Distributed- and shared-memory parallelizations of assignment-based data association for multitarget tracking★

Robert L. Popp[a], Krishna R. Pattipati[b] and Yaakov Bar-Shalom[b]

[a]*Alphatech Inc., 50 Mall Road, Burlington, MA 01803, USA*

E-mail: robert.popp@alphatech.com

[b]*Department of ESE, University of Connecticut, Storrs, CT 06269, USA*

To date, there has been a lack of efficient and practical distributed- and shared-memory parallelizations of the data association problem for multitarget tracking. Filling this gap is one of the primary focuses of the present work. We begin by describing our data association algorithm in terms of an Interacting Multiple Model (IMM) state estimator embedded into an optimization framework, namely, a two-dimensional (2D) assignment problem (i.e., weighted bipartite matching). Contrary to conventional wisdom, we show that the data association (or optimization) problem is *not* the major computational bottleneck; instead, the *interface* to the optimization problem, namely, computing the rather numerous gating tests and IMM state estimates, covariance calculations, and likelihood function evaluations (used as cost coefficients in the 2D assignment problem), is the primary source of the workload. Hence, for both a general-purpose shared-memory MIMD (Multiple Instruction Multiple Data) multiprocessor system and a distributed-memory Intel Paragon high-performance computer, we developed parallelizations of the data association problem that focus on the interface problem. For the former, a coarse-grained dynamic parallelization was developed that realizes excellent performance (i.e., *superlinear speedups*) independent of numerous factors influencing problem size (e.g., many models in the IMM, dense/cluttered environments, contentious target-measurement data, etc.). For the latter, an SPMD (Single Program Multiple Data) parallelization was developed that realizes near-linear speedups using relatively simple dynamic task allocation algorithms. Using a real measurement database based on two FAA air traffic control radars, we show that the parallelizations developed in this work offer great promise in practice.

Keywords: distributed-memory parallelization, shared-memory parallelization, data association, assignment problem, task allocation problem, multitarget tracking

★This research was conducted at the University of Connecticut and sponsored by grants AFOSR F409620-93-1-0399, ONR/BMDO N00014-91-J-1950, and NASA NGT-303104.

1. Introduction

In recent years, there has been increasing interest in applying parallel processing techniques to computationally intensive problems associated with multitarget tracking [1–4,10,27,30,34–37]. In general, a *multitarget tracking problem* is one of detecting an unknown number of targets, and estimating target state parameters (e.g., position, velocity, acceleration) using noisy measurements from one or more sensors in the presence of spurious observations and occasional missed detections. To be useful (practical), parallel processing is often necessary for such problems because of the limited computational performance of tracking algorithms on conventional uniprocessor systems [2,27,28,30,36,37]. Hence, in developing a practical solution to a multitarget tracking problem, in addition to solving the conventional *data association* and *state estimation* problems, often an efficient *parallelization* must also be developed.

1.1. Data association

The problem of data association, namely, partitioning measurements across lists (e.g., sensor scans) into tracks and false alarms so that accurate estimates of true tracks can be recovered, has been extensively studied for many years. A probabilistic interpretation is typically given to the data association problem, consequently, the partitioning of measurements into tracks and false alarms may be done in a variety of ways, e.g., (in order of decreasing complexity) the Multiple Hypotheses Tracking (MHT) method [9,39], the multidimensional SD ($S \geq 3$) assignment method [16,17,26,28, 31–33], the Joint Probabilistic Data Association (JPDA) method [9], and numerous two-dimensional (2D) assignment methods [5,6,10,19,24,34–36].

For a given multitarget tracking problem, the decision as to which algorithm to employ in solving the data association problem is typically motivated by several factors, including: (i) tracking accuracy requirements, often determined by the characteristics of the problem and the multitarget scenario in the surveillance region (e.g., passive/active sensors, dense/sparse targets, cluttered, existence of crossing, splitting, and/or merging targets, etc.), and (ii) computational requirements, often determined by choices made with respect to (i) and the computational resources available.

Historically, the MHT algorithm has proven to be a fairly accurate and reliable approach to data association (especially in dense environments) since it is the only approach that can truly provide an optimal data association solution. However, its practicality and feasibility have been hampered since it requires an enumeration of an exponentially increasing number of feasible joint association hypotheses to evaluate probabilities. To be feasible, MHT is typically performed within a time window and on a pruned set to limit its otherwise exploding computational requirements.

On the other hand, assignment-based approaches to data association, such as SD and 2D assignment, are discrete mathematical optimization formulations of the data association problem and have been shown to be practical and feasible alternatives to

MHT. For SD assignment, the main challenge to overcome is that of solving the ensuing NP-hard multidimensional assignment problem [29]. However, satisfactory tracking and computational performance can be realized using existing SD assignment algorithms that provide good *suboptimal* solutions, of quantifiable accuracy, and in pseudo-polynomial time [16,17,26,28,31–33].

For sparse environments, MHT, SD, and JPDA are typically not necessary for data association because little is gained in accuracy at the expense of large computational overheads associated with these methods. Single-scan processing (i.e., 2D assignment) is often practical since it has been shown to be a fairly efficient and accurate approach to data association in such environments. With this approach, a state estimator, such as a Kalman filter or an Interacting Multiple Model (IMM) [7] state estimator, is embedded into the 2D assignment framework (e.g., weighted bipartite matching). The state estimator provides a score (i.e., likelihood) of how likely a particular measurement from a given scan originated from a particular target track. The problem is then one of finding an optimal assignment (using any well-known (pseudo) polynomial-time assignment algorithm [5,11,22]), in a (global) *maximum likelihood* sense, of measurements from the latest scan to the most likely tracks from the previous scans.

1.2. State estimation

Inherent to many typical and/or realistic multitarget tracking (surveillance) problems is: (i) the existence of multiple targets in the surveillance region, and (ii) uncertain target dynamics, namely, target motion having constant course and speed (non-maneuvering) segments interspersed with arbitrary maneuvers (often characterized as piecewise constant). The classical Kalman filter – the workhorse of state estimation – has been found to provide only marginal tracking performance for such problems because of the single motion model assumption [7,8]. In contrast, by using different state equations in different time intervals, the IMM state estimator [7] is capable of automatically adapting itself to different modes of target motion, and has exhibited excellent tracking performance for various multitarget tracking problems [8,46]. However, the IMM estimator also imposes an increased computational burden in terms of additional state estimates, covariance calculations, and likelihood function evaluations.

A literature survey [3,4,18,40] of parallel state estimators reveals a plethora of *fine-grained* parallelizations proposed over the years. Typically, many numerically-intensive computations inherent to state estimation are parallelized, such as coordinate transformations, state and covariance estimates, linear algebraic operations such as matrix multiplication, Cholesky (square root) factorization, and inner products. However, for multitarget tracking problems, a fine-grained parallelization proves to be computationally inefficient because it is static (fixed) *within* the state estimator. An alternative, as we propose in this research, is a *coarse-grained* (dynamic) parallelization *across* multiple track states, where, unlike a fine-grained parallelization, excellent computational performance is obtainable independent of numerous factors influencing problem size (e.g., many models in the IMM, dense/cluttered environments, contentious target-measurement data, etc.) [37,38].

1.3. Parallelization

With a data association algorithm in hand and, given a parallel architecture (i.e., shared-/distributed-memory), important tasks to resolve in developing an efficient parallelization are: (i) analyze a dynamically changing time-varying workload, (ii) identify (in)dependent tasks within the computation, and (iii) distribute the workload across the processor set in such a way as to maximize computations (of independent tasks) proceeding in parallel, while minimizing interprocess communication (IPC), synchronization overheads, queueing delays, and load imbalances. Concerning (i) and (ii), even for relatively sparse multitarget problems, such as civilian air traffic surveillance, workload analysis (see table 1) shows that, contrary to conventional wisdom, the *interface* to the data association problem, namely, computing the rather numerous independent gating and IMM state estimates, covariance calculations, and likelihood function evaluations (used as cost coefficients in the 2D assignment problem), is the major computational bottleneck (rather than the data association problem itself).

Concerning (iii), in particular, as it pertains to a distributed-memory system, a *task allocation* algorithm must be considered as part of the parallelization since, without one, such systems tend to quickly become unbalanced [12,13,20,21,41–44]. In general, the objective of a task allocation algorithm is one of finding an allocation of tasks (jobs) across a set of distributed processors to minimize workload imbalances. To date, most task allocation algorithms proposed in the literature can be classified as employing graph-theoretic [30,41], mathematical programming [13], or dynamic (heuristic) [20,21,44] techniques. Techniques based on the former two are often infeasible and impractical for many realistic problems because: (i) enumerative searches in seeking optimal solutions are often required, (ii) task processing and communication costs are assumed static and known, a priori, and (iii) IPC costs among tasks are assumed dominant, with a tendency to over-allocate tasks onto a single processor, consequently creating increased workload imbalances. Alternatively, as we demonstrate in this research, dynamic approaches are attractive because often they are fast, extensible, considerably less complex, and react well to a dynamically changing workload [20,21,44]

1.4. Scope

We begin this paper by describing in section 2 our multitarget tracking problem (i.e., a sparse air traffic surveillance problem based on two FAA air traffic control radars) and our serial multitarget tracking algorithm in terms of its core components.

In section 3, we motivate the parallelization work associated with this research by providing a workload analysis of our multitarget tracking algorithm. In sections 4 and 5, we present brief expositions of both a shared- and distributed-memory parallelization of our multitarget tracking algorithm, developed on a general-purpose shared-memory MIMD multiprocessor system and a distributed-memory Intel Paragon high-performance computer (HPC), respectively. In section 6, we describe in detail several dynamic (heuristic) task allocation algorithms developed for the distributed-memory HPC parallelization of our multitarget tracking algorithm. Using our air traffic surveillance measurement database, in section 7, we provide comprehensive results demonstrating the excellent performance of the described parallelizations. In section 8, we provide concluding remarks.

2. Multitarget tracking algorithm

In this section, we provide a brief discussion of our multitarget tracking algorithm, termed IMM-2D, illustrated in figure 1. The algorithm consists of four primary components: (i) *Obtain Scan Measurements*, (ii) the *Data Association Interface Problem* which, in our tracking approach, is primarily concerned with coarse and fine gating

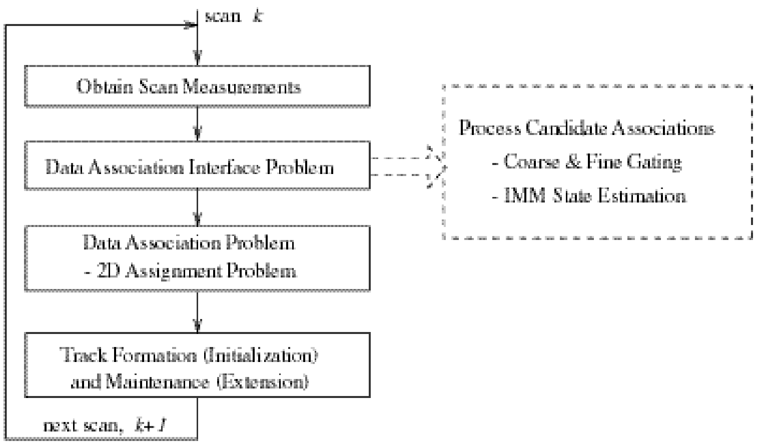

Figure 1. Multitarget tracking algorithm (IMM-2D) block diagram.

(defined shortly) and IMM state estimation, (iii) the *Data Association Problem* (via a 2D assignment problem), and (iv) *Track Formation and Maintenance*. Interested readers can find a more thorough presentation of IMM-2D in [34–37,45,46].

2.1. Obtain scan measurements

Since it is not the main focus of this paper, we omit presenting details concerning the radar sensor systems used in this work and simply refer interested readers to appendix A and [45] for more details. For the air traffic surveillance problem considered in this work, scans of (radar) measurements from two L-band FAA air traffic control radars were reported and collected (approximately every 10 seconds apart). One radar was located at Remsen, NY, where 26 scans were collected, while the other radar was located at Dansville, NY, where 29 scans were collected. A particular scan, say scan k, consists of $M(k)$ detection reports containing a number of primary radar (skin) returns (i.e., slant range, azimuth angle) and secondary (beacon) returns (i.e., slant range, azimuth angle, altitude). Because state estimation is done in a three-dimensional (3D) Cartesian coordinate system, conversion from a radar's polar frame of reference is necessary [9,46].

2.2. Data association interface problem

In this section, we describe the two tasks that must be performed when interfacing with the data association (2D assignment) problem, specifically, *state estimation*, via an IMM estimator, and various coarse and fine *gating* tests. Of these two tasks, the former is the *single most costly* in terms of *processing cost*, and comprises a significant fraction of the workload in IMM-2D.

2.2.1. IMM state estimation

For the air traffic surveillance problem used in this work, the IMM was employed for state estimation. Since it is not the main focus of the present work, we omit a detailed exposition of the IMM here and simply provide a brief discussion of the state estimation problem in general. Interested readers can refer to appendix B and [7,9] for a more detailed presentation of the material.

State estimation provides: (i) a score of how likely a particular measurement from a given scan originated from a particular target track, i.e., it provides a *measurement-to-track* association likelihood that serves as the basis for an assignment cost coefficient in the 2D assignment problem, and (ii) an estimate of the target state. In the state estimation problem, we assume that the target state evolves according to a known linear dynamic model, corrupted by process noise, and driven by a known input, i.e.,

$$x(t_{m_k}) = F(\delta)x(t_{m_{k-1}}) + G(\delta)\upsilon(t_{m_{k-1}}), \tag{2.1}$$

where, in the present work, $x(\ ) = [\xi\,\dot{\xi}]'$ is a second-order kinematic model with 3D coordinates ξ, $\delta = t_{m_k} - t_{m_{k-1}}$ is the time interval, $F(\cdot)$ is the state transition matrix, $G(\cdot)$ is the disturbance matrix, and $\upsilon(\cdot)$ is zero-mean, white Gaussian *process* noise with (known) covariance matrix $Q(\cdot)$. Furthermore, the measurements are linear functions of the target state corrupted by measurement noise, i.e.,

$$z(t_{m_k}, m_k) = Hx(t_{m_k}) + w(m_k), \tag{2.2}$$

where $m_k = 1,\ldots,M(k)$ denotes the m_kth measurement from the kth scan, H is the measurement matrix, and $w(\cdot)$ is zero-mean, white Gaussian *measurement* noise with (known) covariance matrix $R(\cdot)$.

2.2.2. Coarse and fine gating

To reduce the number of likelihood functions to evaluate in the IMM which, in terms of processing cost, is the single most costly operation to perform, *gating* – a pruning technique to filter out highly unlikely candidate associations – is used. A track *gate* is the region in measurement space in which the true measurement of interest will lie, in view of all uncertainties, with some (high) probability [9]. A measurement within the gate is a candidate for association to the corresponding track.

Define the set of *candidate associations* at the kth scan by

$$C(k) \triangleq \{(n, m_k) : (n, m_k) \quad N(k-1) \times M(k)\}, \tag{2.3}$$

where $N(k-1) \times M(k)$ denotes the cross product of the track and measurement sets, with $|C(k)| = N(k-1)M(k)$. In IMM-2D, for each $(n, m_k) \quad C(k)$, a *coarse* gating test is first invoked, denoted by

$$\mathcal{G}_c : C(k) \to \{0,1\}, \tag{2.4}$$

which consists of a *maximum velocity* gate and, if applicable (i.e., measurement falls within track's maximum velocity gate), a high process noise Kalman filter *elliptical* gate. $\mathcal{G}_c(n, m_k) = 1$ denotes that measurement m_k fell within both of track n's maximum velocity and elliptical gates, while $\mathcal{G}_c(n, m_k) = 0$ denotes that measurement m_k fell outside either of track n's gates. For maximum velocity gating, we use the following test:

$$\frac{\| z(t_{m_k}, m_k) - H\hat{x}(t_{m_{k-1}}) \|}{(t_{m_k} - t_{m_{k-1}})} \overset{?}{>} \text{MAX\_SPEED}, \tag{2.5}$$

where the relation "$\overset{?}{>}$" compares the left-hand side term with the right-hand side term and returns true if the former is greater than the latter, and false otherwise. The residual terms in the numerator above are defined as follows: $z(t_{m_k}, m_k)$ is the m_kth measurement in scan k having time stamp t_{m_k}, H is the measurement matrix, and $\hat{x}(t_{m_{k-1}})$ is the track state estimate at time $t_{m_{k-1}}$. The MAX_SPEED parameter is an assumed upper bound on the target's maximum velocity. For elliptical gating, we use a likelihood ratio test, i.e.,

$$-\log\left(\frac{P_D \Lambda_{kf}(n, m_k)}{\Lambda(0, m_k)}\right) \overset{?}{>} 0, \tag{2.6}$$

where $\Lambda_{kf}(\cdot)$ denotes the likelihood function specified in the Kalman filter, the detection probability $P_D = 0.95$ is based on FAA standards [45,46], and the false alarm pdf $\Lambda(0, m_k)$ is the spatial density of the false measurements (assumed uniformly probable over the sensor's field of view) [9].

Define the set of candidate measurement-to-track associations passing the coarse gating test $\mathcal{G}_c(\cdot)$ by

$$\mathcal{L}(k) \triangleq \{(n, m_k) \quad C(k) : \mathcal{G}_c(n, m_k) = 1\}, \tag{2.7}$$

where each $(n, m_k) \quad \mathcal{L}(k)$ requires the application of a *fine* gating test (based on a negative log-likelihood function to be defined in section 2.3). Denote the fine gating test by

$$\mathcal{G}_f : \mathcal{L}(k) \to \{0,1\}, \tag{2.8}$$

where $\mathcal{G}_f(n, m_k) = 0$ denotes that candidate association (n, m_k) is not to be considered in the 2D assignment problem because it is more likely that measurement m_k corresponds to a false alarm than to track n. Conversely, $\mathcal{G}_f(n, m_k) = 1$ denotes that candidate association (n, m_k) is to participate in the 2D assignment problem, where the cost of assigning measurement m_k to track n is $c_{nm_k} < 0$ (defined next).

2.3. *Data association problem*

Data association is the decision process of *linking* measurements (from successive scans) of a common origin (i.e., a target or false alarm) such that each measurement is associated with at most one origin. In IMM-2D, we formulate the data association problem as a 2D assignment problem. Specifically, $M(k)$ measurements from the latest scan k are to be assigned to the $N(k-1)$ most likely existing tracks from the previous scans using a global cost minimization function [9] (based on a maximum likelihood function). Specifically, let $n = 0,\ldots, N(k-1)$ denote a particular track from the set of existing tracks (including a *dummy* track $n = 0$), and $m_k = 0,\ldots, M(k)$ denote a particular measurement from the latest set (scan) of measurements (including a *dummy* measurement $m_k = 0$). Define the binary *assignment* variable

$$\chi_{nm_k} = \begin{cases} 1 & \text{if } m_k \text{ is assigned to } n, \\ 0 & \text{otherwise.} \end{cases} \tag{2.9}$$

Note that $\chi_{n0} = 1$ implies that track n is unassociated and has missed a detection in the latest scan. Furthermore, $\chi_{0m_k} = 1$ implies that measurement m_k is unassociated, i.e., not assigned to any of the $N(k-1)$ existing tracks, but, instead, assigned to the dummy track (false alarm or new track initiation). Since measurement errors within a scan are independent of each other, maximizing the likelihood ratio, consisting of the joint pdf-probability [7,9] of measurements given their origins and the corresponding detection events, over the set of feasible assignments can be cast into the following 2D assignment problem:

$$\text{minimize} \quad \sum_{n=0}^{N(k-1)} \sum_{m_k=0}^{M(k)} c_{nm_k} \chi_{nm_k} \tag{2.10}$$

$$\text{subject to} \quad \sum_{m_k=0}^{m(k)} \chi_{nm_k} = 1, \qquad n = 1, \quad , N(k-1),$$

$$\sum_{n=0}^{N(k-1)} \chi_{nm_k} = 1, \qquad m_k = 1, \quad , M(k),$$

where the cost of assigning measurement m_k to track n is

$$c_{nm_k} = \begin{cases} 0 & \text{if } n \text{ or } m_k = 0, \\ -\log\left(\dfrac{\Lambda(n, m_k)}{\Lambda(0, m_k)}\right) & \text{if } -\log(\ ) < 0, \\ \infty & \text{otherwise.} \end{cases} \tag{2.11}$$

The numerator in the above expression, – log(·), which is based on the likelihood function $\Lambda(\cdot)$ from the IMM estimator (via equation (B.5) in appendix B), is the likelihood that the m_kth measurement at scan k originated from the nth track, and the denominator is the likelihood that the m_kth measurement corresponds to none of the existing tracks (e.g., a false alarm). The occurrence of false alarms is assumed uniformly probable over the sensor's field of view (see appendix A and [46]).

2.4. *Track formation and maintenance*

The following measurements, unassociated in the 2D assignment problem, are used to initialize new tracks and are denoted as the *track formation set*, i.e.,

$$\mathcal{T}^{\mathcal{F}}(k) \triangleq \begin{cases} \{1, \ldots, M(k)\} & k = 1, \\ \left\{ \begin{array}{l} m_k : \chi_{0 m_k} = 1, \\ m_k = 1, \ldots, M(k) \end{array} \right\} & k > 1. \end{cases} \tag{2.12}$$

Initializing (forming) tracks based on a single measurement is somewhat different from the common two-point differencing technique [7]. However, in air traffic surveillance, this approach is sufficient since the difference in times between scans is often relatively short.

The following tracks, denoted as the *track maintenance set*, are extended (the corresponding state vector is updated via equation (B.10) in appendix B) with new measurements at scan k based on the solution to the 2D assignment problem, i.e.,

$$\mathcal{T}^{\mathcal{M}}(k) \triangleq \{n : \chi_{nm_k} = 1, \ n = 1, \ldots, N(k-1), m_k = 1, \ldots, M(k)\}. \tag{2.13}$$

The following tracks, denoted as the *drop track set*, are terminated (dropped) if the tracks do not get updated with measurements from the 2D assignment problem within a *drop track threshold* γ of consecutive scans, i.e.,

$$\mathcal{T}^{\mathcal{D}}(k) \triangleq \begin{cases} & k = 1, \\ \left\{ \begin{array}{l} n : \gamma_n(k) > \gamma, \\ n = 1, \ldots, N(k) \end{array} \right\} & k > 1, \end{cases} \tag{2.14}$$

where γ spans approximately 100 seconds ($\gamma = 20$) in this work, and $\gamma_n(k)$, the *track age* of the nth track at scan k, is the discrete difference, in scans, between the current scan k and the most recent previous scan where track n was updated, i.e.,

$$\gamma_n(k) = \max_j \{j : m_j \neq 0,\ m_j \quad \Phi_n(k),\ 1 \le j \le k-1\}. \tag{2.15}$$

The second term on the right-hand side of equation (2.15) determines the scan number at which track n was last updated, where the *measurement history* corresponding to the nth track at scan k is defined by

$$\Phi_n(k) \triangleq \{m_j : \chi_{nm_j} = 1,\ 1 \le j \le k-1\}. \tag{2.16}$$

The *track set* at scan k after solving the data association problem is then

$$\mathcal{T}(k) \triangleq \begin{cases} & k = 0, \\ \mathcal{T}^{\mathcal{F}}(k) & k = 1, \\ \mathcal{T}^{\mathcal{M}}(k) \cup \mathcal{T}^{\mathcal{F}}(k) - \mathcal{T}^{\mathcal{D}}(k) & k > 1. \end{cases} \tag{2.17}$$

3. Workload

A convention seen over the years in optimization research, namely, the inputs to the optimization problem assumed given, a priori, and computationally insignificant, would have yielded results far from beneficial for the relatively sparse optimization (2D assignment) problem parallelized in this work. In taking a step back to examine the entire scope of the problem at hand, we identified the *interface* to the optimization problem as being the computational bottleneck. We suspect this has analogs in other application areas as well.

Table 1

Workload distribution within IMM-2D.

IMM-2D component	# of filter models in IMM			
	1 (KF)	2	3	5
Obtain scan measurements	0.8%	0.6%	0.5%	0.3%
Data association interface (gating, IMM)	94.3%	94.7%	95.2%	96%
Data association problem (2D assignment)	1.1%	0.8%	0.6%	0.2%
Track maintenance and formation	3.8%	3.9%	3.7%	3.5%

Specifically, as shown in table 1, the vast majority of the workload in IMM-2D comprises processing the set of candidate associations in *interfacing* with the 2D assignment (data association) problem. In fact, this constitutes 94.3%, 94.7%, 95.2%, and 96% of the workload when IMM-2D is configured with 1, 2, 3, and 5 filter models in the IMM, respectively. Recall, as described in section 2.2, interfacing with the data association problem consists of performing numerous gating and 2D assignment cost coefficient calculations (based on the likelihood function from the IMM state estimator). If one were to follow conventional wisdom in this optimization problem, namely, target the assignment algorithm itself for parallelization, 1.1%, 0.8%, 0.6%, and 0.2% of the workload would have been parallelized, respectively.

Clearly, for a sparse air traffic surveillance problem such as the one we have here, a parallel assignment algorithm would yield, at best, minimal benefits. However, although not negating the significance of the interface problem, for dense air traffic surveillance problems, where data association may subsume a significant fraction of the total computation, a parallel assignment algorithm, such as those described in [5,11,22], would provide increased benefit. In this work, since it is the significant computational bottleneck, we developed parallelizations of the *interface* to the 2D assignment (optimization) problem in IMM-2D.

4. IMM-2D shared-memory parallelization

The shared-memory computing environment used for this work consisted of several general-purpose MIMD multiprocessors, namely, a 2- and 4-processor SPARC-station 20, and a 12-processor SPARCcenter 2000. A simple model of the 4-processor SPARCstation 20 architecture is illustrated in figure 2. The software utilized consisted of Solaris 2.4.2, which includes the SunOS 5.4.2 UNIX operating system (OS) and

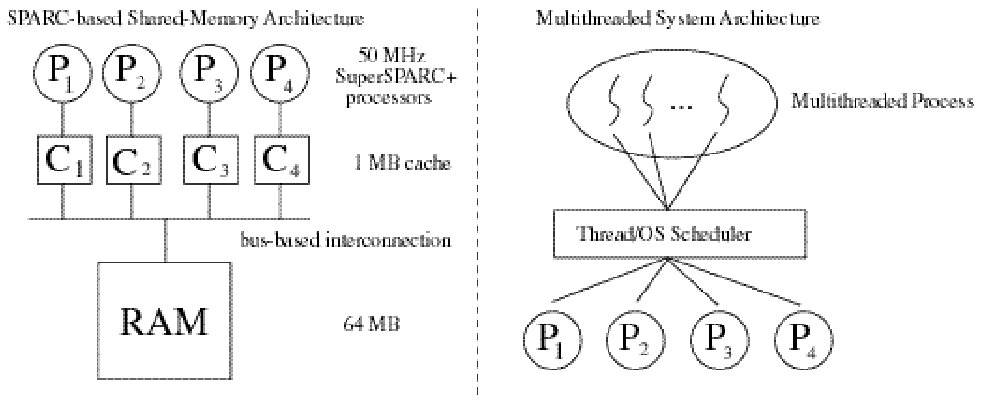
SPARC-based Shared-Memory Architecture
P_1
P_2
P_3
P_4
50 MHz SuperSPARC+ processors
C_1
C_2
C_3
C_4
1 MB cache
bus-based interconnection
RAM
64 MB
Multithreaded System Architecture
Multithreaded Process
Thread/OS Scheduler
P_1
P_2
P_3
P_4

Figure 2. Model of shared-memory architecture.

the multithreaded system architecture, which we used as our parallel processing interface. Multithreading separates a UNIX process into lightweight independent *threads*, each of which (concurrently) executes a sequence of the process's instructions. Threads are dispatched across the processor set indirectly via a two-level scheduling hierarchy. Threads implicitly communicate via shared memory; consequently, synchronization mechanisms (e.g., mutual exclusion) must be supported to allow threads to cooperate in accessing shared data.

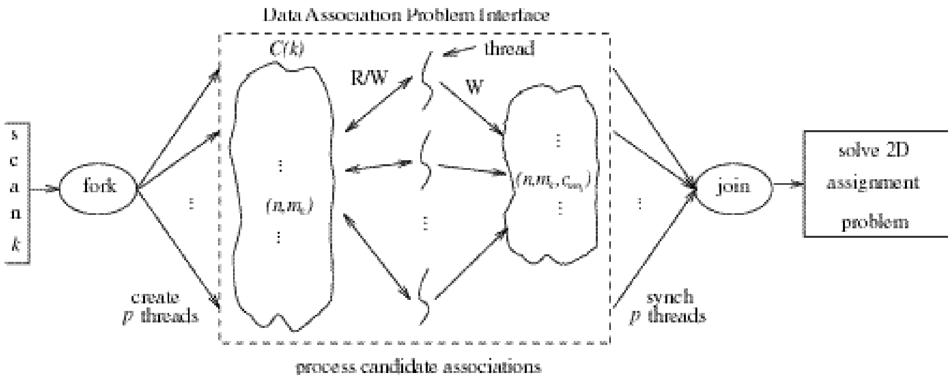
Data Association Problem Interface
C(k)
thread
R/W
W
s c a n k
fork
(n,m_c)
(n,m_c,c_{nm_c})
join
solve 2D assignment problem
create p threads
synch p threads
process candidate associations

Figure 3. Task graph of IMM-2D shared-memory parallelization.

The coarse-grained shared-memory parallelization developed for the data association (2D assignment) interface problem in IMM-2D is based on the supervisor/worker model (see figure 3). The supervisor thread initially *forks* some number of worker threads, say p, to process the set of candidate associations, $C(k)$, defined by equation (2.3) in section 2.2.2. Once forked, the supervisor awaits processing of $C(k)$ to be completed by the p worker threads via a *join* operation. Worker threads, asynchronously and in parallel, process some (parameter) number of candidate associations per serialized critical section access across mutually exclusive track and measurement data. Recall that the processing of a candidate association consists of performing two coarse gating tests (the second is performed only if the first succeeds) and, if both gating tests succeed, a fine gating test is applied, i.e., a negative log-likelihood score is calculated based on equation (2.11). Since the processing cost corresponding to each candidate association is not uniform, maximum load balancing is achieved by dynamic allocation of candidate associations across threads. Upon completing processing of $C(k)$, the supervisor solves the global 2D assignment problem.

5. IMM-2D distributed-memory parallelization

The distributed-memory computing environment used for this work consisted of a 32-node Intel Paragon HPC. Hardware specifications for the Paragon are illustrated in figure 4. Paragon OSF/1, a subset of the standard OSF/1 UNIX operating system, based on the Mach 3.0 microkernel, is a distributed OS that is resident on all nodes. Due to the lack of multithreading support in Paragon OSF/1 at the time of this work, only one of the processors within each node was utilized. Because the Paragon is a distributed-memory architecture, processors communicate with one another using messages via a message-passing interface (MPI) supported by Paragon OSF/1.

The distributed-memory parallelization developed for the data association (2D assignment) interface problem in IMM-2D is also based on the supervisor/worker model (see figure 5). Specifically, p workers process, in parallel, the set of candidate

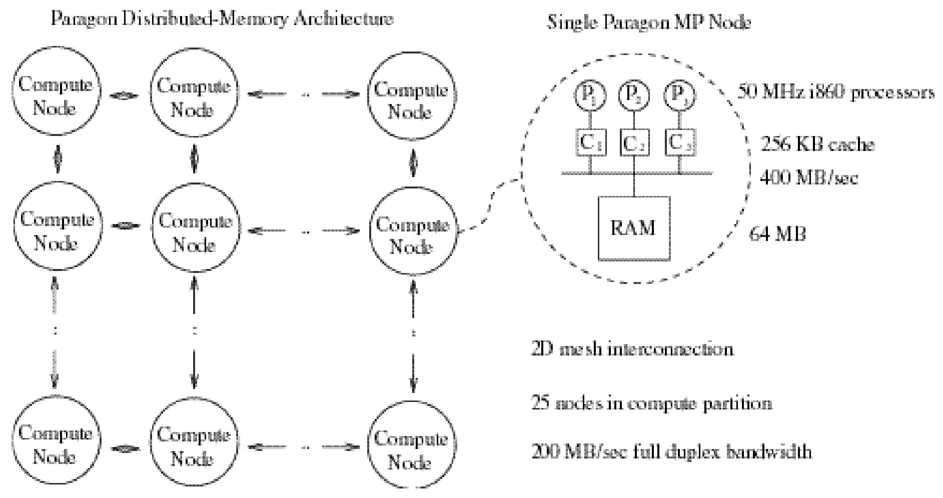

Figure 4. Model of distributed-memory architecture.

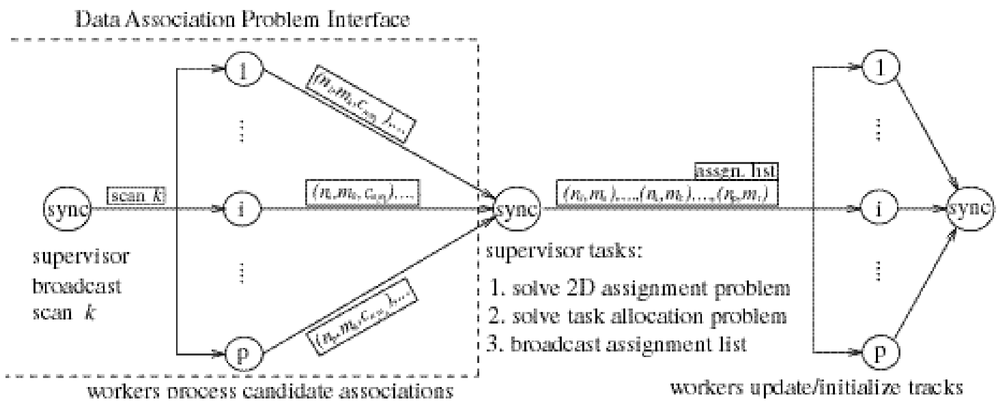
Data Association Problem Interface
sync
scan k
1
i
p
supervisor broadcast scan k
$(n_1, m_1, c_{n_1m_1}), \ldots$
$(n_i, m_i, c_{n_im_i}), \ldots$
$(n_p, m_p, c_{n_pm_p}), \ldots$
assgn. list
$(n_1, m_1), \ldots, (n_i, m_i), \ldots, (n_p, m_p)$
supervisor tasks:
1. solve 2D assignment problem
2. solve task allocation problem
3. broadcast assignment list
workers process candidate associations
workers update/initialize tracks

Figure 5. Task graph of IMM-2D distributed-memory parallelization.

associations, $C(k)$, based on their mutually exclusive and exhaustive local track sets. Upon completing processing, workers asynchronously send to the supervisor messages containing a list of tuples consisting of track IDs, measurement IDs, association costs (i.e., 2D assignment cost coefficients), and workload bids (defined shortly). Upon collecting all such lists, the supervisor solves the global 2D assignment problem. The assignment list is then broadcast to all workers, enabling them to update (extend/drop) existing tracks and/or initialize (form) new tracks. The critical issue for the distributed-memory parallelization is the allocation of new tracks (tasks), namely, the set of unassociated measurements from the 2D assignment problem (i.e., the track formation set $\mathcal{T}^{\mathcal{F}}(\cdot)$ defined by equation (2.12)), across the worker processor set in such a way that load imbalances are minimized.

6. Task allocation problem

Each of the dynamic (heuristic) task allocation algorithms developed in this work assume that tasks are equivalently executable logically on any homogeneous processor. We developed source-initiated, non-preemptive task allocation algorithms, that is, once assigned by the supervisor, a task stays assigned for the duration of its lifetime (until the track is dropped). No a priori assumptions are made about the task size (cost). All but one algorithm use *bidding* in allocating tasks, i.e., the supervisor seeks bids from all workers via a (past, present, or predicted future) workload cost, and, based on such bids, selects the lowest bidders to assign new tasks.

6.1. *Task definition*

For the problem at hand, we define a *task* to be a track's inseparable set of candidate associations to process over the lifetime of the track. The task communication cost is negligible since tasks do not communicate amongst one another, whereas the task processing cost, namely, the gating tests and likelihood function evaluations, is significant. Moreover, the task processing cost is *uncertain* at the time of task allocation. Consequently, heuristic solutions to the task allocation problem are necessary, where the workload, namely, the processing of candidate associations, is to be distributed across the processor set *indirectly* via an allocation of tracks.

We chose a track as the task granularity, as opposed to an individual candidate association, for the following reasons. Allocating n tracks across p processors has space complexity of $O(n)$, whereas allocating candidate associations has complexity of $O(pn)$. Furthermore, an increase of roughly three orders of magnitude in message size ensues when allocating individual associations versus tracks, i.e., the former requires entire track states to be communicated, whereas the latter requires only track IDs.

6.2. *Problem formulation*

Given p worker processors, say $i = 1,\ldots,p$, define the ith worker's local *track formation set* at scan k, after the 2D assignment problem has been solved, by

$$\mathcal{T}_i^{\mathcal{F}}(k) \triangleq \{m_k : m_k = i(\mathrm{mod}\ p),\ m_k \quad \mathcal{T}^{\mathcal{H}}(k)\}, \tag{6.1}$$

where (mod p) denotes the modulus p operation and $\mathcal{T}^{\mathcal{H}}(k)$ the *mapped track formation set* at scan k, is determined by a mapping function $\mathcal{H}(\cdot)$ defined (forthcoming) by the task allocation algorithm.

Per scan, after solving the 2D assignment problem, the supervisor broadcasts to the worker processors the re-mapped *assignment list*, i.e.,

$$A(k) \triangleq \mathcal{T}^{\mathcal{M}}(k) \bigcup \mathcal{T}^{\mathcal{H}}(k), \tag{6.2}$$

where $\mathcal{T}^{\mathcal{M}}(k)$ is the track maintenance set as defined in equation (2.13) and $\mathcal{T}^{\mathcal{H}}(k)$, the mapped track formation set, is determined by a (one-to-one) mapping function,

$$\mathcal{H} : \mathcal{T}^{\mathcal{F}}(k) \to \mathcal{T}^{\mathcal{H}}(k). \tag{6.3}$$

In effect, $\mathcal{H}(\cdot)$ maps the IDs of new tracks to initialize from $\mathcal{T}^{\mathcal{F}}(k)$ to $\mathcal{T}^{\mathcal{H}}(k)$ in such a way that tracks in $\mathcal{T}^{\mathcal{H}}(k)$, having possibly remapped track IDs, become allocated (by way of the modulus operator) to the appropriate worker's local track set $\mathcal{T}_i(k)$ via its track formation set $\mathcal{T}_i^{\mathcal{F}}(k)$.

6.3. *Dynamic task allocation algorithms*

We now briefly present five dynamic task allocation algorithms developed as part of this work. Critical to the performance of each task allocation algorithm is the basis and effectiveness of the bid (cost) obtained from workers in measuring the workload. Interested readers can find a more thorough description of the algorithms in [34–36].

6.3.1. *Cyclic modulo p*

The allocation of tasks for this heuristic is based on a (static) uniform cyclic splitting scheme via the modulus p operator. Per scan, the supervisor cycles through the set of p workers assigning to each worker, in turn, the next assignable track. Unlike a cooperative solution, this heuristic does not incorporate workload information sharing between the supervisor and workers. However, although a uniform distribution of tracks follows, a uniform distribution of tasks (i.e., workload) may not. The mapped track formation set, determined by $\mathcal{H}(\cdot)$, is defined by

$$\mathcal{T}^{\mathcal{H}}(k) \triangleq \{\mathcal{H}(m_{k_j}) : \mathcal{H}(m_{k_j}) = N(k-1) + j,\ j = 1, \quad , |\mathcal{T}^{\mathcal{F}}(k)|, m_k \quad \mathcal{T}^{\mathcal{F}}(k)\}. \tag{6.4}$$

6.3.2. *First-come first-served (FCFS)*

The allocation of tasks for this heuristic is based on the FCFS discipline. Per scan, the supervisor infers the degree of each worker's workload relative to all workers based on the order in which workers report back to the supervisor with their lists of tuples, i.e., workers reporting early are assumed to have a lighter workload. Based on the workers' report ordering, the supervisor makes the appropriate load balancing decisions. Specifically, if

$$\omega_{p_1}(k) \prec \omega_{p_2}(k) \prec \cdots \prec \omega_{p_p}(k),$$

where $\omega_{p_i}(k) \prec \omega_{p_j}(k)$, $1 \le p_i, p_j \le p$, $i \ne j$, implies worker p_i reported back to the supervisor before worker p_j, then define the mapped track formation set, determined by $\mathcal{H}(\cdot)$, as the set of track IDs mapped to workers p_i, $i = 1, \ldots, p$ via their local track formation sets $\mathcal{T}_{p_i}^{\mathcal{F}}(k)$ based on the above ordering, i.e.,

$$\mathcal{T}^{\mathcal{H}}(k) \triangleq \left\{ \begin{array}{l} \mathcal{H}(m_{k_j}) : j = 1, \ldots, |\mathcal{T}^{\mathcal{F}}(k)|, \\ m_k \in \mathcal{T}^{\mathcal{F}}(k), \mathcal{H}(m_{k_j}) = \min\{n : n = p_j (\mathrm{mod}\, p), N(k-1) < n < \mathcal{H}(m_{k_{j+1}})\} \end{array} \right\}. \tag{6.5}$$

6.3.3. Track age

The allocation of tasks for this heuristic is based on a dynamically changing statistical track age metric, which, per scan, serves as an indicator of the *present* state of the workload. The motivation for this heuristic is as follows. A mature track (i.e., a track not updated in the data association problem over many scans), will have large spatial gates with many candidate measurements to associate with, and, consequently, induce an increase in workload due to additional likelihood function evaluations [34,35]. At scan k, recalling that $\gamma_{n_i}(k)$, defined in equation (2.15), denotes the age of the n_ith track in local track set $\mathcal{T}_i(k-1)$, the ith worker will have tracks with ages

$$\Gamma_i(k) \triangleq \{\gamma_{n_i}(k) : n_i \in \mathcal{T}_i(k-1), n_i = 1, 2, \ldots, N_i(k-1)\}. \tag{6.6}$$

The mean track age of tracks in $\mathcal{T}_i(k-1)$ at scan k is

$$\bar{\gamma}_i(k) = \frac{1}{N_i(k-1)} \sum_{n_i=1}^{N_i(k-1)} \gamma_{n_i}(k) \tag{6.7}$$

and the range (variance) of track ages of tracks in $\mathcal{T}_i(k-1)$ is

$$\Delta_i(k) = \max_{\substack{n_i \neq n_j \\ 1 \leq n_i, n_j \leq N_i(k-1)}} \{\gamma_{n_i}(k) - \gamma_{n_j}(k)\}. \tag{6.8}$$

Define the normalized present workload cost for the ith worker at scan k by

$$\omega_i(k) = \frac{1}{\gamma}(\bar{\gamma}_i(k) + \Delta_i(k)) + \frac{N_i(k-1)}{N(k-1)}, \tag{6.9}$$

where γ refers to the drop track threshold described previously. Each worker sends to the supervisor, in addition to its list of tuples, a task allocation cost (bid) defined by equation (6.9). Tracks from the mapped track formation set $\mathcal{T}^{\mathcal{H}}(k)$ are allocated to workers having the smallest workload costs (lowest bids). Assume, without loss of generality, that the workload costs for the p workers are ordered as follows:

$$\omega_{p_1}(k) \leq \omega_{p_2}(k) \leq \cdots \leq \omega_{p_p}(k),$$

where $\omega_{p_i}(k) \leq \omega_{p_j}(k)$, $1 \leq p_i, p_j \leq p$, $i \neq j$, implies worker p_i has less present workload cost than worker p_j. Hence, as in the previous case, $\mathcal{T}^{\mathcal{H}}(k)$ is defined by equation (6.5).

6.3.4. 1-step # likelihood evaluations predictor

The allocation of tasks for this heuristic is based on a *predicted* number of likelihood functions to evaluate at the next scan, based on a state feedback estimator. Per scan, this metric serves as an indicator of *future* workloads based on the *past* and *present* likelihood histories. In (low-) high-pass mode, the estimator monitors the (past) present workload and reacts (slowly) quickly to changes in the state. Let $\lambda_i(k)$ denote the number of likelihood functions evaluated by the ith worker at scan k. Define the predicted number of likelihoods to be evaluated by the ith worker at scan $k+1$ by

$$\hat{\lambda}_i(k+1) = \alpha\,\hat{\lambda}_i(k) + (1-\alpha)\lambda_i(k), \tag{6.10}$$

where α is a tuning parameter. Assume, without loss of generality,

$$\omega_{p_1}(k) \leq \omega_{p_2}(k) \leq \cdots \leq \omega_{p_p}(k),$$

where $\omega_{p_i}(k) = \hat{\lambda}_{p_i}(k+1)$ as defined by equation (6.10). Again, $\mathcal{T}^{\mathcal{H}}(k)$ is defined by equation (6.5).

6.3.5. Mean # likelihood evaluations

The allocation of tasks for this heuristic is based on the *mean* number of likelihood functions evaluated, which, per scan, serves as an indicator of *past* or *present* workloads. In the ideal load balanced case, each worker would have a proportional number of likelihoods to evaluate, i.e.,

$$\bar{\lambda}(k) = \frac{1}{p}\sum_{i=1}^{p}\lambda_i(k). \tag{6.11}$$

Define the (present) mean workload cost for the ith worker at scan k by

$$\omega_i(k) = \frac{\lambda_i(k)}{\bar{\lambda}(k)}. \tag{6.12}$$

Similarly, in general, if we want to take into account past workloads, over k scans, each worker would have a proportional number of likelihoods to evaluate, i.e.,

$$\bar{\bar{\lambda}} = \frac{1}{p}\sum_{l=1}^{k}\sum_{i=1}^{p}\lambda_i(l). \tag{6.13}$$

Thus, define the (past) time-averaged workload cost for the ith worker at scan k by

$$\beta_i(k) = \frac{1}{\bar{\bar{\lambda}}}\sum_{l=1}^{k}\lambda_i(l). \tag{6.14}$$

Assume the mean (time-averaged) workload costs are ordered as follows:

$$\omega_{p_1}(k) \leq \omega_{p_2}(k) \leq \cdots \leq \omega_{p_p}(k),$$
$$\beta_{p_1}(k) \leq \beta_{p_2}(k) \leq \cdots \leq \beta_{p_p}(k),$$

respectively, where, in both expressions, $\omega_{p_i}(k) \leq \omega_{p_j}(k)$, $(\beta_{p_i}(k) \leq \beta_{p_j}(k))$, $1 \leq p_i$, $p_j \leq p$, $i \neq j$, implies worker p_i has less mean (time-averaged) workload cost than worker p_j. Again, as before, $\mathcal{T}^{\mathcal{H}}(k)$ is defined by equation (6.5).

Thus, to summarize, in table 2 we provide the general characteristics of each of the presented task allocation algorithms used in the distributed-memory parallelization of IMM-2D. The heuristic algorithms are organized in terms of increasing time and space complexity. Note that in the *Information reactiveness* column, low-pass implies a slowly-varying (conservative) task allocation algorithm, whereas high-pass implies a rapidly-varying (aggressive) task allocation algorithm.

Table 2

Characteristics of task allocation algorithms.

Heuristic	Behavior	Share info.	Basis of cost (workload)	Information reactiveness	Rel. overhead time and space
mod p	static	no	none	low-pass	none
FCFS	dynamic	no	present	high-pass	low
$\hat{\Lambda}_{(\alpha \to 0)}$	dynamic	yes	mostly present	high-pass	moderate
$\hat{\Lambda}_{(\alpha \to 0.5)}$	dynamic	yes	past and present	moderate	moderate
$\hat{\Lambda}_{(\alpha \to 1)}$	dynamic	yes	mostly past	low-pass	moderate
$\bar{\Lambda}_{(\omega_i(k))}$	dynamic	yes	present	moderate	moderate/high
$\bar{\Lambda}_{(\beta_i(k))}$	dynamic	yes	past and present	low-pas	moderate/high
track age	dynamic	yes	present	moderate	high

7. Results

7.1. Preliminaries

When presenting the performance of the various parallelizations of IMM-2D, we use a conventional formulation for parallel *efficiency*, i.e.,

$$\mathcal{E} = \frac{\tau_1}{p\tau_p}, \tag{7.1}$$

where τ_1 denotes the sequential execution time utilizing 1 processor, and τ_p denotes the parallel execution time utilizing p processors. Furthermore, the correlation coefficient plot in figure 9 is based on a standard formulation, i.e.,

$$\rho_{ii}(k) = \frac{r_{ii}(k)}{r_{ii}(0)}, \tag{7.2}$$

where the auto covariance function, given mean $\bar{y}_i$, is defined as

$$r_{ii}(k) = \frac{1}{S-1}\sum_{s=1}^{S}[y_i(s) - \bar{y}_i]\,[y_i(s-k) - \bar{y}_i]. \tag{7.3}$$

7.2. Shared-memory parallelization results

In this section, based on the 2-, 4-processor SPARCstation 20*s*, and the 12-processor SPARCcenter 2000, we demonstrate the computational performance of the coarse-grained parallelization of IMM-2D, making comparisons with both a fine-grained parallelization and a serial implementation of IMM-2D. In the fine-grained parallelization, the multiple filter models of the IMM estimator are evenly distributed across the processor set in a manner similar to the approach described in [3,4]. Since the multiprocessors used in this work are time-shared systems, the execution times of IMM-2D depended, in part, on random system events such as the system load and thread schedule order. Consequently, Monte Carlo simulations were performed, and all results presented represent the means of those simulations with standard errors less than 3%.

In figure 6, we plot the execution times of the sequential, the coarse- and fine-grained shared-memory parallelizations of IMM-2D on the 2- and 4-processor SPARC-station 20. Note that *Mach 2* and *Mach 5* simply denote the speed of an aircraft to be

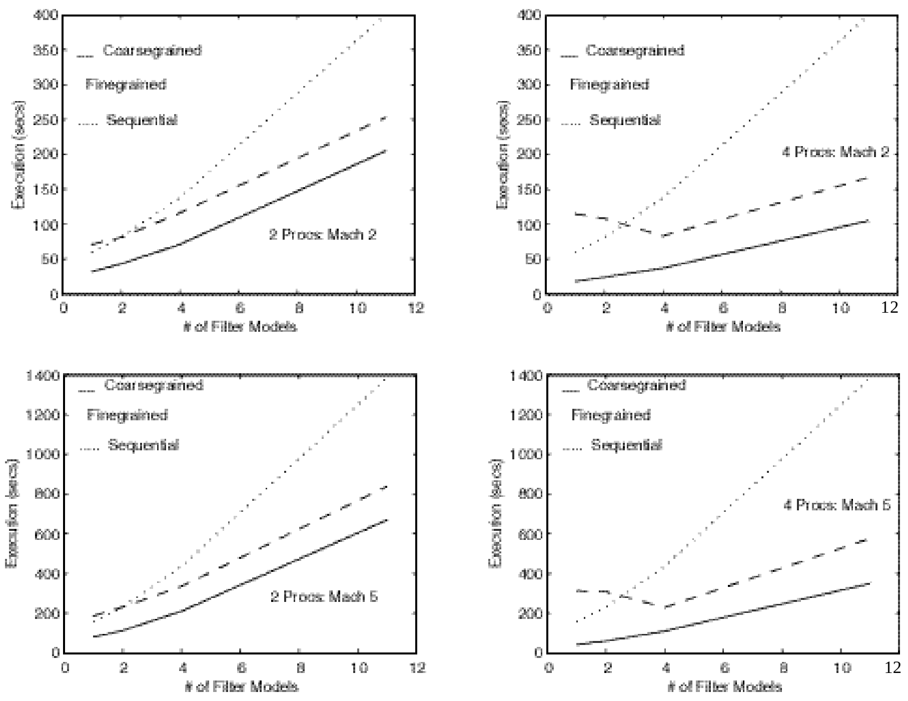

Figure 6. Execution time of IMM-2D on a 2- and 4-processor SPARCstation 20 using various coarse gating policies. Horizontal axis denotes # of filter models in IMM configuration.

two and five times the speed of sound, respectively. The significance of these two parameters is as follows. As described in section 2.2.2, one of the coarse gating policies employed in IMM-2D is based on an aircraft's maximum velocity. Increasing the magnitude of the *Mach* parameter will increase the spatial area of the track's maximum velocity gate, thus increasing the processing cost associated with the track since a greater number of measurements will fall within the gate.

Clearly, the coarse-grained parallelization demonstrates superior execution time performance over the fine-grained parallelization for any number of models used in the state estimator. Moreover, the performance of the fine-grained parallelization is dependent on the number of models used, yielding marginal performance improvements only when using an unrealistically large number of filter models. Furthermore, when configured with a small to moderate number of IMM models (< 3 which is considered sufficient/suitable for air traffic surveillance problems), the fine-grained parallelizations yield *execution time greater than sequential time*.

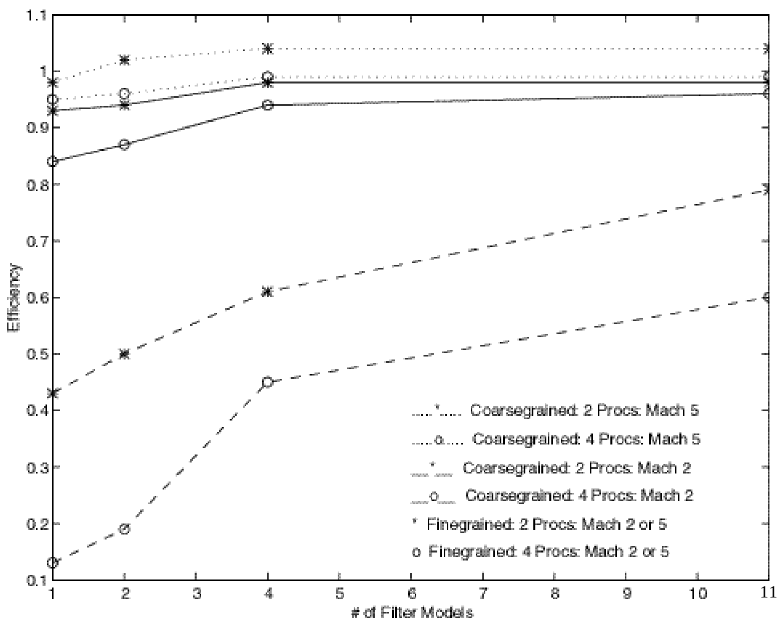

Figure 7. Parallel efficiency of IMM-2D on a 2- and 4-processor SPARC-station 20 using various coarse gating policies. Horizontal axis denotes # of filter models in IMM configuration.

In figure 7, we plot the parallel efficiency for both the coarse- and fine-grained shared-memory parallelizations of IMM-2D. Clearly, as figure 7 illustrates, the computational performance of the coarse-grained parallelization is *independent* of the number of filter models used in the IMM estimator, whereas the fine-grained parallelization performs rather inefficiently unless the number of filter models used is unrealistically high. Furthermore, as shown in figure 7, near-unity efficiency and, given a large enough problem (e.g., 2 processors using Mach 5 coarse gating policy), even greater than unity efficiency (i.e., *superlinear speedup*) is possible. When non-algorithmic issues such as context switches, effective memory size and access costs, and scheduling order are considered, superlinear speedups in practice may indeed occur [23,25].

Many factors determine the scale-up performance of a parallel algorithm, in particular, the multiprocessor architecture and the problem size are directly related. In the context of multitarget tracking, the problem size is a function of numerous factors, including the number of actual and false alarm tracks, noise/clutter intensity, target density, contentiousness of the target-measurement data, and the number of filter models used in the IMM. When considering the *scalability* of IMM-2D, a highly desirable feature would be for it to maintain the same *high* level of performance (e.g., efficiency) on larger multiprocessor systems as it did on the smaller ones, and do so in terms of *any* of the factors that can influence the problem size. Certainly, IMM-2D having this property would make it *robust*, and it could easily adapt, without modification, to other diverse multitarget tracking problems.

Unlike the fine-grained parallelization, the coarse-grained parallelization of IMM-2D scales when any of the factors influencing problem size increase. In particular, in figure 8, we plot the speedup for the coarse-grained parallelization on a 12-processor SPARCcenter 2000 for several problem instances. In *(Instance (original))*, all factors influencing the problem size are unchanged, whereas, in *(Instance(+500%))*

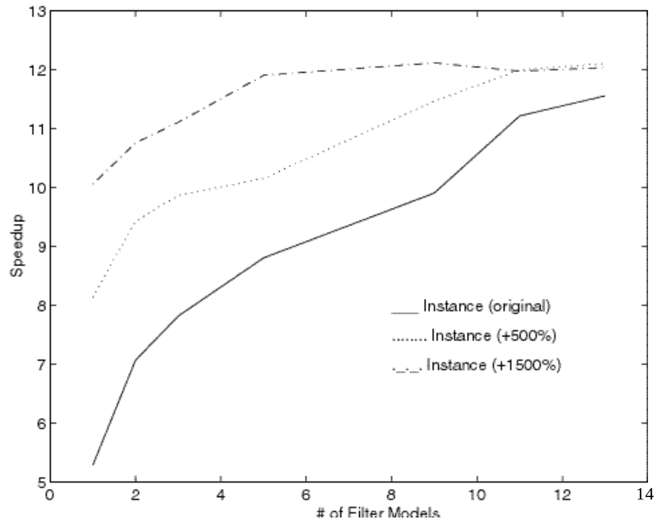

Figure 8. Speedup of IMM-2D on a 12-processor SPARCcenter 2000 for several problem instances. Horizontal axis denotes # of filter models in IMM configuration.

and *(Instance (+1500%))*, a more dense multitarget scenario was simulated by increasing the problem size (relative to *(Instance (original))*) in terms of track set size by 500% and 1500%, respectively. From the plot, we can see marginal speedup results when the problem size is too small (i.e., *(Instance (original))*) for this particular multiprocessor system. However, as the problem size increases, the scalability of the coarse-grained parallelization is evident, approaching near-linear speedup with fewer filter models, and linear with many.

7.3. Distributed-memory parallelization results

For the distributed-memory parallelization, we present performance results for IMM-2D on the 32-node Intel Paragon HPC using the various task allocation algorithms described in this paper. However, before discussing performance issues, for IMM-2D to benefit from load balancing in a distributed-memory environment, the degree of randomness in the workload across worker processors must be assessed. If purely random, then anything we do in terms of load balancing will be futile. In fact, only if some degree of correlation exists, across scans, in the workload should load balancing be pursued.

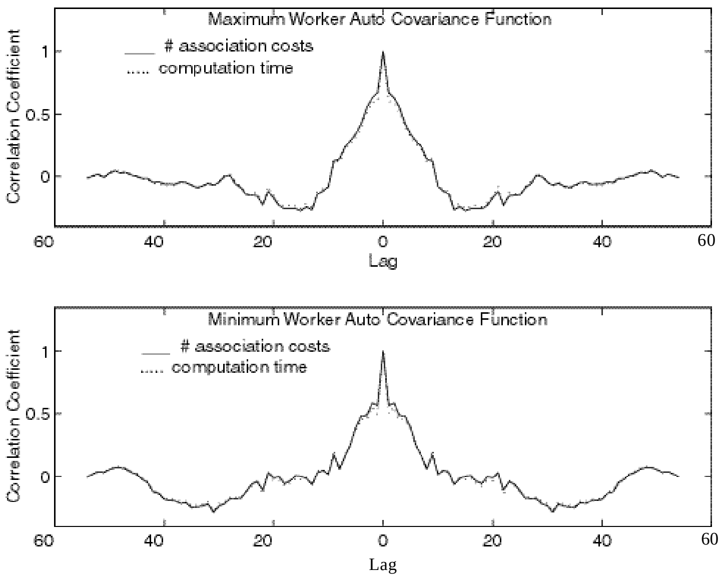

Figure 9. Auto covariance of association costs and computation time of IMM-2D on the Intel Paragon using the mod p heuristic for the task allocation algorithm.

In figure 9, we plot, over scans processed, the auto covariance function, in terms of association costs computed (where a likelihood function based on the IMM was evaluated) and computation time, for the maximum and minimum loaded workers.

Figure 9 provides a statistical indication of the randomness in the workload by correlating, across scans, both the association costs computed and computation time for the maximum and minimum loaded workers based on using the mod *p* heuristic in the task allocation algorithm. Clearly, for lags with run [0, 5], a correlation exists (>50%), namely, heavily (lightly) loaded workers tend to retain the same level of workload in the near future, i.e., within 5 scans. Hence, because the workload is not purely random, candidate associations can be allocated, indirectly, in such a way as to ensure load balancing.

As illustrated in figure 10, the execution time of the distributed-memory parallelization of IMM-2D is plotted using the various task allocation algorithms presented. Not only do they all yield very similar execution time performance, figure 10 shows a near-linear decrease in time over the number of processors *p*. However, in terms of scale-up, the execution time performance curve for IMM-2D started to flatten out

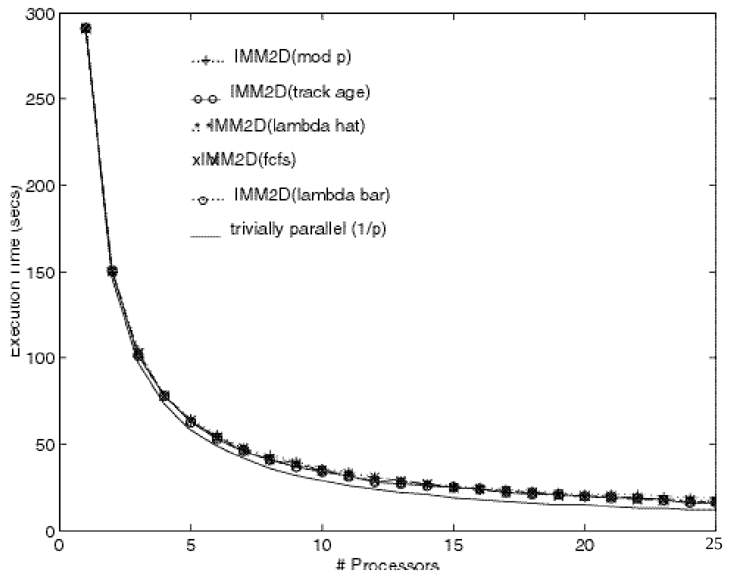

Figure 10. Execution time of IMM-2D on the 32-node Intel Paragon using various task allocation algorithms.

considerably at 25 nodes. At >32 nodes, the performance curve for IMM-2D started to increase. Since there are numerous factors that can influence the problem size in a multitarget tracking problem, scale up analysis is rather difficult. Consequently, we leave it as a future endeavor to perform this analysis properly.

In figure 11, we plot the efficiency of the distributed-memory parallelization of IMM-2D using the various task allocation algorithms. The plots follow what we would expect from a diminishing return function such as efficiency, i.e., for smaller *p*, near-unity efficiencies are obtainable based on each of the task allocation algorithms, whereas, for larger *p*, the efficiency diminishes.

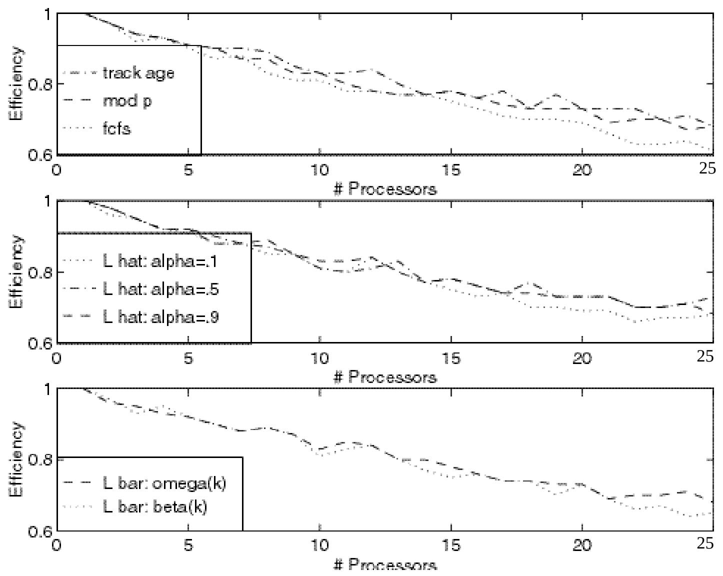

Figure 11. Efficiency of IMM-2D on the 32-node Intel Paragon using various task allocation algorithms.

Other interesting observations about figure 11 can be made. Firstly, the FCFS heuristic consistently performed worse than the other heuristic algorithms. This occurs primarily because the FCFS heuristic makes task allocation decisions based on an inferior bid, i.e., the first worker to report back to the supervisor is not necessarily the least loaded worker. Secondly, the simple static mod p heuristic performed rather well in comparison to the dynamic, more complex information sharing heuristics. This occurs primarily because of the additional overhead incurred by the latter heuristics, i.e., sorting, a mapping of track IDs, and a task allocation bid function calculation. Moreover, since IMM-2D was moderately loaded on the HPC for the given multitarget problem, a random task allocation heuristic will, in general, provide a fairly balanced load [12].

In table 3, we provide the mean efficiency improvement of the functions plotted in figure 11 relative to the mod p heuristic. Consistent with other researchers [20, 21,43], table 3 shows that, in the context of multitarget tracking, relatively simple heuristic task allocation algorithms can yield excellent performance in practice. In general, IMM-2D performed best when the task allocation algorithm's reactiveness to information concerning the workload being shared between workers and the supervisor was moderate. Specifically, a task allocation algorithm employing a complex heuristic (e.g., track age, $\hat{\Lambda}$, $\bar{\Lambda}$) provides, relative to the simple mod p heuristic, a decrease in performance of –0.1 (i.e., $\hat{\Lambda}_{(\alpha \to 0)}$) in the worst case and a marginal performance improvement of 3.5% (i.e., $\hat{\Lambda}_{(\alpha \to 0.5)}$) in the best case.

Table 3

Efficiency improvement, relative to mod *p*, of IMM-2D configured with various task allocation algorithms.

Mean efficiency improvement

$$\bar{\delta}(\mathcal{E}_g, \mathcal{E}_h) = \frac{1}{p} \sum_{i=1}^{p} \frac{\mathcal{E}_h(i) - \mathcal{E}_g(i)}{\mathcal{E}_g(i)}$$

g	h						
mod p	FCFS	$\hat{\Lambda}_{(\alpha \to 0)}$	$\hat{\Lambda}_{(\alpha \to 0.5)}$	$\hat{\Lambda}_{(\alpha \to 1)}$	Track age	$\bar{\Lambda}_{(\omega_i(k))}$	$\bar{\Lambda}_{(\beta_i(k))}$
$\bar{\delta}(\mathcal{E}_g, \mathcal{E}_h) \approx$	– 3%	– 0.1%	3.5%	1.5%	2%	1.5	0%

8. Conclusions

In this work, in the context of a sparse multitarget air traffic surveillance problem, we showed (via workload analysis) that the *interface* to the data association (optimization) problem was the computational bottleneck, as opposed to the optimization problem itself. We then described a coarse-grained shared-memory parallelization of the data association interface problem that yielded excellent performance results, i.e., superlinear speedups, scalable, performance independent of numerous factors influencing problem size, e.g., many models in the IMM, large track/scan set sizes, or contentious target-measurement multitarget data due to clutter, dense scenarios, and/or coarse gating policies. In the case of a fine-grained parallelization, the performance was dependent on the number of filter models used, yielding negligible throughput for any number of filter models, marginal speedups when many models were used, and worse execution time than sequential time when three or less filter models were used. We then described an SPMD distributed-memory parallelization of the data association interface problem that also yielded excellent performance (i.e., near-linear speedups) configured with relatively simple task allocation algorithms. Furthermore, consistent with other research efforts, in the context of multitarget tracking, we showed that using relatively simple dynamic (heuristic) task allocation algorithms can offer great promise in practice.

Appendix A: Sensor specifications

The sensor parameters for the radars are given in table 4, where f_p is the pulse repetition frequency, β_θ is the vertical beamwidth, β_φ is the horizontal beamwidth, τ is the pulse width, $R_{\max}$ is the maximum range, Δr is the range resolution cell and c is the speed of light. From the above sensor parameters, the range and azimuth standard deviations (i.e., σ_r and σ_φ, respectively) are determined using the assumption that the range and azimuth measurements are uniformly distributed in the corresponding resolution cells. Hence, $\sigma_r = \Delta r/(2\sqrt{3})$ and $\sigma_\varphi = \beta_\varphi/(2\sqrt{3})$. The altitude standard deviation $\sigma_h = 17.6$m and the probability of detection $P_D = 0.95$ are chosen based on FAA standards [45,46].

Table 4

Radar specifications.

	f_p (Hz)	β_θ	β_φ	τ (μs)	$R_{\max} = c/(2f_p)$	$\Delta r = c\tau/2$
R	340	5.4°	1.3°	6	441.2 km	0.9 km
D	350	3.75°	1.2°	1.8	428.6 km	0.27 km

The probability density function (pdf) of extraneous measurements (false alarms) is the inverse of the sensor's field of view volume, denoted by Ψ. We assume that Ψ is approximately constant and proportional to the resolution cell volume, i.e., $\Psi = K(\Delta t)\,(R_{\max}\beta_\theta)\,(R_{\max}\beta_\varphi)$. The factor K (e.g., $3^3 = 27$) is needed since the gate is, in general, spread over several adjoining resolution cells.

Appendix B: IMM state estimation

The IMM estimator used in IMM-2D [46], illustrated in figure 12, is an augmented version of the IMM estimation algorithm [7]. As figure 12 depicts, the various computations involved in one cycle of the IMM estimator can be divided into: (i) *Interaction/Mixing*, (ii) *Filtering*, (iii) *Update of mode and mixing probabilities*, and (iv) *Combination of state estimates and covariances*. The IMM assumes the motion of a target for which a corresponding track will develop follows one of r possible filter models (or modes) between two successive detections. The augmented IMM estimator combines the likelihoods $\Lambda_j(\cdot)$, $j = 1,\dots,r$ from the individual mode-matched filters to yield an overall likelihood $\Lambda(\cdot)$. We omit detailed exposition of the IMM algorithm here, since it is not the main focus of this paper, and simply present a brief summary of the relevant equations. We refer interested readers to [46] for a more descriptive presentation of the material.

The *mixed state estimate* and *covariance* are given by

$$\hat{x}_{0j}(t_{m_{k-1}}) = \sum_{i=1}^{r} \hat{x}_i(t_{m_{k-1}})\mu_{i|j}(t_{m_{k-1}}), \tag{B.1}$$

$$\begin{aligned} P_{0j}(t_{m_{k-1}}) = {} & \sum_{i=1}^{r} [\hat{x}_i(t_{m_{k-1}}) - \hat{x}_{0j}(t_{m_{k-1}})][\hat{x}_i(t_{m_{k-1}}) - \hat{x}_{0j}(t_{m_{k-1}})]'\mu_{i|j}(t_{m_{k-1}}) \\ & + \sum_{i=1}^{r} P_i(t_{m_{k-1}})\mu_{i|j}(t_{m_{k-1}}). \end{aligned} \tag{B.2}$$

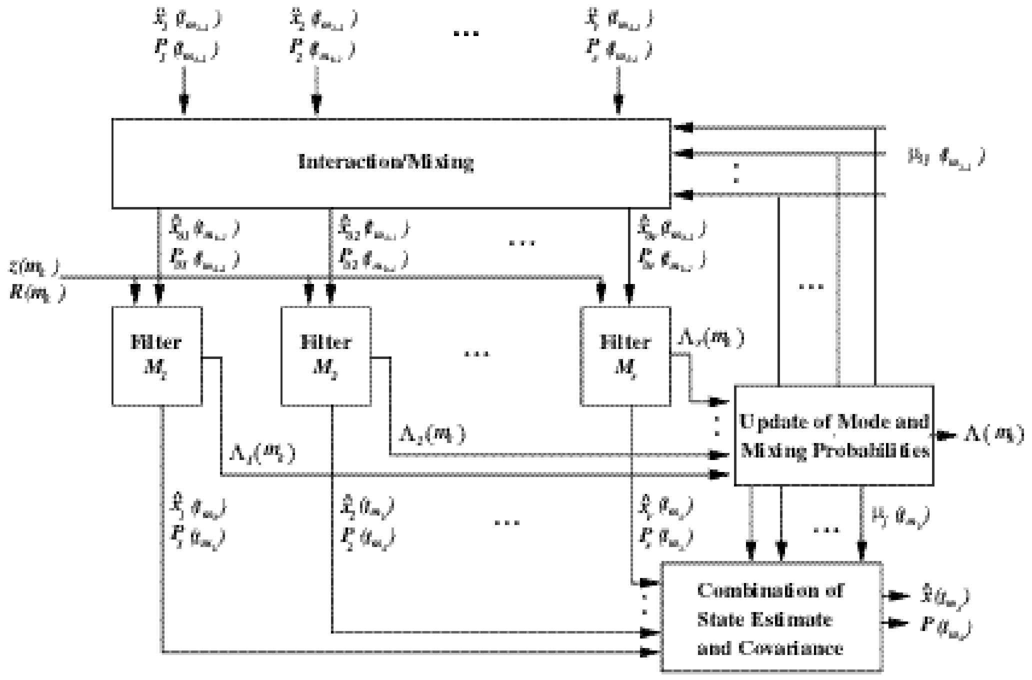

Figure 12. Augmented IMM estimator.

The *mixing probability* is given by

$$\mu_{i|j}(t_{m_{k-1}}) = \frac{\mu_i(t_{m_{k-1}})}{\sum_{i=1}^{r} \mu_i(t_{m_{k-1}})}, \tag{B.3}$$

where $\pi_{ij} \triangleq \mathrm{P}\{u(t_{m_k}) = j \mid u(t_{m_{k-1}}) = i\}$ is the model transition probability. The *mode probability* is given by

$$\mu_j(t_{m_k}) = \frac{\Lambda_j(m_k)\sum_{i=1}^{r} \mu_i(t_{m_{k-1}})\pi_{ij}}{\Lambda(m_k)}. \tag{B.4}$$

The *combined likelihood of the IMM estimator* is given by

$$\Lambda(m_k) = \sum_{j=1}^{r}\sum_{i=1}^{r} \Lambda_j(m_k)\mu_i(t_{m_{k-1}})\pi_{ij}, \tag{B.5}$$

and each filter model (mode) *likelihood function* is given by

$$\Lambda_j(m_k) = |2\pi S_j(m_k)|^{-1/2} \exp\left\{-\tfrac{1}{2}\alpha_j(m_k)\right\}, \tag{B.6}$$

where the *normalized innovation squared* is

$$\alpha_j(m_k) = \nu_j(m_k)'[S_j(m_k)]^{-1}\nu_j(m_k), \tag{B.7}$$

and the *measurement residual* and *residual covariance* are

$$\nu_j(m_k) = z(m_k) - HF_j(\delta_k)\hat{x}_{0j}(t_{m_{k-1}}), \tag{B.8}$$

$$S_j(m_k) = R(m_k) + H[F_j(\delta_k)P_{0j}(t_{m_{k-1}})F_j(\delta_k)' + G_j(\delta_k)Q_j(\delta_k)G_j(\delta_k)']H', \tag{B.9}$$

where $\delta_k = t_{m_k} - t_{m_{k-1}}$.

The *updated state estimate* and *covariance* are given by

$$\hat{x}_j(t_{m_k}) = F_j(\delta_k)\hat{x}_{0j}(t_{m_{k-1}}) + W_j(m_k)\nu_j(m_k), \tag{B.10}$$

$$\begin{aligned} P_j(t_{m_k}) = {} & [F_j(\delta_k)P_{0j}(t_{m_{k-1}})F_j(\delta_k)' + G_j(\delta_k)Q_j(\delta_k)G_j(\delta_k)'] \\ & - W_j(m_k)S_j(m_k)W_j(m_k)', \end{aligned} \tag{B.11}$$

where the *filter gain* is

$$W_j(m_k) = [F_j(\delta_k)P_{0j}(t_{m_{k-1}})F_j(\delta_k)' + G_j(\delta_k)Q_j(\delta_k)G_j(\delta_k)']H'[S_j(m_k)]^{-1}. \tag{B.12}$$

And lastly, the *combined state estimate* and *covariance* are given by

$$\hat{x}(t_{m_k}) = \sum_{j=1}^{r} \hat{x}_j(t_{m_k})\mu_j(t_{m_k}), \tag{B.13}$$

$$P(t_{m_k}) = \sum_{j=1}^{r} [\hat{x}_j(t_{m_k}) - \hat{x}(t_{m_k})][\hat{x}_j(t_{m_k}) - \hat{x}(t_{m_k})]'\mu_j(t_{m_k}) + \sum_{j=1}^{r} P_j(t_{m_k})\mu_j(t_{m_k}). \tag{B.14}$$

References

[1] T.G. Allen, Multiple hypothesis tracking algorithms for massively parallel computers, *Proceedings of the SPIE Conference on Signal and Data Processing of Small Targets*, Orlando, FL, 1992.

[2] D.P. Atherton, E. Gul, A. Kountzeris and M.M. Kharbouch, Tracking multiple targets using parallel processing, IEE Proceedings on Control, Theory and Applications 137(1990)225–234.

[3] D.P. Atherton and H.-J. Lin, Parallel implementation of IMM tracking algorithm using transputers, IEE Proceedings on Radar, Sonar Navigation 141(1994)325–332.

[4] A. Averbuch, S. Itzikowitz and T. Kapon, Parallel implementation of multiple model tracking algorithms, IEEE Trans. Parallel and Distributed Systems 2(1991)242–252.

[5] E. Balas, D. Miller, J. Pekny and P. Toth, A parallel shortest augmenting path algorithm for the assignment problem, J. ACM (1991)985–1004.

[6] M. Balinski, Signature methods for the assignment problem, Operations Research 33(1985)527–536.

[7] Y. Bar-Shalom and X.R. Li, *Estimation and Tracking: Principles, Techniques and Software*, Artech House, Boston, MA, 1993.

[8] Y. Bar-Shalom and X.R. Li, Design of an interacting multiple model algorithm for air traffic control tracking, IEEE Trans. Control Systems Technology 1(1993) 186–194. Special issue on Air Traffic Control.

[9] Y. Bar-Shalom and X.R. Li, *Multitarget-Multisensor Tracking: Principles and Techniques*, YBS, Storrs, CT, 1995.

[10] D. Bertsekas, The auction algorithm: A distributed relaxation method for the assignment problem, Annals of Operations Research 14(1988)105–123.

[11] D. Bertsekas and D. Castanon, Parallel synchronous and asynchronous implementations of the auction algorithm, Parallel Computing 17(1991)707–732, 1991.

[12] T.L. Casavant and J.G. Kuhl, A taxonomy of scheduling in general-purpose distributed computing systems, IEEE Trans. Software Engineering 14(1988)141–154.

[13] T.C.K. Chou and J.A. Abraham, Load balancing in distributed systems, IEEE Trans. Software Engineering 8(1982)401–412.

[14] I.J. Cox and M.L. Miller, On finding ranked assignments with application to multi-target tracking and motion correspondence, IEEE Trans. Aerospace and Electronic Systems 32(1995)486-489.

[15] I. Cox, M. Miller, R. Danchick and G. Newnam, A comparison of two algorithms for determining ranked assignments with application to multitarget tracking and motion correspondence, IEEE Trans. Aerospace and Electronic Systems 33(1997)295–300.

[16] S. Deb, K.R. Pattipati and Y. Bar-Shalom, A multisensor-multitarget data association algorithm for heterogeneous sensors, IEEE Transactions on Aerospace and Electronic Systems 29(1993)560–568.

[17] S. Deb, M. Yeddanapudi, K. Pattipati and Y. Bar-Shalom, A generalized *S*-D assignment algorithm for multisensor-multitarget state estimation, IEEE Trans. Aerospace and Electronic Systems 33 (1997)523–538.

[18] U.B. Desai and B. Das, Parallel algorithms for Kalman filtering, *Proceedings of the American Control Conference*, Boston, MA, June 1985.

[19] O. Drummond, D. Castanon and M. Bellovin, Comparison of 2-D assignment algorithms for sparse, rectangular, floating point, cost matrices, J. of SDI Panels on Tracking, Institute for Defense Analyses, Alexandria, VA, No. 4/1990, December 1990, pp. 81–97.

[20] D.L. Eager, E.D. Lazowska and J. Zahorjan, Adaptive load sharing in homogeneous distributed systems, Proceedings IEEE 34(1986)662–675.

[21] K. Efe, Heuristic models of task assignment scheduling in distributed systems, IEEE Computer (June 1982)50–56.

[22] A.V. Goldberg, S.A. Plotkin, D.B. Shmoys and E. Tardos, Using interior-point methods for fast parallel algorithms for bipartite matching and related problems, SIAM J. Computing 21(1992)140–150.

[23] R. Janben, A note on superlinear speedup, Parallel Computing 4(1987)211–213.

[24] R. Jonker and A. Volgenant, A shortest augmenting path algorithm for dense and sparse linear assignment problems, J. Computing 38(1987)325–340.

[25] D. Parkinson, Parallel efficiency can be greater than unity, Parallel Computing 3(1986)261–262.

[26] K.R. Pattipati and S. Deb, Comparison of assignment algorithms with applications to the passive sensor data association problem, *Proceedings of the IEEE International Conferences on Control and Application*, Jerusalem, Israel, 1989.

[27] K.R. Pattipati, T. Kurien, R.-T. Lee and P.B. Luh, On mapping a tracking algorithm onto parallel processors, IEEE Trans. Aerospace and Electronic Systems 26(1990)774–791.

[28] K.R. Pattipati, S. Deb, Y. Bar-Shalom and R. Washburn, A new relaxation algorithm and passive sensor data association, IEEE Trans. Automatic Control 37(1992)197–213.

[29] W.P. Pierskalla, The multidimensional assignment problem, Operations Research 16(1968)422–431.

[30] K.A.L. Piriyakumar and C.S.R. Murthy, Optimal scheduling of parallel tasks of tracking problem onto multiprocessors, IEEE Trans. Aerospace and Electronic Systems 32(1996)722–731.

[31] A. Poore and N. Rijavec, Multitarget tracking, multidimensional assignment problems, and Lagrangian relaxation, *Proc. SDI Panels on Tracking*, August, 1991, pp. 51–74.

[32] A.B. Poore and N. Rijavec, A Lagrangian relaxation algorithm for multidimensional assignment problems arising from multitarget tracking, SIAM J. Optimization 3(1993)544–563.

[33] A.B. Poore and N. Rijavec, Partitioning multiple data sets via multidimensional assignments and Lagrangian relaxation, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, American Mathematical Society, 1995.

[34] R.L. Popp, K.R. Pattipati, Y. Bar-Shalom and R.R. Gassner, Multitarget tracking slgorithm parallelization for distributed-memory computing systems, *Proceedings of the 5th IEEE Symposium on High Performance Distributed Computing, Syracuse*, NY, August 1996.

[35] R.L. Popp, K.R. Pattipati and R.R. Gassner, Multitarget tracking parallelization for high-performance computing architectures, *Proceedings of the 4th SCS Symposium on High Performance Computing*, New Orleans, LA, April 1996.

[36] R.L. Popp, K.R. Pattipati and R.R. Gassner, Heuristic task assignment algorithms Applied to multi-sensor-multitarget tracking, *SPIE Conference on Signal and Data Processing of Small Targets* (#2759), Orlando, FL, April 1996.

[37] R.L. Popp, K.R. Pattipati, Y. Bar-Shalom and R.A. Ammar, Shared-memory parallelization of the data association problem in multitarget tracking, IEEE Trans. Parallel and Distributed Systems 8(1997).

[38] R.L. Popp, K.R. Pattipati, Y. Bar-Shalom and M. Yeddanapudi, Parallelization of a multiple model multitarget tracking algorithm with superlinear speedups, IEEE Trans. Aerospace and Electronic Systems 33(1997)281–290.

[39] D.B. Reid, An algorithm for tracking multiple targets, IEEE Trans. Automatic Control 24(1979)423–432.

[40] I.B. Rhodes, A parallel decomposition for Kalman filters, IEEE Trans. Automatic Control 35(1990) 322–324.

[41] C.C. Shen and W.H. Tsai, A graph matching approach to optimal task assignment in distributed computing systems using a minimax criterion, IEEE Trans. Computer 34(1985)197–203.

[42] J.A. Stankovic, An application of Bayesian decision theory to decentralized control of job scheduling, IEEE Trans. Computer 34(1985)117–130.

[43] Y.T. Wang and R.J.T. Morris, Load sharing in distributed systems, IEEE Trans. Computer 34(1985) 204–217.

[44] J. Xu and K. Hwang, Heuristic methods for dynamic load balancing in a message-passing multi-computer, J. Parallel and Distributed Computing 18(1993)1–13.

[45] M. Yeddanapudi, Y. Bar-Shalom, K.R. Pattipati and R.R. Gassner, MATSurv: Multisensor air traffic surveillance system, *Proceedings of the SPIE Conference on Signal and Data Processing of Small Targets* (#2561), San Diego, CA, July 1995.

[46] M. Yeddanapudi, Y. Bar-Shalom and K.R. Pattipati, IMM estimation for multitarget-multisensor air traffic surveillance, IEEE Proceedings 85(1997)80–94.