Evaluation of the Jonker-Volgenant-Castanon (JVC) assignment algorithm for track association

Donald B. Malkoff

Lockheed Martin Advanced Technology Laboratories
1 Federal Street, A&E 3W, Camden, New Jersey 08102

ABSTRACT

The Jonker-Volgenant-Castanon (JVC) assignment algorithm was used by Lockheed Martin Advanced Technology Laboratories (ATL) for track associations in the Rotorcraft Pilot's Associate (RPA) program. RPA is Army Aviation's largest science and technology program, involving an integrated hardware/software system approach for a next generation helicopter containing advanced sensor equipments and applying artificial intelligence "associate" technologies. ATL is responsible for the multisensor, multitarget, onboard/offboard track fusion. McDonnell Douglas Helicopter Systems is the prime contractor and Lockheed Martin Federal Systems is responsible for developing much of the cognitive decision aiding and controls-and-displays subsystems. RPA is scheduled for flight testing beginning in 1997.

RPA is unique in requiring real-time tracking and fusion for large numbers of highly-maneuverable ground (and air) targets in a target-dense environment. It uses diverse sensors and is concerned with a large area of interest. Target class and identification data is tightly integrated with spatial and kinematic data throughout the processing. Because of platform constraints, processing hardware for track fusion was quite limited. No previous experience using JVC in this type environment had been reported.

ATL performed extensive testing of the JVC, concentrating on error rates and run-times under a variety of conditions. These included wide ranging numbers and types of targets, sensor uncertainties, target attributes, differing degrees of target maneuverability, and diverse combinations of sensors. Testing utilized Monte Carlo approaches, as well as many kinds of challenging scenarios. Comparisons were made with a nearest-neighbor algorithm and a new, proprietary algorithm (the "Competition" algorithm). The JVC proved to be an excellent choice for the RPA environment, providing a good balance between speed of operation and accuracy of results.

Keywords: Jonker-Volgenant-Castanon, JVC, assignment, data fusion, tracking, track fusion, data association, Rotorcraft Pilot's Associate, RPA, Competition, optimization.

1. INTRODUCTION

1.1 RPA

RPA is Army Aviation's largest science and technology program, involving an integrated hardware/software system approach for a next generation helicopter containing advanced sensor equipments and applying artificial intelligence "associate" technologies. Lockheed Martin Advanced Technology Laboratories is responsible for the multisensor, multitarget, onboard/offboard track fusion[1]. McDonnell Douglas Helicopter Systems is the prime contractor and Lockheed Martin Federal Systems is responsible for developing much of the cognitive decision aiding and controls-and-displays subsystems. RPA is scheduled for flight testing beginning in 1997.

RPA is unique in requiring real-time tracking and fusion for large numbers of highly-maneuverable ground (and air) targets in a target-dense environment. It uses inputs from diverse onboard and offboard sensors including an advanced MMW "Longbow" radar, FLIR, BCIS (Battlefield Combat ID System, for IFF), Improved Data Modem (IDM) for teammate sensor data, various electro-optical sensors (RF, IR, EO, and Laser), offboard sensor sources reporting via the Joint Tactical Information Distribution System (JTIDS, for JSTARS, AWACS, E2C and FAADSC21), Generic Intel Receiver (GIR for TADIXS-B/TRAP, TIBS, and TRIXS), and pre-mission EOB. Inputs are often time-delayed and intermittent, and involve large ranges of positional error ellipses and data uncertainty. The various sensors run at a wide range of update rates. While the overall strategy is for track-to-track fusion, sensor inputs are not limited to strictly kinematic tracks with known velocity estimates, but also include contact reports, line-of-bearing reports, and group reports.

RPA missions are concerned with a large area of interest (300 km by 300 km) containing as many as two hundred objects to be tracked. Target class and identification data is tightly integrated with spatial and kinematic data throughout the processing. Because of mission constraints, "active" onboard sensors that provide the most timely and accurate data are often shut down. Specific sensors may degrade because of weather, clutter, range, target velocity, or lack of target motion (moving target indicator radar).

1.2 The Assignment Problem -- A Bottleneck

Because of platform constraints, RPA processing hardware for track fusion was severely limited. Indications were that the track and data fusion application would have access to two Silicon Graphics R4400 CPU processors (~125 MIPS each), of which one would be assigned primarily to data inputs and outputs, and the other CPU performing most of the data fusion. Despite the hardware constraints, 10 Hz was chosen as the RPA data fusion update rate, to keep up with the sensor rates and cope with highly maneuverable ground targets. This meant that in each 0.1 second, data fusion (one CPU) had to fully process up to two hundred inputs from each of every one of the sensors. All the data had to be fused, with a 1 Hz output of a final, best-estimate, track file, containing positional, kinematic, and a vast store of identification and other attribute information for every object situated in the area of interest. A considerable period of investigation was carried out by ATL to determine the primary processing bottleneck, and to estimate the quality and runtime performance of all the candidate conventional algorithmic approaches to each functional category within the scope of track and data fusion. These functional categories included the traditional tasks of preprocessing, positional and kinematic estimations, data association, track assignments, track fusion, track identification, and track maintenance.

Results of the investigation indicated that track assignments constituted the most critical bottleneck. Here, possibly hundreds of known "central track files" have to be correlated with similar numbers of new reports coming from the sensor. Then decisions are made as to which specific central track is represented by which specific new report. A brute force assignment algorithm, looking at all possible combinations so as to achieve a globally optimum assignment of all tracks, is an "n-factorial" problem ("n" refers to the number of central track files), which is intractable for this many targets. Over the years, a number of algorithms have been proposed, including: (1) Dynamic programming methods, having computational complexities in the neighborhood of $O(n^4)$ and possibly somewhat less, (2) Multiple Hypothesis Tracking algorithms, of which many different versions abound, which attempt to manage the n-factorial problem by drastic pruning of the tracks that are subjected to multiple hypotheses (alternative pairwise combinations), and also by drastically restricting the number of updates over which hypotheses can be maintained, and (3) Nearest Neighbor algorithms which make no attempt to find globally optimum matches, allowing each track to be assigned to the most similar report of those still unassigned.

These "conventional" algorithms tend to either perform rapidly with large numbers of errors (for example, Nearest Neighbor assignment algorithms), or, alternatively, produce correct assignments after computational delays not compatible with system requirements (for example, Multiple Hypothesis Tracking algorithms). Both cases may lead to the output of erroneous tracks, either due to incorrect assignments or to incomplete assignment processing. It was our conclusion that neither approach could be effectively used for RPA.

1.3 Alternative Assignment Algorithms

ATL's focus then shifted to the investigation of alternative assignment algorithms possessing the potential for both real-time processing and reliable results, within the framework of RPA. When none was apparent, we began an effort to develop one, that eventually was referred to as the "Competition Algorithm".

Competition Algorithm: An algorithm was sought that was *globally nearly optimal* in assignment performance, yet ran to completion fairly rapidly. The algorithm, developed by the author, has a computational complexity of $O(n^2)$. It requires little in the way of overhead and the total C-code is less than a single page. While iterative in nature, it is guaranteed to converge to its solution within a fixed (ten or less) number of runs. Having a neural network architecture, it is amenable to total parallel processing, with a computational complexity then of $O(n)$, or linear. The name

"Competition" was derived from the manner in which the algorithm proceeds to solve the optimization problem: It forces the central track files to compete with the new sensor reports for minimal costs, each seeking to achieve a global minimum from their own perspective. The Competition algorithm, however, is not fully developed to its most efficient version, has not been extensively tested, and has not yet been used in a real-world application. It was at this point that we became aware of the existence of the Jonker-Volgenant-Castanon (JVC) algorithm. Because the JVC was described as a particularly fast algorithm and superior in results to the Nearest Neighbor, and because it was a completed, coded, public domain, validated algorithm, we felt impelled to evaluate its usefulness for RPA.

2. THE JVC ALGORITHM

The JVC is a later rendition of an algorithm developed in stages over many years. A forerunner, the JV (Jonker-Volgenant) algorithm, was significantly improved by David Castanon at the request of Oliver Drummond. The JVC was documented in 1990 [2] along with complete FORTRAN program code. We proceeded to convert the code to C language and integrate the resulting program into our simulator for Monte Carlo testing and comparisons with both the Nearest Neighbor and Competition algorithms. We were able to find no reports of previous experience using JVC in an RPA type environment. The original documentation often refers to "sparse cost matrices" as the application for JVC, but in RPA, target numbers and proximities are anticipated to be far greater than in more traditional tracking applications. The original application of JVC was to deal with the tracking of intercontinental ballistic missiles where targets largely traveled in straight lines, at long range (so that apparent movements were small), widely separated, and the processing update rates could be very slow. We needed to know how well the JVC would scale up to higher target densities and to high levels of target maneuverability, both in terms of accuracy and run-time performance. The suitability of JVC for RPA environments was unclear.

JVC Strategy: The JVC code is sequential in nature, containing two phased functions: The first is a version of the "Auction" algorithm[3], followed by a modified version of Munkres' algorithm[4]. The Auction is known to progress quickly at first, then slowly and asymptotically reaching an optimal solution if permitted to proceed to completion. In contrast, the Munkres begins slowly, but ends rapidly as more and more possibilities are eliminated, also producing a globally optimal assignment result. The hope was that a combination of these two algorithms would achieve both good runtime performance and nearly optimal results. The strategy was to make as many quick and easy assignments as possible with a short invocation of the Auction, then to dispose of the remaining, smaller subset of track decisions with the Munkres. In many ways, the Auction algorithm resembles simulated annealing; along the way, the Auction may make decisions that are incorrect or non-optimal. Therefore, the interruption of the Auction before completion meant that optimality was no longer guaranteed and erroneous assignments might be handed off to the Munkres, which represents an efficient version of dynamic programming.

This report focuses on the Auction, since the Munkres is a relatively straight-forward, well known algorithm that performs a globally optimum assignment whose correctness is dependent only on the quality of it's inputs. The Auction, in contrast, is not so well known, its mechanism of action appears to be poorly understood by much of the tracking community, and its results can vary considerably depending on conditions, parameters, and usage.

2.1 The Auction Algorithm in JVC

Auction Assignment Mechanism: There also are many versions of the Auction portion of JVC. They have been described as "Jacobi-like relaxation methods for solving a dual problem." It is important to understand the mechanism of action of the Auction algorithm, in order to better assess its suitability for specific types of applications. I will attempt to do this, in a descriptive, non-mathematical manner.

Consider a typical implementation of the Nearest Neighbor algorithm to track assignments:

1. Begin with the list of central track files; Select the next central track file to be assigned.
2. Examine the list of all new reports (gated, usually) eligible for assignment to the central track file.
3. Choose the new report closest to (most highly correlated with) the central track file.
4. Make this assignment, irrevocably.
5. Remove that new report from the eligibility list for the remaining central track files.
6. Proceed to the next central track file on the list of those that remain unassigned.

	NEW REPORTS			
	1	2	3	4
TRACK A	53	61	56	70
TRACK B	56	57	60	59
TRACK C	96	98	81	99
TRACK D	66	68	54	77

Figure 1. Initial cost matrix for four central track files (A - D) and their corresponding four new reports (1 - 4).

7. Repeat the above process until no more tracks remain or one of the lists is exhausted.

The Auction algorithm differs from Nearest Neighbor in several key ways. Its basic steps are outlined below:

1. Enter all the central track files into a first-in, first-out processing queue. Select the track at the head of the queue.
2. Find the Two Best Reports for this Track: The Auction considers more than the one closest new report as done in step 3 of the Nearest Neighbor. It looks at the *two* closest reports. For example, refer to Figure 1 which represents the initial cost matrix for four hypothetical central track files and their corresponding new reports, where the goal is to minimize costs (smaller is better). Initially, the two best choices for track "A" are assignments to either report 1 (cost = 53) or report 3 (cost = 56).
3. Determine "Best" by Computing Assignment Prices: Each report is associated with a variable referred to as its "dual price". The dual price is initially set to zero. For each possible pairing of a report with this central track file, the Auction algorithm computes the difference between the correlation cost less the dual price, and this result will be referred to here as the "*assignment price*". The two best assignment prices are selected. In this example, the two best are report 1 (its assignment price is 53 - 0) and report 3 (its assignment price is 56 - 0).
4. Compute the Difference in the Two Currently Best Assignment Prices: It then computes the difference between the two assignment price results obtained in step 3: (53 - 0) - (56 - 0) = -3.
5. Assign the Track to the Best Report: Track "A" is then assigned to the best report, report 1, since its assignment price for Track A is the smallest value (53 - 0).
6. Compute a New Dual Price for the Report to Which the Track is Assigned by Adding the Results of Step 4 to the Old Dual Price for that Report: (0 -3) = -3.
7. Handle the Bumped Track, If There Was One: If the Report to which the current track was just assigned had previously been assigned to some other track, now that other track must be reassigned to a different new report before proceeding to process the remaining tracks. Place the bumped track at the head of the processing queue.

	DUAL PRICES			
	REPORT 1	REPORT 2	REPORT 3	REPORT 4
INITIAL VALUES	0	0	0	0
STEP 1 TRACK A ASSIGNED TO REPORT 1	-3	0	0	0
STEP 2 TRACK B ASSIGNED TO REPORT 2	-3	-2	0	0
STEP 3 TRACK C ASSIGNED TO REPORT 3	-3	-2	-18	0
STEP 4 TRACK D ASSIGNED TO REPORT 1	-4	-2	-18	0
STEP 5 TRACK A REASSIGNED TO REPORT 1	-10	-2	-18	0
STEP 6 TRACK D REASSIGNED TO REPORT 2	-10	-4	-18	0
FINAL TRACK B REASSIGNED TO REPORT 4	-10	-4	-18	-2

Figure 2. Successive dual price values for the example of Figure 1, following each step of the algorithm. A step represents the assignment of one central track file. This illustrates the dynamic nature of the dual prices.

8. Repeat for the Next Track in the Queue: This process (steps 1 through 8) is repeated until some threshold number of Auction iterations is completed, at which point all remaining unassigned tracks and reports are transferred to the Munkres algorithm.

Figure 2 illustrates the step-by-step changes in the dual prices for the example of Figure 1, where each "step" refers to the handling of the next track in the processing queue.

Revocable Assignments: In the Auction algorithm, the initial assignment of track "A" to report 1 is not considered irrevocable, as it would in the Nearest Neighbor algorithm. At any time, if any central track file decides that its best choice for assignment is a report that had previously been assigned to some other central track file, the Auction algorithm invokes the process of 'musical chairs': A new assignment is made, displacing the previously assigned track, which now must seek some alternative new report.

Each assignment action results in an adjustment of the dual price for the report that is newly assigned to the track. The adjustment is computed by adding, to the old dual price of the report, the difference in assignment prices between the two current best choices for the central track file. This means that the dual price for any report may be *dynamically* modified over time as tracks are assigned, bumped, and re-assigned.

Interpretations: It is common to describe the dual price by terms that evoke similarities to economic actions like "bidding" and "prices" and "profits". Readers of this literature are left with the suggestion that dual prices reflect the *accumulated* penalty that would have to be paid in order to bump, or undo, a previously made sequence of previous assignments, i.e., the price that would have to be paid in terms of the estimated loss in global costs due to the reassignment of a number of tracks to accept their second (or third . .) choice of new reports instead of their first. But this is not an accurate interpretation of the term, because dual prices dynamically change. Earlier assignment decisions are made, in part, on the basis of dual price values that later become invalidated by dual price updates and track reassignments. Those earlier dual prices are not then recomputed using the new, updated ones from other reports. These now erroneous values, however, continue to be propagated and used as though, once made, they would never be subject to change. This dynamic changing of dual price values may possibly produce assignment errors. Moreover, because of this dynamism, dual prices certainly don't represent the accurate cost of a cascade of reassignment decisions. For example:

In Figure 2, track A is first assigned to report 1, and report 1 acquires a dual price of -3. Track B is then assigned to report 2 which acquires a dual price of -2. Following this, track C is assigned to report 3, for which a "dual price" of "-18" is computed. This value of "-18" for report 3 was based upon the assumption that:

- The dual price for report 1 is "-3", which in turn is based upon the initial first choice dual prices of "0" for report 1 and "0" for report 3 that were used when computing the assignment of track "A" to report 1. The new track C *assignment* price for track C/report 1 will be (96 - (-3)) = 99.

- The previous dual price for report 3 is "0" . Therefore the new assignment price for track C/report 3 will be (81 - 0) = 81.

- The new dual price increment for report 3 will now be their difference, (81 - 99) = -18. Note that these dependencies now form a cyclic loop: The dual price of report 3 is "-18", which is, in turn, based upon the assumption that the dual price of that very same report 3 is really "0". In other words: The current values for an initial set of parameters is computed; Then a second set of parameters have their values computed based upon the initial parameter set; Following this, the initial parameters change their values, but there is no updating of the values for the second set of parameters; Subsequently, this mixture of new and old data is used for decision making as though they are all current and consistent with one another.

In subsequent steps, computational validity is even more stretched: The dual price for report 1 is decremented to "-4", and later still to "-10". Yet the report 3 dual price value of "-18" remains constant, even though it was based upon the assumption of a report 1 dual price of -3, which has now changed. All pretense at computational correctness goes out the window when, eventually, reports 1 and 3 are *compared* with one another via their assignment prices which are *both* incorrect *and* dependent on one another, and those resulting assignment prices are then used for new assignment decisions.

The point is that, while economic interpretations have a certain appeal, they do not logically account for the computations that are in fact performed. Nonetheless, dual prices are used effectively by the Auction algorithm as a pseudo-quantitative guide in making assignment decisions so that the global assignment cost is generally directed toward an optimal (in this case, minimal) sum for the correlations/costs of the assigned track/report pairs. As we will see below, there are certain conditions that minimize the probability of errors.

Sources for Errors: Bertsekas[3] holds that the Auction algorithm will always terminate and will provide an optimal assignment solution, provided that (1) A small, additional increment of e is made to the magnitude of the dual price updates, (2) $e < 1/n$ (where "n" is the number of tracks), (3) The cost values are integers, and (4) The Auction algorithm is processed all the way to completion (not interrupted). In this regard, note that the JVC does not require that the Auction proceeds to completion, and uses floating point cost values rather than integers. There is an additional factor that contributes to non-optimality (assignment errors) by the Auction algorithm during the intermediate processing (iterations), *before* final completion:

- Dependency on Sequential Order: Since assignments are considered and implemented one track-report pair at a time, outcomes of this Auction algorithm are order-dependent. Obviously, bad choices will result that later *may* be corrected for as the algorithm finds an even better assignment with a central track file considered later in the processing sequence. These bad choices represent errors of assignment, will degrade the final outcome of the assignment process, and destroy optimality, *if* not corrected for eventually. If the algorithm is allowed to run to completion, the corrections will ultimately be made; otherwise, the errors become permanent and a flawed list of remaining unassigned tracks and reports is sent to the Munkres for "optimal" assignments.

Similarities to Finite Difference Tables: If the original, new report dual prices are viewed as true values, or terms, of some function, the accumulated dual prices can be viewed as first, second, . . nth order finite differences of those values. In this case, however, the nth order differences may feed back into the *original* dual price terms, recursively, *without* updating the previously computed first, second, . . order differences accordingly, as was discussed in the previous example. This is equivalent to introducing errors into the original dual prices, as though one were dealing with sampled data and attempting to reconstruct or interpolate the original function. The important point here is that errors introduced into the terms of an original function are *greatly magnified* by computing the finite differences[5] of those terms. Generally, users are advised not to compute anything beyond fourth order differences, beyond which errors are too large for the results to be meaningful.

Tracking Implications: What this implies for multitarget tracking is that large numbers of tracks in close proximity (or, having similar correlation values) will have strong probabilities of erroneous assignments using even a few iterations of the sequential Auction algorithm. If the correlation values are purely spatial, this error tendency will be very strong. If the correlation values incorporate spatial and kinematic data, probably somewhat less so, since the tracks will thereby be somewhat more separable. If the correlation functions are multidimensional in nature, that is, incorporating spatial, kinematic, and identification data, they will more likely be widely distributed and thereby minimize the order of the differences and the competition between different central track files for the same new reports. However, there is no formula available to judge in advance the critical limits of these parameters dealing with number of tracks, spread of the correlation values, target density, and so on.

Customizing Parameters: To make matters more confusing, the user can constrain the maximum number of iterations permitted for the Auction portion of JVC. In addition, there is a parameter that allows the user to stipulate that certain amounts of non-optimality are acceptable, in which case he can influence which tracks are processed by the Auction vs those to be processed by the Munkres. It is therefore possible to conceive of parameter settings that, for certain target environments, will happen to send all or almost all tracks to the Munkres for processing, in which case the JVC runtime performance will degrade considerably and amount to a pure dynamic programming solution. Or, in contrast, the settings can cause prolonged processing by the Auction, in which case there may be generated large numbers of errors that will not be corrected unless the Auction runs to completion, similar to the simulated annealing approach. Exactly what particular parameter settings will be most effective will necessarily depend upon the chosen application scenarios and the computational complexities of the two component algorithms.

Computational Complexity and Run-Time Performance: The Auction algorithm, in a 'best case' scenario (where the cost matrix is very, very sparse and there is no assignment contention for any track) will have a computational complexity of $O(n^2)$, where "n" represents the number of central track files in a square matrix. Bertsekas describes the "worst-case" run-time complexity of the "pure" Auction algorithm (similar to the version we have used) as $O(A(n + nC/e))$, where A represents the number of possible assignment arcs, C is the maximum cost value, n is the number of tracks, and e is a scaling factor usually set to less-than C/n. A run-time complexity of this magnitude is unacceptable for an RPA type environment, where there are large numbers of tracks, many sensors, and very rapid update rates. The desirable strategy must, therefore, be one of a very brief, limited use of the Auction for elimination of the easier, widely separated tracks, so the Munkres will have relatively few tracks with which to contend. This presupposes that the target environment (target density) is one where this is possible.

3. THE NEED FOR METRICS

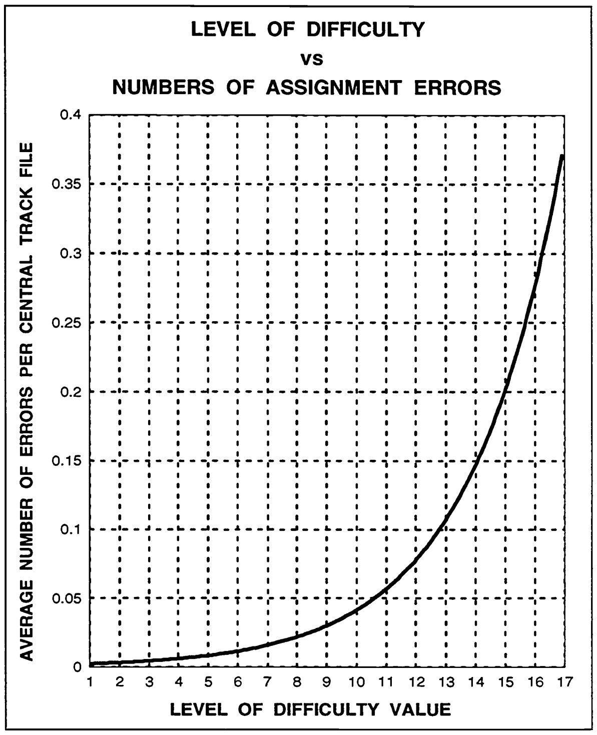


Figure 3. The number of assignment errors grows exponentially as the level of difficulty of the scenario increases.

In the literature, JVC is recommended for application to "sparse matrix problems" [2], where a sparse matrix is defined as one which contains elements that disallow assignment. But there are no quantitative guidelines available to partition sparse from non-sparse matrices with reference to the suitability of any particular scenario and/or its cost functions. How many elements have to be unassignable? How evenly distributed does this 'sparseness' have to be? Is it truly necessary for the elements of the matrix to be unassignable, or is it sufficient to have widely separated cost values?

"Level of Difficulty" as a Metric: It is absolutely essential to determine the limitations of JVC in terms of the target environment as reflected in the cost matrix, and also whether or not a particular application such as RPA has gone beyond the bounds of a "sparse assignment problem" that is suitable for JVC. More generally, we need to be able to assess the appropriateness of any assignment algorithm. An algorithm that outputs no errors for a scenario characterized by (1) few tracks (2) that are traveling slowly (3) at constant speeds (4) in straight lines and (5) far apart from one another is not necessarily better than another algorithm that produces several errors with a scenario characterized by (1) many (2) close targets (3) traveling in erratic paths (4) at high speeds (5) that change unpredictably. Consequently, we devoted considerable time to this objective, defining a formula that readily permits the fair comparison of one algorithm with another, and provides assessment of the level of difficulty of any scenario, partial or total.

Level of Difficulty Measurement: Logically, the number of track assignment errors should increase proportionally as the average number of targets per unit volume exceeds some minimal threshold. Similar claims can be made regarding other factors, such as the target scalar speed, target maneuverabilities, sensor resolutions and uncertainties, sensor update rates, target ID, and any others that might impact the magnitude of the ambiguity facing assignment algorithms. But these are the very same factors that should be incorporated into the correlation "similarity measure" (cost function) that ends up in the cost matrix. It seems reasonable, therefore, to use the cost function itself as the main contributing input to a level of difficulty measure. This would have the added benefit of reducing the computational requirement for computing such a measure. Keep in mind that the cost function is a heuristic, and there are no independent measures for judging the "goodness" of any cost function other than estimates based upon empirical tests that look at resulting assignment error rates: For a given scenario or set of scenarios, the best cost function will lead to smaller global costs (smaller is better), and to fewer assignment errors.

In deciding how to apply the cost functions to come up with an estimate of the level of difficulty, it seems obvious that it must involve some comparison between two quantities:

- Quantity A is based upon the cost function between a central track file and the sensor report that is known to represent (be derived from) its ground truth i.e., its correct assignment mate, and
- Quantity B considers the costs of all pairs of the central track file with all reports other than quantity A.

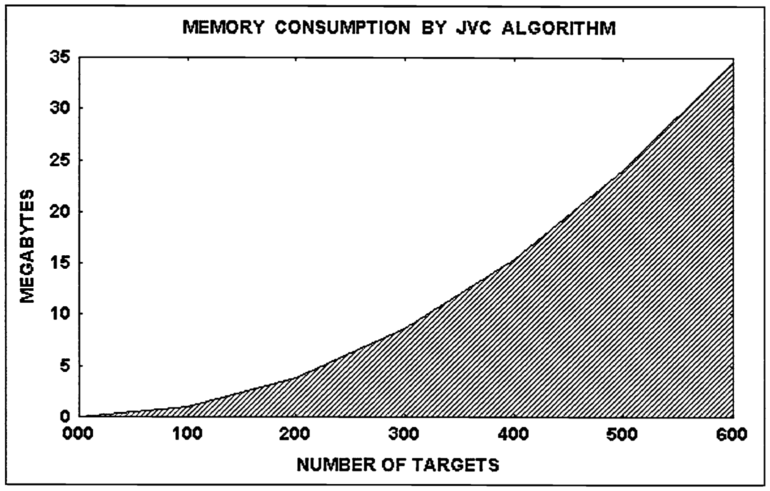


Figure 4. Processing memory requirements are minimal for the JVC.

Inspection of the mechanism of action of assignment algorithms leads to the common sense conclusion that errors result from contention between these two quantities. The formula we use as a measure of the level of difficulty was designed to quantitatively assess and compare the magnitude of this "contention". It provides an exponential increase in assignment errors as the level of difficulty measurement increases. The results of testing against simulated scenarios is shown in figure 3. In making use of this formula when testing in the system simulations, it is important to prohibit the actual fusion step between central track files and their assigned sensor reports. Otherwise, due to any incorrect assignments, the errors of one iteration will be propagated into all ensuing iterations and falsely report the accumulated error rates in a way that does not truly or fairly reflect the actual performance of the assignment algorithm or accurately assess the level of difficulty. Note that the formula uses the cost function values to compute the level of difficulty, and assumes that these adequately reflect all the factors that are important for this computation. In truth, however, this is not usually the case. Most systems only make use of inter-target spatial distances and/or kinematics to compute the costs. This observation on the level of difficulty formula suggests that some additional steps can be taken to improve the performance of assignment algorithms, since that performance is very dependent upon the nature of the cost values.

Heuristics for Improving Performance: Since the level of difficulty formula strictly depends upon cost functions, all attributes that can potentially contribute to the cost function can be used to impact the level of difficulty measurement. This implies that there are many opportunities for improving performance for any assignment algorithm by reliably improving the spread, or separation, of the costs that go into the cost matrix, thereby diminishing the level of difficulty that is presented to the algorithm. Several heuristics for this purpose are listed below. The first two of these impact the similarity measures that are incorporated into the cost matrix, producing correlation values for targets that are more likely to be sufficiently separable than they were previously, and will therefore result in less contention for assignments. The last, which forms the basic strategy of JVC, acts by limiting exposure of each component algorithm(s) to only those tracks or scenarios that, by nature, are more likely to be easily or rapidly separable by that particular algorithm:

1. Increasing the sensor and data fusion processing update rates, so new reports are relatively much closer to their corresponding central track files, or
2. Adding highly valued features to the correlation functions, such as identification data, to create more powerful similarity functions, or
3. Using Nearest Neighbor or Auction approaches to deal with large numbers of targets whose spatial and kinematic attributes are naturally, sufficiently different (leading to small target densities) so as to permit easy differentiation into widely separate categories -- And using dynamic programming or MHT or Munkres-type algorithms where there are small numbers of targets which are closely packed.

Obviously, the preliminary decisions regarding the choice of an assignment algorithm and the formulation of a metric (the level of difficulty) for evaluating performance led to the next phase of our program, consisting of tests and analyses to validate our decisions.

4. THE TESTING ENVIRONMENT

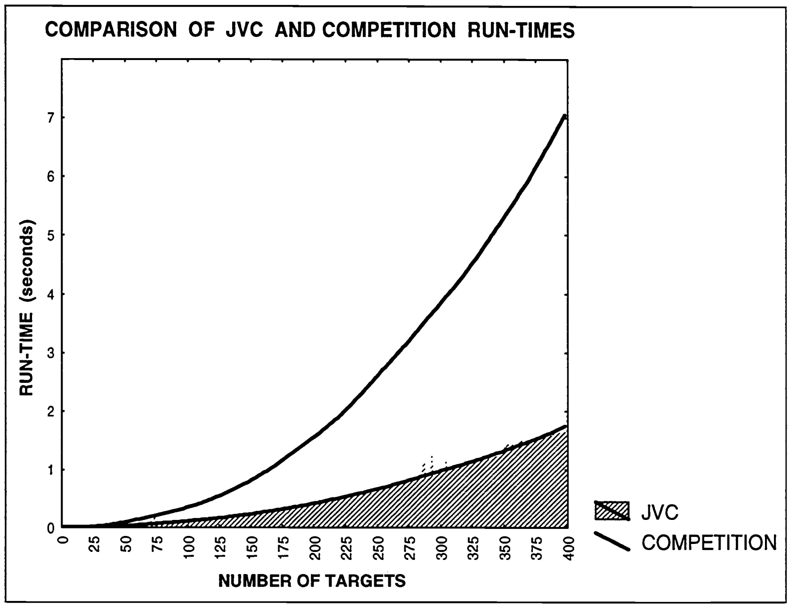


Figure 5. The JVC algorithm runs significantly faster than the Competition algorithm when dealing with larger numbers of targets.

Extensive simulator-based testing of the JVC algorithm has been on-going for several years at ATL as part of an overall program for data fusion algorithm development and evaluation. This will lead up to field testing, scheduled to begin during FY97. The simulator-based testing involved the use of Monte Carlo runs as well as fixed scenarios. The current phase of RPA emphasizes mobile ground targets (especially tanks). Testcases were constructed to specifically explore the impact of intersecting tracks, spatial uncertainty, attribute fusion, contributions of target class and ID data, various degrees of maneuverability and speed, target density, and sensor update rates. Measurements were made of memory consumption, assignment errors, global costs, numbers and sizes of clusters, run time performance, and the level of difficulty for individual tracks, clusters, updates, and whole scenarios. Comparisons were made between the JVC algorithm, the Competition algorithm, and a Nearest Neighbor assignment algorithm. Particularly in the Monte Carlo runs, tests were conducted over a wide range of parameter values, even beyond the limits of realism, in order to sufficiently challenge the algorithms. This was especially true for the degree of maneuvering and for target densities.

4.1 Test Results

JVC Memory Consumption: Judging from the original Fortran code, a conscientious attempt was made by algorithm designers to minimize the run time utilization of memory. In our testing with 200 simultaneous targets, about 3 MB of memory was required by the JVC, increasing proportionally to 34 MB for 600 targets (Figure 4).

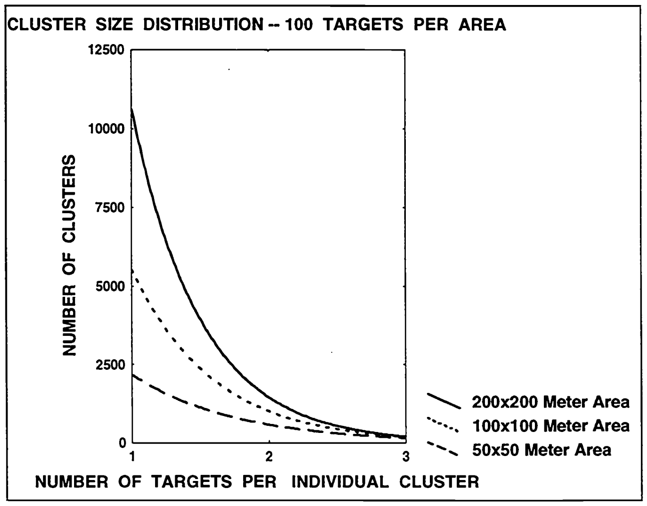
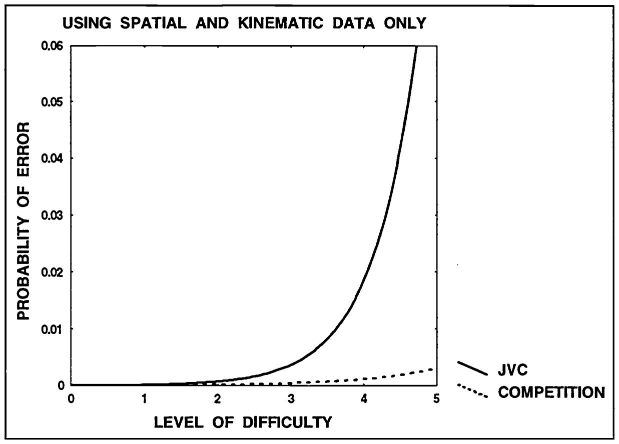


Figure 6. Even when dealing with very dense target environments consisting of many hundreds of closely packed targets, the data fusion algorithms are able to partition almost all the targets into clusters containing less than four targets.

Run-Time Performance: Studies of assignment run time requirements were performed on a SUN Sparc 10 workstation (Figure 5). They demonstrate that the JVC is clearly much faster than the Competition algorithm. Note that when dealing with target clusters of size less than 25, the run time performance differences are negligible. Studies of the effectiveness of the RPA data fusion clustering algorithms show that even in unrealistically dense target environments, the *average* size of target clusters is less than four (figure 6), in which case there is no distinction in run time performance between the two algorithms. Of course, the average size is not always the case, in which case the slower Competition algorithm could conceivably fail to complete the assignments in real-time, despite the ten-fold speed up in data association achieved by our unique linear-complexity clustering algorithm. Studies have shown that, in a typical simulated RPA



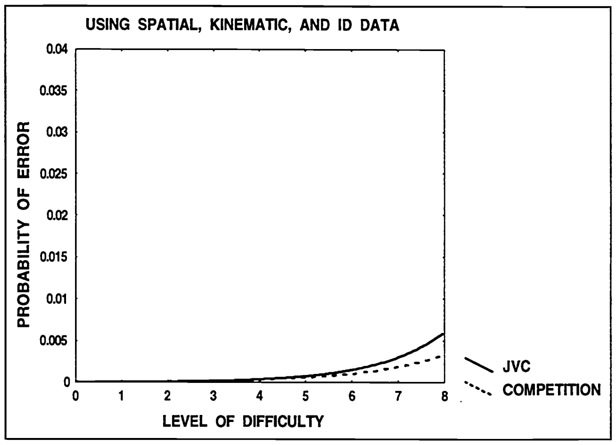


Figure 7. The Competition algorithm clearly is better able to avoid assignment errors when dealing with very difficult scenarios, even when only spatial and kinematic data is available.

Figure 8. Both algorithms significantly improve performance when class and ID target attribute data is available. Again, the Competition performance is better.

mission environment, the JVC assignment algorithm satisfies the run time constraints of dealing with 200 targets and a full complement of sensor inputs.

Assignment Errors: Figure 7 illustrates the relationship between probability of assignment errors vs the track level of difficulty, for both the JVC and Competition algorithms. In this case, the Monte Carlo simulation used only spatial and kinematic track attributes, and did not use target class and ID data. Note that for very difficult scenarios, the Competition algorithm significantly outperforms the JVC. In figure 8, where class and ID track attributes are now added, both algorithms have significantly improved their performance and are virtually indistinguishable. JVC, by the addition of ID attributes, has overcome the advantage of the Competition algorithm. In Figure 9, the number of assignment errors for JVC is plotted against the number of ground targets within a very small fixed area of interest. When evaluating these examples, it is important to understand that the target densities used are far in excess of what is reasonable or, in some cases, even physically possible. It would be equivalent to hundreds of vehicles maneuvering radically at high speeds and only meters apart. These densities were used to stress the algorithms and seek out their breaking points. Note that adding kinematic data significantly improves results over the use of spatial data alone, and the impact of target class and ID data is equally pronounced. Similar studies, using a Nearest Neighbor algorithm, resulted in very large numbers of assignment errors (20 percent or more).

These and many additional fixed scenario and Monte Carlo studies were used at ATL to compare various track fusion algorithms that we evaluated for possible use in RPA. What remained was (1) to determine the typical levels of difficulty that characterize the simulated RPA scenario environments, and (2) to test the JVC against those scenarios, so these Monte Carlo studies can be validated as good predictors of algorithm performance.

Estimations of RPA Levels of Difficulty: Documented RPA testcases included consideration of many variations of the following:

- 10 to 200 ground targets,
- Inter-target distances of 11 to 500 meters,
- Mixtures of sensor types, including JSTARS, TRIXS, JTIDS, GIR, ASE, TAS, and EOB,
- Data dropouts and intermittent reports,
- Stationary targets and ground target motion up to 80 Km/hour,
- Various ranges to target and various sensor resolutions,
- Widely ranging maneuvering vs straight line paths,
- Various degrees of path intersections, and
- Various amounts of class and ID attributes

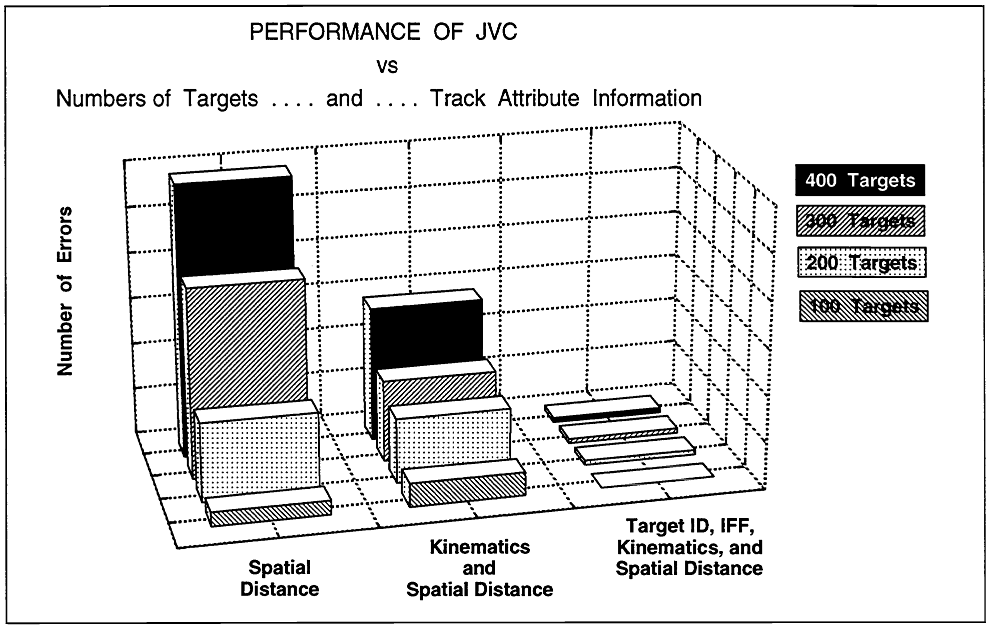


Figure 9. The use of increasing amounts of valuable target information enables the JVC to significantly improve performance, resulting in a marked reduction in errors.

Some of the testcases were designed to stress the algorithms, and others were designed to approximate specific mission requirements. A detailed report[6] is published separately in an internal Lockheed Martin document.

Average level of difficulty values for these testcases ranged from 1.000 to a maximum of 1.614, with 1.000 representing the situation where there is no assignment contention (minimum possible value). Error rates for these same scenarios ranged from 0.00% to 6.419%. When instances of errors are separately tabulated, their level of difficulty ranges from 2.059 to 9.956 units. The errors, along with their corresponding high level of difficulty values, occur in the most severe scenarios, where targets are at a distance of 250 Km, the reports are delayed in time and associated with large sensor error ellipses, not accompanied by class or ID attribute data, and where the track data comes from a single sensor only, rather than from multiple sources that are fused.

Unfortunately, the individual test data tracks have not yet been partitioned according to their level of difficulty, and so the RPA data deals with averages for all tracks in a scenario. Perusal of the data suggests, however, that (1) the Monte Carlo estimations of the relationship between level of difficulty vs error probability hold true for these simulated RPA testcases, and (2) simulated RPA scenarios conform with our own view of a sparse matrix, in the sense that levels of difficulty are found to be almost entirely less than 2.0 units, and mostly less than 1.2 units. This fits well with the observation that, in running the JVC, the Munkres portion is seldom invoked, with two iterations of the Auction serving to completely process all the tracks.

5. DISCUSSION AND CONCLUSIONS

There are several key issues we sought to resolve with our studies, with the following results:

- **Sparse Matrix**: As previously discussed, JVC is recommended for application to "sparse matrix problems", but there are no quantitative guidelines available to partition sparse from non-sparse matrixes with reference to the suitability of any particular scenario and/or its cost functions. Our goal became one of formulating a method for quantitatively evaluating these inputs and thereby determining suitability of any particular algorithm. In other words, we sought to be able to describe an algorithm in terms of its capability for handling a certain range of input "levels of difficulties" which would take into consideration all contributors to the difficulty of the track fusion task (in this case, the task of track assignments), such as number of targets, velocities, maneuverabilities, sensor resolutions and uncertainties, update rates, target densities (proximities), and other similar attributes. The result of development and testing in a Monte Carlo environment was a formula that is easily computed and proved to be a reliable, robust measure that is generally applicable regardless of scenario or type of algorithm being tested.

- **Characterization of RPA Scenarios:** In order to determine if the JVC algorithm is a good fit with the scenarios of RPA, we had to estimate the scenarios that are mission compatible with RPA, simulate those scenarios, and apply our level of difficulty formula. The results indicated that the algorithm/problem match is a good one, i.e., the levels of difficulty of estimated and simulated RPA scenarios are well within the capabilities of the JVC algorithm. Of course, *it is premature and fraught with danger to surmise at this point what the actual missions, scenarios, and sensor inputs will be like for a system not yet fielded.* But this is not an unreasonable approach for an algorithm evaluation phase, especially where the results are so unequivocal.

- **Validation Through Simulation Tests:** The JVC assignment algorithm was applied to simulated RPA scenarios, evaluated for error rates and runtime performance, and compared with alternative assignment algorithms. The results showed excellent correlations between error rates and track levels of difficulty. Moreover, the overall performance of JVC was characterized by few assignment errors and excellent runtimes, despite some very difficult scenarios. The Monte Carlo studies and the formula that resulted from them were validated.

- **Heuristics for Performance Improvements:** Workers in this field often herald the theoretical benefits of sensor fusion and especially the possible advantage of using target identification information in tracking. This current work clearly confirms and quantifies those benefits, and gives us a method for identifying key factors that can improve performance through their impact on the separations of the cost function values.

Our conclusion is that the JVC is a powerful assignment algorithm for use in multisensor, multitarget track fusion applications. It provides a good balance between requirements for speed of operation vs correct track assignments. The level of difficulty measure we employed has been of paramount importance in the development and evaluation of assignment algorithms as well as in the creation of challenging scenario simulations. Critical to both the assignment algorithms and the performance metrics is the need to formulate cost functions that provide the widest separation possible between contenders for assignments. We have demonstrated the benefits of incorporating target identification data into the cost functions, thereby providing the necessary separation and leading to very low assignment error rates despite the difficult scenarios and conditions we tested against.

ACKNOWLEDGEMENTS

The author is grateful for the insights provided by David Castanon, and for his validation testing of our JVC C-code.

REFERENCES

1. Malkoff, Dr. Donald and Angela Pawlowski, "RPA Data Fusion," in *Proceedings of the 9th National Symposium on Sensor Fusion*, published by Infrared Information Analysis Center, Volume 1, pp. 23-36, September 1996.
2. Drummond, O.E., D.A. Castanon, and M.S. Bellovin, "Comparison of 2-D Assignment Algorithms for Sparse, Rectangular, Floating Point, Cost Matrices," Journal of the SDI Panels on Tracking, Institute for Defense Analyses, Alexandria, VA, Issue No. 4, pp 4-81 to 4-97, Dec. 15, 1990.
3. Bertsekas, Dimitri P., "Auction Algorithms", in *Linear Network optimization: Algorithms and Codes*, The MIT Press, Cambridge, MA, pp 167 - 243, 1991.
4. Munkres, James, "Algorithms for the Assignment and Transportation Problems," J. Soc. Indust. Applied Math., Vol. 5, No. 1, pp. 32-38, March 1957.
5. Scarborough, James B., *Numerical Mathematical Analysis*, The Johns Hopkins Press, Baltimore, pp. 52-54, 1962
6. "Data Fusion Build 4 Test Cases/Reports for the Rotorcraft Pilot's Associate Data Fusion Subsystem", Lockheed Martin internal document ID RPA-LMATL-TCR4, Owego, NY, December 31, 1996.

i

Further author information - D.B.M. (correspondence): Email: dmalkoff@atl.lmco.com; Telephone: 609-338-3985