Flight Test Data Cycle Map Optimisation

David Panton Centre for Industrial and Applicable Mathematics
Maria John University of South Australia
Stephen Lucas Mawson Lakes South Australia 5095
and
Andrew Mason Engineering Science University of Auckland

August 31, 2006

Abstract

Flight testing of aircraft in which new design concepts, problems, and deficiencies are examined, involves the collection of a large quantity of data or test parameters allocated onto blocks within a telemetry frame or Data Cycle Map (DCM). Individual parameters are sampled at different rates and may vary in both their word length and the number of words used to represent them. During a flight test these parameters are digitally represented and placed in a certain order within the DCM before being transmitted to the ground. When planning a flight test it is necessary to determine the order in which parameters are placed on the DCM to guarantee the integrity of the data, and other industry standards, for example, relating to the length of frames and the requirement for periodic placement of parameters.

This process is usually performed manually with some computer assistance, but typically involves several week's work for a number of people. As flight tests become more complex, restrictions on bandwidth and personnel have become limiting factors.

We present an algorithm for the design of DCMs and discuss situations in which the placement of sets of parameters on such frames is not possible using a number-theoretic analysis of the data cycle map structure. A fast and efficient optimisation approach for constructing DCMs using a set covering Integer Programming formulation (DCM-Opt) is discussed. A comparison of efficiencies for this algorithm and DCM designs produced by AutoTelem^{TM} will be discussed.

1 Introduction

Aircraft flight testing involves the collection of data from specially instrumented aircraft flown in planned testing missions. Typically several hundred **parameters**, such as speed, altitude, mechanical stress, pressure, etc. are sampled in such flight testing missions. Samples are taken from sensors and transmitted to a ground receiving station. Prior to this transmission the data is multiplexed into a data structure called a **data cycle map (DCM)**. The individual parameters may vary in the number of words required to store them and the length of the word. Typically one to three words with word lengths of from 8 to 32 bits are required. In addition, sampling rates will also vary from one parameter to another. The way in which the data cycle map may be constructed is subject to certain standards as set down by the Inter Range Instrument Group (IRIG) [1]. While these regulations are too numerous to discuss here, the essential requirements relevent to this discussion can be summarised as follows:-

- Parameters must be placed on the map in accordance with their word lengths and sample rates;
- The map data structure may be viewed as an array called a **major frame** whose rows are called **minor frames**;
- Minor frames can be no longer than 512 16 bit words in length;
- Major frames can consist of no more than 256 minor frames;

- Each minor and major frame must start with frame synchronisation words and contain a frame id;
- Each parameter must be periodically spaced on the map. The length of the period is determined by the sample rate and the length of the **major frame**;
- A data cycle map is a repetition of several major frames.

Since some parameters are collected at lower sample rates than others, such parameters may not appear on every minor frame, but at least once on every major frame. Such parameters are said to be **subcommutated**. On the other hand, parameters which occur at least once on each minor frame are said to be **supercommutated**.

The process of designing a data cycle map prior to a flight test is complex and time consuming, [2, 3]. This process may take several week's work for a number of people. Further, as flight tests become more complex, the size of the map required may extend the limits of available bandwidth, necessitating more than one flight to collect the data. It is therefore important that the process of data cycle map construction is as efficient as possible. Efficiency in this case involves not only the time taken to construct the map, but also the amount of unused space on the major frame. A more precise definition of efficiency will be discussed later. From an operational viewpoint DCMs are constructed using what are essentially computer assisted manual systems (for example FTIMS [7]). More recently, a technique has been developed for taking a generated DCM and applying an improvement heuristic to it [7], however it has been shown that while this technique improves on the manually generated solution it gives results which are far from optimal. To date, optimisation techniques applied to DCM construction have not been used in an operational environment. Past experience with DCM generation has shown that when the data satisfies certain power of two relationships fast solutions can be found. This idea has been addressed in [8] where it has been extended to consider certain $2^n p^k$ relationships, for some prime number p, in the parameter sample rates. Efficiencies obtained using this method are low however. The software package AutoTelem^{TM} developed by QUEST Integrated Inc. uses a local search procedure to produce near optimal DCMs and has been commissioned by the United States Airforce. At the time of writing AutoTelem^{TM} was still in the evaluation phase.

This paper discusses the process of data cycle map construction from the data input phase to the generation of optimal telemetry frames. In section 2 we discuss the design of the DCM based on the nature of the input data and the constraints imposed by the IRIG regulations. The efficiency of a DCM will be defined in terms of our ability to generate frames which minimise the amount of unused space. Some interesting number-theoretic results will be discussed concerning our ability to generate efficient frames. The relationship between generating DCMs and juggling will also be considered. In section 3 we discuss the use of a set covering model as an optimisation tool for generating optimal DCMs. Section 4 discusses the results of a comparative study between DCM-Opt and AutoTelem^{TM} with respect to their relative efficiencies and speed of execution.

2 The Data Cycle Map Structure

2.1 Introduction

Construction of a data cycle map is a complex process. The complexity lies not only in the number of parameters required to be placed on the map, but more particularly in the fact that, individually, they must be equally spaced. Manual methods for constructing maps are faced with avoiding coincident placements. This avoidance becomes more and more difficult as the map construction process proceeds. By necessity, manually constructed maps therefore contain a very high percentage of 'empty' space as a result of avoiding coincident placement.

In considering the application of optimisation methods to data cycle map construction we need to take into account our ability to manage the task in terms of memory and computational requirements. The construction of an entire major frame for example will almost certainly require a model which is too unwieldy and unmanageable. For this reason it will be necessary to decompose the problem into smaller tasks whose solutions can be used as building blocks for solving the larger problem. Our approach therefore will be to construct minor frames whose replication can be used to define the entire major frame. In doing this it will be essential that parameter sample rates are preserved and that parameter periodicity is also achieved, along with other IRIG standards to be discussed later. To assist in this decomposition approach we will represent each parameter in the DCM in terms of a specified number of words. Each word will occupy a 'slot' in the DCM. The use of bits to occupy slots and represent parameters would be potentially more efficient in a situation where maps required parameters with varying word lengths, however the resulting models would be unwieldy. In our model we select a uniform word length equal to the largest used in the map, resulting in unused bits for parameters with shorter word lengths. Despite this potential reduction in efficiency our ability to find optimal solutions using this approach will outweigh this disadvantage.

2.2 Notation

Consider the following notation to be used in this section:

N	=	Number of distinct parameter sample rates
r_i	=	Required sample rate for parameter i (samples/second)
d_i	=	Number of signals sent at rate r_i
p_i	=	Sample rate for parameter i in the minor frame
m_i	=	Period in word intervals for parameter i in the minor frame
w_i	=	Number of words required for parameter i
globalw	=	Global word length (bits)
mfl	=	Minor frame length (words)
mfr	=	Minor frame rate (number of frames/second)
nmf	=	Number of minor frame in each major frame
nfb	=	Number of bits in designed frame
E	=	Efficiency

2.3 The structure of the minor frame

The design of a suitable telemetry frame is dependent on several factors. Apart from periodicity and minimum required sample rates, IRIG Class I conditions also require that transmission bit rates fall within a minimum and maximum level, each minor frame has two header synchronisation words and a minor frame id word, there is a common word length for each parameter, minor frames contain no more than 512 16 bit words, and that there are no more than 256 minor frames in a major frame. Although we will not be concerned with many of these operational issues it is important that they are taken into account when designing a suitable telemetry frame. In order to accommodate all parameters on the frame in 'slots' of equal length we will adopt a common or global word length *globalw*. This may vary from one data set to another, however for the purposes of this discussion we assume it is 16 bits in length. We also assume that the header words are placed at the beginning of each minor frame, but the frame id may be placed anywhere in the frame. Within these constraints the task is to place the measured parameters onto a frame according to their required sample rates and periodicity. As we will see the periodicity will depend on the size of the frame. There are normally several options available on the size of the frame, however it is our task to select a frame size in which the total amount of unused space is as small as possible.

The efficiency E of a frame design can be defined as the ratio

$$E \;=\; \text{required bits}/nfb,$$

where the bit rate in the frame design

$$nfb \;=\; mfr * mfl * globalw.$$

Since the required bit rate is given by $r_i * d_i * w_i * globalw$ the efficiency E can be written

$$E \;=\; \sum_{i=1}^{N} \;(r_i * d_i * w_i * globalw)/nfb. \tag{1}$$

Note that the required number of bits (numerator) does not include header or frame id words but that these are included in computing the value of nfb.

Since in general it is too hard to design the entire major frame at once we consider an approach where we design a set of minor frames which when taken together constitute the entire major frame. As long as the r_i values are minimally satisfied and their placement is periodic it is not important that each parameter is placed in each minor frame. In most cases a parameter will appear at least once in each minor frame, and we say it is *supercommutated.* If however a parameter does not appear in every minor frame, but as long as it occurs at least once in the major frame, we say it is *subcommutated.* To see how this situation arises we consider the following example:-

Example 1: Consider $N = 4$ parameter classes and assume that *globalw* $= 16$. Sample rates, the number of signals and the parameter word sizes are shown in table 1.

r_i	1	5	6	25
d_i	1	3	2	1
w_i	1	1	1	1

Table 1: A data set with 5 parameter classes.

In the process we will describe it is necessary to place the r_i values in increasing order as given in table 1. We now select one of the sample rates as the minor frame rate mfr and let $mfr = r_k$. In fact r_k must be selected so that minimum and maximum bit transmission rates are not exceeded, however we will not discuss this issue and always assume that the choice of r_k is legal. In this case the value of r_k represents the number of times the minor frame occurs per second. We now divide each of the input data rates r_i by r_k to give the vector

$$\{\frac{r_1}{r_k}, \frac{r_2}{r_k}, \ldots, \frac{r_N}{r_k}\},$$

where the k^{th} element $r_k/r_k = 1$, all elements to the left of the k^{th} are less than 1 and all elements to the right of the k^{th} are greater than 1. To illustrate, suppose we choose $r_k = 5$ giving the vector

$$\{\frac{1}{5}, \frac{1}{1}, \frac{6}{5}, \frac{25}{5}\}.$$

We now rewrite this as a vector of minor frame rates $\{p_1, p_2, \ldots, p_N\}$ which in this case is $\{5, 1, 2, 5\}$. Numbers to the right of 1 are rounded up ($\lceil r_j/r_k \rceil \quad j > k$) which effectively means that they may be oversampled in the minor frame. The resulting rates in this case represent parameters which are *supercommutated* in the minor frame. Thus parameter three will occur twice in each minor frame and parameter four five times. Since the frame rate is 5 in this case then the minor frame is repeated 5 times per second resulting in parameter three appearing 10 times per second, compared with its required sample rate of 6 per second. Numbers to the left of 1 must be rescaled so that their numerators are all equal to 1. The rescaled values are then inverted to give minor frame rates for *subcommutated* parameters. Thus the value of 5 for parameter 1 means that this parameter will occur only once in every 5 minor frames. In order to ensure that all parameters will fit into a minor frame with periodicity satisfied we need to add up the space required for both subcommutated and supercommutated parameters together with 3 slots for the header and frame id words. This value may have to be increased to the nearest multiple of the LCM of the supercommutated parameters to ensure that each supercommuated minor frame rate $p_j \quad j > k$ divides the minor frame length and hence periodicity is satisfied. In general the space required by the supercommutated data is given by

$$\sum_{i=k}^{N} p_i * d_i * w_i. \tag{2}$$

In our example this gives a value of 12. In general the space required by the subcommutated data is given by

$$\sum_{i=1}^{k-1} w_i \lceil d_i/p_i \rceil, \tag{3}$$

which in our example yields 1. In addition we add 3 words for the header and frame id, giving a total of 16. Since 5 does not divide 16 this must be increased to 20 which is the nearest multiple of LCM(2,5). Thus we have a minor frame length of $mfl = 20$. To summarise, we are now generating minor frames each of length 20 16 bit words. Parameter one will be placed in only one of these (any one will do), while parameters two, three, and four will be placed 1, 2, and 5 times respectively at appropriate positions to ensure wrap-around periodicity. Note that the number of minor frames required is in general given by

$$nmf \; = \; \text{LCM}(p_1, p_2, \ldots, p_k) \tag{4}$$

which will guarantee that subcommutated parameters can also be periodically placed. In our example $nmf \; = \; 5$. Finally we need to compute the efficiency of this construction. The required bit rate is 848, whereas the designed bit rate is 1600. Our efficiency is therefore 848/1600 or 53%. $\square$

The process described in example 1 can be carried out for all possible values of r_k and a value of E calculated for each. A number of factors may influence the value of E. These include the degree of rounding up required to determine p_i values leading to oversampling and hence wasted space, the gap between the nominal minor frame length and the value mfl required to satisfy periodicity and the minor frame rate necessary for periodicity of the subcommutated parameters. For very large data sets all or many of the choices may violate the maximum minor frame length of 8192 bits. In this situation a simple device for restoring legality is to introduce header splits into the minor frame. In this situation the header can be thought of as another parameter with a sample rate determined by the required number of splits. For example suppose that the most efficient frame design gives a frame length of 842 16 bit words, including the original header of two words and a frame id of one word. We introduce another two word header and treat it as a parameter with a sample rate of 2. The additional 2 words can be added to the original nominal minor frame length which will in most cases still give a frame length less than or equal to 842. If necessary the revised mfl may need to be adjusted up by adding another LCM factor of the supercommutated parameters. When the new mfl value is computed (assume for the sake of this arguement this is still 842), our original minor frame of length 842 can now be replaced by two minor frames each of length 421. Periodic placement of the header will guarantee that they are placed at the start of each of these two new frames. The header splitting strategy will work as long as the total number of minor frames does not exceed the limit of 256. If this limit is exceeded the data set is too large to conform to IRIG class I conditions.

Preprocessing of the data will examine all options and their relative efficiencies. Under normal circumstances the option which has the highest efficiency will be chosen and will generate a feasible minor frame set. Under certain circumstances however feasible minor frames will not exist. An example will illustrate.

Example 2: Consider the data given in table 2

r_i	1	3	5
d_i	1	2	1
w_i	1	1	1

Table 2: A data set with 3 parameter classes.

Table 3 shows all possible rates, minor frame lengths and efficiencies. Of the three optional frame rates the most efficient is a single frame with efficiency 80%.

r_k	mfr	mfl	$E(\%)$
1	1	15	80
3	3	8	50
5	5	7	34

Table 3: Alternative frame rates and efficiencies for data from table 2

The natural choice in this case is a frame rate of 1 with no subcommutation. The periods m_i for each parameter i are given by mfl/p_i giving $\{15, 5, 3\}$ with a space of 2 words reserved at the beginning for the header and an additional space for the frame id. It is easy to show by construction that placement of the parameters on this frame is impossible without avoiding coincidence. In this case we are forced to try a less efficient frame construction. This issue will be explored in the next section.

2.4 The non-coincident placement of parameters on a minor frame

The example in the previous section fails because of the relative prime relationship between the periods for parameters two and three. As we will establish later in this section, this is a sufficient condition for coincidence to occur, however it is not a necessary condition, and a much deeper analysis of the problem is required. Some insight into this problem can be gained by reference to the Chinese Remainder Theorem in its more general form (CRT) [6]. which can be stated as follows:-

'The set of n linear congruences $x \equiv a_i(mod\ m_i)$ has a solution x **if and only if** the greatest common divisor of every pair of moduli, $(m_i, m_j),\ i, j = 1, \ldots, n,\ i \neq j$ divides the corresponding $a_i - a_j$'.

In relation to our problem, the m_i represent parameter periods, and the a_i the initial parameter placements. As we will see from the following examples, our ability to find a solution for the variable x is related to the issue of coincident placement. Note that for examples in this section we will assume that all parameters are supercommutated. The analysis is easily extendable to the more general case since the size of the minor frame is determined as a multiple of the LCM of all supercommutated parameters. In addition we will assume without loss of generality that a single word header is used.

Example 3: Consider 3 parameters with $p_1 = 1,\ p_2 = 4,\ p_3 = 8$. We assume all parameters have word length 1, hence $\text{LCM}(p_1, p_2, p_3) = 8$, $\mathit{mfl} = 16$, and corresponding periods are $m_1 = 16,\ m_2 = 4,\ m_3 = 2$. It is easy to find a non-coincident solution in this case. Indeed if our initial placement for parameter 3 is $a_3 = 2$, parameter 2 $a_2 = 3$, and parameter 1, $a_1 = 5$, for example, then the associated congruences can be written $x \equiv 2(\mathit{mod}\ 2),\ x \equiv 3(\mathit{mod}\ 4),\ x \equiv 5(\mathit{mod}\ 16)$. Note that in this case no unique pair (m_i, m_j) divides $a_i - a_j$. Non-coincident placement thus corresponds to the property $(m_i, m_j) \not| \ a_i - a_j$ for every unique pair i, j. No solution x can be found for the set of congruence relations in this case.

Example 4: Now consider 3 parameters with $p_1 = 1,\ p_2 = 4,\ p_3 = 6$. We assume all parameters have word length 1, hence $\mathit{mfl} = 12$ and corresponding periods are $m_1 = 12,\ m_2 = 3,\ m_3 = 2$. In this case whatever initial placements are chosen coincidence is unavoidable. For example suppose our initial placement for parameter 1 is in position 5, parameter 2 in position 3, and parameter 3 in position 2, then the associated congruences can be written $x \equiv 5(\mathit{mod}\ 12),\ x \equiv 3(\mathit{mod}\ 3),\ x \equiv 2(\mathit{mod}\ 2)$. We see that since $(m_2, m_3) = 1$, this will divide any difference $a_2 - a_3$ and hence whatever initial placement we select for parameters 2 and 3, coincidence is unavoidable.

What should now be clear from these examples is that non-coincidence of placement corresponds to being able to find (m_i, m_j) values which do **not** divide $a_i - a_j$. Moreover we need this situation to apply for **all** such i, j combinations. In this situation a solution x can never be found, and so in a sense we are seeking conditions in which no solution x can occur for any subset of the congruences.

We have already identified a situation in which coincidence cannot be avoided, as illustrated in example 4, where two periods are relatively prime. This observation can be formalised in the following result:

Theorem 1: Let $p_1, p_2, \ldots, p_n$ be the sample rates of n parameters in a minor frame with associated periods $m_1, m_2, \ldots, m_n$. A sufficient condition for the coincidence of parameters i and j in the frame is that periods m_i and m_j are relatively prime.

Proof: Consider p_i and p_j, and their associated periods m_i and m_j, which are relatively prime. Assume that parameter i is placed first at position a_i, then subsequent placements are defined by $a_i + r_i m_i$ with $0 \le r_i < p_i$. Parameter j can now be placed at positions $a_j + r_j m_j$ where $0 \le r_j < p_j$ and $a_j \neq a_i + k m_i$ for integer k. Consider the condition for which there is at least one value for which positions of parameters i and j coincide, namely $a_i + r_i m_i = a_j + r_j m_j$. This can be rewritten as $r_i m_i - r_j m_j = a_j - a_i$. This represents a diophantine equation in variables r_i and r_j. It is well known that a necessary and sufficient condition for such equations to have solutions is that the greatest common divisor of m_i and m_j divides $a_j - a_i$, i.e. $(m_i, m_j) | a_j - a_i$. We note that if m_i and m_j are relatively prime then $(m_i, m_j) = 1$ which will always divide $a_j - a_i$. This means that whatever choice of initial values, a coincidence of positions will occur and hence periodic assignment is not possible in this case. □

There is an interesting parallel between juggling and the placement of parameters on a telemetry frame [4]. A common notation used in juggling literature is the site swap where a juggling pattern can be represented by a set $H = \{h_i\}\ i = 1, \ldots, F$ where each h_i represents a height function of a throw, and F is the number of throws in the pattern. For example the sequence 51414 denotes a period 5 pattern with 3 balls. Heights in this case correspond to periods for the DCM problem. Periodic in this context refers to the pattern rather than the height of the balls. If we call the balls A, B, C this gives the pattern $ABBCC$ where ball A has constant period 5 but balls B and C alternate from period (height) 1 to period 4. The number of balls in this pattern is given by the average height which is 3 and the site swap pattern is repeated periodically. Unless the average height is an integer this is not a feasible juggling pattern. This corresponds to the DCM requirement that the length of the minor frame is divisible by all parameter periods. Such site swap patterns are not guaranteed to be periodic however. The conditions under which periodic juggling patterns occur have been investigated in [5].

Theorem 2: A site swap sequence $h_0h_1 \ldots h_{n-1}$ is periodic if and only if $(h_t + t)(mod\ n)$ is a permutation of $\{0, 1, \ldots, n-1\}$, where t gives the position in the sequence $0, 1, \ldots, n-1$.

For example the site swap pattern 51414 gives the permutation {02143}. This result also allows for 'empty hand' throws so that missing positions in the sequence denoted by 0s can occur. To illustrate, consider a simple DCM requirement of three parameters with sample rates r_i : 1 3 6. Ignoring the need for a frame header this requires a frame length of 12 to ensure periodic placement with associated periods m_i : 12 4 2 and hence there are 2 empty slots. We need a permutation for the sequence 12 4 4 4 2 2 2 2 2 2 0 0. Labelling the parameters (balls) $A\ B\ C$ it is easy to find a site swap sequence which is periodic, for example

$$A\ C\ B\ C\ 0\ C\ B\ C\ 0\ C\ B\ C.$$

The associated permutation in this case is {0 3 6 5 4 7 10 9 8 11 2 1}. While it is easy to find a periodic pattern and associated permutation (with constant height balls) in this case, in general the task of finding such a permutation is at least as hard as the original problem. The juggling result requires a simple test for a *given* pattern, whereas in our case a pattern must first be found. We are therefore more concerned with determining conditions for detecting coincidence which are based on the initial data. One such sufficient condition for coincidence has already between discussed, namely when two periods are relatively prime. However we require necessary conditions for coincidence so that when these conditions are not satisfied we can be certain that coincidence will not occur.

2.5 Coincidence Conditions

Analysis of a large number of data sets has led to the following observations:-

Definition: A *coincident set of order k* is a set of periods $m_i, \ i = 1, \ldots, k$ for which each distinct gcd $(m_i, m_j), \ i, j = 1, \ldots, k \ (i < j) \ = \ k - 1$.

An example of a coincident set of order 3 is $m_1 = 10, \ m_2 = 6, \ m_3 = 4$. In this case we have $(m_1, m_2) = (m_1, m_3) = (m_2, m_3) = 2$.

Theorem 3: Consider a set of n parameters with periods m_i and associated initial placements a_i, $0 < a_i \leq m_i$. A sufficient condition for coincident placement is that the periods contain a coincident set.

Proof: First, we make the observation that when $k = 2$ and $(m_1, m_2) = 1$ we have the case of two relatively prime periods considered earlier. Suppose m_i form a coincident set of order k. Observe that with k distinct values of initial placements a_i it is impossible to avoid at least one difference $a_i - a_j$ divisible by $k - 1$. Since $(m_i, m_j) \ = \ (k - 1)$ for all $i, j = 1, \ldots, k \ i < j$ then at least one coincidence must occur as a consequence of the CRT. □

Example 6: Consider five parameters with $p_1 = 1, \ p_2 = 6, \ p_3 = 10, \ p_4 = 30, \ p_5 = 30$, all with word length 1. This gives $mfl = 90$ with associated periods $m_1 = 90, \ m_2 = 15, \ m_3 = 9, \ m_4 = 3, \ m_5 = 3$. In this case we have 4 periods $m_2, \ldots, m_5$ with $(m_i, m_j) = 3, \ i < j$, and hence a coincident set of order 4. For example with the initial placements as shown $|4|5|2|4|5|3| \cdots$ we see that the corresponding congruences are $x \equiv 1(mod\ 3), \ x \equiv 2(mod\ 3), \ x \equiv 3(mod\ 15), \ x \equiv 6(mod\ 9)$. In the last two of these we have $(15, 9) = 3 | 6 - 3$.

Note that for small data sets proof of coincidence can be achieved by construction. For larger data sets the number of permutations of placements makes the constructive approach impossible. In the next section we will discuss an optimisation model which implicitly performs this construction process and hence can be used to test for coincidence. Validation of these results is also available using the AutoTelem^{TM} software.

While a coincident set is sufficient to ensure coincidence, it is not necessary. Consider the following example:

Example 7: It can be shown that coincident placement is unavoidable with the periods $m_1 = 120, m_2 = 20, m_3 = 12, m_4 = 4, m_5 = 2$. These periods contains no relatively prime pairs or coincident sets however.

Observation of a large number of data sets has suggested the following conjecture which if proven strengthens the coincident set result.

Conjecture: Consider the sequence of periods $m_1, \ldots, m_N$ where period m_i has d_i associated signals and periods are placed in descending order. Periods in the subset m_1 represent the minor frame length. Take the subsets of periods $m_2, \ldots, m_N$ and treat each repeated period independently. Assume there are a total of k such independent periods. Compute all gcd pairs $(m_i, m_j) \ \ i \ < \ j$, and compute the average gcd over all such pairs. If this average is $\leq \ k - 1$ then coincidence of parameters will occur.

Omission of the highest period parameter can be justified on the basis that none of these parameters is repeated in the placement process. They need only be placed in one position in the frame and there is guaranteed to be space for them. Every data set tested for which the average as defined above is $\leq k-1$ gives no feasible solution and hence coincidence. In example 7, the average gcd as defined above is equal to 3 $(k-1)$. An isolated case has been found for which coincidence occurs with an average $> k-1$ thus the conjecture is at best a more general sufficiency condition for coincidence.

Despite the absence of a necessary condition for coincidence, DCM generation can be carried out with relative efficiency. Models discussed in the following section allow us to quickly test a number of options and take necessary action if an option fails. Never the less, research is continuing in order to develop a better understanding of the process.

3 Optimisation Models

Three alternative models were considered to generate optimal or near optimal DCMs. The first considered was an Integer Programming (IP) formulation in which parameters were directly assigned to a minor frame with all desired properties. This method proved feasible but too slow to be operationally useful. The second model considered involved the use of Genetic Algorithms. While this approach has some potential it is not sufficiently developed to be a contender at this stage. The third approach which has proved successful has been to use a set covering model.

3.1 Set Covering Model- DCM-Opt

Having determined the length of the minor frame *mfl* we can now enumerate all possible placements for a given parameter on this frame. Each placement pattern will be referred to as a *tour*. For example, consider the data from table 1 with a minor frame rate of 5 and a minor frame length of *mfl* = 20. Consider the parameter with required rate 25 and a supercommutated rate of 5. This parameter will have period 4 on the minor frame and has the following tours as shown in table 4.

Our problem is to select one tour for each parameter so that the minor frame is covered with no overlap. Generation of the parameter tours automatically ensures that they are periodic. Consider the following model:-

Define the binary variable x_i to be 1 if column i is selected, and 0 otherwise. Note that columns can be divided into subsets S_i, one for each parameter, and that only one column from each subset can be chosen.

$$\text{Minimise} \qquad \sum_k s_k \tag{5}$$

$$\text{subject to} \qquad \sum_i a_{ik} x_i + s_k \;\leq\; 1 \quad \forall k, \tag{6}$$

tour → frame position	1	2	3	4
1	1			
2	.	1		
3	.		1	
4	.			1
5	1			
.	.	.	.	
.	.	.	.	
9	1			
10	.	1		
11	.		1	
12	.			1
13	1			
.	.	.	.	
.	.	.	.	
20				1

Table 4: Enumeration of all tours for parameter period 4.

where $a_{ik} = 1$ if parameter i covers position k and 0 otherwise, and s_k is a slack variable for frame position k. Thus if frame position k is not covered by a parameter tour the slack variable $s_k = 1$. In addition the constraints

$$\sum_{r \in S_i} 1_r = 1 \quad \forall i, \tag{7}$$

where $1_r = 1$ for each element in the set, ensure that only one column is selected from each parameter set.

The objective is to minimise the sum of the slack variables s_k which is consistent with a solution having as little unused space as possible. Savings in the number of variables can be made by noting that in the generation of the parameter tours there is considerable duplication since parameters with the same minor frame sample rate and word requirements will have identical tour sets. Parameters can be grouped into parameter classes and tours generated for each class. In this case the constraints which ensure that only one parameter is selected from each set are modified so that the appropriate number are selected from each class. Once the optimal solution is found, members of each class are assigned arbitrarily to each tour and the minor frame map reconstructed.

An interesting feature of this model is that we know *à* priori what the optimal solution objective value is, since we know that a non-coincident solution must leave a number of empty slots in the minor frame equal to the difference between the computed minor frame length mfl and the nominal minor frame length. This enables us to set the tightest possible bound on the optimal solution, thereby greatly reducing the size of the branch and bound tree. Two other observations are important. First, when these models are solved using CPLEX^{TM} 7.0 we have the option of seeking feasible rather than optimal solutions since in this case any feasible solution is automatically optimal. Second, Dual Simplex is used as the default LP solver, however when the number of columns in the model becomes significantly greater than the number of rows we switch to the Primal LP solver. Both of these strategies contribute significantly to reductions in execution times.

4 Results

A total of 1870 data sets representing bit requirements ranging from 640 to 1119968 were selected for analysis. Each algorithm was compared for the efficiency of its frame design and execution time.

All results were generated on a PC with clock speed 733 MHz. AutoTelem was run under Windows 95, while DCM-Opt was run under Red Hat Linux v6.2.

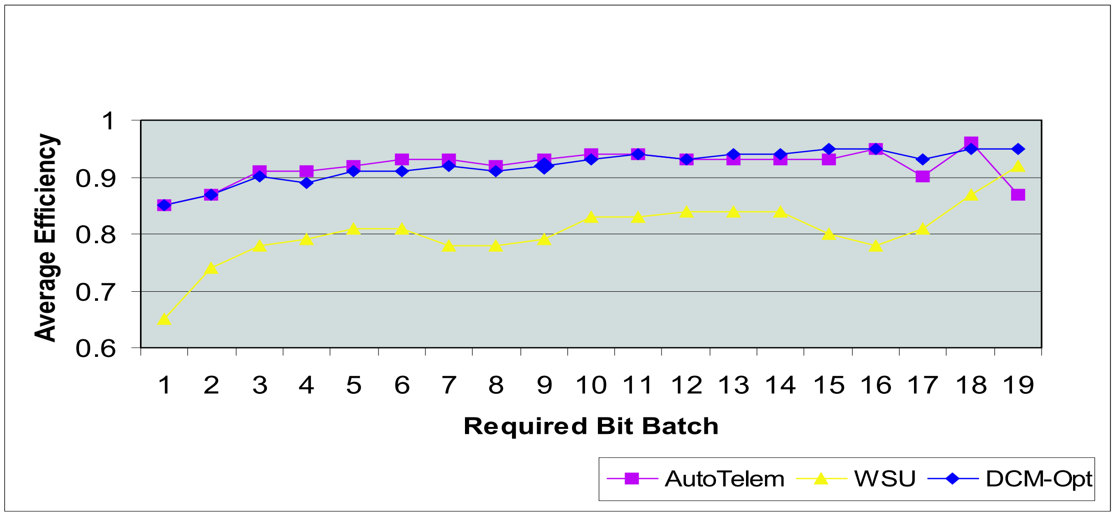

Figure 1: Average efficiency for each algorithm, over batches of 100 data sets which have been ordered in size of increasing bit requirements.

The relative efficiencies of DCM-Opt and AutoTelem^{TM} are displayed in Figure 1. These graphs represent the average efficiency as a function of the size of the input file required bits. All results were sorted in order of increasing required bits and the efficiency values averaged over batches of size 100 (except for the last batch which is of size 70). The median batch size is shown on the figure for each batch. It can be observed that for batches in the mid to high range of required bits the DCM-Opt algorithm generally produces solutions with a higher efficiency than for AutoTelem^{TM}, whereas for data sets in the low to mid range of required bits the reverse is true. Without knowledge of the algorithm used in AutoTelem^{TM} we surmise that this is a consequence of the header splitting strategy used for DCM-Opt, as described in Section 2.3, which will tend to be executed for larger data sets.

A comparison of execution times is shown in Figure 2. The results in this case are also sorted in order of increasing required bits and times are averaged over batches of size 100.

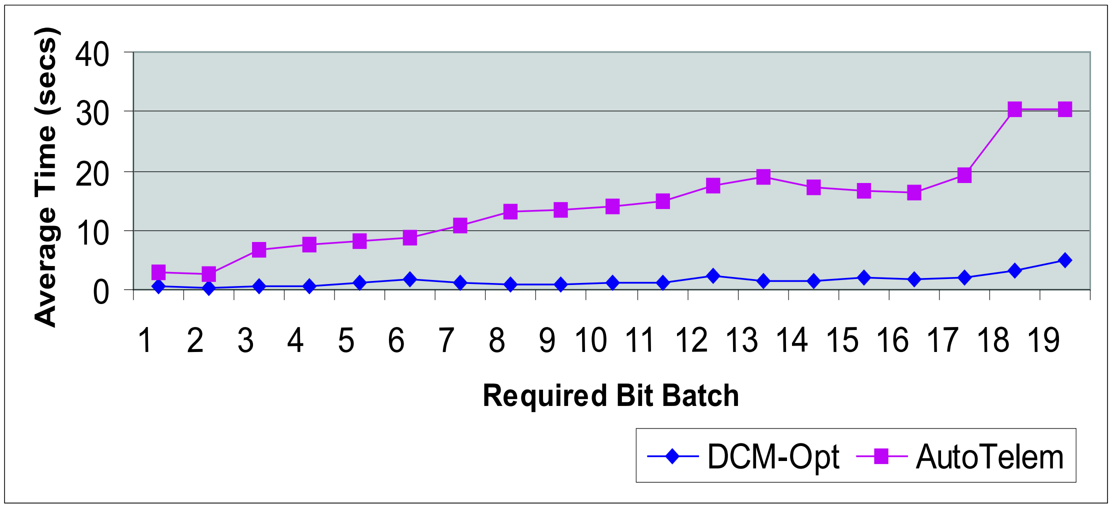

Figure 2: Average execution time in seconds for AutoTelem^{TM} and DCM-Opt, over batches of 100 data sets which have been ordered in size of increasing bit requirements.

It is clear that DCM-Opt executes faster than AutoTelem^{TM} across the range of data set batches. Both algorithms have low execution times in absolute terms compared with current frame generation practice.

Acknowledgements

This work was partly sponsored by an internal ITEE Divisional Grant from the University of South Australia and also by United States Airforce Contract number F04700-00-P-1181.

References

[1] Telemetry Group, Range Commanders Council, **IRIG Standard 106-96, Telemetry Standards**, Secretariate, Range Commanders Council, U.S. Army White Sands Missile Range, NM, (May 1996).

[2] Jones, Charles H. and Gardner, Lee S., **Telemetry Mapping Algorithm Project (TMAP)**, International Test & Evaluation Association Workshop, Lancaster, CA, (April 1997).

[3] Jones, Charles H. and Gardner, Lee S., **Automated Generation of Telemetry Formats** Proceedings of International Telemetering Conference, vol. XXXII, 1996, pp 635-644.

[4] Jones, Charles H. **Telemetry and Juggling** Proc. International Telemetry Conf. Vol XXXVI, Paper 00-20-2, 2000.

[5] Buhler, Joe, Eisenbud, David, Graham, Ron, Wright, Colin, **Juggling Drops and Descents** American Mathematical Monthly Vol 101 no. 6 pp 507-519, 1994.

[6] Griffen, Harriet, **Elementary Theory of Numbers** McGraw-Hill, New York 1954.

[7] Samaan, Mouna M., **Data Management of Flight Test Telemetry Frames**, Masters Thesis University of South Australia, 1998.

[8] Hudelson M., Webb W., **Telemetry Frame Generation Based on the 'Rule of $\mathbf{2^n p^k}$'** International Test & Evaluation Association Conference, Lancaster, CA, (April 2000).