COMPLEXITY OF PCM FORMATTING

Charles H. Jones and Lee S. Gardner
412th TW/TS
Edwards AFB, CA 93524

ABSTRACT

How difficult is it to develop a pulse code modulation (PCM) stream data format? Specifically, given a size, in bits, and a set of parameter sample rates, how hard is it to find a mapping of the sample rates that fits into the frame size -- if one even exists? Using telemetry trees this paper will show that the number of possible mappings for a given set of parameters and sample rates grows exponentially in terms of the number of parameters. The problem can thus be stated in terms of finding a specific instance, or showing that no such instance exists, among an exponentially large number of potential mappings. Although not proof, this provides strong evidence that the PCM format design problem is NP-complete (meaning it is a member of nondeterministic polynomial space but not a member of deterministic polynomial space). That is, that the problem can not be solved in polynomial time and would take a computer years or centuries to solve relatively small instances of the problem. However, if the problem requirements are relaxed slightly, telemetry trees can be used to reduce the PCM formatting problem to *linear* time in terms of the number of parameters. This paper describes a technique that can provide an optimal and fully packed PCM format.

INTRODUCTION

Consider this common telemetry scenario: Given a set of measurements that need to be collected and transmitted, each with its own sample rate, design a PCM stream data format, also called a data cycle map (DCM), that allows timely collection and transmission of the measurements. In this design process, the size of the DCM can be changed, but the transmitted bit rate is fixed by hardware. How should one approach this problem? Often the first step to designing the DCM is to take the bit rate and, using some methodology, come up with the size of the matrix in terms of the number of words per minor frame and the number of minor frames per major frame, the dimensions of the PCM matrix. Then, the challenge is to fit a given set of sample rates into the matrix. Just how complex a problem is this?

Circumstantial evidence suggests that designing a DCM is not easy - most people or organizations use a manual design process. The few computer-automated DCM design programs enforce very stringent constraints on the user. This paper goes beyond this qualitative evidence by using complexity theory to quantify the difficulty of this problem. Specifically, this paper formalizes the DCM design problem (DDP) and analyzes its complexity to show that it is a *very* hard problem. Also included are a brief review of complexity theory, an analysis of the DDP's complexity, and a discussion of the results. The paper then shows how a reduction of the problem's domain leads to a corresponding reduction in the problem's complexity. The domain reduction studied is one that is almost universally used by DCM designers. By combining this domain reduction with telemetry trees, a quick, simple, and elegant algorithm for designing DCMs results. Since the formalization of the problem is also a theoretical abstraction from the reality of the world of telemetry and instrumentation, the paper goes on to discuss practical difficulties related to developing a DCM.

THE DCM DESIGN PROBLEM (DDP)

Informally, the DDP is: Given a set of parameters, their sample rates and a frame size, fit the samples into the frame. This requires that the samples for each parameter be periodic, i.e., equally spaced in the frame. The equal spacing must include 'around the corner'. That is, since the PCM frame repeats, for two consecutive frames the spacing between the last sample of the first frame and the first sample of the second frame must be the same spacing as between any other contiguous samples. The DDP can be formally posed as an integer tiling problem.

First, we present some definitions:

- A *telemetry frame of order M* is the set of integers $\{0,..., M-1\}$.
- An *nk telemetry tile* is a set of integers of the form $t = \{s, s+k, s+2k, ..., s+nk\}$ where s, k, and n are integers and $nk = M-1$.
- A *telemetry tiling*, $\mathcal{T}$, of a telemetry frame F of order *M* is a set of telemetry tiles such that no two elements of any two tiles are the same integer. Further, all integers of, and only those integers of, F are covered by elements of the tiles. Formally, $\mathcal{T} = \{t_i\}$ such that $t_i \cap t_k = \varnothing, i \neq k$ and $F = \bigcup t_i$ where each t_i is a telemetry tile. A telemetry tile represents a parameter and the elements of a telemetry tile correspond to the samples of a parameter. The condition that $nk = M-1$ guarantees that the samples maintain equal spacing 'around the corner'. A telemetry tiling thus represents a telemetry format for a completely filled telemetry frame.

Formally, the *DCM Design Problem (DDP)* can be stated:

> Given a set of sample rates $\{r_i\}$ and a telemetry frame of order M, find a tiling $\{t_i\}$ such that, for each i, t_i is an $r_i k_i$ telemetry tile with $k_i = M / r_i$ if such a tiling exists.

To be completely formal, a distinction between *partial* tilings and *complete* tilings needs to be made. If the total number of samples does not equal the frame size then a partial tiling is acceptable. The analysis will be limited to complete tilings since they represent worst case scenarios.

A REVIEW OF COMPLEXITY THEORY

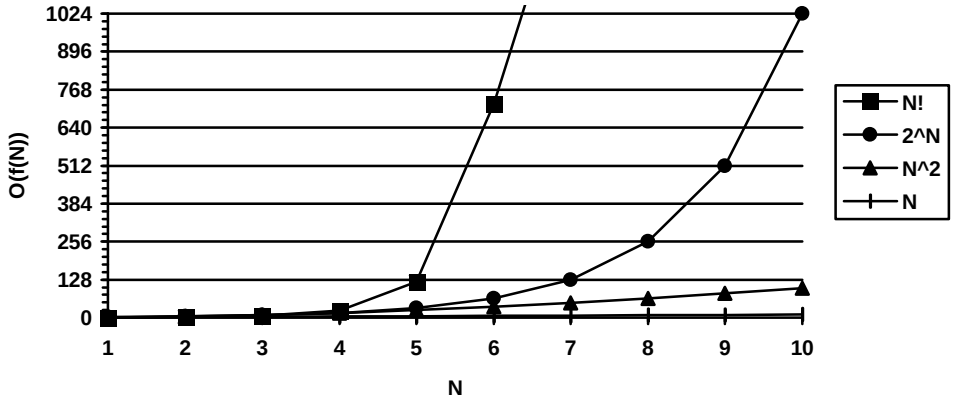
When analyzing the complexity of an algorithm, the number of steps (or computer instructions) it takes to execute the algorithm is counted. For example, in this nested loop:

```
FOR I =1 TO N DO BEGIN
        FOR J = 1 TO N DO BEGIN
                (DO SOMETHING)
        END FOR
END FOR
```

there are N^2 iterations performed. Let us say that it takes C instructions to '(DO SOMETHING)'. Then the number of instructions to implement the nested loop is CN^2.

When discussing relative complexities of algorithms 'Big O' notation is used to indicate the *order* of complexity. For example, the above nested loop is $O(N^2)$ (read either as 'Big O of N^2' or 'of order N^2'). Note that the constant C is not considered. When talking about the order of complexity of an algorithm, multiplication by a constant is not included in the Big O notation.

What is the significance of the order of complexity? Consider the relative graphs for different equations:



If an algorithm is $O(N)$ (or linear), then even for very large N the number of operations grows proportionately with *N*. However, if an algorithm is *N!* then for even small N, say 100, the number of operations can be astronomical.

Realizing that a given computer takes a certain amount of time to execute an instruction, one can see that the order of complexity helps determine how long it will take for a computer to solve a problem. There is a well-known dividing point where computers can be expected to handle an algorithm of a certain complexity. If an algorithm has polynomial complexity $O(N^K)$ for some constant K, then a computer can be expected to implement the algorithm in a reasonable amount of time. However, if an algorithm is exponential or greater ($\geq O(K^N)$ for some constant K) then it may take a computer years or centuries to complete the algorithm for relatively small N. This difference essentially defines the distinction between (deterministically) polynomial time algorithms and NP-Complete (nondeterministically polynomial) algorithms. (See [1] or most any college text on discrete mathematics or automata for further information on complexity and a formal definition of NP-Complete.)

In order to meaningfully discuss the complexity of an algorithm, the N must relate to something. For the DDP, the N will represent the number of parameters.

THE BRUTE FORCE APPROACH

Consider trying to place a set of telemetry tiles into a telemetry frame. The brute force method of doing so is to place the first tile starting in the first slot, place the second tile starting in the second slot, see if the tiles overlap, i.e., they cover any of the same integers. If they do not overlap then place the third tile starting in the third slot, otherwise, backtrack and start over by placing the first tile starting in the second slot, etc. If this technique is used, how many tilings might have to be tried before exhausting all possibilities? Here is an estimate of the Big O of this algorithm.

Assume there are P parameters or tiles. Each tile's placement is uniquely determined by any one element of the tile. The number of tilings to be tried can be determined by considering placing each tile in one of P slots. There are P choices for the first slot; $P-1$ choices for the second slot; $P-2$ choices for the third slot; etc. This gives $P*(P-1)...*2*1$ possible tilings. In other words, the number of tilings to try is $O(P!)$ - an order of factorial. This is even more complex than exponential!

A better approach is to note that the number of samples in a tile limits the positions in which a particular tile can be put. For example if the frame size is M and a tile has a sample rate of s then the tile can only be put in the frame with the first sample in one of the first M/s positions of the frame. Using this limitation a smaller set of arrangements to try can be calculated. Let M be the frame size, $\{s_i\}_{i=1,...,S}$ be the different sample rates in descending order, and $n(s_i)$ be the number of parameters (tiles) with sample rate s_i. Recall that the binomial coefficient $C(n,k) = n!/k!(n-k)!$ is the number of ways of putting k identical objects into n slots. Then the number of tilings to try is:

$$\prod_{i=1}^{S} C(\frac{M}{s_i} - \sum_{j=1}^{i-1} \frac{M}{s_j}, n(s_i))$$

For each sample rate, in order, there are the number of places to put the first sample minus the tiles already placed, choose the number of tiles with that sample rate. In general, $C(n,k) = O(n^{\min(k,n-k)})$. Thus, roughly speaking, this is $O(M^{\sum n(s_i)}) = O(M^P)$ where $\sum n(s_i) = P$ is the number of parameters (tiles). This reduces the complexity to exponential rather than factorial.

Perhaps there is a way of looking at only the valid tilings (those that have no intersecting tiles) instead of every possible tiling. How many valid tilings are there? Telemetry trees can be used to estimate this.

TELEMETRY TREES

A *Telemetry Tree* [2,3] is a tree that has nodes labeled with integers and meets the following criteria:

1. Each sibling of a parent node has the same label.
2. The sum of the labels of siblings equals the label of the parent.

If the top node of a Telemetry Tree is labeled M then the Telemetry Tree *is of order* M. The trees in Figure 1 are telemetry trees.

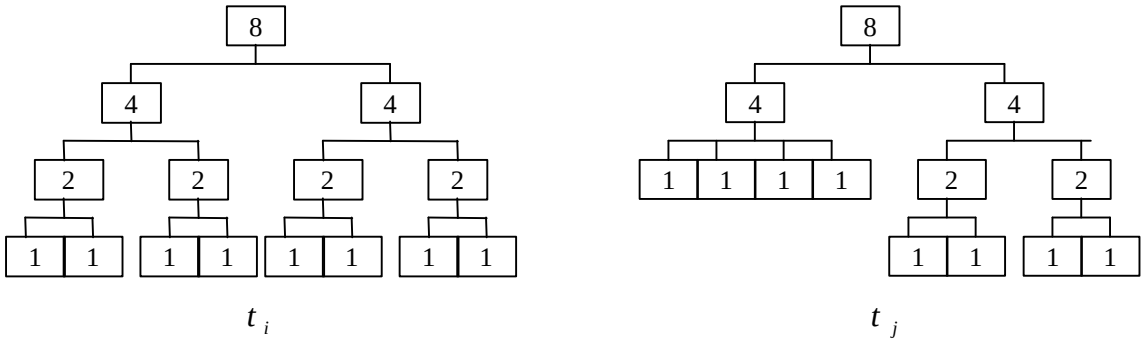


Figure 1: Some Telemetry Trees of Order 8.

Through a simple algorithm described in [2] it is possible to associate a telemetry tiling (and therefore a DCM) with a subset of nodes of a telemetry tree. This association provides a method of analyzing the number of tilings associated with a telemetry tree. For the current analysis it is enough to know that any subtree of a telemetry tree is also a telemetry tree. The analysis is simplified by using *maximal* telemetry trees. A telemetry tree is maximal if, for all child nodes, dividing the label of the parent by the label of the child produces a prime. In Figure 1, t_i is maximal, whereas t_j is not.

Let $M = \prod p_i$ be the size of a telemetry tree J where the p_i are the (not necessarily unique) prime factors of M. Let $N(M)$ be the number of mappings for J. If there are p_j children of the top node of J, then the size of the children of the top node is M / p_j and

$$N(M) = 1 + \left(N\left(\frac{M}{p_j} \right) \right)^{p_j}$$

The 1 is derived from choosing the top node as the only tile in the mapping. Otherwise any of the ways of mapping out each child can be chosen. This gives rise to the exponential term since there are p_j such children. Since each level of the tree reduces the node size by some p_i:

$$N(M) = 1 + \left(N\left(\frac{M}{p_1} \right) \right)^{p_1} \geq N\left(\frac{M}{p_1} \right)^{p_1} \geq \left(N\left(\frac{M}{p_1 p_2} \right)^{p_1} \right)^{p_2} \geq \ldots \geq p_m^{\,p_1 p_2 \cdots p_{m-1}} = p_m^{\,\frac{M}{p_m}}.$$

In other words, the number of mappings for a given tree of order M is roughly $O(p^M)$ - i.e., exponential.

The next question is how many trees of order M are there? Computer calculations do not indicate an exponential number of trees. However, using those calculations the authors conjecture that the supremum of the number of trees approaches exponential as the frame size approaches infinity. This leads to there being, in the worst case, a very large number of valid tilings to consider.

CONSEQUENCES

The formulas just derived give quantifiable, yet still circumstantial, evidence that the DDP is NP-Complete; they do not prove it. To prove a problem is NP-Complete, it is necessary to reduce a problem that is known to be NP-Complete to the problem at hand in polynomial time. That is, show that a solution to the problem leads to a solution of the NP-Complete problem with no more than polynomial extra steps. Although the authors have not proven that the form of the DDP described above is NP-Complete, there is a form that can be proven to be so. A formal proof is outside the scope of this paper, but an outline of the proof is presented here.

The exact cover problem (ECP) is known to be NP-Complete [1]. This problem is: Given a collection of sets all being subsets of some set F, is there a subcollection whose union is F such that each pair of sets in the subcollection is disjoint. The master set in our setting is the telemetry frame. The collection of subsets is the set of tiles potentially representing the parameters of certain sample rates. One reason the ECP is not equivalent to the stated form of the DDP, is that not every possible subset of the integers represents a telemetry tile as they have been defined. Telemetry tiles have been defined to have a very periodic nature. If the DDP is generalized to allow aperiodic tiles, then the ECP is nearly equivalent to the DDP. This generalization is not unreasonable since the Class II PCM formats of the IRIG 106 Standard [4] allow for aperiodic sample transmissions. Another reason the ECP is not exactly equivalent to the DDP is that the ECP will accept any exact covering. In contrast, the DDP requires a covering with particular properties - there must be exactly so many tiles each of a certain size (i.e., sample rate). Thus, the generalized DDP is slightly more difficult than the ECP. Considering all this, the authors conjecture that the DDP, as originally stated, is NP-Complete.

The consequence of all this is that, if a computerized solution to the DDP is to be developed, a significant simplification of the problem as stated needs to be made. If such a change is not made, no computer will be able to quickly solve the problem for arbitrary cases. One such change was described in [2,3]. Another, simpler, change is described below.

A DOMAIN REDUCTION SOLUTION

An almost universal domain reduction of the DDP is what the authors call the ‘power of 2 rule’. Instead of allowing arbitrary sample rates, sample rates are restricted to powers of 2 (e.g. 1, 2, 4, 8, ...). It can be shown that there is only one (maximal) telemetry tree of order 2^N for all integers N; in contrast to the nearly exponential number of telemetry trees conjectured above for an arbitrary frame size.

Thus, if the frame size and all the sample rates are powers of 2, then that single tree can be used to assign parameters to nodes of the tree. The only complication arises from the algorithm in [2] which associates nodes with parameters. That algorithm states that once an association between a node and a parameter is made, no ancestor or descendant of that node can be associated with any other parameter. So, after sorting the sample rates in descending order, the node assignments are made left to right and top to bottom. There is no backtracking necessary. For example, consider the tree t_i in Figure 1 and assume the sample rates are 4, 2, 1, 1. First use the left most 4. This eliminates the left subtree. Then use the left most 2 in the remaining subtree. This eliminates the left three node subtree of the right subtree. This leaves the use of the two right most leaves for the 1's. This provides an algorithm of $O(P)$ where P is the number of parameters. That is, a solution to the problem which is of linear order.

As stated so far, the algorithm requires using a frame size which is a power of 2. A simple modification gives some flexibility in frame sizes. Use the highest sample rate as the node size for the first level of the tree (below the top node). Then the same algorithm can be applied to any tree of an order which is a multiple of this sample rate. For example, if the highest sample rate is 16 then this algorithm can be applied to frame sizes of 16, 32, 64, 96, ...

INSTRUMENTATION CONSTRAINTS

The algorithm just described does not take into consideration the limitations imposed by actual instrumentation systems. The following is a first attempt at categorizing these constraints. For each of these, it is worth noting whether they are constraints on the DCM formatting algorithm itself or constraints on the *inputs* to the algorithm. There are certainly more constraints than are listed here.

Sample Rate Limits: Hardware limits and the limits of physics may constrain how often a parameter can be sampled. This is a prime example of a constraint on the inputs to the formatting algorithm.

1. Controller Speed Limits - All controllers have maximum sample request/receive rates.
2. Bit Rate Limits - All actual instruments have a maximum bit rate.
3. Data Size Limit - The next sample must wait until the current sample is sent.
4. Settling Time - Some sensors take time to "settle" before sampling can proceed.

Bus Polling: Many systems use several buses to talk to different arrays of sensors. Typically only one bus can be polled at a time. As such, each bus is polled periodically with their samples interleaved into the telemetry stream. In terms of telemetry trees, this is

the equivalent of assigning a subtree to each bus. Thus, multiple buses are really multiple applications of the formatting algorithm.

Frame Size Constraints: Most systems constrain the frame size. The frame size can also be constrained by limits on the bit rate. This is another limit on the inputs to the algorithm which should be identified before the algorithm is executed.

Subframe Constraints: Some systems limit the number or size of the subframes within a major frame. The two telemetry tiling algorithms presented to date do not use subframes at all. Subframes are used mostly to provide periodic synchs - which are essentially confidence factors. The algorithms presented view these as just another parameter. As such, this is one type of constraint that would have to be incorporated into the algorithms presented.

Fixed Word Sizes: Most systems use a fixed word size; usually 10, 12, or 16 bits. This causes two problems: wasted bits due to samples that are smaller than the fixed word size; and split data words due to samples that are larger than the fixed word size. This may very well be one of the more perplexing issues related to DCM design. One approach is to simply ignore the issue and use software to decommutate data across word boundaries. However, this may require onboard hardware changes in order to produce a data stream where samples cross word boundaries as well as special decommutation software. So, most likely, this issue would have to be incorporated into the formatting algorithm.

SUMMARY

At least some forms of the DCM design problem are quantifiably very complex and brute force methods for solving the standard DDP are not an acceptable basis for automation efforts. However, the benefits of automating this process appear to be significant - both in terms of reduced manpower to support telemetry and instrumentation functions and in terms of more efficient PCM mappings. Further pressure to obtain these benefits comes from budget constraints and continuing, foreseeable reallocations of the telemetry spectrum. To fully realize efficient mappings that can be designed algorithmically will require a simplification of the DCM design problem. This, in turn, will require a paradigm shift in the telemetry world. It is possible this paradigm shift will require the development and re-instrumenting of telemetry systems with more flexible hardware. The question becomes whether or not the factors pressing for more efficient automated mappings are sufficient to force such a change.

BIBLIOGRAPHY

[1] Hopcroft, John E. and Ullman, Jeffrey D., Introduction to Automata Theory, Languages, and Computation. Addison-Wesley Publishing Company, Reading, MA, (1979).

[2] Jones, Charles H. And Gardner, Lee S., "Automated Generation of Telemetry Formats," in Proceedings International Telemetering Conference, vol. XXXII, (1996), pp. 635-644.

[3] Jones, Charles H. And Gardner, Lee S., "Telemetry Mapping Algorithm Project (TMAP)", International Test & Evaluation Association Workshop, Lancaster, CA, http://www.edwards.af.mil/itea/papers/paper09.doc, (April 1997).

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