Theoretical Computer Science 100 (1992) 105–135
Elsevier

Pinwheel scheduling with two distinct numbers*

Robert Holte, Louis Rosier, Igor Tulchinsky and Donald Varvel

Department of Computer Sciences, The University of Texas at Austin, Austin, TX 78712-1188, USA

Communicated by P. Young
Received February 1989
Revised January 1991

Abstract

Holte, R., L. Rosier, I. Tulchinsky and D. Varvel, Pinwheel scheduling with two distinct numbers, Theoretical Computer Science 100 (1992) 105–135.

"The Pinwheel" is a real-time scheduling problem based on a problem in scheduling satellite ground stations but which also addresses scheduling preventive maintenance. Given a multiset of positive integers $A=\{a_1, a_2, \ldots, a_n\}$, a schedule S for A is an infinite sequence over $\{1, 2, \ldots, n\}$ such that any subsequence of length a_i $(1 \leqslant i \leqslant n)$ contains at least one i. Schedules can always be made cyclic; that is, a segment can be found that can be repeated indefinitely to form an infinite schedule. Interesting questions include determining whether schedules exist, determining the minimum cyclic schedule length, and creating an online scheduler. The "density" of an instance is defined as $d=\sum_{i=1}^{n} 1/a_i$. It has been shown that any instance with $d>1.0$ cannot be scheduled. In the present paper we limit ourselves to instances in which A contains elements having only two distinct values. We prove that all such instances with $d \leqslant 1.0$ can be scheduled, using a scheduling strategy based on balancing. The schedule so created is not always of minimum length, however. We use a related but more complicated method to create a minimum-length cyclic schedule, and prove its correctness. The former is computationally easier to obtain but not necessarily minimal. The latter, although still obtainable in polynomial time, requires significantly more computation. In addition, we show how to use either method to produce a fast online scheduler. Thus, we have solved completely the three major problems for this class of instances.

1. Introduction

The "pinwheel" problem [2] is motivated by the performance requirements of a ground station that processes data from a number of satellites (or mobile

*This work was supported in part by U.S. Office of Naval Research Grant No. N00014-86-K-0763 and National Science Foundation Grant No. CCR-8711579. A preliminary version was presented at the *14th International Symposium on Mathematical Foundations of Computer Science*, Porabka-Kozubnik, Poland.

sensors). The ground station can process data from only one satellite at a time, no preemption of processing is allowed, and the time necessary for acquiring and processing data from a satellite is exactly one "time unit". Each satellite may commence sending data at any time, but must repeat the same data for a specified number of time units. If the interval specified for satellite x is a time units, the ground station can ensure processing its data by assigning it a time slot in *any* interval of length a. A schedule is therefore an infinite sequence of satellite designations such that each satellite is scheduled at short enough intervals that no data can be lost.

The pinwheel is a formalization of the satellite scheduling problem. Given a multiset of positive integers $A=\{a_1, a_2, \ldots, a_n\}$, a schedule S is an infinite sequence over $\{1, 2, \ldots, n\}$ such that any subsequence of length a_i $(1 \leqslant i \leqslant n)$ consecutive entries ("slots") contains at least one i. For example, "1 2 1 2 ..." is a schedule for $A=\{2, 3\}$. Notice that the first (second) satellite is scheduled at least once within any interval consisting of 2 (3) or more "time units". If a schedule exists, there is a finite length string that may be repeated indefinitely to form a schedule. We call this a *cyclic schedule*. The name "pinwheel" derives from this fact. For $A=\{2, 3\}$, for example, the shortest cycle length is 2, corresponding to the cyclic schedule "1, 2".

The *density* of an instance is defined as $\sum_{i=1}^{n} 1/a_i$. The justification for the name "density" is that in a cyclic schedule, i occupies at least $1/a_i$ of the slots. Clearly, if the density of an instance is greater than 1.0 the instance cannot be scheduled. If the density of an instance is 1.0 (termed *dense*) there is insufficient space for any item i to be scheduled any more than this minimum. The schedule can therefore be thought of as being densely packed. If the density of an instance is less than 1.0 (termed *nondense*) some item (or items) i will be scheduled in more than $1/a_i$ of the slots. A pinwheel instance gives rise to three main problems:

1. The *pinwheel decision problem* concerns whether a given instance can be scheduled.

2. The *pinwheel scheduling problem* involves producing a "useful" representation of a schedule. For the satellite scheduling problem, the primary motivation of this work, "useful" means that a ground station controller with limited memory must be able to select the next satellite quickly.

3. The *minimum pinwheel scheduling problem* involves finding a "useful" representation of a minimum length cyclic schedule.

What, then, constitutes a "useful" representation of a schedule? In light of the fact that the minimum cycle length may be exponential in the length of the input [2], we suggest that a fast enough program to select the next satellite might serve better than a portion of an actual schedule. What is needed is a *fast online scheduler* or *FOLS* – a program that generates the scheduling sequence in constant time per item generated. A useful solution to the pinwheel scheduling problem, then, is a program that takes as input an instance of the pinwheel problem and produces as output a corresponding FOLS, provided one exists. A FOLS might take the form of

a program P:

P: α;
Do
β
forever

where α is an initialization code segment that runs in no worse than polynomial time and β is a "simple" segment of straight-line code that can be made to run in precisely a "time unit". On each iteration of the DO-loop β selects items for a fixed number of slots. Thus, P generates the scheduling sequence in constant time per item generated. In [2] a family of complexity classes was defined in terms of the respective complexities of the scheduler generator and scheduler. We will show that the pinwheel problem restricted to instances with only two distinct numbers is in the class "S-P-C," for scheduling-polynomial-constant. That means that there exists a program that runs in polynomial time that determines whether a schedule exists, and if so generates a scheduler that runs in constant time per item scheduled. This, then, constitutes our working definition of a "useful" representation of a schedule.

The pinwheel is one of a growing family of hard-real-time scheduling problems [4, 5, 6, 8], the closest relative of which is the periodic maintenance problem of [9]. The periodic maintenance problem is motivated by the need to schedule a mechanic's time to perform periodic maintenance. Recast into our terminology this problem requires item i to be scheduled *exactly* every a_i slots. That is, if item i is scheduled into slot k, it must also be scheduled into slot $k+pa_i$ for all nonnegative integers p. This is indeed the case for our dense instances [2], so such pinwheel instances are also instances of the periodic maintenance problem. The difference appears in the case of nondense instances. The periodic maintenance problem does not allow an item to be scheduled early. We do not allow empty slots in a pinwheel schedule. Thus, the pinwheel problem is concerned with scheduling the server's time as tightly as possible, while the periodic maintenance problem is concerned with minimizing the downtime of the machines being serviced. Whether the pinwheel or the periodic maintenance problem applies to a particular real-world problem depends on whether it is desirable or acceptable to perform the maintenance slightly early on some occasions.

For every instance of the single-server periodic maintenance problem there is a corresponding instance of the pinwheel problem, and a schedule for the former may be transformed into a schedule for the latter. This may be done by "padding" the periodic maintenance instance with new items whose frequency is the least common multiple (LCM) of the given items, yielding a dense pinwheel instance. The pinwheel instance has a schedule if and only if the original periodic maintenance instance does. A cyclic schedule for the pinwheel instance can then be transformed into a cyclic schedule for the periodic maintenance instance by changing to "blank" all those slots allocated to the new items. Thus, the pinwheel is a generalization of the periodic maintenance problem. Because of the padding, pinwheel instances may be exponentially longer than the corresponding periodic maintenance instances; thus complexity

results may not transfer. Finally, the inclusion is proper, so pinwheel schedules do not generally imply periodic maintenance problem schedules. For example, $A = \{2, 3\}$ has a pinwheel schedule but not a periodic maintenance schedule.

The pinwheel problem has been addressed previously in [2]. There it was determined that if a pinwheel instance can be scheduled, then there exists a cyclic schedule of length no greater than $\prod_{i=1}^{n} a_i$. Exponential length schedules are often necessary. The decision and scheduling problems for dense instances of up to three distinct numbers can be solved in polynomial time. The minimum schedule length for those dense instances that can be scheduled is the LCM of the numbers mentioned in the problem instance. For general dense instances, the complexity of the decision problem appears to depend on the representation of problem instances. With the standard multiset representation it is in NP but is not known to be NP-hard. However, given the compact representation described below, it is NP-hard [2, 7, 1]. Dense instances with only two distinct numbers can always be scheduled. For dense instances with three distinct numbers, a global greatest common divisor greater than one is a necessary but not a sufficient condition for schedulability. For dense instances with four or more distinct numbers, a greatest common divisor greater than one is neither necessary nor sufficient. FOLSs can be constructed in polynomial time for all schedulable dense instances of up to three distinct numbers.

In this paper we investigate pinwheel instances of all densities, but limited to only two distinct numbers. An example is $\{6, 6, 6, 15, 15, 15, 15, 15, 15, 15\}$. A more compact representation is the ordered quadruple (6, 3, 15, 7). The quadruple representation is defined as (x, a, y, b) where x and y are the distinct numbers of the multiset representation and a and b specify the number of occurrences of each. In this representation, x and y specify the *frequencies* and a and b specify cardinality of items with the respective frequencies. We will use this more compact representation for the remainder of this paper. Since we are here treating only the case of two distinct numbers, the following properties hold: (1) $a > 0$, (2) $b > 0$, (3) $x/a > 1$, and (4) $y/b > 1$.

The restriction to only two distinct numbers leads to useful and interesting results. Typically, many of the satellites that must be monitored will be identical. If in fact they have only two distinct periods the resulting schedules have several desirable properties. Furthermore, we have found some of these properties to be intriguing and unintuitive.

This paper contains the first complete set of results for a class of nondense instances. Schedules for dense instances exhibit certain regularities that make them easier to reason about. For instance, the minimum schedule length for instances that can be scheduled is the LCM of the distinct numbers. Slots assigned to item i must occur exactly a_i slots apart. These and related properties do not hold for nondense instances. This difference is reflected in the methods we have used to address the dense and nondense classes. The methods used in dense pinwheel instances and the related periodic maintenance problem involve the use of divisibility and number theory. Nondense instances have required additional techniques, notably concerned with the properties of floor and ceiling functions. Some results concerning dense instances turn

out to be special cases of the present work. Thus, while the proofs in Sections 3 through 5 may be tedious, the resulting theorems are quite powerful.

This paper provides a comprehensive treatment of all three problems for general instances with only two distinct numbers. As it turns out, all such instances with density $\leqslant 1.0$ can be scheduled. That this is so seems neither obvious nor particularly intuitive. There are very simple instances, both dense and nondense, with as few as three distinct numbers, that cannot be scheduled. We introduce three functions of an instance – each of which yields the length of a cyclic schedule. One of these functions yields the minimum such length. The two functions not guaranteed to give the minimum schedule length are easy to compute, and the other can be computed in deterministic polynomial time but seems to require significantly more computation. That any of these functions should yield the length of a cyclic schedule also seems unintuitive. We prove many interesting and intriguing properties of these functions in an effort to reconcile our results with the corresponding results for dense instances – where the length function (LCM) seemed far more intuitive.

Subsequently for each of the length functions introduced we illustrate how to obtain a cyclic schedule of that length and prove its correctness. Our method involves the use of *partitioning functions*. These are functions from the natural numbers to the natural numbers that serve to partition the slots of the potential schedule into two sets – one set for the a items of frequency x and one set for the b items of frequency y. The first such function, called *Place*1, maps the natural numbers into the slot numbers in the first set; the second, called *Place*2, maps the natural numbers into the slot numbers in the second set. For instance, if $A=\{2,3\}$ then we might use as partitioning functions $Place1(i)=2i$ and $Place2(i)=2i+1$. That is, the item of frequency 2 occupies the even-numbered slots and the item of frequency 3 occupies the odd-numbered slots. To be used in this way, functions must have a number of properties, among which is being monotone increasing. The above examples and the partitioning functions we use later with scheduling algorithms are obviously monotone increasing.

Lastly, we show how to develop in each case the corresponding FOLS. The constructions are all polynomial-time implementable; in fact, two are computationally easy. Hence, the pinwheel scheduling problem as well as the minimum-cycle problem for this class of instances belongs to the class S-P-C.

The remainder of this paper is organized as follows. In Section 2 we introduce the various functions that will be used to solve the decision, scheduling, and minimum-cycle problems. We also relate the known facts about dense instances to this set of problems. In Section 3 we introduce and prove an important lemma that is used in Sections 4 and 5. This lemma will be used to show that the partitioning functions never assign the same slot to both frequencies. Section 4 answers the decision problem in the affirmative with the introduction of a scheduling algorithm. It is based on the intuitive notion of distributing the slots allocated to a particular frequency as evenly as possible. It does not in general produce the shortest cyclic schedule, however. In this section we also introduce a FOLS incorporating this scheduling strategy. The FOLS may be generated in deterministic polynomial time.

Section 5 applies the idea of even distribution in a different way to yield a solution of optimal length.

2. Some length functions and their properties

In this section we define four functions concerning the length of cyclic schedules and prove some of their properties. In each case, A refers to an instance $A=(x, a, y, b)$ and n refers to a potential schedule length. The functions are:

1. $H_1(A)$ is the first of three functions that identify cyclic schedule lengths. The proof that a schedule of this length can always be constructed will be presented in Section 4.

2. $H_2(A)$ is similar to $H_1(A)$, and is the second of the three cyclic schedule length functions.

3. $M(A,n)$ gives the difference between the potential schedule length n and the minimum number of slots that must be available in a cyclic schedule of that length.

4. $LM(A)$ represents the least n such that $M(A,n)=0$. $LM(A)$ turns out to be the minimum schedule length. The proof that a schedule of this length can always be constructed will be presented in Section 5. This is the third and final cyclic schedule length function.

We will prove a series of properties for these functions. Of course, an important property of H_1, H_2, and LM is that they represent cyclic schedule lengths. Those theorems will be proved in Sections 4 and 5. In this section we prove the following properties:

- For dense instances A, $H_1(A)=H_2(A)=LM(A)=LCM(x,y)$. This is important because the minimum cyclic schedule length for dense instances is $LCM(x,y)$. This reconciles the present results with the work reported in [2] concerning dense instances. (See Theorems 2.1 and 2.5.)
- $M(A,n)\geqslant 0$ is necessary for there to exist schedules of length n. This will be established by proving that M represents the difference between n and the minimum number of slots the instance A requires in a schedule of length n. Unfortunately, $M(A,n)\geqslant 0$ is not a sufficient condition (See Theorems 2.2 and 2.4 and Corollary 2.3).
- There is no cyclic schedule of length less than $LM(A)$. This will be proved by showing that for the least n for which $M(A,n)$ is nonnegative, $M(A,n)=0$. (See Theorem 2.8 and Corollaries 2.12–2.14.)
- For nondense A, $LM(A)<LCM(x,y)$ and $H_1(A)$ and $H_2(A)$ are at least $LM(A)$. That is, for nondense instances the minimum cyclic schedule length is less than $LCM(x,y)$ and H_1 and H_2 are at least that minimum. This also reconciles with [2], in that the minimum schedule length may now be seen to be less than or equal to $LCM(x,y)$. (See Theorem 2.8.)
- $LM(A)$ may be computed in deterministic polynomial time. Since $LM(A)$ yields the

minimum schedule length, we would like to be able to evaluate it in a reasonable amount of time. (See Theorem 2.15.)

One of the results reported in [2] is that dense instances with only two distinct numbers may always be scheduled. The minimum schedule length for such instances is the LCM of the two numbers. In our notation, for an instance $A=(x, a, y, b)$, the minimum schedule length for a dense instance (i.e., those where $a/x+b/y=1$) is $LCM(x, y)$. Extending this research to nondense instances ($a/x+b/y<1$) leads to several questions. Can all nondense instances be scheduled? If so, what are some schedule lengths? What is the minimum schedule length? Is it always less than or equal to $LCM(x, y)$? We answer all of these questions. To do so we use several functions.

That all instances with only two distinct numbers can be scheduled seems neither obvious nor particularly intuitive. There are very simple instances, both dense and nondense and with as few as three distinct numbers, that cannot be scheduled. A dense example is $\{2, 3, 6\}$ and a nondense example is $\{2, 3, 100\}$. (In fact, for all $n>0$, $\{2, 3, n\}$ cannot be scheduled.)

We do not propose the direct generation of infinite schedules. We will work with cyclic schedules, which may be repeated as needed. Thus, our proofs involve first computing a feasible cyclic schedule length, then using a pair of partitioning functions to partition that schedule length into slots assigned to items of frequency x and slots assigned to items of frequency y. (Recall that the partitioning functions map the natural numbers into the sequence of slots assigned to the corresponding frequency.) While the partitioning functions with unrestricted range may be used to create an infinite schedule, we restrict the range to generate only enough values for a cyclic schedule. Each set of slots will be scheduled by cycling through the corresponding indices. Consider the instance $A=(15, 7, 6, 3)$. The indices 1 through 7 correspond to the items of frequency 15 and the indices 8, 9, and 10 correspond to the items of frequency 6. The "15" slots will be numbered "1, 2, 3, 4, 5, 6, 7, 1, 2, 3, 4, 5, 6, 7, 1, 2, 3, ..." and the "6" slots will be numbered "8, 9, 10, 8, 9, 10, 8, 9, 10, ...".

We introduce here two length functions, $H_1(A)$ and $H_2(A)$. Each turns out to be the length of a cyclic schedule.

$$H_1(A)=\frac{y \cdot LCM(a, y-b)}{y-b}, \tag{1}$$

$$H_2(A)=\frac{x \cdot LCM(b, x-a)}{x-a}. \tag{2}$$

H_2 can be derived from H_1 by interchanging a with b and x with y. Throughout the paper we only deal formally with H_1. Symmetry implies the corresponding results for H_2. H_2 is presented because it is another easily derived cyclic schedule length, which in some cases will be smaller than H_1.

The intuitive justification of H_1 as a cyclic schedule length is the following. A segment of length y has room for b items of frequency y and therefore $y-b$ items of

frequency x. In α segments there are $\alpha \cdot (y-b)$ items of frequency x. The minimum α such that $\alpha \cdot (y-b)$ is a multiple of a is $LCM(a, y-b)/(y-b)$.

In Section 4 we prove that there exist schedules of these lengths for any instance of density $d \leqslant 1.0$. We also introduce a FOLS for schedules of these lengths.

The following theorem shows that for dense instances this scheduling method produces the same schedule length as was reported in [2].

Theorem 2.1. *For any dense instance* $A=(x, a, y, b)$, $H_1(A)=H_2(A)=LCM(x, y)$.

Proof. We prove that $H_1(A)=LCM(x, y)$. The remainder follows from symmetry.

$$H_1(A)=\frac{y \cdot LCM(a, y-b)}{y-b} \tag{3}$$

$=\{$By definition of dense $a/x+b/y=1$, so $y-b=ay/x.\}$

$$\frac{xy}{ay} \cdot LCM\left(a, \frac{ay}{x}\right) \tag{4}$$

$=\{$It was proved in [2] that $d=GCD(x, y)>1$. Let $x=x'd$ and $y=y'd.\}$

$$\frac{dx'y}{ay} \cdot LCM\left(a, \frac{ady'}{dx'}\right) \tag{5}$$

$=\{$Algebra$\}$

$$\frac{dx'}{a} \cdot LCM\left(a, \frac{a}{x'} \cdot y'\right) \tag{6}$$

$=\{$The LCM divides $y'a$, so we may replace it by $y'b/k.\}$

$$\frac{dx'}{b} \cdot \frac{y'b}{k} \tag{7}$$

$=\{$Algebra$\}$

$$\frac{dx'y'}{k}. \tag{8}$$

But the numerator is $LCM(x, y)$. For dense instances no cyclic schedule shorter than $LCM(x, y)$ is possible [2], and it will be shown in Section 4 (without reference to the present theorem) that there exists a cyclic schedule of length H_1. Therefore, $k=1$ and $H_1(A)=LCM(x, y)$. □

The importance of H_1 and H_2 is that they are relatively easy to compute, the FOLS that can be generated from them may be entirely good enough, the associated proof is relatively straightforward, and in many (but not all) instances either H_1 or H_2 is the minimum schedule length. An instance in which neither H_1 nor H_2 is the minimum

schedule length is $A=(15,7,6,3)$, where $H_1=42$ and $H_2=45$. $LCM(15,6)=30$, and the following is a cyclic schedule of length 29:

"1, 8, 9, 2, 10, 3, 8, 4, 9, 5, 10, 6, 8, 7, 9, 1, 10, 2, 8, 3, 9, 4, 10, 5, 8, 6, 9, 7, 10"

Note that the instance is nondense and the minimum schedule length is less than the *LCM*. Throughout this paper we use this example to illustrate our procedures. When necessary to make a particular point we will also use $(14,9,6,2)$. Another illustrative example is $A=(24,13,7,3)$. We invite the interested reader to follow our procedures with this example as well. Note that $H_1=91$, $H_2=72$, $LCM(24,7)=168$, and the minimum-length cyclic schedule is of length 47.

Next we derive the function that determines the minimum cyclic schedule length. We will first show why it is necessary; then in Section 5 we will prove that it is also sufficient.

Not all potential schedule lengths are plausible. In fact, a fairly simple analysis serves to eliminate many numbers as plausible schedule lengths. This analysis is based on a consideration of the minimum number of slots a given item must occupy in a schedule.

Theorem 2.2. *Given an instance $A=(x,a,y,b)$ and a potential cyclic schedule length n, the a items of frequency x require at least $a\lceil n/x\rceil$ slots and the b items of frequency y require at least $b\lceil n/y\rceil$ slots.*

Proof. If n is a multiple of x, the schedule requires at least an/x slots for the a items of frequency x. Consider one such item. Divide the potential schedule up into n/x segments of x slots each. Each such segment must contain at least one of the item under consideration. Therefore, each of the a items with the value x requires at least n/x slots, so an/x slots are required for all of them.

If n is not a multiple of x, the a items require at least $a\lceil n/x\rceil$ slots. Again, the argument is based on how many slots are required for one of the a items. Divide the n slots of the potential schedule up into $\lceil n/x\rceil$ segments, all but one of them of size x. The remaining segment is shorter. If any of the size x segments contains more than one of the item under consideration then there must be at least $\lceil n/x\rceil$ in the entire schedule. Therefore we may assume there are exactly $\lfloor n/x\rfloor$ slots in the size x segments occupied by the item under consideration. Without loss of generality we may assume that the size x segments are contiguous. If the item occurs in slot j of the first such segment, it can occur no later than slot j of the second such segment, and by induction no later than slot j in the last such segment. Now consider the shorter segment. It is preceded by $x-j$ slots in which the item is not scheduled and followed (in the next cycle) by $j-1$ such slots. But since n is not a multiple of x, the short segment must consist of at least one slot. The short segment thus falls in an interval of at least $(x-j)+(j-1)+1=x$ slots. That interval must contain a slot for the item, and

the only place for it is in the short segment. Thus $\lceil n/x \rceil$ slots are required for each item of frequency x, for a total of $a\lceil n/x \rceil$.

Since if n is a multiple of x then $n/x = \lceil n/x \rceil$, we may conclude that $a\lceil n/x \rceil$ slots are required for the a items of frequency x. A symmetrical argument may be used to show that $b\lceil n/y \rceil$ slots are required for the b items of frequency y. □

This leads to the following function:

$$M(A,n) = n - a\left\lceil \frac{n}{x} \right\rceil - b\left\lceil \frac{n}{y} \right\rceil \tag{9}$$

Corollary 2.3. *$M(A,n) \geqslant 0$ is a necessary condition for there to exist a cyclic schedule of length n.*

Proof. Follows from the theorem. □

M thus defines a series of "windows of opportunity." Wherever M is nonnegative there is the possibility of a cyclic schedule. To provide the reader with a picture of some of these "windows of opportunity" we include this plot of $M(A,n)$ for $A = (14, 9, 6, 2)$. (See Fig. 1.)

If $A = (x, a, y, b)$ is dense, $M(A,n) = 0$ only when n is a multiple of $LCM(x,y)$, and is otherwise negative. When A is nondense, however, $M(A,n)$ is positive almost everywhere; in fact, for $A = (x, a, y, b)$, $a/x + y/b < 1.0$, $\exists n_0$ such that $\forall n$, $n > n_0$, $M(A,n) > 0$. These two facts are easy to show. Proofs are left to the interested reader.

We have mentioned that A can always be scheduled, and have just pointed out that $M(A,n) \geqslant 0$ is a necessary condition for schedules of length n. Is nonnegative M also a sufficient condition? For example, if $A = \{2, 3\}$ then $M(A,6) = 1$ and there is indeed

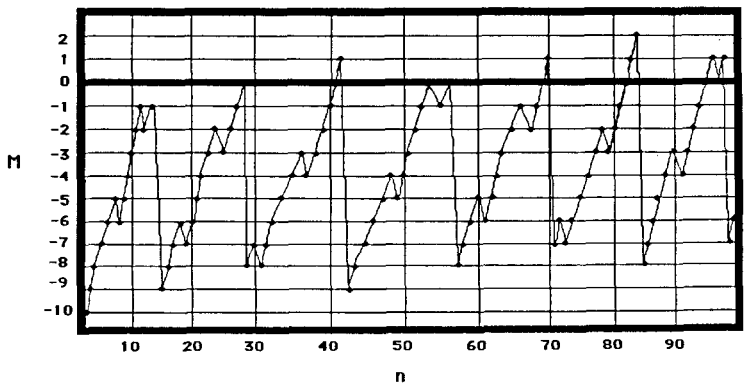

Fig. 1. $M(A,n)$ for $n = 1$ to 100. Windows of opportunity occur at the following values of n: 28, 41, 42, 54, 56, 69, 70, 82, 83, 84, 95, 96, 97, and 98.

a cyclic schedule of length 6. Thus, one might suspect that A can be scheduled in length n whenever $M(A,n)\geqslant 0$. This is not the case.

Theorem 2.4. *There exist A and n such that $M(A,n)>0$ and A has no cyclic schedule of length n.*

Proof. Let $A=(14,9,6,2)$ and $n=42$. $M(A,n)=1$ but there is no cyclic schedule of length 42.

Each of the two items of frequency 6 requires at least 7 slots. If one such item occupies exactly 7 slots in the cyclic schedule, then slots assigned to it must be exactly 6 slots apart, and never less. There are now two cases:

Case 1: Both items of frequency 6 are allocated more than 7 slots. Each item requires at least 8 slots. The 9 items of frequency 14 require a total of at least 27 slots. $27+8+8=43$, and there are only 42 slots available. Therefore, no such schedule is possible.

Case 2: At least one item of frequency 6 is allocated every 6th slot. Without loss of generality, let the evenly-scheduled item of frequency 6 occupy even-numbered slots. Only one item of frequency 14 can occupy any even-numbered slots. The reason for this involves viewing the even-numbered slots in isolation. An item of frequency 6 occupies every third even-numbered slot and an item of frequency 14 occupies every seventh such slot. But 3 and 7 are relative prime and conflict for a slot. Although such a conflict may be avoided by assigning a fourth slot to a 14, this can only be done for one of them. The others must always occupy odd-numbered slots. But this requires at least 24 odd-numbered slots, and there are only 21. Thus, no schedule of this sort is possible either. This exhausts the cases. $A=(14,9,6,2)$ cannot be scheduled in 42 slots. □

We will show, however, that there are schedules for all n such that $M(A,n)=0$. For $A=(14,9,6,2)$, $M(A,41)=0$ and the following is a cyclic schedule of length 41:

"1, 3, 4, 2, 5, 6, 1, 7, 8, 2, 9, 10, 1, 11, 3, 2, 4, 5, 1, 6, 7, 2, 8, 9, 1, 10, 11, 2, 3, 4, 1, 5, 6, 2, 7, 8, 1, 9, 10, 2, 11"

We will further show that if n is the least number such that $M(A,n)\geqslant 0$, then $M(A,n)=0$. Therefore n is the minimum cyclic schedule length. Thus, another function of interest represents the smallest n for which $M(A,n)$ is zero:

$$LM(A)=\min(n)\ni M(A,n)=0 \tag{10}$$

It was established in [2] that the minimum cyclic schedule length for a dense instance of two distinct numbers is $LCM(x,y)$. In this paper we show that $LM(A)$ is the minimum cyclic schedule length for general instances. We begin by reconciling the two results.

Theorem 2.5. *For a dense instance $A=(x,a,y,b)$, $LM(A)=LCM(x,y)$.*

Proof. The proof is in two parts. First we show that $M(A, LCM(x,y))=0$, and then we show that if n is not a common multiple of x and y then $M(A,n)<0$.

Lemma 2.6. *If an instance $A=(x,a,y,b)$ is dense, then $M(A, LCM(x,y))=0$.*

Proof. Note that for dense instances $a/x+b/y=1.0$. We begin with $LCM(x,y)$ substituted for n in Eq. (9):

$$
\begin{aligned}
M(A, LCM(x,y)) &= LCM(x,y) - a\left\lceil \frac{LCM(x,y)}{x} \right\rceil - b\left\lceil \frac{LCM(x,y)}{y} \right\rceil \\
&= \{\text{For integer } m,\ m=\lceil m \rceil\ \} \\
&\quad LCM(x,y) - a\frac{LCM(x,y)}{x} - b\frac{LCM(x,y)}{y} \\
&= \{\text{Algebra}\} \\
&\quad LCM(x,y) - LCM(x,y)\left(\frac{a}{x}+\frac{b}{y}\right) \\
&= \{\text{Definition of dense}\} \\
&\quad LCM(x,y) - LCM(x,y) = 0. \qquad \square
\end{aligned}
$$

Lemma 2.7. *If n is not a common multiple of x and y, then $M(A,n)<0$.*

Proof. $n(a/x+b/y)=a(n/x)+b(n/y)=n$. Note that $l/m=\lceil l/m \rceil$ if and only if l is a multiple of m, and that here n is not a multiple of both x and y. Thus, either $\lceil n/x \rceil > n/x$ or $\lceil n/y \rceil > n/y$ (or both), so $a\lceil n/x \rceil + b\lceil n/y \rceil > n$. Therefore, $n - a\lceil n/x \rceil - b\lceil n/y \rceil < 0$. $\square$

Proof of Theorem 2.5 (*conclusion*). The above two lemmas complete the proof. $\square$

Next we prove the interesting additional result that $LM(A) < LCM(x,y)$ for nondense instances. This tends to support the intuitive expectation that lower density should in some way be associated with shorter schedules.

Theorem 2.8. *For nondense instances $A=(x,a,y,b)$, $LM(A) < LCM(x,y)$.*

Proof. We will show that for nondense instances, $M(A, LCM(x,y))>0$; that $M(A,1)<0$; and that $\forall i,j,\ i<j$ and $M(A,i)<0$ and $M(A,j)>0$, $\exists k,\ i<k<j$, such that $M(A,k)=0$. (The last says, roughly, that any ascending sequence of M values that crosses zero is at some point equal to zero.) This requires a series of lemmas.

Lemma 2.9. *For a nondense instance A, $M(A, LCM(x,y))>0$.*

Proof. $M(A, LCM(x, y))$ reduces as in Lemma 2.6 to

$$LCM(x, y) - LCM(x, y)\left(\frac{a}{x} + \frac{b}{y}\right)$$

Here, by the definition of nondense, $a/x + b/y < 1$, so $LCM(x, y)(a/x + b/y) < LCM(x, y)$, completing the proof. □

Lemma 2.10. $M(A, 1) < 0$.

Proof. $M(A, 1) = 1 - a\lceil 1/x \rceil - b\lceil 1/y \rceil$. Each of the second and third terms is at least 1. The entire expression is negative. □

Lemma 2.11. *$\forall i, j$, $i < j$ and $M(A, i) < 0$ and $M(A, j) > 0$, $\exists k$, $i < k < j$, such that $M(A, k) = 0$.*

Proof. Note that $a\lceil n/x \rceil + b\lceil n/y \rceil \geqslant a\lceil (n-1)/x \rceil + b\lceil (n-1)/y \rceil$. Therefore, if

$$n - \left(a\left\lceil \frac{n}{x} \right\rceil + b\left\lceil \frac{n}{y} \right\rceil\right) > n - 1 - \left(a\left\lceil \frac{n-1}{x} \right\rceil + b\left\lceil \frac{n-1}{y} \right\rceil\right)$$

then the second terms are equal and the difference is exactly one. Assume for a contradiction that no suitable k exists. Then at some point between i and j there must exist an m such that $M(A, m) > 0$ and $M(A, m-1) < 0$. But M can only take on integer values, so $M(A, m) - M(A, m-1) > 1$. That is a contradiction, and k exists. □

Proof of Theorem 2.8 (*conclusion*). We have established that $M(A, LCM(x, y)) > 1$ and $M(A, 1) < 1$, so by the last lemma there exists some k, $1 < k < LCM(x, y)$, such that $M(A, k) = 0$. This completes the proof of Theorem 5. □

Corollary 2.12. *If n_0 is the least n for which $M(A, n)$ is nonnegative, then $M(A, n_0) = 0$.*

Proof. Follows from Lemma 2.11. □

Corollary 2.13. *$LM(A)$ is the least n for which $M(A, n)$ is nonnegative.*

Proof. Follows from the definition of $LM(A)$ and Corollary 2.12. □

Corollary 2.14. *No cyclic schedule of length less than $LM(A)$ exists.*

Proof. Follows from Corollaries 2.13 and 2.3. □

Theorem 2.15. *$LM(A)$ may be computed in deterministic polynomial time.*

Proof. $LM(A)$ may be cast as the minimum of four separate instances of integer linear programming in at most five variables, for which there are known polynomial-time algorithms [3]. In each instance the variable n is to be minimized, and other variables must be introduced to deal with the ceilings. The construction follows.

Case 1: n is not a multiple of either x or y. Find the minimum n satisfying

$$q_1 \cdot x + r_1 = n, \qquad q_2 \cdot y + r_2 = n,$$
$$0 < r_1 < x, \quad 0 < r_2 < y,$$
$$n - a(q_1 + 1) - b(q_2 + 1) = 0,$$

which is an integer linear programming instance in 5 variables – q_1, r_1, q_2, r_2, and n.

Case 2: n is a multiple of x but not y. Find the minimum n satisfying

$$q_1 \cdot x = n, \qquad q_2 \cdot y + r_2 = n,$$
$$0 < r_2 < y, \qquad n - a \cdot q_1 - b(q_2 + 1) = 0,$$

which is an ILP instance in 4 variables – q_1, q_2, r_2, and n.

Case 3: n is a multiple of y but not of x.

$$q_1 x + r_1 = n, \qquad q_2 y = n,$$
$$0 < r_1 < x, \qquad n - a(q_1 + 1) - b \cdot q_2 = 0,$$

which is again an ILP instance in 4 variables – q_1, r_1, q_2, and n.

Case 4: n is a multiple of both y and x.

$$q_1 x = n, \qquad q_2 y = n,$$
$$n - a \cdot q_1 - b \cdot q_2 = 0,$$

which is an ILP instance in 3 variables – q_1, q_2 and n.

$LM(A)$ is the minimum of the solutions to these four cases. □

The fact that dense instances have shortest cyclic schedule length LCM, and that nondense instances have shortest cyclic schedule length less than LCM, is satisfying and by no means obvious.

In this section we have introduced several functions concerned with cyclic schedule lengths. H_1 and H_2 will be shown to be lengths of cyclic schedules, and they are easy to compute. M serves to eliminate those potential schedule lengths that cannot have cyclic schedules due to insufficient space. We will show in Section 5 that cyclic schedules exist for all A and n such that $M(A, n) = 0$. The function $LM(A)$ yields the least n such that $M(A, n) \geq 0$ – and in fact at such an n $M(A, n) = 0$. We will show in Section 5 that this is in fact a cyclic schedule length, and thus the minimum cyclic schedule length.

H_1 and H_2 can be computed in linear time using standard algorithms. That computing LM seems to involve solving integer linear programming in five variables

will become important in Sections 4 and 5 when we illustrate how to produce a FOLS that will generate a schedule with cycles of length H_1, H_2, or LM.

3. Partitioning without collisions

In this section we will prove an important lemma about the partitioning functions that we will use for scheduling. Recall that partitioning functions are used to assign slots to items of a particular frequency. There are seven conditions that must hold for the partitioning functions and associated cyclic schedule length to constitute a valid scheduling strategy, and six will require separate proofs for the two scheduling methods. One does not. It proves that pairs of partitioning functions of a particular form have disjoint ranges, and that therefore the two functions never select the same slot. (Note that the example partitioning functions of Section 1, $2i$ and $2i+1$, have this property.) In the statement of the lemma, $i+\lceil iz \rceil$ on the left-hand side represents one of the partitioning functions and $j+\lfloor j/z \rfloor+1$ represents the other function. By proving that they can never be equal, whatever values are chosen for i and j, we show that the functions indeed define disjoint sets.

Lemma 3.1. $\forall$ *real* z *and natural* i,j, $i+\lceil iz \rceil \neq j+\lfloor j/z \rfloor+1$.

Proof. Assume, for a contradiction, that $\exists i,j,z$ such that

$$i+\lceil iz \rceil = j+\lfloor j/z \rfloor+1. \tag{11}$$

Let $k=\lfloor j/z \rfloor$. Substituting into Eq. (11) yields

$$i+\lceil iz \rceil = j+k+1. \tag{12}$$

Because $\lfloor m \rfloor \leqslant m \leqslant \lfloor m \rfloor+1$, we have $k \leqslant j/z < k+1$, and therefore

$$j<(k+1)z \tag{13}$$

and

$$kz \leqslant j. \tag{14}$$

We now consider two cases: $i \geqslant k+1$ and $i<k+1$.

Case 1: $i \geqslant k+1$. We will start with the identity

$$i+\lceil iz \rceil = i+\lceil iz \rceil. \tag{15}$$

{Substituting $k+1$ for i on the right side}

$$i+\lceil iz \rceil \geqslant k+1+\lceil (k+1)z \rceil. \tag{16}$$

{From (12) and (16)}

$$j+k+1 \geqslant k+1+\lceil (k+1)z \rceil. \tag{17}$$

{Cancelling common terms}

$$j \geqslant \lceil (k+1)z \rceil. \tag{18}$$

But $\lceil (k+1)z \rceil \geqslant (k+1)z$ and by (13) and transitivity of $>$ and $\geqslant$ we get $j > j$, a contradiction.

Case 2: $i < k+1$. This is equivalent to $i \leqslant k$. Multiplying by z and combining with (14) we get

$$iz \leqslant kz \leqslant j; \tag{19}$$

{$r \leqslant s \Rightarrow \lceil r \rceil \leqslant \lceil s \rceil$ and if t is an integer, then $s \leqslant t \Rightarrow \lceil s \rceil \leqslant t$}

$$\lceil iz \rceil \leqslant \lceil kz \rceil \leqslant j. \tag{20}$$

{Adding i to each comparand}

$$i + \lceil iz \rceil \leqslant i + \lceil kz \rceil \leqslant i + j. \tag{21}$$

{Transitivity of $\leqslant$}

$$i + \lceil iz \rceil \leqslant i + j. \tag{22}$$

{From (22) and (12)}

$$j + k + 1 \leqslant i + j. \tag{23}$$

{Cancelling common terms}

$$k + 1 \leqslant i, \tag{24}$$

which contradicts the case assumption.

This exhausts the cases, resulting in a contradiction in each. Therefore no such x, i, and j exist. □

This important lemma will be used in each of the following two sections.

4. Two solutions of nonoptimal length

Although we have mentioned that neither H_1 nor H_2 is in all cases the optimal cyclic schedule length, we feel that the scheduling method for schedules of length H_1 (and H_2) is nevertheless important. Such schedules are relatively easy to compute, and the method is relatively easy to prove correct. Furthermore, one can design a FOLS based on this scheduling method, and the fact that the corresponding cyclic schedule is not of minimum length may not be of great importance. Finally, we hope that this result will help to characterize the circumstances under which the strategy of even distribution leads to schedules. This is important in light of the fact that the scheduling strategy in the next section is a more sophisticated implementation of the same basic idea.

Our method involves creating a scheduling algorithm based on allocating slots to x and y as evenly as possible. As motivation, however, we first consider another potential strategy.

Since it is known that all dense instances with only two distinct numbers can be scheduled [2], an obvious approach is to define a set of operations that can transform any arbitrary instance into a dense instance. While it is indeed possible to define operations that transform a sufficiently nondense instance $A=(x, a, y, b)$ into a denser instance A' such that A is schedulable if A' is, we have discovered no such set of transformations that is guaranteed to result in a dense instance. Perhaps some additional transformations would make this approach viable, but while such a set of transformations might lead to a proof that all of these instances can be scheduled, it would probably result in a cyclic schedule of more than the minimum length. The reason for this is that denser instances tend to have longer minimum schedule lengths. We therefore abandoned this line of investigation.

The remainder of this section concerns the scheduling method based on H_1 (and by symmetry, H_2) and two partitioning functions. Given a schedule length and a way of partitioning slots between items of frequencies x and y, a scheduling algorithm is easy. The items of frequency x have indices $1, 2, \ldots, a$ and those of frequency y have indices $a+1, a+2, \ldots, a+b$. They are simply scheduled in order into the selected slots, with the sequence of indices repeated as necessary. The same method will be used in Section 5, but with a different function providing the length of the cyclic schedule and of course with different partitioning functions.

We now introduce our first pair of partitioning functions, *Place*1 and *Place*2. *Place*1 identifies slots for items of frequency x and *Place*2 identifies slots for items of frequency y. n is the schedule length – here $H_1(A)$.

$$Place1(i)=i+\left\lceil \frac{ib}{y-b} \right\rceil, \quad 0 \leqslant i < \frac{n}{y}(y-b), \tag{25}$$

$$Place2(i)=\left\lfloor \frac{iy}{b} \right\rfloor + 1, \quad 0 \leqslant i < \frac{n}{y}b. \tag{26}$$

Note that $Place2(i)$ is an algebraic simplification of $i+\lfloor i(y-b)/b \rfloor + 1$, and that if we replace $b/(y-b)$ by z, we get $Place1(i)=i+\lceil iz \rceil$ and $Place2(i)=i+\lfloor i/z \rfloor + 1$. Thus Lemma 3.1 can be used to show that *Place*1 and *Place*2 never select the same slot. Note also the range restrictions. As noted in Section 2, partitioning functions from the naturals to the naturals may be used to create infinite schedules. We have chosen to restrict the functions so as to schedule only the first n slots, which will be shown to constitute a cyclic schedule. These n slots may then be repeated as many times as necessary.

This scheduling method amounts to distributing slots for items of frequency x as evenly as possible over the first y slots. This creates a "skeleton" of length y, which will be repeated as many times as necessary to reach length $H_1(A)$. (Recall that $H_1(A)$ is

Table 1

i	*Place*1(i)	Contents	i	*Place*2(i)	Contents
0	0	1	0	1	8
1	2	2	1	3	9
2	4	3	2	5	10
3	6	4	3	7	8
4	8	5	4	9	9
5	10	6	5	11	10
6	12	7	6	13	8
7	14	1	7	15	9
8	16	2	8	17	10
9	18	3	9	19	8
10	20	4	10	21	9
11	22	5	11	23	10
12	24	6	12	25	8
13	26	7	13	27	9
14	28	1	14	29	10
15	30	2	15	31	8
16	32	3	16	33	9
17	34	4	17	35	10
18	36	5	18	37	8
19	38	6	19	39	9
20	40	7	20	41	10

a multiple of y.) *Place*1 selects slots by skipping slots in the ratio of $b/(y-b)$, which is the ratio of slots not needed for x to slots needed for x. The ceiling term represents slots skipped – that is, when incrementing i causes the ceiling term to increase, a slot is skipped. Finally, the limiting formula $(n/y)(y-b)$ is derived from the fraction of slots used for x $(y/(y-b))$.

We now give an example of this scheduling method. The instance $A=(15, 7, 6, 3)$ yields $H_1(A)=42$ and for *Place*1 and *Place*2 yields the values and contents as shown in Table 1.

Interleaving the "Contents" lists as specified by *Place*1 and *Place*2 yields the schedule

"1, 8, 2, 9, 3, 10, 4, 8, 5, 9, 6, 10, 7, 8, 1, 9, 2, 10, 3, 8, 4, 9, 5, 10, 6, 8, 7, 9, 1, 10, 2, 8, 3, 9, 4, 10, 5, 8, 6, 9, 7, 10"

Establishing the correctness of the length and partitioning functions requires the proof of seven lemmas, which we will provide for each scheduling method.

Theorem 4.1. *The method given above results in a cyclic schedule for an instance* $A=(x, a, y, b)$.

Proof. In the following, n is used for the schedule length $(H_1(A))$ for convenience and

for consistency with Section 5. This method is correct if and only if all of the following hold:

1. $\forall i,j$, $Place1(i) \neq Place2(j)$; that is, $Place1$ and $Place2$ never select the same slot.
2. $Place1(a\lceil n/x \rceil - 1) < n$; that is, there are enough slots for the items of frequency x.
3. $Place2(b\lceil n/y \rceil - 1) < n$; that is, there are enough slots for the items of frequency y.
4. $Place1(i+a) - Place1(i) \leqslant x$; the correctness condition for items of frequency x.
5. $Place2(i+b) - Place2(i) \leqslant y$; the correctness condition for items of frequency y.
6. $\exists k$ such that $Place1(i+ak) - Place1(i) = n$; that is, the schedule cycles correctly for items of frequency x.
7. $\exists k$ such that $Place2(i+bk) - Place2(i) = n$; that is, the schedule cycles correctly for items of frequency y.

The first condition requires no collisions, as explained in Section 3. That is, $Place1$ and $Place2$ actually partition slots 0 through $n-1$ into disjoint subsets. The next two conditions require that $Place1$ ($Place2$) selects enough slots for the $a\lceil n/x \rceil$ ($b\lceil n/y \rceil$) items of frequency x (y) that must be scheduled in the n slots of a cyclic schedule. Conditions 4 and 5 require that the next slot available for a particular item of frequency x (y) occur no more than x (y) slots from the last one. The last two conditions require that the nth slot beyond a scheduling of a particular item also contains that item. This establishes that such an infinite schedule actually consists of repetitions of a cyclic schedule of length n.

Lemma 4.2. $\forall i,j$, $Place1(i) \neq Place2(j)$.

Proof. This follows from Lemma 3.1 using $z = b/(y-b)$. □

Lemma 4.3. $Place1(a\lceil n/x \rceil - 1) < n$.

Proof.

$$Place1\left(a\left\lceil\frac{n}{x}\right\rceil - 1\right) = a\left\lceil\frac{n}{x}\right\rceil - 1 + \left\lceil\frac{b(a\lceil\frac{n}{x}\rceil - 1)}{y-b}\right\rceil \tag{27}$$

$$\leqslant \{\text{Because } a/x + b/y \leqslant 1 \text{ implies } x \geqslant ay/(y-b)\}$$

$$a\left\lceil\frac{n}{x}\right\rceil - 1 + \left\lceil\frac{b\left(a\left\lceil\frac{n(y-b)}{ay}\right\rceil - 1\right)}{y-b}\right\rceil \tag{28}$$

$$= \{\text{Algebra}\}$$

$$a\left\lceil \frac{n}{x} \right\rceil - 1 + \left\lceil \frac{ab\left\lceil \frac{n(y-b)}{ay} \right\rceil}{y-b} - \frac{b}{y-b} \right\rceil \tag{29}$$

$$= \{\text{Because the expression in the inner ceiling is integral.}\}$$

$$a\left\lceil \frac{n}{x} \right\rceil - 1 + \left\lceil \frac{bn}{y} - \frac{b}{y-b} \right\rceil \tag{30}$$

$$= \{\text{Because } bn/y \text{ is integral}\}$$

$$a\left\lceil \frac{n}{x} \right\rceil - 1 + \frac{bn}{y} + \left\lceil -\frac{b}{y-b} \right\rceil \tag{31}$$

$$\leqslant \{\text{Because } n/y \text{ is integral, and } a\lceil n/x \rceil + b\lceil n/y \rceil \leqslant n\}$$

$$n - 1 + \left\lceil -\frac{b}{y-b} \right\rceil, \tag{32}$$

which is less than n. □

Lemma 4.4. $Place2(b\lceil n/y \rceil - 1) < n$.

Proof.

$$Place2\left(b\left\lceil \frac{n}{y} \right\rceil - 1\right) = \left\lfloor \frac{(b\lceil \frac{n}{y} \rceil - 1)\, y}{b} \right\rfloor + 1 \tag{33}$$

$$= \{\text{Algebra}\}$$

$$\left\lfloor y\left\lceil \frac{n}{y} \right\rceil - \frac{y}{b} \right\rfloor + 1 \tag{34}$$

$$= \{n = H_1(A), \text{ which is a multiple of } y.\}$$

$$\left\lfloor n - \frac{y}{b} \right\rfloor + 1 \tag{35}$$

$$\leqslant \{\lfloor m \rfloor \leqslant m\}$$

$$n - \frac{y}{b} + 1. \tag{36}$$

But because $y/b > 1$, this is less than n. □

Lemma 4.5. *Place*1$(i+a) - $*Place*1$(i) \leqslant x$.

Proof.

$$Place1(i+a) - Place1(i) = i + a + \left\lceil \frac{(i+a)b}{y-b} \right\rceil - i - \left\lceil \frac{ib}{y-b} \right\rceil \tag{37}$$

$$= \{\text{Algebra}\}$$

$$a + \left\lceil \frac{ib}{y-b} + \frac{ab}{y-b} \right\rceil - \left\lceil \frac{ib}{y-b} \right\rceil \tag{38}$$

$$\leqslant \{\lceil r \rceil - \lceil s \rceil \leqslant \lceil r-s \rceil\}$$

$$a + \left\lceil \frac{ib}{y-b} + \frac{ab}{y-b} - \frac{ib}{y-b} \right\rceil \tag{39}$$

$$= \{\text{Algebra}\}$$

$$a + \left\lceil \frac{ab}{y-b} \right\rceil = \left\lceil a + \frac{ab}{y-b} \right\rceil = \left\lceil \frac{ay}{y-b} \right\rceil \tag{40}$$

$$\leqslant \{\text{Because } a/x + b/y \leqslant 1 \text{ implies } x \geqslant ay/(y-b)\}$$

$$x. \quad \square \tag{41}$$

Lemma 4.6. *Place*2$(i+b) - $*Place*2$(i) \leqslant y$.

Proof.

$$Place2(i+b) - Place2(i)$$

$$= \left\lfloor \frac{(i+b)y}{b} \right\rfloor + 1 - \left\lfloor \frac{iy}{b} \right\rfloor - 1 \tag{42}$$

$$= \{\text{Algebra}\}$$

$$\left\lfloor \frac{iy}{b} + y \right\rfloor - \left\lfloor \frac{iy}{b} \right\rfloor = \left\lfloor \frac{iy}{b} \right\rfloor + y - \left\lfloor \frac{iy}{b} \right\rfloor = y. \quad \square \tag{43}$$

Lemma 4.7. $\exists k$ *such that* *Place*1$(i+ak) - $*Place*1$(i) = n$.

Proof. We will use $LCM(a, y-b)$ (a multiple of a) for ak.

$$Place1(i+LCM(a,y-b))-Place1(i)$$

$$=i+LCM(a,y-b)+\left\lceil\frac{(i+LCM(a,y-b))b}{y-b}\right\rceil-i-\left\lceil\frac{ib}{y-b}\right\rceil \tag{44}$$

$$=\{\text{Algebra}\}$$

$$LCM(a,y-b)+\left\lceil\frac{ib}{y-b}+\frac{b\cdot LCM(a,y-b)}{y-b}\right\rceil-\left\lceil\frac{ib}{y-b}\right\rceil \tag{45}$$

$$=\{\text{Because } LCM(a,y-b)/(y-b) \text{ is integral}\}$$

$$LCM(a,y-b)+\frac{b\cdot LCM(a,y-b)}{y-b} \tag{46}$$

$$=\{\text{Algebra}\}$$

$$\frac{(y-b)\cdot LCM(a,y-b)+b\cdot LCM(a,y-b)}{y-b}$$

$$=\frac{y\cdot LCM(a,y-b)}{y-b}=n. \quad \square \tag{47}$$

Lemma 4.8. *$\exists k$ such that $Place2(i+bk)-Place2(i)=n$.*

Proof. We will use $\lceil n/y\rceil$ for k, so bk becomes $b\lceil n/y\rceil$. Note that because n/y is integral, we may replace $b\lceil n/y\rceil$ with nb/y.

$$Place2\left(i+\frac{nb}{y}\right)-Place2(i)=\left\lfloor\frac{\left(i+\frac{bn}{y}\right)y}{b}\right\rfloor+1-\left\lfloor\frac{iy}{b}\right\rfloor-1 \tag{48}$$

$$=\{\text{Algebra}\}$$

$$\left\lfloor\frac{iy}{b}+n\right\rfloor-\left\lfloor\frac{iy}{b}\right\rfloor=n. \quad \square \tag{49}$$

Proof of Theorem 4.1 (*conclusion*). Lemmas 4.2–4.8 complete the proof of Theorem 4.1. □

Corollary 4.9. *All instances with only two distinct numbers and density at most* 1.0 *have schedules. Thus, the pinwheel decision problem is trivial for this class of instances.*

Proof. The above algorithm takes an arbitrary instance and generates a schedule. □

We will now show how to produce fast online schedules for pinwheel instances with two distinct numbers. Our algorithm requires as input only a cyclic schedule length n,

the numbers a and b from the instance description, and the partitioning functions *Place*1 and *Place*2. The latter are assumed to perform their computations in constant time.

In the terminology used in Section 1, the part of the program before the **do** is α and the part between **do** and **forever** is β. The schedule length n and the partitioning function *Place*1 can be generated in deterministic polynomial time. α runs in constant time, as does β.

Note that the following FOLS can be used with either the scheduling method of this section or that of Section 5. In fact, it works with any scheduling method that provides a cyclic schedule length and partitioning functions, providing the partitioning functions run in constant time and have the above seven properties.

```
P1 := 0;
P2 := 0;
Slot := 0;
Ia := 1;
Ib := a + 1;
do
  if Place1(P1) = Slot then begin
    Output(Ia);
    Ia := Ia + 1; if Ia > a then Ia := 1;
    P1 := P1 + 1;
    if Place1(P1) ≥ n then
      P1 := 0;
  end
  if Place2(P2) = Slot then begin
    Output(Ib);
    Ib := Ib + 1; if Ib > a + b then Ib := a + 1
    P2 := P2 + 1;
    if Place2(P2) ≥ n then
      P2 := 0;
  end;
  Slot := (Slot + 1) mod n
forever
```

Note that the above program meets all the requirements to be a FOLS, and that it can be generated in deterministic polynomial time. Thus we have the following theorem.

Theorem 4.10. *The pinwheel scheduling problem restricted to instances with only two distinct numbers is in S-P-C.*

Proof. Follows from Theorem 4.1 and the above program. □

The reader can now see that the FOLS generating schedules of length H_1 (H_2) is easy to obtain since H_1 (H_2) can be computed in linear time. The FOLS generating a schedule of length LM is harder to obtain since computing LM seems to require solving instances of integer linear programming in five variables (Theorem 2.8). The resulting FOLS, however, is potentially faster since the modular arithmetic involves working with shorter bit strings.

In this section we have introduced the idea of scheduling by means of partitioning functions. Different partitioning functions will be used in the following section. We have proved that for every instance A cyclic schedules exist of lengths $H_1(A)$ and $H_2(A)$. The proof will serve as a pattern for the similar proof in the next section. The proof is constructive; thus, we have shown how such schedules may be constructed. Finally, we have demonstrated a method for using the length computed by $H_1(A)$ ($H_2(A)$) and the function *Place*1 for constructing a FOLS.

5. A solution of optimal length

The method given in Section 4 creates a schedule for any instance with only two distinct numbers and density at most 1.0, but the schedule created is not always the shortest one. Sometimes the minimum schedule length is not a multiple of either x or y, and thus cannot be either H_1 or H_2. Recall that for the instance $A=(15, 7, 6, 3)$ there is a schedule of length 29 but $H_1(A)$ and $H_2(A)$ are 42 and 45 respectively. In this section we show how to use Eq. (10) to create a schedule of optimal length. In Section 6 we will show how to construct a FOLS based on this scheduling strategy, as well as the scheduling strategy of the previous section.

We will prove that there is a cyclic schedule of every length n for which $M(A, n)=0$. This will establish in particular that there is a cyclic schedule of length $LM(A)$, and that $LM(A)$ is therefore a lower bound as well as an upper bound on least cyclic schedule length. Our method, as before, is to define an algorithm in terms of partitioning functions *Place*1 and *Place*2, and then use the functions to prove the correctness of the algorithm. Although more complex, the proof parallels that of Theorem 4.1 in the previous section.

The new *Place*1 and *Place*2 closely resemble their counterparts in Section 4. There are many possible functions that capture the informal notion of distributing things as evenly as possible over a sequence of slots. Here, the number of slots needed in a cycle of length n for items of frequency x is $a\lceil n/x\rceil$ and the number needed for y (and therefore not needed for x) is $b\lceil n/y\rceil$. As before, their ratio determines when a slot should be skipped. We have again selected two functions that partition the sequence of slots into two disjoint sets. They are:

$$Place1(i)=i+\left\lceil i\,\frac{b\lceil \frac{n}{y}\rceil}{a\lceil \frac{n}{x}\rceil}\right\rceil, \quad 0\leqslant i<a\left\lceil\frac{n}{x}\right\rceil \tag{50}$$

and

$$Place2(i)=i+\left\lfloor i\,\frac{a\lceil \frac{n}{x}\rceil}{b\lceil \frac{n}{y}\rceil}\right\rfloor+1,\quad 0\leqslant i<b\left\lceil\frac{n}{y}\right\rceil. \tag{51}$$

Note that with the substitution

$$z=\frac{b\lceil \frac{n}{y}\rceil}{a\lceil \frac{n}{x}\rceil},$$

these functions too are of the form $Place1(i)=i+\lfloor iz\rfloor$ and $Place2(i)=i+\lfloor i/z\rfloor+1$. This allows the use of Lemma 3.1 to show that the sets of slots selected by *Place*1 and *Place*2 are indeed disjoint.

The cyclic schedule is produced by assigning items of frequency x to the slots selected by *Place*1 and items of frequency y to the slots selected by *Place*2. The slots are assigned to items in cyclic order.

For example, the instance $A=(15,7,6,3)$ yields $LM(A)=29$ and for *Place*1 and *Place*2 yields the values and contents as shown in Table 2. Interleaving the "Contents" lists as specified by *Place*1 and *Place*2 yields the schedule given in Section 4, namely

"1, 8, 9, 2, 10, 3, 8, 4, 9, 5, 10, 6, 8, 7, 9, 1, 10, 2, 8, 3, 9, 4, 10, 5, 8, 6, 9, 7, 10"

Theorem 5.1. *The method given above results in a cyclic schedule for an instance* $A=(x,a,y,b)$.

Proof. This method is correct if and only if all of the following hold:

1. $\forall i,j$, $Place1(i)\neq Place2(j)$; that is, *Place*1 and *Place*2 never select the same slot.
2. $Place1(a\lceil n/x\rceil-1)<n$; that is, there are enough slots for the items of frequency x.

Table 2

i	$Place1(i)$	Contents	i	$Place2(i)$	Contents
0	0	1	0	1	8
1	3	2	1	2	9
2	5	3	2	4	10
3	7	4	3	6	8
4	9	5	4	8	9
5	11	6	5	10	10
6	13	7	6	12	8
7	15	1	7	14	9
8	17	2	8	16	10
9	19	3	9	18	8
10	21	4	10	20	9
11	23	5	11	22	10
12	25	6	12	24	8
13	27	7	13	26	9
			14	28	10

3. $Place2(b\lceil n/y\rceil-1)<n$; that is, there are enough slots for the items of frequency y.

4. $Place1(i+a)-Place1(i)\leqslant x$; the correctness condition for items of frequency x.

5. $Place2(i+b)-Place2(i)\leqslant y$; the correctness condition for items of frequency y.

6. $\exists k$ such that $Place1(i+ak)-Place1(i)=n$; that is, the schedule cycles correctly for items of frequency x.

7. $\exists k$ such that $Place2(i+bk)-Place2(i)=n$; that is, the schedule cycles correctly for items of frequency y.

Note that these seven conditions are exactly the same as the seven conditions in Section 4. The proofs differ somewhat because of the different value of n and the different definitions of *Place*1 and *Place*2.

Lemma 5.2. $\forall i, j$, $Place1(i)\neq Place2(j)$.

Proof. This follows from Lemma 3.1 using $z=b\lceil\frac{n}{y}\rceil/a\lceil\frac{n}{x}\rceil$. □

Lemma 5.3. $Place1(a\lceil n/x\rceil-1)<n$.

Proof.

$$Place1\left(a\left\lceil\frac{n}{x}\right\rceil-1\right)=a\left\lceil\frac{n}{x}\right\rceil-1+\left\lceil\left(a\left\lceil\frac{n}{x}\right\rceil-1\right)\frac{b\lceil\frac{n}{y}\rceil}{a\lceil\frac{n}{x}\rceil}\right\rceil$$

$$=\{\text{Algebra}\}$$

$$a\left\lceil\frac{n}{x}\right\rceil+\left\lceil b\left\lceil\frac{n}{y}\right\rceil-\frac{b\lceil\frac{n}{y}\rceil}{a\lceil\frac{n}{x}\rceil}\right\rceil-1$$

$$\leqslant\{\lceil m\rceil-1\leqslant m\}$$

$$a\left\lceil\frac{n}{x}\right\rceil+b\left\lceil\frac{n}{y}\right\rceil-\frac{b\lceil\frac{n}{y}\rceil}{a\lceil\frac{n}{x}\rceil}$$

$$=\{\text{When } M(A,n)=0\text{, the sum of the first 2 terms is } n\}$$

$$n-\frac{b\lceil\frac{n}{y}\rceil}{a\lceil\frac{n}{x}\rceil},$$

which is less than n. □

Lemma 5.4. $Place2(b\lceil n/y\rceil-1)<n$.

Proof.

$$Place2\left(b\left\lceil\frac{n}{y}\right\rceil-1\right)=b\left\lceil\frac{n}{y}\right\rceil-1+\left\lfloor\left(b\left\lceil\frac{n}{x}\right\rceil-1\right)\frac{a\left\lceil\frac{n}{x}\right\rceil}{b\left\lceil\frac{n}{y}\right\rceil}\right\rfloor+1$$

$$=\{\text{Algebra}\}$$

$$b\left\lceil\frac{n}{y}\right\rceil+\left\lfloor a\left\lceil\frac{n}{x}\right\rceil-\frac{a\left\lceil\frac{n}{x}\right\rceil}{b\left\lceil\frac{n}{y}\right\rceil}\right\rfloor$$

$$\leqslant\{\lfloor m\rfloor\leqslant m\}$$

$$b\left\lceil\frac{n}{y}\right\rceil+a\left\lceil\frac{n}{x}\right\rceil-\frac{a\left\lceil\frac{n}{x}\right\rceil}{b\left\lceil\frac{n}{y}\right\rceil}$$

$$=\{\text{When } M(A,n)=0\text{, the sum of the first 2 terms is } n\}$$

$$n-\frac{a\left\lceil\frac{n}{x}\right\rceil}{b\left\lceil\frac{n}{y}\right\rceil},$$

which is less than n. □

Lemma 5.5. $Place1(i+a)-Place1(i)\leqslant x$.

Proof.

$$Place1(i+a)-Place1(i)=i+a+\left\lceil(i+a)\frac{b\left\lceil\frac{n}{y}\right\rceil}{a\left\lceil\frac{n}{x}\right\rceil}\right\rceil-i-\left\lceil i\frac{b\left\lceil\frac{n}{y}\right\rceil}{a\left\lceil\frac{n}{x}\right\rceil}\right\rceil$$

$$=\{\text{Cancelling } i\text{, multiplying through}\}$$

$$a+\left\lceil i\frac{b\left\lceil\frac{n}{y}\right\rceil}{a\left\lceil\frac{n}{x}\right\rceil}+\frac{b\left\lceil\frac{n}{y}\right\rceil}{\left\lceil\frac{n}{x}\right\rceil}\right\rceil-\left\lceil i\frac{b\left\lceil\frac{n}{y}\right\rceil}{a\left\lceil\frac{n}{x}\right\rceil}\right\rceil$$

$$\leqslant\{\lceil r\rceil-\lceil s\rceil\leqslant\lceil r-s\rceil\text{, Cancelling terms}\}$$

$$a+\left\lceil\frac{b\left\lceil\frac{n}{y}\right\rceil}{\left\lceil\frac{n}{x}\right\rceil}\right\rceil$$

$$=\{\text{When } M(A,n)=0,\ b\lceil n/y\rceil=n-a\lceil n/x\rceil\ \}$$

$$a+\left\lceil\frac{n-a\left\lceil\frac{n}{x}\right\rceil}{\left\lceil\frac{n}{x}\right\rceil}\right\rceil$$

$$= \{\text{Algebra}\}$$

$$a + \left\lceil \frac{n}{\lceil \frac{n}{x} \rceil} - a \right\rceil$$

$$= \{\text{For integer } s, \lceil r - s \rceil = \lceil r \rceil - s\}$$

$$\left\lceil \frac{n}{\lceil \frac{n}{x} \rceil} \right\rceil \leqslant \left\lceil \frac{n}{\frac{n}{x}} \right\rceil = x. \qquad \square$$

Lemma 5.6. *Place2*$(i+b) - $*Place2*$(i) \leqslant y$.

Proof.

$$Place2(i+b) - Place2(b) = i + b + \left\lfloor (i+b) \frac{a\lceil \frac{n}{x} \rceil}{b\lceil \frac{n}{y} \rceil} \right\rfloor + 1 - i - \left\lfloor i \frac{a\lceil \frac{n}{x} \rceil}{b\lceil \frac{n}{y} \rceil} \right\rfloor - 1$$

$$= \{\text{When } M(A, n) = 0,\ a\lceil n/x \rceil = n - b\lceil n/y \rceil .\}$$

$$b + \left\lfloor i \frac{a\lceil \frac{n}{x} \rceil}{b\lceil \frac{n}{y} \rceil} + \frac{n - b\lceil \frac{n}{y} \rceil}{\lceil \frac{n}{y} \rceil} \right\rfloor - \left\lfloor i \frac{a\lceil \frac{n}{x} \rceil}{b\lceil \frac{n}{y} \rceil} \right\rfloor$$

$$= \{\text{Removing integer } b, \text{ cancelling.}\}$$

$$\left\lfloor i \frac{a\lceil \frac{n}{x} \rceil}{b\lceil \frac{n}{y} \rceil} + \frac{n}{\lceil \frac{n}{y} \rceil} \right\rfloor - \left\lfloor i \frac{a\lceil \frac{n}{x} \rceil}{b\lceil \frac{n}{y} \rceil} \right\rfloor .$$

$$\leqslant \{r/\lceil s \rceil \leqslant r/s\}$$

$$\left\lfloor i \frac{a\lceil \frac{n}{x} \rceil}{b\lceil \frac{n}{y} \rceil} + y \right\rfloor - \left\lfloor i \frac{a\lceil \frac{n}{x} \rceil}{b\lceil \frac{n}{y} \rceil} \right\rfloor$$

$$= \{\text{Removing integer } y.\}$$

$$y + \left\lfloor i \frac{a\lceil \frac{n}{x} \rceil}{b\lceil \frac{n}{y} \rceil} \right\rfloor - \left\lfloor i \frac{a\lceil \frac{n}{x} \rceil}{b\lceil \frac{n}{y} \rceil} \right\rfloor = y \qquad \square$$

Lemma 5.7. $\exists k$ *such that* *Place1*$(i + ak) - $*Place1*$(i) = n$.

Proof. We will use $\lceil \frac{n}{x} \rceil$ for k.

$$Place1\left(i + a\left\lceil \frac{n}{x} \right\rceil\right) - Place1(i) = i + a\left\lceil \frac{n}{x} \right\rceil + \left\lceil \left(i + a\lceil \tfrac{n}{x} \rceil\right) \frac{b\lceil \frac{n}{y} \rceil}{a\lceil \frac{n}{x} \rceil} \right\rceil$$

$$- i - \left\lceil i \frac{b\lceil \frac{n}{y} \rceil}{a\lceil \frac{n}{x} \rceil} \right\rceil$$

$$= \{\text{Multiplying through}\}$$

$$a\left\lceil \frac{n}{x} \right\rceil + \left\lceil i \frac{b\lceil \frac{n}{y} \rceil}{a\lceil \frac{n}{x} \rceil} + b\left\lceil \frac{n}{y} \right\rceil \right\rceil - \left\lceil i \frac{b\lceil \frac{n}{y} \rceil}{a\lceil \frac{n}{x} \rceil} \right\rceil$$

$$= \{b\lceil n/y \rceil \text{ is integral.}\}$$

$$a\left\lceil \frac{n}{x} \right\rceil + b\left\lceil \frac{n}{y} \right\rceil + \left\lceil i \frac{b\lceil \frac{n}{y} \rceil}{a\lceil \frac{n}{x} \rceil} \right\rceil - \left\lceil i \frac{b\lceil \frac{n}{y} \rceil}{a\lceil \frac{n}{x} \rceil} \right\rceil$$

$$= \left\{ M(A,n) = 0 \Rightarrow n = a\left\lceil \frac{n}{x} \right\rceil + b\left\lceil \frac{n}{y} \right\rceil . \right\}$$

$$n. \qquad \square$$

Lemma 5.8. *$\exists k$ such that $Place2(i+bk) - Place2(i) = n$.*

Proof. We use $\lceil \frac{n}{y} \rceil$ for k.

$$Place2\left(i + b\left\lceil \frac{n}{y} \right\rceil\right) - Place2(i) = i + b\left\lceil \frac{n}{y} \right\rceil + \left\lfloor \left(i + b\left\lceil \frac{n}{y} \right\rceil\right) \frac{a\lceil \frac{n}{x} \rceil}{b\lceil \frac{n}{y} \rceil} \right\rfloor$$

$$+ 1 - i - \left\lfloor i \frac{a\lceil \frac{n}{x} \rceil}{b\lceil \frac{n}{y} \rceil} \right\rfloor - 1$$

$$= \{\text{Multiplying through}\}$$

$$b\left\lceil \frac{n}{y} \right\rceil + \left\lfloor i \frac{a\lceil \frac{n}{x} \rceil}{b\lceil \frac{n}{y} \rceil} + a\left\lceil \frac{n}{x} \right\rceil \right\rfloor - \left\lfloor i \frac{a\lceil \frac{n}{x} \rceil}{b\lceil \frac{n}{y} \rceil} \right\rfloor$$

$$= \{a\lceil n/x \rceil \text{ is integral.}\}$$

$$b\left\lceil \frac{n}{y} \right\rceil + a\left\lceil \frac{n}{x} \right\rceil + \left\lfloor i \frac{a\lceil \frac{n}{x} \rceil}{b\lceil \frac{n}{y} \rceil} \right\rfloor - \left\lfloor i \frac{a\lceil \frac{n}{x} \rceil}{b\lceil \frac{n}{y} \rceil} \right\rfloor$$

$$= \left\{ M(A,n) = 0 \Rightarrow n = a\left\lceil \frac{n}{x} \right\rceil + b\left\lceil \frac{n}{y} \right\rceil . \right\}$$

$$n. \qquad \square$$

Proof of Theorem 5.1 (*conclusion*). Lemmas 5.2–5.8 complete the proof of Theorem 5.1. □

Again, the cyclic schedule length $n = LM(A)$ and the partitioning functions *Place*1 and *Place*2 may be used to generate a FOLS rather than an actual cyclic schedule. In

fact, the FOLS is exactly the one given in Section 4. Now, of course, n is $LM(A)$ and *Place*1 and *Place*2 are the ones given in this section. This leads to the following theorem.

Theorem 5.9. *The minimum pinwheel scheduling problem restricted to instances with only two distinct numbers is in S-P-C.*

Proof. Follows from Theorem 5.1, the program in Section 4, and the above discussion. □

In Section 2 we proved that no cyclic schedule shorter than $LM(A)$ is possible. In this section we have demonstrated how to create a cyclic schedule of length $LM(A)$. We also show how to generate in polynomial time a FOLS that produces that schedule. Thus we have demonstrated a complete solution to the minimum cycle problem.

6. Conclusions

In this paper we have addressed the pinwheel real-time scheduling problem restricted to two distinct integers. Given an instance of this problem, we have identified the following three questions:

1. The pinwheel decision problem: Can it be scheduled?
2. The pinwheel scheduling problem: Can a "useful" representation of a schedule be produced?
3. The minimum pinwheel scheduling problem: Can a "useful" representation of the shortest cyclic schedule be produced?

In Section 4 we proved that instances with only two distinct integers can always be scheduled, thus taking care of the first item. Our solutions to the remaining two problems involve the concept of a FOLS: a Fast OnLine Scheduler. A FOLS is a program that generates the scheduling sequence in constant time per item generated. A suitable FOLS that can be generated in deterministic polynomial time constitutes a useful representation of a schedule. In Section 4 we showed how to generate a FOLS for certain easily-computed cyclic schedule lengths. In Section 5 we showed how to generate a FOLS for the minimum cyclic schedule length. While this length is not as easy to compute as the lengths used in Section 4, the computation may still be done in deterministic polynomial time. Thus we have solved all three problems.

Acknowledgment

We thank C.L. Liu for directing our attention to [9] and for other helpful comments.

References

[1] S. Baruah, L. Rosier, I. Tulchinsky and D. Varvel, The complexity of periodic maintenance, in: *Proc. 1990 Internat. Computer Symp.*, pp. 315–320.

[2] R. Holte, A. Mok, L. Rosier, I. Tulchinsky and D. Varvel, The pinwheel: a real-time scheduling problem, in: *Proc. 22nd Hawaii Internat. Conf. on System Science*, 1989, pp. 693–702.

[3] H. Lenstra, Integer programming with a fixed number of variables, *Math. Oper. Res.* **8**(4) (1983) 538–548.

[4] J.Y-T. Leung and M.L. Merrill, A note on preemptive scheduling of periodic, real-time tasks, *Inform. Process. Lett.* **11** (1980) 115–118.

[5] C.L. Liu and J.W. Layland, Scheduling algorithms for multiprogramming systems in a hard-real-time environment, *J. ACM* **20** (1973) 46–61.

[6] A.K. Mok, Fundamental design problems of distributed systems for the hard-real-time environment, Ph.D. thesis, MIT Laboratory for Computer Science, 1983.

[7] A. Mok, L. Rosier, I. Tulchinski and D. Varvel, Algorithms and complexity of the periodic maintenance problem, in: *Proc. 15th Symp. on Microprocessing and Microprogramming (EUROMICRO '89)*, 1989, pp. 657–664.

[8] A.K. Mok and S. Sutanthavibul, Modeling and scheduling of dataflow real-time systems, in: *Proc. IEEE Real-Time Systems Symposium*, 1985, pp. 178–187.

[9] W.D. Wei and C.L. Liu, On a periodic maintenance problem, *Oper. Res. Lett.* **2**(2) (1983) 90–93.