[7] Vidyasagar, M. (1985)
Control System Synthesis, A Factorization Approach.
Cambridge, MA: M.I.T. Press, 1985.

[8] Barmish, B. R. (1994)
New Tools for Robustness of Linear Systems.
New York: Macmillan, 1994.

On Finding Ranked Assignments with Application to Multitarget Tracking and Motion Correspondence

Within the target tracking community there is strong interest in computing a ranked set of assignments of measurements to targets. These *k*-best assignments are then used to determine good approximations to the data association problem. Much earlier work described algorithms which either had exponential worst case time or were not guaranteed to return the *k*-best assignments. Most recently, Danchick and Newnam [4] described a fast algorithm for finding the *exact k*-best hypotheses. However, in the worst case, $k!$ linear assignment problems must be solved. This correspondence describes an algorithm originally due to Murty [6] for optimally determining a ranked set of assignments in polynomial time and which is linear in *k*.

I. INTRODUCTION

There is considerable interest in the data association or motion correspondence problem, i.e., in determining which measurements originate from which features or targets in the environment. This problem has been extensively studied by the multitarget tracking and surveillance communities [2] where two algorithms have received considerable attention; the multiple hypothesis filter (MHT) of [9] and the joint probabilistic data association filter (JPDAF) of Bar-Shalom [2]. While the MHT and JPDAF algorithms are quite different in spirit, they share two common stages. The first validates measurements to targets (or features) in order to construct an *association* cost matrix. A non-zero entry, $d_{i,j}$, indicates that measurement z_i validated to target y_j and the value $d_{i,j}$ denotes the cost of matching z_i with target y_j. Next, both the MHT and JPDAF require the enumeration of *all* legal assignment.[1] Each assignment also called an hypothesis, has an associated cost computed from a probability value which is used in subsequent calculations. Unfortunately, enumeration of all the hypothesis matrices is NP-hard.[2]

In general, a very good approximation can be obtained by looking at only the k most likely sets of assignments, the remaining hypotheses having negligible effect due to their very small probabilities. There is therefore strong interest in determining a ranked set of assignments. Of course, determining the k-best assignments must be accomplished efficiently, i.e., it must not involve the enumeration and subsequent sorting of all possible sets. Recently, several papers have proposed approximate solutions to this problem. Nagarajan, et al. [7] present an algorithm in which the k-best hypotheses are generated by an "easy search process instead of going through an extensive enumeration." The authors do not provide a theoretical analysis of the computational complexity of their branch and bound scheme. However, while evidence is presented to demonstrate that in some cases a very significant reduction in computation is achieved, in the worst case the cost may still be exponential. Brogan [1] provides an algorithm for determining a ranked set of p assignments. However, this set is not guaranteed to be the p-best, i.e., it is possible to miss certain good combinations. A sufficient condition is provided to determine $q \leq p$ such that the first q assignments are optimal. Once again, no formal analysis of the computational complexity is provided.

Danchick and Newman [4] recognize that finding the best hypothesis can be formulated as a classical linear assignment problem, see Section II, and then show how modifications to the cost matrix followed by repeated solutions to these new assignment problems allow the k-best assignments to be computed. Danchick and Newnams' algorithm has two disadvantages. First, in the worst case, Danchick and Newnams' algorithm requires the solution of $k!$ linear assignment problems. Though the average case is expected to be considerably better, it is highly desirable to reduce the order of this dependency. Second, at the end of each iteration (or "sweep")

Manuscript received August 11, 1993; revised February 11 and July 23, 1994.

IEEE Log No. T-AES/31/1/08032.

[1]A *legal* assignment of measurements to targets must satisfy the constraints that 1) a measurement can originate from only a single source target (often called the uniqueness constraint [10]) and, 2) that a target can give rise to no more than one measurement per cycle. These constraints require that a legal set of assignments, represented as a *hypothesis* matrix, have no more than a single non-zero entry in any row and/or column.

[2]It is related to determining the permanent of a (association) matrix [5]. The importance of the permanent representation is that Valiant [11] has shown that the evaluation of the permanent of a general 0–1 matrix is NP-hard. Consequently, Collins and Uhlmann [3] infer that to compute the exact solution to the JPDA (and MHT) must also be NP-hard.

Danchick and Newnams' algorithm must identify and eliminate duplicate assignments.

In this correspondence, we describe an algorithm originally due to Murty [6] to optimally determine the k-best assignments in polynomial time. The number of linear assignment problems that must be solved is *linear* in k. In fact, "the computations required at each stage are the solving of at most $(n-1)$ assignment problems, each of sizes $2,3,\ldots,n$" [6]. The algorithm avoids solving duplicate assignment problems, thereby eliminating the need to compare and delete duplicate hypotheses. Finally, in the average case, the dimension of the assignment problems that must be examined decreases with k.

II. ALGORITHM

Consider first the problem of finding the single most probable hypothesis. This can be cast as a weighted bipartite matching problem by constructing a bipartite graph in which each node on one side represents one of the measurements, each node on the other represents one of the targets, and each arc, $\langle z_i, y_j, l\rangle$, gives the log likelihood l, that measurement z_i should be assigned to target y_j. The log of the likelihood of a given assignment can be found by summing the log likelihoods of all the arcs that it specifies.

Finding the best hypothesis, then, is a matter of finding the assignment that maximizes this sum. This is an instance of the classical assignment problem from combinatorial optimization, and can be solved very efficiently in polynomial time [8].

To find the k-best we use Murty's algorithm [6] for producing assignments in order of decreasing likelihood. This algorithm is guaranteed to find the k-best assignments in polynomial time. A brief description of Murty's algorithm follows.

Given a solution S to an assignment problem P, we can partition the assignment problem into a list of new problems with the following properties.

1) The set of valid solutions for any one of the problems in the list doesn't intersect with the set of solutions for any other problem in the list. That is, there are no duplicate problems.

2) The union of the sets of valid solutions for all the problems in the list is exactly the set of solutions for problem P, minus solution S.

Murty gives a method for computing this partitioning in $O(N^2)$ time.

For the k-best algorithm, a list of problem/solution pairs is kept. Each pair consists of an assignment problem and its best solution. The list is initialized with the initial problem to be solved. In each iteration, the best solution is found, then removed from the list, and replaced with its partitioning. So, in the first iteration, the single best solution S_0 is found to the problem, and the list is altered so the set of possible solutions no longer contains S_0. The next iteration gives the next-best solution S_1 and changes the list so that possible subsequent solutions no longer include S_1 or S_0, and so on. Fig. 1 outlines the algorithm.[3] The partitioning is performed by the loop in Step 4.4.

```
1. Find the best solution, $S_0$, to $P_0$ (this can be done using a standard
   algorithm like the Hungarian method)

2. Initialize the list of problem/solution pairs with $< P_0, S_0 >$

3. Clear the list of solutions to be returned

4. For $i$ = 1 to $k$, or until the list of problem/solution pairs is empty

    4.1 Search through the list of
        problem/solution pairs, and find the pair, $< P, S >$ that has the
        best solution value
    4.2 Remove $< P, S >$ from the list of problem/solution pairs
    4.3 Add $S$ to the list of solutions to be returned
    4.4 For each triple, $< y, z, l >$, found in $S$
      4.4.1 Let $P' = P$
      4.4.2 Remove the triple $< y, z, l >$ from $P'$
      4.4.3 Look for the best solution, $S'$, to $P'$
      4.4.4 If $S'$ exists
        4.4.4.1 Add $< P', S' >$ to the set of problem/solution pairs
      4.4.5 From $P$, remove all triplicates that include y, and all
            triples that include $z$, except $< y, z, l >$ itself.  (This reduces
            the dimension of the problem by one).
```

Fig. 1. Marty's algorithm for finding the k-best solutions to an assignment problem, P_0.

III. DISCUSSION

We have implemented Murty's ranked assignment algorithm as part of an MHT tracking algorithm with very good results. Our implementation does not employ a number of further possible optimizations such as tayloring the Hungarian method for use in Murty's algorithm, or keeping the set of problem/solution pairs on a heap structure instead of a simple list.

There are some simple modifications to Murty's algorithm which make it highly suitable for use in an implementation of the MHT method. These include the following.

1) Solving multiple assignment problems at the same time.

In each iteration of the MHT algorithm, we want to find the k-best descendents of the k-best hypotheses from the previous iteration. This involves finding the k-best solutions to k different assignment problems. The k solutions could come from any subset of this set of problems: they might all be solutions to just one of the problems, they might each be a solution to a different problem, half might be solutions to one problem and the rest solutions to different problems, etc.

The obvious way of using Murty's algorithm for this task is to run it k times, once for each assignment problem, and then choose the k best solutions from the resulting set of k^2 solutions. However, if instead,

[3]Note that Murty's algorithm is applicable to both square and rectangular assignment matrices.

we simply change Steps 1 and 2 of Murty's algorithm to initialize the list of problem/solution pairs with all k problems at the beginning, and then run the rest of the algorithm only once, the correct result will be obtained for considerably less computation.

2) Changing the termination condition.

Murty's algorithm need not be restricted to getting only the k-best solutions. Instead, it can be viewed as a ranked queue of solutions, allowing for more sophisticated methods of deciding when to stop generating new hypotheses. This only entails changing the termination condition of the loop in Step 4.

We have had good results terminating hypothesis generation whenever the likelihood of the next hypothesis is below some fixed percentage of the likelihood of the first (most likely) hypothesis. Nagarajan, et al. use a similar termination condition [7].

3) Allowing multiple possible assignments between the same target and measurement.

In some implementations of the MHT algorithm, targets can follow a number of alternative motion models [2] meaning that a given measurement could be assigned to a given target in any of a variety of ways. This would be represented in our implementation as multiple, alternative arcs between the same two nodes in the bipartite graph, each with a different log likelihood.

Only two changes are required to make the algorithm in Fig. 1 handle this. First, each arc must be further labeled with the motion model it is based on, so, instead of representing assignment problems as lists of $\langle y,z,l\rangle$ triples, we use lists of $\langle y,z,l,m\rangle$ where m is a motion model. Steps 4.4, 4.4.2, and 4.4.5 of the algorithm must be modified accordingly.

Second, the routine for finding the best solution to an assignment problem, which is called in Steps 1 and 4.4.3, must be modified so that it ignores all but the most likely arc between any given pair of nodes.

We implemented Murty's algorithm using the Hungarian method [8] to solve each assignment problem. The Hungarian method has time complexity $O(N^3)$. There are faster weighted bipartite matching algorithms but the Hungarian method was most accessible. The complexity of Murty's algorithm using the Hungarian method is $O(kN^4)$. This is the worst case time, and the average time is considerably better. In order to investigate the average case performance of Murty's algorithm we ran it on 100 random problems of 19 different sizes, using a MIPS R4000 running at 50 MHz. Each problem of size N required the assignment of N measurements to N tracks. Every one of the possible N^2 assignments was given a random log likelihood.

Fig. 2 shows the average times to initialize the list of problem/solution pairs (Steps 1 and 2), and to compute the first five best solutions. The time shown for each solution is the time to make one pass through

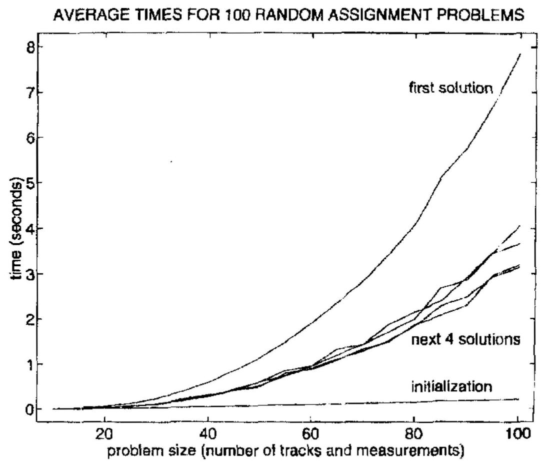

Fig. 2. Time to compute the best and k-best assignments for vary problem sizes.

the loop in Step 4 of the algorithm, so it includes the time to partition the problem in preparation for the next solution. Fig. 2 indicates that the average time to get each solution is roughly $O(N^3)$, rather than the worst case $O(N^4)$. This can be expected, since most assignment problems can be solved by just using the single most likely assignment for each measurement. These trivial cases take $O(N^2)$ time for the Hungarian method. The loop is Step 4.4 of Murty's algorithm then adds an extra factor of N.

It is interesting to note that it takes considerably more time to find the partitioning of the first solution than to find the partitioning of each of the 4 subsequent solutions. The time for partitioning depends on the size of the problem chosen in Step 4.1 of the algorithm. In the worst case, this size should be constant. But as the algorithm progresses, the average size of the problems in the set of problem/solution pairs decreases, so the average size of the problem chosen also decreases. Fig. 3 illustrates this phenomenon. The top graph shows the time taken for each successive solution to a size 20 assignment problem. The lower graph shows the average size of the problem chosen in Step 4.1 of each iteration of loop 4. The same patterns were found for all other problem sizes we examined.

Of course, the problems in these tests are much harder than the average problems found in an actual implementation of the MHT algorithm. It is extremely unlikely to have each of 100 measurements validate to each of 100 targets. A more realistic test, in which each measurement validated to 5 randomly selected targets, with random log likelihoods, revealed a constant increase in speed of roughly a factor of 3.

In conclusion, we believe that an efficient algorithm for generating a ranked set of assignments allows the practical application of Reid's MHT algorithm to many applications. Several researchers have recognized the importance of finding ranked assignments but do not appear to have been aware of Murty's algorithm for

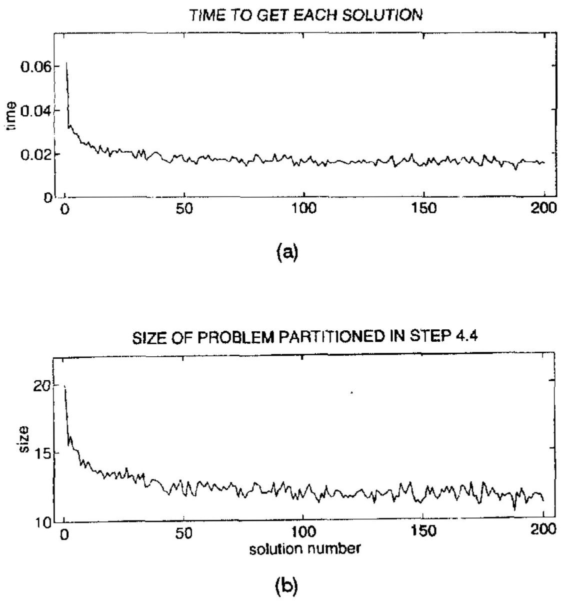

Fig. 3. (a) Time taken to compute k solutions (1–200) for fixed size problem (20×20). (b) Shows how size of subproblem reduces as k increases.

generating such a ranking. We believe this algorithm will become the preferred hypothesis generation algorithm for MHT and JPDAF multitarget tracking applications.

ACKNOWLEDGMENTS

It is a pleasure to thank Satish Rao and Bob Tarjan for guiding the authors to the appropriate literature and for many useful discussions.

INGEMAR J. COX
NEC Research Institute
4 Independence Way
Princeton, NJ 08540

MATT L. MILLER
Pilies 32-10
Vilnius
Lithuania

REFERENCES

[1] Brogan, W. L. (1989)
Algorithm for ranked assignments with applications to multiobject tracking.
IEEE Journal of Guidance, **12**, 3 (1989), 357–364.

[2] Bar-Shalom, Y., and Fortmann, T. E. (1988)
Tracking and Data Association.
New York: Academic Press, 1988.

[3] Collins, J. B., and Uhlmann, J. K. (1992)
Efficient gating in data association with multivariate distributed states.
IEEE Transactions on Aerospace and Electronic Systems, **28**, 3 (1992), 909–915.

[4] Danchick, R., and Newnam, G. E. (1993)
A fast method for finding the exact n-best hypotheses for multitarget tracking.
IEEE Transactions on Aerospace and Electronic Systems, **29**, 2 (1993), 555–560.

[5] Minc, H. (1978)
Encyclopedia of Mathematics and Its Applications, Vol. 6, *Permanents.*
Reading, MA: Addison-Wesley, 1978.

[6] Murty, K. G. (1968)
An algorithm for ranking all the assignments in order of increasing cost.
Operations Research, **16** (1968), 682–687.

[7] Nagarajan, V., Chidambara, M. R., and Sharma, R. N. (1987)
Combinatorial problems in multitarget tracking—A comprehensive survey.
IEE Proceedings, Pt. F, **134**, 1 (1987), 113–118.

[8] Papadimitriou, C. H., and Steiglitz, K. (1982)
Combinatorial Optimization.
Englewood Cliffs, NJ: Prentice-Hall, 1982.

[9] Reid, D. B. (1979)
An algorithm for tracking multiple targets.
IEEE Transactions on Automatic Control, **AC-24**, 6 (Dec. 1979), 843–854.

[10] Ullman, S. (1979)
The Interpretation of Visual Motion.
Cambridge, MA: MIT Press, 1979.

[11] Valiant, L. G. (1979)
The complexity of computing the permanent.
In *Theoretical Computer Science*, **8** (1979), 189–201.

Robust Failure Detection with Parity Check on Filtered Measurements

A new development in failure detection based on the parity check method is presented. The novelty lies in that the parity check is now performed on a set of *filtered* measurements. With a modest increase in complexity, the ability to reduce the effect of model uncertainties now becomes an inherent merit of the new detection scheme.

MAIN RESULTS

Consider the model of a linear discrete time system

$$\begin{aligned} x(k+1) &= Ax(k) + B_1 w(k) + B_2 u(k) + E\delta(k) \\ y(k) &= Cx(k) + Dw(k). \end{aligned} \tag{1}$$

The signal $B_1 w \in R^n$ represents uncertainties in the model, $Dw(k)$ is the measurement noise, $y \in R^r$ is the measured variable, $u \in R^m$ is the control input, and $\delta(k)$ represents the failures associated with actuators and components. The $E\delta$ term is present only when failures have occurred, and matrix E, which is sometimes called a failure signature, can be time

Manuscript received January 5, 1994.

IEEE Log No. T-AES/31/1/08030.

This work was supported partially by the National Science Foundation under Grant ECS9209983.