Scan Scheduling Specification and Analysis[*]

Bruno Dutertre

System Design Laboratory
SRI International
Menlo Park, CA 94025

May 24, 2000

[*]This work was partially funded by DARPA/AFRL under BAE System subcontract RG2206 under prime contract F30602-99-C0167

Abstract

We investigate the construction of fixed schedules for an EW receiver that ensures that n disjoint frequency bands are periodically visited for a given time interval. Two parameters τ_i and T_i, such that $0 < \tau_i < T_i$, are given for each frequency band i: τ_i is the required dwell time and T_i is the required revisit time for band i. The problem is to compute a global schedule, such that, for all i, the receiver covers band i for an interval of length τ_i in every interval of length T_i. We give necessary and sufficient conditions for this problem to have solutions. We prove that the problem is NP-complete. We present a depth-first search algorithm for obtaining a solution and discuss simplification techniques that reduce the search space.

Contents

1 Introduction **2**

2 Patterns and Schedules **4**

3 Greatest Common Divisor **5**

4 Feasibility Conditions **7**
4.1 Compatibility of Two Patterns 7
4.2 Schedule Feasibility 10

5 Algorithms **12**
5.1 Theoretical Complexity 12
5.2 Search-Space Reduction 15
5.3 Schedule Construction Algorithm 22

6 Conclusion **23**

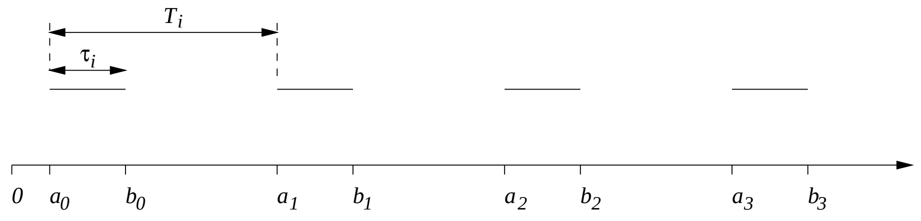


Figure 1: Successive Dwells for a Frequency Band i

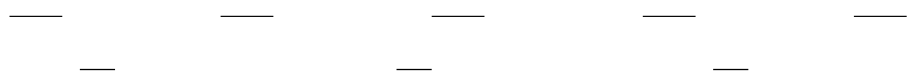
1 Introduction

We consider the problem of constructing a fixed scan schedule for a single EW receiver. The receiver bandwidth is divided into n disjoint frequency bands that can be covered only one at a time. The receiver must periodically visit each of these n bands for a specific duration. For each band i, two parameters τ_i and T_i specify how long and how often band i must be revisited, respectively. T_i is the *revisit time* and τ_i the *dwell time* for band i. We assume that τ_i and T_i are real numbers such that $0 < \tau_i < T_i$. The schedule is required to include, for each frequency band i, a sequence of dwell intervals as depicted in Figure 1. The receiver covers band i in each interval $[a_t, b_t)$, and other frequency bands between b_t and a_{t+1}.

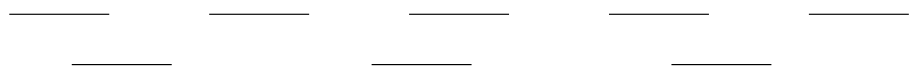
Globally, a schedule can be described by n triples of parameters (a_i, τ_i, T_i) for $i = 1, \ldots, n$, where τ_i is the dwell time for band i, T_i the revisit time for band i, and a_i the start time of the first dwell interval covering band i. These parameters must satisfy the following constraints:

$$0 < \tau_i < T_i \quad \text{and} \quad 0 \leqslant a_i \leqslant T_i - \tau_i.$$

These parameters determine the successive dwell intervals for each frequency band. However, not all possible values for the parameters a_i, τ_i, T_i are acceptable. Since the receiver can cover only a single frequency band at a time, we must ensure that the dwell intervals for distinct bands do not overlap. The following figure shows an example of feasible schedule with $n = 2$, $a_1 = 0$, $\tau_1 = 0.5$, $T_1 = 2$, $a_2 = 0.66$, $\tau_2 = 0.33$, and $T_2 = 3$.



On the other hand, it is easy to see that no feasible schedule exists in case $\tau_1 = 1, T_1 = 2$, and $\tau_2 = 1, T_2 = 3$. Constructing such a schedule would amount to aligning the two sequences of intervals below in such a way that no two intervals overlap. This is clearly impossible.



In general, we are given n pairs of parameters τ_i and T_i, such that $0 < \tau_i < T_i$. The problem is to find whether there exist n real numbers $a_1, \ldots, a_n$ such that $0 \leqslant a_i \leqslant T_i - \tau_i$ for all i, and the schedule defined by the parameters a_i, τ_i, and T_i is feasible.

A simple test is to check whether the total resource utilization is no more than 1. Assume a schedule, defined by the above set of parameters, is feasible. Let T be an arbitrary positive number and let m_i be the number of full dwell intervals for band i that occur in the interval $[0, T]$. Since the dwell intervals for different bands do not overlap, we have

$$\sum_{i=1}^{n} m_i \, \tau_i \quad \leqslant \quad T.$$

Now m_i is equal either to $\lfloor T/T_i \rfloor$ or to $\lfloor T/T_i \rfloor + 1$, and this implies that $(T/T_i - 1) < m_i$.[1] This gives

$$\sum_{i=1}^{n} \tau_i \, (\frac{T}{T_i} - 1) \quad < \quad T,$$

that is,

$$T \, \left[\left(\sum_{i=1}^{n} \frac{\tau_i}{T_i} \right) - 1 \right] \quad < \quad \sum_{i=1}^{n} \tau_i .$$

Since this inequality holds for arbitrary T, we must have

$$\sum_{i=1}^{n} \frac{\tau_i}{T_i} \quad \leqslant \quad 1. \tag{1}$$

The sum on the left-hand side of this equation is the total resource utilization. If this quantity is more than 1, then no feasible schedule can be obtained.

Condition (1) is clearly necessary, but as shown by the preceding counterexample, it is not sufficient to ensure feasibility. Although $1/2 + 1/3 < 1$, there is no feasible schedule for $\tau_1 = 1$, $T_1 = 2$, and $\tau_2 = 1$, $T_2 = 3$. In practice, it is acceptable to reduce the revisit time for any band: spending more time covering band i than required does not usually reduce the overall performance. In this simple example, we could then solve the problem by changing T_2 to 2. Under these new conditions, we can construct a feasible schedule. However, decreasing the revisit times does not always work. For example, take $n = 3$ and the following parameters:

i	τ_i	T_i
1	1	2
2	1	3
3	1	7

Although the total resource utilization $1/2+1/3+1/7$ is smaller than 1, this set of parameters does not admit a feasible schedule. Reducing the revisit times for band 2 or 3 does not work either. To accommodate band 2, it is necessary to change T_2 to 2, but once this is done, the resource utilization for the two first bands equals 1, and there is no resource available for band 3.

This paper presents necessary and sufficient conditions on dwell times and revisit times to ensure schedule feasibility. Section 2 introduces the notations and gives basic definitions

[1] $\lfloor T/T_i \rfloor$ denotes the largest integer smaller than or equal to T/T_i.

and results. Section 3 presents the essential mathematical notions used for obtaining feasibility results, namely, a generalization of the notion of greatest common divisor to arbitrary real numbers. Section 4 gives the key results of this paper. It establishes necessary and sufficient conditions of schedule feasibility. Section 5 discusses the algorithmic complexity of the problem and presents an algorithm for computing schedules. Section 6 summarizes the results presented in the paper.

2 Patterns and Schedules

In any schedule, the dwell intervals for a frequency band are characterized by three parameters: the dwell time, the revisit time, and the start of the first dwell interval. We call such a triple of parameters a *scan pattern* or *pattern* for short.

Definition 1 *A scan pattern is a triple* (a, τ, T) *of real numbers such that* $0 < \tau < T$ *and* $0 \leqslant a \leqslant T - \tau$.

Given a scan pattern $A = (a, \tau, T)$, the successive dwell intervals $[a_t, b_t)$ of A (for $t \in \mathbb{N}$) are defined by the following equations[2]:

$$\begin{aligned} a_t &= a + t\,T \\ b_t &= a_t + \tau. \end{aligned}$$

All the dwell intervals are then of length τ and two successive dwell intervals are separated by a delay $T - \tau$. We denote by $\underline{A}$ the set of reals that belong to any of these intervals:

$$\underline{A} = \bigcup_{t \in \mathbb{N}} [a_t, b_t).$$

If A corresponds to frequency band i then $\underline{A}$ is the set of times when the receiver covers i.

We denote by $\mathbb{R}^+$ the set of nonnegative real numbers. A time $x \in \mathbb{R}^+$ belongs to $\underline{A}$ if and only if there exists $t \in \mathbb{N}$ such that $a_t \leqslant x < b_t$. To obtain a more convenient condition we extend the notion of remainder of a Euclidean division to real numbers as follows.

- Given any real u, let $\lfloor u \rfloor$ denote the largest integer smaller than or equal to u:

$$\lfloor u \rfloor \in \mathbb{Z} \quad \text{and} \quad \lfloor u \rfloor \leqslant u < \lfloor u \rfloor + 1.$$

- Given any real u and any positive real v, let $u \bmod v$ be defined by

$$u \bmod v = u - \left\lfloor \frac{u}{v} \right\rfloor v.$$

The number $u \bmod v$ is to be interpreted as the remainder of the division of u by v and, similarly, $\lfloor u/v \rfloor$ behaves like the usual quotient of an integer division. We have

$$\begin{aligned} u \bmod v = r \quad &\Longleftrightarrow \quad 0 \leqslant r < u \text{ and } \exists i \in \mathbb{Z} :\ u = v\,i + r \\ \lfloor u/v \rfloor = i \quad &\Longleftrightarrow \quad i \in \mathbb{Z} \text{ and } v\,i \leqslant u < v\,(i+1)\ . \end{aligned}$$

We also say that u is divisible by v or that u is a multiple of v if $u = i\,v$ for some integer i. This is equivalent to $u \bmod v = 0$.

Using these notations, we obtain the following necessary and sufficient condition for x to belong to $\underline{A}$.

[2]Using intervals closed on the left and open on the right simplifies the analysis but is not crucial.

Lemma 1 *If $A = (a, \tau, T)$ then for any $x \in \mathbb{R}^+$, we have*

$$x \in \underline{A} \iff (x - a) \bmod T < \tau.$$

Let $A = (a, \tau, T)$ and $A' = (a', \tau', T')$ be two scan patterns. These two patterns *overlap* if $\underline{A} \cap \underline{A'} \neq \emptyset$. Otherwise, we say that A and A' are *compatible*. Let $[a_t, b_t)_{t \in \mathbb{N}}$ be the dwell intervals of pattern A and $[a'_t, b'_t)_{t \in \mathbb{N}}$ be the dwell intervals of A'. To determine whether A and A' overlap, it is sufficient to consider only the starting points of the dwell intervals.

Lemma 2 *A and A' overlap if and only if there is $t \in \mathbb{N}$ such that $a_t \in \underline{A'}$ or there is $u \in \mathbb{N}$ such that $a'_u \in \underline{A}$.*

The proof is trivial.

Since $a_t = a + t\ T$ and $a'_u = a' + u\ T'$, we obtain the following result by combining Lemma 1 and Lemma 2.

Proposition 3 *Let $A = (a, \tau, T)$ and $A' = (a', \tau', T)$ be two patterns; then A and A' overlap if and only if one of the two following conditions is satisfied:*

$$\exists m \in \mathbb{N} :\ (a - a' + m\ T) \bmod T' < \tau'$$

$$\exists m \in \mathbb{N} :\ (a' - a + m\ T') \bmod T < \tau.$$

A *schedule* S is a set of $n \geqslant 1$ scan patterns $A_i = (a_i, \tau_i, T_i)$ for $i = 1, \dots, n$. The schedule is *feasible* if all these patterns are pairwise compatible, that is, the dwell intervals for different frequency bands do not overlap. Given desired dwell times $\tau_1, \dots, \tau_n$ and desired revisit times $T_1, \dots, T_n$, the problem is to determine whether there exist n real numbers $a_1, \dots, a_n$ such that

- $0 \leqslant a_i \leqslant T_i - \tau_i$ for $i = 1, \dots, n$,
- the schedule S defined by the patterns $A_i = (a_i, \tau_i, T_i)$ is feasible.

We also need an algorithm or a technique for computing the values $a_1, \dots, a_n$ when they exist.

3 Greatest Common Divisor

Proposition 3 indicates that the feasibility problem reduces to finding the range of an expression of the form $(x + m\ T) \bmod T'$ for m varying over the natural numbers, and for fixed reals $T' > 0$, $T > 0$, and x. In other words, we must find all the real numbers c such that $c = (x + m\ T) \bmod T'$ for some $m \in \mathbb{N}$. This is related to the more general problem of finding the reals c such that

$$\exists i \in \mathbb{Z}, j \in \mathbb{Z} : c - x = i\ T + j\ T'.$$

In case T and T' are integers, the numbers of the form $i\ T + j\ T'$ are the multiples of the greatest common divisor of T and T'. To solve the general case, we start by extending the gcd to arbitrary non-zero reals T and T'.

Assume then that T and T' are two non-zero reals. Let E be the set of numbers defined by

$$E \quad = \quad \{y \mid \exists i \in \mathbb{Z}, j \in \mathbb{Z} : \ y = i\,T + j\,T'\}.$$

Clearly if x and $y > 0$ belong to E then $x \bmod y$ also belongs to E. Also, if $x \in E$ and $i \in \mathbb{Z}$ then $i\ x \in E$. In the remainder of this section, we show that this set has a smallest positive element d provided T/T' is a rational number, and that this d is the gcd of T and T'. We then investigate the case when T/T' is irrational.

Lemma 4 *Assume E has a smallest positive element d; then we have for all real y,*

$$y \in E \iff \exists i \in \mathbb{Z} : \ y = i\ d.$$

Proof: Since d is the smallest positive element of E, we have $d \in E$, so any y of the form $i\ d$ is also an element of E. Conversely, assume $y \in E$ and let $r = y \bmod d$. Clearly, r also belongs to E and we have $0 \leqslant r < d$. Since d is the smallest positive element of E, r must be 0. This means that y is a multiple of d. □

Lemma 5 *E has a smallest positive element if and only if T/T' is a rational number.*

Proof: Assume E has a smallest positive element d. Since both T and T' belong to E they are both multiples of d (by the previous lemma). There are then two non-zero integers i and i' such that $T = i\ d$ and $T' = i'\ d$, and $T/T' = i/i'$ is rational. Conversely, assume T/T' is a rational number, that is, there are two integers $p \neq 0$ and $q \neq 0$ such that

$$\frac{T}{T'} \quad = \quad \frac{p}{q}.$$

Let $u = T/p = T'/q$; then we have $u \neq 0$, $T = p\ u$, and $T' = q\ u$. As a consequence, we can write

$$E \quad = \quad \{y \mid \exists i \in \mathbb{Z}, j \in \mathbb{Z} : \ y = (i\ p + j\ q)\ u\}.$$

Consider the set

$$F \quad = \quad \{e \mid e > 0 \text{ and } \exists i \in \mathbb{Z}, j \in \mathbb{Z} : \ e = i\ p + j\ q\}.$$

F is a set of positive integers; it has then a smallest element d_0. If $u > 0$, let $d = d_0\ u$ and otherwise let $d = -d_0\ u$; then d is the smallest positive element of E. □

Lemma 6 *Assume T/T' is rational and let d be the smallest positive element of E; then d is the greatest common divisor of T and T'.*

Proof: Since T and T' belong to E, d divides both by Lemma 4. Since d also belongs to E, it is of the form $i\ T + j\ T'$ for some integers i and j. If c divides both T and T' then c divides $i\ T + j\ T'$, that is, c divides d. □

In case T/T' is not rational, T and T' do not have any common divisors and then they do not have a gcd. By Lemma 5, we also know that the set E does not have a smallest positive element. We strengthen this result by showing that $i\ T + j\ T'$ can be an arbitrarily small positive number.

Lemma 7 *If T/T' is irrational then, for any positive real ϵ, there are two integers i and j such that*

$$0 < i\,T + j\,T' < \epsilon.$$

Proof: Let E^+ be the set of positive elements of E; then E^+ has a greatest lower bound g and $g \geqslant 0$. By Lemma 5, g does not belong to E^+. We have then

$$\forall \epsilon > 0 :\ \exists x \in E :\ g < x < g + \epsilon.$$

Since all elements of E are of the form $i\,T + j\,T'$ for some integers i and j, we just have to show that $g = 0$. Assume this is not the case, that is, $g > 0$. There is then $x \in E$ such that $g < x < g + g/2$, and there is also $y \in E$ such that $g < y < x$. Since $x > y$, $\lfloor y/x \rfloor \geqslant 1$. Let $r = x \bmod y$; then r belongs to E^+ and we have

$$r \quad = \quad x - \lfloor x/y \rfloor\ y \quad \leqslant \quad x - y \quad < \quad g/2.$$

This implies $r < g$ and contradicts the assumption that g is the greatest lower bound of E^+. As a consequence g cannot be positive so we have $g = 0$. □

In summary, we have established the following results:

- If T/T' is a rational number then T and T' have a gcd d, and for all $y \in \mathbb{R}$ there are integers i and j such that $y = i\,T + j\,T'$ if and only if d divides y.
- If T/T' is not a rational number then T and T' do not have a gcd and for all positive real ϵ, there are integers i and j such that $0 < i\,T + j\,T' < \epsilon$.

4 Feasibility Conditions

4.1 Compatibility of Two Patterns

We consider two arbitrary patterns $A = (a, \tau, T)$ and $A' = (a', \tau', T')$ and we study conditions for A and A' to be compatible. By Proposition 3, these two patterns overlap if and only if one or both of the two following properties are satisfied:

$$\exists m \in \mathbb{N} :\ (a - a' + m\,T) \bmod T' < \tau'$$

$$\exists m \in \mathbb{N} :\ (a' - a + m\,T') \bmod T < \tau$$

We consider a slightly more general case and study the existence of integers m that satisfy the inequality

$$(b + m\,T') \bmod T < \tau,$$

where T and T' are positive reals, $0 < \tau < T$, and b is an arbitrary real. Two cases must be considered depending on whether T/T' is rational or not.

Rational Case

Lemma 8 *Assume T/T' is rational and let d be the gcd of T and T'. Let c be a real such that $0 \leqslant c < T$; then*

$$(\exists m \in \mathbb{N}:\ (b + m\ T') \bmod T = c) \quad \Longleftrightarrow \quad d \text{ divides } c - b.$$

Proof: If $(b + m\ T') \mathrm{mod} T = c$ with $m \in \mathbb{N}$ then there is $i \in \mathbb{Z}$ such that $b{+}m\ T'{+}i\ T = c$; so $c - b = m\ T' + i\ T$ and, by the preceding results, d divides $c - b$.

Conversely, assume d divides $c - b$. There are then $i \in \mathbb{Z}$ and $j \in \mathbb{Z}$ such that $c - b = i\ T + j\ T'$, and then $c = b + j\ T' + i\ T$. Since $0 \leqslant c < T$, this implies that $\lfloor (b + j\ T')/T \rfloor = -i$ and then

$$(b + j\ T') \bmod T \quad = \quad c.$$

Since d divides both T and T', there are two positive integers u and v such that $T = u\ d$ and $T' = v\ d$. Hence, we have $v\ T = u\ T'$. Let $k \in \mathbb{N}$ be large enough so that $j + k\ u \geqslant 0$ and let $m = j + k\ u$. Then $m \in \mathbb{N}$ and we have

$$\begin{aligned} (b + m\ T') \bmod T &= (b + j\ T' + k\ u\ T') \bmod T \\ &= (b + j\ T' + k\ v\ T) \bmod T \\ &= (b + j\ T') \bmod T \\ &= c. \ \Box \end{aligned}$$

Assuming that T/T' is rational and that $d = \gcd(T, T')$, let B be the set of reals of the form $(b + m\ T') \bmod T$ for $m \in \mathbb{N}$:

$$B \quad = \quad \{c \mid \exists m \in \mathbb{N}:\ c = (b + m\ T') \bmod T\}.$$

By Lemma 8, this set can equivalently be defined by

$$B \quad = \quad \{c \mid 0 \leqslant c < T \ \text{ and } \ \exists k \in \mathbb{Z}:\ c = b + k\ d\}.$$

B is then a nonempty finite set and its smallest element is $c_0 = b \bmod d$. This leads to the following proposition.

Proposition 9 *Assume T/T' is rational and let $d = \gcd(T, T')$; then*

$$(\exists m \in \mathbb{N}:\ (b + m\ T') \bmod T < \tau) \quad \Longleftrightarrow \quad b \bmod d < \tau.$$

Irrational Case

In case T/T' is irrational then $(b + m\ T') \bmod T$ can be arbitrarily small. To show this, we first refine Lemma 7 as follows.

Lemma 10 *If T/T' is irrational then for any positive real ϵ, the two following properties are satisfied:*

$$\exists i \in \mathbb{Z}, j \in \mathbb{Z}:\ j \geqslant 0 \ \text{ and } \ 0 < i\ T + j\ T' < \epsilon$$

$$\exists i \in \mathbb{Z}, j \in \mathbb{Z}:\ j \leqslant 0 \ \text{ and } \ 0 < i\ T + j\ T' < \epsilon$$

Proof: Let $\alpha = \min(\epsilon, T')$. By Lemma 7, there are two integers i and j such that $0 < i\,T + j\,T' < \alpha$. Let $x = i\,T + j\,T'$, $k = \lfloor T'/x \rfloor$, and $y = T' \operatorname{mod} x$. Since $0 < x < \alpha \leqslant T'$, we have $k \geqslant 1$. By definition of mod, we have $0 \leqslant y < x$ and since T/T' is not rational, $y \neq 0$. Furthermore, $y = T' - k\,x$, so we obtain

$$0 < i\,T + j\,T' < \epsilon \;\text{ and }\; 0 < -k\,i\,T + (1 - k\,j)\,T' < \epsilon.$$

It is easy to see that j and $1 - k\,j$ cannot be both positive or both negative. □

Proposition 11 *If T/T' is irrational then for any positive real ϵ, there is $m \in \mathbb{N}$ such that*

$$(b + m\,T') \bmod T < \epsilon.$$

Proof: If $\epsilon \geqslant T$, the inequality is trivially true for any m, so we can assume $\epsilon < T$.

Our objective is to obtain a real y of the form $i\ T + m\ T'$ with $m \in \mathbb{N}$ and such that $-b < y < \epsilon - b$. For such a y, we get

$$0 < b + m\,T' + i\,T < \epsilon.$$

Since $\epsilon < T$ this implies that $\lfloor (b + m\,T')/T \rfloor = -i$ and then that

$$(b + m\,T') \bmod T < \epsilon.$$

To obtain appropriate y and m, we consider the two following cases:

- $b \leqslant 0$. By the preceding lemma, there are $i \in \mathbb{Z}$ and $j \in \mathbb{Z}$ such that $j \geqslant 0$ and $0 < i\,T + j\,T' < \epsilon$.
- $b > 0$. By the preceding lemma, there are $i \in \mathbb{Z}$ and $j \in \mathbb{Z}$ such that $j \leqslant 0$ and $0 < i\,T + j\,T' < \epsilon$.

In both cases, let $x = i\,T + j\,T'$, $k = \lfloor -b/x \rfloor + 1$, and $y = k\,x$. This implies $-b < y < \epsilon - b$ and $y = i\ k\ T + j\ k\ T'$, so we can take $m = j\ k$. In the first case, $k \geqslant 1$ and $j \geqslant 0$, and, in the second case, $k \leqslant 0$ and $j \leqslant 0$ so we have $m \geqslant 0$ in both cases as required. □

Main Theorem

The following theorem gives a necessary and sufficient condition for two scan patterns to be compatible. The theorem is an immediate consequence of the preceding propositions.

Theorem 12 *Let $A = (a, \tau, T)$ and $A' = (a', \tau', T')$ be two scan patterns. A and A' are compatible if and only if T/T' is rational and the two following conditions are satisfied:*

$$\begin{array}{rcl} (a' - a) \bmod d & \geqslant & \tau \\ (a - a') \bmod d & \geqslant & \tau', \end{array}$$

where d is the greatest common divisor of T and T'.

In the remainder of this section, we assume that T/T' is a rational number and that $d = \gcd(T, T')$. Assume A and A' are compatible. Using the theorem, we get

$$(a' - a) \bmod d + (a - a') \bmod d \quad \geqslant \quad \tau + \tau'.$$

For any real x that is not a multiple of d we have

$$x \bmod d + (-x) \bmod d \quad = \quad d,$$

so the preceding inequality gives $\tau + \tau' \leqslant d$. We have proved the following important property.

Proposition 13 *If $A = (a, \tau, T)$ and $A' = (a', \tau', T')$ are compatible then*

$$\tau + \tau' \quad \leqslant \quad d.$$

As a consequence, we have

$$\frac{\tau}{d} + \frac{\tau'}{d} \quad \leqslant \quad 1,$$

and since $d \leqslant T$ and $d \leqslant T'$,

$$\frac{\tau}{T} + \frac{\tau'}{T'} \quad \leqslant \quad 1.$$

This confirms a result we already established. For two scan patterns to be compatible, their resource utilization must be no more than 1.

Proposition 13 shows that the condition $\tau + \tau' \leqslant d$ is necessary for two compatible patterns $A = (a, \tau, T)$ and $A' = (a', \tau', T')$ to exist. The condition is actually sufficient.

Proposition 14 *Let τ, τ', T, and T' be such that $0 < \tau < T$ and $0 < \tau' < T'$. There exist a and a' such that $0 \leqslant a \leqslant T - \tau$, $0 \leqslant a' \leqslant T' - \tau'$, and the patterns $A = (a, \tau, T)$ and $A' = (a', \tau', T')$ are compatible if and only if $\tau + \tau' \leqslant d$.*

Proof: We have already shown that the condition was necessary. Now assume $\tau + \tau' \leqslant d$ and take $a = 0$ and $a' = \tau$. Since $0 < \tau < d$, we get

$$\begin{aligned}(a' - a) \bmod d \quad &= \quad \tau \bmod d \quad = \quad \tau \\ (a - a') \bmod d \quad &= \quad -\tau \bmod d \quad = \quad d - \tau \quad \geqslant \quad \tau'.\end{aligned}$$

We have $a' \leqslant T' - \tau'$ since $a' = \tau$ and $\tau + \tau' \leqslant d \leqslant T$, and we also have $a \leqslant T - \tau$ since $a = 0$. By Theorem 12, the two patterns $A = (a, \tau, T)$ and $A' = (a', \tau', T')$ are compatible. □

4.2 Schedule Feasibility

Consider now a schedule S that consists of n patterns $A_i = (a_i, \tau_i, T_i)$ for $i = 1, \ldots, n$. The schedule is feasible if all these patterns are pairwise compatible. The results from the previous section settle the case $n = 2$. If $n = 2$, S is feasible if and only if all the following conditions are satisfied:

- T_1/T_2 is rational,
- $(a_2 - a_1) \bmod d \geqslant \tau_1$,
- $(a_1 - a_2) \bmod d \geqslant \tau_2$,

where $d = \gcd(T_1, T_2)$. Furthermore, given τ_1, τ_2, T_1, and T_2 such that T_1/T_2 is rational, a corresponding feasible schedule S exists if and only if $\tau_1 + \tau_2 \leqslant d$.

Theorem 12 and Proposition 13 immediately generalize to arbitrary $n \geqslant 2$.

Theorem 15 *The schedule S is feasible if and only if, for all i and j in $\{1, \ldots, n\}$ such that $i \neq j$, the two following conditions are satisfied:*

- *T_i/T_j is rational*
- $(a_i - a_j) \bmod \gcd(T_i, T_j) \ \geqslant \ \tau_j$.

Proposition 16 *If the schedule S is feasible then for all i and j in $\{1, \ldots, n\}$ such that $i \neq j$,*

$$\tau_i + \tau_j \ \leqslant \ \gcd(T_i, T_j).$$

On the other hand, Proposition 14 does not generalize. The conditions given by Proposition 16 are necessary but not sufficient to ensure the existence of a schedule with revisit times $\tau_1, \ldots, \tau_n$ and dwell times $T_1, \ldots, T_n$, unless $n = 2$. To see this, consider the case where $n = 3$, $T_1 = T_2 = T_3 = 2$, and $\tau_1 = \tau_2 = \tau_3 = 1$. Although we have

$$\begin{aligned}
\tau_1 + \tau_2 &\leqslant \gcd(T_1, T_2) \\
\tau_1 + \tau_3 &\leqslant \gcd(T_1, T_3) \\
\tau_2 + \tau_3 &\leqslant \gcd(T_2, T_3),
\end{aligned}$$

there is no feasible schedule for these parameters since the total resource utilization is more than 1.

Theorem 15 shows that constructing a schedule is equivalent to solving a system of inequalities. Assuming all the quotient T_i/T_j are rational numbers, schedule feasibility is equivalent to solving a system S of n^2 inequalities of the following form:

$$\begin{array}{cccccc}
(a_1 - a_2) & \bmod & \gcd(T_1, T_2) & \geqslant & \tau_2 \\
 & & \vdots & & \\
(a_1 - a_n) & \bmod & \gcd(T_1, T_n) & \geqslant & \tau_n \\
(a_2 - a_1) & \bmod & \gcd(T_2, T_1) & \geqslant & \tau_1 \\
 & & \vdots & & \\
(a_2 - a_n) & \bmod & \gcd(T_2, T_n) & \geqslant & \tau_n \\
 & & \vdots & & \\
(a_n - a_1) & \bmod & \gcd(T_n, T_1) & \geqslant & \tau_1 \\
 & & \vdots & & \\
(a_n - a_{n-1}) & \bmod & \gcd(T_n, T_{n-1}) & \geqslant & \tau_{n-1}
\end{array}$$

$$\begin{array}{ccccc}
0 & \leqslant & a_1 & \leqslant & T_1 - \tau_1 \\
 & & \vdots & & \\
0 & \leqslant & a_n & \leqslant & T_n - \tau_n .
\end{array}$$

Proposition 16 and the test on resource utilization give two necessary conditions for this system to have solutions. The following section gives an algorithm for constructing a feasible schedule by solving the inequalities.

5 Algorithms

We explore practical ways of constructing a feasible schedule, given parameters $T_1, \ldots, T_n$ and $\tau_1, \ldots, \tau_n$. For this purpose, we restrict our attention to the case where all the parameters are integers and we also search for integer solutions of the system of inequalities. This is a reasonable restriction since the temporal quantities (i.e., T_i, τ_i, a_i) must be multiples of a common time unit, as an implementation must rely on a base discrete clock. Under these assumptions, schedule feasibility can be transformed into a linear integer programming problem that can be solved using standard algorithms [2]. However, more specialized algorithms that take into account the peculiarities of the problem are likely to be more efficient in practice. We present one such algorithm in Section 5.3 but, first, we study the theoretical complexity of the problem and we discuss techniques for reducing the size of the solution space.

5.1 Theoretical Complexity

The scan-scheduling feasibility problem can be described as follows:

- Each instance of the problem is given by an integer $n \geqslant 1$, n dwell times $T_1, \ldots, T_n$ and n revisit times $\tau_1, \ldots, \tau_n$. All the dwell times and revisit times must be positive integers and satisfy $\tau_i < T_i$.
- The question is to determine whether there exist n integers $a_1, \ldots, a_n$ such that the schedule defined by the n patterns $A_i = (a_i, T_i, \tau_i)$ is feasible.

We denote this problem by SCAN-SCHEDULING. The two following propositions show that SCAN-SCHEDULING is NP-complete.

Proposition 17 SCAN-SCHEDULING *belongs to NP.*

Proof: By the preceding results, solving an instance of SCAN-SCHEDULING is equivalent to solving the system of inequalities S. Because of the constraint

$$0 \leqslant a_i \leqslant T_i - \tau_i,$$

each variable a_i can take a finite number of values. There is then a finite number of candidate solutions. A nondeterministic algorithm for SCAN-SCHEDULING need only guess n values $a_1, \ldots, a_n$ and check whether they satisfy the system S. This can be done in polynomial time, as there exists a polynomial-time algorithm (Euclid's algorithm [2]) for computing the required greatest common divisors. $\square$

Proposition 18 SCAN-SCHEDULING *is NP-hard.*

Proof: We show this by reducing PARTITION to SCAN-SCHEDULING. PARTITION is defined as follows [1]:

- Each instance of the problem is given by a finite set A and a mapping s from A to $\mathbb{N}^+$.
- The question is to determine whether there exists a subset A' of A such that

$$\sum_{x \in A'} s(x) \quad = \quad \sum_{x \in A - A'} s(x).$$

This problem is NP-complete as shown in [1].

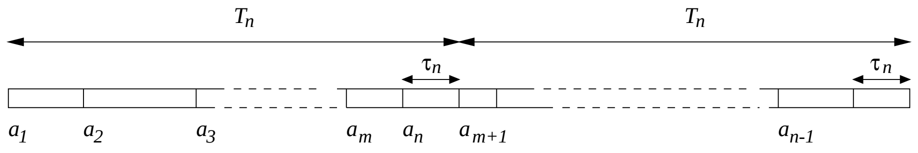
Given an instance (A, s) of PARTITION, an instance of SCAN-SCHEDULING can be constructed as follows:

- $n = |A| + 1$,
- We choose an arbitrary value for τ_n, say $\tau_n = 1$, and set
$$T_n \quad = \quad \tau_n + \left\lfloor \frac{\sum_{x \in A} s(x)}{2} \right\rfloor .$$
- Let $x_1, \ldots, x_{n-1}$ be the elements of A; then for $i = 1, \ldots, n-1$, we define τ_i and T_i by
$$\begin{aligned} \tau_i &= s(x_i) \\ T_i &= 2T_n. \end{aligned}$$

The resource utilization is maximal:
$$\sum_{i=1}^{n} \frac{\tau_i}{T_i} \quad = \quad 1,$$
and we have $\gcd(T_i, T_j) = T_n$ if $i = n$ or $j = n$ and $\gcd(T_i, T_j) = 2T_n$ otherwise.

If A' is a solution of the PARTITION instance, we have
$$\sum_{x \in A'} s(x) \quad = \quad \sum_{x \in A - A'} s(x).$$
We can assume without loss of generality that $A' = \{x_1, \ldots, x_m\}$ for some index $m < n-1$, and then
$$\sum_{i=1}^{m} s(x_i) \quad = \quad \sum_{i=m+1}^{n-1} s(x_i).$$
By definition of T_n and $\tau_1, \ldots, \tau_{n-1}$ this gives
$$\sum_{i=1}^{m} \tau_i \quad = \quad \sum_{i=m+1}^{n-1} \tau_i \quad = \quad T_n - \tau_n.$$
Let $a_1, \ldots, a_n$ be defined as follows:
$$\begin{aligned} a_1 &= 0 \\ a_2 &= a_1 + \tau_1 \\ &\vdots \\ a_m &= a_{m-1} + \tau_{m-1} \\ a_{m+1} &= T_n \\ a_{m+2} &= a_{m+1} + \tau_{m+1} \\ &\vdots \\ a_{n-1} &= a_{n-2} + \tau_{n-2} \\ a_n &= a_m + \tau_m \quad = \quad T_n - \tau_n. \end{aligned}$$

It is routine to check that $(a_1, \ldots, a_n)$ satisfies the system of inequalities for this instance of SCAN-SCHEDULING. The corresponding schedule is as follows:



Conversely, given a solution $(a_1, \ldots, a_n)$ to the SCAN-SCHEDULING instance, we obtain a solution of the PARTITION instance by reversing the previous construction. We consider the interval $[a_n, a_n + 2T_n)$. Since it is of length $T_i = 2Tn$, this interval must contain a dwell interval $[u_i, u_i + \tau_i)$ for all $i = 1, \ldots, n-1$. It also contains two dwell intervals for band n, namely, the two intervals $[a_n, a_n + \tau_n)$ and $[a_n + T_n, a_n + T_n + \tau_n)$. We define the set A' as follows:

$$A' \quad = \quad \{x_i \mid u_i < a_n + T_n\},$$

so that,

$$\sum_{x \in A'} s(x) \quad = \quad \sum_{u_i < a_n + T_n} \tau_i.$$

Since the intervals $[u_i, u_i + \tau_i)$ for $x_i \in A'$ do not overlap and are included in $[a_n + \tau_n, a_n + T_n)$, we have

$$\sum_{x \in A'} s(x) \quad \leqslant \quad T_n - \tau_n. \tag{2}$$

Similarly,

$$\sum_{x \in A - A'} s(x) \quad = \quad \sum_{u_i \geqslant a_n + T_n} \tau_i \quad \leqslant \quad T_n - \tau_n. \tag{3}$$

By construction, we know that

$$\sum_{x \in A'} s(x) + \sum_{x \in A - A'} s(x) \quad = \quad \sum_{x \in A} s(x) \quad \geqslant \quad 2(T_n - \tau_n).$$

Using (2) and (3), we obtain then

$$\sum_{x \in A'} s(x) \quad = \quad \sum_{x \in A - A'} s(x) \quad = \quad T_n - \tau_n,$$

that is, A' is a solution of the original PARTITION instance.

We have then shown that the PARTITION instance (A, s) has a solution if and only if the SCAN-SCHEDULING instance constructed from it has a solution. Clearly, the construction can be done in polynomial time. Since PARTITION is NP-complete, SCAN-SCHEDULING is NP-hard. □

Propositions 17 and 18 imply that SCAN-SCHEDULING is NP-complete. It is then likely that any schedule-construction algorithm has exponential time complexity in the worst case. The following section describes an algorithm for constructing a feasible scan schedule and gives some experimental results on its average complexity.

5.2 Search-Space Reduction

The construction of a feasible schedule requires solving the system of inequalities S defined previously. We are given n dwell times $\tau_1, \ldots, \tau_n$ and n revisit times $T_1, \ldots, T_n$, and we look for n integers $a_1, \ldots, a_n$ that satisfy the two following sets of constraints:

$$S_0 : \left\{ \begin{array}{ccccc} (a_1 - a_2) & \mathrm{mod} & d_{1,2} & \geqslant & a_2 \\ & & \vdots & & \\ (a_n - a_{n-1}) & \mathrm{mod} & d_{n,n-1} & \geqslant & a_{n-1} \end{array} \right.$$

$$S_1 : \left\{ \begin{array}{ccccc} 0 & \leqslant & a_1 & \leqslant & T_1 - \tau_1 \\ & & \vdots & & \\ 0 & \leqslant & a_n & \leqslant & T_n - \tau_n, \end{array} \right.$$

where $d_{i,j} = \gcd(T_i, T_j)$. Because of S_1 there are a priori at most

$$\prod_{i=1}^{n} (T_i - \tau_i + 1)$$

candidate solutions to examine. The following results show that the solution space can often be considerably reduced.

Lemma 19 *Let $(a_1, \ldots, a_n)$ be a solution of S_0. Given $k \in \{1, \ldots, n\}$, let $(a'_1, \ldots, a'_n)$ be defined by*

$$a'_i \quad = \quad (a_i - a_k) \bmod T_i$$

then $(a'_1, \ldots, a'_n)$ is a solution of S.

Proof: For every i, a'_i is equal to $a_i - a_k + u_i T_i$, for some integer u_i. We then have

$$a'_i - a'_j \quad = \quad a_i - a_j + u_i T_i - u_j T_j.$$

Since $d_{i,j}$ divides both T_i and T_j, this gives

$$(a'_i - a'_j) \bmod d_{i,j} \quad = \quad (a_i - a_j) \bmod d_{i,j}.$$

The tuple $(a'_1, \ldots, a'_n)$ satisfies then S_0.

Let $r = (a_i - a_k) \bmod d_{i,k}$. Since $(a_1, \ldots, a_n)$ is a solution of S_0, we have

$$\begin{array}{rcl} (a_i - a_k) \bmod d_{i,k} & \geqslant & \tau_k \\ (a_k - a_i) \bmod d_{i,k} & \geqslant & \tau_i, \end{array}$$

and then $\tau_k \leqslant r \leqslant d_{i,k} - \tau_i$. We also have $T_i = b d_{i,k}$ for some positive integer b. Now $a_i - a_j$ is equal to $q d_{i,j} + r$, for some $q \in \mathbb{Z}$, and q can be written

$$q \quad = \quad pb + c$$

where $0 \leqslant c \leqslant b-1$. This gives

$$\begin{aligned} a_i - a_k &= pbd_{i,k} + cd_{i,k} + r \\ &= pT_i + cd_{i,k} + r, \end{aligned}$$

and, as a consequence,

$$\begin{aligned} a'_i &= (a_i - a_k) \bmod T_i \\ &= cd_{i,k} + r \\ &\leqslant (b-1)d_{i,k} + d_{i,k} - \tau_i \\ &\leqslant T_i - \tau_i. \end{aligned}$$

This shows that $(a'_1, \ldots, a'_n)$ satisfies S_1. □

This lemma says that if S has any solution then there exists a solution $(a'_1, \ldots, a'_n)$ with $a'_k = 0$. Intuitively, this simply amounts to translating the scan schedule so that the first dwell interval for band k starts at 0. When searching for a solution to S, we can then pick an arbitrary variable a_k and set it to 0.

We can also reduce the interval of possible values for a_i. For $i \in \{1, \ldots, n\}$, let L_i be the least common multiple of the numbers

$$d_{i,1}, \ldots, d_{i,i-1}, d_{i,i+1}, \ldots, d_{i,n}.$$

Since T_i is a multiple of each of these numbers, T_i is also a multiple of L_i.

Lemma 20 *Assuming $(a_1, \ldots, a_n)$ is a solution of S, let $(a'_1, \ldots, a'_n)$ be defined by*

$$a'_i = a_i \bmod L_i,$$

then $(a'_1, \ldots, a'_n)$ is a solution of S.

Proof: Every a'_i is of the form $a_i + u_i L_i$ for some integer $u_i \in \mathbb{Z}$. Then $a'_i - a'_j = a_i - a_j + u_i L_i - u_j L_j$, and, since $d_{i,j}$ divides both L_i and L_j,

$$(a'_i - a'_j) \bmod d_{i,j} = (a_i - a_j) \bmod d_{i,j}.$$

Since $a_i \geqslant 0$, $a'_i = a_i \bmod L_i$ cannot be larger than a_i, so we have

$$0 \leqslant a'_i \leqslant T_i - \tau_i.$$

Hence, $(a'_1, \ldots, a'_n)$ is a solution of S. □

As a consequence of this lemma, it is sufficient to search for solutions $(a_1, \ldots, a_n)$ of S_0 where $0 \leqslant a_i < L_i$. It is also easy to see that the solution $(a'_1, \ldots, a'_n)$ as defined in Lemma 20 is such that $0 \leqslant a'_i \leqslant L_i - \tau_i$ for all i. This means that we can replace S_1 by the following inequalities:

$$\left\{ \begin{array}{ccccc} 0 & \leqslant & a_1 & \leqslant & L_1 - \tau_1 \\ & & \vdots & & \\ 0 & \leqslant & a_n & \leqslant & L_n - \tau_n. \end{array} \right.$$

Since L_i divides T_i, the upper bound on a_i can be much smaller than $T_i - \tau_i$. The size of the search space is now reduced to

$$\prod_{i=1}^{n} (L_i - \tau_i + 1).$$

It is possible to further reduce the search space using the following theorem whose proof is given in the following section. Let $M_1, \dots, M_n$ be defined as follows:

$$\begin{array}{rcl} M_1 & = & 1 \\ M_2 & = & d_{1,2} \\ M_3 & = & \mathrm{lcm}(d_{1,3}, d_{2,3}) \\ & \vdots & \\ M_n & = & \mathrm{lcm}(d_{1,n}, \dots, d_{n-1,n}) \end{array}$$

Every M_i is then a divisor of L_i.

Theorem 21 *If S_0 has solutions, then S_0 has a solution $(a_1, \dots, a_n)$ such that*

$$\begin{array}{ccccc} 0 & \leqslant & a_1 & < & M_1 \\ & & \vdots & & \\ 0 & \leqslant & a_n & < & M_n. \end{array}$$

Since $M_1 = 1$, this solution is such that $a_1 = 0$ and, as shown in the proof of Lemma 19, $(a_1, \dots, a_n)$ is also a solution of S. To solve S, it is then sufficient to search for solutions of S_0 where $a_1 = 0$, $a_2 < M_2, \dots, a_n < M_n$. Since every M_i divides L_i, the search space is potentially much smaller than that given by the preceding lemma.

The theorem also suggests an incremental approach for constructing scan schedules. We start by setting a_1 to 0. Then we look for a positive integer $a_2 < M_2 = d_{1,2}$ such that

$$(a_1 - a_2) \bmod d_{1,2} \ \geqslant\ \tau_2 \ \text{ and } \ (a_2 - a_1) \bmod d_{1,2} \ \geqslant\ \tau_1,$$

or, equivalently,

$$\tau_2 \ \leqslant\ (a_1 - a_2) \bmod d_{1,2} \ \leqslant\ d_{1,2} - \tau_1.$$

If such an a_2 is found, we now look for $a_3 < M_3$ such that

$$\tau_3 \ \leqslant\ (a_1 - a_3) \bmod d_{1,3} \ \leqslant\ d_{1,3} - \tau_1$$

$$\tau_3 \ \leqslant\ (a_2 - a_3) \bmod d_{2,3} \ \leqslant\ d_{2,3} - \tau_2.$$

If we can find such an a_3, we proceed incrementally with a_4; otherwise, we try another value for a_2.

Proof of Theorem 21

Theorem 21 relies on the following variant of the Chinese Remainder Theorem.

Proposition 22 *Let d_1 and d_2 be two non-zero integers and let a_1 and a_2 be two integers. There exists an integer x such that d_1 divides $x - a_1$ and d_2 divides $x - a_2$ if and only if $\gcd(d_1, d_2)$ divides $a_1 - a_2$.*

Proof: If d_1 divides $x - a_1$ and d_2 divides $x - a_2$, then $\gcd(d_1, d_2)$ divides both $x - a_1$ and $x - a_2$, and, as a consequence, $\gcd(d_1, d_2)$ divides $(x - a_2) - (x - a_1) = a_1 - a_2$.

Conversely, let us assume $\gcd(d_1, d_2)$ divides $a_1 - a_2$. By definition, there are two integers u and v such that $\gcd(d_1, d_2) = ud_1 + vd_2$; $a_1 - a_2$ can then be written

$$a_1 - a_2 \quad = \quad \alpha u d_1 + \alpha v d_2,$$

where $\alpha \in \mathbb{Z}$. Let $x = a_1 - \alpha u d_1$; then we have

$$\begin{array}{rcl} x - a_1 & = & -\alpha u d_1 \\ x - a_2 & = & a_1 - a_2 - \alpha u d_1 \quad = \quad \alpha v d_2. \end{array}$$

Hence, d_1 divides $x - a_1$ and d_2 divides $x - a_2$. $\square$

The Chinese Remainder Theorem is a special case of the previous proposition where d_1 and d_2 are mutually prime, that is, $\gcd(d_1, d_2) = 1$. The following lemmas are used to generalize Proposition 22 to n numbers $d_1, \ldots, d_n$.

Lemma 23 *Let x be an integer such that d_1 divides $x - a_1$ and d_2 divides $x - a_2$. For any integer y, the two following propositions are equivalent:*

- *d_1 divides $y - a_1$ and d_2 divides $y - a_2$*
- *$\mathrm{lcm}(d_1, d_2)$ divides $y - x$.*

Proof: If d_1 divides $y - a_1$ and d_2 divides $y - a_2$ then d_1 divides $(y - a_1) - (x - a_1) = y - x$ and d_2 divides $(y - a_2) - (x - a_2) = y - x$ so $y - x$ is a common multiple of d_1 and d_2, that is, $\mathrm{lcm}(d_1, d_2)$ divides $y - x$.

Conversely, if $\mathrm{lcm}(d_1, d_2)$ divides $y - x$ then d_1 divides $y - x$ and d_2 divides $y - x$, so d_1 divides $(y - x) + (x - a_1)$ and d_2 divides $(y - x) + (x - a_2)$, that is, d_1 divides $y - a_1$ and d_2 divides $y - a_2$. $\square$

Lemma 24 *For all non-zero integers x, y, and z, we have*

$$\gcd(x, \mathrm{lcm}(y, z)) \quad = \quad \mathrm{lcm}(\gcd(x, y), \gcd(x, z)).$$

Proof: Let $p_1, \ldots, p_k$ be the prime factors of x, y, and z. There are natural numbers $\alpha_1, \ldots, \alpha_k$, $\beta_1, \ldots, \beta_k$, and $\gamma_1, \ldots, \gamma_k$ such that

$$\begin{array}{rcl} x & = & p_1^{\alpha_1} \ldots p_k^{\alpha_k} \\ y & = & p_1^{\beta_1} \ldots p_k^{\beta_k} \\ z & = & p_1^{\gamma_1} \ldots p_k^{\gamma_k}. \end{array}$$

Let $\delta_i = \min(\alpha_i, \max(\beta_i, \gamma_i))$ and $\sigma_i = \max(\min(\alpha_i, \beta_i), \min(\alpha_i, \gamma_i))$; then

$$\begin{array}{rcl} \gcd(x, \mathrm{lcm}(y, z)) & = & p_1^{\delta_1} \ldots p_k^{\delta_k} \\ \mathrm{lcm}(\gcd(x, y), \gcd(x, z)) & = & p_1^{\sigma_1} \ldots p_k^{\sigma_k}. \end{array}$$

It is easy to check that $\delta_i = \sigma_i$ so the left-hand sides of these two equations are equal. $\square$

The general form of Proposition 22 is the following.

Proposition 25 *Let $d_1, \ldots, d_n$ be n non-zero integers and $a_1, \ldots, a_n$ be n integers. If we have*

$$\gcd(d_i, d_j) \quad \text{divides} \quad a_i - a_j$$

for all i and j such that $1 \leqslant i < j \leqslant n$, then there exists an integer x such that

$$\begin{array}{ccc} d_1 & \text{divides} & x - a_1 \\ & \vdots & \\ d_n & \text{divides} & x - a_n. \end{array}$$

Proof: We reason by induction on n. The proposition is trivially true if $n = 1$.

Now we assume the proposition true for $n \geqslant 1$. Let $d_1, \ldots, d_{n+1}$ and $a_1, \ldots, a_{n+1}$ be numbers such that $\gcd(d_i, d_j)$ divides $a_i - a_j$ whenever $i < j$.

By Proposition 22, since $\gcd(d_n, d_{n+1})$ divides $a_n - a_{n+1}$, there is $x_0 \in \mathbb{Z}$ such that

$$d_n \quad \text{divides} \quad x_0 - a_n \tag{4}$$

$$d_{n+1} \quad \text{divides} \quad x_0 - a_{n+1}. \tag{5}$$

Let $d'_1, \ldots, d'_n$ and $a'_1, \ldots, a'_n$ be defined as follows:

$$\begin{aligned} d'_i &= d_i \text{ if } i < n \\ d'_n &= \operatorname{lcm}(d_n, d_{n+1}) \\ a'_i &= a_1 \text{ if } i < n \\ a'_n &= x_0. \end{aligned}$$

For two indices i and j such that $1 \leqslant i < j < n$, $\gcd(d'_i, d'_j)$ divides $a'_i - a'_j$ by assumption. For $i < n$ and $j = n$, Lemma 24 gives

$$\begin{aligned} \gcd(d'_i, d'_j) &= \gcd(d_i, \operatorname{lcm}(d_n, d_{n+1})) \\ &= \operatorname{lcm}(\gcd(d_i, d_n), \gcd(d_i, d_{n+1})). \end{aligned}$$

By assumption, $\gcd(d_i, d_n)$ divides $a_i - a_n$. By (4), and since $\gcd(d_i, d_n)$ is a divisor of d_n, $\gcd(d_i, d_n)$ also divides $x_0 - a_n$. It follows that $\gcd(d_i, d_n)$ divides $a_i - x_0 = a'_i - a'_n$. Using (5), we get by a similar reasoning that $\gcd(d_i, d_{n+1})$ divides $a'_i - a'_n$. Hence, $a'_i - a'_n$ is a common multiple of $\gcd(d_i, d_n)$ and $\gcd(d_i, d_{n+1})$ so

$$\gcd(d'_i, d'_n) \quad \text{divides} \quad a'_i - a'_n.$$

We can then use the induction hypothesis with $d'_1, \ldots, d'_n$ and $a'_1, \ldots, a'_n$: there is an integer x such that

$$\begin{array}{ccc} d'_1 & \text{divides} & x - a'_1 \\ & \vdots & \\ d'_n & \text{divides} & x - a'_n. \end{array}$$

Then $d'_n = \mathrm{lcm}(d_n, d_{n+1})$ divides $x - x_0$. By Lemma 23, this implies that d_n divides $x - a_n$ and d_{n+1} divides $x - a_{n+1}$. We have then found an integer x such that

$$\begin{array}{rcl} d_1 & \text{divides} & x - a_1 \\ & \vdots & \\ d_n & \text{divides} & x - a_n \\ d_{n+1} & \text{divides} & x - a_{n+1}. \end{array}$$

The proposition is then satisfied for $n + 1$. By induction it is true for arbitrary n. □

Let $T_1, \dots, T_n$ be n positive integers and let $d_{i,j} = \gcd(T_i, T_j)$. To prove Theorem 21, we show that for any tuple of integers $(a_1, \dots, a_n)$, there exists a tuple $(a'_1, \dots, a'_n)$ such that

$$\begin{array}{ccccc} 0 & \leqslant & a'_1 & < & M_1 \\ & & \vdots & & \\ 0 & \leqslant & a'_n & < & M_n, \end{array}$$

and

$$d_{i,j} \quad \text{divides} \quad (a'_i - a_i) - (a'_j - a_j)$$

for all $i \in \{1, \dots, n\}$ and $j \in \{1, \dots, n\} - \{i\}$.

The construction is incremental. Given an index k such that $1 \leqslant k \leqslant n - 1$ and k numbers $(a'_1, \dots, a'_k)$ that satisfy

$$\begin{array}{ccccc} 0 & \leqslant & a'_1 & < & M_1 \\ & & \vdots & & \\ 0 & \leqslant & a'_k & < & M_k, \end{array}$$

and

$$d_{i,j} \quad \text{divides} \quad (a'_i - a_i) - (a'_j - a_j) \tag{6}$$

for all $i \in \{1, \dots, k\}$ and $j \in \{1, \dots, k\} - \{i\}$, we show that there exists an integer a'_{k+1} such that $0 \leqslant a'_{k+1} < M_{k+1}$ and

$$\begin{array}{rcl} d_{1,k+1} & \text{divides} & (a'_{k+1} - a_{k+1}) - (a'_1 - a_1) \\ & \vdots & \\ d_{k,k+1} & \text{divides} & (a'_{k+1} - a_{k+1}) - (a'_k - a_k). \end{array}$$

First, given $i \in \{1, \dots, k\}$ and $j \in \{1, \dots, k\}$ such that $i < j$, we have

$$\begin{aligned} \gcd(d_{i,k+1}, d_{j,k+1}) &= \gcd(\gcd(T_i, T_{k+1}), \gcd(T_j, T_{k+1})) \\ &= \gcd(T_i, T_j, T_{k+1}) \\ &= \gcd(d_{i,j}, T_{k+1}). \end{aligned}$$

By (6), this implies that $\gcd(d_{i,k+1}, d_{j,k+1})$ divides $(a'_i - a_i) - (a'_j - a_j)$. Using Proposition 25, there is then an integer x such that

$$\begin{array}{ccc} d_{1,k+1} & \text{divides} & x - (a'_1 - a_1) \\ & \vdots & \\ d_{k,k+1} & \text{divides} & x - (a'_k - a_k). \end{array}$$

Let $a'_{k+1} = (a_{k+1} + x) \bmod M_{k+1}$. We have $0 \leqslant a'_{k+1} < M_{k+1}$ and M_{k+1} divides $a'_{k+1} - a_{k+1} - x$. By definition,

$$M_{k+1} \quad = \quad \operatorname{lcm}(d_{1,k+1}, \dots, d_{k,k+1}),$$

so we have

$$\begin{array}{ccc} d_{1,k+1} & \text{divides} & (a'_{k+1} - a_{k+1}) - x \\ & \vdots & \\ d_{k,k+1} & \text{divides} & (a'_{k+1} - a_{k+1}) - x. \end{array}$$

We then obtain

$$\begin{array}{ccc} d_{1,k+1} & \text{divides} & (a'_{k+1} - a_{k+1}) - (a'_1 - a_1) \\ & \vdots & \\ d_{k,k+1} & \text{divides} & (a'_{k+1} - a_{k+1}) - (a'_k - a_k), \end{array}$$

as required.

Starting with $a'_1 = 0$ and iterating this construction for $k = 1, \dots, n-1$, we obtain a tuple $(a'_1, \dots, a'_n)$ such that

$$d_{i,j} \quad \text{divides} \quad (a'_i - a_i) - (a'_j - a_j)$$

whenever $1 \leqslant i < j \leqslant n$. Given two such indices i and j, we get that

$$d_{i,j} \quad \text{divides} \quad (a'_i - a'_j) - (a_i - a_j),$$

which is equivalent to

$$(a'_i - a'_j) \bmod d_{i,j} \quad = \quad (a_i - a_j) \bmod d_{i,j}.$$

Hence, if $(a_1, \dots, a_n)$ is a solution of the system S_0 of inequalities, $(a'_1, \dots, a'_n)$ is another solution of S_0 and, by construction, it satisfies

$$\begin{array}{ccccc} 0 & \leqslant & a'_1 & < & M_1 \\ & & \vdots & & \\ 0 & \leqslant & a'_n & < & M_n. \end{array}$$

This completes the proof of Theorem 21.

5.3 Schedule Construction Algorithm

To perform some experiments, we developed a schedule construction algorithm based on Theorem 21. The algorithm is based on a depth-first search with backtracking. A tuple $(a_1, \dots, a_k)$ is a partial solution of S_0 if the constraints

$$(a_i - a_j) \bmod d_{i,j} \quad \geqslant \quad \tau_j$$

are satisfied for all $i \in \{1, \dots, k\}$ and $j \in \{1, \dots, k\} - \{i\}$. The algorithm attempts to extend such a partial solution by finding a value a_{k+1} such that

$$\begin{array}{ccccc} \tau_{k+1} & \leqslant & (a_1 - a_{k+1}) \bmod d_{1,k+1} & \leqslant & d_{1,k+1} - \tau_1 \\ & & \vdots & & \\ \tau_{k+1} & \leqslant & (a_k - a_{k+1}) \bmod d_{k,k+1} & \leqslant & d_{k,k+1} - \tau_k. \end{array}$$

If such an a_{k+1} can be found, then $(a_1, \dots, a_{k+1})$ is a partial solution of S_0. The algorithm then attempts to extend it incrementally until a full solution $(a_1, \dots, a_n)$ is obtained. If no such a_{k+1} can be found, the algorithm backtracks and looks for another partial solution $(a'_1, \dots, a'_k)$. The algorithm terminates either when a full solution is reached or when all partial solutions have been tried and eliminated. As shown by Theorem 21, the search can be limited to $a_1, \dots, a_n$ such that $0 \leqslant a_i < M_i$. Pseudo code is given in Figure 2 that describes the recursive search procedure used by the algorithm. Given a number k and a partial solution $a = (a_1, \dots, a_k)$, the procedure `Search(k, a)` either returns a full solution that extends $(a_1, \dots, a_k)$ or `nil` if no such solution can found. It uses an auxiliary function `Consistent(a, x)` that tests whether $(a_1, \dots, a_k, x)$ is a partial solution, that is, whether x satisfies the constraints

$$\tau_{k+1} \quad \leqslant \quad (a_i - x) \bmod d_{i,k+1} \quad \leqslant \quad d_{i,k+1} - \tau_i.$$

The search is initiated by calling `Search(0,[])`.

Our actual implementation is based on this simple algorithm but uses a more sophisticated approach to detect early that a partial solution $(a_1, \dots, a_k)$ cannot be extended. For this purpose, we associate domains $D_{k+1}, \dots, D_n$ with the remaining variables $a_{k+1}, \dots, a_n$. Each D_i is the set of values for a_i that are consistent with $(a_1, \dots, a_k)$:

$$D_i \quad = \quad \{x \mid 0 \leqslant x < M_i \text{ and } \forall j \leqslant k : \tau_i \leqslant (a_i - x) \bmod d_{i,j} \leqslant d_{i,j} - \tau_j\}.$$

If D_i is empty then $(a_1, \dots, a_k)$ cannot be extended to a full solution. In addition, we use a heuristic based on the size of the domains $D_{k+1}, \dots, D_n$ to order the search.

Initial experiments show that random instances of the scan-scheduling problem are very likely to violate the necessary conditions of the form

$$\tau_i + \tau_j \quad \leqslant \quad \gcd(T_i, T_j).$$

For n randomly chosen revisit times, it is very likely that at least one of $\gcd(T_i, T_j)$ is very small.

To obtain random instances that do not violate these conditions, we use the following approach. First we choose a base number T and two integers k_0 and k_1 such that $k_0 \leqslant k_1$. Then we construct randomly n revisit times T_i that are all multiples of T, and all between

```
// The problem is stored in two global arrays
// T[1],...,T[n]: revisit times
// t[1],...,t[n]: dwell times

Search(k, a):
  if k=n
    return a   // a full solution has been found
  else
    found := false
    x := 0
    while (not found and x<M[k])
      if (Consistent(a, x))
        b := Search(k+1, add(a, x))
        found := b != nil
      endif
      x := x + 1
    endwhile
    if found
      return b
    else
      return nil
    endif
  endif
```

Figure 2: Schedule Construction Algorithm

$T_{\min} = k_0 T$ and $T_{\max} = k_1 T$. For any two such T_i and T_j, $\gcd(T_i, T_j)$ is then a multiple of the base number T. We then select a maximal dwell time $\tau_{\max}$ no larger than $T/2$ and generate randomly n dwell times $\tau_1, \ldots, \tau_n$ in the range $1, \ldots, \tau_{\max}$. This ensures that $\tau_i + \tau_j \leqslant T$ and then the necessary conditions are satisfied.

By choosing appropriate values of T, k_0, k_1, $\tau_{\max}$, and n, we can control the average utilization of randomly generated instances. Initial experiments show that the practical performance of the search algorithm and the likelihood of an instance being feasible are strongly related to the utilization. Table 1 summarizes some of our first experiments with $T = 100$, $T_{\min} = 1000$, and $T_{\max} = 1500$. Each row of the table corresponds to a different choice of $\tau_{\max}$, and then a different average utilization U. In each case, we generated 100 random instances consisting of 50 revisit times and 50 dwell times, and ran the algorithm on each of these instances with a timeout of 60 s (CPU time). The experiments were performed on a PC with a 550 MHz Pentium III, running Linux 2.2.5-15. The last column of Table 1 shows the average search time for the instances that did not cause a timeout.

6 Conclusion

Determining whether a set of n dwell times and a set of n revisit times are compatible is a central issue in periodic scan scheduling. We have given a mathematical characterization of schedule feasibility as a set of linear inequalities. The problem can be solved algorithmically but is NP-complete. We proposed an algorithm that searches for a solution in a depth-first manner and defined several simplifications that reduce the search space. Initial (and par-

$\tau_{\max}$	U	Feasible	Infeasible	Timeout	CPU time
49	1.00	0	100	0	0.05s
44	0.90	0	100	0	0.19s
39	0.80	0	100	0	0.35s
34	0.70	0	98	2	3.03s
29	0.60	5	63	32	18.13s
24	0.50	60	0	40	1.54s
19	0.40	86	0	14	0.34s
14	0.30	93	0	7	0.13s
9	0.20	100	0	0	0.18s
4	0.10	100	0	0	0.33s

Table 1: Algorithm Performance

tial) experiment showed that high sensor utilization is hard to reach for randomly generated instances.

Open issues include a more careful study of the achievable utilization for a given set of n revisit times. It can be shown that a utilization of 1 can be obtained under certain conditions on the revisit times, for example, if the revisit times are harmonic. One such example is used in the proof of Proposition 18. In general, studying instances of the problem with a special structure that make them easier to solve or that ensures some other desirable property remains unexplored.

References

[1] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 1979.

[2] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, 1989.