DARPA Autonomous Negotiating Teams (ANTS) Program
Project Final Report of The Ohio State University
(Team: BAE (Sanders), Northeastern Univ.,
Stanford Research Inst. (SRI), and Ohio State University)

Design and Analysis of Stable Distributed Dynamic Scan Scheduling Policies[*]

Kevin M. Passino[†]

Dept. Electrical Engineering, The Ohio State University
2015 Neil Avenue, Columbus, OH 43210-1272

Abstract

The problem of how to dynamically schedule a receiver to detect multiple emitters is modeled as a sensor resource allocation problem. It is shown how to specify a variety of receiver-based dynamic scan scheduling strategies that dynamically focus a sensor in different frequency bands so as to develop and maintain good information about an emitter environment. We contrast dynamic and "fixed" scheduling policies, show how to incorporate emitter priorities, and we use simulation analyses to provide insights into the design and behavior of the policies. We explain how a "response surface methodology" can be used to design optimal dynamic scan schedulers. Moreover, we introduce the idea of using the "simultaneous perturbation stochastic approximation" method for optimal dynamic scan scheduler design when there are many emitters. Next, it is shown how to characterize and analyze stability properties (e.g., boundedness) of a variety of dynamic scan scheduling methods. We discuss a "universal stabilizing mechanism" for dynamic scan schedulers and propose two dynamic scan scheduling methods that exploit emitter environment information in order to try to optimize scheduler performance. Moreover, we provide some stability and simulation results on "distributed dynamic scan scheduling" where there are multiple receivers that try to coordinate their actions to maintain good information about an emitter environment. In this case the simulation results clearly show that there are advantages to "collaborative" distributed dynamic scan scheduling. The paper closes with a brief summary of the key ideas and a list of potential future directions.

[*]Version: 7/03/02. The section on distributed dynamic scan schedulers was completed by application of the results developed in collaboration with Mr. Alvaro Gil on distributed resource allocation for another application domain; the author gratefully acknowledges the contributions of his work via this collaboration.

[†]The author may be contacted at (614)-292-5716, fax: (614)-292-7596, `passino@ee.eng.ohio-state.edu`, `http://eewww.eng.ohio-state.edu/~passino`

Contents

1 The Scheduling Problem

First, based on the description given in [18] we provide a brief overview of the emitter and then receiver characteristics in order to establish our mathematical notation. Assume that we know that there are a fixed number of N emitters. For convenience we label them with positive integers so the set of emitters is given by

$$E = \{1, 2, \ldots, N\}$$

Suppose that we know the frequency band of each emitter. Let t denote time. Emitter $j \in E$ outputs pulses and suppose that the signal $e^j(t)$ is an envelope for such pulses in the sense that $e^j(t) = 0$ when the emitter is not illuminating and $e^j(t) = 1$ when it is. Given this, $e^j(t)$ is a square wave where the times t, such that $e^j(t) = 1$, represent the times when the emitter is illuminating. Let k_e be the index of the k_e^{th} illumination. Also, let

$$\tau_{ei}^j(k_e), j \in E, k_e = 1, 2, \ldots$$

denote the length of the *emitter illumination time* for the k_e^{th} illumination time by emitter $j \in E$. Next, let

$$\tau_{eip}^j(k_e), j \in E, k_e = 1, 2, \ldots$$

denote the corresponding *emitter illumination period*. Generally, the values of $\tau_{ei}^j(k_e)$ and $\tau_{eip}^j(k_e)$ are not known; however, once an emitter is detected and identified, possibly considering a list of a priori emitters that are likely in the current setting, it may be possible to estimate their values. Note also, that with an aircraft-based receiver, the motion of the aircraft can change the apparent length of $\tau_{ei}^j(k_e)$ and hence $\tau_{eip}^j(k_e)$ each time k_e there is an illumination.

Next, we discuss the receiver. The receiver is aircraft-based and its job is to scan different frequency bands in search of emitters. It is assumed that detection of an emitter $j \in E$ requires that the receiver is tuned to the proper frequency band for some *fraction* of the time $\tau_{ei}^j(k_e)$ (e.g., as defined by receiving some fixed number of pulses from the emitter). The command to scan a specific frequency band at a particular time for a specific time period is specified via a "control description word" (CDW). Suppose that we use the time index k_r for the k_r^{th} CDW. Let

$$\tau_{rs}^i(k_r), i \in E, k_r = 1, 2, \ldots$$

denote the k_r^{th} time when the receiver begins its scan for an emitter in frequency band $i \in E$ (note that since we assume that we know which frequency band each emitter is in, we can use the label for the emitter to also indicate the frequency band of the emitter). Next, let

$$\tau_{rd}^i(k_r), i \in E, k_r = 1, 2, \ldots$$

be the *dwell time* for the k_r^{th} scan for emitter/frequency band $i \in E$ and

$$\tau_{rr}^i(k_r), i \in E, k_r = 1, 2, \ldots$$

be the *revisit time* for this case (i.e., the length of time until scan $k_r + 1$). Of course, you can define the revisit time in terms of $\tau_{rs}^i(k_r + 1)$ and $\tau_{rd}^i(k_r)$, but we explicitly define the length of the revisit time for convenience. Finally, note that it may take only a fraction of the emitter illumination time $\tau_{ei}^j(k_e)$ for the receiver to fully detect emitter $j \in E$. Hence, if more than one emitter is illuminating in one time period of sufficient length, it may be possible for the receiver to switch between multiple illuminating emitters during that period (perhaps many times).

The dynamic scan scheduling problem involves using past sensed information (e.g., times of detections) and possibly a priori information on emitter characteristics to choose at the k_r^{th} step the emitter/frequency to focus on and tune to (which we denote by $i^*(k_r)$), and the values of $\tau_{rd}^{i^*(k_r)}(k_r)$ and $\tau_{rr}^{i^*(k_r)}(k_r)$. Note that since the receiver platform is moving, and hence the emitter illumination times and periods change in an unpredictable fashion, it is likely that it will be necessary to adjust $\tau_{rd}^{i^*(k_r)}(k_r)$ and $\tau_{rr}^{i^*(k_r)}(k_r)$ at each step. Clearly, however, a priori information on emitter characteristics may provide good information on how to guess at these values.

2 Dynamic Scan Scheduling Policies

In this section we will define a variety of dynamic scan scheduling (DSS) policies and provide insights into their operation. Our focus is on the case where no special information is known about the frequency of the emitter illuminations, aside from assuming that the amount of time between emitter illuminations is bounded. While this ensures that our approaches are "robust" to emitter illumination timing changes or uncertainty (e.g., due to aircraft motion), it does not allow us to exploit information that is sometimes known about an emitter environment to improve performance (e.g., information from an "active emitter table" (AET)). On the other hand, the policies defined in this section are very easy to implement in real-time as the memory and processing requirements are quite minimal.

How do the approaches here relate to other scheduling problems discussed in the literature? Basically, for the scheduling problem described in [18], and building on basic ideas in sequencing and scheduling [1, 4, 5], the approach here expands on the work in [13] using the time-based policies that were first studied in [2] and utilizes a discrete-event system [3, 6] theoretic framework for stability analysis [9, 10] that is described in detail in [12, 11]. The work we do later in this paper on distributed dynamic scan scheduling, both in stability analysis and simulations, is an extension of the work done in these earlier cases, and it has applicability far beyond the present context in general dynamic resource allocation problems (e.g., in manufacturing and process control).

2.1 Scheduling Problem Model

To define a dynamic scan scheduling policy it is first necessary to define the scheduling problem. We do this here by defining a model of the scheduler that is on board the receiver, and the scheduling environment (e.g., how emitter characteristics influence scheduling).

2.1.1 Scheduler Input-Output Variables

First, let

$$T_i(t), i \in E, t \geq 0$$

denote the *last time at which the emitter of type j was detected* (this is defined by the instant t' when by focusing on emitter i we get $T_i(t') = 0$). Suppose that initially,

$$T_i(0) = 0, i \in E$$

so that we act as though initially we had simultaneously detected all the emitters, which is clearly physically impossible. Note, however, that this is a good initialization considering the fact that below our DSS policies will make decisions about which emitter to focus on based on the sizes of the $T_i(t)$, $i \in E$. Basically, for many policies this initialization indicates that at $t = 0$ there is no priority to seek one emitter rather than any other one. For many policies, an initialization with $T_i(0) > T_{i'}(0)$ for $i \neq i'$ would indicate an initial preference to first scan for the i^{th} emitter over emitter i'.

Note that if the receiver is not operating, then clearly

$$T_i(t) \to \infty, i \in E, t \to \infty$$

since it will never detect an emitter. The goal of the DSS is to try to avoid $T_i(t) \to \infty$ for any $i \in E$ and indeed it will try to keep the $T_i(t)$ values *as small as possible* since this represents that the receiver has recently detected each emitter and hence has good information about the emitters. It is assumed that each emitter will *persistently* periodically illuminate so that there is a finite amount of time between illuminations; this is assumed since if some emitter $i \in E$ only emits for a finite amount of time then at some point it will clearly be impossible to detect it again so that $T_i(t) \to \infty$ as $t \to \infty$.

Generally, we will view the schedulers as "controllers" that take as inputs the $T_i(t)$, $i \in E$, and choose a CDW that characterizes the type of scan that should be taken next. This is shown in Figure 1. The manner in which the scheduler senses emitters, computes "quality of service" (QoS) measures, and chooses CDWs will be explained when we define the DSSs below.

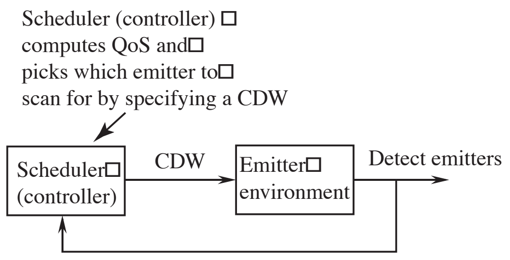

Scheduler (controller) computes QoS and picks which emitter to scan for by specifying a CDW
Scheduler (controller)
CDW
Emitter environment
Detect emitters

Figure 1: DSS scheduler viewed as a controller.

2.1.2 Delays Induced by the Receiver and Environment

Let $\delta(t) > 0$ denote a processing delay of the receiver that may represent the delay from several sources. For instance, it may be used to represent the amount of time that it takes for the receiver to switch from scanning in one frequency band for one emitter i to another frequency band for another emitter j, $j \neq i$. We will call this type of delay $\delta_{i,j}$ and assume it is a fixed known delay. For convenience, we will assume that these switching delays are all the same and will denote that value by $\delta_s = \delta_{i,j}$ for all $i, j \in E$. The variable $\delta(t)$ may also incorporate delays in being able to detect an emitter. For instance, each emitter has a basic frequency of illumination that is driven by a variety of characteristics including the rate of rotation of the radar. Suppose that for a known emitter type i there is some bound δ^i on the amount of time that it would take for the receiver to get the first pulse from the emitter if that was the only emitter (frequency band) that the receiver focused on and tuned to (clearly this would depend on the emitter illumination period). Notice that getting the first pulse does not correspond to achieving a detection of an emitter. Suppose that $\delta_e(t)$ denotes the delay incurred by the receiver in finding the first emitter pulse, from the time that it gets switched to focus on the emitter. Note that if we let

$$\bar{\delta} = \max_i \left\{\delta^i\right\}$$

then $\delta_e(t) \leq \bar{\delta}$. Let

$$\delta(t) = \delta_s + \delta_e(t)$$

For convenience, we let δ denote a constant that is the least upper bound on $\delta(t)$ so that $\delta(t) \leq \delta$ (i.e., we simply remove the time index to denote the least upper bound on the variable).

To summarize, when a CDW (or CDWs) is issued to focus on emitter i there is a delay to switch to the receiver to focus on it, and then there is an additional (time-varying) delay since the illumination might not yet have occurred (e.g., due to the radar rotating). This additional delay is shorter than $\bar{\delta}$. After these two types of delays occur we assume that the receiver knows something is in the frequency band that is being focused on (e.g., it may have detected a single pulse in the illumination, but it may not have fully detected the emitter).

2.1.3 Receiver Processing Rate

We will suppose that the receiver may take additional time to detect an emitter that it has not detected for a long period of time (i.e., we think of the receiver as having successively more difficult times finding an emitter that it has not found for longer periods of time since it, in a sense, becomes "desynchronized" with that emitter and cannot easily determine when it will emit). To quantify this characteristic we will use parameters

$$a_i, i \in E$$

where intuitively $1/a_i$ represents a "rate" at which the receiver processes information about emitters in order to detect them. These a_i parameters require further explanation. Consider the case where there is only one emitter ($N = 1$), named "emitter 1." Suppose that at some time t', the amount of time that has elapsed since the last time emitter 1 was detected is $T_1(t') > 0$ as shown in Figure 2.

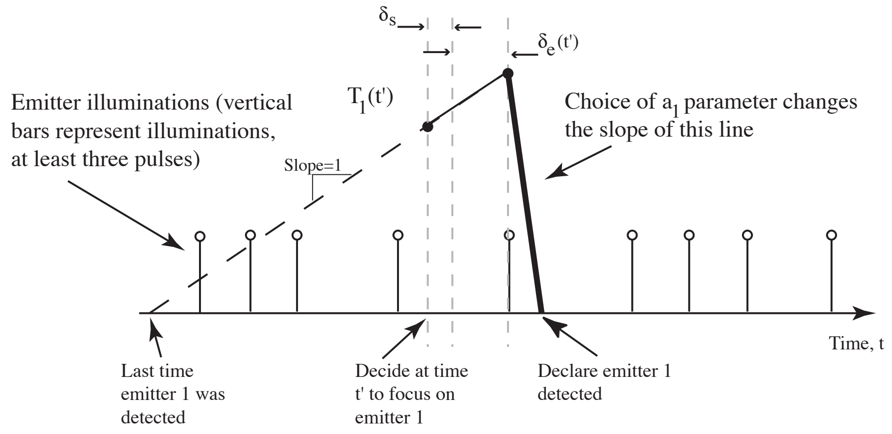

δ_s
$\delta_e(t')$
$T_1(t')$
Emitter illuminations (vertical bars represent illuminations, at least three pulses)
Slope=1
Choice of a_1 parameter changes the slope of this line
Time, t
Last time emitter 1 was detected
Decide at time t' to focus on emitter 1
Declare emitter 1 detected

Figure 2: Illustration of timing of receiver decision-making and emitter illuminations.

In Figure 2, at time $t' + \delta_s$ the receiver has switched its focus to emitter i. So, starting at $t' + \delta_s$ the receiver is looking for emitter i and before $t' + \delta_s + \delta^1$ we know that an emitter illumination will occur. Name the delay between achieving a switch in focus to the time where an emitter pulse is first found $\delta_e(t')$. Then, at time $t' + \delta_s + \delta_e(t')$ the receiver initiates the completion of the identification of emitter 1 (this is where the peak in T_1 occurs in Figure 2) and the amount of time that it takes to do that is dictated by the a_1 parameter (as you will see below, smaller values of a_1 correspond to it taking shorter amounts of time to fully detect the emitter). We declare emitter 1 "detected" at the time at which T_1 is decreased to zero. Next, we need to further clarify the meaning of the a_i parameters by explaining how they produce the slope of the bold line in Figure 2 and hence quantify how long it takes to detect an emitter. Also, we need to explain how the receiver chooses which emitter to focus on. To do this, we will introduce a specific policy and explain how to interpret the a_i, $i \in E$ parameters.

2.2 Scan for Emitter Ignored for the Longest Time

First, let D_{k_r} denote the time at which the DSS chooses a CDW (i.e., it is the *decision time* when it chooses which emitter to focus on), and suppose that $D_1 = 0$. A DSS policy that scans for the emitter that was ignored for the longest time makes choices of which emitter to focus on such that at D_{k_r} the DSS chooses to focus on emitter $i^*(k_r)$ such that

$$T_{i^*(k_r)}(D_{k_r}) \geq T_i(D_{k_r}), \forall i \in E \tag{1}$$

and focuses on it until it detects it (e.g., it issues a CDW that indicates to set the dwell time $\tau_{rd}^{i^*(k_r)}(k_r)$ long enough to detect the emitter or emits a sequence of CDWs such that detection is achieved).[1] If there is more than one maximizer, then the DSS will simply choose one of these at random.

2.2.1 Receiver Decision Times and Processing Rate

First, notice that the scan(s) for the emitter $i^*(k_r)$ start after some delay, and then it may take some additional (but finite time) to detect the emitter ($\delta_e(D_{k_r}) \leq \bar{\delta}$), and still more time based on how long it has been since the emitter was last detected (i.e., the effect of the a_i). Note also that while the CDWs are issued, and unsuccessful scans occur, the length of the time since the last detection is still increasing. Hence, the times when the DSS makes decisions are given by

$$D_{k_r+1} = D_{k_r} + \delta(D_{k_r}) + a_{i^*(k_r)} T_{i^*(k_r)}(D_{k_r}) + (D_{k_r+1} - D_{k_r})\, a_{i^*(k_r)} \tag{2}$$

[1] Clearly, either approach, continually issuing CDWs that indicate a short dwell time, or one CDW that indicates a long dwell time, is not practical since in the real system if there is a significant length of time during which you know that there will be no illumination by any emitter, then the receiver should go search in other frequency bands for other possible emitters. Here, we see that the "wasting" of the CDWs only makes sense since we assume that there are a fixed number of known emitters.

Here, the next decision point D_{k_r+1} is where the time when the detection of the last emitter that was focused on is detected and this formula gives the time D_{k_r+1} when the next decision will be made. The value of D_{k_r+1} is given by the sum of four terms. The first term is simply the last decision point D_{k_r}. The second term is the delay $\delta(D_{k_r})$ that is incurred due to the need to retune the receiver on a different frequency band, and possibly incorporating additional detection delay so that

$$\delta(D_{k_r}) = \delta_s + \delta_e(D_{k_r})$$

Third, the term $a_{i^*(k_r)} T_{i^*(k_r)}(D_{k_r})$ is the amount of time it takes to detect emitter $i^*(k_r)$ that arises due to the fact that we have not detected it for some time (note the proportionality—if it has not been detected for a long time, then it will take more time to find it and this represents that emitters that have not been detected for a long time become more difficult to detect). Finally, the fourth term quantifies that there is additional time needed to detect the emitter simply because during the time that the search for the emitter is occurring, even when it is focused on, the length of time since the last detection continues to increase (we do not consider an emitter $i^*(k_r)$ fully detected until $T_{i^*(k_r)}(D_{k_r+1}) = 0$).

Using simple algebra to rearrange Equation (2), we get

$$D_{k_r+1} = D_{k_r} + \frac{\delta(D_{k_r}) + a_{i^*(k_r)} T_{i^*(k_r)}(D_{k_r})}{1 - a_{i^*(k_r)}} \tag{3}$$

Notice that as expected the delay δ directly influences the rate at which CDWs can be issued. Also, however, this equation shows us that the length of time between decisions can be lengthened if a particular emitter has been ignored for too long due to the effects of the a_i parameters. In fact, using Equation (2) it is now possible to complete the explanation of Figure 2 and how to interpret the a_i parameters. What is the effect of the a_i parameters on how fast an emitter is detected? Notice that we incur the delay $\delta(t)$, and from Figure 2 we see that the slope of the bold line dictates then how fast we achieve detection. What is the slope of the bold line in Figure 2? We use simple geometry to determine this. First, notice that the peak value

$$T_{i^*(k_r)}(D_{k_r} + \delta_s + \delta_e(D_{k_r})) = T_{i^*(k_r)}(D_{k_r}) + \delta_s + \delta_e(D_{k_r})$$

since the slope of the dashed line in Figure 2 is unity. Next, notice that Equation (3) gives the amount of time between the decision time D_{k_r} and time of detection D_{k_r+1} so that the slope of the bold line in Figure 2 is

$$-\left\{ \frac{T_{i^*(k_r)}(D_{k_r}) + \delta_s + \delta_e(D_{k_r})}{\frac{\delta_s + \delta_e(D_{k_r}) + a_{i^*(k_r)} T_{i^*(k_r)}(D_{k_r})}{1 - a_{i^*(k_r)}} - (\delta_s + \delta_e(D_{k_r}))} \right\}$$

which with some simple algebra reduces to

$$-\frac{(1 - a_{i^*(k_r)})}{a_{i^*(k_r)}} \tag{4}$$

In a moment you will see that it is necessary that $a_{i^*(k_r)} < 1$. Using this fact, Equation (4) indicates how fast detection occurs as shown in Figure 3. With small values of a_i (high values of $1/a_i$, the rate of processing by the receiver in trying to detect) we get fast detection, and with larger ones we get slower detection. So, how do we interpret the a_i parameters? They are parameters used to model how difficult it is to detect an emitter, where if an emitter has not been detected for a long period of time, then it can become more difficult to detect.

2.2.2 Capacity Condition

Clearly, it is *necessary* that the "capacity condition"

$$\rho = \sum_{i=1}^{N} a_i < 1 \tag{5}$$

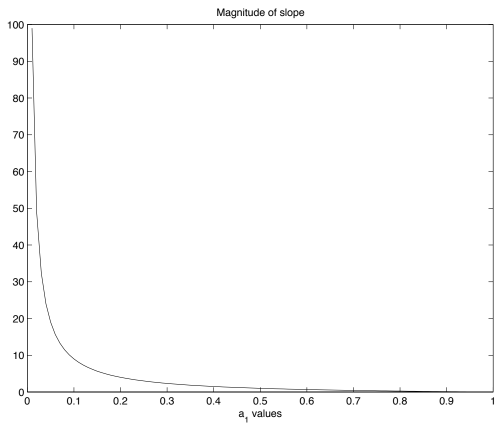


Figure 3: Magnitude of the slope of the bold line in Figure 3 for various values of a_1.

be satisfied in order for *any* policy to ensure that the values of $T_i(t)$, $i \in E$, remain bounded. How should this capacity condition be interpreted? Intuitively, it says that it must be case that even if the emitters can get more difficult to detect if they have not been detected for a long time, the receiver must be able to operate "fast enough" to be able to find them. For instance, Equation (5) is satisfied if for each $i \in E$,

$$a_i < \frac{1}{N}$$

This shows us that as the number of emitters grows, it is possible that the capacity of the receiver is overwhelmed and it is being given too much work so that there is no way that it can keep up so it will end up being the case that $T_i \to \infty$ for at least some $i \in E$ (or more than one i).

Equation (5) can be used to gain insight into the operation of DSSs by using the ideas in [13]. First, note that you can think of a_i as the amount of "load" (or the number of time units of "work") that is brought to the receiver at each time instant by emitter i. Hence, if the receiver is to succeed, on average the receiver can only afford to spend a portion $(1-\rho)$ of its total time being idle. If you assume that the delay $\delta(t)$ is a constant δ, then each decision time when we switch from focusing on one emitter to another costs δ time units of idle time; hence, the average frequency of decision times is bounded above by

$$\frac{1-\rho}{\delta}$$

Now, if ρ is very close to one (representing a receiver that is heavily loaded), $(1-\rho)\delta^{-1}$ is very small so the frequency of switching between different emitters is low (which means that it can take a long time for the receiver to find each emitter so that the receiver will tend to have large $T_i(t)$ values and hence will not perform as well). Similar concepts hold in the analysis of queueing systems.

2.3 Scan for an Emitter Ignored More Than the Average One

There are a wide variety of possible DSS policies. Next, we consider one that is more general than the one of the previous subsection given in Equation (1) in the sense that at each decision point D_{k_r} it could make *exactly* the same decision as it did there, but also it could make other choices.

The particular policy is given by choosing the emitter to focus on that has been ignored more than the average time that all the emitters have been ignored. In particular, at D_{k_r} the DSS chooses to focus on emitter $i^*(k_r)$ such that

$$T_{i^*(k_r)}(D_{k_r}) \geq \frac{1}{N}\sum_{i=1}^{N} T_i(D_{k_r}) \tag{6}$$

and focuses on it until it detects it (in a similar way to the policy of the last section). Note that Equation (3) also holds for this policy.

Note that for this policy *any* emitter can be focused on that has been ignored for more time than the average emitter has. How does the policy choose which particular emitter to focus on? One simple approach is to simply randomly choose one. However, more sophisticated strategies are possible. For instance, it could try to optimize some other system quantity (e.g., the number of CDWs expended for a particular emitter), or it may use Equation (6) to provide a set of possible emitters to choose and then use "emitter priorities" (some indication of which emitter is most important) to choose the one to focus on. In the simulations of the next section when we study this policy we will assume that emitter i has priority i and higher values of i correspond to higher priorities.

2.4 Scan for an Emitter That May be Most Difficult to Find

A DSS policy that scans for the emitter that may be the one that is most difficult to find makes choices of which emitter to focus on such that at D_{k_r} the DSS chooses to focus on emitter $i^*(k_r)$ such that

$$a_{i^*(k_r)} T_{i^*(k_r)}(D_{k_r}) \geq a_i T_i(D_{k_r}), \forall i \in E \tag{7}$$

and focuses on it until it detects it. If there is more than one maximizer, then the DSS will simply choose one of these at random. Clearly, this is similar to the DSS that scans for the emitter that has been ignored for the longest time that was given in Equation (1). Here, however, we have the scalings by the a_i parameters and this changes the DSS. Intuitively, since a_i is the amount of "load" you can think of this DSS as choosing the emitter to scan for that may be the most difficult one to find.

2.5 Scan for an Emitter Expected to be Most Difficult to Detect

The policy to developed next is modeled after the one in [13] that has been found to be very effective in a different class of resource allocation problems. Recall from our earlier analysis that if you pick emitter $i^*(k_r)$ to focus on

$$T_{i^*(k_r)}(D_{k_r}) + \delta_s + \delta_e(D_{k_r})$$

is the peak that $T_{i^*(k_r)}(D_{k_r})$ reaches before the emitter is detected. Note that in general we do not know $\delta_e(D_{k_r})$ since it depends on how the receiver decision times are aligned with the emitter illumination times, two quantities that are assumed uncorrelated in the actual environment. A known bound, however, on the peak value is given by

$$T_{i^*(k_r)}(D_{k_r}) + \delta_s + \delta_e(D_{k_r}) \leq T_{i^*(k_r)}(D_{k_r}) + \delta_s + \delta^{i^*(k_r)}$$

Hence, emitters with larger δ^i values (i.e., ones with possibly lower frequency illuminations) can be considered on average more difficult to detect in this framework. Also, the a_i parameters, which model another characteristic of the difficulty of emitter detection, will also affect how soon detection can occur.

Consider choosing emitter $i^*(k_r)$ to focus on at time D_{k_r} if

$$i^*(k_r) = \arg\max_i \left\{ w_i \left(\frac{T_i(D_{k_r}) + \delta_s + \delta^i}{\frac{(1-a_i)}{a_i}} \right) \right\} \tag{8}$$

where $w_i > 0$, $i \in P$ are weighting factors. Notice that in this formula the numerator is the bound on the peak value and the denominator is the magnitude of the slope of the bold line in Figure 2 given by Equation (4). Why divide by the slope in the above formula? If the slope is greater in magnitude (smaller a_i value), this corresponds to an easier-to-detect emitter and this will result in Equation (8) with a *reduced* emphasis on focusing on that emitter. Hence, the strategy picks the emitter to focus on that is *expected* to be the most difficult to detect in the sense that it estimates which emitter will take the longest time to detect and selects it (assuming $w_i = 1$ for all i). To see this via geometrically, notice via Figure 2 that the numerator $T_i(D_{k_r}) + \delta_s + \delta^i$ in Equation (8) should be thought of as estimate of where the peak occurs and we divide it by the slope; hence, this value is the length of time that elapses from the time that the peak occurs, until detection.

The weighting factors w_i can be chosen to force the emitter to focus on some emitters more than others. Equal weighting would correspond to the choice of $w_i = 1$ for all $i \in E$. If $w_i >> w_j$, $i \neq j$, then Equation (8) will tend to choose i rather than j to focus on. This may be useful in some emitter environments since it provides a way to indicate which emitter should be focused on. While the weighting factors provide an opportunity to tune the policy, there is however no guarantee that this policy will be better than any of the others introduced above according to typical performance measures. Generally, you would want to choose the weights so as to make the scheduler perform as good as possible (where you define what is meant by "good").

2.6 Policies Based on Emitter Priority

Above, we introduced two ways to introduce priorities of emitters into scheduling policies. First, in Equation (6) we used priority as a "secondary" selection mechanism to choose among the set of emitters that have been ignored longer than the average one. Second, in Equation (8) we introduced the weighting factors w_i which allow us to emphasize the processing of one emitter more than another (and this will be illustrated in the simulation examples in Section 3). In this subsection we will introduce yet another priority scheme, but one that integrates the consideration of emitter priorities so that emitter priority is neither a secondary consideration nor set by secondary weighting parameters that have a loose connection with the emitter priorities.

To do this we introduce a set of parameters $p_i > 0$, $p_i \in \Re$, $i \in E$, that represent the emitter priorities (larger values correspond to higher priorities). We will allow the designer to take two different views of the priority parameters:

1. *Emitter environment information:* You can assume that the values of the parameters p_i, $i \in E$, are set a priori and remain constant throughout the mission. Hence, you can view them as part of the a priori information about the emitter environment.

2. *Design parameters:* Alternatively, you may view the priority parameters as design parameters that can be tuned (e.g., via extensive premission simulations of the emitter environment) before a mission. Their values are fixed for a particular mission, and hence will help govern the behavior of the policy as the mission executes.

How do we integrate the priority parameters into *each* of the policies defined in the previous subsections? For example, how can we use them to modify the policy in Equation (1) where we chose to focus on the emitter that was ignored the longest. Here, we simply *scale* T_i by p_i, $i \in E$ in each of the cases and then make all decisions based on the same formulas as above, but with T_i replaced by p_iT_i, $i \in E$. What is the effect of such a scaling? It serves to scale the lengths of times that the emitters have been ignored, with higher weights given to emitters with higher priorities. For such policies to be stable it is clearly necessary that we modify our capacity condition. With priorities, we require that

$$\rho_p = \sum_{i=1}^{N} p_i a_i < 1 \tag{9}$$

be satisfied to ensure that the values of $T_i(t)$, $i \in E$, remain bounded.

How does the scaling affect the behavior of the policies? While it is clear that emitters $i \in E$ with T_i scaled by higher values of p_i will have p_iT_i grow faster (the slope of the line representing the growth is p_i), the behavior is also affected by the range of values that you allow for the priorities. For instance, if you dictate that your priorities $p_i \in (0, 1]$, $i \in E$, then if you were given some a_i values that satisfied Equation (5), the p_i and a_i values would also satisfy Equation (9). Hence, if you use a proper range of values for the priority parameters any policy that satisfies the capacity condition without priorities, will satisfy Equation (9). Note that there is really no reason why you cannot make the choice of $p_i \in (0, 1]$, $i \in E$, since they are simply used to rank order the emitters. It is also interesting to note that if you repeat the analysis in Sections 2.1 and 2.2, the result in Equation (4) still holds (due to cancellations of the priority parameters in the algebra); hence, simulation of the class of priority policies discussed here is quite similar to the earlier policies.

To summarize, *you can embed the priority parameters into any of the above policies.* For instance, Equation (1), when converted to a priority scheme using this approach becomes one where the DSS chooses to focus on emitter $i^*(k_r)$ such that

$$p_{i^*(k_r)} T_{i^*(k_r)}(D_{k_r}) \geq p_i T_i(D_{k_r}), \forall i \in E \tag{10}$$

In this way you can have a policy that selects emitters based on both priorities and how long they have been ignored. The scheduling policies in earlier subsections are modified in a similar manner. Finally, note that you can still use the two priority schemes we discussed earlier in conjunction with this priority scheme.

2.7 Viewpoint of Scheduling as On-Line Optimization

Next, note that we can provide an interpretation of the three above policies in terms of optimization. The key is to think of the decision-making in terms of *optimizing* a cost function J_p, and that J_p is the result of a computation made in the scheduler (controller) so that it can make scheduling decisions.[2] With this view, we have the following:

- Scan for Emitter Ignored for the Longest Time: Here for the policy in Equation (1) we have

  $$J_p = -\max\{T_i(D_{k_r}) : i = 1, 2, \ldots, N\}$$

  and hence in trying to maximize J_p we try to minimize the longest time that the receiver ignores any emitter. In this way, the scheduler tries focus on emitters so as to keep the values of $T_i(t)$ low so that the receiver has good information about the emitters.

- Scan for an Emitter Ignored More Than the Average One: Here for the policy in Equation (6) we have

  $$J_p = -\sum_{i=1}^{N} T_i(D_{k_r})$$

  and hence in trying to maximize J_p we try to minimize the average time that the receiver ignores any emitter. Again, the scheduler tries to focus on emitters so as to keep the values of $T_i(t)$ low so that the receiver has good information about the emitters. Here, however, it makes decisions in a different manner since it tries to maximize a different J_p.

Using this same approach it is simple to specify J_p measures for the other policies we defined above. For instance, for the policy in Equation (7) we have $J_p = -\max\{a_i T_i(D_{k_r}) : i = 1, 2, \ldots, N\}$ and hence in trying to maximize J_p we try to minimize the longest time that the receiver ignores any emitter, but scaled by the "load" of the emitter. For Equation (8) our J_p would quantify the desire to keep the peaks of the $T_i(t)$ as low as possible (which may or may not result in a lower average delay). Clearly, if you embed a priority scheme via the priority parameters p_i, $i \in E$, the same concepts hold.

Note that the above J_p measures should not be thought of as measures of mission success, but as instantaneous measures that are used to guide decisions about which emitter to scan for. Achieving an instantaneous optimization does not necessarily result in making optimal decisions to try to ensure that the receiver gets the best information over the long term. If you are familiar with "model predictive control" (MPC) the J_p is analogous to the cost function used in MPC that is used to pick the control input at each time step and a measure of scheduling performance is analogous to achieved closed-loop performance of an MPC strategy. There are two general types of cost functions, one used to make decisions at each time step, and another used to assess the performance of the system over a long time period after many decisions are made and applied. Clearly, there are influences between the two types of cost functions.

2.8 Dynamic Vs. Fixed Scan Schedules

In what sense are the above scan schedules "dynamic." The term dynamic is used to contrast with the current approaches where "fixed" scan schedules are used. These schedules are developed before the aircraft mission, indicate the search pattern a priori, and the schedule is not changed based on information that is

[2]The quantity J_p is called the "quality of service" (QoS) in [18].

gathered during the mission. A dynamic scan schedule changes its behavior in response to what is sensed during the mission. Some dynamic scan schedules could be initialized with a fixed schedule, and then as information is gathered it could evolve into a dynamic one. Other schedules, such as the ones given above, are inherently dynamic in that after the first time they detect an emitter they can change their sequence of scans/emitters to focus on depending on sensed information.

It is thought that dynamic scan schedules may offer advantages over fixed schedules since they change in response to the current environment of emitters. For example, it could be that the fixed schedule was developed with poor information, the emitter environment may change before the mission is executed, or while the mission is occurring. It is hoped that these changes can be sensed and profitably incorporated into the DSS so as to improve the quality of information that the receiver is able to schedule.

How exactly are the DSSs of this section "dynamic"? Consider how a typical schedule operates, e.g., the one based on focusing on the emitter that has been ignored for the longest time. Suppose that we are at some decision point. Suppose that the DSS picks the emitter that has been ignored for the longest time. First, there is a delay due to the need to switch the receiver to tune to a new frequency band, there may also then be a delay due to the need to wait for the emitter to illuminate, and finally there may be additional time needed since the emitter has not been focused on for some time (the a_i parameters quantify how much). Now, suppose that due to aircraft movements the emitter illumination comes a bit later than when it was expected but within the fixed known bound. The scheduler's next decision is then delayed, and hence it is slower to make its next decision and hence its next scan (i.e., its behavior changes). Now, clearly it is also the case that the frequency of the emitter illuminations in the environment will change the sequence of scans for the emitters (note that one DSS can be used for a variety of possible environments). The DSS policy will try to keep good information about all the emitters somewhat independent of their illumination frequencies, but it will tend to be the case that higher frequency illuminators will be found easier, but may at times be ignored in favor of scanning for an emitter with a lower frequency illumination if that helps to keep information about it up-to-date. These characteristics are further explained in the next section via a simulation analysis of the DSS policies.

3 Design and Simulation of Dynamic Scan Schedulers

In this section we will simulate the DSS policies of the last section in order to provide insights into their operation. The code for the simulation is Matlab 5.2 and it is given in the Appendix (email the author for an electronic copy). Moreover, we will discuss several issues in how to design DSS policies.

3.1 Simulation Approach and Performance Measures

For convenience we simulate the emitter environment and receiver as a discrete-time system. We will use a sampling period of $T_s = 0.01$ and in all our simulations we will have $N = 4$ emitters. Each emitter will be characterized by a sequence of illuminations, that we simply model as unity height signal at some sampling instant, and where there is no illumination the signal height is zero. For instance, for all our simulations below we will have the emitter illumination sequences shown in Figure 4. We use different frequencies of illumination for different emitters, but for simplicity we keep the illumination frequencies constant (for emitter $i = 1, 2, 3, 4$ we have them appear every 1, 1.1, 1.2, and 1.3 sec.).

Suppose that we know that the bounds on the spacing between illuminations are

$$\delta^1 = 1.05,\ \delta^2 = 1.15,\ \delta^3 = 1.25,\ \delta^4 = 1.35$$

Notice that these are simply bounds for periods given in Figure 4. We choose $\delta_s = 0.03$. To model detection difficulty, and in order to satsify the capacity condition we choose

$$a_1 = 0.1,\ a_2 = 0.2,\ a_3 = 0.3,\ a_4 = 0.1$$

This gives $\sum_{i=1}^{4} a_i = 0.7$ which represents that the receiver will be quite busy in detecting emitters (lower values of this sum correspond to light loads).

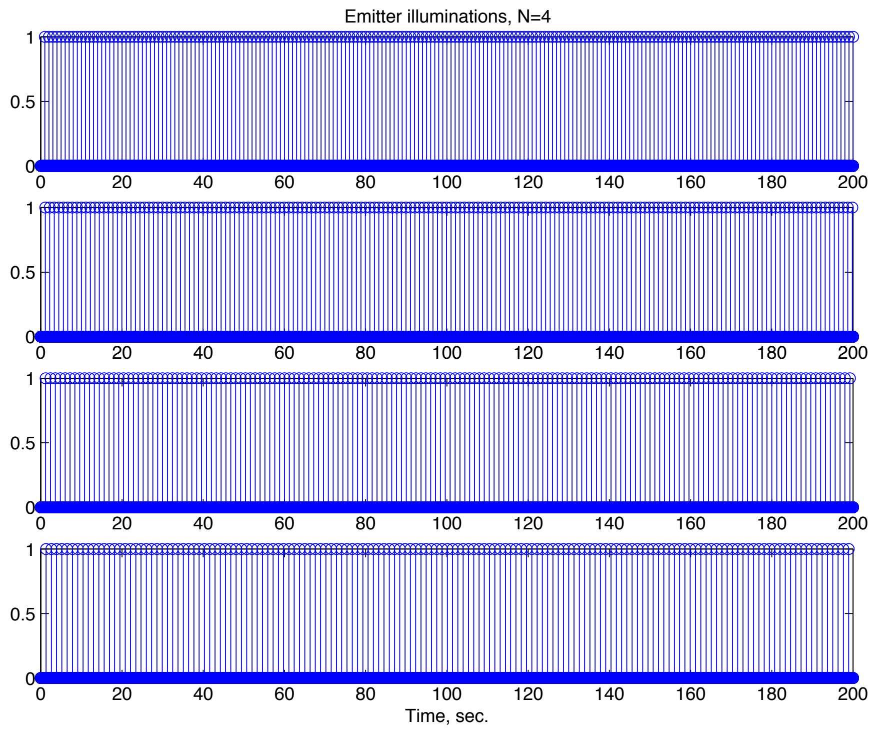


Figure 4: Emitter illumination sequences, for $N = 4$ emitters (emitter $i = 1$ is the top plot, $i = 2$ is the next one down, $i = 3$ is below that, and $i = 4$ is the bottom plot).

There are several ways to measure performance of the DSS strategies. Here, we will compute the average of the length of time since any emitter has been detected

$$\frac{1}{N}\sum_{i=1}^{N} T_i(k)$$

at each step k. We will also compute the time average of this quantity (i.e., the time average of the average values) and the maximum average value achieved over the entire simulation run. We will compute the maximum time that any emitter has been ignored at each time step k

$$\max_{i=1}\{T_i(k)\}$$

We will also compute the time average of this quantity (i.e., the time average of the maximum values) and the maximum of the maximum values achieved over the entire simulation run. In order to measure how well we have focused on higher priority emitters we will use

$$\frac{1}{N}\sum_{k} i^*(k)$$

where $i^*(k)$ is the emitter chosen as step k. Clearly, higher values of this measure will correspond to the case where on average higher priority emitters were focused on, in the case where we use i to both label the emitters and as a priority parameter.

3.2 DSS Strategy Behavior: Focus on Longest Ignored

Here, we will illustrate the performance of the DSS strategy in Equation (1) that chooses the emitter to focus on that has not been detected for the longest period of time.

First, consider Figure 5 where the top plot shows $i^*(t)$, the emitter being focused on at each time. The plot below it shows $T_1(t)$, and the bottom plot shows $T_2(t)$. From the top plot it is interesting to note that the sequence of emitters that are focused on is: $1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, \ldots$. But, the lengths of time that each is focused on is different due to how the receiver decision times happen to line up with the emitter illuminations and due to the a_i values. Notice the periodic behavior of the $T_1(t)$ and $T_2(t)$ plots (due to the switching from focusing on one emitter to another). Figure 6 shows a similar plot, but for emitters 3 and 4. Notice that the periodic behavior of T_3 and T_4 is different from those shown in Figure 5. Ultimately, the pattern of the behavior of the $T_i(t)$ depends on the pattern of emitter pulses, the a_i values, the delay values, and how the emitter illuminations align with the decision times.

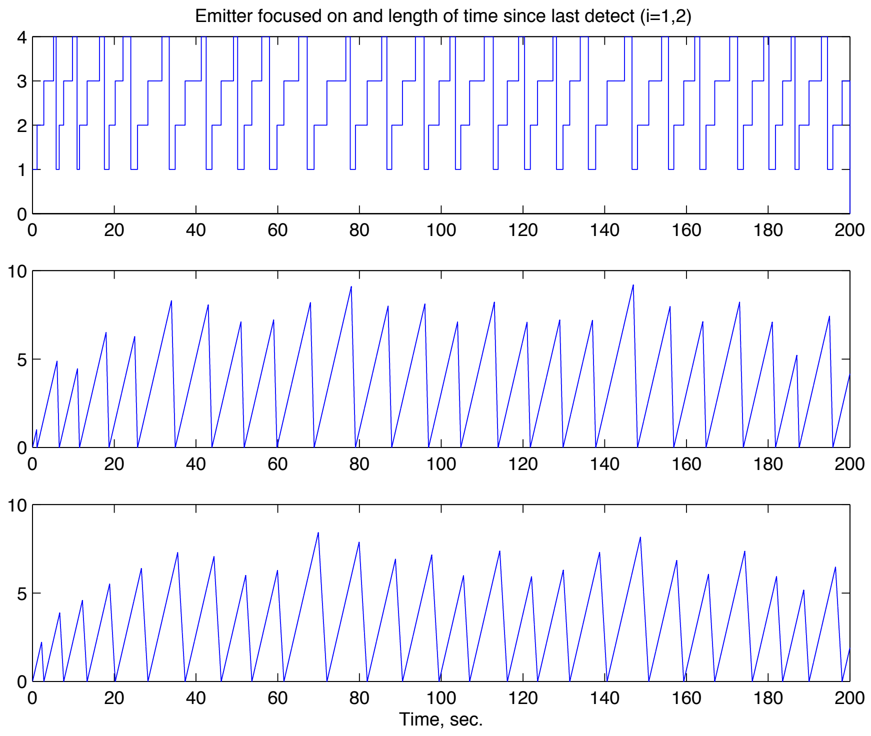


Figure 5: DSS decisions, and $T_i(t)$ for emitters 1 and 2.

Figure 7 shows a summary view of the dynamics of the DSS process. There, in the top plot, we also plot the average of the priorities of the emitters (assuming that priorities are defined by the i indices). The bottom plot shows the dynamics by showing all the $T_i(t)$ functions on one plot so that you can see the pattern of switching, and the maximum amount of time that the receiver ignores any emitter. In Figure 8 we plot the performance measures of the average length of time since the last detection and maximum length of time since the last detection (and their average values as the straight lines).

Next, the program outputs some numeric values of the performance measures: (i) The time average of the priorities is 2.5670, (ii) the time average of the average values of the lengths of times waited is 3.4066, (iii) the maximum of the average values of the lengths of times waited is 5.7949, (iv) the time average of the maximum values of the lengths of times waited is 5.8297, and (v) the maximum of the maximum values of the lengths of times waited is 9.6199. The time average of the average values is 3.4066, and this provides a good measure of scheduler performance. What does this value mean? It means that on average the receiver detected each emitter each 3.4066 seconds. Is this good performance? Notice that the emitter illuminations occurred every 1, 1.1, 1.2, and 1.3 seconds (emitters $i = 1, 2, 3, 4$ respectively). Considering the relative low rates of processing to detect the emitters, and the delays in switching and waiting for illuminations, this appears to be reasonably good performance. Clearly the performance could go up or down if the frequency or timing of the emitter illuminations changed.

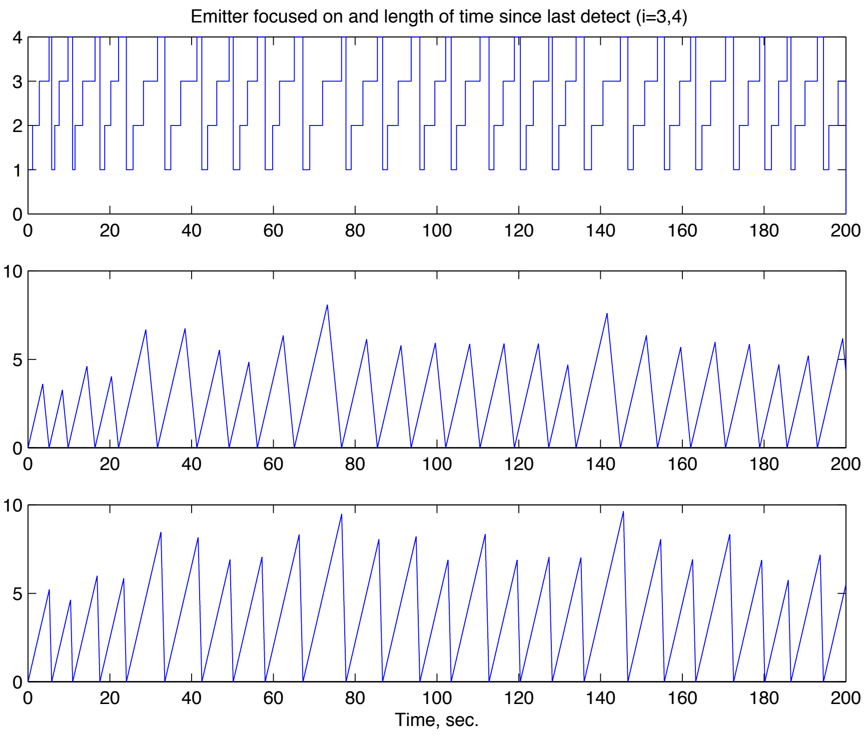


Figure 6: DSS decisions, and $T_i(t)$ for emitters 3 and 4.

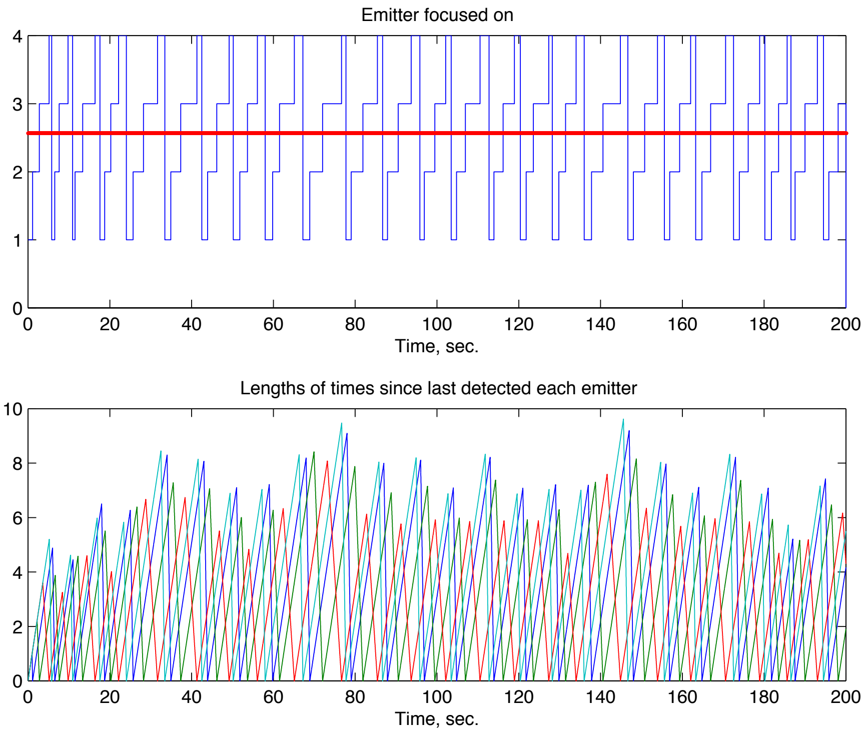


Figure 7: DSS decisions, and $T_i(t)$ for emitter $i = 1, 2, 3, 4$.

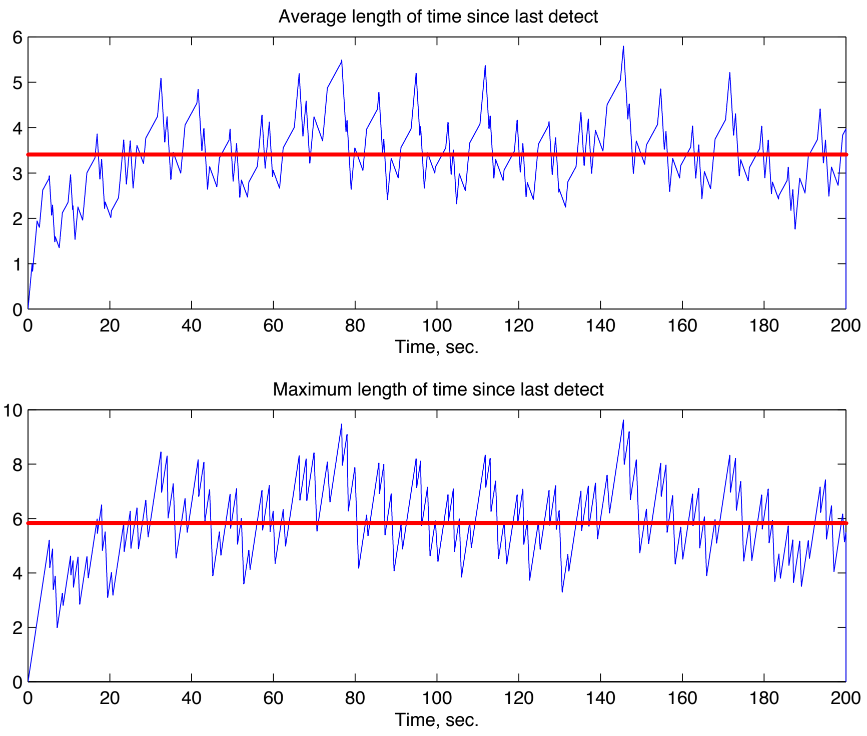


Figure 8: Performance measures (average and maximum times since last detection) and the time averages of their values.

3.3 Effect of Focusing on Higher Priority Emitters

Next, we use the strategy in Equation (6) that picks the emitter that has been ignored longer than the average one. For the set of emitters that has been ignored longer than the average one, we choose the one that has highest priority (i.e., emitter i with the greatest value of i). In this way we study how priorities enter into DSS by augmenting the strategy with a priority scheme. In this case we get Figures 9 and 10. We see in Figure 9 that the sequence of emitters that are focused on is different from the previous strategy, and that the sequence is not periodic in the same way (e.g., it is not a simple $1, 2, 3, 4$ sequence). Also, we see that the average value of the priority of the emitter that is focused on is a bit higher, as we would expect. The bottom plot in Figure 9 shows quite a different behavior that the bottom plot in Figure 7; notice that here there is not an equal "balance" in focusing since we see that the average values of the $T_i(t)$ are quite different (e.g., see the occasional peaks). Next, notice that in Figure 10 we get poorer performance than that shown in Figure 8.

To quantify the performance further notice that the numeric performance measures are: (i) the time average of the priorities is 2.6896, (ii) the time average of the average values of the lengths of times waited is 3.8204, (iii) the maximum of the average values of the lengths of times waited is 6.4525, (iv) the time average of the maximum values of the lengths of times waited is 7.6574, and (v) the maximum of the maximum values of the lengths of times waited is 15.7399. This clearly shows that while we get *slightly* better focusing on higher priority emitters, we get poorer performance for all the other performance measures. We have *paid a price* in focusing on high priority emitters, by ignoring other emitters for longer periods of time.

3.4 Tuning DSS Strategy Parameters

As we saw with Equations (8) and (10) there are ways to define DSSs in terms of a set of parameters that specify how they make decisions (e.g., weights or priorities that modify J_p). For instance, we could specify the w_i weights such that there is a high emphasis on focusing on one emitter. To do this you simply make

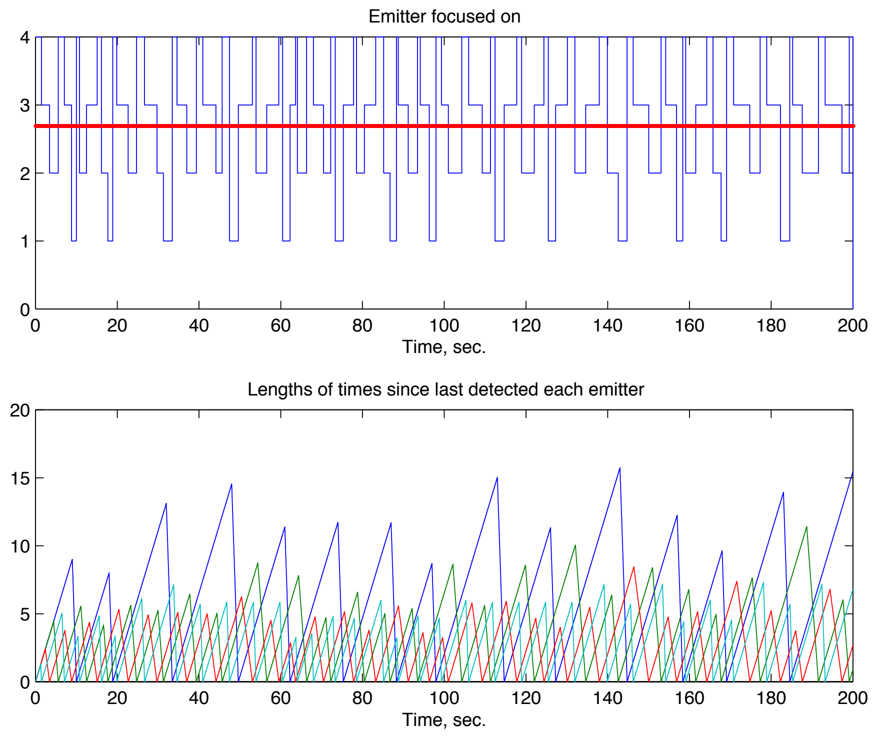


Figure 9: DSS decisions, and $T_i(t)$ for emitters $i = 1, 2, 3, 4$.

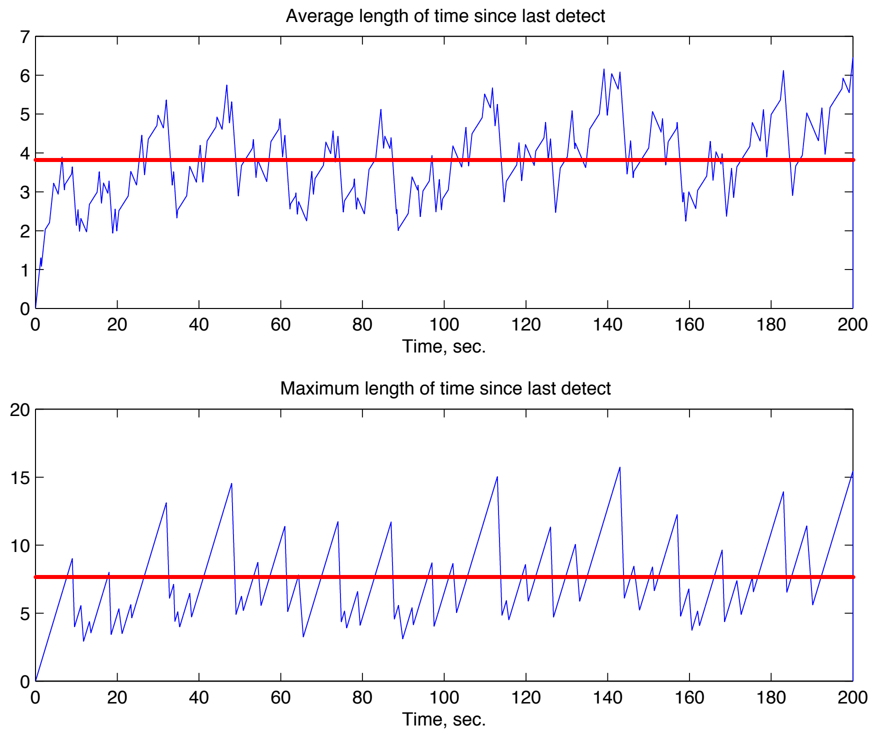


Figure 10: Performance measures (average and maximum times since last detection) and the time averages of their values.

one w_i value much larger than the others. This will result in frequent focusing on the corresponding emitter. Suppose that we are not concerned with emitter priority, or that all the emitters have the same priority.

Can we tune the w_i values in Equation (8) in order to try to improve the performance measures? That is, can we use the parameters to simply try to improve performance, rather than emphasize focusing on a particular high priority emitter? The answer is yes, and to illustrate this we ran a few simulations tuning the w_i values with a focus on trying to minimize time average of the average values of the lengths of times waited. We obtained $w_1 = 4$, $w_2 = 2$, $w_3 = 1$, and $w_4 = 4$ and we get the performance in Figure 11. The tuning strategy used was to try a set of w_i values and look at the $T_i(t)$ plots. Then the value of w_i was increased a bit for the emitter that had higher peak values in order to try to make the strategy focus on that emitter more heavily.

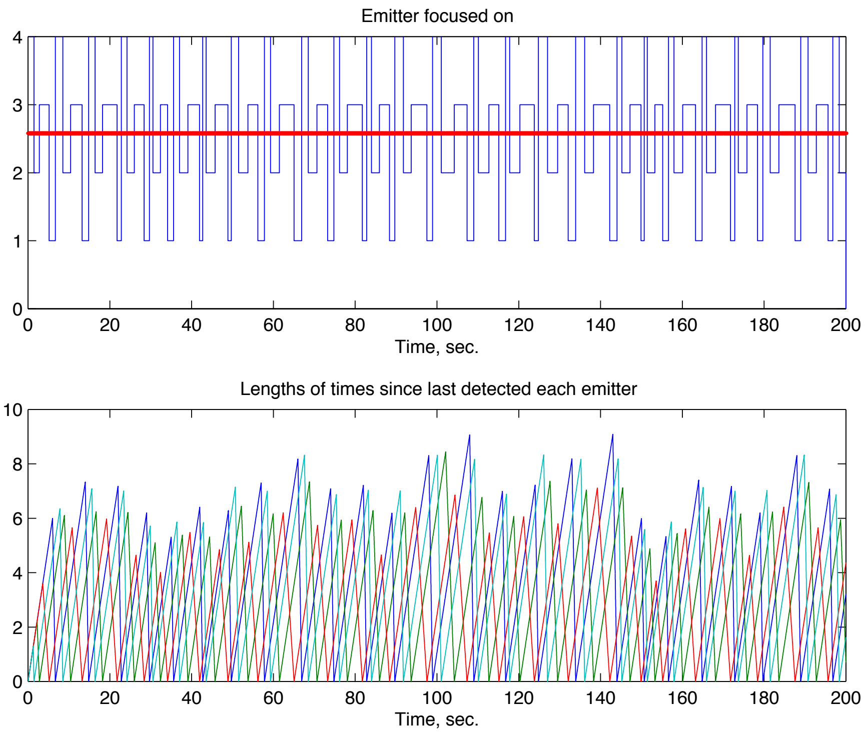


Figure 11: DSS decisions, and $T_i(t)$ for emitter $i = 1, 2, 3, 4$.

The performance for this new set of w_i values is quantified via the following: (i) the time average of the priorities is 2.5802, (ii) the time average of the average values of the lengths of times waited is 3.2755, (iii) the maximum of the average values of the lengths of times waited is 5.3599, (iv) the time average of the maximum values of the lengths of times waited is 5.6423, and (v) the maximum of the maximum values of the lengths of times waited is 9.0899.

Notice that compared to the result in Section 3.2 we have tuned the w_i values to get a better value for time average of the average values of the lengths of times waited (there we obtained 3.4066). Is there further room to improve the perfomance of the scheduler? This seems likely as the tuning process did not involve consideration of too many values of the parameters. It should be clear that the tuning problem can be quite difficult, especially if there are many emitters.

4 Optimization of DSS Policy Parameters: Response Surface Methodology

In this section we discuss the "response surface methodology" as an approach to tune DSS parameters.

4.1 Response Surface Methodology for DSS Design

Is there a systematic approach to tune the w_i weights to improve performance? Optimization techniques may be used to try to maximize some long-term measure of mission success by tuning the w_i parameters for a particular environment (or class of environments). Such methods would allow for off-line a priori design of dynamic scan schedules based on known information (philosophically similar to fixed scan schedules); *but the schedulers are still dynamic and hence information obtained during the mission can be exploited to enhance performance.*

One approach to solve an optimization problem as general as this is to use a "response surface methodology" (RSM) [8] where Monte Carlo simulations are used to characterize the performance of a DSS with a particular set of parameters for range of possible emitter environments. A plot of the performance measure vs. DSS parameters provides a "response surface" that characterizes how well the DSS will perform over a whole range of DSS parameters. One can then perform optimization over this surface to find the set of DSS parameters that will perform the best for the given emitter environment. What is, intuitively, the point where optimal DSS performance is achieved? Intuitively, the optimization methods will seek to tune scheduler parameters so as to "balance" the focusing on the emitters so that it does not pay attention to any one emitter for a long time. Note that such a balance was achieved to a certain extent for the first policy (the one based on Equation (8)) in the last section (see Figure 7 where the peaks of the T_i plots are nearly equal, and this is what results in the lowest value for the time average of the average values of the lengths of times waited, a critically important performance measure). Is there room for improvement over the performance for that policy? If you study Figure 7 carefully you will see that a perfect type of balancing was not achieved there; it seems that there is room for improvement, especially considering the tuning we did in the last section. However, a systematic approach is needed to try to find the optimal set of DSS parameters.

Is there a systematic approach to design DSSs? Yes. First, note however that the partial derivative of typical performance measures with respect to the parameters of a DSS may not exist or may be difficult to work with (e.g., in the typical decision-rules there are discontinuities). This makes the direct application of gradient methods for optimizing performance problematic. One approach to solve such problems is to use a "nongradient" method such as pattern search or direct search. Also, you could use genetic algorithms to try to minimize some performance measure. However, compared to these methods, the response surface methodology offers not only values of the optimal parameters, but also the "response surface" that indicates how performance will improve or degrade with changes in the parameters. Moreover, it includes a methodology to try to minimize the computations needed in trying to find the optimal parameters. The disadvantage of the methodology is that for large numbers of design parameters (e.g., for large numbers of emitters) it is difficult to gain insights, and the complexity of constructing a response surface for representative emitter environments may be very high. In such situations it may be best to employ certain deterministic or stochastic optimization methods (which we discuss later).

4.2 Example: DSS Parameter Design, $N = 3$ Emitters

We will study how to tune the w_i values in Equation (8) in order to improve the time average of the average length of time an emitter is ignored. A similar methodology would be used to study performance optimization for other performance measures. We study the case for $N = 3$ emitters since there are only 3 parameters to tune for this case, and generally what matters for the w_i values is the relative size of the weights. Hence, we will fix $w_2 = 1$ and will vary w_1 and w_3 across all combinations of the values

$$w_1 \in \{0.1, 0.2, 0.3, \ldots, 3\}$$

and

$$w_3 \in \{0.1, 0.2, 0.3, \ldots, 3\}$$

The response surface will hold values of the time average of the average time that the emitters are ignored. We use the same emitter environment as earlier, expect we remove the fourth emitter, and use $\delta^1 = 1.3$, $\delta^2 = 1.6$, and $\delta^3 = 2.3$. We simulate it for 100 sec. Also, we let $a_i = \epsilon \frac{1}{N}$ where $\epsilon = 0.1$ (a change in a_i parameters compared to the earlier example).

The response surface for the DSS of Equation (8) and the above range of w_1 and w_3 values is shown in Figure 12. This surface indicates that the best gains are $w_1 = 1.3$ and $w_3 = 0.7$ (found via performing a simple minimization over the response surface values at the cross-points on the mesh in the figure). For these gains the value on the response surface is a time average of average wait times of 0.9485.

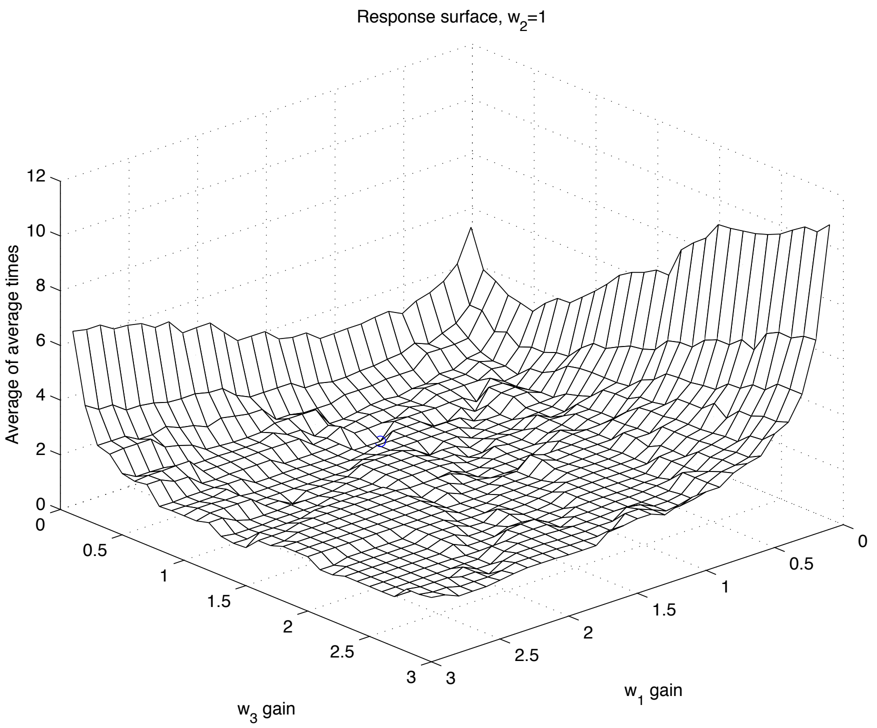


Figure 12: Response surface of time average of the average times that emitters wait to be focused on.

The surface clearly shows that choosing the w_1 and w_3 values too small relative to w_2 results in degraded performance; this performance degradation is due to an over-emphasis on focusing on some emitters rather than others (i.e., poor "balancing" in focusing results for that case). It is interesting to note that the frequency of illuminations of emitter 1 is greater than that of emitter 2, which is in turn greater than that of emitter 3. Also, the a_i parameters are all the same and the δ^i are ordered similarly. Hence, the "optimal" choice of weights shows that it is best in this case to place emphasis on higher frequency emitters over lower frequency ones. The other interesting feature of Figure 12 is that there seem to be a relatively wide range of values for w_1 and w_2 that result in nearly the same performance.

4.3 Robustness, Design Point Choice, and Complexity

We close this section with a discussion of four related central topics in response surface methodology: response surface construction and robust design, how to pick "design points," how to approximate a complex response surface, and how to use optimization methods to find an optimal design.

4.3.1 Robust DSS Design via Response Surfaces

It is important to point out that the w_i values are only "optimal" for the stated conditions. If you change the emitter frequencies, δ^i values, a_i values, etc. the "best" design parameters are likely to change. Why? Consider using all the same parameters as in the last subsection, but where you change the receiver so that it has different a_i values. In this case, the DSS will in general make different decisions, and hence the performance measure and response surface shape will change.

Of particular importance is the fact that the response surface changes for different conditions in the emitter environment. For example, if you know more about the emitter environment, you may know tighter bounds on the delays and hence have different (smaller) values for the δ^i parameters. This too will lead to the DSS making different decisions and hence a different performance measure value and response surface shape. Even more important is the fact that the environment is *uncertain* and the ability of a DSS to perform well in spite of this uncertainty is a very desirable characteristic. This feature is referred to as "robustness." Design of optimal robust DSS is a very important problem, and one way to approach this problem is to use a response surface methodology.

The approach to robust DSS design is to compute multiple performance values at each "design point" (e.g., w_1 and w_3 values for the above example), and to compute the average of the performance values and use that for the response surface. How do multiple performance values arise for each design point? You could generate multiple variations in the timing of illuminations that represent possible variations in the emitter environment. Each variation would by itself result in a different response surface, with a resulting indicated "optimal design." By simulating many possible expected emitter environments and constructing a response surface based on averages of performance measures at each design point you are providing a DSS that will be "robust" to the expected variations in the environment.

4.3.2 Choosing Design Points: "Design of Experiments"

Above, we have ignored issues of complexity that arise due to the possibility that there may be many "independent" design variables. We only considered the $N = 3$ case but with only $n = 2$ two design variables. In this case it was simple to grid on each of the two dimensions, and plot the resulting three-dimensional response surface. What if the number of emitters N is much larger? Certainly visualization becomes much more difficult, essentially requiring you to fix all but two (or three) variables and plot the response surface for the other ones.

The other problem that arises is due to the "curse of dimensionality" if you grid the design variable space with a uniform grid. For instance, suppose that there are n_G points (grid partitions) on each dimension. Then, using uniform gridding there will be

$$(n_G)^n$$

"design points." For example, for the DSS we had $n = 2$ design variables and $n_G = 30$ for a total of 900 design points. What if we had tuned all three w_i parameters? Then, $n = 3$ and the number of design points is 27,000. Next, suppose that we tuned a DSS for an emitter environment with $N = 100$ emitters and we tune all the w_i values so $n = N$ (not at all unrealistic). Then, there would be $30^{100} = 5.1538 \times 10^{147}$ design points and this would likely present a serious challenge, no matter what type of computer you own.

The common solution to such a problem is to be very careful about which design points are considered. We were not careful for the $N = 3$ DSS case since the problem is not very complex, and even there it should be clear that we "wasted" some computations (e.g., notice that in flat regions of the response surface we could have computed far fewer points). The typical approach in response surface methodology to choose design points is to use concepts from "design of experiments" (DOE). For instance, suppose that there are only two values for each design variable on each dimension. In this case $n_G = 2$ so that the number of design points becomes 2^n. This choice of design points is called the "2^n factorial design" and is a common choice for practical response surface methodology (at least for initial "screening" to determine which variables are key factors influencing response surface shapes).

As an example, notice that for the $N = 3$ DSS this choice would correspond to picking the four corners of the design space considered earlier. How would this have worked for the $N = 3$ DSS? By studying the earlier response surfaces you will notice that this would have resulted in a choice of $w_1 = w_3 = 3$ which is not as good as when we used more design points (but it may be acceptable). Note that the 2^n factorial design considers simple slopes in each dimension and hence can in general provide a rough approximation to where the optimal design is. However, this may be necessary when you work with practical problems when there are many dimensions.

Notice, however, that with the 2^n factorial design approach the curse of dimensionality still holds due to the exponent. In practical problems, for large n, it then sometimes becomes necessary to use a "fractional factorial design" where a fraction of the set of 2^n design points is used (e.g., only one point on some dimensions). Clearly, for the DSS design problem such an approach would generally result in an even worse design that the 2^n factorial design would provide. Generally, then, you see that the ability to test more design points, *provided that they are chosen judiciously*, will generally result in better designs. We pay for getting a good design by additional computations, *and* by the need for good insights into how to choose representative design points.

4.3.3 Response Surface Construction is Function Approximation

While in this section we simply plotted the surface using actual data from the simulations, it is clearly also possible to approximate the data using function approximation methods (i.e., interpolation methods where nonlinear function that serves as an approximator is tuned via a least squares or gradient method). To do this you simply choose an appoximator structure and optimization method for constructing the approximator by tuning its parameters. In tradional RSM linear or polynomial models are often used ("first" or "second order" models), and least squares is used to fit the models to the data. It may be productive to use neural or fuzzy systems as approximators, and least squares or gradient methods to tune them. DOE provides a method to choose the training data set and choice of the data set is one of the key issues in RSM, just as it is in approximator design.

It is important to note that the fundamental principles of learning as function approximation hold here. For example, the size of the approximator should be bounded by the number of design points, otherwise poor generalization can occur; in this case the surface could suggest an optimal design point that is far different from the actual one. Clearly, if you only have a few design points, you will only be able to use a simple (e.g., linear) approximator structure. Moreover, it is important to keep in mind that if you have a finite amount of data it may be just as good to simply find the minimum computed value of the response surface (just like we did for the DSS design problem above). Clearly, however, it could be possible that an interpolation of the data could suggest a better design.

4.3.4 Design Via Optimization Over A Response Surface

As shown earlier, to obtain an "optimal design" you find the minimum point on the response surface. If you do not use a simple brute force approach where you find the minimum of all computed points on the response surface, you can perform an optimization over the response surface via some gradient method. In fact, the most common approach in traditional RSM is to use steepest descent gradient methods. Clearly, however, it may be better to use some other optimization method (e.g., conjugate gradient or Levenberg-Marquardt).

Finally, note that it is best to think of the approximation of the surface as providing a way to consider design points without actually having to compute them; if you have a good interpolator, then working with the response surface is almost the same as working with the simulator (or experiment) used to generate response surface points (the "almost" is governed by the approximation accuracy). Hence, practitioners often think of the use of the approximator and optimization over that surface as showing a path of design points that leads to an optimal design, that *you do not have to compute* since they are similar to points that you have already computed the performance for. Hence, the use of the approximator for the actual surface is thought of as a way to help cope with computational complexity.

5 Optimization of DSS Policy Parameters: Simultaneous Perturbation Stochastic Approximation (SPSA)

There are a variety of difficulties that can be encountered in applying a RSM for DSS design. While RSM can provide good design insights for low dimensional problems, it can result in many wasted computations for high dimensional ones (e.g., for "flat" regions of the surface), with little to offer in the way of design insights. Moreover, if the response surface is relatively smooth it may be possible to simply perform a type of "hill climbing" by generating a sequence of (hopefully) increasingly better design points, *without* having to produce the entire cost surface. Traditional hill climbing algorithms, however, such as gradient methods cannot be used for this type of design problem due to the fact that we do not in general have an analytical characterization of the gradient. It is possible to use a simple central difference method to approximate the gradient; however, there are other methods that can perform just as good as such a method, but where far fewer computations are needed. One such method, that has been used in a variety of nongradient optimization problems, is the "simultaneous perturbation stochastic approximation" (SPSA) method [14, 15, 16, 17].

Below, we first introduce the SPSA algorithm, contrast it with the use of gradient approximation approaches, and then we will give an example of how to use the approach for a simple optimization problem. In the next subsection we will show how it can be used for DSS parameter design.

5.1 SPSA Algorithm

Consider minimizing $J(\theta)$ by adjusting $\theta \in \Re^p$. We assume that the gradient $\nabla J(\theta)$ is not known analytically and that we cannot measure or compute values of $\nabla J(\theta)$ for any $\theta \in \Re^p$. Assume that given any θ we can compute or measure $J(\theta)$ to obtain

$$J_n(\theta) = J(\theta) + w$$

where w is noise, so that we can obtain noisy measurements of $J(\theta)$ at θ (e.g., it could be that the expected value $E[w] = 0$ and variance $E[w^2]$ is finite).

Consider the parameter update formula

$$\theta(j+1) = \theta(j) - \lambda_j g(\theta(j), j) \tag{11}$$

where $g(\theta(j), j) \in \Re^p$ is an estimate of $\nabla J(\theta(j))$ at $\theta(j)$ and $\lambda_j > 0$ is a scalar step size (a typical choice for λ_j is one where its value decreases in size as the number of iterations increases). The dependence on j is included in $g(\theta(j), j)$ since at two different iterations with the same $\theta(j)$ we may use different approximations to the gradient (more details will be given on this below). Note that a standard gradient "projection method" can be used to keep the parameters within a known bounded (convex) region.

Here, a "simultaneous perturbation" approximation is used for $g(\theta(j), j)$. In particular, each component of the approximation, $i = 1, 2, \ldots, p$, to the gradient is chosen as

$$g_i(\theta(j), j) = \frac{J_n(\theta(j) + c_j \Delta(j)) - J_n(\theta(j) - c_j \Delta(j))}{2 c_j \Delta_i(j)} \tag{12}$$

where $c_j > 0$ for all j (a typical choice is to use a sequence of c_j whose values decrease in size as the number of iterations increases) and

$$\Delta(j) = \begin{bmatrix} \Delta_1(j) \\ \vdots \\ \Delta_p(j) \end{bmatrix}$$

is a random perturbation vector. The components of the vector $\Delta(j)$ should be independently generated from a zero mean probability distribution and a theoretically valid choice is to use a Bernoulli ± 1 distribution for each ± 1 outcome. In this way, the $\theta(j) \pm c_j \Delta(j)$ lie on corners of a hypercube centered at $\theta(j)$; it is at these values that J_n is computed in Equation (12). Note that projection can be used to keep the generated parameters in a known (convex) bounded region.

Note that if $p = 2$, then the $\Delta(j)$ are the corners of a unit square (i.e., one with unit magnitude for each edge) so for each j

$$\Delta(j) \in \left\{ \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ -1 \end{bmatrix} \right\}$$

In general, there are 2^p possible $\Delta(j)$ values (how does this compare to the 2^p-factorial design used for RSM?). If $\theta(j) = [2, 2]^\top$ and $c_j = 1$, then the four corners of the square centered at $\theta(j)$ where we *might* make calculations for values of J_n in Equation (12) are shown in Figure 13. For this example, if $\Delta(j) = [1, 1]^\top$, then

$$\begin{aligned} \theta^+(j) &= \theta(j) + c_j \Delta(j) = \begin{bmatrix} 3 \\ 3 \end{bmatrix} \\ \theta^-(j) &= \theta(j) - c_j \Delta(j) = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \end{aligned}$$

which are the upper right and lower left corners (denoted with large black dots) of the square centered at $\theta(j)$ in Figure 13 (and these values are swapped if $\Delta(j) = [-1,-1]^\top$). The points on the other diagonal of the square are chosen if $\Delta(j) = [1,-1]^\top$ or $\Delta(j) = [-1,1]^\top$. For the $\Delta(j) = [1,1]^\top$ case Equation (12) gives for this example

$$g_1(\theta(j), j) = g_2(\theta(j), j) = \frac{J_n(\theta^+(j)) - J_n(\theta^-(j))}{2c_j}$$

so the approximation to the gradient is computed via a type of "central difference" approximation. If $\Delta(j) = [-1,-1]^\top$, then you get *exactly* the same approximation for the gradient as for when $\Delta(j) = [1,1]^\top$ (why?). Also, the points $\Delta(j) = [1,-1]^\top$ or $\Delta(j) = [-1,1]^\top$ both lead to the same approximations for the gradient, but a generally a different one from that obtained when $\Delta(j) = [-1,-1]^\top$ (why?).

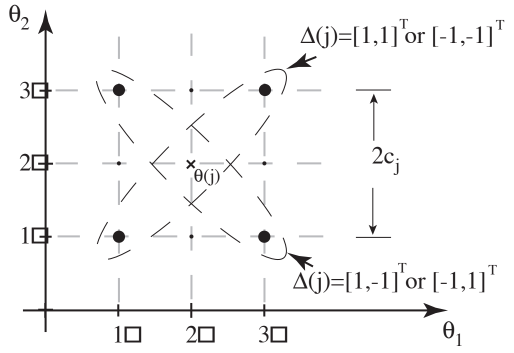


Figure 13: Illustration of what values are used in computing an approximation to the gradient in the SPSA method.

It is useful to envision, via Figure 13, how the SPSA algorithm operates. Intuitively, at each iteration it generates at random two points on a hypercube centered at $\theta(j)$ that are on a diagonal. Then it uses these to compute the approximation to the gradient that is used to specify the direction of the update in Equation (11). For instance, consider the case for the above example where $J_n(\theta) = \theta^\top\theta$ with no noise (so $\theta = 0$ is the global minimum) and note that for one approximation to the gradient it can choose it will be that $J_n(\theta^+(j)) - J_n(\theta^-(j) = 0$ (so there will be no update) and for the other choice $J_n(\theta^+(j)) - J_n(\theta^-(j) > 0$ so that the parameters will be updated to move toward the optimum point (and if the step size is of appropriate size it will not overshoot the origin). Also note that if you had a no-noise case with $J_n(\theta) = (\theta-\theta^*)^\top(\theta-\theta^*)$ so θ^* is the global minimum, then clearly there could be a choice of θ^* such that for the above case one gradient approximation would lead to a parameter update that would decrease the cost, while for the other one it could *increase* the cost. Next, note that since the c_j and λ_j sequences of values decrease at each iteration, the size of the hypercube and the size of changes to the parameter values at each iteration do also; hence, for iterations at the beginning the algorithm will generally make larger updates and thereby explore larger regions of the optimization space. As time progresses, the size of the hypercube and parameter updates decreases and hence (under reasonably general conditions) the algorithm will converge.

5.2 Algorithm Complexity and Convergence

In general, for SPSA if you use the Bernoulli $\pm$ distribution for Δ there are 2^p possible $\Delta(j)$ vectors and $2^{(p-1)}$ possible diagonals on the hypercube centered at $\theta(j)$. Hence, if there is no noise (i.e., $w(j) = 0$ for all j), then using the SPSA approach there are in general $2^{(p-1)}$ possible approximations to the gradient at each iteration, and hence one of $2^{(p-1)}$ possible update directions is chosen at each iteration (note that it is not 2^p possible directions since for the no-noise case specific values of the cost are computed and hence the sign of their difference is fixed once the points at which the costs are computed are fixed). Contrast this with the stochastic gradient method where analytical gradient information is used, no noise is used for the measurement of $J(\theta)$, and one of an *infinite* number of possible update directions is chosen. For the stochastic gradient method we have constraints on the perturbations to the gradient; here, the use of points on the hypercube centered at $\theta(j)$ constrains the size and directions of the update. Moreover, choosing c_j and λ_j as decreasing sequences for SPSA is conceptually consistent with the constraints needed for convergence for the stochastic gradient method.

Notice that for the SPSA there are only *two* perturbations taken and these are used to compute all components of $g(\theta(j), j)$. The approach where a central-difference approximation is used for the gradient has been called the "Keifer-Wolfowitz finite difference stochastic approximation" (FDSA) algorithm when it is applied to minimization of J_n. Notice that with FDSA if there is no noise in measuring $J(\theta)$, then there is only *one* possible update direction at each iteration. Notice that for FDSA p scaled unit vectors are used, one for computing each component of $g(\theta(j), j)$. To contrast FDSA and SPSA, use the above example and note that for FDSA the four points that are used to compute $g(\theta(j), j)$ are the small black dots shown in Figure 13. Note that FDSA needs all four of these points; the large black dots in Figure 13 represent points that SPSA *might* need.

Hence, the above example illustrates that while there are $2p$ calculations of J_n needed by FDSA at each iteration for the computation of the approximation to the gradient, only two such calculations are needed by SPSA (where if $p = 1$ the methods are equivalent). In fact, it has been shown in [14] that under reasonably general conditions SPSA and FDSA achieve the same level of statistical accuracy for a given number of iterations and

$$\frac{\text{Number of measurements of} J_n(\theta) \text{in SPSA}}{\text{Number of measurements of} J_n(\theta) \text{in FDSA}} \rightarrow \frac{1}{p}$$

as the number of measurements gets large. This is an important property since it shows that SPSA could be more efficient for large scale optimization problems, and that this is certainly the case when comparing to the FDSA method. The types of conditions for convergence that are needed for the SPSA method include certain conditions on c_j and λ_j (that are guaranteed with the choices to be outlined in the next section), the variance on the noise on the cost function must satisfy certain bounds, the size of $\theta(k)$ must almost surely be bounded for all k, a stationary point must be "attractive" in a certain way, and the estimate must visit a certain region near the stationary point infinitely often [14].

5.3 Guidelines for Choosing SPSA Parameters

There are several parameters that must be specified for the SPSA algorithm and here we outline some of the guidelines in [15] for their choice. First, choose

$$\lambda_j = \frac{\lambda}{(\lambda_0 + j)^{\alpha_1}}$$

where $\lambda > 0$, $\lambda_0 > 0$, and $\alpha_1 > 0$, and

$$c_j = \frac{c}{j^{\alpha_2}}$$

where $c > 0$ and $\alpha_2 > 0$. However, if the θ_i have very different magnitudes you may want to use a different λ_j for each of the p dimensions.

Some actual values that have been found useful in applications are

$$\alpha_1 = 0.602$$

and

$$\alpha_2 = 0.101$$

which are effectively the lowest allowable ones that satsify theoretical conditions. However, values $\alpha_1 \in [0.602, 1)$ and $\alpha_2 \in [0.101, \frac{1}{6}]$ may work also. In fact $\alpha_1 = 1$ and $\alpha_2 = \frac{1}{6}$ are the "asymptotically optimal" values so if the algorithm runs for a long time it may be beneficial to switch to these values. With this choice, if the noise is significant you may need to choose λ smaller and c larger than in a low-noise case.

With the Bernoulli ± 1 choice for the components of $\Delta(j)$, set c to a level that is approximately the standard deviation of the noise $w(j)$ to keep the components of $g(\theta(j), j)$ from being too large in magnitude. If there is no noise term $w(j)$, then you should choose some small value $c > 0$. You can choose λ_0 to be approximately 10% of the maximum number of iterations and λ to try to achieve a certain amount of change in the cost function values at each iteration.

5.4 Design Example: SPSA for DSS Design

Here, we show how to use SPSA for the design of DSS strategies, by continuing the study in Section 4. There, we tuned the w_i parameters of the DSS strategy in Equation (8). Here, we will use the SPSA to automate the tuning of the parameters of the same DSS strategy.

5.4.1 Design Problem and Algorithm Design

Our goal is to obtain performance that is better than that which we obtained via manual tuning where we obtained a time average of the average values of the lengths of times waited of 3.2755 for $w_1 = 4$, $w_2 = 2$, $w_3 = 1$, and $w_4 = 4$. One approach would be to start with these values to initialize the parameter vector. Here, we do not do this in order to consider the case where we had done no a priori tuning. Here, we start with $w_i = 1$, $i = 1, 2, 3, 4$.

Recall that we had $N = 4$ and

$$\delta^1 = 1.05,\ \delta^2 = 1.15,\ \delta^3 = 1.25,\ \delta^4 = 1.35$$

These are bounds for appearance periods given in Figure 4. We have $\delta_s = 0.03$. We have

$$a_1 = 0.1,\ a_2 = 0.2,\ a_3 = 0.3,\ a_4 = 0.1$$

and this gives $\sum_{i=1}^{4} a_i = 0.7$.

For the SPSA algorithm we define the cost function to be the time average of the averge amount of time each emitter is ignored. We bound the parameter values to be between 0.1 and 10, $\alpha_1 = 0.602$, $\alpha_2 = 0.101$, $\lambda = 0.5$, $\lambda_0 = 10$, and $c = 0.25$. We allow the SPSA algorithm to run for 100 iterations.

5.4.2 SPSA Design Results

The results of running the algorithm are shown via a plot of the cost function in Figure 14 and a plot of the parameters values explored during the search in Figure 15. We see that the cost decreases from the intial value, then the algorithm searches, but does not improve the performance.

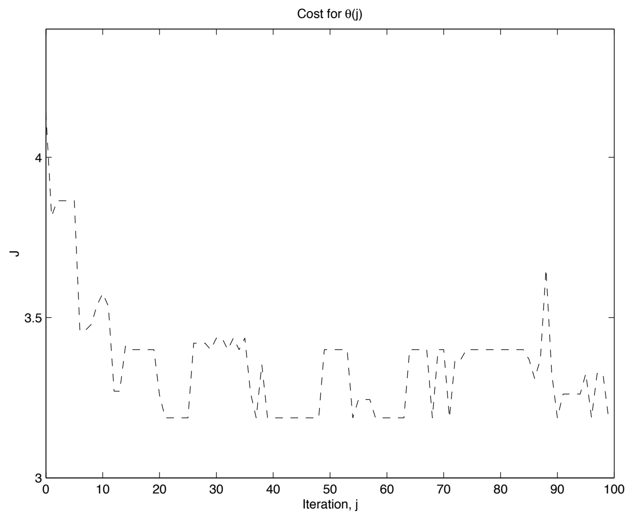


Figure 14: Values of cost function for SPSA for each iteration.

At iteration 22 the cost is 3.1871 and the parameter values are

1.91840.78880.45851.4590

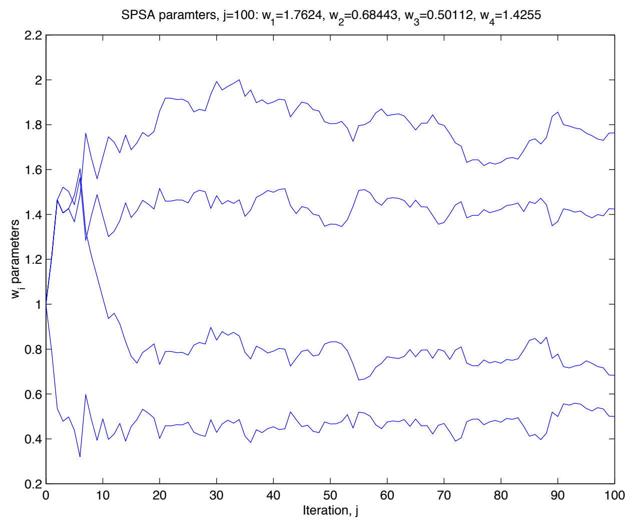


Figure 15: Parmeter values for DSS strategy explored by SPSA for each iteration.

Compare this with the values that we obtained via manual tuning. First, the SPSA did better in terms of reducing the time average of the average amount of time that emitters are ignored. Note, however, that the SPSA found a similar *relative* weighting to what we had found manually in the sense that it found that the strength of weighting, from highest to lowest, should be w_1, w_4, w_2, w_3. Recall that this ordering was arrived at in the manual tuning case by iterative tuning where we raised a weight if a certain emitter was not getting enough attention. SPSA seems to have tried to automatically achieve a sort of balancing of focusing in order to improve performance.

6 Stability Analysis of Dynamic Scan Schedulers

In this section the first three policies defined earlier will be proven to be stable, given that the capacity condition in Equation (5) holds. Stability of the policy defined in Equation (8) can be studied using a similar proof procedure. Moreover, it is simple to extend the analysis below to the case where priority parameters are added as discussed in Section (2.6). In the next section we will explain how to design a strategy that will stabilize *any* scheduling policy.

6.1 Stability Properties of DSSs Based on "Longest Ignored" or "Ignored More Than the Average"

We begin with the policies defined in Equations (1) and (6).

Theorem 1: Assume that Equation (5) holds. The DSS policies where the emitter that was ignored the longest time, or one that has been ignored longer than the average one, as defined in Equations (1) and (6), have the following properties:

- They are *stable* in that
$$\sup_{t\geq 0}\{T_i(t)\} < B_i, i \in E$$
for some $B_i > 0$, $i \in E$ so that they will not ignore any emitter for too long.
- A specific bound on the ultimate longest time that the receiver will ignore any emitter is given by
$$\lim_{t\to\infty} \sup \sum_{i=1}^{N} T_i(t) \leq \delta \left[\frac{\sum_{i=1}^{N} a_i}{\underline{a}} + \frac{\bar{a}N}{\underline{a}\left(1-\sum_{i=1}^{N} a_i\right)} \max_i \left\{ \frac{-a_i + \sum_{i=1}^{N} a_i}{a_i} \right\} \right]$$

where $\underline{a} = \min_i\{a_i\}$ and $\bar{a} \max_i\{a_i\}$.

Proof: Let

$$V(t) = \sum_{i=1}^{N} a_i T_i(t)$$

be a "Lyapunov-like" function (strictly speaking it is not a Lyapunov function because $T_i(t)$ is not the state of the system, e.g., due to the presence of the delays which make it an infinite dimensional system). You can think of $V(t)$ as the amount of work that the receiver needs to do at time t in order to obtain perfect information about all the emitters.

The proof to follow focuses on the policy where the emitter is chosen that has been ignored longer than the average one; however, a special case of this is when the one that is ignored the longest is chosen at each decision point so the above bounds hold for that DSS also.

Note that since $T_{i^*(k_r)}(D_{k_r+1}) = 0$ ($i^*(k_r)$ was the emitter that was just detected),

$$V(D_{k_r+1}) = \sum_{i=1}^{N} a_i T_i(D_{k_r+1}) = \sum_{i \neq i^*(k_r)}^{N} a_i T_i(D_{k_r+1})$$

Also,

$$\sum_{i \neq i^*(k_r)}^{N} a_i T_i(D_{k_r+1}) = \sum_{i \neq i^*(k_r)} a_i \left(T_i(D_{k_r}) + (D_{k_r+1} - D_{k_r})\right)$$

since when the receiver is focusing on emitter $i^*(k_r)$ the amount of time that all other emitters are ignored increases by $(D_{k_r+1} - D_{k_r})$ for each i, $i \neq i^*(k_r)$. Rearrange this equation to obtain

$$V(D_{k_r+1}) = V(D_{k_r}) - a_{i^*(k_r)} T_{i^*(k_r)}(D_{k_r}) + (D_{k_r+1} - D_{k_r}) \sum_{i \neq i^*(k_r)} a_i \tag{13}$$

Now, use Equation (3) to obtain

$$V(D_{k_r+1}) \leq V(D_{k_r}) - \alpha(i^*(k_r)) T_{i^*(k_r)}(D_{k_r}) + \beta(i^*(k_r)) \tag{14}$$

where

$$\alpha(i) = \frac{a_i \left(1 - \sum_{j=1}^{N} a_j\right)}{1 - a_i}$$

and

$$\beta(i) = \delta \frac{\left(-a_i + \sum_{j=1}^{N} a_j\right)}{1 - a_i}$$

Note that $\alpha(i) > 0$ and $\beta(i) > 0$ for all $i \in E$. To understand how Equation (14) is found, using Equation (3) note that since $\delta(D_{k_r}) \leq \delta$

$$V(D_{k_r+1}) \leq V(D_{k_r}) - a_{i^*(k_r)} T_{i^*(k_r)}(D_{k_r}) + \left[\sum_{j \neq i^*(k_r)} a_j\right] (1 - a_{i^*(k_r)})^{-1} \left[\delta + a_{i^*(k_r)} T_{i^*(k_r)}(D_{k_r})\right]$$

The term due to δ creates $\beta(i^*(k_r))$. For the remaining terms, besides $V(D_{k_r})$, by grouping we get

$$-a_{i^*(k_r)} \left(1 - \frac{\sum_{j \neq i^*(k_r)} a_j}{1 - a_{i^*(k_r)}}\right) T_{i^*(k_r)}(D_{k_r}) =$$

$$-a_{i^*(k_r)} \left(\frac{1 - \sum_{j=1}^{N} a_j}{1 - a_{i^*(k_r)}}\right) T_{i^*(k_r)}(D_{k_r})$$

and this is used to define $\alpha(i^*(k_r))$.

Next, notice that due to the definition of *either* DSS

$$\alpha(i^*(k_r))T_{i^*(k_r)}(D_{k_r}) \geq \alpha(i^*(k_r))\frac{1}{N}\sum_{i=1}^{N} T_i(D_{k_r})$$

and due to the definition of $\bar{a}$

$$\alpha(i^*(k_r))\frac{1}{N}\sum_{i=1}^{N} T_i(D_{k_r}) \geq \alpha(i^*(k_r))\frac{1}{N}\bar{a}^{-1}\sum_{i=1}^{N} a_i T_i(D_{k_r})$$

(since $\frac{a_i}{\bar{a}} \leq 1$). But notice that

$$\alpha(i^*(k_r))\frac{1}{N}\bar{a}^{-1}\sum_{i=1}^{N} a_i T_i(D_{k_r}) = \alpha(i^*(k_r))\frac{1}{N}\bar{a}^{-1}V(D_{k_r}) \tag{15}$$

Combine this with Equation (14) to get

$$V(D_{k_r+1}) \leq \left[1 - \bar{a}^{-1}N^{-1}\alpha(i^*(k_r))\right] V(D_{k_r}) + \beta(i^*(k_r)) \tag{16}$$

Subtract $\bar{a}N \max_i \frac{\beta(i)}{\alpha(i)}$ from both sides of Equation (16) and after a bit of algebra you get

$$\begin{aligned} V(D_{k_r+1}) - \bar{a}N \max_i \frac{\beta(i)}{\alpha(i)} \quad &\leq \quad \left[V(D_{k_r}) - \bar{a}N \max_i \frac{\beta(i)}{\alpha(i)}\right]\left[1 - \bar{a}^{-1}N^{-1}\alpha(i^*(k_r))\right] \\ &+ \quad \beta(i^*(k_r))\left[1 - \frac{\alpha(i^*(k_r))}{\beta(i^*(k_r))}\max_i \frac{\beta(i)}{\alpha(i)}\right] \end{aligned}$$

Focus for a moment on the last term in this equation, and notice that

$$\beta(i^*(k_r))\left[1 - \frac{\alpha(i^*(k_r))}{\beta(i^*(k_r))}\max_i \frac{\beta(i)}{\alpha(i)}\right] \leq 0$$

How do you get the last inequality? Note that $\beta(i) > 0$. If the $\max_i \frac{\beta(i)}{\alpha(i)}$ term is maximized at some value j, then clearly this value divided by any value considered in the maximization will be greater than or equal to one.

Now, we have

$$\left[V(D_{k_r+1}) - \bar{a}N \max_i \frac{\beta(i)}{\alpha(i)}\right] \leq \left[V(D_{k_r}) - \bar{a}N \max_i \frac{\beta(i)}{\alpha(i)}\right]\left[1 - \bar{a}^{-1}N^{-1}\alpha(i^*(k_r))\right] \tag{17}$$

But, notice that the second term on the right-hand side of this equation

$$\left[1 - \bar{a}^{-1}N^{-1}\alpha(i^*(k_r))\right] = 1 - \bar{a}^{-1}N^{-1}\min_i\left\{\frac{a_i\left(1 - \sum_{j=1}^{N} a_j\right)}{1 - a_i}\right\} = 1 - \bar{a}^{-1}N^{-1}\left[\frac{\underline{a}\left(1 - \sum_{j=1}^{N} a_j\right)}{1 - \underline{a}}\right]$$

Notice that

$$0 < \frac{\left(1 - \sum_{j=1}^{N} a_j\right)}{1 - \underline{a}} < 1$$

and

$$0 < \frac{\underline{a}}{\bar{a}} < 1$$

so that

$$0 < 1 - \bar{a}^{-1}N^{-1}\alpha(i^*(k_r)) < 1$$

which makes the mapping in Equation (17) contractive so that

$$\lim_{k_r \to \infty} \sup \left\{ V(D_{k_r}) - \bar{a} N \max_i \frac{\beta(i)}{\alpha(i)} \right\} = 0$$

But this (ultimate) bound is in terms of only the decision points D_{k_r}, $k_r = 1, 2, 3, \ldots$ Due to the delay δ, the $T_i(t)$ values can rise higher at times t not at the decision points. However, for $D_{k_r} \leq t \leq D_{k_r+1}$

$$V(t) \leq V(D_{k_r} + \delta)$$

But notice that

$$V(D_{k_r} + \delta) = \sum_{i=1}^{N} a_i T_i(D_{k_r} + \delta) = \sum_{i=1}^{N} a_i T_i(D_{k_r}) + \delta \sum_{i=1}^{N} a_i = V(D_{k_r}) + \delta \sum_{i=1}^{N} a_i$$

This gives us

$$\lim_{t \to \infty} \sup V(t) \leq \delta \sum_{i=1}^{N} a_i + \bar{a} N \max_i \frac{\beta(i)}{\alpha(i)}$$

and since

$$\lim_{t \to \infty} \sup \sum_i T_i(t) \leq \frac{1}{\underline{a}} \lim_{t \to \infty} \sup V(t)$$

we know

$$\lim_{t \to \infty} \sup \sum_i T_i(t) \leq \frac{\delta \sum_{i=1}^{N} a_i}{\underline{a}} + \frac{\bar{a} N}{\underline{a}} \max_i \frac{\delta \left(-a_i + \sum_{j=1}^{N} a_j \right)}{a_i \left(1 - \sum_{j=1}^{N} a_j \right)}$$

which gives the desired result. ■

Note that since the above bound for Theorem 1 may be conservative for some situations, it would be of interest to specify "tight" bounds since this would provide good guarantees for bounding the maximum time that an emitter is ignored.

6.2 Stability Properties of DSSs Based on Emitters That are "Difficult to Find"

Next, we will study the stability properties of the other policy defined in the last section where we get a different bound on the maximum length of time that an emitter will be ignored by the receiver. The analysis, is however, only slightly different and depends on the above proof.

Theorem 2: Assume that Equation (5) holds. The DSS policies defined in Equation (7) has the following properties:

- It is *stable* in that

$$\sup_{t \geq 0} \{T_i(t)\} < B_i, i \in E$$

  for some $B_i > 0$, $i \in E$ so that it will not ignore any emitter for too long.

- A specific bound on the ultimate longest time that the receiver will ignore any emitter is given by

$$\lim_{t \to \infty} \sup V(t) \leq \frac{\delta(N-1)}{1 - \sum_{i=1}^{N} a_i} \left(-\underline{a} + \sum_{i=1}^{N} a_i \right) + \delta \sum_{i=1}^{N} a_i \leq \delta \left[\sum_{i=1}^{N} a_i \right] \frac{N - \sum_{i=1}^{N} a_i}{1 - \sum_{i=1}^{N} a_i}$$

  where $\underline{a} = \min_i\{a_i\}$ and $\bar{a} \max_i\{a_i\}$.

Proof: Use the ideas from the proof for Theorem 1 and note that Equation (15) in this case is

$$\alpha(i^*(k_r))T_{i^*(k_r)}(D_{k_r}) = \frac{\alpha(i^*(k_r))}{a_{i^*(k_r)}}a_{i^*(k_r)}T_{i^*(k_r)}(D_{k_r}) \geq \frac{\alpha(i^*(k_r))}{(N-1)a_{i^*(k_r)}}\sum_{i=1}^{N} a_i T_i(D_{k_r}) = \frac{\alpha(i^*(k_r))}{(N-1)a_{i^*(k_r)}}V(D_{k_r})$$

The $N-1$ factor appears, rather than N, for one i, $T_i = 0$. To complete the proof simply take the same approach as in the remainder of the proof of Theorem 1, below Equation (15). ■

So, do these bounds give an indication of which of the three policies is "best"? Unfortunately, they generally do not since the bounds can be conservative. It is for this reason that simulation analysis is generally needed to analyze the *performance* of particular policies and determine which is best for a particular emitter environment.

7 Universal Stabilizing Mechanism for Dynamic Scan Schedules

At times there is significant knowledge about the emitter environment and receiver that is relevant to the design of DSSs. There is then a natural tendency to incorporate this information in the specification of the DSS, often in the form of "scheduling heuristics." The problem with this approach, however, is that the resulting policies may end up being somewhat nonstandard and there may be concerns about whether they will be stable.

Fortunately, the approach in [7] to specifying a "universal stabilizing mechanism" (USM) for any scheduling policy actually holds for the scan scheduling problem (actually, the approach in [7] was developed for a fixed size delay and we have a time-varying but bounded delay; however, the proofs there can be directly extended to our case with no difficulty). This mechanism can then be applied to *any* heuristically constructed DSS, and you will be ensured that the overall policy will be stable. It would also be possible to use the "stream modifier" approach that is described in [11], but the approach taken here is more natural for this application.

In this section we introduce the USM from [7] and briefly discuss its use for the "control-based" DSS in [18]. In the next section we introduce some schedulers that exploit emitter domain information to try to enhance scheduler performance, and which can be stabilized by the USM introduced here.

7.1 Universal Stabilizing Mechanism

This section is based on [7]. The key fact is that for the resource allocation problems we consider here, as long as the capacity condition is satisfied, it is possible to define a USM which when used to supervise a scheduling policy will always result in stable operation. To define the USM, let Q denote a first-come first-serve (FCFS) priority queue for emitters that have been ignored for a long time. For instance, if emitter i becomes too large we will have criteria for entering the queue at some time t'. If emitters i and j are in the priority queue, and j entered it before i,

$$Q = (\ldots, i, j, \ldots)$$

then when this priority queue is serviced emitter j will be taken off the queue and scanned before emitter i (the "tail" of the queue is the first emitter listed after the "(" and the "head" of the queue is the emitter listed just before the ")" in the definition of Q above). We need some additional parameters to specify the USM. Let $L > 0$ be a large number satisfying

$$L > \frac{N\delta}{1-\rho}$$

where ρ is specified in Equation (5), N is the number of emitters, and δ is the bound on the maximum delay. Next, let

$$H_i > 0, \ i \in E$$

denote a set of parameters, the interpretation of which will become clear as we define the USM.

The USM is implemented by the following set of rules:

1. *Truncation Rule:* The receiver can process no emitter i longer than La_i time units. This means that if at time $t' + \delta_s + \delta_e(t')$ the receiver starts to try to detect emitter i, then it can only try to detect it no longer than up to the time $t' + \delta_s + \delta_e(t') + La_i$. If detection occurs before that time, then the policy acts as usual and selects another emitter to focus on. If, however, it has not yet detected emitter i by this time it is forced to make a new decision (which could entail switching emitters). Note that if it does switch to another emitter we assume that the progress it had made on emitter i is used, but that the time since it was last detected, T_i, begins to increase again.

2. *Rule for entering Q:* Emitter i enters the tail of the priority queue Q at time t if: we have not just decided to scan for i or are currently scanning to detect i, and $T_i(t) > H_i$ (hence, the H_i are thresholds for when an emitter is placed in the queue).

3. *Emitter selection rule:* If Q is not empty when the receiver has finished a scan for an emitter (either by achieving $T_i = 0$ or via rule 1 above), then the emitter at the head of the priority queue Q (i.e., FCFS) is chosen.

4. *Rule for leaving Q:* An emitter i leaves Q at the time $t' + \delta_s + \delta_e(t')$ where t' is the time point when emitter i was selected by rule 3.

5. *Rule for processing-time for an emitter from Q:* If emitter i from Q is chosen for a scan, then beyond the time t' defined in rule 4 it is processed for La_i time units unless it is detected (i.e., $T_i = 0$) before this time elapses.

Notice that this is not simply another policy. It actually defines a "supervisor" for any DSS policy (e.g., ones that exploit heuristic information from the problem domain) that ensures it will result in stable operation. If you have constructed a stable policy, and you choose L and the H_i, $i \in E$, large enough, then the USM will *never* intervene. The USM simply truncates the processing of emitters that are not found fast enough, and via Q makes sure that emitters that have been ignored for too long will get attention. How do we pick L and the H_i parameters? If you pick $H_i = 0$, $i \in E$, then the USM simply enforces a type of FCFS policy on emitters with $T_i > 0$, but it stops processing any emitter that is focused on too long. In this case, the USM always intervenes. As you increase the size of L and the parameters H_i, $i \in E$, the USM intervenes less frequently.

What is the value of the USM? In a sense it frees the designer of DSSs from being concerned about the stability of the myriad of possible DSSs (but of course, the stability analysis of Section 6 is still useful, particularly if the bounds that are obtained are tight or if the analysis helps to clarify how to design the policy). You can adopt a design philosophy where you construct a very complicated DSS, possibly exploiting heuristic ideas about how to achieve the best performance. Then, you can augment such policies with the USM and be assured that you will obtain stable operation. Essentially, the USM allows the designer to focus on the design of DSSs to improve scheduling *performance*. To illustrate this point, in the next two sections we will study the design and performance of two heuristic policies, ones based on our intuitions about the problem domain. First, however, we briefly discuss how the USM could be used for another scheduling method.

7.2 Stabilization of the Control Based Dynamic Scan Scheduler

The USM strategy described in the previous subsection applies directly to the scheduling (control) strategy defined in [18] and hence shows a way to ensure that policy is stable (of course, provided that the capacity condition is satisfied). Augmenting the control-based strategy with the USM ensures that if stability becomes a problem, the control-based strategy is abandoned and proper attention is given to emitters that have not been visited for a long time. If no emitter has been ignored for too long, the scheduling strategy in [18] would be used again. If, however, the control-based strategy is stable and the parameters of the USM are chosen properly, then the USM will never intervene (and hence it is as if the USM were not even present).

8 Model Predictive Control for Dynamic Scan Scheduling

Once stability is assured, it is critical to focus on how to achieve good performance (stability does *not* imply good performance). Good performance is achieved via design of good dynamic scan schedulers. It is a general principle in resource allocation that incorporation of problem domain informtation into the scheduler can help it make better decisions that then lead to better performance. In this section we discuss a general methodology for incorporating emitter environment and receiver information into the DSS. This approach is based on (i) using models of aspects of the problem domain to predict the results of making various scheduling decisions, (ii) selecting between the alternatives the approach that minimizes some type of (short-term) performance index, and (iii) implementing the scheduling decision that is predicted to achieve the best performance. In this section we will discuss an approach to such a "model predictive control" approach to dynamic scan scheduling.

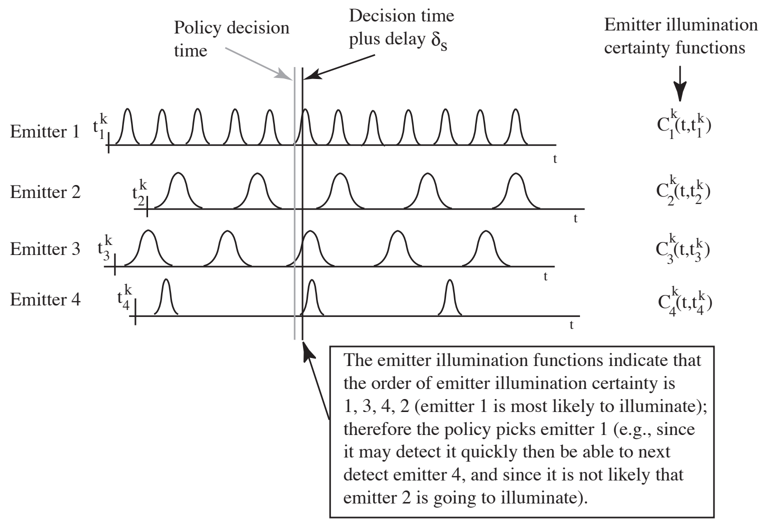
8.1 Model Predictive Control Via Active Emitter Table Information

The policies consider up to this point do not incorporate significant a priori information that may be available about the likely frequency of emitter illuminations (but the policy in Equation (8) did use information about delays to predict and decide which emitter to focus on). For example, if the receiver has identified the radar/emitter type it may have a good guess of when the next illumination time will be, or if it has observed a fixed pattern it may have a guess. Without using such a priori information that may be contained in a list of information about expected emitters for the mission and their characteristics (an AET), in their current form the DSSs may go scan for an emitter even though it is unlikely that it will be found for some period of time, during which the receiver could more profitably search for other emitters.

How can such a priori information be incorporated? This is a topic for future study, but it certainly appears to be feasible to proceed in the following manner. First, suppose that we use a "certainty of illumination" function

$$C_i^k(t, t_i^k)$$

for each emitter i, that is defined along the time-line $t \geq t_i^k$ starting from the time t_i^k when emitter i was last detected (i.e,. from the time that the emitter was detected for the k^{th} time). Suppose that this function has values in the range of $[0, 1]$, with 0 representing that it is unlikely that there will be a illumination, 0.5 representing uncertainty about whether there will be a illumination, and 1 representing that you are certain that there will be a illumination (based on a priori information). Now, suppose we define a policy that at each decision point simply picks the emitter to scan for that is most likely to illuminate (and perhaps taking into account any delay in switching the receiver to a new frequency band). See Figure 16.



The emitter illumination functions indicate that the order of emitter illumination certainty is 1, 3, 4, 2 (emitter 1 is most likely to illuminate); therefore the policy picks emitter 1 (e.g., since it may detect it quickly then be able to next detect emitter 4, and since it is not likely that emitter 2 is going to illuminate).

Figure 16: Dynamic scan scheduler scheme that exploits information from an active emitter table.

In the figure notice that there are illumination certainty functions for four emitters. Emitter one is predicted to be a higher frequency illuminator, and the width of each of the humps quantifies the certainty of occurrence of illumination; hence illuminations are most certain at the peaks. Notice that emitters 2 and 3 are predicted to have a similar (lower) frequency illuminations, but the precise timing of the illuminations is not as certain and this is quantified via the spreads of the humps being larger. Emitter 4 is predicted to be a lower frequency illuminator, but the certainties of when the illuminations will occur is similar to that specified for emitter 1. Note that the parameters defining the $C_i^k(t, t_i^k)$ functions (e.g., the points where the peaks occur and the spreads) could be estimated in some situations by some other on-board aircraft system, and then the AET table can be updated and the $C_i^k(t, t_i^k)$ functions used by the scheduler could be changed to be consistent with the AET. Finally, note that these certainties could be scaled by emitter "priorities" so that the policy could choose to scan for the highest priority emitter that is likely to produce an illumination.

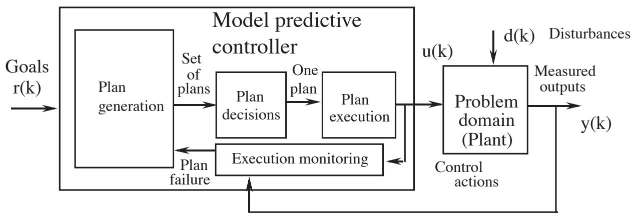
Will this result in a stable policy? No, not if that is all that is used in the DSS. It could be that you have bad a priori information so that bad guesses are made and the illuminations are never found for an emitter and so the length of time that it is ignored goes to infinity (representing that it ultimately it knows nothing about the emitter). We can, however use the USM of the previous section to ensure stable operation.

8.2 Discussion: General Model Predictive Control Strategies

In most engineering applications we can simulate, to a reasonable degree of accuracy, the domain in which we make decisions. For instance, in this paper we have simulated the emitter environment in Section 3. The actual emitter environment is certainly somewhat different that what our simulations would lead us to believe. Let us suppose, however, that we can simulate the emitter environment reasonably well, at least in its broad characteristics. Furthermore, suppose that we can simulate this model of the emitter environment *in real-time* on board the aircraft. Would such a simulation provide useful information to help decide which emitter to focus on? Below, we consider this question by discussing one way to incorporate a simulated emitter environment into a scheduling policy.

Suppose that we use the model of the emitter environment to predict how the receiver will perform using different policies or orders of scanning for emitters. Suppose that we use the model to predict M different behaviors that result from M different "partial scan schedules" of length N (a specification of a sequence of a sequence of N emitters to scan). The strategy is shown in Figure 17. As shown in the figure, for the MPC strategy we rank order the M different partial scan schedules and choose the best one, and then we scan for the emitter specified as the first one to scan for in the best partial scan schedule. The process repeats at the next decision point, i.e., when that emitter is detected. We think of N as specifying a "receding horizon" or length of time we predict ahead in time. For a very uncertain emitter environment it typically does not make sense to make N very large since the predictions will typically become more inaccurate as we predict farther ahead in time. If, however, your model is good and you have sufficient computational resources you may want to predict into the future for longer periods of time so that the best possible emitter is chosen to focus on.

You can think of the general MPC DSS as a more sophisticated version of the AET-based strategy discussed in the last section. Clearly, if information were gathered on-line during a mission you may be able to profitably update the model that is used in the general MPC strategy (this would then lead to "adaptive model predictive control" where you would have a model estimator operating in tandem with the prediction strategy). The AET-based strategy discussed in the last section would provide a simple learning/planning strategy via the estimation of the $C_i^k(t, t_i^k)$ functions (e.g., via estimation of the peaks and spreads of the humps on those functions). Clearly it is not difficult to incorporate an emitter priority scheme in any of these approaches. Will MPC-type policies result in stable scheduling? In general, probably not, since you can never have a perfect model of the environment (e.g., since it can change over time, may be noisy, or certain aspects about it may be very difficult to model). However, once again we could use the USM to ensure that we obtain stable operation.



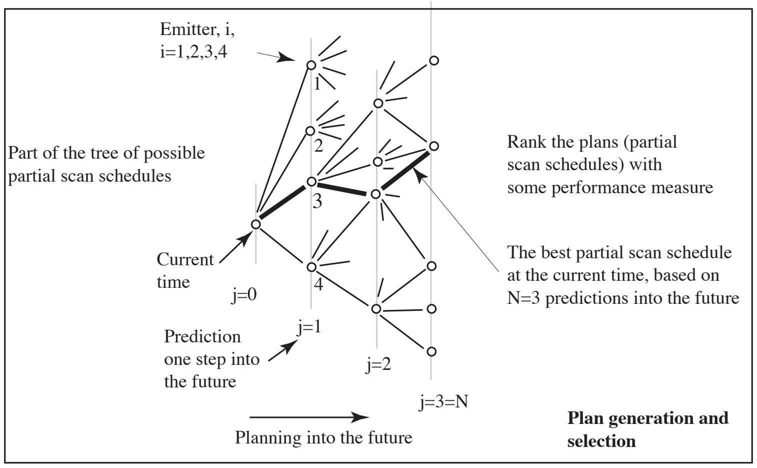

Emitter, i, i=1,2,3,4
1
2
3
4
Part of the tree of possible partial scan schedules
Rank the plans (partial scan schedules) with some performance measure
Current time
j=0
j=1
j=2
j=3=N
The best partial scan schedule at the current time, based on N=3 predictions into the future
Prediction one step into the future
Planning into the future
Plan generation and selection

Figure 17: Model predictive control for dynamic scan scheduling.

9 Design and Stability Analysis of Distributed Dynamic Scan Schdulers (DDSS)

When there are multiple receivers and a communication channel in betweeen the platforms (e.g., aircraft) holding the receivers there is the possibility of coordinating actions of pointing at emitters so that the group can perform better than if there were no communication channel. In this section we show how to model this possibility and derive some stability conditions for this case. In fact, here we will in a sense generalize the earlier results by assuming that each emitter has a priority that is known a priori, and that we call p_i, $i \in E$. This value scales all the T_i values and the capacity condition of earlier becomes

$$\rho = \sum_{i=1}^{N} p_i a_i < 1, p_i \in (0,1], i \in P \tag{18}$$

and *all the earlier results still hold,* with appropriate modifications made to the bounds. Here, we will also use emitter priority and we will assume that there are a fixed number of M receivers. We seek to develop strategies that use only a minimal amount of information from other platforms to minimize the communication overhead; clearly, it is generally the case that with better communication channels you should be able to achieve better cooperation and hence a more effective strategy.

9.1 Multiple Receiver Capacity Condition and Decision Timing

First of all, we define the "capacity condition" for the M receiver case as

$$\rho = \sum_{i=1}^{N} a_i p_i < M \tag{19}$$

where $1/a_i$ and p_i, $i \in E$ are the same as earlier. This equation clearly shows the potential of collaboration. There is the potential to increase the processing rate of the system using more receivers.

Let the set of receivers be denoted by $R = \{1, 2, \ldots, M\}$, $M < N$. Moreover, we define the set $U(t) \subset E$ as the set of emitters not processed by any receiver at the current time t while the set $U_j^a(t) = \{i_j^*(t)\} \cup U(t)$ is the subset of emitters available for processing by receiver $j, j \in R$. Also, $i_j^*(t)$ is the emitter currently being processed by receiver j. In addition to that, we can similarly define the set $A(t)$ as the set of emitters processed by a group of receivers at the current time t. Note that $P = U \cup A$.

It is important to clarify some issues related to the timing of decisions. Whenever a receiver $j, j \in R$ takes a decision, it proceeds to broadcast the new set $U(t)$ available to the rest of receivers. In fact, for the theory below we will first assume that the set $U(t)$ can be updated instantaneously, which is equivalent to assuming that we have no communication delays. After the theorem, however, we will explain that inclusion of a bounded delay will *not* destroy stability properties; it will simply raise the bounds.

Let $D_{k_r^j}$ be the time when receiver j, $j \in R$, decides to focus on an emitter, and assume that $D_{1^j} = 0$. Let $D_{k_r^j+1}$ be the next decision point for receiver j. Moreover, this is when the detection of the last emitter occurred on receiver j. Just as in the $M = 1$ case, we need a relationship between D_{k_r} and D_{k_r+1}. Recall that all M recievers will inform to each other about the new set $U(t)$ whenever they make a decision. The fundamental reason that the receivers need to transmit the sets $U(t)$ rests upon the fact that each receiver j will focus on a new emitter at time $D_{k_r^j+1}$ based on the emitters available in the set $U_j^a(t)$. Hence, we need to introduce a new decision time denoted as $D_{k_r^{j*}}$. This is the closest decision time taken by any reciever $m, m \neq j$ to the decision time $D_{k_r^j+1}$. Note that if any receiver m, $m \in R$, $m \neq j$, does not take a decision between times $D_{k_r^j}$ and $D_{k_r^j+1}$ then $D_{k_r^{j*}}$ is just equal to $D_{k_r^j}$. But it could be the case that either one or some receivers m take decisions between the $D_{k_r^j}$ and $D_{k_r^j+1}$ times. Therefore, we know that $D_{k_r^j} \leq D_{k_r^{j*}} < D_{k_r^j+1}$.

The times when receiver j makes decisions, based this time $D_{k_r^{j*}}$ and provided that $D_{k_r^j} \leq D_{k_r^{j*}} \leq D_{k_r^j} + \delta(D_{k_r^j})$, are given by

$$D_{k_r^j+1} = D_{k_r^{j*}} + (D_{k_r^j} + \delta(D_{k_r^j}) - D_{k_r^{j*}}) + a_{i_j^*(k_r^{j*})} T_{i_j^*(k_r^{j*})}(D_{k_r^{j*}}) + \left(D_{k_r^j+1} - D_{k_r^{j*}}\right) a_{i_j^*(k_r^{j*})} p_{i_j^*(k_r^{j*})} \tag{20}$$

We know that $D_{k_r^j} + \delta(D_{k_r^j}) - D_{k_r^{j*}} \leq \delta(D_{k_r^j})$, so we can substitute this in Equation (20) to obtain after a little algebra

$$D_{k_r^j+1} - D_{k_r^{j*}} \leq \frac{\delta + a_{i_j^*(k_r^{j*})} T_{i_j^*(k_r^{j*})}(D_{k_r^{j*}})}{1 - a_{i_j^*(k_r^{j*})} p_{i_j^*(k_r^{j*})}} \tag{21}$$

Now, we need to study the case when $D_{k_r^j} + \delta(D_{k_r^j}) < D_{k_r^{j*}} < D_{k_r^j+1}$. For this case,

$$D_{k_r^j+1} = D_{k_r^{j*}} + a_{i_j^*(k_r^{j*})} T_{i_j^*(k_r^{j*})}(D_{k_r^{j*}}) + a_{i_j^*(k_r^{j*})} p_{i_j^*(k_r^{j*})} \left(D_{k_r^j+1} - D_{k_r^{j*}}\right)$$

Solving the last equation for $D_{k_r^j+1} - D_{k_r^{j*}}$, we get that

$$D_{k_r^j+1} - D_{k_r^{j*}} = \frac{a_{i_j^*(k_r^{j*})} T_{i_j^*(k_r^{j*})}(D_{k_r^{j*}})}{1 - a_{i_j^*(k_r^{j*})} p_{i_j^*(k_r^{j*})}} \tag{22}$$

We can show that Equation (22) has the same bound as Equation (21). Hence, Equation (22) can be written as

$$D_{k_r^j+1} - D_{k_r^{j*}} \leq \frac{\delta + a_{i_j^*(k_r^{j*})} T_{i_j^*(k_r^{j*})}(D_{k_r^{j*}})}{1 - a_{i_j^*(k_r^{j*})} p_{i_j^*(k_r^{j*})}}$$

Therefore, if $D_{k_r^j} \leq D_{k_r^{j*}} < D_{k_r^j+1}$ then we can bound $D_{k_r^j+1} - D_{k_r^{j*}}$ by Equation (21).

9.2 DDSS Strategies: Cooperative Scheduling of Scans

In this section we will define several strategies for DDSS. Later, we will perform stability analysis for these strategies.

9.2.1 Focus on M Emitters Ignored for the Longest Time

The DDSS strategy chooses to focus on emitter $i_j^*(k_r^j)$ such that

$$T_{i_j^*(k_r^j)}(D_{k_r^j}) \geq T_{i_j}(D_{k_r^j}), \forall i_j \in U_j^a(D_{k_r^j}), \forall j \in R \tag{23}$$

and focuses on it until it detects it.

Therefore, each receiver chooses at each time instant different emitters to focus on, and each one will choose a new emitter based on the current $U_j^a(t)$ set. In this particular case, receivers will choose the M emitters that have been more ignored the most since last detection. If there is more than one maximizer for each receiver, then the DDSS strategy will simply choose one of these at random.

9.2.2 Focus on M Emitters Ignored More Than the Average One

For this case, each receiver, at $D_{k_r^j}$, chooses to focus on emitter $i_j^*(k_r^j)$ such that

$$T_{i_j^*(k_r^j)}(D_{k_r^j}) \geq \frac{1}{N-M} \sum_{i \in U_j^a(D_{k_r^j})} T_i(D_{k_r^j}), \forall i_j \in U_j^a(D_{k_r^j}), \forall j \in R \tag{24}$$

and focuses on it until it detects it. Once again, each receiver focuses on different emitters at the same time, choosing this time M emitters from the set $U_j^a(t)$ that have been ignored for more time than the average emitter has. If there are several emitters that satisfy this equation then the strategy simply choose one randomly.

9.2.3 Focus on M Emitters Expected to Be Most Difficult to Detect

Suppose that receiver j chooses emitter $i_j^*(k_r^j)$ to focus on at time $D_{k_r^j}$ if

$$i_j^*(k_r^j) = \arg\max_{i_j} \left\{ w_{i_j} \left(\frac{T_{i_j}(D_{k_r^j}) + (\delta_s^j + \delta^{i_j}) p_{i_j}}{\frac{(1-a_{i_j}p_{i_j})}{a_{i_j}}} \right) \right\}, \forall i_j \in U_j^a(D_{k_r^j}), \forall j \in R \tag{25}$$

where $w_{i_j} > 0$ are weighting factors.

9.3 Stability Analysis of DDSS Strategies

Next, we will study the stability properties of the DDSS strategies defined by Equations (23) and (24). Analysis of the case for M emitters that are the most difficult to detect proceeds in a similar manner.

Theorem 3: Assume that Equation (19) holds. Moreover, assume that $a_i p_i < M/N, \forall i \in E$, and that $M < N$. The DDSS strategies where the emitter that was ignored the longest time, or one that has been ignored longer than the average one, as defined in Equations (23) and (24) have the following properties:

- They are *stable* in that
$$\sup_{t \geq 0} \{T_i(t)\} < B_i, i \in P$$
for some $B_i > 0$, $i \in E$ so that they will not ignore any emitter for too long.

- A specific bound on the ultimate longest time that receivers will ignore any emitter is given by
$$\lim_{t\to\infty} \sup \sum_{i=1}^{N} T_i(t) \leq \frac{\delta}{\underline{a}} \left(M a_i p_i + \sum_{i=1}^{N} a_i p_i - M \underline{ap} \right) + (N-M) \frac{\bar{a}}{\underline{a}} \max_i \left[\frac{\delta(\sum_{i=1}^{N} a_i p_i - M \underline{ap})}{a_i (1 - a_i p_i - \frac{1}{M} \sum_{i=1}^{N} a_i p_i + \underline{ap})} \right]$$
where $\underline{a} = \min_i\{a_i\}$, $\underline{ap} = \min_i\{a_i p_i\}$, and $\bar{a} = \max_i\{a_i\}$

Proof: Let

$$V(t) = \sum_{i=1}^{N} a_i T_i(t)$$

be a "Lyapunov-like" function. The proof to follow focuses on the strategy where the emitter i_j is chosen by receiver j, $j \in R$ that has been ignored longer than the average one; however, a special case of this is when the one that is ignored the longest is chosen at each decision point so the above bounds hold for that DDSS strategy also.

Now, we try to split the $V(t)$ function into two terms: the emitters that are currently being attended $(U(t))$ and the ones that are not being attended $(A(t))$. Thus,

$$V(t) = \sum_{i=1}^{N} a_i T_i(t) = \sum_{i \in U(t)} a_i T_i(t) + \sum_{i \in A(t)} a_i T_i(t)$$

Since there are M receivers focusing on different emitters at the same time, we try to define a "Lyapunov-like" "sub-function" for each receiver such that when we sum the sub-functions we obtain $V(t)$. Hence, we look for an equivalence term of both sets $U(t)$ and $A(t)$. Then,

$$\sum_{i \in U(t)} a_i T_i(t) = M \sum_{i \in U(t)} \frac{a_i T_i(t)}{M} = \sum_{j=1}^{M} \left(\sum_{i \in U_j(t)} \frac{a_i T_i(t)}{M} \right)$$

and

$$\sum_{i \in A(t)} a_i T_i(t) = \sum_{j=1}^{M} a_{i_{j^*}} T_{i_{j^*}}(t)$$

Recall that $\{i_j^*\}$ is the emitter being currently attended by receiver j. We are ready to write a new equation for $V(t)$ as follows:

$$V(t) = \sum_{i \in U(t)} a_i T_i(t) + \sum_{i \in A(t)} a_i T_i(t) = \sum_{j=1}^{M} \left(a_{i_{j^*}} T_{i_{j^*}}(t) + \sum_{i \in U_j(t)} \frac{a_i T_i(t)}{M} \right) \tag{26}$$

Define the sub-function $V_j(t)$ for each receiver j through the following equation:

$$V_j(t) = a_{i_{j^*}} T_{i_{j^*}}(t) + \sum_{i \in U_j(t)} \frac{a_i T_i(t)}{M} \tag{27}$$

It is important to note that the sub-function $V_j(t)$ computes the amount of work that every receiver needs to do in order to attend at time t the emitter that is currently attending plus the emitters that are unattended. However, observe the fact that the unattended set is being artificially shared among all receivers when the set $U_j(t)$ is divided by the number of receivers (see second term of Equation (27)).

Note that since $T_{i_j^*(k_r^{j^*})}(D_{k_r^j+1}) = 0$ ($i_j^*(k_r^{j^*})$ was the emitter that was just detected by detector j,$j \in R$), and that $D_{k_r^{j^*}}$ is the closest decision point to $D_{k_r^j+1}$ for detectors j, $j \in R$ where $k_r^{j^*} = 1, 2, 3, \ldots$,

$$\begin{aligned} \sum_{j=1}^{M} V_j(D_{k_r^j+1}) &= \sum_{j=1}^{M} \left[a_{i_j^*(k_r^{j^*})} T_{i_j^*(k_r^{j^*})}(D_{k_r^j+1}) + \sum_{i \in U_j(D_{k_r^j+1})} \frac{a_i T_i(D_{k_r^j+1})}{M} \right] \\ &= \sum_{j=1}^{M} \left[\sum_{i \in U_j(D_{k_r^j+1})} \frac{a_i T_i(D_{k_r^j+1})}{M} \right] \end{aligned} \tag{28}$$

There is a particular difference between Equations (26) and (28). While Equation (26) is evaluated at a specific time t, Equation (28) is evaluated at the M decision times of every receiver. Since these M decision times can occur at different times, then it is clear that there is a time misalignment among all receiver decisions. Also,

$$\sum_{j=1}^{M}\left[\sum_{i\in U_j(D_{k_r^j+1})}\frac{a_iT_i(D_{k_r^j+1})}{M}\right]=\sum_{j=1}^{M}\left\{\sum_{i\in U_j(D_{k_r^{j*}})}\left[\frac{a_i}{M}T_i(D_{k_r^{j*}})+\frac{a_i}{M}p_i(D_{k_r^j+1}-D_{k_r^{j*}})\right]\right\}$$

Recall that $D_{k_r^{j*}}$ is the closest decision time taken by any receiver to the decision time $D_{k_r^j+1}$. For this reason, we can assure that $U_j(D_{k_r^j+1})=U_j(D_{k_r^{j*}})$ since the unattended set U will not change in t, $D_{k_r^{j*}}\leq t\leq D_{k_r^j+1}$. Now, we use Equations (21) and (27), and the fact that $\delta(D_{k_r})\leq\delta$ to obtain

$$\begin{aligned}\sum_{j=1}^{M}V_j(D_{k_r^j+1}) &\leq \sum_{j=1}^{M}\left\{V_j(D_{k_r^{j*}})-a_{i_j^*(k_r^{j*})}T_{i_j^*(k_r^{j*})}(D_{k_r^{j*}})+\sum_{i\in U_j(D_{k_r^{j*}})}\frac{a_ip_i}{M}\left(\frac{\delta+a_{i_j^*(k_r^{j*})}T_{i_j^*(k_r^{j*})}(D_{k_r^{j*}})}{1-a_{i_j^*(k_r^{j*})}p_{i_j^*(k_r^{j*})}}\right)\right\}\\ &\leq \sum_{j=1}^{M}\left\{V_j(D_{k_r^{j*}})-a_{i_j^*(k_r^{j*})}T_{i_j^*(k_r^{j*})}(D_{k_r^{j*}})\left(1-\frac{\sum_{i\in U_j(D_{k_r^{j*}})}\frac{a_ip_i}{M}}{1-a_{i_j^*(k_r^{j*})}p_{i_j^*(k_r^{j*})}}\right)+\frac{\delta\sum_{i\in U_j(D_{k_r^{j*}})}\frac{a_ip_i}{M}}{1-a_{i_j^*(k_r^{j*})}p_{i_j^*(k_r^{j*})}}\right\}\end{aligned}$$

Now, we define $\alpha(i_j)$ and $\beta(i_j)$

$$\alpha(i_j)=a_{i_j}\left(1-\frac{\sum_{i\in U_j(D_{k_r^{j*}})}\frac{a_ip_i}{M}}{1-a_{i_j}p_{i_j}}\right)$$

and

$$\beta(i_j)=\frac{\delta\sum_{i\in U_j(D_{k_r^{j*}})}\frac{a_ip_i}{M}}{1-a_{i_j}p_{i_j}}$$

It is clear that $\beta(i_j)>0$ and we need to find out whether $\alpha(i_j)$ is also greater than zero. We know that

$$0<a_ip_i<\frac{M}{N} \tag{29}$$

Also,

$$0<\sum_{i=1}^{N}a_ip_i<M$$

$$0<\sum_{i\in U_j}a_ip_i<\frac{M(N-M)}{N}$$

$$0<\frac{1}{M}\sum_{i\in U_j}a_ip_i<\frac{N-M}{N} \tag{30}$$

Since the term $a_{i_j}>0$ in the variable $\alpha(i_j)$, we just need to study whether $1-\frac{\sum_{i\in U_j(D_{k_r^{j*}})}\frac{a_ip_i}{M}}{1-a_{i_j(k_r^{j*})}p_{i_j(k_r^{j*})}}>0$. But, we know that

$$1-\frac{\max_i(\sum_{i\in U_j}\frac{a_ip_i}{M})}{1-\max_i(a_ip_i)}<1-\frac{\sum_{i\in U_j}\frac{a_ip_i}{M}}{1-a_ip_i}<1-\frac{\min_i(\sum_{i\in U_j}\frac{a_ip_i}{M})}{1-\min_i(a_ip_i)} \tag{31}$$

Substituting the correspondence bounds of Equations (29) and (30) into Equation (31), we obtain

$$1-\frac{\frac{N-M}{N}}{1-\frac{M}{N}}<1-\frac{\sum_{i\in U_j}\frac{a_ip_i}{M}}{1-a_ip_i}<1$$

Thus, we get the bounds that we are interested in

$$0 < 1 - \frac{\sum_{i \in U_j} \frac{a_i p_i}{M}}{1 - a_i p_i} < 1 \tag{32}$$

Therefore, we know that $\alpha(i_j)$ is greater than zero. Now, we express $\sum_{j=1}^{M} V_j(D_{k_r^j+1})$ as a function of the parameters $\alpha(i_j)$ and $\beta(i_j)$. Thus,

$$\sum_{j=1}^{M} V_j(D_{k_r^j+1}) \leq \sum_{j=1}^{M} \left\{ V_j(D_{k_r^{j^*}}) - \alpha(i_j^*(k_r^{j^*}))T_{i_j^*(k_r^{j^*})}(D_{k_r^{j^*}}) + \beta(i_j^*(k_r^{j^*})) \right\} \tag{33}$$

We use the definition of *either* DDSS strategy

$$\alpha(i_j^*(k_r^{j^*}))T_{i_j^*(k_r^{j^*})}(D_{k_r^{j^*}}) \geq \frac{\alpha(i_j^*(k_r^{j^*}))}{(N-M)\bar{a}} \left[\sum_{i \in U_j^a(D_{k_r^{j^*}})} a_i T_i(D_{k_r^{j^*}}) \right]$$

But notice that

$$\begin{aligned}
\alpha(i_j^*(k_r^{j^*}))T_{i_j^*(k_r^{j^*})}(D_{k_r^{j^*}}) &\geq \frac{\alpha(i_j^*(k_r^{j^*}))}{(N-M)\bar{a}} \left[a_{i_j^*(k_r^{j^*})} T_{i_j^*(k_r^{j^*})}(D_{k_r^{j^*}}) + \sum_{i \in U_j(D_{k_r^{j^*}})} a_i T_i(D_{k_r^{j^*}}) \right] \\
&\geq \frac{\alpha(i_j^*(k_r^{j^*}))}{(N-M)\bar{a}} \left[a_{i_j^*(k_r^{j^*})} T_{i_j^*(k_r^{j^*})}(D_{k_r^{j^*}}) + \sum_{i \in U_j(D_{k_r^{j^*}})} \frac{a_i}{M} T_i(D_{k_r^{j^*}}) \right] \\
&\geq \frac{\alpha(i_j^*(k_r^{j^*}))}{(N-M)\bar{a}} V_j(D_{k_r^{j^*}})
\end{aligned}$$

Combine this with Equation (33) to get

$$\sum_{j=1}^{M} V_j(D_{k_r^j+1}) \leq \sum_{j=1}^{M} \left\{ V_j(D_{k_r^{j^*}}) \left(1 - \frac{\alpha(i_j^*(k_r^{j^*}))}{(N-M)\bar{a}} \right) + \beta(i_j^*(k_r^{j^*})) \right\} \tag{34}$$

Subtract $\sum_{j=1}^{M} \underbrace{(N-M)\bar{a} \max_{i_j} \frac{\beta(i_j)}{\alpha(i_j)}}_{b}$ from both sides of Equation (34) and after a bit of algebra we get that

$$\sum_{j=1}^{M} \left\{ V_j D_{(k_r^j+1)} - b \right\} \leq \sum_{j=1}^{M} \left\{ \left[V_j(D_{k_r^{j^*}}) - b \right] \left[1 - \frac{\alpha(i_j^*(k_r^{j^*}))}{(N-M)\bar{a}} \right] + \beta(i_j^*(k_r^{j^*})) \left[1 - \frac{\alpha(i_j^*(k_r^{j^*}))}{\beta(i_j^*(k_r^{j^*}))} \max_{i_j} \frac{\beta(i_j)}{\alpha(i_j)} \right] \right\}$$

Notice that

$$\beta(i_j^*(k_r^{j^*})) \left[1 - \frac{\alpha(i_j^*(k_r^{j^*}))}{\beta(i_j^*(k_r^{j^*}))} \max_{i_j} \frac{\beta(i_j)}{\alpha(i_j)} \right] \leq 0$$

Now, we have

$$\begin{aligned}
\sum_{j=1}^{M} \left\{ V_j(D_{k_r^j+1}) - b \right\} &\leq \sum_{j=1}^{M} \left\{ \left[V_j(D_{k_r^{j^*}}) - b \right] \left[1 - \frac{\alpha(i_j^*(k_r^{j^*}))}{(N-M)\bar{a}} \right] \right\} \\
&\leq \sum_{j=1}^{M} \left\{ \left[V_j(D_{k_r^{j^*}}) - b \right] \left[1 - \frac{\min_{i_j} \alpha(i_j^*(k_r^{j^*}))}{(N-M)\bar{a}} \right] \right\} \qquad (35)
\end{aligned}$$

But, notice that

$$\left[1 - \frac{\min_{i_j} \alpha(i_j^*(k_r^{j^*}))}{(N-M)\bar{a}}\right] = 1 - \frac{\min_{i_j}\left\{a_i\left(1 - \frac{\sum_{i \in U_j(D_{k_r^{j^*}})} \frac{a_i p_i}{M}}{1-a_i p_i}\right)\right\}}{(N-M)\bar{a}}$$

$$= 1 - \frac{\left\{\underline{a}\left(1 - \frac{\sum_{i \in U_j(D_{k_r^{j^*}})} \frac{a_i p_i}{M}}{1-\max_i(a_i p_i)}\right)\right\}}{(N-M)\bar{a}}$$

By using the bounds of Equations (29) and (30), we obtain that

$$1 - \frac{\frac{N-M}{N}}{1-\frac{M}{N}} < 1 - \frac{\sum_{i \in U_j} \frac{a_i p_i}{M}}{1-\max_i(a_i p_i)} < 1$$

Hence,

$$0 < 1 - \frac{\sum_{i \in U} \frac{a_i p_i}{M}}{1-\max_i(a_i p_i)} < 1$$

Furthermore,

$$0 < \frac{\underline{a}}{\bar{a}(N-M)} < 1$$

so that

$$0 < 1 - \frac{\min_{i_j} \alpha(i_j^*(k_r^{j^*}))}{(N-M)\bar{a}} < 1 \tag{36}$$

Recall that Equation (35) is based on the decisions taken at times $D_{k_r^{j^*}}$ and $D_{k_r^j+1}$. However, it is not clear so far how these decisions occur as the time goes since M receivers are taking decisions independently at least to some extent. Furthermore, it is not even clear whether all intervals $A_{k_r} = [D_{k_r^{j^*}}\ D_{k_r^j+1}]$ are concatenated in such a way that $\bigcup_{k_r=1}^{K} A_{k_r} \subset \Re^+$. Therefore, we define a mapping that establishes how consecutive decisions occur for every $D_{k_r^{j^*}}$ and $D_{k_r^j+1}$, $\forall j, j \in R$. Let D_{k_r} be equal to $D_{k_r^{j^*}}$, D_{k_r+1} be equal to $D_{k_r^j+1}$, and consequently, $A_{k_r} = [D_{k_r}\ D_{k_r+1}]$, $k_r = 1, 2, \ldots, K$, K is the number of decisions made by M receivers (considering that at $D_{1^j} = 0$ is made just one decision instead of M decisions). The mapping is generated by

$$D_{k_r} = \max(D_{k_r^1}, D_{k_r^2}, \ldots, D_{k_r^M}) \tag{37}$$

$$D_{k_r+1} = \min(D_{k_r^1+1}, D_{k_r^2+1}, \ldots, D_{k_r^M+1}) \tag{38}$$

$$k_r = 1, 2, \ldots, K$$

Note that the k_r^j indices in the elements of D_{k_r} and D_{k_r+1} could have different values for every receiver j at a given time t. The above mapping have the following properties:

1. There is no other decision time between $D_{k_r^{j^*}}$ and $D_{k_r^j+1}$
2. $\bigcup_{k_r=1}^{K} A_{k_r}$ is a continuous interval. Moreover, $\bigcup_{k_r=1}^{K} A_{k_r} = [D_1\ D_{K+1}] \subset \Re$
3. All intervals A_{k_r} are disjoint, i.e., $\bigcap_{k_r=1}^{K} A_{k_r} = \emptyset$

We proceed to prove property 1 by contradiction. Hence, we assume that there is one decision time $D_{k_{r^*}}$ that has occurred between decisions $D_{k_r^{j^*}}$ and $D_{k_r^j+1}$. There are two possibilities in this scenario: The first one is that $D_{k_{r^*}}$ belongs the set of elements of D_{k_r}. If this is the case and according to the definition of the decision closest to $D_{k_r^j+1}$, then $D_{k_{r^*}}$ is just equal to $D_{k_r^{j^*}}$ The other possibility is that $D_{k_{r^*}}$ belongs the set of elements of D_{k_r+1}. If that is true, then according to Equation (38), $D_{k_{r^*}}$ has to strictly be equal to D_{k_r+1}. Therefore, based on the above results we have shown that there is no possibility that any other decision time occurs between times $D_{k_r^{j*}}$ and $D_{k_r^j+1}$.

Next, we prove property 2 by induction. Recall that A_{k_r} is defined by $A_{k_r} = [D_{k_r}\ D_{k_r+1}]$. If $k_r = 1$ then $A_1 = [D_1\ D_2]$ If $k_r = 2$ then $A_2 = [D_2\ D_3]$. Note that the right value in the interval A_1 becomes the left value in the interval A_2. We can better understand this by analyzing Equations (37) and (38). While Equation (37) takes into consideration the maximum value of the "past" decisions taken by receivers (M decisions in total) and, Equation (38) considers the minimum value of the "current" decisions taken by all receivers (M decisions in total). Since we assume that at time $D_1 = 0$, all receivers focus on different emitters and they take future decisions asynchronously, it turns out that $D_{k_r} < D_{k_r+1}, \forall k_r, k_r = 1, 2, \ldots$ Hence, the smallest value (D_{k_r+1}) of the elements in Equation (38) will always be the largest value of the elements in Equation (37) for the next decision (index k_r is incremented by 1).

We assume that $\bigcup_{k_r=1}^{K-1} A_{k_r} = [D_1\ D_{K-1}] \subset \Re$ and want to show that this holds for $k_r = K$. If we unite the interval A_K to both sides of the assumed equation, we obtain

$$\bigcup_{k_r=1}^{K-1} A_{k_r} \bigcup A_K = [D_1\ D_{K-1}] \bigcup [D_{K-1}\ D_K] = [D_1\ D_K]$$

Since this is the formula for $k_r = 1, 2, \ldots, K$, we conclude that the time interval $\bigcup_{k_r=1}^{K} A_{k_r}$ is continuous for all k_r as well as $\bigcup_{k_r=1}^{K} A_{k_r} = [D_1\ D_K] \subset \Re$.

By using the results obtained in property 2, we prove property 3. Note that

$$A_1 \bigcap A_2 = \{D_2\}$$

$$A_2 \bigcap A_3 = \{D_3\}$$

and

$$A_1 \bigcap A_2 \bigcap A_3 = \emptyset$$

Thus, two contiguous intervals have a common element, whereas three or more contiguous ones have no common elements. Therefore, we can prove by induction that $\bigcap_{k_r=1}^{K} A_{k_r} = \emptyset$.

The bounds obtained in Equation (36) and the mapping properties of Equations (37) and (38) make Equation (35) contractive so that

$$\sum_{j=1}^{M} \left\{ V_j(D_{k_r^j+1}) - (N-M)\bar{a} \max_{i_j} \frac{\beta(i_j)}{\alpha(i_j)} \right\} = 0$$

Next we show that due to the delay δ, the $T_{i_j}(t)$ values can rise higher at times t not at the decision points $D_{k_r^{j*}}$ and $D_{k_r^j+1}$. First, we consider t for $D_{k_r^j} \le D_{k_r^{j*}} \le t \le D_{k_r^j} + \delta$ and let $t = D_{k_r^j} + \delta$ in Equation (27) to obtain

$$\begin{aligned} V_j(D_{k_r^j} + \delta) &= V_j(t + D_{k_r^j} + \delta - t) = a_{i_{j^*}} T_{i_{j^*}}(t + D_{k_r^j} + \delta - t) + \sum_{i \in U_j(t + D_{k_r^j} + \delta - t)} \frac{a_i T_i(t + D_{k_r^j} + \delta - t)}{M} \\ &= a_{i_{j^*}} T_{i_{j^*}}(t) + (D_{k_r^j} + \delta - t) a_{i_{j^*}} p_{i_{j^*}} + \sum_{i \in U_j(t)} \frac{a_i T_i(t)}{M} + \sum_{i \in U_j(t)} \frac{a_i p_i (D_{k_r^j} + \delta - t)}{M} \\ &= V_j(t) + \underbrace{(D_{k_r^j} + \delta - t)(a_{i_{j^*}} p_{i_{j^*}} + \sum_{i \in U_j(t)} \frac{a_i p_i}{M})}_{x} \end{aligned} \tag{39}$$

Since $x \ge 0\ \forall t, D_{k_r^{j*}} \le t \le D_{k_r^j} + \delta$, it follows that

$$V_j(t) \le V_j(D_{k_r^j} + \delta) \tag{40}$$

Now let t be in the interval $D_{k_r^j} + \delta < D_{k_r^{j*}} \leq t \leq D_{k_r^j+1}$ and Equation (27) is used once again to obtain

$$V_j(D_{k_r^j+1}) = V_j(t + D_{k_r^j+1} - t) = a_{i_{j^*}} T_{i_{j^*}}(t + D_{k_r^j+1} - t) + \sum_{i \in U_j(t+D_{k_r^j+1}-t)} \frac{a_i T_i(t + D_{k_r^j+1} - t)}{M}$$

But recall that $a_{i_{j^*}} T_{i_{j^*}}(t + D_{k_r^j+1} - t) = 0$. Thus,

$$\begin{aligned} V_j(D_{k_r^j+1}) &= \sum_{i \in U_j(t+D_{k_r^j+1}-t)} \frac{a_i T_i(t + D_{k_r^j+1} - t)}{M} \\ &= \sum_{i \in U_j(t)} \frac{a_i T_i(t)}{M} + \sum_{i \in U_j(t)} \frac{a_i p_i (D_{k_r^j+1} - t)}{M} \end{aligned}$$

Use Equation (22) (making $D_{k_r^{j*}}$ equal to t), and Equation (27) for the first term in Equation (41) to obtain

$$\begin{aligned} V_j(D_{k_r^j+1}) &= V_j(t) - a_{i_{j^*}} T_{i_{j^*}}(t) + \frac{a_{i_{j^*}} T_{i_{j^*}}(t)}{1 - a_{i_{j^*}} p_{i_{j^*}}} \sum_{i \in U_j(t)} \frac{a_i p_i}{M} \\ &= V_j(t) - \underbrace{a_{i_{j^*}} T_{i_{j^*}}(t) \left[1 - (1 - a_{i_{j^*}} p_{i_{j^*}})^{-1} \sum_{i \in U_j(t)} \frac{a_i p_i}{M} \right]}_{y} \end{aligned}$$

Since $y \geq 0 \; \forall t, D_{k_r^{j*}} \leq t \leq D_{k_r^j+1}$, it follows that

$$V_j(t) \leq V_j(D_{k_r^{j*}}) \tag{41}$$

Now we revisit the case when $D_{k_r^j} \leq D_{k_r^{j*}} \leq t \leq D_{k_r} + \delta$, and use Equation (40) to obtain

$$\lim_{t\to\infty} \sup \sum_{j=1}^{M} V_j(t) \leq \sum_{j=1}^{M} \left[V_j(D_{k_r^j} + \delta) \right] = \sum_{j=1}^{M} \left[V_j(D_{k_r^{j*}} + D_{k_r^j} + \delta - D_{k_r^{j*}}) \right] \tag{42}$$

Let $t = D_{k_r^{j*}}$ in Equation (39) and substitute this result in Equation (42) to get

$$\lim_{t\to\infty} \sup \sum_{j=1}^{M} V_j(t) \leq \sum_{j=1}^{M} \left\{ V_j(D_{k_r^{j*}}) + (D_{k_r^j} + \delta - D_{k_r^{j*}})(a_{i_j^*(k_r^{j*})} p_{i_j^*(k_r^{j*})} + \sum_{i \in U_j(D_{k_r^{j*}})} \frac{a_i p_i}{M}) \right\}$$

But if $D_{k_r^j} \leq D_{k_r^{j*}} \leq t \leq D_{k_r} + \delta$ then $D_{k_r^j} + \delta - D_{k_r^{j*}} < \delta$. Furthermore,

$$\lim_{t\to\infty} \sup \sum_{i=1}^{N} a_i T_i(t) = \lim_{t\to\infty} \sup \sum_{j=1}^{M} V_j(t)$$

and since

$$\lim_{t\to\infty} \sup \sum_{i=1}^{N} T_i(t) \leq \frac{1}{\underline{a}} \lim_{t\to\infty} \sup \sum_{j=1}^{M} V_j(t)$$

We know

$$\lim_{t\to\infty} \sup \sum_{i=1}^{N} T_i(t) \leq \sum_{j=1}^{M} \left\{ \frac{\delta}{\underline{a}} (a_{i_j} p_{i_j} + \sum_{i \in U_j(D_{k_r^{j*}})} \frac{a_i p_i}{M}) + (N - M) \frac{\bar{a}}{\underline{a}} \max_{i_j} \left[\frac{\delta \sum_{i \in U_j(D_{k_r^{j*}})} \frac{a_i p_i}{M}}{a_{i_j} (1 - a_{i_j} p_{i_j} - \sum_{i \in U_j(D_{k_r^{j*}})} \frac{a_i p_i}{M})} \right] \right\} \tag{43}$$

On the other hand, according to Equation (41) we obtain the following result for the case when $D_{k_r^j}+\delta < D_{k_r^{j*}} \le t \le D_{k_r^j+1}$

$$\lim_{t\to\infty} \sup \sum_{i=1}^{N} T_i(t) \le \sum_{j=1}^{M} \left\{ (N-M)\frac{\bar{a}}{\underline{a}} \max_{i_j} \left[\frac{\delta \sum_{i\in U_j(D_{k_r^{j*}})} \frac{a_i p_i}{M}}{a_{i_j}(1-a_{i_j}p_{i_j} - \sum_{i\in U_j(D_{k_r^{j*}})}\frac{a_i p_i}{M})} \right] \right\} \tag{44}$$

Notice that the bound obtained in Equation (44) is an ideal one. That bound tells us that when one receiver j focus on an emitter and detects it, the rest of receivers ($m \neq j, \forall j, m \in R$) will completely detect all emitters that are currently attending at time t, $D_{k_r^j}+\delta < t < D_{k_r^j+1}$. However, this is practically impossible to happen all the time because of the presence of the term δ^i in every receiver. This is a pathological case and we discard it. Therefore, the bound for every $T_i(t)$ is given by Equation (43). Moreover, we get rid of the dependence of the bound on $D_{k_r^{j*}}$ (we need to divide this equation by M) by using Equation (43). Then,

$$\begin{aligned}
\lim_{t\to\infty} \sup \sum_{i=1}^{N} T_i(t) &\le \sum_{j=1}^{M} \left\{ \frac{\delta}{\underline{a}} \left(a_{i_j}p_{i_j} + \frac{1}{M}\sum_{i=1}^{N} a_i p_i - \underline{ap} \right) + (N-M)\frac{\bar{a}}{\underline{a}} \max_{i_j} \left[\frac{\delta(\frac{1}{M}\sum_{i=1}^{N} a_i p_i - \underline{ap})}{a_{i_j}(1-a_{i_j}p_{i_j} - \frac{1}{M}\sum_{i=1}^{N} a_i p_i + \underline{ap})} \right] \right\} \\
&\le \frac{\delta M}{\underline{a}} \left(a_i p_i + \frac{1}{M}\sum_{i=1}^{N} a_i p_i - \underline{ap} \right) + M(N-M)\frac{\bar{a}}{\underline{a}} \max_{i} \left[\frac{\delta(\frac{1}{M}\sum_{i=1}^{N} a_i p_i - \underline{ap})}{a_i(1-a_i p_i - \frac{1}{M}\sum_{i=1}^{N} a_i p_i + \underline{ap})} \right] \\
&\le \frac{\delta}{\underline{a}} \left(M a_i p_i + \sum_{i=1}^{N} a_i p_i - M\underline{ap} \right) + (N-M)\frac{\bar{a}}{\underline{a}} \max_{i} \left[\frac{\delta(\sum_{i=1}^{N} a_i p_i - M\underline{ap})}{a_i(1-a_i p_i - \frac{1}{M}\sum_{i=1}^{N} a_i p_i + \underline{ap})} \right]
\end{aligned}$$

which gives the desired result. ■

Note that if there are $M = N-1$ receivers working together then the bound is given by

$$\lim_{t\to\infty} \sup \sum_{i=1}^{N} T_i(t) \le \frac{\delta}{\underline{a}} \left((N-1)a_i p_i + \sum_{i=1}^{N} a_i p_i - (N-1)\underline{ap} \right) + \frac{\bar{a}}{\underline{a}} \max_{i} \left[\frac{\delta(\sum_{i=1}^{N} a_i p_i - (N-1)\underline{ap})}{a_i(1-a_i p_i - \frac{1}{N-1}\sum_{i=1}^{N} a_i p_i + \underline{ap})} \right]$$

Also, note that the effect of finite-length delays in keeping the set of unattended emitters up to date has a similar effect to the delays δ that we consider in the proof in that they simply raise the bounds that we obtain; hence, we do not further consider them here.

10 Simulation of DDSS Strategies

In this section we illustrate the behavior of one DDSS strategy and then compare its performance to two noncooperative strategies to illustrate potential benefits of cooperation via DDSS.

10.1 DDSS Strategy Behavior

Here, we simulate the case when there are 2 receivers ($M = 2$) working together for detecting the appearances of four emitters ($N = 4$). Figure 18 shows the dynamics of every $T_i(t)$ when we implement the strategy defined in Section 9.2.2. Compared to a similation for a non-cooperative case the peaks are actually lower (we omit the simulation plots here). In Figure 19 we can see the performance measures for this case.

Figure 20 shows the dynamics of the decisions taken by the pair of detectors. We can make some observations about these dynamics. First of all, note the fact that every receiver focuses on a different emitter at every time t, $0 \le t \le 100$. Second, we let two receivers take decisions at the same time t (if that is the case), however, we establish a priori the order in where the decisions will be made by receivers. This order is established in the following way: $j = 1, 2, \ldots, M$, that is, for example, suppose that recievers 1,3, and M detect 3 emitters at time t, then receiver 1 will focus on a new emitter ignored the longest in the set $U(t)$, then receiver 2 will do the same with the new set $U(t)$ and finally it will be time for reciever M to focus on a new emitter. Although the logic mentioned above is implemented in our algorithm, we assume that when two or more recievers take decisions at the same time, the delay time between the *first* decision and the m^{th} one is negligible compared to the sampling time T_s.

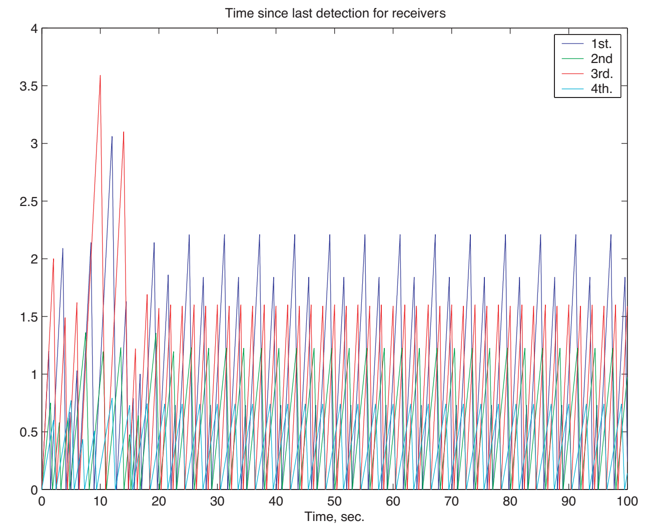


Figure 18: $T_i(t)$ for emitters $i = 1, \ldots, 4$ when there are 2 receivers working together.

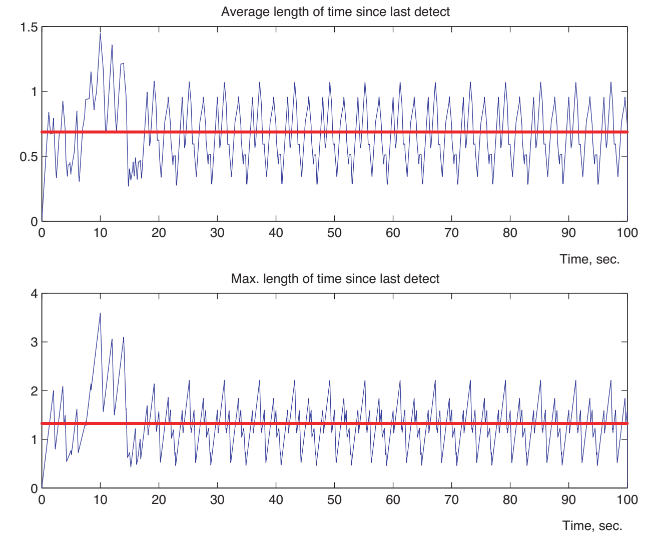


Figure 19: Performance measures: average and maximum length time since last detection and their respective average values for DDSS.

10.2 Comparative Analysis with Noncooperative Case: Benefits of Cooperation

Next, we show in Table 1 a summary of the performance measures simulations, including the corresponding noncooperative cases. Here, we call "efficiency" the percentage of emitter illuminations that are detected in each case and show the values for each of the four emitters, along with the corresponding non-cooperative cases.

In every respect the cooperative strategy outperforms the noncooperative counterparts as expected. It does signficantly better in terms of the earlier performance measures of the average of the average times ignored and maximum of the average times ignored. Also, in terms of efficiency the DDSS strategy significantly outperforms the noncooperative ones. This is expected since there is twice the processing power when there are two receivers; however, notice that it *more than halves* the performance measures and more than doubles the efficiency. At first glance this may seem surprising. However, it is not since this is a standard "emergent" property from cooperation. We pay the price for this added performance since we have

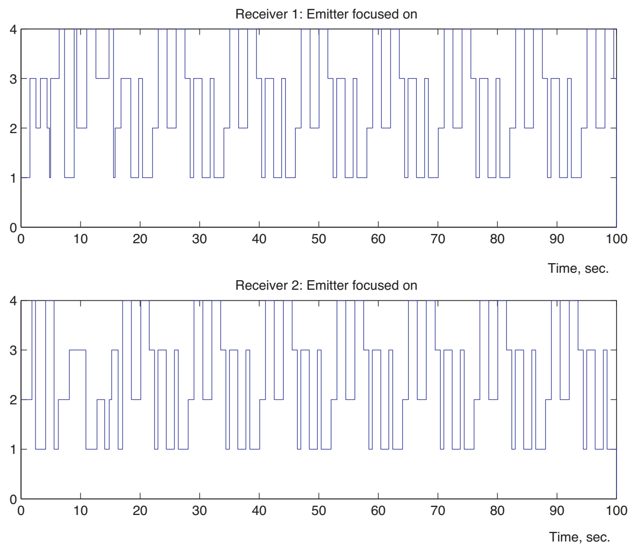


Figure 20: Emitter focused on by detectors ($M = 2$).

to implement a communication system in order to get such good performance. Basically, the DDSS can do much better since it ignores emitters less by better keeping an available receiver focused on emitters that will appear next.

11 Conclusions and Future Directions

In this paper we have developed a variety of strategies for stable dynamic scan scheduling. We discussed how to simulate them and highlighted a variety of issues in design of DSSs. Moreover, we showed that certain DSSs are stable, or can be stabilized with a universal stabilizing mechanism. Also, we showed that a response surface methodology and the SPSA are useful in DSS design. We discussed a model predictive control strategy where the focus is on how to improve DSS performance, while maintaining stable operation. Finally, we showed one way to generalize the results to the muliple-platform case where more than one receiver coordinates the use of information to develop a coordinated schedule for scanning.

There are a variety of potential research directions:

1. **Performance:** The key future direction is to focus on how to achieve good performance for stable dynamic scan scheduling policies. Section 8 shows a potential approach to achieve high performance dynamic scan scheduling. Integration of the use of SPSA with an MPC strategy would result in a strategy for DSS design that tries to incorporate information about the emitter environment and receiver, and then to tries to tune the resulting DSS to optimize its performance.

2. **Adaptive DSS strategies:** Can estimation of the $C_i^k(t, t_i^k)$ functions (e.g., via estimation of the peaks and spreads of the humps on those functions) provide a more flexible DSS that is more robust to changes in emitter illumination pattern changes? Such a strategy would result in an adaptive model predictive control-based DSS strategy.

3. **Unknown number of emitters:** What if the number of emitters is unknown? Can we specify stable DSS policies? It seems that development of such strategies would require studying how to allocate "idle time" of receivers to look in other frequency bands for emitters. Clearly, this would cost in performance. How much time can be spent looking for unknown emitters, when you already know a fixed number exist?

4. **Distributed dynamic scan scheduling (DDSS):** Clearly, it is possible to build on the results

Table 1: Summary of performance measures.

	Ignored the longest	Most difficult to detect	Ignored the longest (distributed)
Average of average values of times	2.97	3.70	0.69
Maximum of average values of times	5.23	6.62	1.33
Efficiency (percentage) of emitter 1 illuminations detected	15.66	20.48	60.24
Efficiency (percentage) of emitter 2 illuminations detected	12.12	6.06	53.03
Efficiency (percentage) of emitter 3 illuminations detected	26.00	18.00	96.00
Efficiency (percentage) of emitter 4 illuminations detected	6.00	4.00	34.00

here in DDSS to include more constraints from the communication network, vehicle dyanmics, and particularly realistic range-constrained sensors.

12 Appendix A: Matlab Code for Dynamic Scan Scheduler Simulations

Below are a few programs that are written in Matlab for the simulation and design of DSS. Not all programs developed for this report are included (e.g., the SPSA algorithm and its use for design or the DDSS algortihsm); however, these are based on the algorithms shown below and can be obtained upon request from the author.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Simulator for Dynamic Scan Schedulers
%
% By: Kevin Passino
% Version: 5/14/01
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


clear

Tmax=200;   % Length of simulation time (real time, in seconds)
Tsample=0.01;  % Sampling period (used to discretize)
Nsteps=(1/Tsample)*Tmax;   % Number of steps to run the simulation
N=4;          % Number of emitters
T=0*ones(N,Nsteps+1); % T(i,k) will denote the number of steps that emitter i has
  % been ignored as of step k.  This simply allocates memory
  % for it.

%------------------------
% Emitter characteristics:
%------------------------

e=0*ones(N,Nsteps+1); % Allocate memory for emitters signals

% Define emitters illumination instants (note that f(i,1) refers to emitter i, and that emitter i
% has priority i, where higher values of i indicate higher priority emitters)
```

```
f(:,1)=0*ones(N,1); % Allocate memory
f(1,1)=1*(1/Tsample);  % Sets that we want an illumination from emitter 1 every f(1,1) steps (1.2 sec.)
 % (this sets the frequency of illuminations)
f(2,1)=1.1*(1/Tsample);
f(3,1)=1.2*(1/Tsample);
f(4,1)=1.3*(1/Tsample);
%f(5,1)=1.4*(1/Tsample);
fcounter(:,1)=ones(N,1); % Allocate memory for counters

% Next, define the bounds delta^i on the maximum amount of time between the illuminations of each emitter
% (note that these must be consistent with the choice for the f(;,1) parameters above that define when the
% illuminations occur).  These are given in real time, not steps.

deltai(1,1)=1.05; % Overbounds a choice of 1.2 above
deltai(2,1)=1.15;
deltai(3,1)=1.25;
deltai(4,1)=1.35;
%deltai(5,1)=1.45;

%--------------------------
% Receiver characteristics:
%--------------------------

% Pick DSS (scheduler)  type

sched=1; % Chooses the policy of choosing the emitter that has been ignored for the longest time
%sched=2; % Chooses the policy of choosing the emitter that is the highest priority one that has
      % been ignored longer than the average one
%sched=3; % Chooses the policy of choosing the emitter that may be the most difficult to find
          % (of course if the a(:,1) are all the same then this is the same as sched=1)
%sched=4; % Chooses the policy of choosing the emitter that may be most difficult to detect
% For sched=4, need weighting factors
w(1,1)=4;
w(2,1)=2;
w(3,1)=1;
w(4,1)=4;
%w(1,1)=1;
%w(2,1)=1;
%w(3,1)=1;
%w(4,1)=1;
peakest=0*ones(N,1); % Allocate memory for computation of estimate of peak for sched=4

% Set the a_i parameters so that the capacity condition is met:

% First, a lightly loaded receiver ("load" in terms of how busy the receiver should be
%                                    for the rate at which it can detect emitters)
% Remember 1/(a(i,1)) is the rate at which can figure out emitter characteristics in
% order to detect it.

%epsilon=0.1;                    % Used to define a_i parameters next
%a(:,1)=epsilon*(1/N)*ones(N,1); % Picks so will meet capacity condition - scale it
                                   % (other choices are possible)

% Lightly loaded receiver; sum=0.2 (also gives differring rates of processing for
% emitters with different delay characteristics)
%a(1,1)=0.09;
%a(2,1)=0.005;
%a(3,1)=0.1;
%a(4,1)=0.005;


% Lightly loaded receiver; sum=0.3 (also gives differring rates of processing for
% emitters with different delay characteristics)
%a(1,1)=0.1;
%a(2,1)=0.05;
%a(3,1)=0.05;
```

```
%a(4,1)=0.1;

% Medium loaded receiver; sum=0.55 (also gives differring rates of processing for
% emitters with different delay characteristics)
%a(1,1)=0.1;
%a(2,1)=0.25;
%a(3,1)=0.15;
%a(4,1)=0.05;

% Relatively overloaded receiver; sum=0.7 (also gives differring rates of processing for
% emitters with different delay characteristics)
a(1,1)=0.1;
a(2,1)=0.2;
a(3,1)=0.3;
a(4,1)=0.1;

% Another choice: even heavier load, sum=0.999
%a(1,1)=0.1;
%a(2,1)=0.2;
%a(3,1)=0.3;
%a(4,1)=0.399;

deltas=3*Tsample; % Sets the delay (in steps) that it takes before the receiver can
                 % switch from focusing on one emitter to another.
 % NOTE: This only accounts for part of the "delta".
 % This is the part due to switching from one emitter to another.
deltacounter=0; % Used to implement the deltas delay

decisiontime=1; % Signals that it is a time (if =1) that pick a new emitters
% to focus on (initially pick one to focus on)
emitterfocusedon=0*ones(Nsteps+1,1); % Initialize variable that indicates which emitter is
% currently being focused on (if in a delay period, means
% the one that it is trying to focus on)
emitterprocess=0; % Flag to indicate that switch delay is over
detectedemitter=0; % Flag that indicates that the emitter that is focused on is detected
% (=0 means not detected, =1 means detected)

%------------------------------------------------
% Variables for measuring performance:
%------------------------------------------------

avgT=0*ones(Nsteps+1,1); % avgT(k,1) denotes the average size of T at each step
 % (average of N values)
maxT=0*ones(Nsteps+1,1); % maxT(k,1) denotes the maximum size of any T at each step
 % (maximum of N values)
avgP=0; % Time average of priorities


%-----------------------------------
% Start simulation loop:
%-----------------------------------


for k=1:Nsteps

for i=1:N  % First generate the illumination signals at step k
if fcounter(i,1)>=f(i,1); % Check if have waited long enough to illuminate
  %(notice here have >= so since due to how the frequency of
  % occurences of illuminations are defined so that f can be  fraction)
e(i,k+1)=1; % Illuminate
fcounter(i,1)=1; % Reset counter
else
e(i,k+1)=0; % Do not illuminate
fcounter(i,1)=fcounter(i,1)+1; % Increment counter
end
end
```

```
% Compute some performance measures that can be used in decision making:
% (and for plotting later)
avgT(k,1)=(1/N)*sum(T(:,k));
maxT(k,1)=max(T(:,k));

if decisiontime~=1;
emitterfocusedon(k,1)=emitterfocusedon(k-1,1); % If not a decision time then same focus as last time
end

% Decide which emitter to focus on:
if decisiontime==1; % Test if it is a decision time

% Policy of choosing the emitter that has been ignored for the longest time
if sched==1
[val,emitterfocusedon(k,1)]=max(T(:,k)); % Pick emitter that has largest T at time k
end % Note that if there is a tie, it is not randomly broken.

% Policy of choosing the emitter that is the highest priority one that has
       % been ignored longer than the average one
if sched==2
for j=1:N  % Consider emitters numbered 1,..,N with N being the highest priority emitter
if T(j,k)>=avgT(k,1)  % Pick the emitter encountered that is above the average (so
% this picks the highest priority one that is above the average)
% Of course there are some wasted computations here...
emitterfocusedon(k,1)=j;
end
end
end

% Policy of choosing the emitter that may be the most difficult to find
if sched==3
[val,emitterfocusedon(k,1)]=max(a(:,1).*T(:,k)); % Scales the T(i,k)
% elements by a(i,1) in making choice
end

% Policy of choosing the emitter that may be most difficult to detect
if sched==4
for j=1:N
peakest(j,1)=w(j,1)*(a(j,1)/(1-a(j,1)))*(T(j,k)+deltas+deltai(j,1));
end
[val,emitterfocusedon(k,1)]=max(peakest); % Picks the emitter expected to have the highest peak
end

decisiontime=0; % Do not decide again until detect that emitter (consider it
% "detected" when T(i,.) goes to zero)

end

if deltacounter>=deltas  % First, implement delay due to switching from one emitter to another
emitterprocess=1;  % Flag to indicate that we are now going to focus on the emitter
% (once deltas is achieved and emitterprocess=1; the counter does not increment
% more and so it always lets emitterprocess=1; to represent that it is continually
% processing it.... and it resets deltacounter only after it has detected)
else
deltacounter=deltacounter+Tsample; % Increment delay since computing T(i,k+1) (note that
% at every time we are focusing on something so
% at every time we need to compute this delay)
% Also, note that below the test for the delay
% accounts for incrementing the counter past delta

end

if emitterprocess==1  % If switch delay is over, then can start trying to detect emitter
if e(emitterfocusedon(k,1),k)==1 % Consider this test for "detection" to only indicate one pulse
detectedemitter=1;  % If detect emitters set flag
end
end  % Note: When for times that emitterprocess==1, after the first one illumination that sets
```

```
     % detectedemitter=1; this loop will have no effect since the if..end will either leave
 % detectedemitter=1; by not being true or if it is true.

if emitterprocess==1 & detectedemitter==1
deltaT=((1-a(emitterfocusedon(k,1),1))/a(emitterfocusedon(k,1),1))*Tsample;
if T(emitterfocusedon(k,1),k)-deltaT>0;  % Must test if will hit zero (to keep nonneg)
T(emitterfocusedon(k,1),k+1)=T(emitterfocusedon(k,1),k)-deltaT;
else
T(emitterfocusedon(k,1),k+1)=0; % Set it to zero since it finishes (an approximation
% that is accurate in the sense that it is conservative)
deltacounter=0; % Reset the delay counter to prepare for next emitter
decisiontime=1; % Signal a decision time since just detected an emitter
detectedemitter=0; % Reset for use in detection of next emitter
emitterprocess=0;  % Reset for next loop
end

else
T(emitterfocusedon(k,1),k+1)=T(emitterfocusedon(k,1),k)+Tsample; % Increment amount of time
% since last detect
end

for i=1:N
if i~=emitterfocusedon(k,1); % If emitter that is focused on is not i
T(i,k+1)=T(i,k)+Tsample; % Update the amounts of time since last focused on each emitter
end
end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot results
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

time=0:Tsample:Tmax;

figure(1)
clf
subplot(411)
stem(time,e(1,:))
title('Emitter illuminations, N=4')
subplot(412)
stem(time,e(2,:))
subplot(413)
stem(time,e(3,:))
subplot(414)
stem(time,e(4,:))
xlabel('Time, sec.')

figure(2)
clf
subplot(311)
stairs(time,emitterfocusedon)
title('Emitter focused on and length of time since last detect (i=1,2)')
subplot(312)
plot(time,T(1,:))
subplot(313)
plot(time,T(2,:))
xlabel('Time, sec.')

figure(3)
clf
subplot(311)
stairs(time,emitterfocusedon)
title('Emitter focused on and length of time since last detect (i=3,4)')
subplot(312)
plot(time,T(3,:))
subplot(313)
```

```
plot(time,T(4,:))
xlabel('Time, sec.')

% Compute the time average of the priorities (like a measure of control energy).
% This gives a measure of how well the scheduling policy did on focusing on higher
% priority emitters

disp('----------------------------------------------------------------')
sched
disp('----------------------------------------------------------------')
disp('The time average of the priorities is=')
avgP=(1/(Nsteps+1))*sum(emitterfocusedon)

figure(4)
clf
subplot(211)
stairs(time,emitterfocusedon)
title('Emitter focused on')
xlabel('Time, sec.')
hold on
plot(time,avgP,'r.')
hold off
subplot(212)
plot(time,T)
xlabel('Time, sec.')
title('Lengths of times since last detected each emitter')

% Compute some performance measures:

disp('The time average of the average values of the lengths of times waited=')
(1/(Nsteps+1))*sum(avgT)  % Computes the time average of the average values

disp('The maximum of the average values of the lengths of times waited=')
max(avgT)  % Computes the maximum of the average values

disp('The time average of the maximum values of the lengths of times waited=')
(1/(Nsteps+1))*sum(maxT)  % Computes the time average of the max values

disp('The maximum of the maximum values of the lengths of times waited=')
max(maxT)  % Computes the maximum of the max values
disp('----------------------------------------------------------------')


figure(5)
clf
subplot(211)
plot(time,avgT,time,(1/(Nsteps+1))*sum(avgT),'r.')
title('Average length of time since last detect')
xlabel('Time, sec.')
subplot(212)
plot(time,maxT,time,(1/(Nsteps+1))*sum(maxT),'r.')
xlabel('Time, sec.')
title('Maximum length of time since last detect')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% End of program
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

13 Appendix B: Matlab Code for Response Surface Construction for DSS Design

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Response Surface Constructor for Design
% Parameters for Dynamic Scan Schedulers
%
% By: Kevin Passino
% Version: 5/14/01
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear

Tmax=100;   % Length of simulation time (real time, in seconds)
Tsample=0.01;  % Sampling period (used to discretize)
Nsteps=(1/Tsample)*Tmax;   % Number of steps to run the simulation
N=3;          % Number of emitters

%-------------------------
% Emitter characteristics:
%-------------------------

e=0*ones(N,Nsteps+1); % Allocate memory for emitter signals

% Define emitter pulse frequencies (note that f(i,1) refers to emitter i, and that emitter i
% has priority i, where higher values of i indicate higher priority targets)

f(:,1)=0*ones(N,1); % Allocate memory
f(1,1)=1.2*(1/Tsample);  % Sets that we want an illumination from emitter 1 every f(1,1) seconds
   % (this "illumination" can represent several pulses).  This essentially models
   % the rotation time of the radar
f(2,1)=1.5*(1/Tsample);  % Sets that we want an illumination from emitter 2 every f(2,1) secs.
f(3,1)=2*(1/Tsample);  % Need one of these for each type of emitter

% Next, define the bounds delta^i on the maximum amount of time between the illuminations of each emitter
% (note that these must be consistent with the choice for the f(;,1) parameters above that define when the
% illuminations occur).  These are given in real time, not steps.

deltai(1,1)=1.3; % Overbounds a choice of 1.2 above
deltai(2,1)=1.6;
deltai(3,1)=2.3;


%---------------------------
% Receiver characteristics:
%---------------------------

% Pick dynamic scan scheduler type

%sched=1; % Chooses the policy of choosing the emitter that has been ignored for the longest time
sched=4; % Chooses the policy of choosing the emitter that may be most difficult to detect
% For sched=4, need weighting factors (for RSM vectors are defined)
w(1,:)=.1:.1:3;
%w(1,:)=.1;
w(2,:)=1;
w(3,:)=.1:.1:3;
%w(3,:)=.1

% Set the a_i parameters so that the capacity condition is met:
epsilon=0.1;                   % Used to define a_i parameters next
a(:,1)=epsilon*(1/N)*ones(N,1); % Picks so will meet capacity condition - scale it
                                    % (other choices are possible)

deltas=1*Tsample; % Sets the delay (in steps) that it takes before the receiver can
```

```
                 % switch from focusing on one emitter to another.
 % NOTE: This only accounts for part of the "delta" used in the paper.
 % This is the part due to switching from one emitter to another.  The
 % other part is the delay due to detection - it changes based on when
 % you decide to focus on an emitter, and then after that time when
 % pulse is detected from that emitter (see use of the variable
 % "detectedemitter" below in several places.

%------------------------------------------------
% Variables for measuring scheduling performance:
% The response surface variable
%------------------------------------------------

avgavgT=0*ones(length(w(3,:)),length(w(1,:))); % avgavgT denotes the time average of the average size of T at each step
    % (average of N values)

%------------------------------------------------
% Set loops to generate response surface (for sched=4)
%------------------------------------------------

for ii=1:length(w(1,:))
for jj=1:length(w(3,:))

% First perform various initializations:

T=0*ones(N,Nsteps+1); % T(i,k) will denote the number of steps that emitter i has
  % been ignored as of step k.
peakest=0*ones(N,1); % Initialize for computation of estimate of peak for sched=4
fcounter(:,1)=ones(N,1); % Initialize for counters


deltacounter=0; % Used to implement the deltas delay

decisiontime=1; % Signals that it is a time (if =1) that pick a new emitter
% to focus on (initially pick one to focus on)
emitterfocusedon=0*ones(Nsteps+1,1);; % Initialize variable that indicates which emitter is
% currently being focused on (if in a delay period, means
% the one that it is trying to focus on)
emitterprocess=0; % Flag to indicate that switch delay is over
detectedemitter=0; % Flag that indicates that the emitter that is focused on is detected
% (=0 means not detected, =1 means detected)

%----------------------------------
% Start simulation loop:
%----------------------------------


for k=1:Nsteps

for i=1:N  % First generate the emitter signals at step k
if fcounter(i,1)==f(i,1); % Check if have waited long enough to emit a pulse
e(i,k+1)=1; % Emit pulse
fcounter(i,1)=1; % Reset counter
else
e(i,k+1)=0; % Do not emit pulse
fcounter(i,1)=fcounter(i,1)+1; % Increment counter
end
end

% Compute some performance measures (QoS) that can be used in decision making:
% (and for plotting later)
avgT(k,1)=(1/N)*sum(T(:,k));
maxT(k,1)=max(T(:,k));

if decisiontime~=1;
emitterfocusedon(k,1)=emitterfocusedon(k-1,1); % If not a decision time then same focus as last time
end
```

```
% Decide which emitter to focus on:
if decisiontime==1; % Test if it is a decision time

% Policy of choosing the emitter that has been ignored for the longest time
if sched==1
[val,emitterfocusedon(k,1)]=max(T(:,k)); % Pick emitter that has largest T at time k
end

% Policy of choosing the emitter that may be most difficult to detect
if sched==4
peakest(1,1)=w(1,ii)*(a(1,1)/(1-a(1,1)))*(T(1,k)+deltas+deltai(1,1));
peakest(2,1)=w(2,1)*(a(2,1)/(1-a(2,1)))*(T(2,k)+deltas+deltai(2,1));
peakest(3,1)=w(3,jj)*(a(3,1)/(1-a(3,1)))*(T(3,k)+deltas+deltai(3,1));
[val,emitterfocusedon(k,1)]=max(peakest); % Picks the emitter expected to have the highest peak
end

decisiontime=0; % Do not decide again until detect that emitter (consider it
% "detected" when T(i,.) goes to zero)

end

if deltacounter>=deltas  % First, implement delay due to switching from one emitter to another
emitterprocess=1;  % Flag to indicate that we are now going to focus on the emitter
% (once deltas is achieved and emitterprocess=1; the counter does not increment
% more and so it always lets emitterprocess=1; to represent that it is continually
% processing it.... and it resets deltacounter only after it has detected)
else
deltacounter=deltacounter+Tsample; % Increment delay since computing T(i,k+1) (note that
% at every time we are focusing on something so
% at every time we need to compute this delay)
% Also, note that below the test for the delay
% accounts for incrementing the counter past delta

end

if emitterprocess==1  % If switch delay is over, then can start trying to detect emitter
if e(emitterfocusedon(k,1),k)==1 % Consider this test for "detection" to only indicate one pulse
detectedemitter=1;  % If detect emitter set flag
D=k; % The time that an emitter pulse was found
end
end

if emitterprocess==1 & detectedemitter==1
if T(emitterfocusedon(k,1),k)-...
((1-a(emitterfocusedon(k,1),1))/a(emitterfocusedon(k,1),1))*Tsample>0;
               % Must test if will hit zero (to keep nonneg)
T(emitterfocusedon(k,1),k+1)=T(emitterfocusedon(k,1),k)-...
   ((1-a(emitterfocusedon(k,1),1))/a(emitterfocusedon(k,1),1))*Tsample;
else
T(emitterfocusedon(k,1),k+1)=0; % Set it to zero since it finishes (an approximation
% that is accurate in the sense that it is conservative)
deltacounter=0; % Reset the delay counter to prepare for next emitter
decisiontime=1; % Signal a decision time since just detected an emitter
detectedemitter=0; % Reset for use in detection of next emitter
emitterprocess=0;  % Reset for next loop
end

else
T(emitterfocusedon(k,1),k+1)=T(emitterfocusedon(k,1),k)+Tsample; % Increment amount of time
% since last detect
end

for i=1:N
if i~=emitterfocusedon(k,1); % If emitter that is focused on is not i
T(i,k+1)=T(i,k)+Tsample; % Update the amounts of time since last focused on each emitter
end
```

```
end

end

avgavgT(jj,ii)=(1/(Nsteps+1))*sum(avgT); % Save value for ii,jj for response surface

end % End ii loop for w(1,ii)
end % End jj loop for w(2,jj)

%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot results
%%%%%%%%%%%%%%%%%%%%%%%%%%%

time=0:Tsample:Tmax;

figure(1)
clf
subplot(311)
stem(time,e(1,:))
title('Emitter illuminations for three emitters')
subplot(312)
stem(time,e(2,:))
subplot(313)
stem(time,e(3,:))
xlabel('Time, sec.')

% Plot the response surface and the best value on that surface

[val,rowindex]=min(min(avgavgT)); % Gives the min value (val) and its row index
[val,colindex]=min(min(avgavgT')); % Gives the min value and its column index

% Best value on the response surface

avgavgTbest=[w(1,rowindex); w(3,colindex)]
avgavgT(colindex,rowindex)

figure(2)
clf
surf(w(1,:),w(3,:),avgavgT)
hold on
plot3(avgavgTbest(1,1),avgavgTbest(2,1),avgavgT(colindex,rowindex),'bo');
view(138,32);
hold off
zoom
colormap(white);
xlabel('w_1 gain');
ylabel('w_3 gain');
zlabel('Average of average times');
title('Response surface, w_2=1');
rotate3d on


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% End of program
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

References

[1] K. Baker. *Introduction to Sequencing and Scheduling.* Wiley, 1974.

[2] Kevin Burgess and Kevin M. Passino. Stable scheduling policies for flexible manufacturing systems. *IEEE Trans. on Automatic Control*, 42(3):420–425, 1997.

[3] Christos G. Cassandras. *Discrete Event Systems.* Richard D. Irwin, Inc. and Aksen Associates, Inc., 1993.

[4] R.W. Conway, W.L. Maxwell, and L.W. Miller. *Theory of Scheduling.* Addison-Wesley, 1967.

[5] S. French. *Sequencing and Scheduling.* Wiley, 1982.

[6] L. Ho, editor. *Discrete Event Dynamic Systems: Analyzing Complexity and Performance in the Modern World.* IEEE Press, NY, 1992.

[7] P.R. Kumar and Thomas J. Seidman. Dynamic instabilities and stabilization methods in distributed real-time scheduling of manufacturing systems. *IEEE Transactions on Automatic Control*, 35:289–298, March 1990.

[8] R.H. Meyers and D.C. Montgomery, editors. *Response Surface Methodology: Process and Product in Optimization Using Designed Experiments.* John Wiley and Sons, Pub., New York, NY, 1995.

[9] A.N. Michel and R.K. Miller. *Qualitative Analysis of Large Scale Dynamical Systems.* Academic Press, NY, 1977.

[10] A.N. Michel and R.K. Miller. *Ordinary Differential Equations.* Academic Press, NY, 1982.

[11] K. Passino and K. Burgess. *Stability Analysis of Discrete Event Systems.* John Wiley and Sons Pub., New York, 1998.

[12] Kevin M. Passino, A.N. Michel, and P.J. Antsaklis. Lyapunov stability of a class of discrete event systems. *IEEE Transactions on Automatic Control*, 37:269–279, February 1994.

[13] J.R. Perkins and P.R. Kumar. Stable, distributed, real-time scheduling of flexible manufacturing/assembly/disassembly systems. *IEEE Transaction on Automatic Control*, 34:139–148, February 1989.

[14] J.C. Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Trans. on Automatic Control*, 37:332–341, 1992.

[15] J.C. Spall. Implementation of the simultaneous perturbation algorithm for stochastic optimization. *IEEE Trans. on Aerospace and Electronic Systems*, 34(3):817–823, 1998.

[16] J.C. Spall. An overview of the simultaneous perturbation method for efficient optimization. *Johns Hopkins APL Technical Digest*, 19(4):482–492, 1998.

[17] J.C. Spall. Stochastic optimization, stochastic approximation, and simulated annealing. In J.G. Webster, editor, *Wiley Encyclopedia of Electrical and Electronics Engineering*, pages 529–542. Wiley, NY, 1999.

[18] Yong Xun, Mieczyslaw M. Kokar, Kenneth Baclawski, Mark MacBeth, Artan Simeqi, and Fengming Zhang. Control based dynamic scan scheduler. *DARPA report*, Nov. 2000.