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.

This research was supported in part by AFRL under ATR Center grant FA8650-07-D-1221-TO1 and by TechFinity Inc, under STTR topic MDA07-T005.

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, $\tau_{PRI} \approx 10^{-3}$). 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

$$T = \kappa N_p^2, \quad \kappa = \alpha_{radar}\tau_{CPU} \tag{1}$$

where τ_{CPU} is the number of seconds/FLOP associated with the CPU and α_{radar} is the number of FLOPs/N_p^2 that is dependent on the radar. For concreteness in this work, we assume that $\kappa \approx 3e10^{-7}$ using a 2.8 GHz CPU.

II-B. Target Model

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

$$\mathcal{C}_0 = \{z = (x, y) : -\sigma_x \leqslant x - \bar{x} \leqslant \sigma_x, -\sigma_y \leqslant y - \bar{y} \leqslant \sigma_y\} \tag{2}$$

where $\bar{z} = [\bar{x}, \bar{y}]$ 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

$$\begin{aligned} [\hat{x} - \sigma_x, \hat{x} + \sigma_x] \quad &\times \quad [\hat{y} - \sigma_y, \hat{y} + \sigma_y] \times \\ [\hat{v}_x - \sigma_{vx}, \hat{v}_x + \sigma_{vx}] \quad &\times \quad [\hat{v}_y - \sigma_{vy}, \hat{v}_y + \sigma_{vy}] \\ [\hat{a}_x - \sigma_{ax}, \hat{a}_x + \sigma_{ax}] \quad &\times \quad [\hat{a}_y - \sigma_{ay}, \hat{a}_y + \sigma_{ay}] \end{aligned} \tag{3}$$

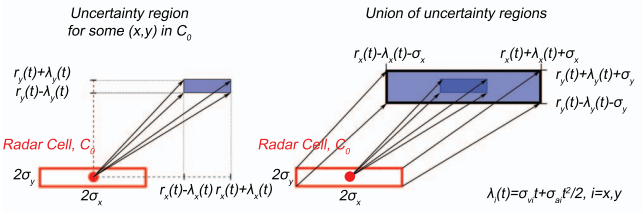

Fig. 1. The small radar cell contains a target with high certainty immediately after revisit. If the target trajectory is (v_x, v_y, a_x, a_y) with confidence $(\sigma_{vx}, \sigma_{vy}, \sigma_{ax}, \sigma_{ay})$, then we can be confident that a target at the center of the radar cell (left) will lie in the rectangular region after an elapsed time of τ secs. When the target can lie anywhere in the radar cell, then we can only be confident that the target will lie in the union of all induced rectangular regions (right).

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

$$\begin{aligned}\mathcal{C}_\tau = \{(x,y) : &-\varepsilon_x(\tau) \leqslant x - \hat{r}_x(\tau) \leqslant \varepsilon_x(\tau), \\ &-\varepsilon_y(\tau) \leqslant y - \hat{r}_j(\tau) \leqslant \varepsilon_y(\tau)\}\end{aligned} \quad (4)$$

where $\varepsilon_j(\tau) = \sigma_j + \sigma_{vj}\tau + \sigma_{aj}\tau^2/2$ and $\hat{r}_j(\tau) = \hat{j} + \hat{v}_j\tau + \hat{a}_j\tau^2/2$ for $j = x, y$. See Fig. 1 for illustration. The area of this region is $|\mathcal{C}_\tau| = 4\epsilon_x(\tau)\epsilon_y(\tau)$.

In SAR tracking non-zero velocities can cause errors in the cross-range (x-) direction [4]. Note that these errors will depend only on the standard errors $(\sigma_{vx}, \sigma_{vy}, \sigma_{ax}, \sigma_{ay})$, since images can be focused to $\hat{v}_x, \hat{v}_y, \hat{a}_x, \hat{a}_y$ with no additional computational cost. For a maximum error of δ_d, this augmented region and its associated area is

$$h(\mathcal{C}_\tau, \delta_d) = \{(x,y) : (u,v) \in \mathcal{C}_\tau, x \in u + [-\delta_d, \delta_d], y = v\} \quad (5)$$

$$|h(\mathcal{C}_\tau, \delta_d)| = 4(\varepsilon_x(\tau) + \delta_d)\varepsilon_y(\tau) \quad (6)$$

We are interested in resolving targets within a cell size, denoted $|\mathcal{C}_0|$. Due to real-time constraints, we assume that only streaming data is available. Thus, after τ seconds since the last revist, the system will be occupied by the task of forming subimages in $h(\mathcal{C}_\tau, \delta_d)$ to reduce uncertainty on the target parameters back down to a $1-\varepsilon_T$ confidence region of size $|\mathcal{C}_0|$. Thus, the load on the CPU is given by

$$q(\tau) = \kappa N_p^2 \gamma(\tau; \delta_d), \quad (7)$$

where $\gamma(\tau; \delta_d) = \frac{|h(\mathcal{C}_\tau, \delta_d)|}{|h(\mathcal{C}_0, \delta_d)|} - 1$ is the growth of the confidence region. Note that δ_d does not depend on τ, but only on the target's trajectory (v_x, v_y, a_x, a_y), which is arbitary 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 (in seconds) on the r-th CPU to revisit and update the n-th target after an elapsed time of τ:

$$q_{r,n}(\tau) = \kappa N_p^2(r,n)\gamma_{r,n}(\tau; \delta_d), \quad (8)$$

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 multiqueueing systems. Let $\mathcal{N} \subset \{1, 2, \ldots, N\}$ 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 ($\mathcal{N}$ is empty).
2) *deploy* the CPU on the target $n \in \mathcal{N}$ that maximizes the weighted service time $\max_{n\in\mathcal{N}} q_{r,n}(\tau_n)$, where τ_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

$$q^{(n)}(\tau) = \max_{j\in\mathcal{N}(n)} q_j(q^{(n-1)}(\tau) + \tau), \quad (9)$$

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

$$\arg\max_{j\in\mathcal{N}(n)} q_j(q^{(n-1)}(\tau) + \tau) = n, \quad (10)$$

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

$$Q^{(N)}(\tau) = \sum_{i=1}^{N} q^{(i)}(\tau), \quad (11)$$

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

$$Q^{(N)}(\tau) = \tau. \quad (12)$$

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 $\tau^* \ll T_{MAX}$.

The value τ^* 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 $H^*(i) = \ln|\mathcal{C}_{\tau^*}(i)|$.

The proof of the above proposition is straightforward but we do not provide details here. The full proof relies on the fact that $q^{(i)}(\tau)$ is monotonic increasing in τ . 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 monotonic 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 $[Q^{(N)}]'(0) = dQ^{(N)}(\tau)/d\tau|_{\tau=0}$ can be shown to be of the form:

$$[Q^{(N)}]'(0) = \sum_{j=1}^{N}\sum_{k=1}^{j}\prod_{i=N-k+1}^{N} q_i'(0) \leqslant \sum_{j=1}^{N}\sum_{k=1}^{j}(q_0')^k, \qquad (13)$$

where we have defined $q_0' = \max_i q_i'(0)$. If $\min_i q_i'(0) > 1$, then necessarily $[Q^{(N)}]' > 1$ 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 $q_0' \geqslant 0$, the series summation formula applied to the right hand side of (13) gives the following proposition:

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

$$[Q^{(N)}]'(0) = \frac{q_0'}{1-q_0'}\left(N - \frac{q_0'}{1-q_0'}(1-[q_0']^N)\right) < 1 \qquad (14)$$

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

$$q_0' = \kappa N_p^2(\sigma_x\sigma_y)^{-1}[\sigma_{vx}\sigma_y + \sigma_{vy}(\sigma_x + \delta_d)] \qquad (15)$$

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 τ^* 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

$$N_{max} = (1-q_0')/q_0' + q_0'/(1-q_0') \qquad (16)$$

Furthermore, since $0 \leqslant 1 - [q_0']^N \leqslant 1$, 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 $R > 1$ 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 $q(\tau) = \max_{n,r} q_{r,n}(\tau)$ and the service load, $q^{(b)}(\tau) = q(q^{(b-1)}(\tau) + \tau)$. In analogy to the previous section, the system loading function is defined as $Q^{(b)}(\tau) = \sum_{i=1}^{b} q^{(i)}(\tau)$. 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

$$\frac{q_0'}{1-q_0'}\left(b - \frac{q_0'}{1-q_0'}(1-[q_0']^b)\right) = 1 \qquad (17)$$

Define $N_{max} = \text{floor}(b^*R)$. 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 τ^* 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 $\rho = \tau^*/\tau_\varepsilon$, where τ_ε is the value of τ that satisfies

$$Q^{(N/R)}(\tau) = Q^{(N_{max}/R)}(\tau^*). \qquad (18)$$

The interpretation is that τ_ε 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 Δ 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 Δ, the stability condition is that there must exist a solution, $\tau = \tau^*$ such that

$$Q^{(N)}(\tau, \Delta) + N\Delta = \tau, \qquad (19)$$

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

$$Q^{(N)}(\tau, \Delta) \leqslant Q^{(N)}(\tau + \Delta), \qquad (20)$$

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

$$Q^{(N)}(\tau + \Delta) + N\Delta = \tau. \qquad (21)$$

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

$$Q^{(N)}(u) = u - (N+1)\Delta. \qquad (22)$$

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 Δ_{max} is defined as the maximum value Δ for which a solution u to (22) exists. When there are N targets and the multi-purpose system spends $\Delta \leqslant \Delta_{max}$ seconds per update performing other tasks we define the excess capacity

$$c_{excess}(\Delta) = 1 - \Delta/\Delta_{max}. \qquad (23)$$

Likewise, we define the multi-purpose system occupancy as

$$\rho(\Delta) = 1 - [(\Delta_{max} - \Delta)N]/Q^{(N)}(u^*). \qquad (24)$$

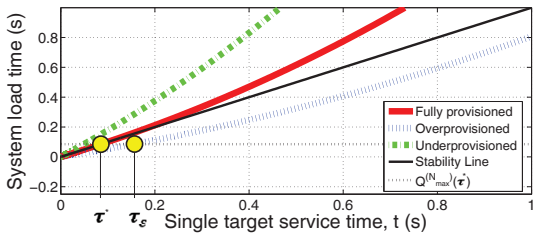

Fig. 2. This figure demonstrates various combinations of N/R. The diagonal line, called the stability boundary, separates two regions of operation. When the load curve is below the diagonal, track is maintained on all targets. Above the stability line, the system is unstable. The red line ($N/R = 17$) shows the fully provisioned case ($\rho = 100\%$). The blue dashed line ($N/R = 9$) is always below the stability line, showing an overprovisioned case where the system keeps all targets in track and has a lot of headroom to spare. The green line ($N/R = 30$) is always above the stability line, representing an underprovisioned case where the system is overwhelmed and tracks are lost. Yellow circles show the locations on the N/R=(17,9) curves with equal system loads.

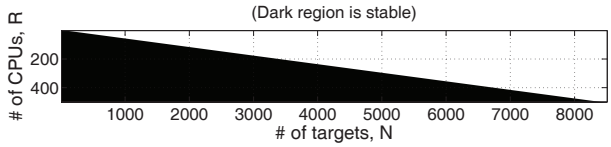

Fig. 3. The system provisioning matrix specifies stability region (dark) as a function of the numbers of radars and the number targets for track-only radar.

V. EXAMPLE APPLICATION

For specified standard errors on tracking accuracy, e.g., available from Kalman tracking covariance estimates, the above results can be used to generate tables and curves on the required number of CPUs, their revisit rates, and their occupancy, for tracking N targets with prescribed track error (entropy). For this scenario, we assume

1) A radar with parameters defined in Section II.
2) $N_p = 250$ corresponding to P_f=10^{-6}, P_d>0.99 for detecting a Swerling II target at a SNR=0 dB [8], Fig. 12.23.
3) Target cell given by (σ_x, σ_y)=$(6, 0.3)$ m.
4) Target trajectory, (v_x, v_y)=$(5, 5)$ m/s, (a_x, a_y)=$(0, 0)$ m/s^2, with std. errors $(\sigma_{vx}, \sigma_{vy})$=(1,1) m/s $(\sigma_{ax}, \sigma_{ay})$=(1,1) m/s^2.
5) $T_{MAX} = 1$ seconds.

V-A. Loading of track-only system

Figure 2 shows results for various numbers of R CPUs and N targets, such that $N/R = 9, 17, 30$, respectively. The figure provides a graphical view of the different stability regions as a function of revisit time, τ, and the target-to-CPU ratio. The different curves represent an overprovisioned system, a critically stable system, and an underprovisioned system, respectively.

Figure 3 provides a graphical representation of stability in track-only provisioning as a function of the number of N targets and R CPUs. The figure shows a matrix whose (i, j) entry is equal to 1 if i CPUs can track j targets stably and equal to 0 otherwise. The dark areas represent the stable operating region.

V-B. Multi-purpose system provisioning

Figure 4 illustrates a computation of the excess capacity, occupancy, and load margin for the same radar as in the previous section

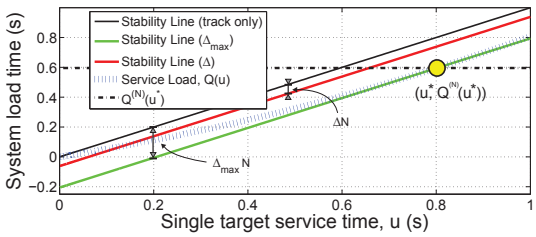

Fig. 4. System loading curves for computing occupancy and excess capacity for the multi-purpose radar tracking example. Fully loaded track-only radar load curve at top is shown for comparison.

but when it is tracking only 9 targets and can devote resources to other tasks. Unlike the case of 17 targets that only intersects the diagonal line $y(u) = u - \Delta$ when $\Delta = 0$, there is a substantial load margin for the case of 9 targets, $\Delta_{max} = 0.206/N$ secs. At this full utilization operating point the radar devotes approximately 11% of its time to tracking and the rest of its time to other tasks. The distance between the upper and lower diagonal lines $y(u) = u$ and $y(u) = u - \Delta_{max}N$ is 0.206 secs. If the actual load for other tasks was set to only $\Delta = 0.06/N$ secs, giving c_{excess} = 0.70 and an occupancy of $\rho(\Delta) = 0.76$, the system would be idle 24% of the time.

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

[1] G. Newstadt, E. Zelnio, L. Gorham, and A. Hero III, "Detection/tracking of moving targets with synthetic aperture radars," in *SPIE Conference Series*, vol. 7699, 2010, p. 16.

[2] A. Hero, D. Castan, D. Cochran, and K. Kastella, "Foundations and applications of sensor management," 2007.

[3] M. Soumekh, *Synthetic aperture radar signal processing with MATLAB algorithms*. Wiley, 1999, vol. 138.

[4] J. Fienup, "Detecting moving targets in SAR imagery by focusing," *Aerosp. Electron. Syst., IEEE Trans.*, vol. 37, no. 3, pp. 794–809, 2001.

[5] P. Brémaud, *Point processes and queues, martingale dynamics*. Springer, 1981.

[6] K. Wasserman, G. Michailidis, and N. Bambos, "Optimal processor allocation to differentiated job flows," *Performance Evaluation*, vol. 63, no. 1, pp. 1–14, 2006.

[7] A. O. H. III, "Sensor management provisioning for multiple target radar tracking systems," Univ. of Michigan, CSPL, Tech. Rep. 407, 2008.

[8] H. Meikle, *Modern radar systems*. Artech House Publishers, 2008.