Dynamic Scan Scheduling *

To be presented at RTSS'02, Austin, TX, Dec. 2002

Bruno Dutertre
System Design Laboratory, SRI International, Menlo Park, CA
bruno@sdl.sri.com

Abstract

We present an approach to computing cyclic schedules online and in real time, while attempting to maximize a quality-of-service metric. The motivation is the detection of RF emitters using a schedule that controls the scanning of disjoint frequency bands. The problem is NP-hard, but it exhibits a so-called phase transition that can be exploited to rapidly find a "good enough" schedule. Our approach relies on a graph-based schedule-construction algorithm. Selecting the input to this algorithm in the phase-transition region ensures, with high probability, that a schedule will be found quickly, and gives a lower bound on the quality of service this schedule will achieve.

1. Introduction

We examine a real-time scheduling problem encountered in the detection of radio-frequency (RF) emitters. A detection system uses a receiver that must scan disjoint frequency bands to intercept signals from the emitters. A *scan schedule* determines how much time is allocated to each band, and is critical to achieving good detection performance. Ideally, the receiver should focus on emitters that are most likely to be present and most critical to a mission. Today's systems rely on a fixed schedule, computed offline from an a priori table of known emitter types. Since the importance of different emitter types may vary during a mission, performance improvements can be expected by dynamically adjusting the scan schedule to current conditions. This requires an algorithm for computing schedules online and under real-time constraints.

The relative importance of each emitter type can be specified by a set of weights. Finding the optimal schedule for a given weight assignment is NP-hard and there is little hope that this can be done online. Instead, we present a method for constructing a "good enough" schedule within a specified deadline. A key aspect is the use of a so-called *phase transition* to select schedule parameters that give good performance, while ensuring that the deadline is met.

*This work was partially funded by DARPA/AFRL contracts F30602-99-C-0169 and F30602-99-C-0167.

Our method relies on an algorithm for constructing schedules with guaranteed detection probabilities. Given parameters $\delta_1, \ldots, \delta_n$ and $\Delta_1, \ldots, \Delta_n$, where n is the number of bands, this algorithm attempts to compute a scan schedule so that the receiver visits band i for a delay δ_i at least once in every interval of length $\Delta_i + \delta_i$. This schedule-construction problem is still NP-hard, but empirical results show that many of its instances can be solved quickly. The key issue is then to rapidly discover parameters $\delta_1, \ldots, \delta_n$ and $\Delta_1, \ldots, \Delta_n$ that ensure a good quality of service but for which a schedule can be efficiently constructed.

This is done by using the utilization, $U = \sum_{i=1}^{N} \frac{\delta_i}{\delta_i + \Delta_i}$, as a hardness indicator. Since quality of service increases with U, one must search for a feasible instance with utilization as high as possible. Intuitively, instances with low utilization are underconstrained and very likely to be feasible. Conversely, instances with utilization close to 1 are likely to be overconstrained and have no solution. Experimental results confirm this intuition. More important, one observes a phase transition similar to what has been noted in many combinatorial search problems (e.g., [9, 11, 13]). There is a small utilization interval $[U_l, U_h]$, in which the fraction of feasible instances decreases sharply. Most instances of utilization less than U_l are feasible, and for such instances a schedule S is found rapidly, and most instances of utilization more than U_h are infeasible. We exploit this phenomenon by searching for feasible instances whose utilization is between U_l and U_h. With high probability, a feasible instance of utilization at least as high as U_l will be rapidly found. The resulting schedule will then have a quality of service at least as good as what U_l gives, and often better in practice.

2. Problem Description

Receivers used for emitter detection cannot cover the whole spectrum of possible emitter signals but work by scanning disjoint frequency bands. We assume these bands indexed from 1 to n, where $n \geqslant 2$. A *scan schedule* specifies the band to cover at every point in time. This schedule produces a sequence of *dwell intervals*, in each of which the receiver is tuned to a particular band for a specified amount

of time. A schedule is an infinite sequence of *control descriptor words* (CDWs) that specify receiver settings for each dwell interval. For our purpose, a CDW is simply a pair $\langle f, d\rangle$ where f is the index of a frequency band and d is a dwell duration. A scan schedule can then be written

$$\langle f_0, d_0\rangle, \langle f_1, d_1\rangle, \ldots, \langle f_t, d_t\rangle, \ldots$$

Accordingly, the receiver must be tuned to band f_0 for a duration d_0, then to band f_1 for a duration d_1, and so forth.[1]

The emitters to detect produce electromagnetic pulses at a fixed frequency. The signal strength at the receiver depends on physical parameters such as range, antenna geometry and orientation, and emitter power. We say that an *illumination* occurs when the signal strength is high enough to enable detection if the receiver is on the correct band. The length of an illumination — called the *illumination time* — varies with the distance between emitter and receiver. For an emitter to be detected and identified, the receiver must be tuned to the proper frequency band when an illumination occurs and must intercept a sufficient number of pulses. This requires the receiver to stay on the emitter's band for a minimum time, called the *duration to detect* the emitter.

At the beginning of a mission, a table is loaded that specifies the types of emitter that may be encountered. Each emitter type E is characterized by its band i_E, its duration to detect D_E, and a nominal illumination time τ_E. These parameters are such that $0 < D_E < \tau_E$ and $1 \leqslant i_E \leqslant n$. We denote by $\mathcal{E}$ the set of emitter types, and by $\mathcal{E}_i$ the set of emitter types in band i. We assume, without loss of generality, that none of the sets $\mathcal{E}_1, \ldots, \mathcal{E}_n$ is empty.

To detect emitters of type E with high probability, the receiver must revisit band i_E at least once in every interval of length $\tau_E - D_E$, for an interval of length at least D_E each time. In most cases, this cannot be satisfied for all the emitters. Tradeoffs must be made and receiver time must be allocated in priority to the emitters most likely to be present and most relevant to the mission.

The emitter table is fixed but the importance of each emitter type may vary. For example, information about previously detected emitters may indicate that some emitter types are more likely to be present than others, or unforeseen events may change mission objectives. Our goal is an online algorithm that enables a receiver to adapt its schedule in response to changes in emitter priorities. The input to such an algorithm consists of the following data:

- n: the number of bands
- $\mathcal{E}$: a finite set of emitter types
- for each element E of $\mathcal{E}$
  - a triple $\langle i_E, D_E, \tau_E\rangle$ characteristic of E
  - a minimal probability of detection $p_E \in (0, 1]$
  - a weight W_E

[1] This is a simplified model. More receiver parameters can be specified for each dwell interval, but these parameters are not relevant here. We also assume that the delay for switching between two bands is negligible.

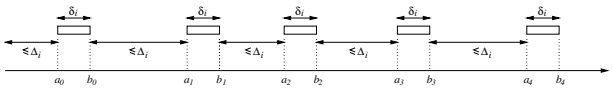


Figure 1. Dwells in a regular schedule

The algorithm must compute a schedule S that satisfies the following *coverage constraint*

$$\forall E \in \mathcal{E} : P_S(E) \geqslant p_E$$

and maximizes the following objective function

$$F(S) \quad = \quad \sum_{E \in \mathcal{E}} W_E P_S(E),$$

where $P_S(E)$ denotes the probability of detecting an illumination of length τ_E from an emitter of type E.

The weights specify the relative importance of each emitter type, and vary during a mission. W_E can be interpreted as a "reward" received whenever an emitter of type E is detected. A good schedule maximizes the expected total reward. The parameters p_E remain constant. They ensure that no emitter type is completely ignored and must be small enough that the coverage constraints can be satisfied.

To work online, the algorithm must compute a schedule within a deadline D that is on the order of 2 seconds. This requirement is, of course, more important than optimality of the solution. A schedule S must be produced on time even if S is not absolutely optimal.

3. Regular Schedules

In the most general form, a scan schedule is an infinite sequence $S = \langle f_t, d_t\rangle_{t \in \mathbb{N}}$. Our approach relies on a more restricted type of schedule that we call *regular*. A schedule S is regular if it satisfies the following requirements:

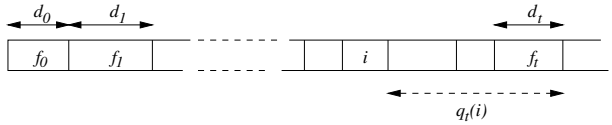
- There are infinitely many dwell intervals for each band i, all of the same length $\delta_i > 0$.
- For each i, there is a constant $\Delta_i > 0$ such that any two successive dwell intervals of band i are separated by a delay no more than Δ_i.

This is illustrated in Figure 1. Every rectangle represents a dwell interval $[a_t, b_t)$ in band i. All these dwells are of length δ_i. The first interval $[a_0, b_0)$ is such that $a_0 \leqslant \Delta_i$ and all subsequent intervals satisfy $a_{t+1} - b_t \leqslant \Delta_i$.

Given an arbitrary schedule $S = \langle f_t, d_t \rangle_{t \in \mathbb{N}}$, let i be a band and let $q_t(i)$ be defined as follows:

$$\begin{aligned} q_0(i) &= 0 \\ q_{t+1}(i) &= \begin{cases} 0 \text{ if } f_t = i \\ q_t(i) + d_t \text{ if } f_t \neq i. \end{cases} \end{aligned}$$

After the t-th CDW, $q_t(i)$ measures the delay since the last occurrence of band i in S, or the delay since the beginning of the schedule if i has not occurred yet:



We call the tuple $q_t = (q_t(1), \ldots, q_t(n))$ the *schedule state* after t steps. By definition, if S is regular then the set

$$Q_i = \{q_t(i) \mid t \in \mathbb{N}\}$$

is bounded. Also, all the elements of Q_i can be written

$$q_t(i) = a_1\delta_1 + \ldots + a_n\delta_n, \quad (1)$$

where $a_1, \ldots, a_n$ are nonnegative integers and $a_i = 0$. Since the dwell times δ_i are positive, this implies that Q_i is finite, and hence has a largest element. For a regular schedule S, we denote this largest element by $\Delta_i(S)$ and the dwell time for band i by $\delta_i(S)$. We also use the notation $\delta(S)$ and $\Delta(S)$ to refer to the two n-tuples:

$$\begin{aligned} \delta(S) &= (\delta_1(S), \ldots, \delta_n(S)) \\ \Delta(S) &= (\Delta_1(S), \ldots, \Delta_n(S)). \end{aligned}$$

For two tuples $\alpha = (\alpha_1, \ldots, \alpha_n)$ and $\beta = (\beta_1, \ldots, \beta_n)$, we write $\alpha \leqslant \beta$ if $\alpha_i \leqslant \beta_i$ for $i = 1, \ldots, n$. A *scan-scheduling instance*, or *instance* for short, is given by two n-tuples of positive reals $\delta = (\delta_1, \ldots, \delta_n)$ and $\Delta = (\Delta_1, \ldots, \Delta_n)$. A regular schedule S such that $\delta(S) = \delta$ and $\Delta(S) \leqslant \Delta$ is a *solution* to the instance. An instance is *feasible* if it has solutions.

The utilization of an instance $(\delta; \Delta)$ is denoted by $U(\delta; \Delta)$ and is defined by

$$U(\delta; \Delta) = \sum_{i=1}^{n} \frac{\delta_i}{\delta_i + \Delta_i}.$$

Obviously, a feasible instance must have utilization no more than 1. On the other hand, there are infeasible instances of arbitrarily low utilization. If δ is fixed there is always a tuple Δ such that $U(\delta; \Delta) = 1$ and the instance $(\delta; \Delta)$ is feasible. For a fixed δ there is also a bound $U_{\min}$ below which all instances are feasible.

A schedule $S = \langle f_t, d_t \rangle_{t \in \mathbb{N}}$ is *cyclic* or *periodic* if there is a positive integer T such that $\langle f_t, d_t \rangle = \langle f_{t+T}, d_{t+T} \rangle$ for all $t \in \mathbb{N}$. The smallest such T is the period of S, and the finite sequence

$$w = \langle f_0, d_0 \rangle, \ldots, \langle f_{T-1}, d_{T-1} \rangle$$

will be called the *generator of* S. Periodic schedules are essential in practice since they can be finitely represented and generated efficiently using simple hardware. In general, a regular schedule may be nonperiodic, but limiting ourselves to cyclic regular schedules is sufficient.

Proposition 1 *If $(\delta; \Delta)$ is a feasible instance then it has a periodic solution.*

Proof: Let $S = \langle f_t, d_t \rangle_{t \in \mathbb{N}}$ be a solution of $(\delta; \Delta)$ and let $(q_t)_{t \in \mathbb{N}}$ be the sequence of states of S. All the states in this sequence belong to the set $Q = Q_1 \times \ldots \times Q_n$. As noted earlier, every Q_i is finite and then Q is also finite. There are then two indices u and u' with $u < u'$ such that $q_u = q_{u'}$. Let $w = \langle f_u, d_u \rangle, \ldots, \langle f_{u'-1}, d_{u'-1} \rangle$, then the schedule generated by w is a solution of $(\delta; \Delta)$. □

The following important property gives a bound on the probabilities of detection achieved by a regular schedule.

Proposition 2 *Let S be a regular schedule, solution of an instance $(\delta; \Delta)$, and let E be an emitter type in band $i_E = i$ such that $D_E \leqslant \delta_i$. The probability of detecting an illumination from E with S is bounded as follows:*

- *If $\Delta_i \leqslant \tau_E - 2D_E$ then $P_S(E) = 1$.*
- *If $\Delta_i > \tau_E - 2D_E$ then*

$$P_S(E) \geqslant \frac{\delta_i + \tau_E - 2D_E}{\delta_i + \Delta_i}.$$

This bound is obtained by considering two successive dwell intervals $[a, b)$ and $[a', b')$ for band i in S, and a random illumination $[x, x + \tau_E]$ where x is uniformly distributed between a and a'. The illumination is detected if it overlaps $[a, b)$ or $[a', b')$ by a delay of at least D_E. This happens if $a \leqslant x \leqslant b - D_E$ or $a' + D_E - \tau_E \leqslant x \leqslant a'$. The result follows since $b = a + \delta_i$ and $a' \leqslant a + \Delta_i$.

This property is the main motivation for choosing regular schedules. For such schedules, maximizing $F(S)$ amounts to finding an instance that maximizes a new objective function

$$H(\delta; \Delta) = \sum_{E \in \mathcal{E}} W_E \min\left(1, \frac{\delta_{i_E} + \tau_E - 2D_E}{\delta_{i_E} + \Delta_{i_E}}\right),$$

and constructing a schedule for that instance. The important simplification is that H now depends only on the $2n$ parameters $\delta_1, \ldots, \delta_n$ and $\Delta_1, \ldots, \Delta_n$. The coverage constraints also translate to constraints on these $2n$ parameters.

4. Schedule Construction

A key problem is to determine whether an instance $(\delta; \Delta)$ is feasible and, if so, to construct a solution. BIN PACKING is polynomially reducible to the scan-scheduling feasibility problem. Thus we have the following.

Proposition 3 *Determining whether a scan-schedule instance $(\delta; \Delta)$ is feasible is NP-hard.*

This gives the worst-case complexity, but many instances can actually be solved efficiently using a graph-exploration technique.

If S is a solution to $(\delta; \Delta)$ then the states of S are of the form $q_t = (q_t(1), \ldots, q_t(n))$ with $q_t(i) \leqslant \Delta_i$. Furthermore, every component $q_t(i)$ can be written $a_1\delta_1 + \ldots + a_n\delta_n$, as in (1). In general, let V_i be the set of numbers that can be written in the form (1) and are smaller than or equal to Δ_i. Then the set $V = V_1 \times \ldots \times V_n$ is finite and contains all the states of any solution of $(\delta; \Delta)$.

Given two elements q and q' of V and a band j, the state q' is *the successor of q by j*, if for $i = 1, \ldots, n$ we have

$$q'(i) \quad = \quad \begin{cases} 0 \text{ if } i = j \\ q(i) + \delta_j \text{ if } i \neq j. \end{cases}$$

This is denoted by $q \xrightarrow{j} q'$. Also, let $\rightarrow$ denote the successor relation on V, that is, the relation defined by

$$q \rightarrow q' \quad \Leftrightarrow \quad \exists j : q \xrightarrow{j} q'.$$

The set V and the relation $\rightarrow$ define a directed graph $G = (V, \rightarrow)$. Its vertices are the elements of V and there is an edge from a vertex q to a vertex q' if and only if q' is a successor of q.

If G contains an infinite sequence of states $(q_t)_{t\in\mathbb{N}}$ such that $q_t \rightarrow q_{t+1}$, then there is a unique sequence of bands $(f_t)_{t\in\mathbb{N}}$ such that $q_t \xrightarrow{f_t} q_{t+1}$ for all t. This sequence defines a regular schedule $S = \langle f_t, d_t\rangle_{t\in\mathbb{N}}$ where $d_t = \delta_{f_t}$ and S is easily seen to be a solution of the instance $(\delta; \Delta)$. Conversely, if S is a solution then the sequence of states of S is an infinite path in G. Since G is finite we have the following.

Proposition 4 *The instance $(\delta; \Delta)$ is feasible if and only if the graph G derived from this instance contains a circuit.*

Assuming G has a circuit, let q be a state on the circuit. It is straightforward to show that q can be reached from the state $q_0 = (0, \ldots, 0)$ of G. This remark, together with Proposition 4, is the basis of our schedule-construction algorithm. Starting from q_0, the algorithm explores the graph G until either a circuit is found or all states reachable from q_0 have been explored. In the former case, a regular schedule S is obtained from the circuit. In the latter case, no regular schedule exists and the instance is not feasible.

Determining whether a circuit exists would be easy if G was small. Unfortunately, the cardinality of V increases exponentially with n. Even with a small number of bands, it is typically infeasible to construct and store in memory the whole graph. Experiments with eight bands have shown that the number of states reachable from $(0, \ldots, 0)$ can attain several million.

Instead, our algorithm relies on a depth-first search that does not require constructing the full graph. A naïve depth-first search is inefficient as a large proportion of the graph may have to be explored before a circuit is found (the whole graph if the instance is not feasible). Several optimizations can significantly reduce the number of nodes to explore:

- One can fix a band j a priori and explore only paths that start with $q_0 \xrightarrow{j} q_1 \ldots$
- It is redundant to explore paths on which the same band occurs twice in a row.
- The search for a circuit can be replaced by the following weaker condition. If a path $q_0 \xrightarrow{f_0} q_1 \xrightarrow{f_1} \ldots \xrightarrow{f_u} q_u$ is found where $q_u \leqslant q_k$ for some state q_k among $q_0, \ldots, q_{u-1}$, then the instance is feasible. The sequence of bands $f_{k+1} \ldots f_u$ gives a schedule.

The most efficient optimization is a pruning technique that checks whether it is possible to add a finite number of bands to a path $q_0 \xrightarrow{f_1} q_1 \xrightarrow{f_2} \ldots \xrightarrow{f_u} q_u$. If the check fails, q_u is not the origin of an infinite path and thus is not on a circuit. There is then no need to explore its successors.

Given a state q, let $d_1, \ldots, d_n$ be defined as follows:

$$d_i \quad = \quad \Delta_i + \delta_i - q(i).$$

If q is the origin of an infinite path in G then every band must occur infinitely often on this path. Starting from q, one can then construct a path of length n on which each band occurs exactly once. Let $q \xrightarrow{f_1} q_1 \ldots q_{n-1} \xrightarrow{f_n} q_n$ be such a path. For each band i, there is k such that $f_k = i$. Let $a_i = \delta_{f_1} + \ldots + \delta_{f_{k-1}}$ and let $b_i = a_i + \delta_i$. Since $q(i) + a_i = q_{k-1}(i)$ and $q_{k-1}(i) \leqslant \Delta_i$, we have $b_i \leqslant d_i$.

Therefore, if q is on a circuit, there exist n nonoverlapping intervals $[a_1, b_1), \ldots, [a_n, b_n)$ such that $[a_i, b_i)$ is of length δ_i and $b_i \leqslant d_i$. The pruning test checks whether such n intervals exist. This can be rephrased as an elementary scheduling problem: We are given n tasks corresponding to each band; task i is of length δ_i and has deadline d_i. We must find whether the n tasks can be executed one at a time, in an order such that all deadlines are met. Executing these n tasks in increasing order of deadlines (EDF scheduling) is optimal for this miniproblem. Our pruning procedure sorts then $d_1, \ldots, d_n$ in increasing order, computes the corresponding intervals $[a_i, b_i)$, and checks that all deadlines

are met. This EDF-based test can be efficiently integrated to a depth-first search, with a limited overhead of $O(n)$ per visited node.

This pruning technique considerably reduces the number of nodes to explore, especially on infeasible instances. It can also be generalized by considering more than one occurrence of each band along a path from q. This generalized EDF-based test improves performance even more. Other details of the algorithm and the heuristic we use to order the exploration are discussed in [3].

5. Phase Transitions

We have experimentally evaluated the schedule-construction algorithm on large sets of randomly generated instances, to examine the relationship between utilization and the likelihood that an instance is feasible.

In each experiment, 20000 instances were constructed randomly. The dwells were chosen independently and uniformly distributed in an interval $[\delta_{\min}, \delta_{\max}]$. Similarly, the n deltas were chosen independently and uniformly distributed in an interval $[\Delta_{\min}, \Delta_{\max}]$. The algorithm was applied to each of the 20000 instances with a timeout of 30 s. Each experiment used different settings for n and the distribution intervals. The utilization and status (either feasible, infeasible, or not solved within the 30 s timeout) of each instance were recorded, as were other data, including the search time for feasible and infeasible instances.

Figure 2 shows how fractions of feasible, infeasible, and unsolved instances vary for two experiments conducted with 8 bands. In both cases, one observes a sharp transition in behavior similar to the phase transition observed in combinatorial search problems [9, 11, 13, 14]. Let $F_s(U)$, $F_i(U)$, and $F_u(U)$ denote the fraction of feasible, infeasible, and unsolved instances observed at utilization U. For both experiments in Figure 2, there is a utilization U_l below which $F_s(U) = 1$, and a utilization U_h above which $F_s(U) = 0$. At a critical point U_c between U_l and U_h, 50% of the instances are feasible. The fraction of unsolved instances reaches its maximum very close to U_c, typically at a utilization equal to $U_c \pm 0.02$. This suggests that hard instances are located in the interval $[U_l, U_h]$, close to U_c. This is confirmed by Figure 3, which shows that the average search time for both feasible and infeasible instances is maximal in the interval $[U_l, U_h]$ with a peak around U_c.

Qualitatively, we observed the same phenomenon for all experiments performed with $n = 8$, but the size of the interval $[U_l, U_h]$ varies depending on the bounds for δ_i and Δ_i. In Figure 2, the left curve has a sharp threshold, with $U_l = 0.85$ and $U_h = 0.92$, and the right curve shows a coarse threshold, with $U_l = 0.61$ and $U_h = 0.92$. The size of the interval $[U_l, U_h]$ and the number of unsolved instances are also correlated. There is a clear difference between the fractions of unsolved instances in the two curves of Figure 2. For the left curve $F_u(U)$ is never more than 1%, while $F_u(U)$ can be as high as 45% for the right curve. Similarly, the average search time is larger in experiments with coarse thresholds (Figure 3).

The same behavior was observed on experiments with larger numbers of bands. Two examples with $n = 20$ are shown in Figure 4. The main difference from previous curves is that very few instances can be determined to be infeasible in 30 s of search time. This is not surprising as the graph derived from an instance grows exponentially with n. To determine that an instance is not feasible, the algorithm must explore an increasingly large number of states as n augments.

In these experiments, the instance parameters were chosen randomly, according to some uniform distribution. In practice, the instances are derived from the emitter table, coverage constraints, and objective function. Given an emitter type E in band i, we must ensure that $P_S(E) \geqslant p_E$. As a consequence of Proposition 2, the two following constraints must be satisfied:

$$\begin{aligned} \delta_i &\geqslant D_E \\ \Delta_i &\leqslant \frac{\tau_E - 2D_E}{p_E} + \frac{\delta_i(1-p_E)}{p_E}. \end{aligned}$$

The most economical setting for δ_i is then

$$\delta_i = \max\{D_E \mid E \in \mathcal{E}_i\}. \quad (2)$$

To satisfy the coverage constraints, Δ_i must be smaller than or equal to the constant

$$B_i = \min\left\{\frac{\tau_E - 2D_E}{p_E} + \frac{\delta_i(1-p_E)}{p_E} \mid E \in \mathcal{E}_i\right\} \quad (3)$$

On the other hand, there is no need to choose Δ_i smaller than the following constant

$$A_i = \min\{\tau_E - 2D_E \mid E \in \mathcal{E}_i\}, \quad (4)$$

since any $\Delta_i \leqslant A_i$ ensures $P_S(E) = 1$ for all the emitters in band i.

In dynamic scan scheduling, δ_i is then fixed a priori from the emitter table, and Δ_i varies in the interval $[A_i, B_i]$. Although these constraints imply a different distribution than in the preceding experiments, phase transitions are still observed. Figure 5 shows how the fraction of feasible instances varies with the utilization for two emitter tables and coverage constraints. As previously, one observes a transition zone $[U_l, U_h]$. The curves were generated by constructing a large number of random instances, with the δ_is fixed and Δ_i uniformly distributed in $[A_i, B_i]$. These curves allow one to experimentally estimate the values of U_l and U_h. This computation must be performed offline as it can require a few hours of CPU time, but it needs to be done only once per emitter table.

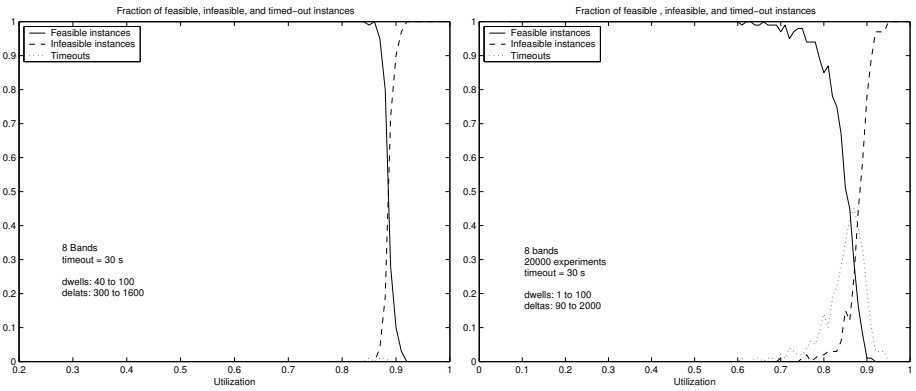


Figure 2. Phase transitions for random instances (8 bands)

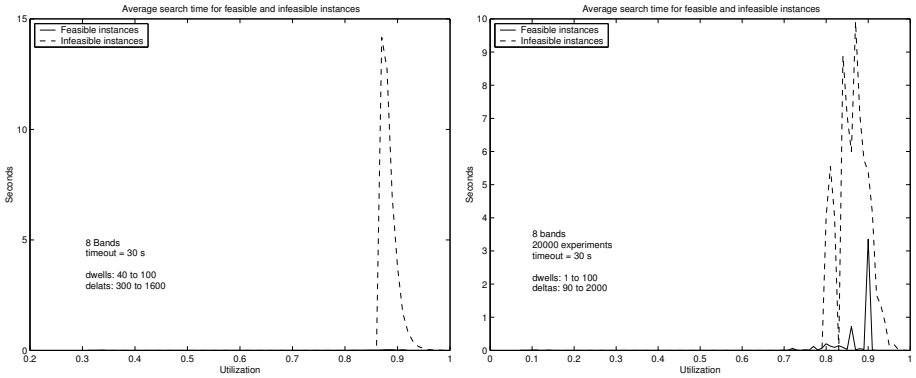


Figure 3. Search time peaks for random instances (8 bands)

6. Dynamic Scan Scheduling

For an emitter table $\mathcal{E}$, the dwell times δ_is are determined by Equation 2, and one can estimate the constants U_l and U_h that delimit the phase-transition region. Our general scan-scheduling strategy is then as follows:

1. Select a utilization bound U_0 between U_l and U_h.
2. Compute the Δ that maximizes quality of service among those that satisfy $U(\delta;\Delta) \leqslant U_0$.
3. Search for a regular scan schedule for $(\delta;\Delta)$, using the graph-exploration algorithm.
4. If a solution is found, attempt to improve performance by repeating the process with a larger U_0. Otherwise, reduce U_0 and repeat the procedure from Step 2.

To ensure termination within a deadline D, we limit the search time in Step 3 using a timeout and stop after a fixed number of iterations.

Step 2 of this algorithm requires solving the following optimization problem:

Find $\Delta_1, \ldots, \Delta_n$ that maximize

$$H(\Delta) \quad = \quad \sum_{E\in\mathcal{E}} W_E \ \min\left(1, \frac{\delta_{i_E} + \tau_E - 2D_E}{\delta_{i_E} + \Delta_{i_E}}\right),$$

and satisfy the constraints

$$A_i \ \leqslant \ \Delta_i \ \leqslant \ B_i \qquad \sum_{i=1}^{n} \frac{\delta_i}{\delta_i + \Delta_i} \ \leqslant \ U_0.$$

The bounds A_i, B_i, and U_0, and the dwell times δ_i are fixed. A solution exists provided U_0 satisfies the inequality

$$U_0 \quad \geqslant \quad \sum_{i=1}^{n} \frac{\delta_i}{\delta_i + B_i}. \tag{5}$$

To solve this problem, the first step is to change variables. Let $x_1, \ldots, x_n$ be defined by $x_i = 1/(\delta_i + \Delta_i)$ for $i = 1, \ldots, n$, and let α_E denote $\delta_{i_E} + \tau_E - 2D_E$. The problem can now be rewritten in the following simpler form:

Find $x_1, \ldots, x_n$ that maximize

$$H'(x_1, \ldots, x_n) \quad = \quad \sum_{E\in\mathcal{E}} W_E \min(1, \alpha_E x_{i_E})$$

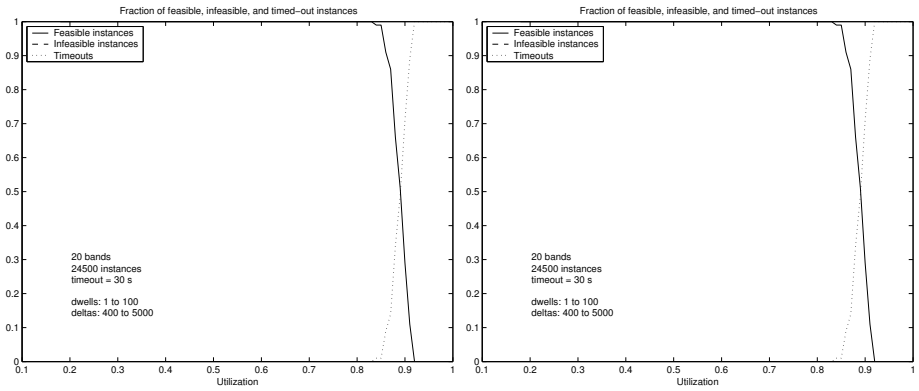


Figure 4. Phase transition for random instances (20 bands)

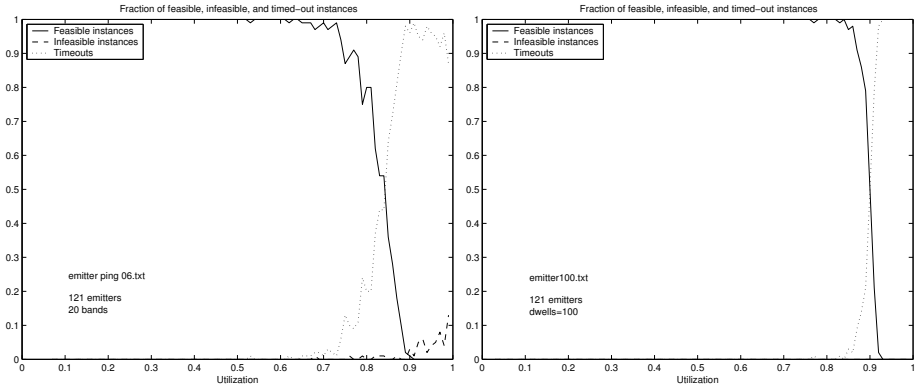


Figure 5. Phase transitions for two emitter tables

$$= \sum_{i=1}^{n} \sum_{E \in \mathcal{E}_i} W_E \min(1, \alpha_E x_i),$$

and satisfy the constraints

$$\frac{1}{\delta_i + B_i} \leqslant x_i \leqslant \frac{1}{\delta_i + A_i} \qquad \sum_{i=1}^{n} \delta_i x_i \leqslant U_0.$$

This is very close to a linear programming problem and an optimal tuple $(x_1, \ldots, x_n)$ can be efficiently computed using the algorithm of Figure 6. This algorithm runs in $O(N \log N)$ time, where N is the total number of emitter types (i.e., $N = |\mathcal{E}|$).

The algorithm starts by setting $x_1, \ldots, x_n$ to their minimal acceptable value so that the coverage constraints are satisfied. The solution is then iteratively improved until the limit U_0 is reached or until all the x_is have their maximal value $1/(\delta_i + A_i)$. At each step, the algorithm picks one emitter type E, say of band i, and increases x_i just enough to achieve 100% probability of detection for E, unless this increases the utilization above U_0. Let ε be the corresponding increment. Changing x_i to $x_i + \varepsilon$ has a cost of $\delta_i \varepsilon$ in utilization, but increases H' by $\sum_{F \in C_i} W_F \alpha_F \varepsilon$, where C_i is the set of emitters F in band i such that $1/\alpha_E \leqslant 1/\alpha_F$. Thus the total gain can be written $agw_E \varepsilon$ where

$$agw_E = \sum_{F \in C_i} W_F \alpha_F.$$

The intuition behind the algorithm is to choose E for which the gain vs. cost ratio agw_E/δ_i is the highest. It is not hard to prove that the algorithm is correct. It always finds an optimal solution $(x_1, \ldots, x_n)$ if condition (5) is satisfied. From this solution, it is trivial to recover the optimal $\Delta_1, \ldots, \Delta_n$ of the original problem.

For a fixed set of weights, this algorithm is monotonic: if Δ and Δ' are the optimal values obtained for two bound U_0 and U_0', respectively, then we have

$$U_0 \leqslant U_0' \;\Rightarrow\; \Delta' \leqslant \Delta.$$

The last issue to resolve is the selection of U_0 at the beginning of each iteration. By the monotonicity property, if schedule construction fails at utilization Y then there is no point in trying $U_0 > Y$. Conversely, if a schedule S was constructed at utilization X then choosing $U_0 < X$ does not make sense. So we use a simple dichotomy process

```
forall E ∈ 𝓔
    let i be the band of E
    let C_i = {F ∈ 𝓔_i | α_F ⩽ α_E}
    agw_E = Σ_{F∈C_i} W_F α_F
    g_E = agw_E / δ_i
L = list of all emitters in decreasing order of g_E
for i = 1 to n set x_i = 1/(δ_i + A_i)
U = Σ_{i=1}^{n} δ_i x_i
while U < U_0 and L ≠ ⟨⟩
    E = first element of L
    remove E from L
    let i be the band of E
    if x_i < 1/α_E
        ε = min(1/α_E − x_i, (U_0 − U)/δ_i)
        x_i = x_i + ε
        U = U + δ_i ε
    endif
end
```

Figure 6. Optimal $x_1, \ldots, x_n$ for a bound U_0

to select U_0. Initially, X and Y are set to U_l and U_h, respectively. At each iteration, U_0 is taken as the midpoint between X and Y, that is, $U_0 = (X + Y)/2$. If a schedule is found at this step, X is set to U_0; otherwise Y is set to U_0. This simple strategy works well in practice, although more sophisticated approaches — that take into account the probability that an instance of utilization U_0 is feasible — could be envisaged.

7. Simulation Results

Simulation was used to compare the performance of dynamic and fixed scan scheduling for several emitter tables and scenarios. For each table, a first schedule S_0 was obtained by assigning equal weight to all emitter types. The performance of a receiver that uses S_0 as a fixed schedule was then compared to a receiver that relies on dynamic scheduling, and uses five successive schedules $S_1, \ldots, S_5$ constructed from five weight files $W_1, \ldots, W_5$. In each weight file, five emitter types are considered critical and have weight 12000; all the other emitter types have weight 100. The schedules S_0 and $S_1, \ldots, S_5$ were all computed using our scan-scheduling algorithm, but S_0 was constructed "offline", that is, with a large deadline of several minutes. On the other hand, $S_1, \ldots, S_5$ were constructed "online", with a deadline D of 2 s CPU time.

For each emitter type E, a simulator synthesized periodic illuminations of length τ_E. The period, phase, and number of illuminations varied randomly. For each weight file W_i, detection results for S_0 and for S_i were collected and a score computed on 30 simulation runs. The score for E takes into account E's weight, as given by W_i, and the first illumination from E that is detected. The score depends on whether E is detected on the first, second, or third illumination, or later (Table 1). If the first three illuminations are not detected then E is considered missed (or detected too late). This incurs a large penalty if E is a critical emitter.

Table 1. Scores

Detected Illumination	Critical	Noncritical
1st	2000	100
2nd	1800	80
3rd	1500	50
4th or more	-10000	0

Figure 7 shows the score (averaged over 30 runs) of fixed and dynamic scheduling for two emitter tables and five weight files. Both tables contain the same number of emitter types, with identical parameters except the duration to detect. For one table, $D_E = 180$ for all E, and in the other $D_E = 200$ for all E. The unit is 10 μs. The top charts in Figure 7 give the total scores and the bottom charts show the score for the critical emitters. Dynamic scheduling outperforms fixed scheduling in these examples, and detects all critical emitters on the first illumination. In the second scenario, fixed scheduling happens to have high probabilities of detection for the critical emitters, and does as well as dynamic scheduling. However, the fixed schedule misses several critical emitters in other scenarios.

Figure 8 gives the total scores (averaged over 30×5 runs) of fixed and dynamic scheduling for increasingly hard emitter tables. The tables were obtained by increasing D_E from 90 to 300. In the easiest case, both approaches achieve detection probabilities close to 1 for all emitter types and have almost maximal scores. The scores get lower as D_E increases, but dynamic scan scheduling always does better than fixed scheduling. In particular, dynamic scheduling has a perfect score on the critical emitters, except for the two hardest tables. For these two tables, the resource requirements for some critical emitters are in conflict, and it is not possible to ensure 100% probability of detection for all of them. On the same two tables, fixed scheduling misses many critical emitters. The same results have been observed for emitter tables with nonuniform D_E.

Simulation shows the benefits of adjusting a scan schedule to the emitter weights. Our algorithm can compute a scan schedule online within a deadline of 2 s, and the resulting schedule largely outperforms a fixed schedule.

8. Related Work

The construction of regular schedules from an instance $(\delta; \Delta)$ is similar to the distance-constrained scheduling problem described in [4]. Han and Lin solve the problem by using pinwheel scheduling [7]. An important difference is that, unlike in [4], preemption is not possible in our case: a dwell interval for a band i cannot be fractioned

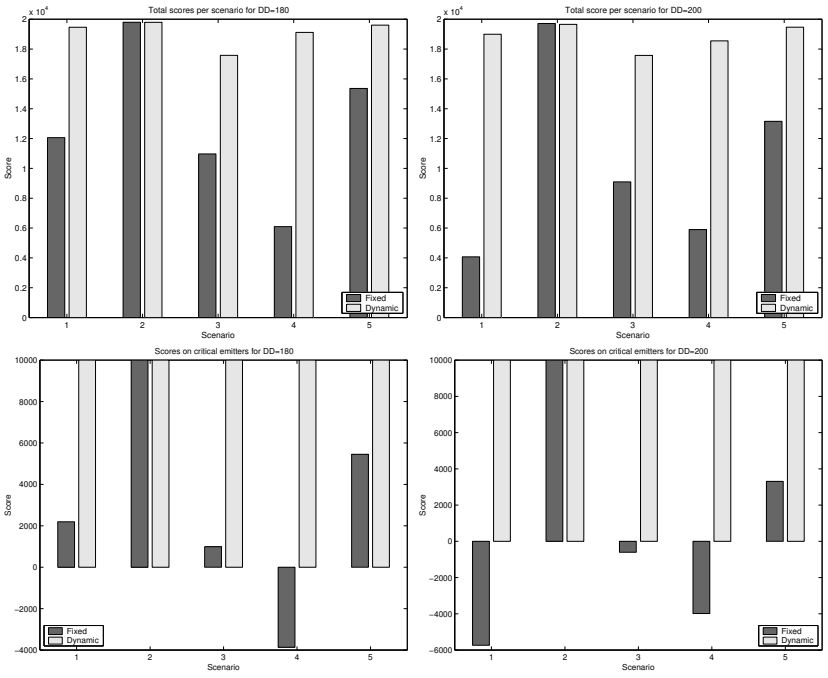


Figure 7. Total scores and scores for critical emitters

in small parts. Despite this restriction, variations of pinwheel scheduling may be applicable to scan scheduling. To the best of our knowledge, existing approaches to pinwheel scheduling (e.g., [1, 2, 10]) require transforming the original instance into one that satisfies adequate algebraic relations. In our case, such transformations would replace the original Δ_is by smaller values, which would both complicate the probability estimates and possibly lead to overconstrained instances. Another difficulty is that pinwheel scheduling typically assumes tasks of unit duration, which in our case would require all δ_is to be equal. The graph exploration we use is applicable to general instances and does not make particular assumptions about the parameters.

Scan scheduling is also related to other nonpreemptive scheduling problems. In traditional contexts — where the objective is to complete all jobs before their deadlines — EDF scheduling is optimal [8]. This is no longer true for scan scheduling. For example, selecting bands in decreasing order of the deadlines d_i (using the notations of Section 4) does not always work. EDF is not optimal because of the dependency between the start time of one dwell and the deadline for the next dwell. It may be better to select a band j before i even though $d_j > d_i$ if $\Delta_j + \delta_j < \Delta_i + \delta_i$.

The MSP.RTL tool [12] can synthesize very general classes of schedules by solving a constraint satisfaction problem expressed in the real-time logic RTL. Our schedule-construction approach relies on an algorithm similar to that discussed in [12]. A main original feature of our approach is to use the algorithm online. Although the worst-case complexity is high, real-time performance is achievable by avoiding instances that are too hard, taking utilization as a hardness indicator, and exploiting the presence of a phase transition. Phase transitions have been observed in many examples of constraint satisfaction problems (CSPs), most notably N-sat (e.g., [9, 11, 13, 14]), but also in other combinatorial problems (see [6] for a survey). Techniques for exploiting phase transitions in CSPs have been investigated in [5] to construct good search heuristics.

9. Conclusion

We have presented an algorithm that is capable of constructing a scan schedule in real time, to improve detection performance as emitter priorities change. The algorithm uses in a novel way a technique for estimating the hardness of specific problem instances. This enables the construction of a schedule online and in real time, even though the problem is NP-hard in general.

Improvements were demonstrated via simulation, but the basic techniques can be extended and generalized for even better performance. In this respect, a possible limitation of the approach presented here is its strict reliance on regular schedules. Such schedules are simple and easy to analyze, but having all the dwells for a band i of the same length can be expensive if the emitters in that band have widely different parameters. Extensions of the basic techniques to

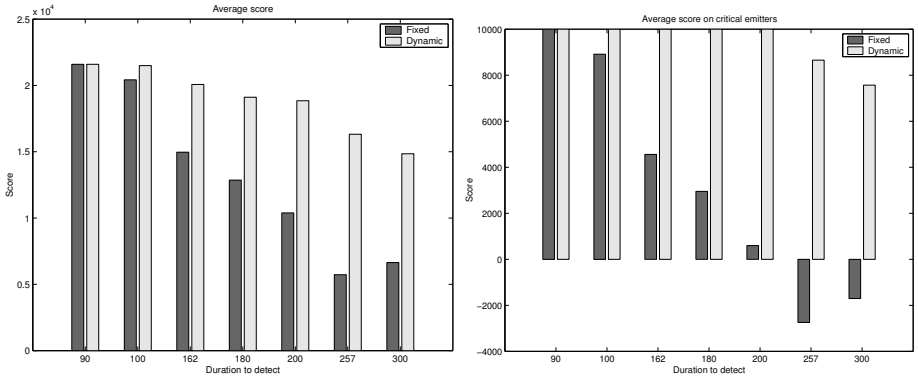


Figure 8. Scores for increasingly hard emitter tables

other types of scan schedule should improve results in such cases.

The techniques presented seem generalizable to many types of nonpreemptive scheduling, as used for example in scheduling communication across buses in distributed systems. More complex models, with or without task preemption, and with or without synchronization between tasks, are possible areas where new algorithms and hardness estimation techniques could be valuable.

The development of theoretical foundations for hardness estimation remains an important issue. All our results are based on empirical evidence, based on a large number of random instances, but we have no rigorous proofs. The experiements strongly suggest that scan scheduling exhibits the same form of phase transition as other types of combinatorial problem. Rigorous proofs that these phenomena exist are particularly difficult, but scan scheduling may have a simpler structure than, say, N-SAT or graph coloring, and be more easily amenable to theoretical study.

References

[1] S. Baruah and A. Bestavros. Pinwheel Scheduling for Fault-Tolerant Broadcast Disks in Real-Time Systems. In *IEEE Conference on Data Engineering*, pages 543–551, Birmingham, UK, April 1997.

[2] M. Y. Chan and F. Chin. General Schedulers for the Pinwheel Problem Based on Double-Integer Reduction. *IEEE Transactions on Computers*, 41(6):755–768, June 1992.

[3] B. Dutertre. A Distributed Dynamic Scan-Scheduling Algorithm. Technical report, System Design Laboratory, SRI International, Menlo Park, CA, February 2002.

[4] C.-C. Han and K.-J. Lin. Scheduling Distance-Constrained Real-Time Tasks. In *Proceedings of the 13th IEEE Real-Time Systems Symposium*, pages 300–308, December 1992.

[5] T. Hogg. Exploiting Problem Structure as a Search Heuristic. *International Journal of Modern Physics C*, 9:13–29, 1998.

[6] T. Hogg, A. Huberman, and C. Williams, editors. *Artificial Intelligence*, volume 81. Elsevier, March 1996. Special issue on *Frontiers in Problem Solving: Phase Transitions and Complexity*.

[7] R. Holte, A. Mok, L. Rosier, I. Tukchinsky, and D. Varvel. The Pinwheel: A Real-Time Scheduling Problem. In *Proceedings of the 22nd Hawaii International Conference on System Science*, pages 693–702, January 1989.

[8] K. Jeffay, D. Stanat, and C. Martel. On Non-Preemptive Scheduling of Periodic and Sporadic Tasks. In *Proceedings of the 12th IEEE Real-Time Systems Symposium*, pages 129–139, December 1991.

[9] S. Kirkpatrick and B. Selman. Critical Behavior in the Satisfiability of Random Boolean Expressions. *Science*, 264:1297–1301, 1994.

[10] S. S. Lin and K. J. Lin. A Pinwheel Scheduler for Three Distinct Numbers with a Tight Schedulability Bound. *Algorithmica*, 19(4):411–426, December 1997.

[11] D. Mitchell, B. Selman, and H. Levesque. Hard and Easy Distributions of SAT Problems. In *Proceedings of AAAI 92*, pages 459–465, Menlo Park, CA, 1992. AAAI Press.

[12] A. Mok, D.-C. Tsou, and R. de Rooij. The MSP.RTL Real-Time Scheduler Synthesis Tool. In *Proceedings of the 17th IEEE Real-Time Systems Symposium*, pages 118–128, Washington, D.C., December 1996.

[13] P. Prosser. An Empirical Study of Phase Transitions in Binary Constraint Satisfaction Problems. *Artificial Intelligence*, 81:81–109, 1996.

[14] B. Selman. Stochastic Search and Threshold Phenomena: AI meets Physics. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'95)*, Montreal, Canada, August 1995.