A Novel Divide-and-Conquer Approach to Data Cycle Map Construction

Maria John[a] and David Panton[b]

Abstract

Aircraft flight testing is employed to evaluate an aircraft's performance; identifying design concepts, problems and deficiencies. The process involves the collection and recording of data such as speed, altitude, and pressure, which is then telemetered to the ground for analysis. The process of recording data; a set of parameters of varying lengths that are sampled at certain rates requires the construction of Telemetry Data Cycle Maps. A brief overview of a set covering Integer Programming formulation and a novel Divide-and-Conquer approach to generate Telemetry Maps will be discussed. In addition, a comparison of the relative efficiencies of the two algorithms using realistic data sets is presented.

Introduction

The process of aircraft flight testing requires the aircraft to be fitted with special instrumentation and execute a pre-planned test mission. The mission includes the collection of data (parameters) that need to be sampled at certain rates as requested by a customer. Customers are generally interested in flight safety and performance data. During a mission, samples are acquired from sensors, positioned accordingly within a data structure; *data cycle map* (DCM), and transmitted to a ground receiving station. The actual frame structure is known as a *major frame*, which is comprised of one or more fixed length rows called *minor frames*. Prior to the construction of a DCM, a uniform global word length must be selected. Typically word lengths from 8 to 32 bits are used and the map is then divided into slots of this length. Each parameter requested for testing has a required sampling rate; number of times it must be sampled per second, and size; number of words it will occupy when it appears in the DCM. Parameters that appear at least once per minor frame are known as *supercommutated* samples. While those that do not appear on every minor frame, but at least once per major frame are considered to be *subcommutated* samples.

The Inter Range Instrument Group (IRIG) [1] has developed a set of standard requirements for the construction of DCMs. A number of the essential requirements are:

- Parameters must be positioned periodically on the DCM according to their required sample rate and word length.
- All minor and major frames begin with frame synchronisation words and contain a frame identification word (frame id).
- A minor frame length can be up to 8192 bits in total.
- The number of minor frames per major frame is limited to 256.

At present DCMs are constructed using computer based manual systems such as FTIMS [6]. The construction of DCMs is considered to be complex, costly and time consuming. Additionally, as customer demands are becoming increasingly more complex the number of requirements to be sampled is growing. One map may take several weeks to generate and require more than one flight for the collection of data. Hence the generation of efficient DCMs has received a considerable amount of attention. Recently a software package, AutoTelem™ has been developed by QUEST Integrated Inc. [2] using a local search procedure to generate near optimal DCMs. This package has been commissioned by the United States Airforce.

In this paper, the process of DCM construction employing a Divide-and-Conquer approach and a set covering Integer Programming model will be

[a] Land Operations Division, Defence Science and Technology Organisation, West Avenue, Edinburgh, South Australia, Australia (Email: maria.john@defence.gov.au)

[b] Centre for Industrial and Applied Mathematics, University of South Australia, Mawson Lakes, South Australia, Australia (Email: david.panton@unisa.edu.au)

discussed. Furthermore, the results produced from a comparative study on the relative efficiency of the set covering technique and the Divide-and-Conquer approach are presented.

Frame Design and Efficiency

Formulating a DCM requires determining the size of the minor frames and the corresponding efficiency required to position each parameter periodically using a given sample rate. In addition to the previously mentioned IRIG constraints there are bit rate limits, consistent with a minimum bit rate of 10 and maximum bit rate of 5 million. The frame design process begins with the selection of a common global word size, *globalw*, where each map is divided into “slots” of this length. This word size can be from 8 to 32 bits. The problem then becomes to select a frame size such that the total number of empty slots is minimised.

There are normally numerous options available in the design of a frame for any given set of parameters, however what is required is the most efficient. Frame efficiency *E*, is measured as follows:

$$E = requiredbits / nfb\,,$$

where *requiredbits* is the minimum size of the DCM (in bits) if each parameter was positioned using its minimum required sample rate and did not need to be periodic, and *nfb* corresponds to the bit rate of the final designed DCM. Each parameter type i has three values associated with it; r_i, the required sample rate in samples per second; d_i, the number of signals; and w_i, the number of words required. Thus, *requiredbits* can be written as

$$requiredbits = \sum_{i=1}^{N} (r_i * d_i * w_i * globalw)$$

The value of *nfb* is given by

$$nfb = mfr * mfl * globalw\,,$$

where *mfr* and *mfl* represent the minor frame rate or number of minor frames per second and minor frame length (in words), respectively. Therefore, the efficiency becomes

$$E = \sum_{i=1}^{N} (r_i * d_i * w_i) / (mfr * mfl)\,.$$

It is necessary to note that the synchronisation and frame id words are included in *nfb*, but not in *requiredbits*.

Hence, to calculate the size of a DCM and its corresponding efficiency value for a set of parameters, it is necessary to ascertain the following primary attributes:

- minor frame rate;
- minor frames per major frame;
- words per minor frame;
- bits per word; and
- actual sampling rates for each parameter type.

In order to demonstrate the process of finding the necessary attributes, consider the data set illustrated in Table 1 and assume that *globalw = 16*. It is necessary to note that the rates are required to be in ascending order.

r_i	d_i	w_i
1	4	1
2	2	1
6	1	1
13	2	1

Table 1 An Eight-Parameter Data Set Divided into Four Data Rate Classes for Example 1.

Minor Frame Rate

The number of data rate classes is N (4). Select one of the sample rates, r_k (say $r_2 = 2$), as the *minor frame rate*, *mfr*. Thus, the minor frame will be repeated 2 times per second. A number of the IRIG constraints are considered in the selection of the minor frame rate: the minimum and maximum bit transmission rates and maximum minor frames per major frame must not be exceeded.

Number of Minor Frames per Major Frame

Divide each of the data rates by r_k to produce the following vector,

$$\left(\frac{r_1}{r_k}, \frac{r_2}{r_k}, ..., \frac{r_N}{r_k} \right) = (1/2, 2/2, 6/2, 13/2).$$

The *k*th element, r_k / r_k is 1, the elements to the left of it are less than 1 and those to the right are greater than 1. Furthermore, elements to the right, representing the supercommutated parameters, must be integers in order to supercommutate properly within the minor frame, and those to the left must have a numerator of 1 and be inverted in order to represent the allowable subcommutated rates. Elements to the right that are not integers are rounded up and elements to the left with a numerator not equal to 1 are rescaled. Thus, producing a vector of subcommutated and supercommutated sample rates,

$$(p_1, p_2, ..., p_N) = (2,1,3,7),$$

where supercommutated parameter i occurs p_i times per minor frame and subcommutated parameter j occurs once every p_j minor frames. For example, subcommutated parameter 1 occurs once every 2 minor frames. Supercommutated parameter 4 occurs 7 times per minor frame and although it has a required sample rate of 13 times per second, actually appears 14 times per second, as each minor frame is repeated 2 times per second.

The *number of minor frames per major frame*, *nmf*, is the least common multiple (*LCM*) of the subcommutation rates,

$$nmf = LCM(p_1, p_2, ..., p_k) = LCM(2,1) = 2.$$

This value guarantees that the subcommutated parameters are periodically placed.

Words per Minor Frame

The *number of words per minor frame*, mfl, consists of the synchronisation and frame id words plus the number of words required to accommodate the subcommutated and supercommutated parameters. In order to ensure periodicity for each supercommutated parameter, each supercommutated minor frame rate must divide the minor frame length. Thus, the nominal length may need to be increased to the nearest multiple of the LCM of the supercommutated parameters. In general the space required by the subcommutated parameters is given by

$$\sum_{i=1}^{k-1} w_i \left\lceil \frac{d_i}{p_i} \right\rceil.$$

The number of words required by the supercommutated parameters is given by

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

Therefore using the above example, the number of words required by the subcommutated and supercommutated parameters is 2 and 19 respectively. Assuming 2 synchronisation words and 1 frame id word, a total of 24 nominal words is required. However, as $p_4 = 7$ does not divide 24, this value is increased to the nearest multiple of $LCM(3,7)$ which is greater than 24. Consequently, the number of words per minor frame or the length of each minor frame is $mfl = 42$. The efficiency of this DCM construction would be 46.43%, as the required bit rate is 624 and the designed bit rate is 1344.

A number of factors may affect the efficiency value,

- The amount of rounding required to determine the supercommutated minor frame rates which lead to oversampling or wasted space.
- The difference between the nominal and final minor frame length required to ensure the periodicity of supercommutated data.
- The selection of the minor frame rate to satisfy the periodicity of subcommutated parameters.

Recall that this frame design has been determined on the basis of selecting $r_k = 2$, however, 3 other options are also available in this example.

In summary, the technique selects a range of minor frame rates. At each candidate rate, subcommutation and supercommutation rates are estimated and adjusted to produce feasible words per minor frame and minor frames per major frame, according to the IRIG constraints. A value of efficiency is calculated for each possible DCM construction. It follows that the aim is to select a minor frame rate and the adjusted data rates in order to generate the most efficient DCM conforming to the IRIG requirements. Several factors may influence this process.

1. For very large data sets, all or many of the choices may violate the maximum minor frame length of 8192 bits. In this case if possible a less efficient frame design is chosen which does not require this number of bits.

2. The data set may exceed the total bit rate maximum of 5 million or the maximum number of minor frames per major frame, and hence no DCM can be constructed.

3. No solution can be found for the highest efficiency despite (1) and (2) not occurring and a frame design with lower efficiency is tried.

Case 3 deals with certain nonconforming examples where placement of parameters on a chosen efficient frame is impossible without avoiding placement coincidence. The overlap of parameters causes the use of a less efficient frame construction. This case is discussed in detail in [3].

Set Covering Integer Programming Model

The ensuing approach uses a set covering Integer Programming model (DCM-Opt) to construct DCMs. DCM-Opt considers the construction of a minor frame to generate an entire DCM. Once a minor frame is established, it is replicated to construct a major frame.

The process of generating a minor frame is based on the principal of "packing" parameters into a frame while minimising the number of unused words and maintaining periodicity within the major frame. As the length, *mfl*, has been determined, it is used to enumerate the set of all possible placements within the minor frame for each parameter type.

Consider each placement pattern as a *tour*. The aim of DCM-Opt is to choose a tour for each parameter, not allowing any overlap within the DCM. Periodicity is ensured automatically during the generation of the tours. The problem is to minimise the amount of unused space or

minimise $$\sum_k s_k ,$$

subject to $$\sum_i a_{ik} x_i + s_k \le 1 \quad \forall k \quad (1)$$

$$\sum_{r \in S_i} 1_r = 1 \qquad \forall i \quad (2)$$

where x_i is 1 if column i is chosen, and 0 otherwise, a_{ik} is 1 if parameter i covers position k and 0 otherwise, and s_k is a slack variable for frame position k which is equivalent to 1 if position k is not covered.

The tours can be divided into subsets S_i for each parameter and constraint 2 ensures that exactly one tour is selected per subset. Savings in the number of variables can be made since parameters with the same minor frame sample rate and word requirements will have identical tour sets and can be grouped into parameter classes with tours generated for each class. The minor frame map is reconstructed by arbitrarily assigning the members of each class to each tour.

DCM-Opt was implemented using C and the optimisation code CPLEX^{TM} 7.0 as the solver.

Divide-and-Conquer Model

The following model employs a Divide-and-Conquer (DaC) approach to generate DCMs. As with DCM-Opt, DaC generates a minor frame in order to construct an entire DCM. The minor frame is replicated according to the final required number of minor frames per major frame, *nmf.*

The fundamental principle of the Divide-and-Conquer paradigm is to divide and solve several smaller problems, combining their outcomes to form a complete solution. Typically the Divide-and-Conquer concept can be employed when solving a large or difficult problem. The original strategy of Divide-and-Conquer is to

- Divide: Divide the problem into similar subproblems (generally of equal size);

- Conquer: Solve each subproblem either directly or recursively; and

- Combine: Combine the subproblems' solutions to create a single global solution.

The DaC model applies an innovative version of the traditional Divide-and-Conquer paradigm to construct *optimal* or near-optimal DCMs.

The aforementioned DCM-Opt requires an a priori process to find an efficient *mfl*. However, this method can not guarantee a feasible *mfl* will be chosen. The selection may include coincident placement of parameters. Thus, the *mfl* option would be deemed infeasible, a less efficient *mfl* would need to be considered, and the placement process recreated.

Consequently, the DaC model evolved in an attempt to develop a simple, robust DCM generating process for all requirements sets. The process can be demonstrated with five key steps, using the example given in Table 2; $n = 4$ parameter classes and $r_k = r_2\,(=5)$.

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

Table 2 An Eight-Parameter Data Set Divided into Four Data Rate Classes for Example 2.

1. Apply the preprocessing method described earlier to calculate the "initial" minor frame rates, p_i and nominal minor frame length, $nmfl$. Thus, if $r_k = r_2\,(=5)$ then $nmfl = 20$.

2. (a) Divide all supercommutated rates by the smallest supercommutated rate, $divisor = p_3 (= 2)$ using $adjustedp_i = \left\lceil {p_i}/{divisor} \right\rceil$ to produce column 5 in Table 3.

r_i	d_i	w_i	p_i	$adjustedp_i$
1	1	1	5	
5	3	1	1	
6	3	1	2	1
25	1	1	6	3

Table 3 Example 2 Data Set, including the first level *adjustedp_i* values.

(b) Modify the original supercommutated rates to incorporate the rounding factor using $p_i = adjustedp_i * divisor$.

(c) Calculate a new minor frame length $(= 24)$ corresponding to the new p_i values.

3. Reapply step 2 to the $adjustedp_i$ rates until the column contains only one element, as can be seen in Table 4.

r_i	d_i	w_i	p_i	$adjustedp_i$	
1	1	1	5		
5	3	1	1		
6	3	1	2	1	
25	1	1	6	3	1

Table 4 Example 2 Data Set, including the second level *adjustedp_i* values.

Therefore, from the initial $nmfl = 20$, the final $mfl = 24$. The minor frame was first divided into 2 subframes of lenth 12 (first round of step 2). The subframe of length 12 was then further divided into 3 subframes of length 4.

4. (a) Capture the commutated sample (parameter 8 from parameter class 4) for the smallest subsubframe from the final column, 6 in Table 4.

(b) Randomly place these parameters in available slots within the subsubframe.

		8	

This small frame is repeated 3 times to produce the subframe of length 12.

It is necessary to note that space must still exist for synchronisation words (say 2) at the beginning of the frame and a frame id anywhere within the frame.

		8				8				8	

5. Repeat step 3 to capture the commutated samples (5, 6, and 7) for the subframe (length 12) from column 5 (Table 4). These commutated samples are randomly placed in available slots in the subframe.

		8				8	5		6	8	7

The subframe is then repeated twice producing a minor frame.

		8				8	5		6	8	7			8				8	5		6	8	7

6. Repeat step 3 to capture the commutated samples (2, 3, and 4) for the minor frame (length 24) from column 4 (Table 4). These commutated samples are placed using the remaining slots in the minor frame.

H	H	8				8	5		6	8	7	2		8			3	8	5	4	6	8	7

Thus, the aim of the model is to continue to divide the nominal minor frame until the smallest subframe is generated with only commutated samples.

The minor frame is then rebuilt by adding in the previous commutated samples, subframe by subframe.

Consequently, an algorithm has been formulated that ensures the production of feasible efficient DCMs for all requirements sets.

Results of the Comparative Analysis

A total of 2292 data sets were analysed in this study. These comprise of a cross section of data ranging from 6352 to 416832
total bit requirements. Originally a slightly larger collection of sets was considered, however this contained several data sets that violated the maximum IRIG limits. All data sets were supplied by the United States Airforce.
Typically the DCM-Opt model locates an efficient, feasible minor frame length for a given DCM requirements set within a small amount of CPU time. However, for some data sets, as stated earlier, coincidence occurs. Thus, an alternative Divide-and-Conquer formulation was investigated, DaC, to build a DCM without requiring its final length to be known a priori, by taking advantage of the array structure associated with Telemetry Maps.

DCM-Opt and DaC are compared in terms of their relative frame design efficiency, using the averaged efficiencies over batches.

Efficiency

The relative efficiency of the two algorithms is depicted in Figure 1.

These graphs represent the average efficiency as a function of size of the required bits. To facilitate easy comparison of results, output was sorted in order of increasing required bits and the efficiency values averaged over batches of size 100 (except for the last which was of size 92). It can be seen that the relative efficiency for both DCM-Opt and DaC is in the high ranges, above 90% for most data sets.
DaC generally requires some oversampling of the original minor frame rates due to the dividing of the minor frame into subframes. However, there is a certain amount of oversampling that occurs when generating the minor frame sample rates initially for both DCM-Opt and DaC. Thus, the additional oversampling does not affect the ability of DaC to construct efficient DCMs. The two methods produce DCM telemetry frames with nearly identical efficiencies.

All results were generated on a PC with clock speed 733 MHz and were run under Red Hat Linux v6.2.

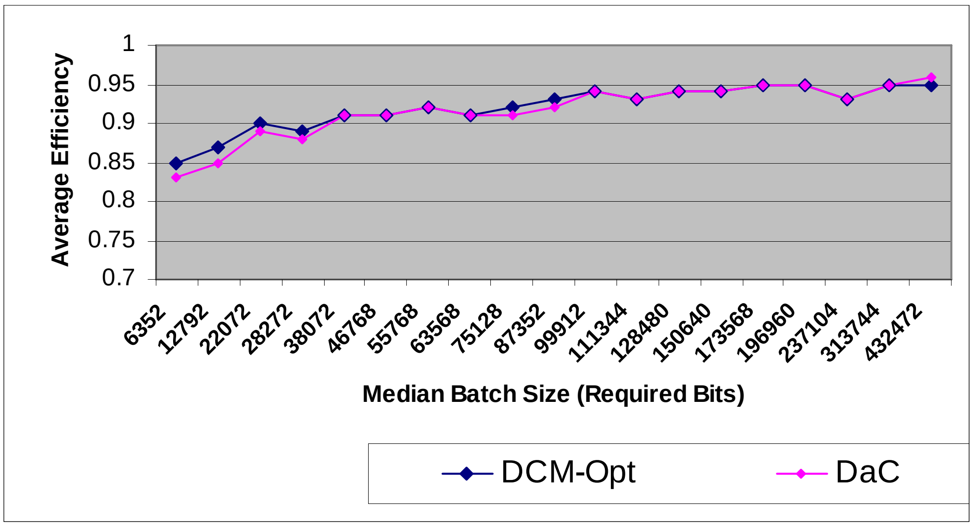

Figure 1 Average efficienct for DCM-Opt and DaC, over batches of 100 darta sets, ordered in size of increasing bit requirements.

Conclusion

Operations Research techniques, particularly Integer Programming, have often been employed to solve real-world problems in defence related research. The prevalence of heuristic techniques is increasing with the complexity and diversity of the problems that arise. Dealing with such complex real-world problems requires the consideration of efficacy, effectiveness and efficiency [5], in order to produce robust approaches that are able to be utilised successfully.

The primary outcome associated with the construction of Data Cycle Maps was the development of automated, comparative Integer Programming and Divide-and-Conquer models able to generate feasible efficient maps while complying with a set of telemetry standards.

Furthermore, it has been shown that DaC, using the simple Divide-and-Conquer paradigm, is comparable with the set covering Integer Programming model, DCM-Opt. DaC is able to generate as efficient telemetry frames for realistic data sets. The motivation behind developing DaC was to illustrate that simple methods can be effective in solving complicated problems.

References

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

[2] T.C. Folsom and P.D. Bondurant, “Automatic Telemetry Frame Formatting,” *Proceedings of the 44th International Instrumentation Symposium,* 1998.

[3] D. Panton, M. John, S. Lucas and A. Mason, “Flight Test Data Cycle Map Optimisation”, *European Journal of Operational Research*, vol 146, pp 48-497, 2003.

[4] D. Panton, M. John and S. Lucas, “Flight Test Data Cycle Map Optimisation”, *4th ITEA Workshop,* 2000.

[5] J.V. Rosenhead and J. Mingers, *Rational Analysis for a Problematic World Revisited,* John Wiley & Sons, Chichester, UK, 2001.

[6] M. M Samaan, “Data Management of Flight Test Telemetry Frames”, Masters Thesis, University of South Australia, 1998.