ_i being true and let $Ind_i(n)$ represent the n -th element of preference-index presentation of ψ_i . Suggested lemma is

$$P(\psi_i) > P(\psi_j) \text{ if } Ind_i(n) \le Ind_j(n) ,$$

for each value of n running from 1 to the number N of known targets. Taking two hypotheses in preference-index presentation by means of this lemma we can conclude, in some cases, which is more likely without actually evaluating the products of their scores. According to the authors, this is one of the main achievement in the reference [1].

For clearness of the notation we will say that a hypothesis, presented in preference-index way, is of level l if the sum of its preference indices is equal l .Thus hypothesis (0,0,0,0) is of level *0*, hypothesis (0,1,0,0) is of level *1* and hypothesis (1,0,2,1) is of level *4*. Likewise, if two hypotheses are subject to the lemma' s rule- $Ind_i(n) \le Ind_j(n)$, we will say that hypothesis j is consequence from hypothesis i , i.e. it can be construct by only adding some values to the preference-index presentation of i . The particular steps of the algorithm stated in [1] are as follow:

Step 1. Creation of hypotheses of level $l+1$ from a given hypothesis at level l can be done by simply incrementing preference indices, one at a time.

Step 2. Feasibility checking of created hypothesis.

Step 3. If hypothesis is feasible we check whether it is consequence from any hypothesis out of candidate hypotheses list:

a) If it is not - we include it in the candidate hypotheses list;

b) If turn out that it is consequence from some of candidate hypothesis we discard it.

Step 4. If hypothesis is not feasible and it is not consequence from any of the hypotheses in candidate hypotheses list we include it in the list of nonfeasible hypotheses for subsequent processing.

Simulation program realizing stated above algorithm shows significant reduction of number of hypotheses to be processed as well as the running time for the task. But if we take an example with $N = 10$ targets and include in the scenario $M = 15$ measurements the combinatorial problem arise in two directions: a) time of processing and b) memory storage limitation (especially for the list of nonfeasible hypotheses for subsequent processing).

4. Extended algorithm. We will propose here an extension of stated above algorithm which, in some extent, overcomes problems arising with large examples and move ahead the limit of its real practice implementation.

Step I. Creation of new hypotheses of level $l+1$ from a given hypothesis at level l will perform by incrementing preference indices one et a time, choosing direction from left to right. But the process will start from the first non-zero index looking from right.

Step II. On this step we check whether the new hypothesis is consequence from some of hypothesis out of candidate hypotheses list:

a) If it is consequence from some of candidate hypothesis we discard it and go to Step I;

b) If it is not consequence from any of candidate hypotheses we continue with the next step.

Step III. Feasibility checking of new-created hypothesis:

a) If hypothesis is feasible we include it in candidate hypotheses list and go to step I;

b) If hypothesis is not feasible we examine the cause of its nonfeasibility:

- if repeated measurements occur at indices' places which are from the left side of right most nonzero index - then we discard this hypothesis end stop processing this branch (connected with level 'l' hypothesis);

- if this is not the case, the new hypothesis is included in the nonfeasible hypotheses list for subsequent processing.

Additional rule: If at level *1* a certain hypothesis turn out feasible, the index corresponding to its nonzero element become 'forbidden', i.e. it will not be used any more for hypotheses generation. By help of this rule we avoid creation of hypotheses which will be consequence from this feasible hypothesis.

It will be interesting to give rationale of some of extensions in the proposed algorithm. Let's take the four hypotheses at level *1* from the example of [1] - (1,0,0,0) , (0,1,0,0,) , (0,0,1,0) and (0,0,0,1). We can generate now four new hypotheses at level *2* from every one of level *1* (Table 4*)*:

Table 4

1,0,0,0	**0,1,0,0**	**0,0,1,0**	**0,0,0,1**
2,0,0,0	1,1,0,0	1,0,1,0	1,0,0,1
1,1,0,0	0,2,0,0	0,1,1,0	0,1,0,1
1,0,1,0	0,1,1,0	0,0,2,0	0,0,1,1
1,0,0,1	0,1,0,1	0,0,1,1	0,0,0,2

It is easy to be noticed that the elements above the main diagonal are the same as these under the main diagonal. But if we would follow the rule added to the **step I** we would avoid . hypotheses duplication saving time of the processor. Or additional examination in **step III**. Let us take the nonfeasible hypothesis (0,1,0,0,0,0) assuming repeated measurements at 2-d and 3-d positions. According to the **step I** we can create five new hypotheses of level *2*: (0,2,0,0,0,0) , (0,1,1,0,0,0) , (0,1,0,1,0,0) , (0,1,0,0,1,0) and (0,1,0,0,0,1). It is easy to conclude that every hypothesis after the second as well as their 'successors' will be nonfeasible. The reason is that the unit in the 2-d place and the zero in the 3-d place of the origin correspond to the repeated measurements. So, we can stop hypotheses generation after the second saving both time of the processor and memory storage.

5. Simulation results. The program realization of Nagarajan algorithm as well as the realization of its extension have been used for numerical experiments. The first experiment include the example in the [1]. We run the example in [1] with Nagarajan's algorithm for proving its correct program realization. The results from the running coincide with the results in the paper. We run the same example with the extended algorithm. If we accept the next abbreviations : **CH** - Number of Created Hypotheses, **HCF** - Number of Hypotheses Checked for Feasibility, **NAG** - Nagarajn's algorithm, **EXT** - Extended algorithm, **T/M** - Number of Targets/Number of Measurements the experimental results will look as follow

Table 5

	NAG	EXT
CH	**90**	**18**
HCF	**32**	**8**

Even in such a simple case with 4 targets and 8 measurements in the cluster the advantage of extended algorithm is obvious.

Another run of experiments with more complicated cases have been made. Table 6 compares created hypotheses and those hypotheses for which feasibility check had to be made. The two last columns contain running time in seconds on Intel 486 processor. Each value in the table was obtained by averaging over 20 independent program runs. It is seen that the case of 6 targets and 10 measurements in one cluster is the limit for implementing the Nagarajan's algorithm (assuming radar with 10 sec. scan).

Table 6

-	**CH**		**HCF**		**Time (in sec.)**	
T/M	**NAG**	**EXT**	**NAG**	**EXT**	**NAG**	**EXT**
5/9	**419**	**89**	**109**	**61**	**1.054**	**0.068**
6/10	**6078**	**628**	**1186**	**366**	**12.1**	**0.14**
7/12	**36583**	**4077**	**8646**	**3179**	**184.7**	**1.05**

The last run of experiments concerns the extended algorithm only and seeks to check the limit of its implementation. It is seen from Table 7 that extended algorithm can process twice more complicated cases then the algorithm from [1].

Table 7

T/M	5/9	6/10	7/12	8/13	9/14	10/15
Time (sec.)	0.086	0.14	1.05	3.00	6.07	8.4

6. Conclusions. In this paper an extension of Nagarajan algorithm [1] has been presented. Using the main ideas and the approach of the authors we have added some additional rules in algorithm processing. This enables to speed up considerably the algorithm work and to move ahead the limit of its practical implementation.

REFERENCES 1. NAGARAJAN, V., M.R. CHIDAMBARA, R.N. SHARMA, Combinatorial problem in multitarget tracking - a comprehensive solution, IEE Proc. 134, 1987, No 1, 113-118. 2. NAGARAJAN, V., M.R. CHIDAMBARA, R. N. SHARMA, New approach to improved detection and tracking performance in track-while-scan radars, Part2: detection, track initiation and association. 3. REID, D.B., An Algorithm for Tracking Multiple Targets, IEEE Transactions on Automatic control, vol. ac-24, No 6, 1979.