Accepted for publication in the IEEE Transactions on Automatic Control, to appear

Computing Budget Allocation for Efficient Ranking and Selection of Variances with Application to Target Tracking Algorithms [1]

Lidija Trailović and Lucy Y. Pao
Department of Electrical and Computer Engineering
University of Colorado at Boulder
CB 425
Boulder, CO 80309-0425
E-mail: Lidija.Trailovic@Colorado.EDU and Lucy.Pao@Colorado.EDU

Abstract:

This paper addresses the problem of ranking and selection for stochastic processes, such as target tracking algorithms, where variance is the performance metric. Comparison of different tracking algorithms or parameter sets within one algorithm relies on time-consuming and computationally demanding simulations. We present a method to minimize simulation time, yet to achieve a desirable confidence of the obtained results by applying ordinal optimization and computing budget allocation ideas and techniques, while taking into account statistical properties of the variance. The developed method is applied to a general tracking problem of N_s sensors tracking T targets using a sequential multi-sensor data fusion tracking algorithm. The optimization consists of finding the order of processing sensor information that results in the smallest variance of the position error. Results that we obtained with high confidence levels and in reduced simulation times confirm the findings from our previous research (where we considered only two sensors) that processing the best available sensor the last performs the best, on average. The presented method can be applied to any ranking and selection problem where variance is the performance metric.

[1]Portions of this paper have appeared in the Proc. American Control Conference, Arlington, VA, June 2001.

1 Introduction

Target tracking problems involve processing position measurements (observed by a sensor) from a target of interest, and producing, at each time step, an estimate of the target's current state (position and velocity). Uncertainties in the target motion and in the measured values, usually modeled as additive random noise, lead to corresponding uncertainties in the target state. In many tracking problems there is an additional uncertainty regarding the origin of the received data, which may or may not include measurement(s) from the targets or random clutter (false alarms). This leads to the problem of data association or data correlation [2].

In this paper we consider a system of N_s (not necessarily identical) sensors tracking T targets using a sequential multi-sensor joint probabilistic data association (MSJPDA) algorithm [3, 9, 10, 11, 17]. A centralized processing architecture is assumed, where all sensor measurements are received by a central processor prior to data fusion. We are interested in optimizing tracking performance of the MSJPDA algorithm. The optimization consists of finding the order of processing sensor information that results in the smallest variance of the position error, *i.e.,* the smallest root mean square (RMS) position error. For the special case of two sensors tracking two targets, this problem was addressed in [18].

In the general case of N_s sensors tracking T targets, this is a difficult optimization problem. First, the design space can be large (there are $N_s!$ possible sensor orders for a system with N_s different sensors). Second, a computationally demanding simulation is needed to obtain a sample of the performance metric for a given sensor processing order. Finally, since target tracking is a stochastic process, many samples (and therefore many simulation runs) are required to achieve high confidence levels in comparing, ranking, and selecting the best performing designs.

Two complementary approaches have emerged to handle optimization problems such as the one considered in this paper. If the goal is to find one or more among the best designs rather than accurate estimates of the performance metrics, ordinal comparisons (as opposed to cardinal evaluations) can be used to achieve faster convergence rates and significantly reduce the computational

effort. The idea of ordinal optimization [13, 14, 16] has been applied to a number of optimization problems [4, 7]. The second, complementary idea is to minimize the computational cost of achieving desired confidence levels in ordinal comparisons and selection by intelligently allocating the computing budget among the designs [5, 6, 7, 8]. The procedure for allocating computational effort to competing designs was formulated as a nonlinear optimization problem in [5]. A steepest ascent method to solve the budget allocation problem was described in [7]. An asymptotic rule for computing budget allocation was presented in [8]. These approaches make use of a lower bound for the probability of correct selection introduced in [5].

A distinguishing feature of the problem considered in this paper is that the performance metric is the variance rather than the mean of a random process. Ordinal ranking and selection procedures therefore must take into account statistical properties of the sampled process variance. In order statistics literature, the problem of selecting the population with the smallest variance was addressed in [12], but the problem of computing budget allocation has not been addressed.

The paper is organized as follows. The sequential implementation of the MSJPDA algorithm is briefly reviewed in Section 2, and the optimization problem is defined in Section 3. An algorithm for finding the best design with a desired (pre-defined) probability of correct selection or within a prescribed total computing budget is presented in Section 4, as well as a discussion of convergence and performance properties of the algorithm. Results of applications of the developed algorithm to the problem of optimization of sensor processing order in the sequential MSJPDA target tracking algorithm are presented in Section 5, and concluding remarks are given in Section 6.

2 MSJPDA Tracking Algorithm

The multi-sensor multi-target tracking problem is to track T targets using N_s sensors in a cluttered environment, where measurements from the sensors are received by a central processor at discrete time intervals. Each measurement can originate from at most one target, and some of the measurements arise from targets while others arise from clutter. Some targets may not yield any

measurements in a particular time interval for a particular sensor, and measurement errors due to measurements from one sensor are assumed to be independent of those from another sensor.

Assume the state $\mathbf{x}^t(k)$, $1 \leq t \leq T$, of each target t is determined by known matrices $\mathbf{F}^t(k)$ and $\mathbf{G}^t(k)$ as follows [2, 3]:

$$\mathbf{x}^t(k+1) = \mathbf{F}^t(k)\mathbf{x}^t(k) + \mathbf{G}^t(k)\mathbf{w}^t(k) \tag{1}$$

where $\mathbf{w}^t(k)$ are Gaussian random noise vectors independent for all time intervals k and all targets t, with zero means and known covariances $\mathbf{Q}^t(k)$.

With N_s sensors, let $M_{s,k}$, $1 \leq s \leq N_s$, be the number of measurements from sensor s at the k-th time interval [2, 3, 11, 17]. The target originated position measurements are determined by

$$\mathbf{z}^t_{s,\ell_s}(k) = \mathbf{H}_s(k)\mathbf{x}^t(k) + \mathbf{v}^t_{s,\ell_s}(k), \tag{2}$$

where $1 \leq t \leq T$, $1 \leq s \leq N_s$, and $1 \leq \ell_s \leq M_{s,k}$. The $\mathbf{H}_s(k)$ matrices are known, and each $\mathbf{v}^t_{s,\ell_s}(k)$ is a zero-mean Gaussian noise vector, with known covariances $\mathbf{R}_s(k)$, uncorrelated with other noise vectors. Given a target t and a sensor s, it is not known which measurement ℓ_s originates from the target. That is the problem of data association whereby it is necessary to determine which measurements originate from which targets.

Further background on target tracking and data association methods can be found in [2, 3, 9, 10, 11, 17, 18]. This paper investigates the sequential implementation of the MSJPDA algorithm, where the measurements from each sensor are processed one sensor at a time [11, 18].

2.1 Sequential Implementation of the Multi-sensor JPDA (MSJPDA)

The computational requirements and the performance of a parallel implementation of the MSJPDA algorithm have been studied in [17]. Computational complexity for the parallel implementation grows exponentially with the number of sensors, which led to the search for other ways of implementing the MSJPDA algorithm that are less complex and still have comparable performance. A sequential implementation of the MSJPDA algorithm was presented in [11], where it was shown

that it has linear growth in complexity with the number of sensors, and that it results in better performance (in terms of both RMS position error and track lifetime metrics) for tracking in cluttered environments.

In the sequential implementation of the MSJPDA algorithm, the measurements from each sensor are processed one sensor at a time. The measurements of the first sensor are used to compute an intermediate state estimate and the corresponding position error covariance for each target. The measurements of the next sensor are then used to further improve this intermediate state estimate. Measurements of each of the next sensors are then used sequentially to further update each state estimate and covariance until all the sensors are exhausted. In processing each sensor's measurements, the actual association being unknown, the conditional estimate is determined by taking a weighted average over all possible associations.

2.2 Simulation of the Sequential MSJPDA

The MSJPDA simulator from [18] which was set up in Matlab was expanded for N_s sensors tracking T targets, with the matrices $\mathbf{F}^t$, $\mathbf{G}^t$, $\mathbf{H}_s$, $\mathbf{Q}^t$, and $\mathbf{R}_s$ assumed known and time-invariant in (1) and (2). The initial state is assumed Gaussian with known mean and covariance. The state vectors $\mathbf{x}^t(k) = [x\ \dot{x}\ y\ \dot{y}]'(k)$ represent the positions and velocities of the targets at time k, and the targets nominally move in straight lines with constant velocity, though corrupted by process noise. The process noise and measurement noise covariances $\mathbf{Q}^t$ and $\mathbf{R}_s$, respectively, and the measurement matrices $\mathbf{H}_i$ are as in [18]:

$$\mathbf{Q}^t = \begin{bmatrix} q & 0 \\ 0 & q \end{bmatrix}, \quad 1 \le t \le T\,, \tag{3}$$

$$\mathbf{R}_s = \begin{bmatrix} r_s & 0 \\ 0 & r_s \end{bmatrix}, \quad \text{and} \tag{4}$$

$$\mathbf{H}_s = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad 1 \le s \le N_s. \tag{5}$$

The method developed in this paper is applicable to general $\mathbf{R}_s$, but we consider only diagonal $\mathbf{R}_s = \mathbf{I}\, r_s$ because it allows us to more easily compare sensor qualities (better sensor has smaller

r_s). The measurements corresponding to the sensor with covariance $\mathbf{R}_1$ are processed first, and measurements from the sensor with covariance $\mathbf{R}_{N_s}$ are processed last in the sequential MSJPDA.

We assume that the position error ε, which is the difference between the true target position and the target position estimate, is a Gaussian random variable with zero mean and standard deviation σ, *i.e.,* $\varepsilon \sim N[0, \sigma]$. The standard deviation σ is the *true* RMS position error for a given design (*i.e.,* a given set of sensors) and system parameters. Our objective is to compare and rank the designs based on the estimated standard deviation $\hat{\sigma}$, *i.e.,* the *measured* RMS position error computed from the data generated by the tracking simulator.

Let ε_r be a sample of the position error computed by the simulator as the difference between the true target position and the target position estimate. After the simulator generates n position error samples $\{\varepsilon_r\}_{r=1}^{n}$, the estimate of the mean of the position error is

$$\bar{\varepsilon} = \frac{1}{n} \sum_{r=1}^{n} \varepsilon_r \, . \tag{6}$$

Under the assumption that the position error is a zero-mean random variable, $\bar{\varepsilon} \approx 0$, the unbiased estimate of the standard deviation of the position error is [15]

$$\hat{\sigma} = \text{measured RMS error} = \sqrt{\frac{1}{n-1} \sum_{r=1}^{n} \varepsilon_r^2}. \tag{7}$$

The standard deviation estimate $\hat{\sigma}$ is the measured RMS position error. This is an approximation to the true RMS position error. By the strong law of large numbers,

$$\sigma = \lim_{n \to \infty} \hat{\sigma} \tag{8}$$

with probability 1. This means that the estimated standard deviation of the position error approaches the true RMS position error as the number n of position error samples generated by the simulator increases.

3 Notation and Problem Formulation

In this section we establish the notation used, and formulate the optimization problem.

S sensor set, $S = \{r_1, r_2, \ldots, r_{N_s}\}$; r_s is the parameter in the sensor noise covariance matrix, defined in (4),

θ_i ith design, an ordered subset of S, $\theta_i = [r_{i1}, r_{i2}, \ldots, r_{iN_i}]$, $N_i \leq N_s$,

Θ design space, $\Theta = \{\theta_1, \ldots, \theta_{N_d}\}$; $N_d = |\Theta|$ is the size of the design space,

σ_i^2 variance of the position error for the design θ_i; σ_i is the true RMS position error for the design θ_i,

n_i number of samples (simulation runs) for the design θ_i,

$\hat{\sigma}_i$ standard deviation estimate, *i.e.,* RMS position error obtained after n_i simulation runs for the design θ_i,

θ_b selected best ranking design, $\hat{\sigma}_b < \hat{\sigma}_i$, for all $i \neq b$,

θ_s second best ranking design, $\hat{\sigma}_b < \hat{\sigma}_s < \hat{\sigma}_i$, for all $i \neq b$, $i \neq s$,

P_{CS} *a posteriori* probability of correct selection,

$$P_{CS} = P\{\sigma_b < \sigma_i, \text{ for all } i \neq b \mid \hat{\sigma}_i \text{ obtained after } n_i \text{ simulation runs for the design } \theta_i\},$$

$APCS$ approximate probability of correct selection,

$EPCS_u$ estimated approximate probability of correct selection if the number of runs n_u for the design θ_u is incremented by 1,

$P^\star$ desired probability of correct selection, *i.e.,* desired confidence level (predefined, *e.g.,* 90%),

N_{runs} total number of simulation runs, $N_{runs} = \sum_{i=1}^{N_d} n_i$,

N_{max} maximum allowed number of simulation runs, $N_{runs} \leq N_{max}$.

The optimization problem of finding the best design on the entire design space can be redefined in two ways:

1. Select the best ranking design θ_b s.t. $P_{CS} \geq P^\star$, while minimizing the total number of simulation runs N_{runs}.

2. Select the best ranking design θ_b s.t. $N_{runs} \leq N_{max}$, while maximizing the probability of correct selection P_{CS}.

4 Ranking and Selection of Variances

In this section statistical properties of variance as a performance metric are summarized and optimization algorithms are described, accompanied with a discussion of convergence and performance properties.

4.1 Statistics of the Position Error Variance

For a design θ_i, the position error ε is a Gaussian random variable with zero mean and standard deviation σ_i. It is desired to compare two designs using the standard deviation σ_i, *i.e.,* the true RMS position error, as the performance metric. We assume that no prior knowledge is available about the performance of any design. Therefore, following the Bayesian model, the comparison is made based on the measured RMS position error (7).

Consider two designs, θ_i and θ_j, with $\hat{\sigma}_i$ and $\hat{\sigma}_j$ computed according to (7). Suppose that $\hat{\sigma}_i < \hat{\sigma}_j$. After the simulation experiments that produced n_i samples of the position error for the design θ_i and n_j samples of the position error for the design θ_j, the design θ_i is selected as the better performing design. To find the confidence of this selection, let us find the *a posteriori* probability p_{ij} that the design θ_i is indeed better than the design θ_j, *i.e.,* that $\sigma_i < \sigma_j$, given $\hat{\sigma}_i < \hat{\sigma}_j$,

$$p_{ij} = P\left\{\sigma_i < \sigma_j \;\middle|\; \hat{\sigma}_i, \hat{\sigma}_j\right\} \tag{9}$$

or, equivalently,

$$p_{ij} = P\left\{\left(\frac{\sigma_j}{\sigma_i}\right)^2 > 1 \;\middle|\; \hat{\sigma}_i, \hat{\sigma}_j\right\}. \tag{10}$$

The random variable x defined by

$$x = \frac{\hat{\sigma}_j^2}{\sigma_j^2} \frac{\sigma_i^2}{\hat{\sigma}_i^2} \tag{11}$$

has a F_{n_i-1,n_j-1} distribution, with (n_i-1) degrees of freedom in the numerator and (n_j-1) degrees of freedom in the denominator. The PDF is given by [20]

$$f(n_i,\, n_j,\, x) = \begin{cases} \dfrac{\Gamma\left(\frac{n_i-1}{2}+\frac{n_j-1}{2}\right)\left(\frac{n_i-1}{n_j-1}\right)^{\frac{n_i-1}{2}} x^{\frac{n_i-1}{2}-1}}{\Gamma\left(\frac{n_i-1}{2}\right)\Gamma\left(\frac{n_j-1}{2}\right)\left(1+\frac{n_i-1}{n_j-1}\, x\right)^{\left(\frac{n_i-1}{2}+\frac{n_j-1}{2}\right)}}, & x > 0 \\ 0, & x \leq 0 \end{cases} \tag{12}$$

where $\Gamma(z)$ is the Gamma function

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt. \tag{13}$$

Given n_i, $\hat{\sigma}_i$, n_j, and $\hat{\sigma}_j$, the probability p_{ij} that the design θ_i produces a smaller variance than the design θ_j can be found as the probability that x is less than $\left(\frac{\hat{\sigma}_j}{\hat{\sigma}_i}\right)^2$,

$$p_{ij} = \text{CDF}\left(n_i, n_j, \left(\frac{\hat{\sigma}_j}{\hat{\sigma}_i}\right)\right), \tag{14}$$

or equivalently [20]

$$p_{ij} = 1 - \text{CDF}\left(n_j,\, n_i,\, \frac{1}{\xi_{ij}^2}\right) = \int_{1/\xi_{ij}^2}^\infty f(n_j, n_i, x) dx \tag{15}$$

where

$$\xi_{ij} = \frac{\hat{\sigma}_j}{\hat{\sigma}_i} > 1 \tag{16}$$

and CDF$(\cdot)$ is the cumulative density function of the random variable x. The probability p_{ij} in (15) can be computed easily.

To illustrate the behavior of p_{ij} as a function of n_i, n_j, and ξ_{ij}, Figures 1 and 2 show p_{ij} when n_i and n_j range from 2 to 100, with ξ_{ij} being a parameter. Figure 1 shows two views (presented to help clarify the discussion) of the surface plot when the two variances being compared differ considerably ($\hat{\sigma}_i = 1.0$ and $\hat{\sigma}_j = 1.4$, producing $\xi_{ij} = 1.4$ in (16)). It can be observed that p_{ij} increases either with n_i, or with n_j, or both. Figure 2 shows two views of the surface plot when the two variances differ only slightly ($\hat{\sigma}_i = 1.0$ and $\hat{\sigma}_j = 1.05$, producing $\xi_{ij} = 1.05$ in (16)). It can

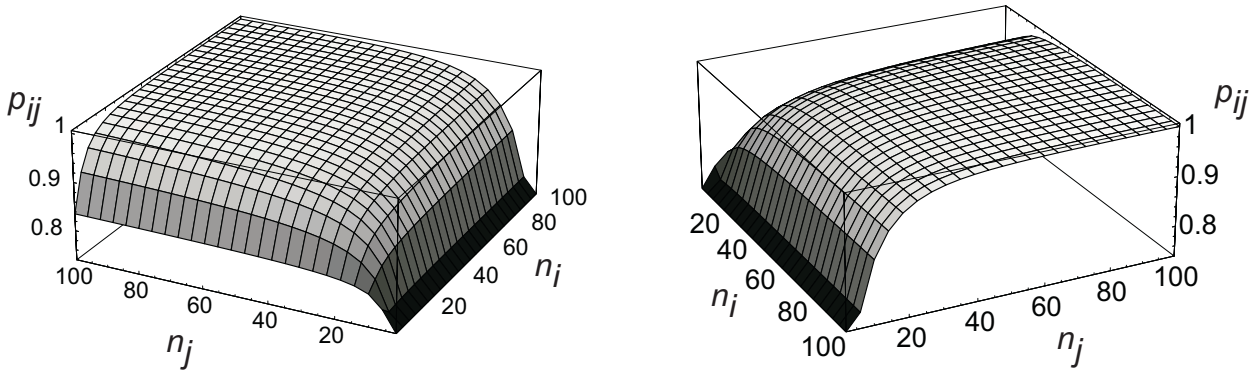

Figure 1: Probability p_{ij} as a function of n_i and n_j for $\xi_{ij} = 1.4$. n_i and n_j range from 2 to 100 and $\xi_{ij} = 1.4$ can be considered a large difference in variances $\hat{\sigma}_i$ and $\hat{\sigma}_j$.

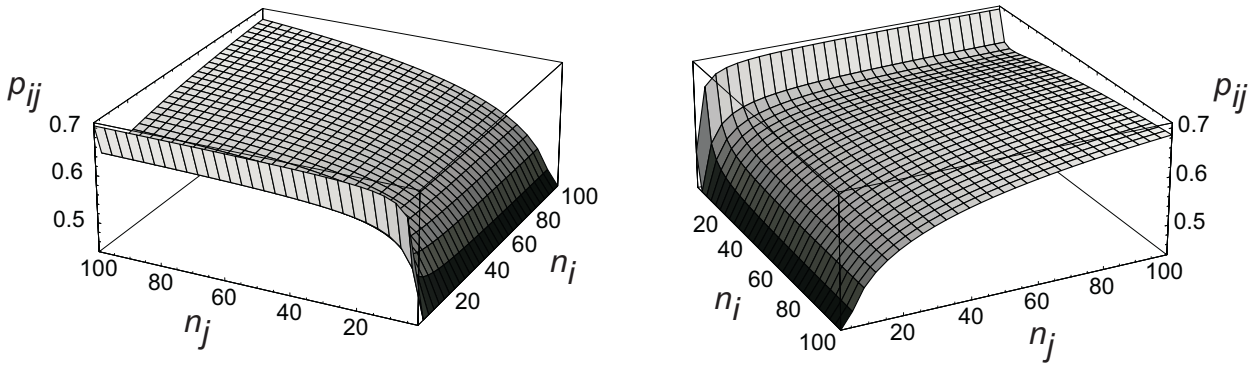

Figure 2: Probability p_{ij} as a function of n_i and n_j for $\xi_{ij} = 1.05$. n_i and n_j range from 2 to 100 and $\xi_{ij} = 1.05$ can be considered a small difference in variances $\hat{\sigma}_i$ and $\hat{\sigma}_j$.

be observed that p_{ij} does not increase monotonically with n_i for a fixed n_j. This behavior of p_{ij} and its impact on developing successfully converging computing budget allocation algorithms will be discussed in more detail in Section 4.4.

In the optimization problem of finding the best design on the entire design space, it is necessary to extend the result for the probability of correct selection p_{ij} among two designs to the probability of correct selection P_{CS}. In general, the probability of correct selection P_{CS} can be computed as described in [12]. However, for an arbitrary number of designs, and an arbitrary number of samples n_i per design θ_i, the numerical computation of P_{CS} is quite involved. We instead make use of the approximate probability of correct selection ($APCS$) introduced in [5].

The approximate probability of the correct selection ($APCS$) can be computed according to [5] as

$$APCS = \prod_{i=1, i \neq b}^{N_d} p_{bi} \,. \tag{17}$$

It has been shown that $APCS$ is an asymptotic lower bound for P_{CS} [7]:

$$P_{CS} \geq APCS \tag{18}$$

so that $APCS$ can replace P_{CS} in the optimization algorithm.

4.2 Uniform Computing Budget Allocation

A simple approach to solve the optimization problem is to perform a sufficiently large, equal number of simulation runs for all designs. This uniform computing budget allocation (UCBA) algorithm is

UCBA algorithm

Perform 2 simulation runs for each of N_d designs θ_i;

Update $\hat{\sigma}_i$ for all i; compute $APCS$;

While ($APCS < P^\star$ and $N_{runs} < N_{max}$)

  Perform one simulation run for each of N_d designs;

  Update $\hat{\sigma}_i$ for all i; compute $APCS$;

End-while

The algorithm starts with performing two simulation runs, because p_{ij} in (15) is undefined for n_i, $n_j < 2$. The algorithm terminates when $APCS$ becomes greater than the desired probability of correct selection $P^\star$, or when the total number of simulation runs N_{runs} exceeds the specified maximum computing budget N_{max}.

4.3 Optimized Computing Budget Allocation

To minimize the total number of simulation runs, we propose an optimized computing budget allocation (OCBA) algorithm where only one simulation run is performed in each iteration, as opposed to simulation of all designs. The design θ_m selected for simulation is the design for which incrementing the number of runs n_m is expected to result in the largest $APCS$. Let us define

$$p_{ij^*} = 1 - \text{CDF}\left(n_j + 1,\, n_i,\, 1/\xi_{ij}^2\right) \tag{19}$$

$$p_{i^*j} = 1 - \text{CDF}\left(n_j,\, n_i + 1,\, 1/\xi_{ij}^2\right) \tag{20}$$

and the estimated approximate probability of correct selection $EPCS_u$ for the design θ_u as

$$EPCS_u = \begin{cases} p_{bu^*} \prod_{i=1, i\neq b, i\neq u}^{N_d} p_{bi}, & u \neq b \\ \\ \prod_{i=1, i\neq b}^{N_d} p_{b^*i}, & u = b\,. \end{cases} \tag{21}$$

This definition of $EPCS_u$ is similar to the definition of $EPCS$ in [7]. After n_1, n_2 $\ldots, n_{N_d}$ runs are completed for designs θ_1, $\theta_2, \ldots, \theta_{N_d}$, $EPCS_u$ is an estimated $APCS$ based on the already available statistical information, but with the number of runs n_u for the design θ_u increased by 1. $EPCS_u$ can be easily computed for all designs. The design θ_m selected for simulation is the one that maximizes $EPCS_u$,

$$m = \arg \max_{1 \leq u \leq N_d} (EPCS_u)\ , \tag{22}$$

and only one simulation run of the MSJPDA tracking algorithm is performed in each iteration. Compared to the more general approach presented in [7], this simple approach of performing only one simulation run in each iteration of the OCBA algorithm is well justified by the fact that our tracking problem has the cost of one simulation run much higher than the cost of all other computations performed in one iteration.

The algorithm with optimized computing budget allocation is:

OCBA algorithm

Perform 2 simulation runs for each of N_d designs θ_i;

Update $\hat{\sigma}_i$ for all i; compute $APCS$ and p_{bs};

While ($APCS < P^{\star}$ and $N_{runs} < N_{max}$)

If ($p_{bs} < P_{init}$ and $n_s < N_{init}$)

Perform one simulation run for each of N_d designs;

Update $\hat{\sigma}_i$ for all i;

Else

Find m such that $EPCS_m = \max_u(EPCS_u)$;

Perform one simulation run for the design θ_m;

Update $\hat{\sigma}_m$;

End-if

Compute $APCS$ and p_{bs};

End-while

The algorithm starts by performing two simulation runs for all designs. Iterations are then performed until $APCS$ exceeds the desired confidence level P^* or the number of simulation runs N_{runs} exceeds the maximum allowed number of runs N_{max}. In each iteration, either all N_d designs are simulated, or only the design θ_m selected according to (22) is simulated. The decision about performing simulation of all designs or just one selected design, and the selection of the parameters N_{init} and P_{init} are related to the convergence issues discussed in the next section. The probability p_{bs}, which is the confidence that the best ranking design θ_b is better than the second best ranking design θ_s, is found as p_{ij} in (15) for $i = b$ and $j = s$.

4.4 Convergence

Suppose that the same number of simulation runs is allocated to each design, as in the UCBA algorithm, $n_1 = n_2 = \ldots = n_{N_d} = n$. Then, it can be shown that [20]

$$\lim_{n\to\infty} p_{bi} = 1, \text{ for all } i \neq b\,. \tag{23}$$

Therefore, $APCS$ defined by (17) in the UCBA algorithm converges to 1 as the number of simulation runs increases, and $APCS \geq P^\star$ is obtained in a finite number of simulation runs. In the OCBA, the number of simulation runs n_i for different designs is not the same, and the proof of convergence requires a more detailed analysis.

Figure 3 illustrates how p_{bs} depends on the number of runs n_s, for several fixed values of n_b. For a fixed n_b, the probability p_{bs} increases monotonically with n_s, but the asymptote for $n_s \to \infty$ is less than 1. Therefore, increasing the number of runs only for the second best ranking design θ_s does not guarantee convergence.

Figure 4 shows how p_{bs} depends on the number of runs n_b, for several fixed values of n_s. For a given n_s and small n_b, the probability p_{bs} initially decreases with n_b. Depending on the value of n_s, p_{bs} may continue to decrease with n_b, or it may reach a minimum after which it starts increasing with n_b. This behavior of p_{bs} is observed for ξ_{bs} close to one, *i.e.,* when designs θ_b and θ_s have $\hat{\sigma}_b$ and $\hat{\sigma}_s$ that do not differ by much. It is a result of comparing standard deviations (RMS position errors) that leads to the F_{n_i-1,n_j-1} distribution. To avoid non-convergence that would result from this counter-intuitive behavior of p_{bs}, our algorithm performs a certain number of iterations with the UCBA algorithm until conditions are met for p_{bs} to become monotonically increasing with n_b. Since $APCS$ is defined by (17), it is only necessary to consider these conditions for p_{bs}, because $\xi_{bi} > \xi_{bs}$ and therefore $p_{bi} > p_{bs}$ for all other designs θ_i.

A sufficient condition for convergence is given by the following: given $\xi_{bs} > 1$, and a sufficiently large n_s, there exists N_{init} such that p_{bs} is monotonically increasing with n_b, for $n_b \geq N_{init}$.

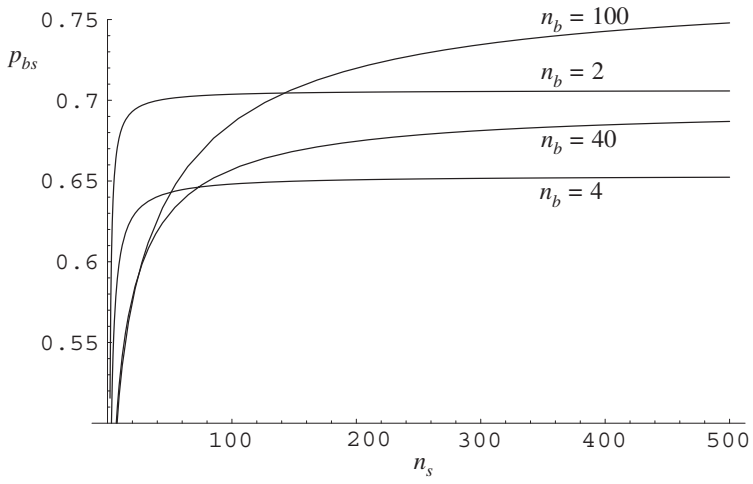

Figure 3: Probability p_{bs} as a function of n_s for several fixed values of n_b and $\xi_{bs} = 1.05$.

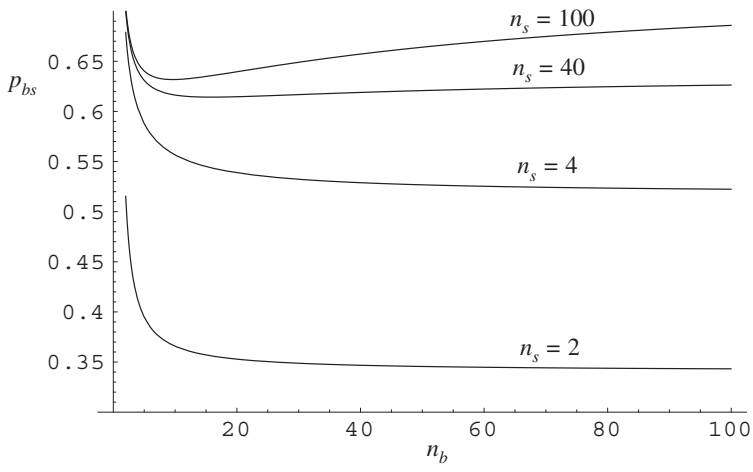

Figure 4: Probability p_{bs} as a function of n_b for several fixed values of n_s and $\xi_{bs} = 1.05$.

Since we are interested in the asymptotic behavior of p_{bs} for large n_b and n_s, according to (8) we can assume that $\hat{\sigma}_b \approx \sigma_b$ and $\hat{\sigma}_s \approx \sigma_s$. To simplify notation, we define

$$\alpha = \frac{n_b - 1}{2} \tag{24}$$

$$\beta = \frac{n_s - 1}{2} \tag{25}$$

$$\xi = \left(\frac{\sigma_s}{\sigma_b}\right)^2 \tag{26}$$

$$g(\beta, \alpha, x) = f(n_s, n_b, x). \tag{27}$$

To find how p_{bs} behaves for large n_s, let us define

$$p_b(n_b, \xi) = \lim_{n_s \to \infty} p_{bs}\,. \tag{28}$$

Following (15),

$$p_{bs} = \int_{1/\xi}^{\infty} g(\beta,\, \alpha,\, x) dx \tag{29}$$

and

$$p_b(n_b, \xi) = \lim_{\beta \to \infty} \int_{1/\xi}^{\infty} g(\beta,\, \alpha,\, x) dx = \int_{1/\xi}^{\infty} \lim_{\beta \to \infty} g(\beta,\, \alpha,\, x) dx. \tag{30}$$

The F-distribution given by (12) can be rewritten, for $n_i = n_b$ and $n_j = n_s$, as

$$g(\beta,\, \alpha,\, x) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\, \Gamma(\beta)}\, \frac{x^{-1}}{(1 + \frac{\alpha}{\beta}\frac{1}{x})^{\beta} (1 + \frac{\beta}{\alpha} x)^{\alpha}}\,. \tag{31}$$

For $\beta \to \infty$,

$$\left(1 + \frac{\alpha}{\beta\, x}\right)^{\beta} \longrightarrow e^{\alpha/x} \tag{32}$$

and

$$\left(1 + \frac{\beta}{\alpha}\, x\right)^{\alpha} \left(\frac{\beta}{\alpha}\, x\right)^{-\alpha} \longrightarrow 1\,. \tag{33}$$

Thus, we have

$$g(\beta,\, \alpha,\, x) \longrightarrow \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\, \Gamma(\beta)}\; e^{-\alpha/x}\; \left(\frac{\alpha}{\beta}\right)^{\alpha}\; x^{-(\alpha+1)} \tag{34}$$

and hence

$$p_{bs} \longrightarrow \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\, \Gamma(\beta)}\; \left(\frac{\alpha}{\beta}\right)^{\alpha}\; \int_{1/\xi}^{\infty} e^{-\alpha/x}\, x^{-(\alpha+1)}\, dx. \tag{35}$$

The integral above can be solved as [1]:

$$\int_{1/\xi}^{\infty} e^{-\alpha/x}\, x^{-(\alpha+1)}\, dx = \left(\frac{1}{\alpha}\right)^{\alpha+1} \left[\Gamma(\alpha+1) - \alpha\Gamma\left(\alpha,\, \xi\alpha\right)\right], \tag{36}$$

where $\Gamma\left(\alpha,\, \xi\alpha\right)$ is the incomplete Gamma function

$$\Gamma\left(\alpha,\, \xi\alpha\right) = \int_{\xi\alpha}^{\infty} t^{\alpha-1} e^{-t} dt. \tag{37}$$

For a fixed α, using the property $\Gamma(\alpha+1) = \alpha\Gamma(\alpha)$, from (35) and (36) we have

$$p_{bs} \longrightarrow \frac{\Gamma(\alpha+\beta)}{\Gamma(\beta)} \frac{1}{\beta^{\alpha}} \left[1 - \frac{\Gamma\left(\alpha,\, \xi\alpha\right)}{\Gamma(\alpha)}\right], \tag{38}$$

where

$$\frac{\Gamma(\alpha+\beta)}{\Gamma(\beta)} \frac{1}{\beta^{\alpha}} = \frac{(\alpha+\beta-1)\ldots(\beta+1)\beta\Gamma(\beta)}{\Gamma(\beta)} \frac{1}{\beta^{\alpha}} \longrightarrow 1 \;\text{ as }\; \beta \longrightarrow \infty\,. \tag{39}$$

Therefore,

$$p_b(n_b, \xi) = p_{bs} \longrightarrow 1 - \frac{\Gamma\left(\alpha,\, \xi\alpha\right)}{\Gamma(\alpha)} \;\text{ as }\; \beta \longrightarrow \infty\,. \tag{40}$$

For large α, the uniform asymptotic expansion of the incomplete Gamma function [19] in (40) yields

$$1 - \frac{\Gamma\left(\alpha,\, \xi\alpha\right)}{\Gamma(\alpha)} \approx \frac{1}{2}\, \mathrm{erfc}\left((1-\xi)\sqrt{\frac{\alpha}{2}}\right). \tag{41}$$

Given $\xi > 1$, the complementary error function in (41) is monotonically increasing and approaches 1 as $\alpha \to \infty$, where α is defined in (24). Therefore, for sufficiently large n_s there exists N_{init} such that p_{bs} is monotonically increasing with n_b, for $n_b \geq N_{init}$. Since $p_{bi} > p_{bs}$ for all other designs θ_i, $i \neq b$, $i \neq s$, we can conclude that $APCS$ defined in (17) asymptotically approaches 1 provided that the OCBA algorithm starts after a sufficiently large number of samples $n_i \geq N_{init}$ has been collected for all designs θ_i.

For the algorithm implementation, it is of interest to quantify the behavior of $p_b(n_b, \xi)$ as a function of n_b. Figure 5 shows plots of $p_b(n_b, \xi)$ in (40) as a function of n_b for $\xi = (1.01)^2$ and $(1.02)^2$. From the plots we can conclude that selecting $N_{init} \geq 35$ (≥ 18) is sufficient to guarantee convergence, given that the measured RMS position errors differ by at least 1% (2%). In the algorithm, the UCBA algorithm iterations are performed until the number of simulation runs for

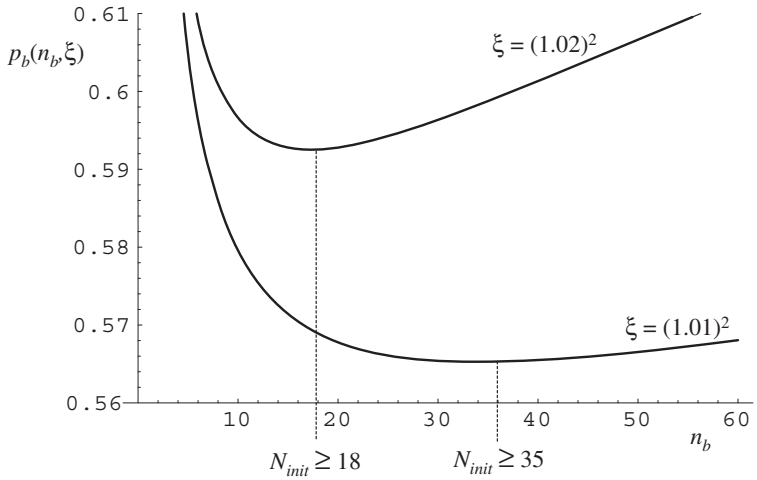

Figure 5: Probability $p_b(n_b, \xi)$ as a function of n_b for $\xi = (1.01)^2$ and $(1.02)^2$.

each design exceeds N_{init}. If the designs differ by more than the lower limiting value ($\xi > 1.01$ or 1.02), the number of runs in the UCBA iteration can use a smaller N_{init}. The initial iterations can also be terminated if $p_{bs} > P_{init}$. The value of $P_{init} = 59\%$ is found as the value of p_{bs} for $n_b = n_s = 2$ and $\xi = 1.334$ where $N_{init} \geq 2$ is sufficient to guarantee convergence. With $P_{init} = 59\%$ and $N_{init} = 35$, the convergence conditions guarantee that $APCS \geq P^\star$ in a finite number of simulation runs, provided that the designs differ by more than 1% (*i.e.,* $\xi \geq 1.01$), but there is no guarantee about the rate of convergence. Generally, it is not of interest to labor over distinguishing between designs that are less than 1% different in performance. However, if necessary, (40) can be used to determine the minimum N_{init} to use for ξ less than 1.01.

4.5 Algorithm Comparison

In this section we compare the OCBA algorithm (Section 4.3) with the UCBA algorithm (Section 4.2) using synthetic data obtained from a random number generator with known σ_i for the design θ_i. In the Monte Carlo experiments with synthetic data the results are averaged over a very large number $(10,000)$ of algorithm runs.

Figure 6 illustrates the comparison of the OCBA algorithm with the UCBA algorithm when

there are $N_d = 10$ competing designs with standard deviations between $\sigma_{min} = 1$ and $\sigma_{max} = 2$, and

$$\frac{\sigma_{max}}{\sigma_{min}} = (\xi_{i,i+1})^{N_d-1} = 2, \quad \text{for } 1 \leq i \leq N_d, \tag{42}$$

that is, any two consecutive designs differ in performance metrics by 8% ($\xi_{i,i+1} = 1.08$). As expected, with both algorithms the average achieved $APCS$ increases with the increase of the total number of runs N_{runs}. However, the OCBA algorithm achieves a high probability of correct selection in significantly fewer simulation runs. For example, to achieve $APCS = 90\%$, the OCBA algorithm requires a total number of runs $N_{runs} = 1000$, while the UCBA algorithm requires $N_{runs} = 2800$. Figure 7 shows a comparison of the two algorithms when the σ's of 10 designs differ even less ($\xi_{i,i+1} = 1.046$). Note that the designs have σ's that are very close, which would make it difficult to distinguish the best one for any optimization algorithm. The OCBA algorithm achieves confidence levels greater than 80% in less than half of the number of simulation runs required by the UCBA algorithm. If higher confidence levels are required, the advantages of the OCBA algorithm in reducing the computational effort are even more significant.

5 Sensor Processing Order Optimization Examples

The algorithms from Sections 4.2 and 4.3 were applied to several examples of optimization of sensor processing order in the MSJPDA tracking simulator. In all simulations, the system noise was $q = 0.01$ and the clutter density was $\lambda = 1.4$, while the number of sensors N_s and the number of targets T varies. Different designs correspond to the different number of identical sensors or to different orders of non-identical sensors. Better sensors have a smaller r_i sensor noise parameter defined in (4). The expected clutter measurements per target gate varies with the quality of the sensor used from 0.08 for the best sensor up to 3.12 for the worst sensor. Tables 1 through 6 summarize results obtained using the OCBA algorithm applied to the sequential MSJPDA. The chosen best design is marked in boldface.

Table 1 presents a comparison of designs consisting of different numbers of identical sensors.

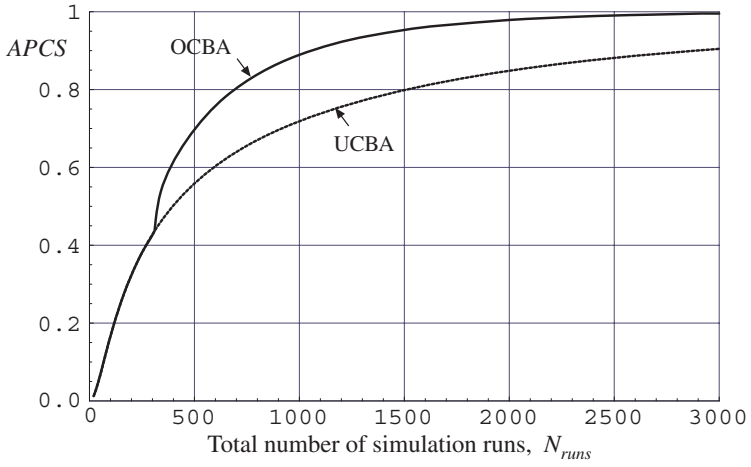

Figure 6: Comparison of the average achieved probability of correct selection $APCS$ as a function of the total number of simulation runs for the uniform computing budget allocation (UCBA) algorithm and for the optimized computing budget allocation (OCBA) algorithm. The number of designs is $N_d = 10$; $\xi^9 = 2$, $\xi = 1.08$.

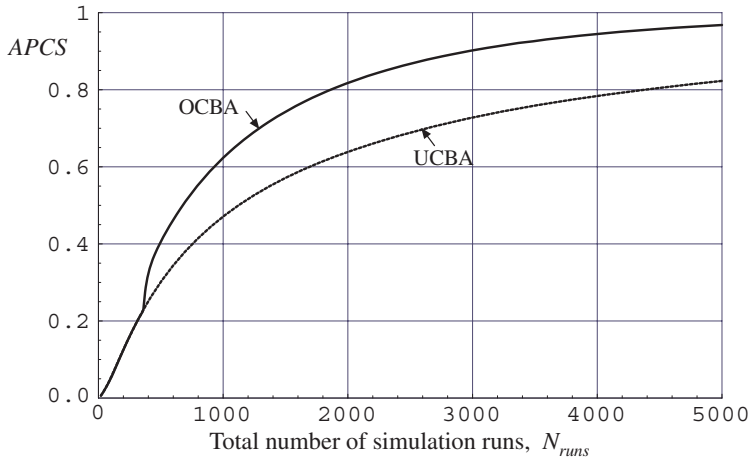

Figure 7: Comparison of the average achieved probability of correct selection $APCS$ as a function of the total number of simulation runs N_{runs} for the uniform computing budget allocation (UCBA) algorithm and for the optimized computing budget allocation (OCBA) algorithm. The number of designs is $N_d = 10$; $\xi^9 = 1.5$, $\xi = 1.046$.

As may be expected, performance improves as the number of sensors increases ($N_s = 5$ has better performance than $N_s = 4$), but for some N_s, improvement sometimes can not be verified in a predefined maximum number of runs N_{max} (comparing designs when $N_s = 5$ and $N_s = 6$, $N_{max} = 200$ was exhausted before the $APCS$ achieved the desired $P^\star = 90\%$).

Table 2 compares orders of sensor processing of a two-sensor system where the overall quality of the two sensors is the same in all cases, that is, $1/r_e = 1/r_1 + 1/r_2$, as in [18], and we arrived at the same conclusion that processing the best sensor last yields the smallest RMS position error. Further, we came to the conclusion in only $N_{runs} = 15$ (in [18] we used 200 Monte Carlo runs) and with $APCS > 95\%$. Results from Table 3 confirm our previous findings [18] that two identical sensors outperform two different quality sensors in any order.

In Table 4, results for a more complex tracking system is presented, with $N_s = 4$ and $T = 5$. The increase of $APCS$ with N_{runs} for these designs is presented in Figure 8. The observation from above still holds, the best design is the one where the best sensor is being processed the last. For these four designs we also ran the UCBA algorithm, and it produced $APCS = 85.27\%$ with $n_i = 193$ runs per design, or $N_{runs} = 772$, while the OCBA algorithm achieved the same confidence in $N_{runs} = 548$.

Results obtained when more diverse sensors are used in the sequential MSJPDA are shown in Table 5, and the increase of $APCS$ with N_{runs} is shown in Figure 9 for the listed eight designs. The conclusion about processing the best available sensor last still holds. For these design choices, improvement over the UCBA algorithm is even more significant. The UCBA algorithm achieved an $APCS = 85.17\%$ in $N_{runs} = 1472$, or $n_i = 184$ per design, while the OCBA algorithm achieved an $APCS = 85.03\%$ in $N_{runs} = 897$.

Table 6 presents another illustration of the same trend, with $N_s = 3$ different sensors and $N_d = 6$ designs, and the best ranking design is again the one with the best sensor processed last. In fact, processing the best sensor last leads to the best two sensor orderings, θ_4 and θ_6.

From the examples shown in this section, some general trends can be observed. Starting from a sufficiently large number of runs, further increasing the number of runs for the best performing

$T = 3$, $N_s = 2, 3, 4, 5, 6$, $P^\star = 90\%$		
design	n_i	RMS_i
$\theta_1 = \{0.01, 0.01\}$	6	0.025676
$\boldsymbol{\theta_2 = \{0.01, 0.01, 0.01\}}$	**11**	**0.016155**
$N_{runs} = 17$, $APCS = 90.04\%$, $\theta_b = \theta_2$		
$\theta_1 = \{0.01, 0.01, 0.01\}$	16	0.016881
$\boldsymbol{\theta_2 = \{0.01, 0.01, 0.01, 0.01\}}$	**25**	**0.012622**
$N_{runs} = 41$, $APCS = 90.12\%$, $\theta_b = \theta_2$		
$\theta_1 = \{0.01, 0.01, 0.01, 0.01\}$	73	0.012566
$\boldsymbol{\theta_2 = \{0.01, 0.01, 0.01, 0.01, 0.01\}}$	**91**	**0.010893**
$N_{runs} = 164$, $APCS = 90.09\%$, $\theta_b = \theta_2$		
$\theta_1 = \{0.01, 0.01, 0.01, 0.01, 0.01\}$	84	0.010625
$\boldsymbol{\theta_2 = \{0.01, 0.01, 0.01, 0.01, 0.01, 0.01\}}$	**116**	**0.009492**
$N_{runs} = 200$, $APCS = 75.25\%$, $\theta_b = \theta_2$		

Table 1: Application of the optimized computing budget allocation algorithm: desired confidence $P^\star = 90\%$ and $N_{max} = 200$ for $T = 3$ targets and different numbers of sensors, $N_s = 2, 3, 4, 5$, and 6.

$T = 2$, $N_s = 2$		
$P^\star = 90\%$		$N_{runs} = 15$
$APCS = 95.12\%$		$\theta_b = \theta_2$
design	n_i	RMS_i
$\theta_1 = \{0.011, 0.11\}$	5	0.21756
$\boldsymbol{\theta_2 = \{0.11, 0.011\}}$	**10**	**0.11361**

Table 2: Application of the optimized computing budget allocation algorithm: desired confidence $P^\star = 90\%$, $N_{max} = 100$, $N_{runs} = 15$, and the selected best ranking design is θ_2 with $APCS = 95.12\%$.

$T = 2$, $N_s = 2$		
$P^\star = 90\%$		$N_{runs} = 13$
$APCS = 95.66\%$		$\theta_b = \theta_1$
design	n_i	RMS_i
$\boldsymbol{\theta_1 = \{0.02, 0.02\}}$	**8**	**0.05263**
$\theta_2 = \{0.011, 0.11\}$	3	0.18364
$\theta_3 = \{0.11, 0.011\}$	2	0.13396

Table 3: Application of the optimized computing budget allocation algorithm: desired confidence $P^\star = 90\%$, $N_{max} = 100$, $N_{runs} = 13$, and the selected best ranking design is θ_1 with $APCS = 95.66\%$.

$T = 5$, $N_s = 4$		
$P^\star = 85\%$		$N_{runs} = 548$
$APCS = 85.24\%$		$\theta_b = \theta_1$
design	n_i	RMS_i
$\boldsymbol{\theta_1 = \{0.1, 0.1, 0.1, 0.01\}}$	**232**	**0.047529**
$\theta_2 = \{0.1, 0.1, 0.01, 0.1\}$	161	0.051999
$\theta_3 = \{0.1, 0.01, 0.1, 0.1\}$	113	0.056401
$\theta_4 = \{0.01, 0.1, 0.1, 0.1\}$	42	0.062513

Table 4: Application of the optimized computing budget allocation algorithm: desired confidence $P^\star = 85\%$, $N_{max} = 1000$, $N_{runs} = 548$, and the selected best ranking design is θ_1 with $APCS = 85.24\%$. $APCS$ as a function of N_{runs} for the listed designs is presented in Figure 8.

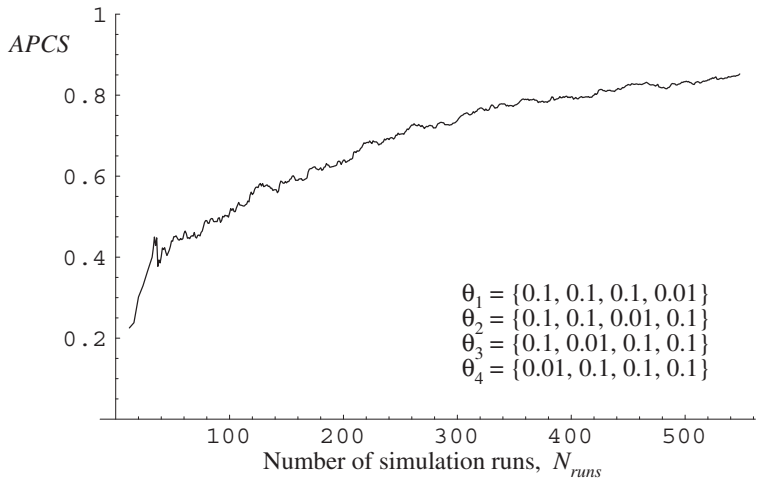

Figure 8: Achieved APCS as a function of total number of simulation runs. Tracking $T = 5$ targets, $\lambda = 1.4$; $\theta_b = \theta_1$ with $APCS = 85.24\%$. The total number of runs was $N_{runs} = 548$.

$T = 5$, $N_s = 4$		
$P^\star = 85\%$		$N_{runs} = 897$
$APCS = 85.03\%$		$\theta_b = \theta_4$
design	n_i	RMS_i
$\theta_1 = \{0.01, 0.1, 0.1, 1.0\}$	42	0.072662
$\theta_2 = \{0.01, 0.1, 1.0, 0.1\}$	55	0.069468
$\theta_3 = \{0.01, 1.0, 0.1, 0.1\}$	39	0.073231
$\boldsymbol{\theta_4 = \{1.0, 0.1, 0.1, 0.01\}}$	**294**	**0.054996**
$\theta_5 = \{1.0, 0.1, 0.01, 0.1\}$	143	0.061807
$\theta_6 = \{1.0, 0.01, 0.1, 0.1\}$	53	0.070067
$\theta_7 = \{0.1, 1.0, 0.01, 0.1\}$	110	0.064531
$\theta_8 = \{0.1, 0.01, 1.0, 0.1\}$	161	0.060987

Table 5: Application of the optimized computing budget allocation algorithm: desired confidence $P^\star = 85\%$, $N_{max} = 1000$, $N_{runs} = 897$, and the selected best ranking design is θ_4 with $APCS = 85.03\%$. $APCS$ as a function of N_{runs} for the listed designs is presented in Figure 9.

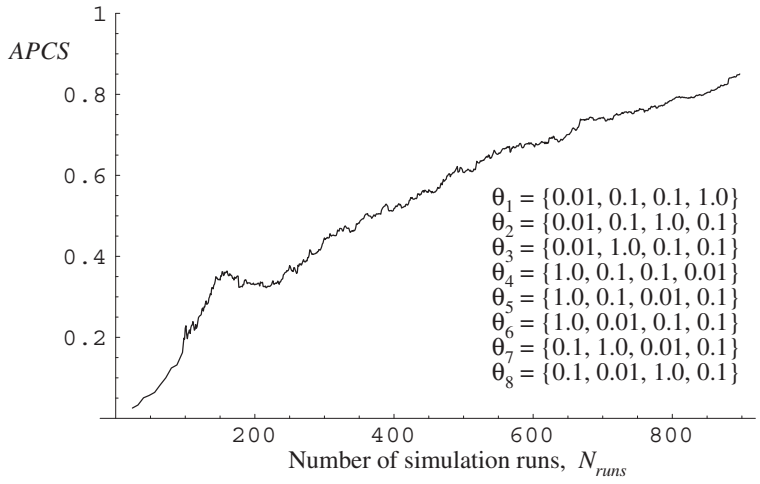

Figure 9: Achieved APCS as a function of total number of simulation runs. Tracking $T = 5$ targets, $\lambda = 1.4$; $\theta_b = \theta_4$ with $APCS = 85.03\%$. Total number of runs was $N_{runs} = 897$.

$T = 5$, $N_s = 3$		
$P^\star = 85\%$		$N_{runs} = 538$
$APCS = 87.98\%$		$\theta_b = \theta_4$
design	n_i	RMS_i
$\theta_1 = \{0.01, 0.1, 1.0\}$	23	0.146661
$\theta_2 = \{0.01, 1.0, 0.1\}$	19	0.147113
$\theta_3 = \{0.1, 0.01, 1.0\}$	117	0.104865
$\boldsymbol{\theta_4 = \{0.1, 1.0, 0.01\}}$	**215**	**0.092072**
$\theta_5 = \{1.0, 0.01, 0.1\}$	19	0.167626
$\theta_6 = \{1.0, 0.1, 0.01\}$	145	0.102838

Table 6: Application of the optimized computing budget allocation algorithm: desired confidence $P^\star = 85\%$, $N_{max} = 1000$, $N_{runs} = 538$, and the selected best ranking design is θ_4 with $APCS = 87.98\%$.

design tends to increase the probability of correct selection the most. Therefore in general, in the optimized computing budget allocation algorithm the computing budget is allocated more to the best designs. Fewer runs are spent on the worst performing designs, which results in reduction of the computation time compared to the uniform computing budget allocation.

6 Conclusion

In this paper we addressed the problem of ranking and selection for stochastic processes, such as target tracking algorithms, where variance is the performance metric, as opposed to the problem of ranking and selection of process means which has been frequently addressed in the literature. We present a method to minimize simulation time, yet to achieve a desirable confidence of the obtained results by applying ordinal optimization and computing budget allocation ideas and techniques, while taking into account statistical properties of the variance.

The developed optimized computing budget allocation algorithm (OCBA) method was applied to evaluate the sequential multi-sensor joint probabilistic data association algorithm, where we searched for the best order of processing sensor information when given non-identical sensors, so that the root-mean square position error performance metric is minimized. To evaluate the performance of different sensor orderings efficiently, the developed OCBA technique attempts to minimize simulation time, that is, achieves a predefined confidence level within a defined computation budget, or returns the confidence level of the achieved results if the computation budget must be exhausted. Results that we obtained with high confidence levels and in significantly reduced simulation times confirm the findings from our previous research (where we considered only two sensors) that processing the best available sensor the last in the sequential multi-sensor data association algorithm produces the smallest root-mean square position error on average.

7 Acknowledgements

This work was supported in part by the Office of Naval Research (Young Investigator Award Grant N00014-97-1-0642 and Grant N00014-02-1-0136) and a University of Colorado Faculty Fellowship. The authors would also like to thank Professor Y. C. Ho of Harvard University for motivating this research.

References

[1] M. Abramowitz and I. A. Stegun, Eds, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables,* Dover Publications, New York, 1972.

[2] Y. Bar-Shalom and T. Fortmann, *Tracking and Data Association*, Academic Press Inc., 1988.

[3] Y. Bar-Shalom and E. Tse, "Tracking in a Cluttered Environment With Probabilistic Data Association," *Automatica*, Vol. 11, pp. 451-460, Pergamon Press, 1975.

[4] C. G. Cassandras, L. Dai, and C. G. Panayiotou, "Ordinal Optimization for a Class of Deterministic and Stochastic Discrete Resource Allocation Problems," *IEEE Trans. Automatic Control*, Vol. 43, No. 7, pp. 881-900, July 1998.

[5] C. H. Chen, "A Lower Bound for the Correct Subset-Selection Probability and Its Application to Discrete-Event System Simulations," *IEEE Trans. Automatic Control*, Vol. 41, No. 8, pp. 1227-1231, August 1996.

[6] C. H. Chen, H. C. Chen, and L. Dai, "A Gradient Approach for Smartly Allocating Computing Budget for Discrete Event Simulation," *Proc. of the 1996 Winter Simulation Conference*, pp. 398-405, 1996.

[7] C. H. Chen, S. D. Wu, and L. Dai, "Ordinal Comparison of Heuristic Algorithms Using Stochastic Optimization," *IEEE Trans. Robotics and Automation*, Vol. 15, No. 8, pp. 44–56, February 1999.

[8] H. C. Chen, C. H. Chen, and E. Yucesan, "Computing Efforts Allocation for Ordinal Optimization and Discrete Event Simulation," *IEEE Trans. Automatic Control*, Vol. 45, No. 5, pp. 960-964, May 2000.

[9] T. E. Fortmann, Y. Bar-Shalom, M. Scheffe, and S. Gelfand, "Detection Thresholds for Tracking in Clutter – A Connection Between Estimation and Signal Processing," *IEEE Trans. Automatic Control*, Vol. 30, No. 3, pp. 221-229, March 1985.

[10] T. E. Fortmann, Y. Bar-Shalom, and M. Scheffe, "Sonar Tracking of Multiple Targets Using Joint Probabilistic Data Association," *IEEE Journal of Oceanic Engineering*, Vol. 8, No. 3, pp. 173-183, July 1983.

[11] C. W. Frei and L. Y. Pao, "Alternatives to Monte-Carlo Simulation Evaluations of Two Multisensor Fusion Algorithms," *Automatica*, Vol. 34, No. 1, pp. 103-110, Pergamon Press, 1998.

[12] S. S. Gupta and M. Sobel, "On Selecting a Subset Containing the Population with the Smallest Variance," *Biometrika*, Vol. 49, Issue 3/4, pp. 495-507, Dec. 1962.

[13] Y. C. Ho, "An Explanation of Ordinal Optimization: Soft Computing for Hard Problems," *Information Sciences*, Vol. 113, pp. 169-172, 1999.

[14] Y. C. Ho, C. G. Cassandras, C. H. Chen, and L. Dai, "Ordinal Optimisation and Simulation," *Journal of the Operational Research Society*, Vol. 51, No. 4, pp. 490-500, 2000.

[15] P. G. Hoel, *Introduction to Mathematical Statistics*, John Wiley, 1962.

[16] T. W. E. Lau and Y. C. Ho, "Universal Alignment Probabilities and Subset Selection for Ordinal Optimization," *Journal of Optimization Theory and Applications*, Vol. 93, No. 3, pp. 455-489, June 1997.

[17] L. Y. Pao, "Centralized Multi-sensor Fusion Algorithms for Tracking Applications," *Control Eng. Practice*, Vol. 2, No. 5, pp. 875-887, 1994.

[18] L. Y. Pao and L. Trailović, "Optimal Order of Processing Sensor Information in Sequential Multi-sensor Fusion Algorithms," *IEEE Trans. Automatic Control*, Vol. 45, No. 8, pp. 1532–1536, Aug. 2000.

[19] N. M. Temme, "The asymptotic expansions of the incomplete gamma functions," *SIAM J. Math. Anal.*, 10, pp. 239-253, 1979.

[20] S. B. Vardeman, *Statistics for Engineering Problem Solving,* IEEE Press, 1994.