AN IMPROVED VERSION OF A MULTIPLE TARGET TRACKING ALGORITHM

Ljudmil BOJILOV

1. Introduction

The construction of algorithms for finding the first K-best solutions to the assignment problem has attracted a great deal of interest in recent years. Starting with the pioneer work of MURTY, [5] the investigations continued with works of DANCHICK and NEWNAM [6] and with the more recent work of MILLER, STONE and COX. [7] In the last of these works, following the Murty's method the authors implement three optimizations, producing a speedup by factor of over 20. On the other hand, the measurements-to-target association as part of a frame of MHT approach can be successfully formulated as a classical assignment problem. So, using an algorithm capable to find exact first K-best solutions of the so formulated assignment problem gives us K hypotheses of highest probability without first generating all feasible hypotheses and then pruning them.[3]

In one of their works, NAGARAJAN, CHIDAMBARA and SHARMA, keeping essentially the REID's approach, proposed an algorithm for finding directly K hypotheses of highest probability. [1] In another of their works, NAGARAJAN and co-authors presented a new approach for calculating probability of each hypothesis.[2] They suggested to utilize information from the signal processor of the radar for improving the tracking process. As a result, in the algorithm of NAGARAJAN [1] the authors consider only two possibilities for any measurement, received at scan k : a) to be originated from one of the tracking targets; or b) to be from a new target. In our previous work [4] we proposed an extension of the algorithm of NAGARAJAN [1] achieving considerable speedup of finding the first K-best hypotheses. In the present work we further improved the extended algorithm and carried out more comprehensive experiments with more sophisticated scenarios and by using more powerful PC processor. As a result, some additional findings are presented, too.

This work is organized as follows. In the next section the main ideas from the work [2] of NAGARAJAN *et al* are outlined including the main expressions of hypotheses

probabilities computation. In section 3 the NAGARAJAN's algorithm is discussed and its principle steps are described. Section 4 contains presentation of the improved version of the algorithm and discussion of additional rules in algorithm processing. In Section 5 Experimental results are included and analyzed in section 5. The results are summarized in Section 6.

2. Problem formulation

In their work [2] NAGARAJAN and co-authors present new approach for calculating probability of each hypothesis. They suggest utilizing information from the signal processor of the radar in order to improve the tracking process. As a result, in the algorithm presented in their companion paper, [1] the authors consider only two possibilities for any measurement, received at scan k: a) to be originated from one of the tracking targets; or b) to be from a new target.

Following the notation from the work of REID, [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

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

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

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

Considering 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 $k-1$

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

Here, C is normalization constant and $\beta\left(i,j_h,\Omega_g^{k-1}\right)$ is probability calculated in uation (2).

3. Nagarajan's algorithm

The most important feature of this formula is that the probability of any new hypothesis is proportional to certain factors already evaluated. The advantage of this feature can be seen in the algorithm, presented below.[1]

Hereafter, we shall assume one hypothesis retained after the $k-1$-th 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's represent factors β from equation (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 targets' indices. The values of β, as it has been mentioned above, can be previously evaluated. Table 1 contains such kind of values from the example cited in the paper of NAGARAJAN *et al.* [1]

The score of any feasible hypothesis will contain eight terms in the product as presented in Table 1. A hypothesis is said to be feasible if no 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: The cost matrix of the example

	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: The cost matrix with normalized elements

	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**

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 means 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 the first row of Table 3 - (0,0,1,0). We can notice that the smaller the index is, the more preferable is the corresponding measurement.

Table 3: The preferred measurements matrix

	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**	**-**	**-**	**-**

Before describing the steps of the algorithm it will be useful to discuss the next lemma. Let $P(\psi_i)$ represents the probability of the hypothesis ψ_i being true and let $Ind_i(n)$ represents the n-th element of preference-index presentation of ψ_i. The suggested lemma is:

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

for each value of n 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. This may be considered as one of the main achievements presented in the first quoted work by NAGARAJAN *et. al.* [1]

For clearness of notation we shall say that a hypothesis, presented in way of preference-indexes, is of level l if the sum of its preference indices is equal to l. Thus the 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*, and so on. Likewise, if two hypotheses are subject to the lemma's rule - $Ind_i(n) \le Ind_j(n)$, we shall say that hypothesis ψ_j is consequence from hypothesis ψ_i, i.e., it can be constructed by only adding some values to the preference-index presentation of ψ_i. The hypotheses generation can be represented like constructing a tree. From every hypothesis (nod) at a given level *l*, *N* hypotheses (branches) of level *l+1* can be generated by simply incrementing preference indices, one at a time. Every generated hypothesis has to be checked: a) for feasibility, and b) if it is a consequence of some of the previously found feasible hypotheses. For the normal algorithm processing three lists have to be maintained: a) list of found feasible hypotheses sorted by their scores (candidate hypotheses); b) list of unfeasible hypotheses for the subsequent processing without calculating the scores; and c) ranked list of the best hypotheses.

The algorithm starts with checking the 0-level hypothesis. If it proves to be feasible, this is the first-best hypothesis (according the lemma presented above) and we put it in the ranked list of best hypotheses. We say that consecutive best hypothesis is found if no meaningful hypothesis of a higher level can be generated. This will be the first hypothesis out of sorted feasible hypotheses list. Every time we find consecutive best hypothesis, we use it for constructing the next hypothesis' tree.

The particular steps of the algorithm stated in the cited work [1] are as follows:

Step 0. We take the just found consecutive-best hypothesis from the top of feasible (or candidate) hypotheses list and begin constructing a tree.

Step 1. Hypotheses generation of level $l+1$ from a given hypothesis at level *l*.

Step 2. Checking feasibility of the created hypothesis.

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

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

b) If it turns out that the checked hypothesis is consequence of some of the candidate hypotheses, we discard it.

Step 4. If the hypothesis is not feasible and it is not consequence of any of the hypotheses in the candidate hypotheses list, we implement another checking –

whether it is consequence from any of the hypotheses in the list of non-feasible hypotheses. If it is not, we include the hypothesis in this list for subsequent processing. Otherwise, we discard it and continue with the next step.

Step 5. We take the subsequent hypothesis from the unfeasible hypotheses list and return to Step 1. If all hypotheses from the unfeasible hypotheses list are already used and the list is empty, we say that we found the next best hypotheses and return to Step 0.

The algorithm terminates when the list of the first K-best hypotheses fills up.

The simulation realizing this algorithm shows significant reduction of the number of hypotheses to be processed, as well as the running time for the task. However, if we take an example with $N = 10$ targets and include in the scenario $M = 15$ measurements, combinatorial problems arise in two directions: a) time of processing and b) memory storage limitation (especially for the list of non-feasible hypotheses for subsequent processing).

4. Extended algorithm

Before describing our extension we shall further elaborate on the NAGARAJAN's algorithm processing. Let us take the four hypotheses at level *1* from the example of their paper [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 hypothesis of level *1* (Table 4):

Table 4: Hypotheses generation process

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

In table 4, the elements above the main diagonal are the same as those under the main diagonal. In our extension we shall avoid hypotheses duplication for saving processor time. There is another part of the algorithm, where needless hypotheses generation is carried out. Let us take the non-feasible hypothesis (0,1,0,0,0,0) assuming repeated measurements at 2-th and 3-rd positions. According to the **step 1**, we can create six new level *2* hypotheses: (1,1,0,0,0,0), (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 third, as well as their 'successors' up to the bottom of the Table 3 will be non-

feasible. The reason is that the unit in the 2-nd place and the zero in the 3-rd place of the origin correspond to the repeated measurements according to the measurement-oriented presentation. So, we can stop hypotheses generation after the second hypothesis saving both time of the processor and memory storage.

Two additional terms are introduced for convenience. Every new hypothesis at a given level is created from some hypothesis of the upper level by incrementing its preference-index presentation at some point. We call this point '*creation point*,' or *CP*. Secondly, in regard to the conclusion that for some unfeasible hypothesis there is no use to continue hypotheses' creation after the point where repeated measurement occurs, we call that point '*breaking point*,' or *BP*. It is obvious that the cycle of hypotheses' generation has to be run from CP to BP only. Moreover, when for some hypothesis out of unfeasible hypotheses list *CP* > *BP*, we discard this hypothesis, cutting off the corresponding part of the hypotheses' tree.

Step 0. The last just found consecutive-best hypothesis from the top of feasible (or candidate) hypotheses list serves to begin the construction of a tree.

Step I. A new hypothesis of level $l+1$ is created from a given hypothesis at level l by incrementing preference indices one at a time; the cycle is run only from CP to BP. When a new hypothesis of level $l+1$ is created, we remember its 'creation point'.

Step II. On this step we check whether the new hypothesis is consequence of any of the hypothesis out of the candidate hypotheses list. If it is consequence of some of the candidate hypothesis, we discard it and go back to Step I. Otherwise continue with the next step.

Step III. If the checked hypothesis is not consequence of any of the candidate hypotheses, we continue with the feasibility check. If the hypothesis is feasible, it is included in the candidate hypotheses list and then we return to step I.

Step IV. If the checked hypothesis is not feasible, we examine the place, where the repeated measurement occurs and remember it as a 'breaking point.' After that, we include it in the list of non-feasible hypotheses for subsequent processing.

Step V. We take the subsequent hypothesis from the unfeasible hypotheses list and return to Step 1. If all hypotheses from the unfeasible hypotheses list are already used and the list is empty, we say that we have found the next-best hypotheses and the return to Step 0.

As in the previous case, the algorithm terminates when the list of the first K-best hypotheses fills up.

There is a sound rationale for some extensions in the proposed algorithm. Starting hypotheses generation from index CP, we avoid redundant steps of the algorithm in

two directions: a) preventing hypotheses duplication, and so, saving processor's time and memory storage, especially for the unfeasible hypotheses list, and b) obviating the checking whether the hypothesis is consequence of any hypothesis out of the unfeasible hypotheses list. This list is much longer than the feasible hypotheses list and, thus, its checking is one of the most time-consuming parts of the algorithm.

The second issue is the 'breaking point.' When the cycle of hypotheses generation is stopped at BP, we truncate significant parts of the hypotheses' tree and so achieve savings of processor time and memory storage. It is important to notice that this second extension is effective only in combination with the first one. Finally, in our extensions feasibility checking and consequence checking are rearranged. The merits are that non-feasibility is not yet a reason to discard a hypothesis, whereas, if it is consequence of any of the feasible hypotheses, we can readily discard it.

5. Simulation results

The program realization of NAGARAJAN algorithm, as well as the realization of its extension has been used for numerical experiments. The first experiment includes the example from the work of NAGARAJAN. [1] We run this example with NAGARAJAN's algorithm for proving the correct program realization. The results from the experiment fully coincide with the results in the original paper. Then, we run the same example with the extended algorithm. If we accept the following abbreviations: **GH** - number of **G**enerated **H**ypotheses, **HCF** - number of **H**ypotheses **C**hecked for **F**easibility, **NAG** - **NAG**arajn's algorithm, **EXT** - **EXT**ended algorithm, **T/M** - number of **T**argets/ **M**easurements, the experimental results may be presented in Table 5.

Table 5: Comparison of the two algorithms on the cited example [1]

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

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

Another series of experiments have been run with 13 different scenarios with increasing complexity. Table 6 compares created hypotheses and those hypotheses for which feasibility checks had to be made. The 6-th and 7-th columns contain running time in seconds for finding the first 100-best hypotheses on a 1.4GHz AMD/XP processor. The last column contains speed advantage ratio. For the simplest cases the running time of the extended algorithm proved to be less than the time

resolution of the computer. Additionally, for the most complicated cases the running time of the original algorithm is out of any reasonable limits; the simulations have not been carried out in such cases. Each value in the table was obtained by averaging over 50 independent program runs with 50 different values of random generator seed. Of course, one and the same random number stream has been used for any one scenario. As it can be seen from Table 6, the scenario with 8 targets and 13 or 14 measurements prove to be the practical implementation limit of Nagarajan's algorithm (assuming radar with 10 sec. scan). In the last and most complicated scenarios (with 15 targets and more than 20 measurements) the extended algorithm reaches its limit for practical implementation, even though the average running time for these scenarios is less than the assumed scan duration of 10 seconds. The problem is that for the some of experiments, i.e. the heaviest scenario, the processing time of the algorithm exceeds 10 seconds.

Table 6: Comparison of the two algorithms performance in terms of generated and checked hypotheses and processing time

Targets/ Measure-ments (**T/M**)	**GH**		**HCF**		**Time (*in seconds*)**		Speed advantage ratio
	NAG	**EXT**	**NAG**	**EXT**	**NAG**	**EXT**	
6/11	1476	-	310	-	0.03	-	-
7/12	5547	-	1117	-	0.355	-	-
8/13	27211	-	6291	-	4.81	-	-
8/14	45550	2913	12650	1760	9.42	0.03	314
9/15	104168	6586	31788	3955	30.83	0.112	275
10/16	190536	15320	72226	10014	67.43	0.327	206
11/17	306024	22724	126458	14688	117.14	0.562	208
12/18	576424	39466	236837	28406	211.75	1.082	196
13/19	-	48627	-	35071	-	1.833	-
13/20	-	65536	-	48103	-	2.296	-
14/21	-	76560	-	57167	-	3.425	-
15/22	-	103787	-	72405	-	5.278	-
15/25	-	133526	-	103478	-	6.843	-

Through an additional experiment we reveal an interesting and very useful feature of the presented algorithm. We have tested the dependence of the processing time on number of first K-best hypotheses with 14T/21M scenario (Table 6). Surprisingly, the experiment exhibits very week dependence of the running time on the number of first best hypotheses found (Table 7).

The explanation of this result is straightforward: for the main part of its work the algorithm determines the first best hypothesis. At that time, the list of candidate hypotheses is full and for finding any subsequent hypothesis only a few additional operations have to be performed. This feature makes the presented algorithm a good alternative to the well-known algorithms, proposed recently, for finding the first K-best hypotheses, directed for use in the framework of the MHT approach.

Table 7: Processing time (*in seconds*) for finding different number of first K-best hypotheses

Rand-Seed values	***Number of the first K-best hypotheses found***					T_{100}/T_1
	1	10	20	50	100	
13	2.03	2.09	2.09	2.14	2.42	1.19
15	5.5	5.5	5.55	5.76	6.37	1.16
17	1.48	1.53	1.59	1.92	2.69	1.82
25	3.13	3.13	3.18	3.24	3.57	1.14
27	2.03	2.09	2.14	2.26	2.58	1.27
33	1.82	1.82	1.87	2.19	2.86	1.57
35	2.2	2.2	2.25	2.8	2.91	1.32
53	2.74	2.75	2.76	2.81	2.91	1.06
55	3.42	3.51	3.52	3.68	4.12	1.20
65	5.06	5.1	5.11	5.28	5.54	1.09

Conclusion

This paper presented an improved version of our extension of NAGARAJAN's algorithm. [1] Defining two points in the hypotheses generation cycle—'*creation point*' and '*breaking point*'—a considerable reduction of hypotheses' tree has been achieved. By rearranging feasibility checking and consequence checking we attain additional pruning of this tree. As combined result of the improvements, the time necessary to implement the presented algorithm has been reduced by more than two

orders of magnitude compared to NAGARAJAN's algorithm. In addition, taking into account the week dependence of the processing time on the number of the best hypotheses found, it can be inferred that the presented algorithm can be successfully implemented besides other algorithms finding the first K-best hypotheses in implementing multiple targets tracking.

Acknowledgment

The work on this paper was supported by the Center of Excellence BIS21 under Grant ICA1-2000-70016.

Notes:

1. V. Nagarajan, M.R. Chidambara, and R.N. Sharma, "Combinatorial Problem in Multitarget Tracking - A Comprehensive Solution," *IEE Proceedings* 134, 1 (1987): 113-118.
2. V. Nagarajan, M.R. Chidambara, and R.N. Sharma, "New Approach to Improved Detection and Tracking Performance in Track-While-Scan Radars, Part2: Detection, Track Initiation and Association," *IEE Proceedings* 134, 1 (1987): 93-98.
3. Donald B. Reid, "An Algorithm for Tracking Multiple Targets," *IEEE Transactions on Automatic Control* vol. AC-24, 6 (1979): 843-854.
4. Ljudmil Bojilov, "An Extension of an Algorithm for Multiple Hypotheses Tracking," *Comptes rendus del'Academie Bulgare de Sciences* 5, 5 (1997): 45-48.
5. Katta G. Murty, "An algorithm for Ranking All the Assignments in Order of Increasing Cost," *Operational Research* 16 (1968): 682-687.
6. Roy Danchick and G. E. Newnam, "A Fast Method for Finding Exact N-Best Hypotheses for Multitarget Tracking," *IEEE Transactions on Aerospace and Electronic Systems* 29, 2 (1993): 555-560.
7. Matt L. Miller, Harold S. Stone, and Ingemar J. Cox, "Optimizing Murty's Ranked Assignment Method," *IEEE Transactions on Aerospace and Electronic Systems* 33, 3 (1997): 851-862.

LJUDMIL VLADIMIROV BOJILOV is researcher in Multisensor Multitarget Tracking and Data Fusion at the Central Laboratory for Parallel Processing, Bulgarian Academy of Sciences, Sofia, since 1989. He received a M.Sc. degree in Nuclear physics from the Sofia University in 1970 and a Ph.D. degree in Electronic circuits optimization from the Technical University of Sofia in 1991. *E-mail*: bojilov@bas.bg.