Compressed representation of a partially defined integer function over multiple arguments

Nina Daskalova, Plamen Mateev, Stela Zhelezova

1. The problem

In OLAP (OnLine Analitical Processing) data are analysed in an n-dimensional cube. The cube may be represented as a partially defined function over n arguments:

$$f(x_1, x_2, \ldots, x_n), \text{ where } x_i \in [1 \div N_i] \text{ for } i \in [1 \div n].$$

Considering that often the function f is not defined everywhere, is there a known way of representing f or the points, in which it is defined, in a more compact manner than the trivial one

$$x_1, x_2, \ldots, x_n, \quad f(x_1, x_2, \ldots, x_n)?$$

The goal is to reduce the time necessary for moving or storing f and using part of the time gained for computations for restoring the original f.

2. The team propositions

We consider the example data for this problem and observe that some argument combinations may be missing because the function is partially defined, but no one combination can repeat. So a combination of function arguments for some particular value is unique (Table1). Our idea is to replace the whole long list of function arguments with a single unique number (ID), which depends on this list. Then the function will be defined in the following way: (ID, f) instead of the trivial one.

The criteria to choose ID are:

- easy calculated from the argument list;
- uniqueness;
- easy to calculate back the argument list from the ID.

We offer two methods to assign ID to each different argument list.

ID	x_1	x_2	x_3	$f(x_1, x_2, x_3)$
	1	2	1	5
	2	2	1	2
	2	2	2	7
	4	2	2	1
	4	3	1	6

Table 1: $f(x_1, x_2, x_3), x_1 \in [1 \div 4], x_2 \in [1 \div 3], x_3 \in [1 \div 2]$.

2.1. ID is a consecutive number of the corresponding argument list

An argument list is a n-tuple over some alphabet (from 1 to $max(N_1, \ldots, N_n)$) so we can order the different n-tuples lexicographically. The number of all possible combinations of function arguments is $N_1 N_2 \ \ldots \ N_n$. Then we use the consecutive number of the argument list in this order as ID.

For our example in Table 1 all possible triples are: $(1,1,1)$; $(1,1,2)$; $(1,2,1)$; $(1,2,2)$; $(1,3,1); \ldots (4,3,2)$. Then instead of presenting f as $(1,2,1,5)$ we use only the couple $(3,5)$ (the argument list $1,2,1$ is number 3 in the accepted lexicographic order).

There are two possibilities to find an ID, defined above.

First, if the firm works with a client many times, it is possible to keep at both applications (for the firm and for the client) all argument lists in lexicographic order and their corresponding numbers.

If the clients are quite different each time and argument lists also change dynamically, such storage is ineffective. In this case the ID number for each argument list can be obtained by:

$$ID = N_2 N_3 \ldots N_n (x_1 - 1) + N_3 N_4 \ldots N_n (x_2 - 1) + \\ \ldots + N_{n-1} N_n (x_{n-2} - 1) + N_n (x_{n-1} - 1) + x_n.$$

To reverse this formula the next procedure could be used:

```
x_n = ID mod N_n; ID = ID div N_n
for  k = (n-1) to 2 do
x_k = ID mod N_k + 1;  ID = ID div N_k
```

2.2. ID is a number in some numeral system (for example binary), different from the decimal one

Now we use $(1111001, 5)$ to present f instead of $(1,2,1,5)$, because $121_{10} = 1111001_2$. Remember that 1 byte = 8 bits and a number in decimal system is 1

byte while 8 positions in binary system are 1 byte. In our concrete example we obtain $3 \times 1 = 3$ byte vs. 1 byte.

Let $(*, *, \ldots, a_i, *, \ldots, *)$ denote an $(n-1)$-dimensional cube. The most frequently solved problem in OLAP is the computation of the sum of values F of a given sub-cube of the n-dimensional cube. In the case of the $(n-1)$-dimensional cube the values F are:

$F(*, *, \ldots, a_i, *, \ldots, *) =$

$\sum_{j_1}^{N_1} \sum_{j_2}^{N_2} \cdots \sum_{j_{i-1}}^{N_{i-1}} \sum_{j_{i+1}}^{N_{i+1}} \cdots \sum_{j_n}^{N_n} f(j_1, j_2, \ldots, j_{i-1}, a_i, j_{i+1}, \ldots, j_n)$

To compute the values of all sub-cubes

$$F(*, x_2, \ldots, x_n), \ldots, F(*, *, x_3, \ldots, x_n), \ldots, F(*, *, \ldots, *)$$

an extra symbol is added to the alphabet of each argument (Table 2). The argument lists are extended but the property, which we use to add an ID, is the same. Therefore our idea can be extended also on the new argument lists.

ID	x_1	x_2	x_3	$f(x_1, x_2, x_3)$
	1	2	1	5
	2	2	1	2
	2	2	2	7
	4	2	2	1
	4	3	1	6
	*	2	1	7
	*	2	2	8
	...	...	...	...
	4	*	*	7
	*	*	*	21

Table 2: $f(x_1, x_2, x_3), x_1 \in [1 \div 4], x_2 \in [1 \div 3], x_3 \in [1 \div 2]$

In addition our group suggests the firm to archive the final records trying another software.

The document compression standard DjVu [1] uses an algorithm called arithmetic coding. Arithmetic coding, invented by Jorma Rissanen, and turned into a practical method by Witten, Neal, and Cleary, achieves a superior compression. Free browser plug-ins and desktop viewers from different developers are available from the djvu.org website.

7z is a compressed archive file format with LZMA compression. Compression ratio results are very dependent upon the data used for the tests. Usually, 7-Zip compresses to 7z format 30-70% better than to zip format. And 7-Zip compresses to zip format 2-10% better than the most of other zip compatible programs [2].

References

[1] Léon Bottou, Patrick Haffner, Paul G. Howard, Patrice Simard, Yoshua Bengio and Yann Le Cun, High Quality Document Image Compression with DjVu, Journal of Electronic Imaging, 7(3): 410–425(1998).

[2] `http://www.7-zip.org/`