Data Association in Multi Target Tracking Using Cross Entropy Based Algorithms

Daniel Sigalov and Nahum Shimkin, *Senior Member, IEEE*

Abstract

Multiple-target tracking (MTT) poses difficult computational challenges related to the measurement-to-track data association problem, especially in the presence of spurious and missing measurements. Different approaches have been proposed to tackle this problem, including various approximations and heuristic optimization tools. The Cross Entropy (CE) method and the related Parametric MinxEnt (PME) method are recent optimization heuristics that have proved useful in many combinatorial optimization problems. They are akin to evolutionary algorithms in that a population of solutions is evolved, however generation of new solutions is based on statistical methods of sampling and parameter estimation. In this work we apply the Cross-Entropy method and its recent MinxEnt variant to the multi-scan version of the data association problem in the presence of misdetections, false alarms, and unknown number of targets. We formulate the algorithms, explore via simulation their efficiency and performance compared to other recently proposed techniques, and show that they obtain state-of-the-art performance in challenging scenarios.

Index Terms

Target tracking, data association, combinatorial optimization, Kalman filtering, cross-entropy method, Monte-Carlo methods

I. Introduction

Multiple-target tracking (MTT) is an essential component of surveillance-related systems. A general formulation of the problem assumes an unknown and varying number of targets that are continuously moving in a given region. In the single-sensor version, the states of these targets are sampled by the sensor and the noisy measurements are provided to the tracking system. The detection probability is not perfect and the targets may go undetected at some sampling intervals. In addition, there are spurious reports of possible targets, or clutter measurements which arise independently of the targets of interest. A

The authors are with the Department of Electrical Engineering, Technion – Israel Institute of Technology, 32000. {dansigal@tx, shimkin@ee}.technion.ac.il

Preliminary versions of this paper have been presented at: The 47th Israel Annual Conf. on Aerospace Sciences, Tel-Aviv, Israel, Feb. 2007; and at ValueTools'08 – the 3rd International Conf. on Performance Evaluation Methodologies and Tools, Athens, Greece, Oct. 2008.

primary task of the MTT system is data association, namely, partitioning the measurements into disjoint sets, each generated by a single source (target or clutter). The secondary goal is estimation of the states based on the measurements originating from the targets of interest. The data association problem may be formulated in several ways. In single scan data association the raw measurements are processed one scan at a time and the target states are updated accordingly. Alternatively, several sets of measurements may be collected and processed together in batch mode – this is the multi scan data association. For an illustration see Fig. 4.

Several methods now exist to handle the data association problem. These may be roughly grouped into two types: Bayesian and non-Bayesian. Among the Bayesian methods, there is the well known Joint Probabilistic Data Association Filter (JPDA) [1], which is a single scan filter where the states of existing targets are to be updated based on the latest set of measurements (scan). Data association is handled by summing over the probabilities of all feasible partitions, where no two targets can share a measurement and each target may be a source of at most one measurement per scan. A shortcoming of the basic JPDA is its inability to initiate and terminate tracks. In addition, calculating the probabilities of all feasible events is NP-hard [2] in the number of targets and measurements and the calculation becomes intractable even for a moderate size of the problem. Another well known approach is the multiple hypotheses tracker (MHT) [3], in which each hypothesis associates past observations with targets and, as a new set of observations arrives, a new set of hypotheses is formed by augmenting the previous ones. The hypothesis with the highest posterior is returned as a solution. MHT is capable of initiating and terminating tracks. However, the number of hypotheses involved in the calculation grows exponentially over time. Thus, in order to overcome this computational complexity, certain pruning and clustering methods must be used at expense of optimality. Another method for handling data association in the Bayesian manner is the Probabilistic Multi-Hypothesis Tracker (PMHT) [4]. The PMHT algorithm employs a more lenient measurement model in comparison to PDAF and MHT. Whereas the latter assume that a target can generate at most one measurement per scan, PMHT drops this constraint, and posits the measurement/target association process as independent across measurements. Similarly to JPDA, the basic PMHT assumes a fixed and known number of targets in the scenario under consideration.

The non-Bayesian approach is characterized by hard measurement-to-track association, such that some cost function is maximized. The problem may then be reformulated as an integer programming problem [5] or, more precisely, as a multidimensional assignment problem, which is NP-hard when the number of sets (scans) to be assigned is greater than or equal to 3 [6]. Therefore, for the multi scan data association, one should invoke some approximations schemes for the multidimensional assignment such

as Lagrangian relaxation techniques that relax some of the problem constraints and solve the relaxed problem by utilizing linear programming techniques [7]. Note, however, that when there are only two sets of data to be assigned, there exist exact, polynomial time solutions which have been combined with particle filter based algorithms in the context of multi-target tracking [8]. Additionally, an algorithm for assigning a set of measurements to the set of predicted target states using the competitive Hopfield neural network was proposed in the recent work [9]. Nonetheless, the simulations presented therein included relatively small tracking scenarios.

Another option to solve the multi scan data association problem is by utilizing stochastic search methods. In [10] the problem was solved by applying the Markov Chain Monte Carlo (MCMC) method to obtain the partition with maximum posterior. Using the Metropolis algorithm, the authors proposed a set of moves for modifying a partition of the measurements, such that sampling from the posterior distribution was possible after a few thousands of moves. They showed a remarkable performance of the algorithm in comparison to the MHT method in terms of accuracy of the solution and running time. However, the algorithm is still susceptible to getting trapped in a strong local maxima. Such behavior is typical of local search algorithms.

The main contribution of this paper is the development of feasible algorithms that solve the multi scan data association problem and are capable of initiating and terminating a varying number of tracks. The general setup and problem definition are very similar to those in [10]. In particular, the basic assumptions for the data association problem are identical to those used in JPDA, MHT, and MCMCDA. Namely, we assume that each measurement may either belong to at most one target or be classified as a false alarm, and at most one measurement may be associated with a given target at a time. (A detailed discussion of these assumptions is provided in section II). The solution approach, however, is different. We invoke the Cross Entropy (CE) method [11] and the related Parametric MinxEnt (PME) method [12] in order to obtain the partition with the highest posterior. CE based schemes are approaches for combinatorial and continuous optimization, and (originally) for estimation of rare-events probabilities. They are inherently global search methods and, therefore, may reduce the risk of getting stuck in shallow local maxima. The main idea is representing the solution space with a set of parameters and defining a probability distribution on these parameters. Then, two successive steps are iterated – sampling from the existing distribution, and updating this distribution using a subset of *elite* (better-valued) samples. The underlying principle of solution improvement is thus akin to evolutionary optimization algorithms (see, e.g. [13]), but the solution generation mechanism is different, and the whole scheme has very few meta-parameters that need to be tuned. The resulting Cross Entropy Data Association (CEDA) and Parametric MinxEnt Data

Association (PMEDA) algorithms are applied here to challenging tracking scenarios, and show improved performance relative to current state-of-the-art techniques.

The structure of this paper is as follows. We formally state the (discrete-time) general multiple-target tracking problem in section II. In section III we outline the CE and PME methods for combinatorial optimization. In section IV we present general purpose CEDA and PMEDA algorithms for multiple target tracking. The algorithms are applied in simulation to dense tracking scenarios and their performance is compared with several popular algorithms in section VI.

II. Problem Definition

A. Preliminaries

Consider a surveillance scenario of duration $T \in \mathbb{Z}^+$. There are K targets moving around the surveillance region $\mathcal{R}$ for some duration $[t_i^k, t_f^k] \subset [1, T]$ for $k = 1, \ldots, K$ where K is an unknown integer. The volume of $\mathcal{R}$ is V and it is scanned periodically by a single sensor having scan period T_s normalized to one time unit. The notation $[t_i, t_j]$ should be interpreted as $\{t_i, t_i + 1, ..., t_j\}$.

B. Target Model

In this subsection we describe the target modeling commonly used in the target tracking literature (see e.g. [14]). Each target k starts at a random position in $\mathcal{R}$ at time t_i^k, moves around $\mathcal{R}$ until t_f^k and disappears. An existing target may disappear at each sampling time with probability p_z and persists with probability $1 - p_z$. The number of new targets arising at each time in $\mathcal{R}$ is modeled to have a Poisson distribution with a parameter $\lambda_b V$, where λ_b is the birth rate of new targets per unit time and volume. The initial position of a new target is uniformly distributed over $\mathcal{R}$. We describe the motion of a target by the discrete-time dynamics $F : \mathbb{R}^d \to \mathbb{R}^d$, where d is the dimension of the state variable, and $x_t \in \mathbb{R}^d$ is the state at time t. The target k moves according to $x_{t+1}^k = F(x_t^k, w_t^k),\ t = t_i^k, \ldots, t_f^k - 1$ where $w_t^k \in \mathbb{R}^d$ are white noise processes. In this work, we consider the same linear dynamical model for each target, namely, if a target is observed ℓ times at $t_i,\ i = 1, ..., \ell$, its dynamic model may be expressed as:

$$x_{t_{i+1}} = A(t_{i+1}, t_i) x_{t_i} + G(t_{i+1}, t_i) w_{t_i}, \tag{1}$$

where w_{t_i} is a white Gaussian noise with covariance matrix Q. A and G are matrices of appropriate sizes, with entries determined by the sampling interval (t_i, t_{i+1}) for each i.

C. Sensor and Measurement Models

We assume that a single sensor scans the surveillance region periodically with scan time T_s of one time unit. Noisy observations of the position of each target are obtained with detection probability P_d.

In addition, the sensor generates false alarms, whose number is assumed to have a Poisson distribution with parameter $\lambda_f V$, where λ_f is the false alarm rate per unit time per unit volume. The origin of each observation (i.e. target or false alarm) is not a-priori known, since each observation is assumed to carry only the cartesian position and the corresponding time tag.

Let n_t be the number of observations at time t, including both noisy observations and false alarms. Let $y_t^j \in \mathbb{R}^m$ denote the j-th observation at time t for $j = 1, \ldots, n_t$, where m is the dimensionality of each observation vector. Each target generates a unique observation at each sampling time if it is detected. We assume a linear observation model, namely, an arbitrary observation at time t_i, $y_{t_i}^j$, is generated as follows:

$$y_{t_i}^j = \begin{cases} C(t_i)x_{t_i} + v_{t_i}, & y_{t_i}^j \text{ is object originated} \\ u_t, & \text{otherwise} \end{cases}, \tag{2}$$

where $v_{t_i} \in \mathbb{R}^m$ is a white Gaussian noise independent of w_{t_i} with covariance matrix R, C is a matrix of appropriate size, and $u_t \sim \text{Unif}(\mathcal{R})$ is the random process of false alarms, assumed to be uniformly distributed in space.

D. Solution Space and Optimization Criteria

Dealing with hard (as opposed to soft) data association we seek for a partition of the measurements into disjoint sets. One of these sets is the collection of false alarms and the others are collections of measurements originating from the same target – one set per target. Let $Y_t = \left\{ y_t^j : j = 1, \ldots, n_t \right\}$ be the set of observations at time t, and $Y_{1:T} = \bigcup_{t \in \{1,\ldots,T\}} Y_t$ be the set of all observations. Ω is defined to be the set of partitions of $Y_{1:T}$ such that, for $\omega \in \Omega$:

1) $\omega = \{\tau_0, \tau_1, \ldots, \tau_K\}$.
2) $\bigcup_{k=0}^{K} \tau_k = Y_{1:T}$ and $\tau_i \cap \tau_j = \emptyset$ for $i \neq j$, i.e., each measurement belongs to at most one target, or is classified as a false alarm.
3) τ_0 is considered as the set of false alarms and $\tau_k,\ k \geq 1$ is considered as the kth track – a set of measurements that are attributed to the kth target[1]

$$\tau_i = \left\{ y_{t_1}^{i_1}, y_{t_2}^{i_2}, \ldots, y_{t_j}^{j} \mid j \in \mathbb{Z}^+, t_1 < t_2 < \ldots < t_j \right\},$$

i.e. each track is comprised of a subset of measurements ordered according to their time tags in an increasing order.

[1]We shall refer to the above set of measurements as a track, although usually a track is the estimated trajectory, that is after filtering (or smoothing) out the measurement noise.

4) $|\tau_k \cap Y_t| \leq 1$ for $k = 1, \ldots, K$ and $t = 1, \ldots, T$. That is, each measurement belongs to one track at most.
5) $|\tau_k| \geq 2$ for $k = 1, \ldots, K$, where $|\tau_k|$ denotes the cardinality of τ_k. That is, a track must contain at least two measurements.

We make two additional assumptions as part of the problem formulation.

A. The maximal velocity of any target is bounded by a known constant $v_{\max}$.
B. The number of consecutive missing observations of any track is bounded by a known constant $d_{\max}$. This assumption may be used as a criterion to distinguish an event of a new targets appearance from an event of a continuation of an existing track.

For further discussion of the last two restrictions the reader is referred to [10].

Any partition ω in Ω is said to be valid or feasible. Once a partition $\omega \in \Omega$ is chosen, the tracks $\tau_1, \ldots, \tau_K \in \omega$ and the set of false alarms $\tau_0 \in \omega$ are completely determined. We thus face a problem of choosing the best (in some appropriate sense) partition ω^* given the set of observations $Y_{1:T}$. This is the so-called measurement oriented approach to data association.

A natural criterion for this approach is the Maximum a-Posteriori probability [3], [10], [14]. That is, looking for the optimal partition ω^* in the MAP sense:

$$\omega^* = \arg\max_{\omega \in \Omega} \mathbb{P}\left\{\omega \mid Y_{1:T}\right\}. \tag{3}$$

E. The Posterior Probability

Expressions for the posterior probability $\mathbb{P}\left\{\omega \mid Y_{1:T}\right\}$ are commonly used in the target tracking literature [1], [3], [10], [14]. The expression (5) below can be obtained from that of [14] and the complete derivation is available in [15]. For each partition ω let m_t denote the number of targets at time t, a_t be the number of new targets at time t, z_t – the number of targets terminated at time t, d_t – the number of target detections at time t, u_t – be number of undetected targets at time t (according to ω i.e. missing measurements within a path), and f_t be number of false alarms at time t. Bearing in mind that n_t is the total number of measurements obtained at time t, it may be easily verified that the following relations hold:

$$\begin{aligned} m_t &= m_{t-1} + a_t - z_t \\ u_t &= m_t - d_t \\ f_t &= n_t - d_t. \end{aligned} \tag{4}$$

The final expression reads,

$$\mathbb{P}\{\omega \mid Y_{1:T}\} = \frac{1}{Z_0} \prod_{\tau\in\omega\setminus\{\tau_0\}} \prod_{i=2}^{|\tau|} \mathcal{N}\left(\tau(t_i); \hat{y}_{t_i}(\tau), B_{t_i}(\tau)\right) \cdot \prod_{t=1}^{T} p_z^{z_t}(1-p_z)^{m_{t-1}-z_t} \cdot P_d^{d_t}(1-P_d)^{u_t}\lambda_b^{a_t}\lambda_f^{f_t}, \tag{5}$$

where Z_0 is a constant that does not depend on ω, $\mathcal{N}(x;\mu,\Sigma)$ is the Gaussian density with mean μ and covariance Σ evaluated at x, $\tau(t_i)$ is the i-th measurement associated with track τ, $\hat{y}_{t_i}(\tau)$ is the i-th predicted measurement obtained from the standard Kalman applied to the measurements associated with track τ, and $B_{t_i}(\tau)$ is the corresponding innovation covariance.

III. Background on CE and PME

A. The CE Method for Combinatorial Optimization

Let $\mathcal{X}$ be a finite set of elements and $S(\cdot)$ be a performance function defined on $\mathcal{X}$. Our goal is to find the maximum of $S(\cdot)$ over $\mathcal{X}$. Namely[2],

$$S(\mathbf{x}^*) = \gamma^* = \max_{\mathbf{x}\in\mathcal{X}} S(\mathbf{x}). \tag{6}$$

A convenient way to introduce the CE method is from the parameter estimation perspective. When solving optimization problems using the CE method, one searches for a probability distribution concentrated near the global extremum of the objective function. Assume we can define a parameterized probability density function $f(\mathbf{x};\mathbf{v})$ on the set $\mathbf{x}\in\mathcal{X}$. The goal is to construct a sequence of parameter vectors $\mathbf{v}_1, \mathbf{v}_2, ...$ such that $f(\mathbf{x};\mathbf{v}_t)$ becomes concentrated around the global optimum $\mathbf{x}^*$ as t increases. This goal is achieved by sampling from $f(\mathbf{x};\mathbf{v}_t)$ and constructing the next parameter vector $\mathbf{v}_{t+1}$ as the Maximum Likelihood estimate of the distribution parameter based on the elite samples. Namely,

$$\hat{\mathbf{v}}_{t+1} = \arg\max_{\mathbf{v}} \ln f(\tilde{\mathbf{X}}_1, ..., \tilde{\mathbf{X}}_{N\rho}; \mathbf{v}), \tag{7}$$

where $\tilde{\mathbf{X}}_1, ..., \tilde{\mathbf{X}}_{N\rho}$ are the $N\rho$ elite samples, achieving the best performance in the current set, and $f(\tilde{\mathbf{X}}_1, ..., \tilde{\mathbf{X}}_{N\rho}; \mathbf{v})$ is the joint density evaluated at $\tilde{\mathbf{X}}_1, ..., \tilde{\mathbf{X}}_{N\rho}$. The new parameter vector defines a new distribution from which we can sample again and repeat the procedure. Instead of updating the parameter vector $\mathbf{v}_t$ directly via the solution of (7) one may use the smoothed update which reduces the probability that some components of $\mathbf{v}_t$ will become degenerate at early stages,

$$\hat{\mathbf{v}}_t = \alpha\tilde{\mathbf{v}}_t + (1-\alpha)\hat{\mathbf{v}}_{t-1},\ 0 \le \alpha \le 1, \tag{8}$$

[2]Note that the variable $\mathbf{x}$ below is not related to the state variable x_t defined in section II. We use this notation to comply with conventions.

where $\tilde{\mathbf{v}}_t$ is the solution obtained from (7). The whole procedure is summarized in Alg. 1. The stopping

Algorithm 1 The CE Algorithm for Optimization.

1: Define $\hat{\mathbf{v}}_0 = \mathbf{u}$. Set $t = 1$ (level counter)
2: Generate $\mathbf{X}_1, \ldots, \mathbf{X}_N$ from $f(\cdot; \mathbf{v}_{t-1})$ and compute the sample $(1-\rho)$-quantile $\hat{\gamma}_t$ of the performances.
3: Find the MLE of the new parameter $\mathbf{v}_t$ based on the set of the elite samples. Namely, solve (7).
4: Smooth the estimate via (8).
5: If stopping criteria are met - stop, otherwise set $t = t + 1$ and reiterate from step 2.

criteria in step 5 of Alg. 1 may be lack of significant improvement for several iterations, or convergence to a degenerate distribution.

Assume now that $\mathbf{X} = (X_1, ..., X_n)$ is a random vector such that each X_i is a discrete random variable that can assume a finite number of values $\{a_1, ..., a_m\}$. The important observation that makes the CE method very easy to apply to various optimization problems, such as the Traveling Salesperson and MaxCut [6], [11], is that in this case there is a simple componentwise analytical solution to (7) that reads [11]

$$\hat{v}_{jk} = \frac{\sum_{i=1}^{N} \mathbb{1}_{\{X_{ij}=a_k\}} \mathbb{1}_{\{S(\mathbf{X}_i)\geq\hat{\gamma}_t\}}}{\sum_{i=1}^{N} \mathbb{1}_{\{S(\mathbf{X}_i)\geq\hat{\gamma}_t\}}}, \tag{9}$$

where X_{ij} is the j-th element of the i-th sample $\mathbf{X}_i$ drawn from $f(\mathbf{x}, \mathbf{v}_{t-1})$ and $\mathbb{1}_{\{A\}}$ is the indicator function of A. Namely, the updated value of each parameter is the relative frequency of the appearance of the corresponding value in the current elite sample.

B. The PME Method for Combinatorial Optimization

Recall that the goal of the CEM was to find a "good" sampling density concentrated near the global optimum of the problem at hand. Another option is to consider the (single constrained) Minimum Cross Entropy (MinxEnt) program that reads

$$\min_{f(\mathbf{x})} \left\{ \mathcal{D}(f|h) = \int \ln \frac{f(\mathbf{x})}{h(\mathbf{x})} f(\mathbf{x})\, d\mathbf{x} = \mathbb{E}_f \left[\ln \frac{f(\mathbf{X})}{h(\mathbf{X})} \right] \right\} \tag{10}$$

subject to the first moment constraint:

$$\mathbb{E}_f S(\mathbf{X}) = \gamma, \tag{11}$$

where f and h are n-dimensional pdf's, $S(\mathbf{x})$ is the known performance function, $\mathbf{x} \in \mathbb{R}^n$, and γ is a performance close to the optimal γ^* in (6). Assuming $h(\cdot)$ is a known pdf that incorporates all the available prior information, the problem is to find the closest to $h(\mathbf{x})$ density $f(\cdot)$ in the Kullback-Leibler

sense subject to the moment constraint. If no prior information is available, $h(\mathbf{x})$ is taken to be uniform. We shall restrict ourselves to the discrete distributions $f(\mathbf{x})$ and $h(\mathbf{x})$ parameterized by parameter vectors $\mathbf{v}, \mathbf{u}$ respectively – $f(\mathbf{x}, \mathbf{v})$ and $h(\mathbf{x}, \mathbf{u})$. The solution of the MinxEnt program is [16]

$$f(\mathbf{x}, \mathbf{v}^*) = \frac{h(\mathbf{x}, \mathbf{u}) \exp\{-S(\mathbf{x})\lambda\}}{\mathbb{E}_{\mathbf{u}}\left[\exp\{-S(\mathbf{X})\lambda\}\right]}, \tag{12}$$

where λ is a constant (temperature) obtained from the following equation

$$\frac{\mathbb{E}_{\mathbf{u}}\left[S(\mathbf{X}) \exp\{-S(\mathbf{X})\lambda\}\right]}{\mathbb{E}_{\mathbf{u}}\left[\exp\{-S(\mathbf{X})\lambda\}\right]} = \gamma, \tag{13}$$

and $\mathbf{X} \sim h(\mathbf{x}, \mathbf{u})$. For $\gamma = \gamma^*$, the optimal temperature is $\lambda^* = -\infty$ and the optimal density $f^*(\mathbf{x})$ is a Dirac delta function located at $\mathbf{x}^*$. Given a successful choice of γ and obtaining the corresponding value of λ we could, in principle approximate the optimal $\mathbf{x}^*$ by generating samples from (12). However, sampling from such distribution is not a trivial task. In fact, no efficient methods are known to exist [17]. Thus, we shall approximate the distribution (12) as a product of marginal densities which will allow easy sampling similarly to the basic CE method. Note that if $h(\mathbf{x}, \mathbf{u})$ is a discrete (multidimensional) distribution with finite support, then so is $f(\mathbf{x}, \mathbf{v}^*)$ and, consequently, all its marginal distributions. Thus, all these distributions are completely determined by their parameters which may be calculated as follows. Assuming as before, that $\mathbf{X} = (X_1, ..., X_n)$ is a random vector such that each X_i is a discrete random variable that can assume a finite number of values $\{a_1, ..., a_m\}$, then the PME estimator of

$$v_{jk} \triangleq \mathbb{P}\{X_j = a_k\} = \mathbb{E}_{\mathbf{v}}\left[\mathbb{1}_{\{X_j = a_k\}}\right]$$

is [12]

$$\hat{v}_{jk} = \frac{\sum_{i=1}^{N} \mathbb{1}_{\{X_{ij} = a_k\}} \exp\{-S(\mathbf{X}_i)\lambda\}}{\sum_{i=1}^{N} \exp\{-S(\mathbf{X}_i)\lambda\}}. \tag{14}$$

It is readily seen that (14) is essentially the same as (9) with indicators $\mathbb{1}_{\{S(\mathbf{X}_i) \geq \hat{\gamma}_t\}}$ being replaced by exponentials $\exp\{-S(\mathbf{X}_i)\lambda\}$. The optimal "temperature" parameter λ is obtained from the (numerical) solution of the stochastic version of (13) such that the single constraint in (11) is satisfied:

$$\frac{\sum_{i=1}^{N} S(\mathbf{X}_i) \exp\{-S(\mathbf{X}_i)\lambda\}}{\sum_{i=1}^{N} \exp\{-S(\mathbf{X}_i)\lambda\}} = \gamma. \tag{15}$$

Similarly to the CE method, we invoke a multi-stage procedure where a sequence of reference parameters $\{\mathbf{v}_t, t \geq 0\}$, a sequence of levels $\{\gamma_t, t \geq 1\}$ and a sequence of temperatures $\{\lambda_t, t \geq 1\}$ are generated. As before, we shall use the latest available information for the prior density. Namely, at stage t, $h(\mathbf{x}, \mathbf{u}) = f(\mathbf{x}, \mathbf{v}_t)$. The whole optimization procedure is summarized in Algorithm 2. For both CE and PME optimization routines we expect the parameter vectors to converge to degenerate ones, such that by

Algorithm 2 The PME Algorithm for Optimization.

1: Define $\hat{\mathbf{v}}_0 = \mathbf{u}$. Set $t = 1$ (level counter).
2: Generate $\mathbf{X}_1, \ldots, \mathbf{X}_N$ from $f(\cdot; \hat{\mathbf{v}}_{t-1})$ and compute the performance mean from $\hat{\gamma}_t = \mathbb{E}_{\hat{\mathbf{v}}_{t-1}} S(\mathbf{X})$.
3: Use the same sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ and solve the stochastic program (15). Denote the solution by $\hat{\lambda}_t$.
4: Update $\hat{\mathbf{v}}_t$ componentwise via (14).
5: If stopping criteria are met - stop, otherwise set $t = t + 1$ and reiterate from step 2.

sampling from the final distribution we shall always obtain optimal or near-optimal solutions. Unlike CE, all samples are used to update the parameters in the PME method (although modifications are possible) and the solution for λ in (15) requires a line-search procedure and usually cannot be done analytically. We thus expect each iteration of the PME method to be slower than its CE counterpart. All "tuning" methods used for CE, such as smoothing and stopping rules, are directly applicable here as well.

IV. CE Based Data Association

In this section we develop a family of CE-based algorithms to solve the multi-scan multi-target tracking problem. Since the only difference between the CE and PME methods for combinatorial optimization is the updating scheme of the parameters, we shall describe both methods together and explain the differences when needed. In order to apply the CE and PME methods to our problem we must specify the parameterized family of pdfs $\{f(\cdot; \mathbf{v})\}$, as well as the parameter vector $\mathbf{v}$ with respect to the data association problem, and the procedure for sampling the solutions from it. We describe these in the following subsections. To this end, we encode the optimization problem as a graph and introduce the randomization on the graph's edges or nodes [11].

A. The Connectivity Graph

Let $n = \sum_{t=1}^{T} |Y_t| = |Y_{1:T}|$ be the total number of observations (both noisy detections and false alarms). We define $G = (V, E)$ to be the *basic connectivity graph* of the problem, where the set of nodes $V = Y_{1:T}$ is the set of all measurements as defined above, and the graph edges are

$$E = \{(y_{t_1}, y_{t_2}) \mid y_{t_1}, y_{t_2} \in Y,\ t_1 < t_2, \|y_{t_2} - y_{t_1}\| \leq (t_2 - t_1) v_{\max},\ t_2 - t_1 \leq d_{\max}\}. \tag{16}$$

This graph connects every node (measurement) with any other node that can be an immediate successor in a feasible track, subject to speed and separation constraints. A feasible track, which is a set of observations with increasing time tags, is represented as a *path* in G, that is

$$\tau = \left\{y^i, i = 1, 2, \ldots, j \mid (y^i, y^{i+1}) \in E,\ i = 1, \ldots, j-1\right\}.$$

The nodes of the graph represent measurements (both noisy detections and false alarms), and the edges represent the possible event that their endpoints are successive measurements from the same target. Note that a node y that has no incoming and outgoing edges is a false alarm by default, and may be removed from the graph and permanently added to the set of false alarms τ_0. Henceforth, we shall assume that all such nodes have been removed from the graph. We identify each valid partition of the measurements with the corresponding partition of the graph nodes. Thus, the goal is to find a partition of the the graph nodes into a set of vertex-disjoint paths $\{\tau_i, i \geq 1\}$ (which will represent tracks) and a set of isolated nodes τ_0 (which will represent false alarms) such that the posterior $\mathbb{P}\{\omega \mid Y_{1:T}\}$, defined in (5), is maximized. Let $S(\omega) = \mathbb{P}\{\omega \mid Y_{1:T}\}$ denote the cost of a partition $\omega \in \Omega$.

In the discussion below we shall denote the graph nodes as $n_i,\ i = 1, ..., |V|$. When no confusion occurs we shall use the notation $i,\ i = 1, \ldots, |V|$ to refer to n_i. We shall assume that the nodes are ordered according to the time stamps of the corresponding measurements. Namely, if n_{i_1} and n_{i_2} belong to scans t_1 and t_2 respectively, such that $t_2 > t_1$, then $i_2 > i_1$. The ordering of the nodes representing measurements within the same scan is arbitrary. An edge between node n_i and node n_j (or, equivalently, between i and j) will be denoted (i, j). An example of the connectivity graph is shown in Fig. 1(a).

B. Distribution of Feasible Partitions

Recall that we need to define a probability distribution on the set of feasible partitions Ω, parameterized by a parameter vector $\mathbf{v}$. This vector will comprise of the following elements:

- $p_b(i)$, $i \in V$: The probability that measurement i, or equivalently node i in the connectivity graph $G = (V, E)$, is an initial node in some path (including a single-node path, namely a false alarm).
- $p_f(i)$, $i \in V$: The probability that a path that reaches node i terminates at it.
- p_{ij}, $(i, j) \in E$: The probability that node j follows node i in a path that goes through i.

All probabilities are naturally required to be in $[0, 1]$. In addition, we require that

$$p_f(i) + \sum_{j:(i,j)\in E} p_{ij} = 1,\ \forall\, i \in V.$$

This means that a path going through node i has probability 1 of either continuing to a neighboring node j or terminating in i. The probabilities $\{p_f(i),\ i \in V\}$ and $\{p_{ij},\ (i, j) \in E\}$ define a stochastic matrix

P which reads

$$P = \begin{pmatrix} 0 & p_{12} & p_{13} & \dots & p_f(1) \\ 0 & 0 & p_{23} & \dots & p_f(2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & p_f(|V|) \\ 0 & 0 & 0 & \dots & 1 \end{pmatrix} \tag{17}$$

(the last line corresponds to an absorbing final state f which terminates each path). The probability of a single path $\tau = (i_1, \dots, i_m)$ in G will thus be proportional to

$$P_{\text{path}}(\tau) \triangleq p_b(i_1) \quad \prod_{l=1}^{m-1} p_{i_l, i_{l+1}} \Bigg) p_f(i_m). \tag{18}$$

Denoting by $\mathcal{B}$ the set of initial nodes of the paths $\{\tau_1, ..., \tau_K\}$ in a partition $\omega = \{\tau_0, \tau_1, ..., \tau_K\}$, the probability of a *feasible* partition ω can now be specified as

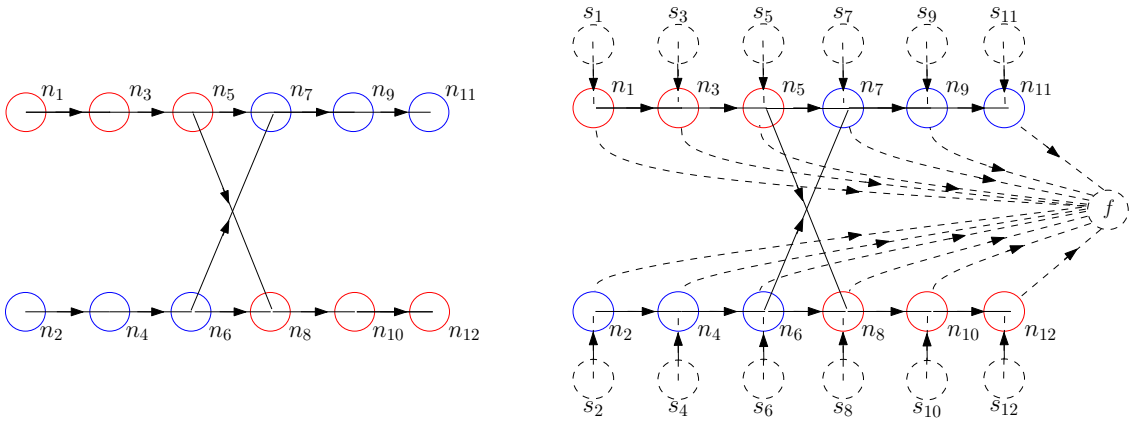
$$f(\omega; \mathbf{v}) = \frac{1}{Z_1} \prod_{k=1}^{K} P_{\text{path}}(\tau_k) \prod_{i \notin \mathcal{B}} (1 - p_b(i)), \tag{19}$$

where Z_1 is a normalization constant that does not depend on ω. The probability of infeasible partitions is identically set to zero.

The probability distribution (19) on the set of feasible partitions Ω is best interpreted through the following sampling process (the actual sampling process we use will be described later in subsection IV-D). For each node $i = 1, ..., |V|$, an independent Bernoulli random variable with success probability $p_b(i)$ is drawn and determines whether that node is a first node in a path. If it is, we start a random walk from i according to the transition probabilities $\{p_{ij}\}$ in (17), until the terminal state f is reached. These paths correspond to the K tracks $\{\tau_1, ..., \tau_K\}$. All nodes that were not visited in any of these paths are classified as false alarms and allocated to τ_0. The resulting partition ω is a valid sample if $\omega \in \Omega$. If ω is not a feasible partition (i.e., it has intersecting paths or paths below the minimal required length), it is rejected and the above procedure is repeated until a feasible partition is reached.

C. The Augmented Connectivity Graph

It will be convenient for illustration and implementation purposes to incorporate in a single graph all the relevant options (and their probabilities), including track initiation and termination. This will be done by augmenting the basic connectivity graph by introducing new nodes and edges. In order to incorporate the possibility of termination of a target, we introduce an additional "sink" node f that represents termination of a track. Each other node n_i in the basic connectivity graph is connected to



(a) The Connectivity Graph (b) The Augmented Connectivity Graph

Figure 1. An example of the connectivity graph (a) and of the ACG (b). Solid circles represent the actual measurements. Node i is denoted as n_i. Dashed circles are the nodes added to the connectivity graph to obtain the ACG.

f by a directed edge represents the event that n_i corresponds to the last detection of the target prior to termination. In order to handle target initiation in arbitrary time and place, we introduce additional n 'start' nodes into the graph – one for each original node of the basic graph. These will be labeled $s_1, s_2, ..., s_n$. Each new node s_i has an outgoing edge leading to the corresponding node $n_i \in V$. The event represented by this edge is that node n_i is the first detection of a track. A path that represents a track with j measurements now contains $j+2$ nodes as follows – $\{s_{n_1}, n_1, ..., n_j, f\}$. We call the extended graph an *augmented connectivity graph* (ACG), denoted $G_A = (V_A, E_A)$, and refer to the original nodes in the basic connectivity graph (representing the actual measurements) as inner nodes. An example of an ACG is shown in Fig. 1(b). For notational convenience, denote a generic node in the ACG as $\bar{n}_i,\ i = 1, ..., |V_A|$. We order these nodes as follows:

$$(\bar{n}_1, ..., \bar{n}_{|V_A|}) = (s_1, ..., s_n, n_1, ..., n_n, f). \tag{20}$$

Let $G_A = (V_A, E_A)$ be the ACG of the problem. The parameter vector $\mathbf{v}_A$ is now defined to comprise of the following transition probabilities

$$\mathbf{v}_A = \left\{ p_{ij}^A,\ (i,j) \in E_A \right\},$$

where p_{ij}^A is the probability that node $\bar{n}_j$ follows node $\bar{n}_i$ in a path that goes through $\bar{n}_i$ in G_A. We set

$$p_{ij}^A = \begin{cases} p_{ij}, & \text{if } \bar{n}_i = n_i,\ \bar{n}_j = n_j \\ p_b(j), & \text{if } \bar{n}_i = s_j,\ \bar{n}_j \neq f \\ p_f(i), & \text{if } \bar{n}_i = n_i,\ \bar{n}_j = f \end{cases}, \tag{21}$$

where $\{p_{ij}, p_b(j), p_f(i)\}$ were defined in subsection IV-B. In addition, define $p_{ff} = 1$ (as before), and

$$p^A_{ij} = 1 - p_b(k) \;\text{ if }\; \bar{n}_i = s_k,\, \bar{n}_j = f,$$

which simply complement to 1 the outgoing probabilities from nodes $\{s_i\}$.

The augmented parameters p^A_{ij} define a stochastic matrix P_A, which has the following form (under the assumed ordering of the nodes in (20))

$$P_A = \begin{pmatrix} \mathbf{0} & B \\ \mathbf{0} & P \end{pmatrix}, \tag{22}$$

where P is defined in (17), and

$$B = \begin{pmatrix} p_b(1) & 0 & \cdots & 0 & 1-p_b(1) \\ 0 & p_b(2) & \cdots & 0 & 1-p_b(2) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & p_b(n) & 1-p_b(n) \end{pmatrix}.$$

The two $\mathbf{0}$ matrices have n columns that correspond to the start nodes $(s_1, \ldots, s_n)$.

Note that P_A may be interpreted as a one-step transition matrix of a directed random walk on the augmented graph G_A.

D. Sampling of Candidate Solutions

We now describe how to sample a partition ω from the distribution defined through the augmented connectivity graph G_A and the corresponding transition matrix P_A. Our goal is to sample efficiently, in the sense that the number of infeasible partitions that need to be rejected should be minimized. We describe the sampling procedure in three stages. First, we describe sampling a single path in G_A. We then proceed with sampling multiple *non-intersecting* paths. We finally describe how to handle paths that fall below the minimal required length.

1) Sampling a Single Path: Sampling a single random path from the graph G_A is performed by picking uniformly at random (u.a.r.) i_1 from $\{1, ..., n\}$ and generating a random walk on G_A starting at s_{i_1} according to the transition matrix P_A until hitting f. Note that since G_A is a directed acyclic graph – after at most T steps the sink node f will be reached and the process will terminate. The resulting path is $\tau = \{s_{i_1}, n_{i_1}, ..., n_{i_j}, f, j \geq 1\}$, or $\{s_{i_1}, f\}$. In the latter case we declare the path as void. Otherwise, the path represents a track through the nodes $\{n_{i_1}, ..., n_{i_j}\}$.

2) Sampling Non-intersecting Paths: To sample a partition that avoids intersecting paths, we generate n potential paths as described above, starting from the nodes $\{s_1, \ldots, s_n\}$ in random order. To avoid track intersection, we invoke an elimination principle similar to the one used in [11] in relation with the Travelling Salesperson Problem. After each single path is sampled, we mark each of its nodes (save for the sink node f) as *occupied*, and eliminate all incoming edges to these nodes, namely nullify the probabilities on these edges. We then re-normalize the transition probabilities of the remaining outgoing edges to 1. This elimination step is equivalent to eliminating the corresponding nodes from the augmented transition matrix P_A in (22), and re-normalizing each row sum to 1. After all n paths have been generated, the remaining internal nodes (that is, those measurements that have not been selected as part of any path) are allocated to the set of false alarms τ_0.

3) Dealing with Short Paths: The partition $\omega = \{\tau_0, \ldots, \tau_K\}$ obtained in the previous stage may still be non-feasible, as some of the tracks $\tau_1, \ldots, \tau_K$ may be too short (recall that we require each feasible track to contain some minimal number of measurements). Rather than rejecting these partitions, which could still lead to a high rejection rate, we redefine measurements contained in short tracks as false alarms, and append them to τ_0. We note that this step entails some modification to our definition of the probability distribution in (19); however the precise expression is not required as part of the algorithm.

Upon completion of these steps we are left with a feasible partition ω. The sampling procedure is formally summarized in Algorithm 3.

Algorithm 3 Basic Sampling Procedure.

Input: The augmented connectivity graph G_A and transition matrix P_A defined in (22).

1: **Initialize:** Set $S = \{s_1, ..., s_n\}$ (remaining start nodes), $k = 1$ (path counter), $P = P_A$.

2: **Start k-th path:** Let $j = 1$ (node counter). Pick s uniformly at random from S. Set $S = S \backslash \{s\}$. Define $v_{k,1} = s$ to be the first node of the k-th path. Set $u = v_{k,1}$ (current node indicator). Set $\tau_k = \{u\}$.

3: **Modify transition matrix:** Eliminate node u from P by setting the u-th column of P to 0 and normalizing the rows to sum up to 1.

4: **Sample next node:** Generate $v_{k,j+1}$ from the distribution formed by the u-th row of P. If $v_{k,j+1} = f$ go to 5. Otherwise set $u = v_{k,j+1}$, $\tau_k = \tau_k \cup u$, $j = j + 1$ and reiterate from 3.

5: **Proceed to next path:** If $S = \emptyset$ go to 6. Otherwise, set $k = k + 1$ and repeat from step 2.

6: **Generate the FA set:** Set τ_0 to the set of all inner nodes in $\{\tau_1, ..., \tau_k\}$. Further, remove all single-node paths from $\{\tau_1, ..., \tau_k\}$, and append them to τ_0. Let K be the number of remaining paths.

Output: $\omega = \{\tau_0, \tau_1, ..., \tau_K\}$.

E. Parameter Update

In order to obtain the new parameter vector $\mathbf{v}_A$ that will define the updated probability distribution, we estimate its components using the best samples that were obtained in the previous iteration. Recall that our goal is maximization of the posterior probability (5). Parameter update is performed by taking the elite samples, according to the cost function (5), and calculating the Maximum Likelihood estimate of $\mathbf{v}_A$. As is well known (e.g. [11, p. 139]), the MLE for each parameter p_{ij}^A is given by

$$\hat{p}_{ij}^A = \frac{N_{ij}}{N_i},$$

where N_{ij} is the number of times the edge (i,j) was used in the elite sample and N_i is the number of times node n_i was visited in that sample. The PME updating is performed in a similar manner as explained in section III. In addition, in our experiments we have used the smoothed update (equation (8)) as described in section III.

F. Basic Initialization Scheme

To invoke the CE-based algorithm one should introduce an initial probability distribution on the set of solutions, parameterized by the vector $\mathbf{v}_0$. In typical applications such as TSP or MaxCut, where no prior information is available, the vector $\mathbf{v}_0$ is often chosen to induce a uniform distribution on the solution space [11]. Our first initialization scheme takes a similar approach, with some required modifications related to track initialization:

1) For internal nodes n_i, $i = 1, \ldots, n$, we assign p_z, the termination probability of a track, to the edge connecting n_i with f and equal probabilities to all outgoing edges from n_i. In case the only outgoing edge is the one leading to f, it is assigned a probability 1.
2) For start nodes s_i, $i = 1, \ldots, n$, we set $p_b(i)$ (the probability that node s_i starts a path) to p_b, a chosen initial probability which is equal for all nodes. The parameter p_b may be empirically tuned, and is related to the known birth rates of the new targets λ_b. The choice of $p_b = 0.3$ was found to work well in our initial experiments.

This simple initialization method works well in small problems, but is inefficient in large instances. A more effective method, which utilizes the dynamical nature of the problem is described in subsection V-C.

V. Improvements to the Basic Algorithm

The algorithm described in section IV provides a basic version of cross-entropy based data association. However, the performance of this algorithm may be greatly enhanced by adding certain key improvements to the sampling scheme, that make use of the specific problem characteristics. In the next three

subsections we discuss (a) history-based sampling that takes account of the target's dynamic model, (b) bidirectional sampling that helps in sampling complete trajectories rather than their tails, (c) an improved initializations scheme for the sampling distribution, and (d) elimination technique for unlikely tracks. These modifications define our final algorithm as used in the simulation experiments. We end this section by discussing the computational complexity of the modified algorithm.

A. History-Based Sampling

Recall that our basic sampling scheme is based the connectivity graph where each inner node corresponds to a single measurement, and the next node in a path is sampled according to a probability distribution that *depends only on the present node*. Assume that measurements contain position estimates only (but not velocity estimates), as is often the case. This essentially implies that the choice of next node may depend only on the current target position, but not on other motion parameters such as the target velocity (direction and speed), which may be estimated from the path sampled so far. To the contrary, the dynamic target model assumed in this work (see equations (1)-(2)) implies that other components of the state vector x_t should be useful in estimating the next target position. These components typically include the target velocity (direction and speed), and possibly higher position derivatives. For concreteness, let us focus the discussion on target velocity.

To integrate target velocity into our sampling scheme, two options may be considered. One is to add a (discretized) velocity variable to each node, which is estimated based on the sampled path so far, and allow the next-node sampling distribution to depend on this variable as well. A simpler option, which we pursue here, is to allow the next-node sampling distribution to depend on the recent two (or more) nodes in the current path, rather than on the last node alone. Clearly, the position data in two sequential nodes encodes the target velocity, position data from three nodes can encode the target acceleration, etc.

Thus, we modify the parametric distribution that underlies the CE and PME algorithms in the following way. Instead of sampling the next node in each path based on the current node alone, we allow the sampling probability to depend on the last r recent nodes (with $r = 2$ used in our simulations). To keep our previous algorithms, it is useful to note that the above modification can be easily translated to an expansion of the connectivity graph, where each possible sequence of r nodes in the original graph now forms a node in the new graph. The actual sampling, initialization and updating schemes described in sections IV-D, IV-F, and IV-E remain the same accept that they are applied to this extended graph.

Clearly, the proposed extension increases the number of parameters to be estimated by the CE and Parametric MinxEnt methods. For $r = 2$, the increase is proportional to the average out-degree of the basic

connectivity graph representing the problem. While the number of parameters increases proportionally, the performance improvement is dramatic.

The preceding extension may also be motivated by considering the sequential sampling schemes used in CE based algorithms as a useful approximation to the full probability distribution. The following discussion may be related to some recent ideas found in [17]. Consider for example the solution for the MinxEnt program (12), which specifies some probability density $f^*(\mathbf{x})$ over the components of $\mathbf{x} = (x_1, x_2, ..., x_n)$. In general, one may write $f^*(\mathbf{x})$ in sequential form:

$$f^*(\mathbf{x}) = f_n(x_n|x_1^{n-1}) \cdots f_2(x_2|x_1) \cdot f_1(x_1), \tag{23}$$

where we have used the shorthand notation $x_i^j = (x_i, x_{i+1}, ..., x_j)$. The idea behind CE and Parametric MinxEnt is to approximate each element in this product by a parameterized distribution, with simplified dependence on past component. The simplest approximation assumes that the components of $\mathbf{x}$ are independent. The next approximation, which is commonly used for path-sampling ([11], [18]), employs the (first-order) Markov chain structure:

$$f^*(\mathbf{x}) \approx h_n(x_n|x_{n-1}) \cdot h_{n-1}(x_{n-1}|x_{n-2}) \cdots h_2(x_2|x_1) \cdot h_1(x_1), \tag{24}$$

where the ith component depends on the corresponding parameter v_i. This is the approximation used in our basic scheme, where the components x_i correspond to the nodes in the connectivity graph, and transitions on that graph are sampled via conditional probabilities of the form $p_{v_i}(x_{i+1}|x_i)$. Alternatively, we may use a less crude approximation by allowing the distribution of the next node in the path to depend on two (or more) previous nodes (second-order Markov chain structure), as follows

$$f^*(\mathbf{x}) \approx h_n(x_n|x_{n-1}, x_{n-2}) \cdot h_{n-1}(x_{n-1}|x_{n-2}, x_{n-3}) \cdots h_3(x_3|x_2, x_1) \cdot h_2(x_2|x_1) \cdot h_1(x_1).$$

This representation translates to the two-node based sampling scheme as proposed above.

B. Bidirectional Path Sampling

Recall that we sample paths by picking at random an s_i node and generating a random walk forward in time, starting at s_i and ending at f. If the node s_i (or n_i) happen to fall in the middle of an actual path, we will clearly miss the first part of this path. Thus, most of the paths sampled in such manner will be "truncated" since only their tails will be sampled. In other words, in order to obtain good solutions with tracks beginning at the actual appearance times of the targets, we might need sufficiently many samples and many iterations of the algorithms. This limitation may be efficiently resolved by allowing *bidirectional* sampling from the starting node s_i. That is, in additional to sampling a forward path in the

usual manner, we also sample *backward* in time from s_i. The two sampled halves are joined to form one complete path.

Sampling backward paths is carried out using a separate structure, which is composed of a backward connectivity graph with suitable sampling parameters. The backward connectivity graph is simply obtained from the connectivity graph by reversing the direction of the graph edges. We augment this graph with a new final node f' (see Fig. 2). However, the start nodes nodes (s_i) are shared with the forward connectivity graph. The relevant sampling parameters for the reverse graphs are as usual the transition probabilities between adjacent edges.

Sampling a complete path now starts, as before, by picking at random a start node s. From that node, a forward path is sampled over the forward connectivity graph, and a backward path is sampled using the reverse graph. Finally, the two halves are concatenated at s to create the complete path. The rest of the algorithm proceeds essentially as before: Paths are valuated according to the cost function (5), and the parameters are updated independently for the forward and backward graphs.

The improved initialization scheme of the next subsection uses the dynamical model of the target state evolution. For the reverse connectivity graph, we use the following "backward" version of the dynamical model (1), namely

$$x_{t_i} = A^{-1}(t_{i+1}, t_i)x_{t_{i+1}} - A^{-1}(t_{i+1}, t_i)G(t_{i+1}, t_i)w_{t_i}. \tag{25}$$

Here we assume that the dynamics matrix $A(t_{i+1}, t_i)$ is invertible (which valid in our simulation models, in particular).

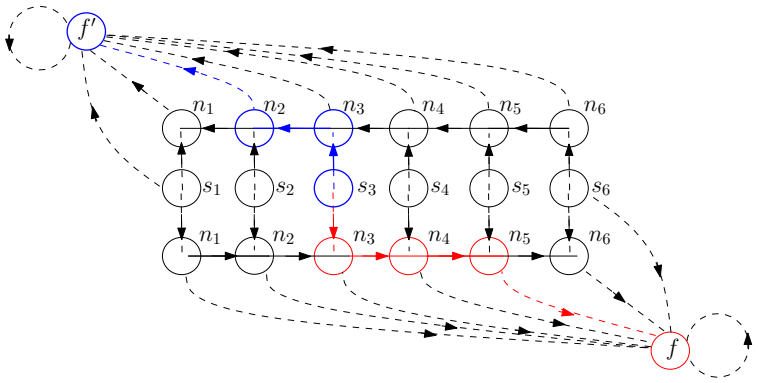
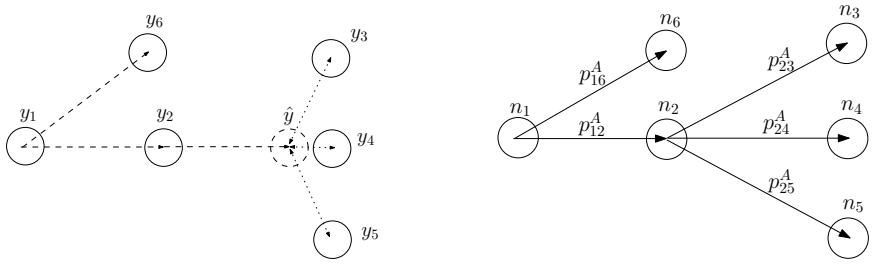


Figure 2. An example of the bi-directional sampling scheme. $s_1, s_2, ..., s_6$ are shared by the original and the backward connectivity graphs. The sampled forward path is $s_3 \to n_3 \to n_4 \to n_5 \to f$. The sampled backward path is $s_3 \to n_3 \to n_2 \to f'$. The resulting path is comprised of the nodes $n_2 \to n_3 \to n_4 \to n_5$.



(a) The actual measurements (b) The corresponding graph nodes

Figure 3. Example of the initialization scheme. (a) Solid circles represent measurements, dashed circle represents predicted measurement. (b) The corresponding part of the connectivity graph is shown.

C. Likelihood-Based Initialization

We next present an improvement on the basic initialization scheme introduced in subsection IV-F. Recall that a uniform initial distribution may be used when no prior information is available. In our case, however, prior information on possible target paths is embedded in the dynamical and measurement models (1)-(2).

For example, if all targets are assumed to obey the (nearly) constant velocity model [19] and the measurement model is accurate, high probability should be assigned to edges that connect nodes representing co-linear measurements. Initialization of the nodes $\{s_1, ..., s_n\}$ that was described in subsection IV-F remains unaltered, and the proposed modification refers to the inner nodes of the graph – $n_i,\ i = 1, ..., n$, where n is the number of the inner nodes representing the actual measurements. We describe the initialization procedure for these nodes with respect to the history-based sampling described in subsection V-A.

To initialize the nodes representing the actual measurements we apply a Kalman Filter to every three (or possibly more – depending on the target model) neighboring nodes of G_A that represent three consecutive measurements. The filter is initialized with the first two measurements using the two-point differencing technique [19]. Namely, we initialize the position as that of the first measurement and the velocity as the difference between the two measurements divided by the difference in their time tags. We can now use the Kalman filter equations to obtain the expected position and covariance of the third measurement. The corresponding probability p_{ij}^A of each edge of the graph is set proportional to the likelihood function of the corresponding measurements relative to that prediction.

To illustrate, consider the situation depicted in Fig. 3. Assume that we have initialized a KF using y_1 and y_2 that were obtained at two consecutive sampling times, and $\hat{y}$ is the resulting predicted measurement to the next sampling time. Assume that the neighbors of n_2 (representing the measurement y_2) in the

connectivity graph are n_3, n_4 and n_5 (representing the measurements y_3, y_4, and y_5 respectively), that carry the same time tag as $\hat{y}$. The probabilities on the edges connecting n_2 to n_3, n_4, and n_5, are p_{23}^A, p_{24}^A, and p_{25}^A. These are determined on the basis of the distance between $\hat{y}$ and y_3, y_4, y_5 respectively, using the assumption that the true measurement deviates from the predicted one according to a Gaussian distribution. Mathematically, this reads,

$$y_i \sim \mathcal{N}(\hat{y}, B),\ i = 3, 4, 5 \tag{26}$$
$$p_{2i}^A \propto \mathcal{N}(y_i - \hat{y}, B),\ \sum p_{2i}^A = 1,$$

where B is the innovation covariance obtained for $\hat{y}$. Clearly, some edges cannot be initialized in such manner, e.g., the edges $n_1 \to n_2$ and $n_1 \to n_6$ in the above example since n_1 has no possible predecessor. We assign those a uniform distribution. In our example $p_{12}^A = p_{16}^A = 0.5$.

As in the basic initialization scheme, the probabilities on the edges linking the inner nodes to f are initialized to the termination probability p_z, which was introduced in section II. The probabilities on the other edges calculated via the above procedure are then re-normalized to sum to 1. In case the only outgoing edge is the one leading to f, it is assigned a probability 1.

D. Elimination of Unlikely Tracks

The basic procedure of section IV together with the improvements described above produce a feasible solution to the multi-scan data association problem having high posterior probability. However, it is possible that among the actual tracks found by the algorithm, several false tracks, or ghosts, will be generated. These tracks are patterns of consecutive false measurements that constitute feasible tracks according to the definition of subsection II-D.

To reduce the number of such false tracks we apply additional likelihood-based test as follows. For each track τ in the final partition ω we compute its likelihood $\ell(\tau)$, and also $\ell_0(\tau)$ which is the likelihood of all measurements in τ classified as false alarms. If $\ell(\tau)$ exceeds $\ell_0(\tau)$ we exclude τ from the tracks $\{\tau_1, ..., \tau_K\}$ of ω and add its measurements to the set of false alarms of $\omega - \tau_0$. This procedure may only increase the posterior probability of the partition ω, thus resulting in a better score of the final solution.

E. Computation Complexity

We now evaluate briefly the running time of the proposed algorithms. The total running time is determined by the number of iterations (sampling-evaluation-updating cycles) until convergence – n_{CE}. Each iteration is determined by the number of sampled candidate solutions – N, the time required to generate each solution, the time required to evaluate the cost of solution, and the time to update the

parameters for the following CE or PME iteration. Recall that n is the number of inner graph nodes representing the actual measurements of the problem, and T is the duration of the surveillance scenario, namely the number of scans to be assigned (see subsection II-A). In the following we summarize the time complexity (in terms of the number of elementary operations) of various procedures. First, we provide the computational requirements of the basic algorithm of section IV and then proceed with the improved scheme of section V.

1) Computational Complexity of the Basic Algorithm:

- The complexity of sampling a candidate solution is $\mathcal{O}(Tn^2)$.
- The complexity of evaluation of the cost of a solution is $\mathcal{O}(nT^2)$.
- The complexity of updating the distribution is $\mathcal{O}(Nn^2)$.
- The complexity of calculation of the initial distribution is $\mathcal{O}(n^2)$.

The overall procedure time requirements are summarized as $\mathcal{O}(n_{CE}(NTn^2 + NnT^2 + Nn^2) + n^2)$.

2) Computational Complexity of the Improved Algorithm: The improved scheme differs from the basic one by an increased number of graph nodes and an increased number of parameters to be estimated. The increase is proportional to the average out-degree of the basic connectivity graph. Denoting this quantity by η, we obtain the following worst case complexity.

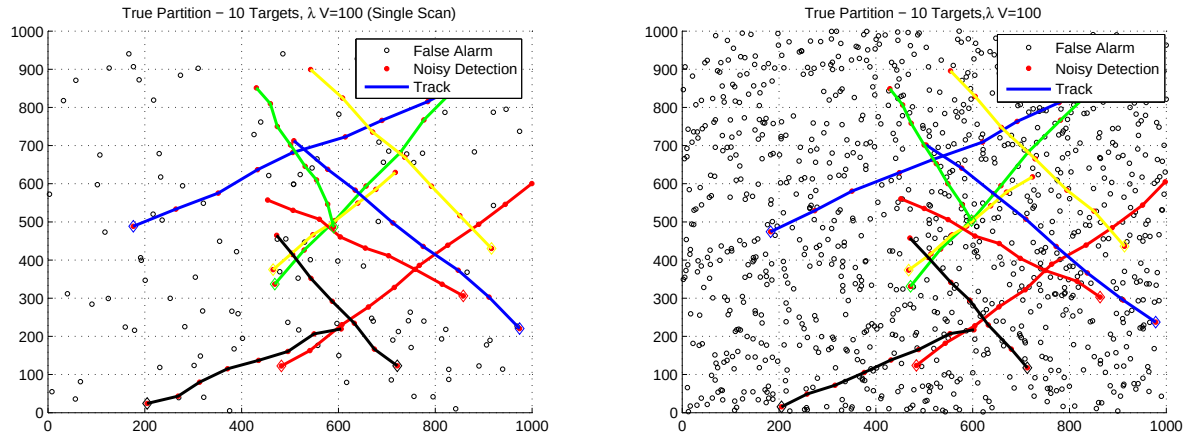
- The complexity of sampling a candidate solution is $\mathcal{O}(Tn^2)$.
- The complexity of evaluation of the cost of a solution is $\mathcal{O}(nT^2)$.
- The complexity of updating the distribution is $\mathcal{O}(N\eta n^2)$.
- The complexity of calculation of the initial distribution is $\mathcal{O}(\eta n^2)$.

Note that the time required for sampling a single candidate solution and evaluating its cost remain unchanged since the path maximal length is not affected by the improved sampling scheme, nor is the maximal number of the paths in the augmented connectivity graph. The complexity of updating the distribution at each iteration, however, increases due to the increased number of parameters and nodes.

The overall procedure time requirements are summarized as

$$\mathcal{O}(n_{CE}(NTn^2 + NnT^2 + N\eta n^2) + \eta n^2). \tag{27}$$

For dense scenarios, where $\eta = \mathcal{O}(n)$ and $T \ll n$ the above time complexity becomes $\mathcal{O}(n_{CE}Nn^3)$. Although n_{CE}, the number of iterations of CE-based algorithms, cannot be bounded in advance, in all our experiments convergence has been obtained within 7 to 10 iterations. A similarly low number of iteration till convergence seems to be typical of other CE applications as well [11].



(a) A single scan

(b) All scans

Figure 4. An example of a typical scenario. Target detections and FAs from all 10 scans are shown (a), FAs from a single scan are shown (b).

VI. Simulation Results

A. General Simulation Setup

We consider a similar simulation setup to that in [10]. A rectangular region on a plane, $\mathcal{R} = [0, 1000] \times [0, 1000] \subset \mathbb{R}^2$ is taken to be the surveillance region (to be specific, length units can be taken as meter and time units as seconds). Each target has a 4-component state vector with position and velocity in the x and y directions, that is $x_t = [p_{xt}, v_{xt}, p_{yt}, v_{yt}]^T$. We have used the Discrete White Noise Acceleration (DWNA) model [19] for the targets dynamics. Namely,

$$x_{t+1} = Ax_t + Gw_t,$$

where

$$A = \mathrm{diag}\left[F_1, F_1\right], \; F_1 = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix}.$$

In addition, the vector process noise is $w_t = [w_{xt} \; w_{yt}]^T$ with covariance $Q = \mathrm{diag}(\sigma_w^2, \sigma_w^2)$, and

$$\mathrm{cov}(Gw_t) = \sigma_w^2 \cdot \mathrm{diag}(Q_1, Q_1), \; Q_1 = \begin{bmatrix} \frac{1}{4}T_s^4 & \frac{1}{2}T_s^3 \\ \frac{1}{2}T_s^3 & T_s^2 \end{bmatrix}.$$

The measurement equation is

$$y_t = \begin{bmatrix} p_{xt} \\ p_{yt} \end{bmatrix} + \begin{bmatrix} v_{xt} \\ v_{yt} \end{bmatrix} = Cx_t + v_t,$$

where

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix},$$

and the (vector) measurement noise is $v_t = [v_{xt} \ v_{yt}]^T$ with covariance matrix $R = \text{diag}(\sigma_v^2, \sigma_v^2)$. We have used a surveillance duration of $T = 10$ scans. Targets appear at a uniformly chosen position from the left bottom or right bottom quadrants of $\mathcal{R}$ respectively. They all move diagonally (in straight lines) with constant velocity randomly chosen between $0.2v_{\max}$ and $0.9v_{\max}$. Each target's appearance and disappearance times are chosen uniformly from the first and last quarters of the surveillance interval respectively. An example of a typical scenario is shown in Fig. 4.

B. Performance Measures

To the best of our knowledge, a standard definition of performance measures for evaluation of MTT algorithms is an open question. Performance evaluation of single-target tracking algorithms (both with and without measurement origin uncertainty) may be quantified by means of the Mean Square Error (MSE). Such criteria are problematic when dealing with association algorithms, since the MSE may provide meaningful insight on the performance only provided the data association is perfect. We thus adopt the following, rather intuitive, measures for performance evaluation suggested in [10].

1) The normalized correct associations (NCA), that is, the number of correct associations made by the algorithm divided by the true number of associations.
2) The incorrect-to-correct association ratio (ICAR) which measures the ratio of incorrect to correct associations.

An *association* here means two consecutive measurements on the same track. Mathematically, for each partition $\omega \in \Omega$, the set of all associations in ω is represented as

$$\text{SA}(\omega) = \{(\tau, t_i^\tau, t_{i+1}^\tau) : i = 1, \ldots, |\tau| - 1, \tau \in \omega\}$$

where t_i^τ is the time of the i-th observation in track τ. The set of *correct* associations in ω relative to ω^*, which is the true partition, is

$$\text{CA}(\omega) = \{(\tau, t, s) \in \text{SA}(\omega) : \tau(t) = \tau^*(t), \tau(s) = \tau^*(s), \text{ for some } \tau^* \in \omega^*\}, \tag{28}$$

where $\tau(t)$ is the measurement at time t associated with track τ. The above measures now read:

$$\text{NCA}(\omega) = \frac{|\text{CA}(\omega)|}{|\text{SA}(\omega^*)|}, \quad \text{ICAR}(\omega) = \frac{|\text{SA}(\omega)| - |\text{CA}(\omega)|}{|\text{CA}(\omega)|}, \tag{29}$$

where $|A|$ is the cardinality of the set A. From the definition, NCA varies between 0 and 1 and it provides a measure of the number of correct associations. In case of perfect associations, NCA $= 1$. NCA does not account, however, for false tracks. On the other hand, ICAR is a positive measure not bounded from above and, informally, it counts false associations – both false tracks and false continuations of true tracks. When no incorrect associations are made, ICAR $= 0$. These performance measures are illustrated in Fig. 5. In addition, we record the number of tracks estimated by the algorithms. For the CEDA and

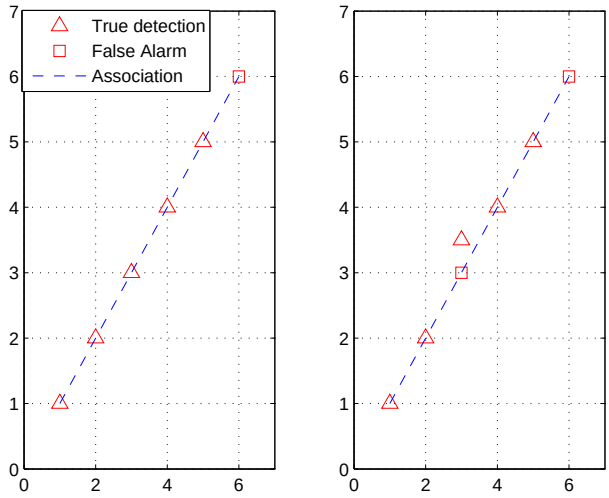


Figure 5. An example of performance measures. NCA $= 1$, ICAR $= \frac{1}{4}$ (left), NCA $= \frac{1}{2}$, ICAR $= \frac{3}{2}$ (right)

PMEDA algorithms we split this number into true and false tracks.

C. Compared Algorithms

The performance of the CE based algorithms was compared to the following data association schemes. First, is the Multiple Hypotheses Tracker (MHT) for which results have been reported in [10]. The authors have used the implementation from [20], [21], which implements pruning, gating, clustering, N-scan-back logic and k-best hypotheses (see [20] for details on these techniques). Second, is the greedy tracker proposed in [10]. It is a batch-mode nearest neighbor multiple-target tracking algorithm. It generates candidate tracks by picking measurements nearest to the predicted states until there are no unused measurements left. Finally is the Markov Chain Monte Carlo Data Association (MCMCDA) also proposed in [10]. The MCMCDA algorithm allows sampling from complex distribution by simulating an ergodic Markov chain which converges to the desired limiting distribution. In [10] the authors constructed

a Markov chain on the set of all feasible partitions of the measurements (see subsection II-D) with the posterior (5) as the limiting distribution. Using a small set of elementary moves (e.g. track birth, track termination, track splitting etc.) sampling from the posterior was possible after a few thousands of moves. Then, a sample with the highest posterior was returned as the solution. The initial state of the Markov chain was provided by the output of the Greedy algorithm. For the performance comparison we have reconstructed the results for the MCMCDA and the greedy algorithms and used the results reported in [10] for the MHT.

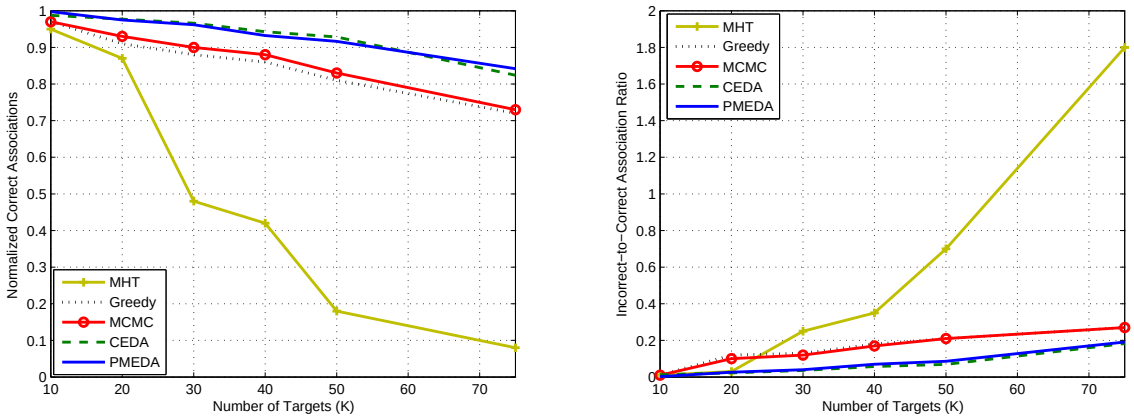
D. Experiments and Results

In this section we test the performance of the proposed algorithms by comparing them with the performance of the methods mentioned in the previous subsection. The algorithms are compared to each other using the performance measures NCA, ICAR and the estimated number of targets described above.

Several options exist to challenge a data association algorithm. The first is by increasing the false alarm rate $\lambda_f V$. In addition, one can decrease the detection probability P_d and increase the density of tracks K. Closely moving targets, low detection probabilities and high false alarm rate make the problem more difficult. In the following we perform three different test sequences in which we modify different parameters to evaluate the performance of the algorithms – the false alarm rate λ_f, the density of tracks K, and the detection probability P_d. Additional simulation tests including the influence of the process and measurement noises, the segment length and robustness to the target model are available in [22]. In all simulations presented below the results are averaged over 8 repeated runs with new measurements generated independently at each run.

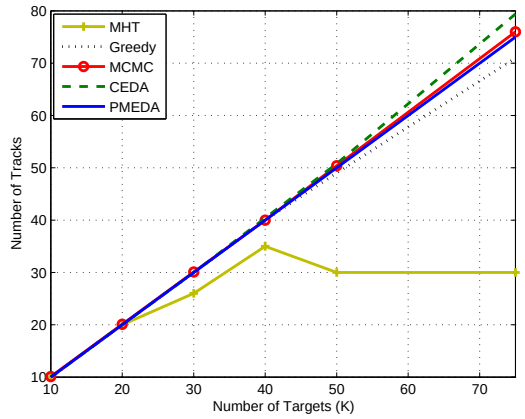
1) Number of Targets: In this sequence we modify the track density. The number of targets K in the scenario is varied between 10 and 75, while keeping the clutter rate low and the detection probability high. Each target moves at constant velocity uniformly chosen between 30 and 120 m/s. All other parameters are fixed as well, namely – $\lambda_f V = 1$, $P_d = 0.999$ and $v_{\max} = 140$ m/s, $p_z = 10^{-2}$, and $\lambda_b V = 1$ such that, on average, there is a single new target at each scan. The average NCAs, ICARs and number of tracks are presented in Fig. 6. It is readily seen that both cross-entropy based algorithms score higher than Greedy and MCMCDA in NCA and better in ICAR, introducing no significant difference between CEDA and PMEDA. Note that MCMCDA improves only slightly the performance achieved by the greedy algorithm. All algorithms outperform the MHT.

2) False Alarms Rate: Here we keep a constant number of $K = 10$ tracks, with targets moving at constant velocity each uniformly chosen between 30 and 120 m/s. The clutter rate varies between

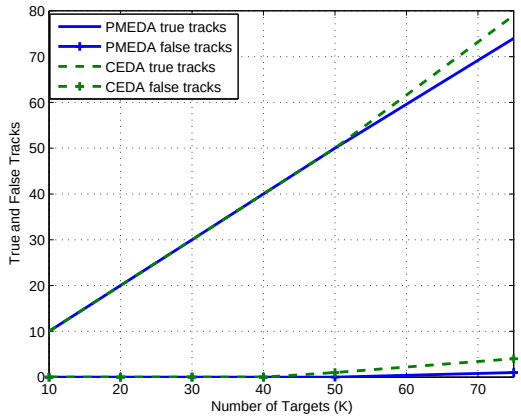


(a) NCA

(b) ICAR



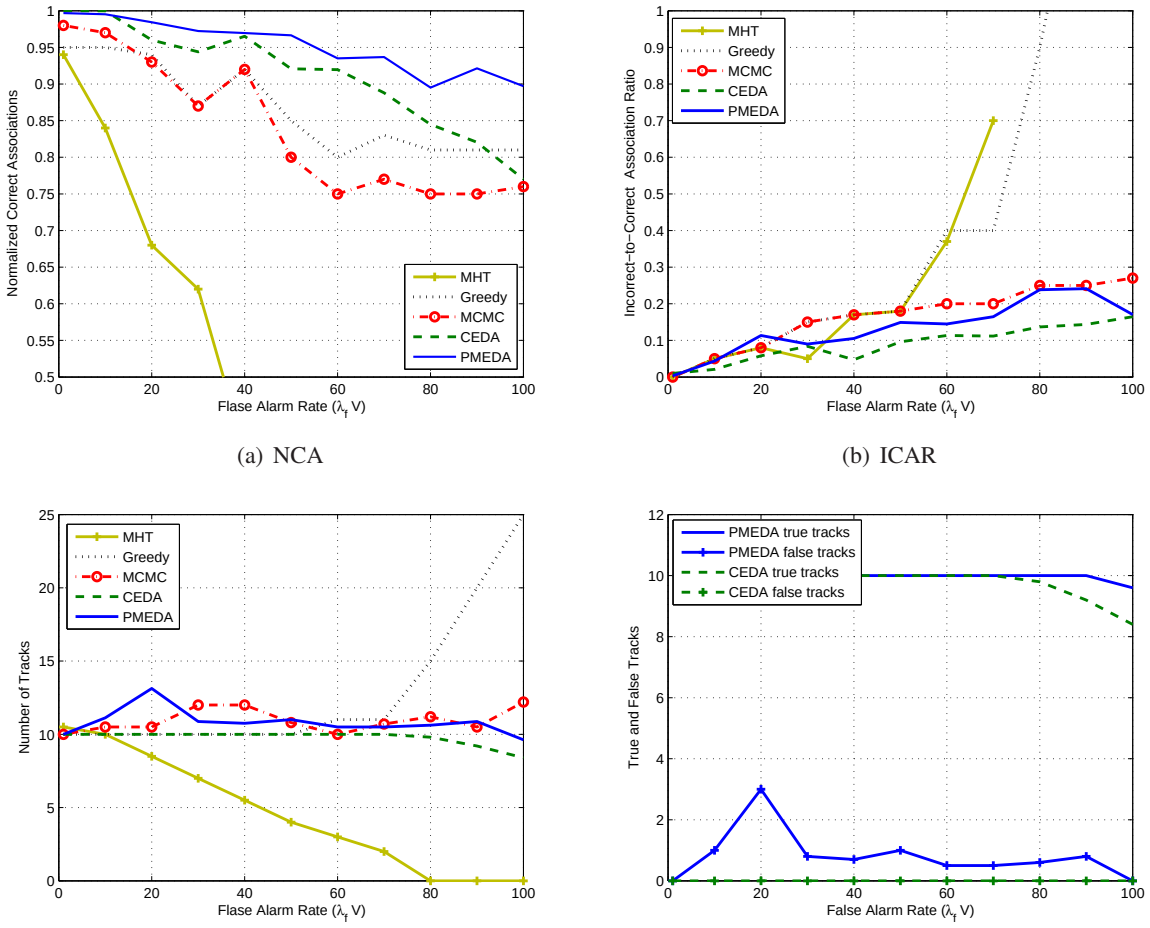
(c) Number of Tracks Detected



(d) Number of True and False Tracks

Figure 6. Simulation results for various values of K, the number of targets.

$\lambda_f V = 1$ and $\lambda_f V = 100$. $P_d = 0.999$, $v_{\max} = 140$ m/s, $p_z = 10^{-2}$, and $\lambda_b V = 1$. Namely, in this experiment we test our algorithms in heavily cluttered environment, keeping the detection probability high and the number of targets low. The results are depicted in Fig. 7. Clear superiority of PMEDA may be noticed with nearly 90% of correct associations. The CEDA algorithm behaves similarly at low and moderate clutter rates, but degrades in NCA performance at high clutter rates. However, false tracks are not produced keeping the ICAR relatively low. We may conclude that, in this application, the updating rule of the PMEDA algorithm, which uses all the samples obtained at a given iteration rather than the elite samples used by CEDA, is preferable. As reported in [10], the MHT algorithm does not make any associations when $\lambda_f V \geq 80$ resulting in zero NCA and unreported ICAR. This is due to various heuristics used in the specific implementation of the MHT which are necessary for finite running times.



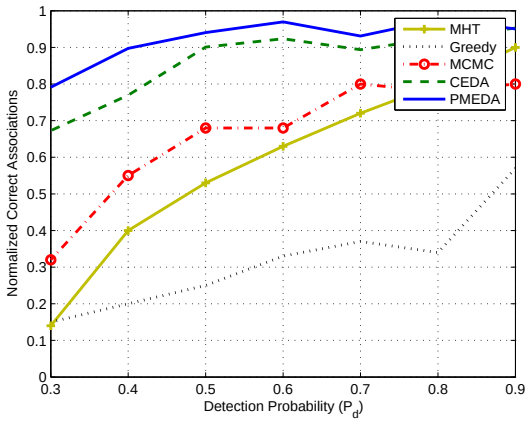
(a) NCA (b) ICAR

(c) Number of Tracks Detected (d) Number of True and False Tracks

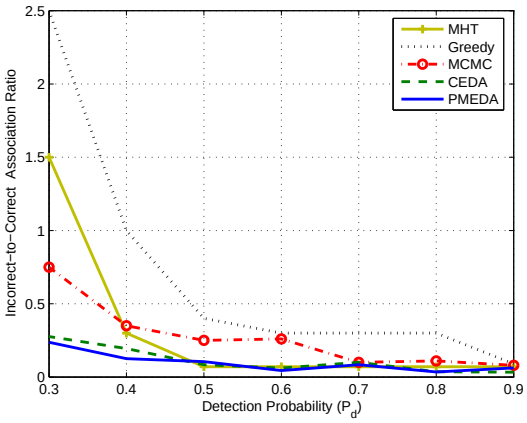
Figure 7. Simulation results for various values of $\lambda_f V$. The actual number of targets is 10.

Additional discussion of the behavior of the MHT in the discussed experiments is provided below. The MCMCDA algorithm being initialized with the output of the greedy algorithm, removes many of the false tracks found at the initialization stage, thus improving the ICAR, but at the same time it degrades the NCA performance by changing some of the correct associations. As a result, the obtained NCA is lower than that of the greedy algorithm by a few percent. The greedy algorithm achieves reasonable performance in terms of the NCA with more than 80% of correct associations, but results in unacceptably large ICAR due to generation of high number of false tracks which also increase the ICAR.

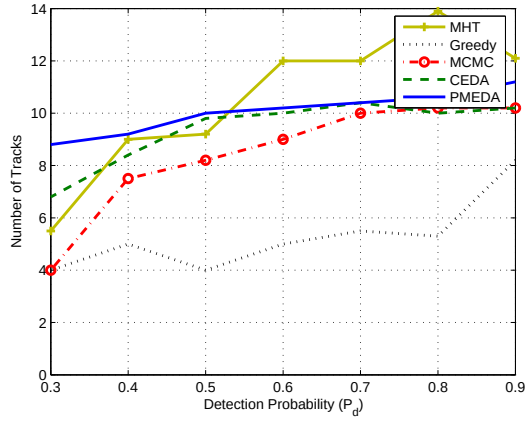
3) Detection Probability: There is a constant number of $K = 10$ tracks, which move at constant velocity each uniformly chosen between 30 and 120 m/s. The clutter rate is kept constant at $\lambda_f V = 1$. The probability of detection varies between $P_d = 0.3$ to $P_d = 0.9$. Due to lower detection probabilities we



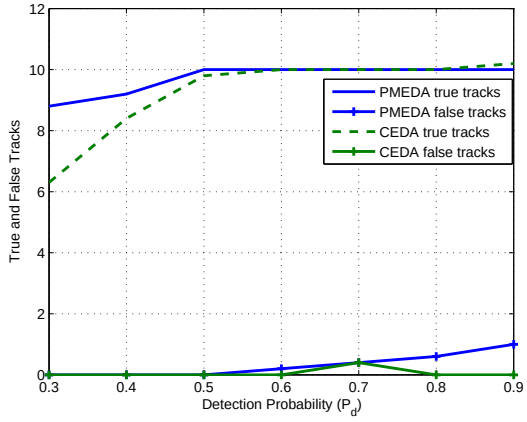
(a) NCA



(b) ICAR



(c) Number of Tracks Detected



(d) Number of True and False Tracks

Figure 8. Simulation results for various values of P_d.

have set $d_{\max} = 5$ since many consecutive missed detections may occur (recall that $d_{\max}$ is the maximum number of consecutive missing observations of any track that was defined in section II). The probability of track termination is $p_z = 0.01$ and the appearance of new tracks is modeled by $\lambda_b V = 1$. The results are depicted in Fig. 8. Both cross entropy based algorithms outperform all other algorithms with some superiority of the PMEDA over CEDA. Unlike the previous tests, the greedy algorithm performs poorly with less than 50% of correct associations. MCMCDA scores better than MHT, but much worse than cross entropy based algorithms especially at very low detection probabilities.

E. Discussion

It it noticeable in the first and second experiments that MCMCDA does not introduce significant improvement in terms of the NCA in comparison to the greedy tracker with which it has been initialized.

This occurs when the greedy algorithm finds a solution that is a strong local maximum of the problem, and MCMC, being a local search method, needs many steps to escape from this strong local extremum. Although in theory the MCMC method is considered a global optimization method, it turns out in this experiment that at this level of problem complexity it quickly got trapped in local minima, from which it could not get out in any reasonable computation time. Thus, MCMC effectively reduces here to local search, while CE based algorithms maintain a more global flavor as witnessed by their performance.

Another interesting phenomenon is the relatively poor performance of the MHT algorithm as reported in [10]. Although in theory MHT is an optimal solution in the MAP sense it performs poorly when the detection probability is low or the false alarm rate is high due to the necessary heuristics used in the algorithm. These heuristics are required as part of all practical implementations of MHT since without them the number of hypotheses grows exponentially fast. MHT with such heuristics may work well when a few hypotheses carry most of the weight. However, when the detection probability is low or the false alarm rate is high, there are many hypotheses with low weight and there is set of dominating hypotheses, so MHT cannot perform well. This explains the poor behavior of the method when tested in the extreme cases illustrated in Fig. 7 and Fig. 8. Similarly, for high values of K the specific implementation of the MHT encounters difficulty in finding tracks, and the most of the returned ones are false. Thus, the total number of tracks in Fig. 6 becomes low and plateau for high K, with the corresponding NCA being low and ICAR being high.

F. Computation Times

All algorithms proposed in this paper were implemented in Matlab® without any code optimizations and ran on a PC with 2.8GHz Intel processor. Recall that the overall performance is determined by the time required to obtain and evaluate a single sample in the CE/PME procedures in addition to the initialization and updating procedures. For dense scenarios, however, the time to obtain a single sample, which requires $\mathcal{O}(Tn^2)$ calculations, dominates the overall computation time. We present in Fig. 9 this empirically found time versus the clutter rate which is proportional to the average number of observations in the problem at hand. It is readily seen that this empirical evidence strongly supports the calculated complexity needed to obtain a single sample.

We note that, as reported in [10], the running times of MHT (implemented in C++) are comparable to the running times of our algorithms (implemented in Matlab). The running times of the MCMCDA algorithm as reported in [10] are much shorter than ours (although a fair comparison is hard since MCMCDA was implemented in C++ as opposed to the Matlab implementation of our algorithms). However, the point in

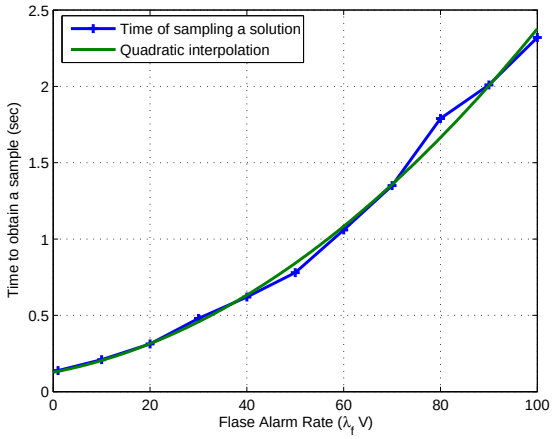


Figure 9. Time required to obtain a single sample.

our algorithms is that better performance is attained. Indeed, the performance curves of MCMCDA tend to flatten after the initial improvement period, so that further improvement should be hard to attain (this point is further discussed in Section VI-E).

VII. Sequential Multi-Scan Tracking

In typical (off-line or on-line) applications, the number of sequential measurements (or scans) is too large to be handled simultaneously by existing multi-scan data association algorithms. In that case, the overall scenario may be divided into shorter segments, of length suitable for the application of a multi-scan association algorithm. Still, it is desirable to maintain continuity of paths between adjacent segments, as well as some interchange of relevant data.

We extend our algorithm to handle long scenarios by dividing the whole surveillance duration into overlapping segments of length T_W each. In each segment we execute the basic CEDA algorithm using the observations which belong to that segment. The overlap region allows to correct associations obtained at the final scans of the previous segment using measurements from the next segment, and vice versa, to initialize the next segment with existing tracks. The idea is illustrated in Fig. 10.

In more detail, suppose the CEDA algorithm has been applied to segment k. Associations obtained in the first part of that segment (before the overlap region) are fixed and cannot be changed. To initialize the algorithm in segment $k+1$, we assign probability 1 to the edges $s_i \to y_i$ for nodes y_i that belong to the first scan of segment $k+1$ and were associated with tracks in segment k. Thus, these nodes are bound to be starting nodes of paths in the new segment. All other probabilities in that segment are initialized as in the single segment case according to the scheme described in subsection V-C.

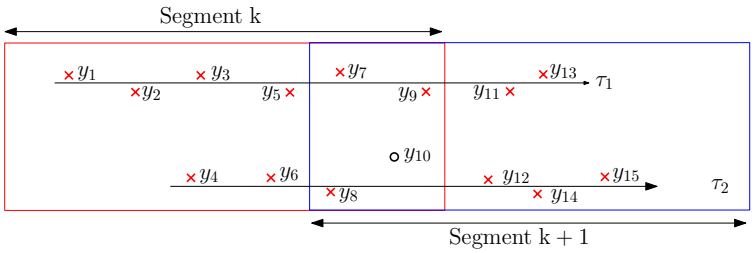


Figure 10. Sequential multi-scam data association. The algorithm is invoked on two overlapping windows. Measurements y_7 and y_9 associated with τ_1 in window $k1$ are used to initialize the track in window $k+1$. Measurements y_{12}, y_{14}, and y_{15} associated with τ_2 in window $k+1$ improve the associations obtained in window k by classifying y_{10} as a false alarm.

The extent of overlap between two adjacent segments is a design parameter which should be chosen using the following considerations. On the one hand, the overlap should be large enough, so that misassociations at the later scans of segment k can be corrected while solving in segment $k+1$. On the other hand, large overlaps increase the number of segments and hence the computation time. However, we have found, via simulation, that performance is not affected significantly when the overlapping length is increased beyond a certain point, and satisfactory results are obtained for small overlaps. The results supporting this statement are summarized in Table I for a small scenario with 20 targets, 45 scans surveillance duration, and segment length of 15 scans. Whereas the worst performance is achieved for

Overlap size	0	3	6	9
NCA	0.89	0.94	0.93	0.95

Table I
THE OVERLAP INFLUENCE ON THE PERFORMANCE

non-overlapping windows, large overlaps do not introduce significant changes. Further testing of the proposed sequential CEDA algorithm was done as follows. We have used the following parameters. The surveillance region is increased to $\mathcal{R} = [0, 10000] \times [0, 10000] \subset \mathbb{R}^2$, and the surveillance duration is increased to $T = 90$ scans. Segment size is $T_W = 15$, and the length of the overlap between adjacent windows is 4 scans. Targets appear and disappear at random times and positions and clutter rate is $\lambda_f V = 10$. $P_d = 0.9$, $d_{\max} = 3$ and maximum velocity is set to $v_{\max} = 230$ m/s. All other parameters remain unchanged. Three test cases are generated with total number of 50 tracks, 100 tracks and 150 tracks. Typical results for the overall NCA and ICAR for the above test cases are summarized in Table II. These results indicate that the proposed algorithm is indeed capable of initiating, tracking and terminating tracks for long scenarios.

Number of targets	NCA	ICAR
50	0.95	0.04
100	0.94	0.04
150	0.92	0.06

Table II
SEQUENTIAL TRACKER PERFORMANCE

VIII. CONCLUSION

We have proposed two related methods for the multi-scan multi-target data association problem based on the CE and PME heuristics. These schemes have been tested in simulation and show improved performance relative to the state-of-the-art algorithms. A major issue in the proposed algorithms is their computation time. Although polynomial in the problem parameters, the computation time is still considerable in challenging scenarios which involve a large number of measurements (ranging from minutes to hours for the more complex scenarios using unoptimized Matlab code as described in section VI). In these cases the proposed algorithms are more suitable for off-line data analysis. Future research directions should address further reduction of the computation time while keeping the advantages of the proposed approach. Furthermore, various extensions to the model should be of interest, including non-linear target dynamics and multisensor measurements.

REFERENCES

[1] Y. Bar-Shalom and T. E. Fortmann, *Tracking and Data Association*. Academic Press, San Diego, 1988.

[2] J. Collins and J. Uhlmann, "Efficient gating in data association with multivariate distributed states," *IEEE Trans. on Aerospace and Electronic Systems*, vol. 28, no. 3, pp. 909–916, July 1992.

[3] D. B. Reid, "An algorithm for tracking multiple targets," *IEEE Trans. on Automatic Control*, vol. AC-24, no. 6, pp. 843–854, December 1979.

[4] R. Streit and T. Luginbuhl, "Probabilistic multi-hypothesis tracking," 1995.

[5] C. L. Morfield, "Application of 0-1 integer programming to multitarget tracking problems," *IEEE Trans. on Automatic Control*, vol. AC-22, no. 3, pp. 302–312, June 1977.

[6] M. Garey and D. Johnson, *Computers and Intractability - A Guide to the Theory of NP-Completeness*, V. Klee, Ed. Bell Telephone Laboratories, 1979.

[7] S. Deb, M. Yeddanapudi, K. Pattipati, and Y. Bar-Shalom, "A generalized s-d assignment algorithm for multisensor-multitarget state estimation," *IEEE Trans. on Aerospace and Electronic Systems*, vol. 33, no. 2, pp. 523–538, April 1997.

[8] W. Ng, J. Li, S. Godsill, and J. Vermaak, "A review of recent result in multiple target tracking," in *Proc. of the 4th International Symposium on Image and Signal Processing and Analysis*, 2005.

[9] Y. Chung, P. Chou, M. Yang, and H. Chen, "Multiple-target tracking with competitive hopfield neural network based data association," *IEEE Trans. on Aerospace and Electronic Systems*, vol. 43, no. 3, pp. 1180–1188, July 2007.

[10] S. Oh, S. Russel, and S. Sastry, "Markov chain monte carlo data association for general multiple-target tracking problems," *IEEE Conf. on Decision and Control*, 2004.

[11] R. Rubinstein and D. Kroese, *The Cross-Entropy Method - A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning*. Springer Science, 2004.

[12] ——, *Simulation and the Monte Carlo Method*. John Wiley & Sons, Inc., 2007.

[13] J. Spall, *Introduction to Stochastic Search and Optimization*. John Wiley & Sons, Inc., 2003.

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

[15] D. Sigalov and N. Shimkin, "Data association in multi target tracking using cross entropy based algorithms," Technion - Israel Institute of Technology, EE Pub. No. 1660, CCIT Report #703, September 2008.

[16] R. Rubinstein, "A stochastic minimum cross-entropy method for combinatorial optimization and rare-event estimation," *Methodology and Computing in Applied Probability*, no. 1, pp. 1–46, 2005.

[17] ——, "Semi-iterative minimum cross-entropy algorithms for rare-events, counting, combinatorial and integer programming," *Methodology and Computing in Applied Probability*, vol. 10, no. 2, p. 121178, April 2008.

[18] P. de Boer, D. Kroese, S. Mannor, and R. Rubinstein, "A tutorial on the cross-entropy method," *Annals of Operations Research*, vol. 134, no. 1, pp. 19–67, February 2005.

[19] Y. Bar-Shalom, X. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation*. New York : Wiley, 2001.

[20] I. Cox and S. Hingorani, "An efficient implementation of reid's multiple hypotheses tracking algorithm and its evaluation for the purpose of visual tracking," *IEEE Trans. on PAMI*, vol. 18, no. 2, pp. 138–150, February 1996.

[21] I. Cox, "Multiple hypotheses tracking code," http://www.adastral.ucl.ac.uk/~icox/.

[22] D. Sigalov, "Data association in multi target tracking using cross entropy based algorithms," Master's thesis, Technion - Israel Institute of Technology, February 2008.