SENSOR MANAGEMENT AND PROVISIONING FOR MULTIPLE TARGET RADAR TRACKING SYSTEMS

Gregory Newstadt Student Member, IEEE, and Alfred O. Hero III, Fellow, IEEE

Dept. of EECS, University of Michigan, Ann Arbor MI 48109-2122, USA
{newstage.hero}@umich.edu

ABSTRACT

System provisioning is the problem of determining the number of resources required to accomplish a complicated system level task, e.g., tracking or discriminating between $N$ targets. This is a central problem in multi-target tracking with synthetic aperture radars where the number of targets can easily exceed the available resources. This paper treats the following conservative sensor provisioning problem: dynamically assign $R$ platforms to process $N$ moving targets in a way that guarantees that the radar maintains track on all targets. We propose a solution to this problem that guarantees a prescribed level of system performance, e.g., multiple target detection and position uncertainty levels, regardless of the scenario. The operational context of the paper is computational provisioning in synthetic aperture radar (SAR) that dynamically assigns different computers to tracking different targets.

Index Terms— sensor management provisioning, prioritized longest queue provisioning, guaranteed uncertainty management, radar tracking, synthetic aperture radar

I. INTRODUCTION

The ability to track multiple targets over a large field of view (FOV) is an integral component of many applications, including traffic monitoring and anomalous behavior detection, among others. Synthetic aperture radars provide the capability to produce high resolution imagery that is robust to environmental conditions (weather, lighting, etc.). Previous work [1] has demonstrated that efficient strategies exist for tracking multiple targets given images formed from the SAR phase histories. However, the ability to do real-time tracking of targets with SAR is often limited by the computational demands of the image formation process.

This work describes a general approach to system provisioning for multiple ‘sensor’ systems that uses the guaranteed uncertainty management (GUM) philosophy. In this paper, we focus on the problem of managing computational resources, where the ‘sensors’ are the CPUs used to form the SAR images of interest from streaming radar samples. By system provisioning we mean using physical models for target detection and estimation to specify fundamental limits on performance (system stability, track entropy, occupancy rate) for a given provisioning of the system (number of CPUs, maximum number of FLOPs, desired standard errors). We have chosen to focus on the computational problem since real-time adaptive SAR sensors do not currently exist. However, it should be noted that the methods developed in this paper can be applied to a wide variety of applications (including managing physical sensors).

The GUM approach is more conservative than standard stochastic scheduling approaches to radar provisioning. In particular, it carries strict and absolute guarantees on the probability of loss of track of the system. This is in contrast to average performance guarantees that have been previously adopted [2] for similar applications. By using this strict performance approach, the sensor management problem becomes non-stochastic and leads to strong results that could not easily be obtained in the less stringent stochastic scheduling context.

The rest of the paper is organized as follows. In section II, we present the target and system models used throughout the paper. Section III develops the theory for guaranteed uncertainty management for a track-only radar including stability conditions and radar provisioning for multiple targets. In section IV, this theory is extended to a multi-purpose system that engages in tracking and other activities such as discrimination and search. Finally, section V presents a numerical example for typical radar parameters.

II. TARGET AND SYSTEM MODELS

II-A. System Model

Assume that streaming samples from a single SAR sensor are available from an X-band sensor with standard parameters ($f_0 \approx 10$ GHz, $BW \approx 500$ MHz, $\tau_{FPRI} \approx 10^{-3}$). Without loss of generality, we will assume that the radar platform travels in the $x$-direction.

We are interested in the computational burden associated with standard image formation from the radar samples using back-projection [3]. The number of FLOPs associated with this process is proportional to the number of radar samples $N_p$ and the number of pixels in the formed image, also proportional to $N_p$. The required time to detect and/or track within a target cell is then

$$T = \kappa N_p^2, \quad \kappa = \alpha_{\text{radar}} \tau_{\text{CPU}}$$

(1)

where $\tau_{\text{CPU}}$ is the number of seconds/FLOP associated with the CPU and $\alpha_{\text{radar}}$ is the number of FLOPs/$N_p^2$ that is dependent on the radar. For concreteness in this work, we assume that $\kappa \approx 3 \times 10^{-7}$ using a 2.8 GHz CPU.

II-B. Target Model

Assume that at time 0 a target is detected in a radar cell

$$C_0 = \{z = (x, y) : -\sigma_x \leq x - \hat{x} \leq \sigma_x, -\sigma_y \leq y - \hat{y} \leq \sigma_y\}$$

(2)

where $\hat{z} = [\hat{x}, \hat{y}]$ is the center position of the cell. From a radar signal processing algorithm, an estimate $(\hat{x}, \hat{y}, \hat{v}_x, \hat{v}_y, \hat{a}_x, \hat{a}_y)$ of target positions and velocities is extracted, along with a set of standard errors $(\sigma_x, \sigma_y, \sigma_{vx}, \sigma_{vy}, \sigma_{ax}, \sigma_{ay})$. This could be the output of a Kalman filter, sigma tracker, particle filter or other common tracking algorithm. From these estimates and standard errors a confidence region for $x, y, v_x, v_y$ having coverage probability of at least $1 - \varepsilon_T$ can be specified. In particular, assume that

$$\begin{align*}
[x - \sigma_x, \hat{x} + \sigma_x] & \times \left[\hat{y} - \sigma_y, \hat{y} + \sigma_y\right] \\
[\hat{v}_x - \sigma_{vx}, \hat{v}_x + \sigma_{vx}] & \times \left[\hat{v}_y - \sigma_{vy}, \hat{v}_y + \sigma_{vy}\right] \\
[\hat{a}_x - \sigma_{ax}, \hat{a}_x + \sigma_{ax}] & \times \left[\hat{a}_y - \sigma_{ay}, \hat{a}_y + \sigma_{ay}\right]
\end{align*}$$

(3)
is such a region. With probability no less than $1 - \varepsilon_\tau$, after an elapsed time of $\tau$ seconds from the last revisit of the target, the above confidence region will map to the union of the uncontrollable number of segments, which can be described by the set

$$C_\tau = \{(x,y) : -\varepsilon_\tau(\tau) \leq x - f_x(\tau) \leq \varepsilon_\tau(\tau),$$

$$-\varepsilon_\tau(\tau) \leq y - f_y(\tau) \leq \varepsilon_\tau(\tau)\}$$

where $\varepsilon_\tau(\tau) = \sigma_j + \sigma_y(t) + \sigma_x(t)^2/2$ and $\bar{f}_j(\tau) = \bar{v}_j(\tau) + \bar{a}_j(\tau)^2/2$ for $x = x, y$. See Fig. 1 for illustration. The area of this region is $|C_\tau| = \delta_\tau$.

In SAR tracking non-zero velocities can cause errors in the cross-range ($xy$) direction. Note that these errors will depend only on the standard errors ($\sigma_x, \sigma_y, \sigma_a, \sigma_y$), since images can be focused to $\bar{v}_x, \bar{v}_y, \bar{a}_x, \bar{a}_y$ with no additional computational cost. For a maximum error of $\delta_\tau$, this augmented region and its associated area is

$$h(C_\tau, \delta_\tau) = \{(x,y) : (u,v) \in C_\tau, x = u + [-\delta_\tau, \delta_\tau], y = v\}$$

where $h(C_\tau, \delta_\tau) = |\delta_\tau + h(C_\tau)| - 1$ is the growth of the confidence region. Note that $\delta_\tau$ does not depend on $\tau$, but only on the target's trajectory ($v_x, v_y, a_x, a_y$), which is arbitrary for any target, and the number of pulses, $N_p$, which is fixed by the user. Moreover, it is assumed that a target's state can be resolved as long as it remains within the neighborhood of the radar cell. The size of the radar cell is a system-dependent quantity that may differ as a function of radar operating mode (stripmap vs. spotlight SAR) or the size of the radar beamwidth, among other parameters. We define $T_{MAX}$ as an upper bound on the target revisit time that guarantees that the target can be resolved.

For $R$ CPUs and $N$ targets, let $q_{r,n}(\tau)$ denote the load on the $r$-th CPU to revisit and update the $n$-th target after an elapsed time of $\tau$:.

$$q_{r,n}(\tau) = \kappa N_p^2(r,n)\gamma_{r,n}(\tau; \delta_\tau),$$

where $\gamma_{r,n}$ and $N_p(r,n)$ are analogously defined as CPU and target dependent quantities that guarantee the performance criteria.

### III. GUARANTEED UNCERTAINTY MANAGEMENT

The problem of utilizing available CPUs in an optimal fashion to detect and track targets falls in the framework of dynamic scheduling of multiple servers to multiple queues (targets) [5], [6]. The sensor manager must assign CPUs to queues of target-revisit jobs in queues that grow as time elapses. Each job may have different service requirements. Generally, solving for the optimal allocation of servers to queues is a difficult, if not intractable, problem. However, several sub-optimal strategies have been proposed. A suboptimal prioritized longest queue (PLQ) strategy is to assign free servers to the longest queues, where each queue is processed by the server that is best matched to the service requirements. The following implementation of this strategy is the ‘largest weighted queue length’ policy proposed in [6] for heterogeneous multiqueuing systems. Let $N \subseteq \{1, 2, \ldots, N\}$ be the number of target tracks not in the process of being revisited.

**Prioritized longest queue (PLQ) sensor scheduling policy:**

When a CPU $r$ is unoccupied and available for assignment to updating a target track then either

1. **idle** the CPU if all target tracks are in process of being revisited ($N$ is empty).
2. **deploy** the CPU on the target $n \in N$ that maximizes the weighted service time $\max_{n \in N} q_{r,n}(\tau_n)$, where $\tau_n$ is the elapsed time since the last revisit of target $n$.

### III-A. Balance equations guaranteeing system stability

Balance equations for stable operation of the system are equations that guarantee that at the time of revisit of a target its service load has not grown larger than it was at the previous revisit. With a single CPU, we drop the index $r$ from $q_{r,n}(\tau)$. Let $q^{(n)}(\tau)$ be the service load to the $n$-th target chosen according to the PLQ policy. For $n = 1, 2, \ldots, N$, we have

$$q^{(n)}(\tau) = \max_{j \in N(n)} q_j(q^{(n-1)}(\tau) + \tau),$$

where $q^{(0)}(\tau) = 0$. To simplify notation, we assume that the targets have been ranked in decreasing order of service load, so that

$$\arg \max_{j \in N(n)} q_j(q^{(n-1)}(\tau) + \tau) = n,$$

and $q^{(n)}(\tau) = q_{n}(q^{(n-1)}(\tau) + \tau)$. Next define the system loading function

$$Q^{(N)}(\tau) = \sum_{i=1}^{N} q^{(i)}(\tau),$$

which is stable when $Q^{(N)}(\tau) < \tau$ (critically when $Q^{(N)}(\tau) = \tau$). If a solution exists, let $\tau = \tau^*$ be the solution of the balance equation

$$Q^{(N)}(\tau) = \tau.$$

**Proposition 1:** For a single CPU tracking $N$ targets the PLQ policy is stable, in the sense that the system maintains bounded tracking errors, if the following conditions hold:

1. a solution to (12) exists;
2. the revisit rate is at least $1/\tau^*$;
3. The target can be resolved, so that $\tau^* < T_{MAX}$.

The value $\tau^*$ can be interpreted as the steady state total time required for the CPU to cycle through a complete sequence of target revisits. The stability result of Proposition 1 is tight in the sense that the system becomes unstable if Conditions 1 and 2 are not satisfied. When stability of the PLQ policy is guaranteed, we have a tight bound on the associated tracking error.

**Corollary 1:** If the system is stable in the sense of Proposition 1, then the track uncertainty region of the $i$-th target will never exceed $H^i(i) = \ln |C_{\tau^i}(i)|$.
The proof of the above proposition is straightforward but we do not provide details here. The full proof relies on the fact that \( q^{(b)}(\tau) \) is monotonically increasing in \( \tau \). We then use mathematical induction to obtain equations (9) as the time required to service the targets, and apply standard load balancing condition of optimal scheduling theory to obtain (12).

III-B. A simple slope criterion for stability

The system load function \( Q^{(N)}(\tau) \) defined in (11) is zero at \( \tau = 0 \) and is smooth, differentiable, and monotonically increasing. Thus a necessary condition for the balance equation (12) to have a solution is that its derivative be less than or equal to 1 at the point \( \tau = 0 \). By induction the derivative \( [Q^{(N)}(\tau)](0) = dQ^{(N)}(\tau)/d\tau|_{\tau=0} \) can be shown to be of the form:

\[
\frac{[Q^{(N)}(\tau)](0)}{q_0} = \sum_{j=1}^{N} \sum_{k=1}^{j} \prod_{i=N-k+1}^{j} (q_i) \leq \sum_{j=1}^{N} \sum_{k=1}^{j} (q_0)^k,
\]

where we have defined \( q_0 = \max_i q_i(0) \). If \( \min_i q_i(0) > 1 \), then necessarily \([Q^{(N)}(\tau)](0) > 1\) so that \( Q^{(N)}(\tau) > \tau \) and the system is unstable. If \( q_0 < 1 \), then the system may be stable. To obtain closed form results we will derive sufficient conditions on \( N \) that guarantee stability by using the upper bound on the right of (13) instead of the exact expression in the middle of (13). This upper bound is attained when all service load functions are identical, \( q_i(0) = q_j(0) \) in which the conditions derived below will also be necessary. Therefore, the conditions will be tight for a worst case scenario but will be more stringent than might be required for a typical scenario. As \( q_0 \geq 0 \), the series summation formula applied to the right hand side of (13) gives the following proposition:

**Proposition 2**: A solution \( \tau^* \) to the balance equation (12) exists if and only if

\[
\frac{[Q^{(N)}(\tau)](0)}{q_0} = \sum_{j=1}^{N} \sum_{k=1}^{j} \prod_{i=N-k+1}^{j} \left(1 - [q_i(0)] \right) < 1
\]

Note that for our scenario, we can obtain \( q_0 \) by differentiating (6) plugging into (7), and evaluating at \( \tau = 0 \), yielding

\[
q_0' = \kappa N^2 \sigma_x \sigma_y^{-1} \sigma_x + \sigma_y (\sigma_x + \delta_d)
\]

Define \( N_{max} \) as the maximum value of \( N \) such that the inequality in Proposition 2 is satisfied. When the CPU is tasked to track \( N_{max} \) targets then the system will be stable (however, we must still verify that the associated \( \tau^* \) is such that condition 3 of Proposition 1 is satisfied). In the case \( N = N_{max} \) the CPU is fully utilized and operating at maximum efficiency. When \( q_0' \) is small, \( N_{max} \) can be found approximately as

\[
N_{max} = (1 - q_0')/q_0 + q_0/(1 - q_0')
\]

Furthermore, since \( 0 \leq 1 - [q_0] \leq 1 \), we can assert that if the number of targets \( N \) exceeds \( N_{max} \) in (16), then no solution to the balance equations exists and the system diverges.

III-C. Extension to multiple CPUs

When there are \( R \geq 1 \) CPUs to manage we can obtain stability conditions in a similar manner to the previous section. Define the ratio of targets per CPU \( b = \text{ceil}(N/R) \) as the smallest integer greater than \( N/R \). Define \( q(\tau) = \max_{n \leq R} q_{n,n}(\tau) \) and the service load, \( q^{(b)}(\tau) = q(2^{b-1}) + q_0(b) + \tau \). In analogy to the previous section, the system loading function is defined as \( Q^{(b)}(\tau) = \sum_0^{b} (q^{(b)}(\tau)) \). Stability conditions and slope conditions can be derived in a similar fashion to the previous section by replacing \( N \) with \( b = \text{ceil}(N/R) \). The details are omitted here, but can be found in [7].

III-D. Determining track-only system occupancy

We can use the Propositions to determine the efficiency of the system in terms of its occupancy rates, defined as one minus the proportion of time a CPU in the system is idle. We assume that the CPUs are scheduled under the PLQ policy. In steady state a stable system of \( R \) CPUs will be at maximum utilization when the system is critically stable. This occurs when there are approximately \( b = N/R \) targets per CPU where \( b \) is the solution to the equation

\[
\frac{q_0}{1 - q_0} \left(b - \frac{q_0}{1 - q_0} (1 - [q_0]) \right) = 1
\]

Define \( N_{max} = \text{floor}(b \cdot R) \). At this critically stable operating point of \( N_{max} \) targets, the CPUs are fully occupied performing just-in-time revisits of the targets. In this case the maximum service load that each target places on the system is \( Q^{(N_{max}/R)}(\tau^*) \) where \( \tau^* \) is the solution of \( Q^{(N_{max}/R)}(\tau) = \tau \). When the same system is assigned to track a fewer number \( N < N_{max} \) of targets, there will be idle time. We define the occupancy of the track-only system as \( \rho = \tau^*/\tau_c \), where \( \tau_c \) is the value of \( \tau \) that satisfies

\[
Q^{(N_{max}/R)}(\tau) = Q^{(N_{max}/R)}(\tau^*)
\]

The interpretation is that \( \tau_c \) is operating point of the system that results in the same loading for the underloaded system tracking \( N \) targets as the fully loaded system tracking \( N_{max} \) targets.

IV. MULTI-PURPOSE SYSTEM PROVISIONING

Finally we turn to scenarios when the system may be engaged in other tasks in addition to tracking. From a computational standpoint, this could be as basic as time needed for transfer of data and communication. At a more abstract level, tasks could include discrimination of targets and/or wide area search for new targets. This is handled by building in headroom into the track update stability equations. Let \( \Delta \) be the additional load in seconds spent after each revisit on tasks other than tracking. Consider the case of a single CPU and \( N \) targets. For a given \( \Delta \), the stability condition is that there must exist a solution, \( \tau = \tau^* \) such that

\[
Q^{(N)}(\tau, \Delta) + N \Delta = \tau
\]

where \( Q^{(N)}(\tau, \Delta) = \sum_{i=1}^{N} q_i(\tau, \Delta) \) and \( q_i(\tau, \Delta) = q_i(N - (N_{max} - 1)) \). Note that since the \( q_i \)'s are monotonically increasing, we have the bound

\[
Q^{(N)}(\tau, \Delta) \leq Q^{(N)}(\tau + \Delta)
\]

where \( Q^{(N)}(\tau) \) is the simpler function defined in (11). Therefore, for specified \( \Delta \), a sufficient condition for stability is that there exist a \( \tau = \tau^* \) such that

\[
Q^{(N)}(\tau + \Delta) + N \Delta = \tau
\]

Reexpressing this in terms of the variable \( u = \tau + \Delta \), we have the equivalent condition that there exist a solution \( u = u^* \) to

\[
Q^{(N)}(u) = u - (N + 1) \Delta
\]

IV-A. Load margin, excess capacity, and occupancy

The load margin represents the maximum additional load that can be accommodated by a tracking system that must perform joint operations such as tracking, detection, etc. The load margin \( \Delta_{max} \) is defined as the maximum value \( \Delta \) for which a solution \( u \) to (22) exists. When there are \( N \) targets and the multi-purpose system spends \( \Delta \leq \Delta_{max} \) seconds per update performing other tasks we define the excess capacity

\[
\Delta_{excess}(\Delta) = 1 - \Delta/\Delta_{max}
\]

Likewise, we define the multi-purpose system occupancy as

\[
r(\Delta) = 1 - [\Delta_{max} - \Delta]/Q^{(N)}(u^*)
\]
VI. CONCLUSIONS

This paper has proposed a conservative approach to sensor resource management for multiple target tracking subject to typical computational resource constraints. The approach requires finding solutions to load balance equations that guarantee system stability. These solutions yield the minimal system requirements for provisioning radars. The solutions guarantee stable tracking with prescribed level of statistical confidence. The provisioning results given here are conservative and specify the system requirements, steady state occupancy, revisit times, and track entropy in terms of the PQL sensor scheduling policy. The PQL policy will always perform at least as well as the performance predictions we provide. One can expect considerably better performance of the system than these predictions for typical scenarios, although there exists a scenario (namely, all targets are equally difficult to track) where the predictions are exact. Less stringent provisioning requirements might be explored using a stochastic optimization.

VII. REFERENCES