Improving the performance of the algorithm for distributing numerics by ranges

Dmitry Strabykin, Vasily Meltsov* and Alexander Krutikov

1 VyatSU, Department of Electronic Computers, Moskovskaya st., 36, 610000 Kirov, Russia

Abstract. The paper considers an accelerated algorithm for distributing numbers over given ranges. The proposed algorithm allows you to quickly distribute integers over predetermined ranges with deterministic boundaries and calculate the number of elements in each of the ranges. If the number of ranges lies in the range from 64 to 256, then the time for solving the problem will be reduced by 9–33 times. The algorithm can be implemented in both software and hardware.

1 Introduction

The proposed algorithm can perform a fast distribution of integers over predefined ranges with deterministic boundaries and count the number of elements in each range. The algorithm can quickly solve data analysis problems involving some distribution or grouping of integers. It can also be used for dataset preprocessing when analyzing big data or for creating training samples. The algorithm can be realized by both software and hardware.

The formal formulation of the problem is as follows. Let there is a set $X$ formed of $N$-bit binary integers $X={x_1,x_2,...,x_n,...,x_N}$ and $D$ ranges of these numerics. It is necessary to distribute the numerics of the set $X$ over the ranges, defining a family of sets $W={w_0,w_1,...,w_j,...,w_{D-1}}$, in which $w_j$ is a subset of the numerics in the $j$-th range ($j=0,1,...,D-1$), and calculate the number of numerics $r_j$ in each range: $R={r_0,r_1,...,r_j,...,r_{D-1}}$.

Thus $w_j=\emptyset$ if $r_j=0$ (no numerics in the given range), otherwise $w_j=\{X_j\}$, where $X_j=\{x_{j0},x_{j1},...,x_{js},...,x_{jS}\}$ is a subset of $S=r_j$ numerics belonging to range $j$.

An applied example is the task of counting pixels of a hue of a certain depth, when the range is a certain hue (a range of shades of a color), and the numeric is an integer corresponding to the color code of a pixel in some encoding, for example, RGB (A). This task is extremely relevant, for example, to analyze the cost of ink when printing on plotters and printers for setting a price tag to a customer. Data analysis, in this case, consists in processing the image obtained during recognition.

The proposed algorithm can also effectively solve the classical classification problem, when it is necessary to determine which previously defined class a pixel belongs to. In this case, the color/hue of a pixel can be considered as a data known part for the classification procedure implementation. If hue acts as the main parameter for classification.
classification, then the issue of each class range boundaries determining must be resolved in advance.

It is assumed that sets of numeric values are represented by arrays in the process of solving the problem. If the boundaries of the ranges are predetermined, the simplest method for solving the problem of determining the range (class) to which a numeric from the source array belongs is an algorithm that involves storing the boundary values of the ranges and sequentially comparing the analyzed numeric with them.

It is assumed that the ranges are pre-sorted and the comparison is made starting from the lower range (from the smallest value of the lower limit) in ascending order. Range numbering starts from one. The algorithm (for all array numerics) in this case is as follows:

1. Accept $N=1$.
2. Read the next numeric from the input array.
3. Compare the numeric with the upper limit of the $N$-th range.
4. If the boundary is greater than the value of the numeric, then increase the contents of the range counter by 1, take $N=1$, go to step 2.
   Otherwise, go to step 5.
5. Increase $N$ by one, go to step 3.

The algorithm is simple in spite of two nested loops: one on the array of incoming numerics, the other on all specified ranges. At the beginning, the algorithm may seem difficult to understand and has high computational complexity. The second part is partially true, because the greatest time of algorithm execution occurs when realizing the so-called limiting case, when the number of ranges is equal to the number of different values of numerics in the array. Then the digit capacity of the array address is equal to the digit capacity of the array numerics, and the number of ranges coincides with the number of different numerics in the array. In this case, according to the number of ranges, the number of memory accesses increases when reading the range boundary values. Reducing the execution time of the described cyclic counting procedure is possible by reducing the number of memory accesses, on which the accelerated algorithm proposed in this paper is based. This algorithm was largely inspired by metric classification algorithms and, to some approximation, works in a similar way. It also shares common features with genetic algorithms.

2 Accelerated algorithm for distributing numerics

The distribution of numerics over ranges is based on a simple analytical relationship between the numeric and the address of the memory cell corresponding to a given range of numerics. Before calculations must be given:

- $k$ — number of bits for recording the maximum numeric.
- $D$ — number of ranges.
- $L$ — length (dimension) of the range.
- $m$ — special parameter.

The number of ranges can be calculated using the formula $D = 2^m$. Ranges have numbers from 0 to $2^m - 1$. The parameter $m$ represents the digit capacity of the range address ($m=0,1,2,...,k$). The range length is the number of different numerics in the range $L = 2^{k-m}$.

The range length has a maximum value $L_{\text{max}} = 2^k$ with one zero range ($m=0, D=1$) and a minimum value $L_{\text{min}} = 1$—if you form $2^k$ ranges ($m=k, D=2^k$). Memory cells from zero to $2^m - 1$ inclusive, in each range, are divided into two fields (possibly, they are located in different memory blocks corresponding to the allocated fields). The first field of the cell is used to count numbers, and the second field is used to set the starting address of the generated array of numerics related to the range that the cell corresponds to.
The processed array of numeric is stored in memory after the cells allocated for counting numbers. The starting addresses of the formed arrays of numeric (when the initial numeric are distributed over the ranges) are allocated taking into account the fact that all the initial numeric may be in the same range. The block diagram of the proposed algorithm is shown in Fig. 1.

Fig. 1. Scheme of the algorithm for distributing numbers over ranges.

In Fig. 1 the following designations are used:

• $M[j] (M[i], M[A])$ - the content of the memory cell with address $j$ ($i, A$).
• $M[j]<r_j>$ - the content of the field $r_j$ of the memory cell with address $j$ for counting numbers in the $j$-th range.
• $M[j]<a_j>$ - the content of the field $a_j$ of the memory cell with address $j$ for setting the starting address of the formed array of numbers in the $j$-th range;
• $x_i$ - $i$-th numeric of the initial array of numeric.

Begin

\[ i := 1; \]
\[ M[j]<r_j> := 0; \]
\[ M[j]<a_j> := A_j; \]
\[ j = 0, \ldots, (D - 1) \]
\[ x_i := M[i] \]
\[ j := x_i / 2 \]
\[ k := m \]
\[ r := M[j]<r_j> \]
\[ a := M[j]<a_j> \]
\[ A := x_i \]
\[ r := r + 1 \]
\[ M[j] = <r> \leftrightarrow <a> \]
\[ A := a + r \]

Yes

\[ i := i + 1 \]

No

End
Before the beginning of numeric distribution on ranges in each cell, in the field for counting of numbers, zero is written. In the starting address field of the formed array of numerics the smallest address of the corresponding range is written. After reading from memory the numeric $x_i$ stored in the initial array of numerics $X$, the address of the memory cell of the range corresponding to the numeric $x_i$ is calculated. For this, the value $j = x_i / (k - m)$ is determined and the binary code of the numeric $x_i$ is shifted by $(k - m)$ bits towards the lower bits with the addition of the vacant bits with zeros. At the calculated address is read the contents of the range memory cell, divided into two fields: $r_j$ for counting numbers and $a_j$ to set the starting address of the array of numerics in the $j$-th range. In the array of the $j$-th range, the address for the received numeric is determined by adding to the starting address of the formed array the numbers of numerics already placed in this array. The obtained address is used to write the next numeric $x_i$ into the range array. After that, the number of numerics in the array being formed is increased by one. The new value of the number of numerics in the selected range together with the starting address of the formed array is written to the memory cell of the range. The described actions are repeated for all numerics of the initial array $X$.

Consider an example (Table 1), illustrating the distribution of numerics over ranges for the case when the maximum numeric bits is $k = 7$, and the number of ranges is $D = 2^3 = 8$.

<table>
<thead>
<tr>
<th>Analyzed numeric ($x_i$)</th>
<th>Range cell address ($j$)</th>
<th>Range cell content</th>
<th>Recording address of the analyzed numeric ($A$)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Numeric count field ($r_j$)</td>
<td>Array address field ($a_j$)</td>
</tr>
<tr>
<td>01101011</td>
<td>00000110</td>
<td>00000000</td>
<td>11100000</td>
</tr>
<tr>
<td>01100111</td>
<td>00000110</td>
<td>00000001</td>
<td>11100000</td>
</tr>
<tr>
<td>00110100</td>
<td>00000011</td>
<td>00000000</td>
<td>10110000</td>
</tr>
<tr>
<td>01100101</td>
<td>00000110</td>
<td>00000010</td>
<td>11100000</td>
</tr>
<tr>
<td>01100000</td>
<td>00000110</td>
<td>00000111</td>
<td>11100000</td>
</tr>
<tr>
<td>00110101</td>
<td>00000011</td>
<td>00000001</td>
<td>10110000</td>
</tr>
<tr>
<td>01110000</td>
<td>00000111</td>
<td>00000000</td>
<td>11100000</td>
</tr>
<tr>
<td>01101111</td>
<td>00000111</td>
<td>00000100</td>
<td>11100000</td>
</tr>
</tbody>
</table>

In the example in the first line of the table to convert the analyzed number $x_1$ to the address of the cell of the range $j$, the binary code of the numeric ($01101011$) is shifted $is$ shifted by $4$ digits ($k - m = 7 - 3 = 4$) towards the lower digits with filling the vacated digits with zeros ($00000110$). Using the obtained address, the contents of the selected range cell consisting of two fields $r_j$ and $a_j$ are read. The address of the analyzed numeric $x_1$ record ($A$) is formed by adding the contents of the number count field ($r_j$) to the contents of the initial address of the formed array field ($a_j$): $A = a_j + r_j (11100000 + 00000000 = 11100000)$. At this address, the analyzed numeric $x_1$ is written to the selected range. The content of the number counting field $r_j$ is incremented by one and written to the address of the selected range.
3 Estimation of computation time reduction

The main time required to assign the analyzed numeric to one of the specified ranges is taken up by memory accesses. Given the large size of arrays, when counting billions of records and, most importantly, millions of ranges, cache-memory and other fast memory cannot accommodate all values, especially range counters, which can be pushed to external memory such as a hard disk.

In the proposed accelerated algorithm, four such accesses are required: reading the analyzed numeric, reading the contents of the range cell, writing the analyzed numeric to the array to be formed, and writing the incremented by one numeric number of this array to the range cell. The time of solving the problem for N initial numerics can be estimated by the formula:

\[ T1 = 4 \times N \times t_{\text{mem}}, \]  

(1)

where \( t_{\text{mem}} \) is the memory access time.

When using an algorithm based on storing the boundary values of the ranges and a sequential comparison of the analyzed number to determine the range (in the presence of D of the ranges), on average, D/2 of such comparisons will be required. Then it will take \((4+D/2)\) memory accesses to distribute numerics into ranges: reading the analyzed numeric, reading the upper value of the first range, reading the upper value of the second range, and so on (until the analyzed numeric is less than the upper value of the corresponding range), reading the cell of the corresponding range, writing the analyzed numeric into the formed array and writing the number of numerics of this array increased by one into the cell of the range. In this case, the average time of solving the problem for N initial numerics can be estimated by the formula:

\[ T2 = (4+D/2) \times N \times t_{\text{mem}} \]  

(2)

The reduction in the time of distribution of numerics into ranges, achieved by applying the proposed algorithm, is defined as follows:

\[ V = \frac{T2}{T1} = 1 + \frac{D}{8} \]  

(3)

Since the computation time \( T1 \) in the proposed algorithm is independent of the number of ranges \( D \), the time gain \( V \) increases linearly as the number of ranges increases.

If the number of ranges is between 64 and 256, the problem solving time will be reduced by 9-33 times (Fig. 2).

4 Conclusion

As you can see, a decrease in memory appeals gives a positive effect of acceleration of the algorithm. The maximum acceleration effect is achieved in a particular case, when the number of ranges is equal to the number of different values of numbers in the original array, and the number of discharges in all numbers is as maximum and equally the memory cells number discharge.

In case of significant digit capacity of the analyzed numerics (for example, 16 digits) the reduction of calculation time in this case will be more than 10,000 times.

The algorithm can be used in real-time systems, where it is necessary to process streams of numerics, correcting the results of calculations as new data arrives.
Fig. 2. Dependence of the reduction (V) of the calculation time on the number of ranges (D).

Here are just some of the subject areas in which the application of the algorithm will be relevant and extremely effective:

• For preprocessing of data arrays in Data Mining tasks [12].
• Before training structural classification/clustering models, including image processing or before mathematical processing using a statistical model [13].
• In real-time systems including real-time operating systems (RTOS). Hard real-time tasks are associated with the struggle for even minimal time improvements to meet the time limit allocated to the task, after which the result becomes irrelevant and the RTOS terminates the task as no longer valuable [14].

One of the most important areas of this algorithm application is the work of the air traffic control service. Radar object signatures, the so-called "fingerprints" are encoded with a string of data representing information about the object in the form of a certain notation. Also to this signature string is added a board identifier, if it is represented by a transponder or otherwise defined by radar [15, 16].

If the signature symbols are represented as ANSI codes and supplemented with obligatory onboard identifiers, then it is possible, by dividing them into ranges, to quickly and accurately analyze how many, from where and which aircraft are in the monitored space [17].

References

3. S. M. Dol, P. M. Jawandhiya, Use of Data mining Tools in Educational Data Mining, in Proceedings of the Fifth International Conference on Computational Intelligence and Communication Technologies, CCICT, Sonepat, India (2022)


8. J. Liu, A Multi-classification Model of Industry Based on Multi-task Learning, in Proceedings of the 3rd International Conference on Computer Science and Management Technology, ICCSMT, Shanghai, China (2022)


17. V. Y. Meltsov, A. A. Lapitsky, Simulation of UAV Intelligent Control System Based on Precedent Knowledge Base, in Proceedings of the Conference of Russian Young Researchers in Electrical and Electronic Engineering, ElConRus, Saint Petersburg, Russian Federation (2022)