Available online at www.ijpe-online.com

International Journal of Performability Engineering

vol. 15, no. 3, March 2019, pp. 1045-1052
DOI: 10.23940/ijpe.19.03.p34.10451052

Fast AIS Data Decoding Algorithm for Multi-Core CPU

Xiangkun Zeng[a,*], Huaran Yan[b], Yingjie Xiao[b] and Xiaoming Yang[c]

[a]*College of Information Engineering, Shanghai Maritime University, Shanghai, 201306, China*
[b]*Merchant Marine College, Shanghai Maritime University, Shanghai, 201306, China*
[c]*Shanghai Offshore Engineering Research Institute, Shanghai Maritime University, Shanghai, 201306, China*

Abstract

According to the provisions of the SOLAS convention, more than 600,000 ships in the world have installed automatic identification systems, which can produce more than 3 billion lines of AIS message data every day. The large amount of AIS message data provides many applications. At the same time, the design of efficient decoding of AIS messages has become a new problem. The former AIS message decoding methods mostly use online real-time decoding and decode the messages line by line according to the order of messages. These methods are slow and inefficient when decoding large amounts of AIS messages, and they cannot use multi-core CPU computing power to the extreme. In this paper, the decoding process is decomposed into several steps such as data reading, single-line message filtering, multi-line message filtering, single-line message binary transcoding, multi-line message binary transcoding, binary code decoding, and preservation results. Because multi-line message filtering and multi-line message binary transcoding are difficult to perform in parallel, these two steps are performed in serial, while the other steps are run in parallel. For binary code decoding, which consumes the largest computing resources in the process, the data is blocked first, and then decoding tasks are created and run in parallel. This method makes full use of the parallel computing capability of multi-core CPU and achieves high speed in the running test.

Keywords: big AIS data; decode; multi-core CPU; parallel

(Submitted on November 12, 2018; Revised on December 10, 2018; Accepted on January 8, 2019)

1. Introduction

According to the provisions of the SOLAS convention, all ships of 300 gross tonnage and upwards engaged on international voyages and cargo ships of 500 gross tonnage and upwards not engaged on international voyages and passenger ships irrespective of size shall be fitted with an Automatic Identification System (AIS) [1].

So many ships around the world have installed automatic ship identification systems. They can broadcast ship dynamic information such as latitude and longitude, speed and heading, and ship static information such as ship names, call signs, drafts, and cargo from radio waves to nearby waters, shore base stations, and satellites. Broadcasting enables the neighboring ships to keep abreast of the static information of all nearby ships on the sea, and they can immediately call and coordinate with each other. This provides great help for navigational safety and maritime management.

There are more than 600,000 ships installed with AIS around the world, sending out AIS messages continuously. Globally, more than 3 billion AIS messages are generated each day [2]. Massive AIS messages offer a variety of applications through the analysis of big AIS data collected by satellites or shore-based base stations, ship pollutant emissions [3-5], ship encounter situations [6], route planning [7], ship collision risk [8-9], marine traffic patterns, channel capacity analysis [10], etc. The analysis of the massive ship AIS message data can also provide data support for a series of assessment reports such as route capacity and port service capabilities [11-12].

AIS message data is coded according to ITU-R M.1371 and IEC61162 [13-14]. To better use the massive AIS messages data, it is necessary to preprocess the historical data and particularly important to decode AIS message data [15-16]. According to the standard, some types of messages are sent in a single line, and some types of messages are divided into

* Corresponding author.
E-mail address: xiangkun_zeng@vip.163.com

multiple lines and sent in sequence. The decoding of a message requires binary transcoding, splicing, and decoding. Therefore, the decoding of AIS messages is generally performed line by line in chronological order.

2. Data Acquisition and Encoding Rules

2.1. Data Acquisition

The AIS device will send out AIS messages when the ship is operating normally. The data center can collect AIS messages through satellites, shore-based base stations, and other devices and then aggregate them on the AIS message storage server through the network. An AIS message is stored as a single line in data storage and is encoded in ASCII (American Standard Code for Information Interchange). After testing, the storage space occupied by a single data file has an impact on the efficiency of the subsequent decoding work, and it is ideal for a file to be stored with about 2 GB of storage. After the data is acquired, the decoding workstation decodes the stored AIS message data files and stores the results in the AIS decoding results storage server. A schematic diagram of the data acquisition, decoding, and storage process is shown in Figure 1.

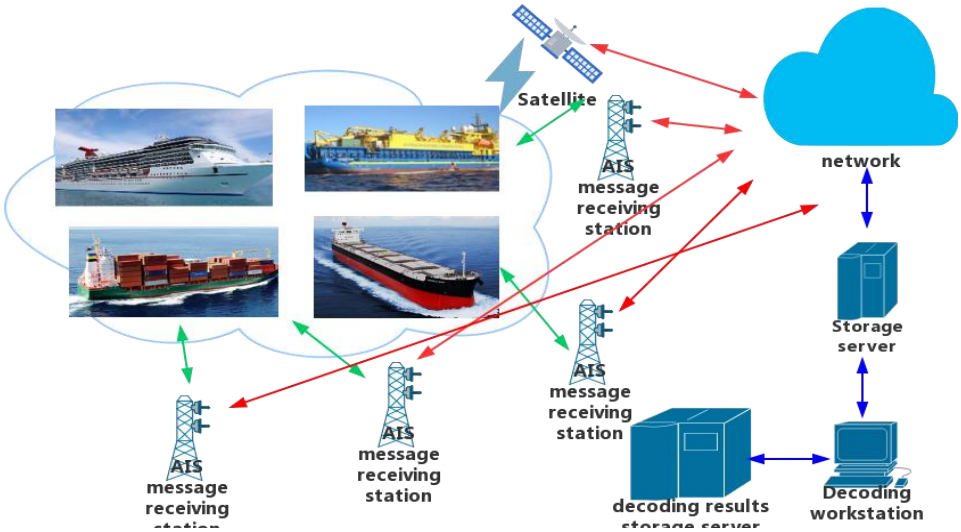

Figure 1. AIS data acquisition, decoding, and saving

2.2. Encoding Rules

Each row of the AIS data file consists of a timestamp field and an AIS message. According to the standard, the AIS message consists of multiple fields. Each field can be a string of valid characters or null characters and is delimited by comma. The definition of the AIS message format is shown in Figure 2.

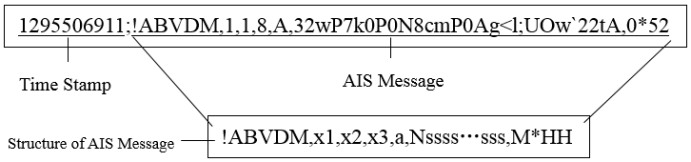

Figure 2. Structure of AIS message

The field with "!ABVDM" indicates the type of statement. The field "X1" indicates how many lines this AIS message has been broken into. If the value of the field "x1" is 1, the message is a single-line message. If the value of the field "x1" is greater than 1, the message is a multi-line message, and the field "x2" represents the sequence number of the message. Field "x3" indicates continuous message identification, and Field "a" indicates AIS channel number. In the field "Nssss…sss", the first position "N" indicates the type of AIS message, and the "ssss…sss" is a string of ASCII characters, each of which represents a 6-bit binary number, as shown in Table 1. The field "M*HH" contains the number of bits filled and the check information.

The common AIS messages are shown in Figure 3. The first three messages are single-line messages, and the last two messages are multi-line messages. The last two messages share a timestamp; when multi-line messages like this are decoded,

they need to be decoded in order.

Table 1. Characters encoding table

Chr1	Binary1	Chr2	Binary2	Chr3	Binary3	Chr4	Binary4	Chr5	Binary5
0	000000	=	001101	J	011010	W	100111	l	110100
1	000001	>	001110	K	011011	`	101000	m	110101
2	000010	?	001111	L	011100	a	101001	n	110110
3	000011	@	010000	M	011101	b	101010	o	110111
4	000100	A	010001	N	011110	c	101011	p	111000
5	000101	B	010010	O	011111	d	101100	q	111001
6	000110	C	010011	P	100000	e	101101	r	111010
7	000111	D	010100	Q	100001	f	101110	s	111011
8	001000	E	010101	R	100010	g	101111	t	111100
9	001001	F	010110	S	100011	h	110000	u	111101
:	001010	G	010111	T	100100	i	110001	v	111110
;	001011	H	011000	U	100101	j	110010	w	111111
<	001100	I	011001	V	100110	k	110011		

```
1295506911;!ABVDM,1,1,8,A,32wP7k0P0N8cmP0Ag<l;UOw`22tA,0*52
1295506911;!ABVDM,1,1,8,B,403t>B1udJ71j`NonbBw?>G00<07,0*2A
1295506911;!ABVDM,1,1,9,B,16:ei@0P0K8UC2JBBn6@jOwD0@l7,0*2B
1295506911;!ABVDM,2,1,1,B,56:F6m0000008mDr221PTp`Tq9Dr22222222221680O<44lf0>RPCQnB,0*35
!ABVDM,2,2,1,B,C`8888888888880,2*0E
```

Figure 3. AIS messages

3. Decoding Algorithm Design

The general AIS message decoding method [17] is mainly oriented to real-time applications, performs the decoding operation while receiving AIS messages, and does not need to consider the need to decode massive AIS messages in a short time. When decoding AIS messages, it only needs to perform message verification, binary transcoding, binary code splicing (if necessary), binary decoding, and saving work line by line after receiving the AIS message, as shown in Figure 4.

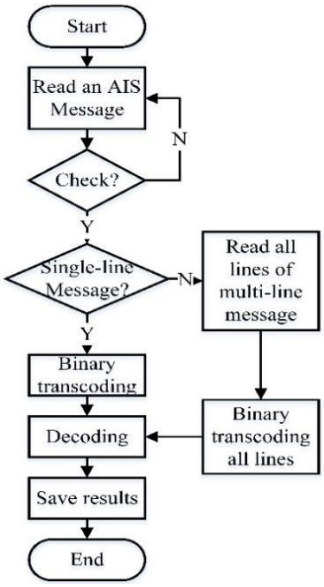

Figure 4. General AIS message decoding process

However, in the context of ship trajectory big data applications, a large amount of AIS messages need to be decoded during data preparation and preprocessing. At this time, the method of decoding AIS messages line by line is inefficient and cannot meet the requirements. Therefore, an efficient offline decoding algorithm for massive AIS messages is designed in this paper. This method can filter AIS messages and select several types of AIS messages for decoding. When the algorithm

is running, it can make full use of the parallel computing capabilities of the computer's multi-core CPU and achieve high decoding efficiency.

3.1. Decoding Workflow

Since a large amount of data has been divided into files with a storage amount of 2GB when the AIS messages file is saved, the algorithm performs decoding according to the workflow shown in Figure 5.

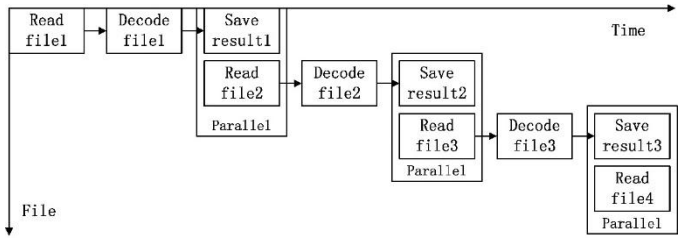

Figure 5. Decoding workflow

After testing, we know that the decoding process of AIS messages takes the longest time and the CPU usage of the computer is the highest. Therefore, we adopt the pipeline operation mode shown in Figure 5 to parallelize the reading of data and the preservation of the results and run the decoding independently.

3.2. Decoding Algorithm

To improve the utilization rate of multi-core CPU as much as possible and guarantee the correctness of decoding at the same time, this text will decompose the process of decoding the message, extract the steps that are difficult to run in parallel, and run them serially instead. The other steps are run in parallel.

First, single-line messages and multiple-line messages are extracted from the AIS messages file. Since multi-line messages need to be decoded while maintaining the original order, single-line messages do not need to be used. The operation of extracting single-line AIS messages runs in parallel, and the serial operation of extracting multi-line messages is performed. Memory mapping is used to read the AIS message data file into memory to improve the speed of file reading and then decode it and save the results. The decoding process of the AIS messages file is shown in Figure 6.

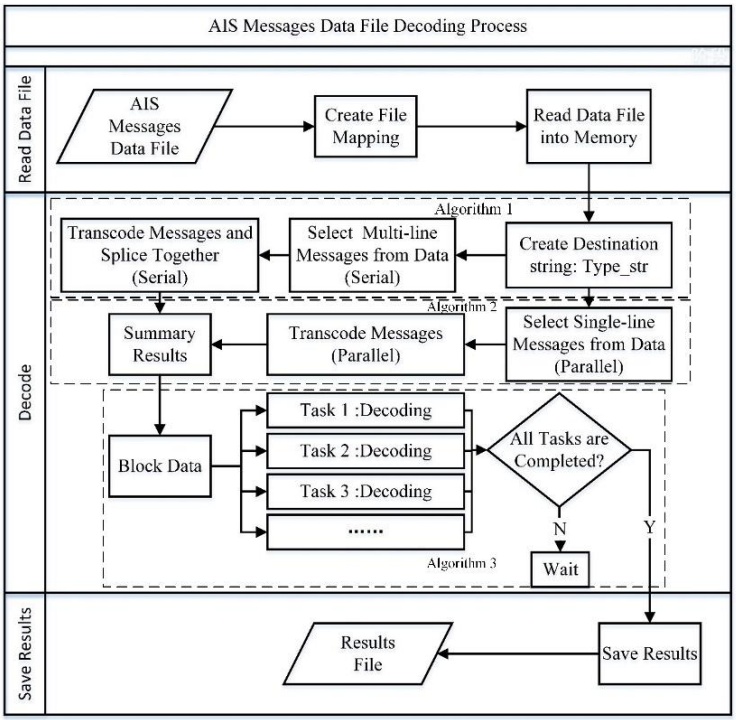

Figure 6. AIS messages data file decoding process

After the data file is read into memory, create a destination string called Type_str, which is composed of characters shown in Table 2, according to needs. There are 27 types of AIS messages. Each message type is represented by an ASCII character when Type_str is generated. For example, messages numbered 1, 2, 3, 5, 18, 19, and 24 are required, and the Type_str is "1235BCH".

Table 2. Encoding of type_str character

Num1	Chr1	Num2	Chr2	Num3	Chr3
1	1	10	:	19	C
2	2	11	;	20	D
3	3	12	<	21	E
4	4	13	=	22	F
5	5	14	>	23	G
6	6	15	?	24	H
7	7	16	@	25	I
8	8	17	A	26	J
9	9	18	B	27	K

Next, the AIS messages are filtered according to the content of Type_str, and serial operation Algorithm 1 converts multi-line messages into binary code. Running Algorithm 2 in parallel converts single-line messages into binary code.

After the execution of Algorithms 1 and 2, the binary encoded results Collection D of single-line messages and the binary encoded results Collection B of multi-line messages are obtained. Algorithm 3 is used to decode the binary code of the AIS messages. Algorithm 3 first divides the data blocks according to the number of cores of the CPU and then creates a task for each data block to perform the final decoding work in parallel. The description of the algorithms is shown in Table 3.

Table 3. The description of the algorithms

Algorithm 1. Convert multi-line messages into binary codes.

Input: **Type_str**, and the AIS Messages Data File **F**

Output: The Binary Codes of Multi-line Messages Collection **B**

B ← Null;
bw ← Null;
For i←1 **to** F.Length
  If F[i].ID in Type_str
    bw.Add(F[i]);
For i←1 **to** bw.Length
  If bw[i].x1 > 1
    Ms ← Null;
    For j←0 **to** bw[i].x1 − 1
      Ms.Add(bw[j]);
  B.Add(Binarycodes(Ms));
  i ← i + bw[i].x1 − 1;
Return B;

Algorithm 2. Convert single-line messages into binary codes.

Input: **Type_str**, and the AIS Messages Data File **F**

Output: The Binary Codes of Single-line Messages Collection **D**

D ← NULL;
H ← CPU.cores − 1;
R[1 to H] ← Create H tasks;
For i←1 **to** H-1
  $R[i].data \leftarrow F\left[\frac{F.Length}{H-1} * (i-1) + 1 \textbf{ to } \frac{F.Length}{H-1} * i\right]$;
R[H].data ← F[$\frac{F.Length}{H-1} * (H-2) + 1$ **to** F.Length];
Parallel.For i←1 **to** H
Parallel.For j←R[i].first **to** R[i].end
  If R[i].data[j].x1=1 and R[i].data[j].ID in Type_str
    D.Add(Binarycodes(R[i].data[j]));
Return D;

Algorithm 3. Decode the binary codes of AIS messages.

Input: Collection **B**, Collection **D**

Output: The Decoding Results Collection **E**

E ← NULL;
θ ← **B** ∪ **D**;
H ← CPU.cores
R[1 to H] ← Create H tasks;
For i←1 **to** H-1
  $R[i].data \leftarrow \theta\left[\frac{\theta.Length}{H-1} * (i-1) + 1 \textbf{ to } \theta.LengthH-1*i\right]$;
R[H].data ← θ[$\frac{\theta.Length}{H-1} * (H-2) + 1$ **to** θ.Length];
Parallel.For i←1 **to** H
  Parallel.For j← R[i].first **to** R[i].end
    E.Add(Decode(R[i].data[j]));
Return E;

When the decoding algorithm of AIS messages file is implemented, the file is first divided into n-1 data blocks according to the number of the computer's core, and then n-1 tasks are created. Each task assigns a block of data at the same time running Algorithm 2, which runs in parallel. Next, create the task n, assign the complete file to the task N, and run Algorithm 1. When all the tasks are completed, binary encoding results of single-line AIS messages and multi-line AIS messages are the combination of Collection D and Collection B, and then it is divided into n data blocks. Finally, the tasks are created to perform the operation of Algorithm 3. Each task allocates a data block until all tasks are completed, and the AIS messages file is decoded. The schematic diagram of algorithm implementation is shown in Figure 7.

4. Algorithm Test

4.1. Test Environment and Test Data

This paper selects a certain number of AIS messages for test. The test workstation is DELL WORKSTATION T7810, the operating system is Windows 7 Enterprise Edition, the programming language is C#, and the development environment is

Visual Studio 2015. The test data is the ship AIS messages from January 2010 to March 2010 in a certain sea area in the East China Sea. There are 65 files with 1,517,728,299 AIS messages.

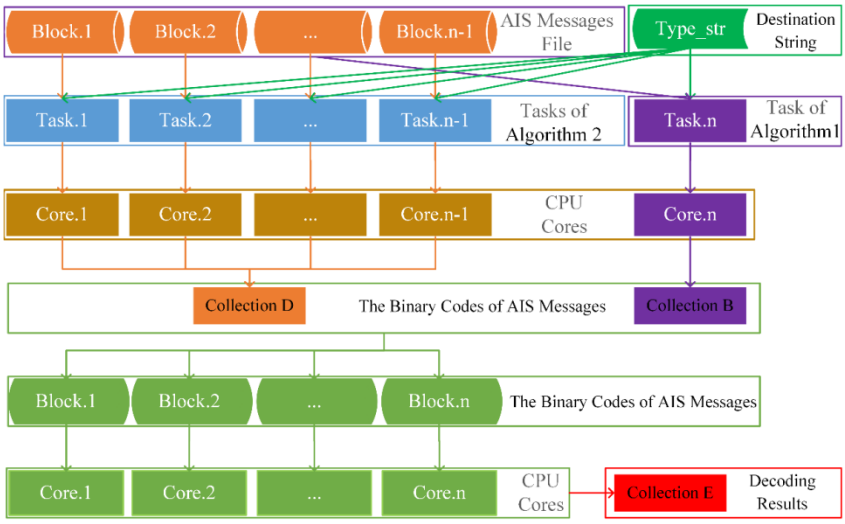

Figure 7. Schematic diagram of algorithm implementation

4.2. Test Results

The decoding of 65 AIS message files is completed at one time on the test workstation, and the processing speed is an average of 19,146.6 AIS messages per second. During the operation of the algorithm, the average reading speed reaches 48.55 thousand lines per second, and the average saving speed reaches 271,800 lines per second. In the decoding process, Algorithm 1 processes an average of 42,000 lines per second, Algorithm 2 processes an average of 233,300 lines per second, and Algorithm 3 processes an average of 41,200 lines per second, as shown in Figure 8.

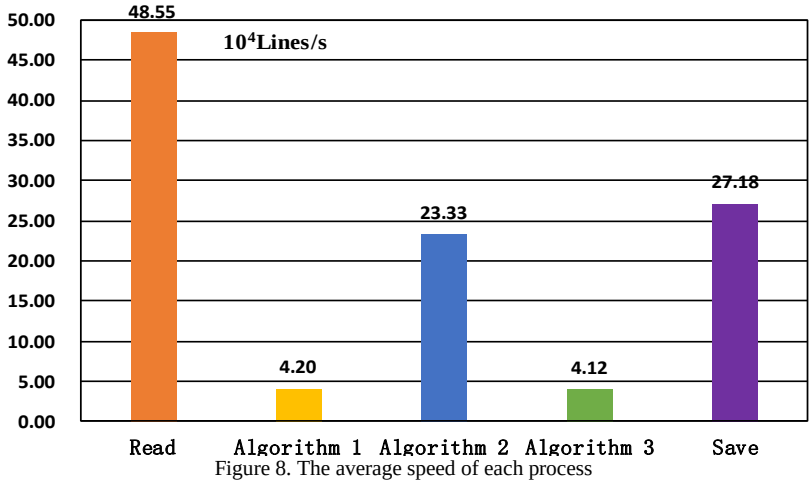

Figure 8. The average speed of each process

Among them, the speed of reading data and saving the result is the best, the speed of Algorithm 2 in the decoding is ideal, and the speed of Algorithms 1 and 3 is slow. Algorithm 1 runs in a single task, while Algorithm 2 performs multiple tasks in parallel. Moreover, the average speed of Algorithm 2 is much higher than that of Algorithm 1, so Algorithm 1 is the speed bottleneck for filtering and binary transcoding. Algorithm 3 must perform extensive integer and floating-point arithmetic, so it has the lowest average speed and is the focus of future algorithm improvement. The running speed distributions of Algorithms 1-3 in 65 files are shown in Figure 9.

In the process of decoding 65 data files, there is a certain fluctuation in the running speeds of the algorithms. When the running speed of Algorithm 1 is fast, the running speed of Algorithm 2 is also fast. Algorithm 3 has low velocity fluctuations and is slower than Algorithms 1 and 2.

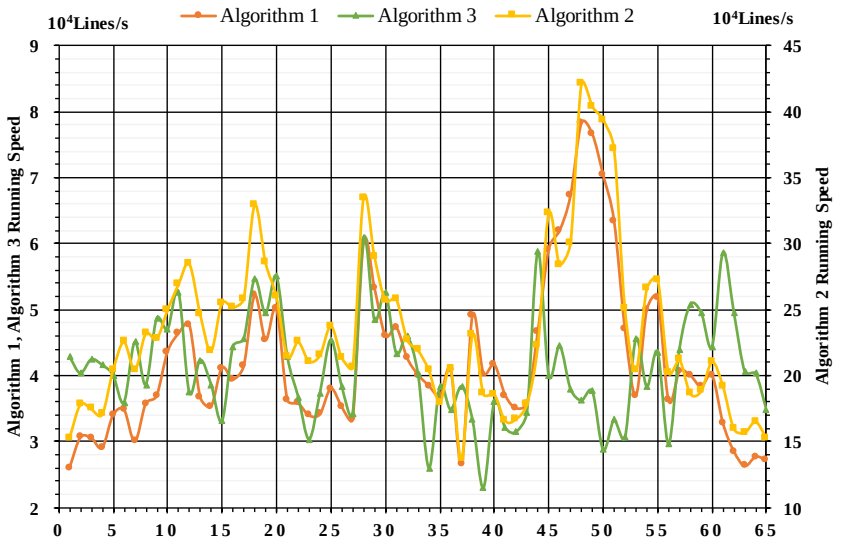

Figure 9. The running speeds of Algorithm 1, Algorithm 2, and Algorithm 3

The analysis of the results finds that Algorithms 1 and 2 mainly perform integer arithmetic, and Algorithm 3 performs floating-point arithmetic. The AIS message data files numbered 48, 49, 50, and 51 run Algorithms 1 and 2 faster because they are suitable for the integer operation, so Algorithm 3 is inconsistent with the fluctuations of Algorithms 1 and 2. Therefore, the speed stability of the algorithm will also be one of the focuses of future algorithm improvement work.

The performance comparison between the algorithm in this paper and the general decoding method is shown in Figure 10. The general AIS message decoding method runs steadily but slowly. The running speed of this algorithm in this paper is eight to ten times faster.

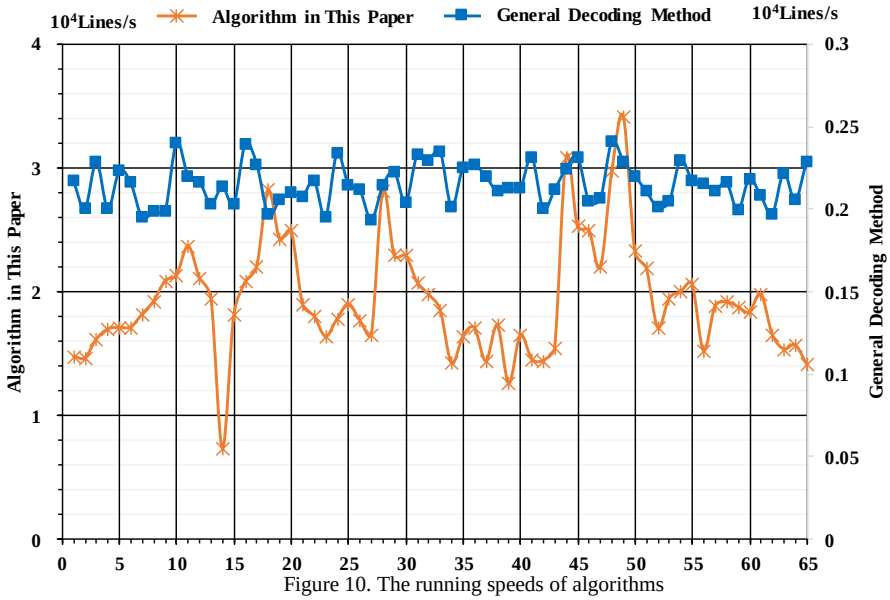

Figure 10. The running speeds of algorithms

5. Conclusions

The test results show that the large AIS message data decoding algorithm proposed in this paper is feasible. In the long-term test of packet decoding of 1,571,728,299 AIS messages, the result of processing 19,146.6 lines per second was obtained, and this algorithm is eight to ten times faster than the general AIS message decoding method. In view of the speed bottleneck problem of the algorithm, in future research, Algorithm 3 can be considered to run on the high-performance GPU, and the CPU runs Algorithms 1 and 2 to improve the overall working efficiency.

Acknowledgements

This research is supported by the Science and Technology Commission of Shanghai Municipality (No. 14170501600, No. 17DZ1101102, No. 18DZ1100901), and the project: Study on Improving the Navigation Efficiency of the Deep Water Channel of the Changjiang Estuary Using the Natural Water Depth of the Slope.

References

1. IMO, "International Convention for the Safety of Life at Sea," SOLAS 2004, IMO, London, pp. 270-271, 2004
2. UNCTAD, "Review of Maritime Transport 2016," UNCTAD/RMT/2016, UNCTAD, pp. 65, 2016
3. M. Winther, J. H. Christensen, and M. S. Plejdrup, "Emission Inventories for Ships in the Arctic based on Satellite Sampled AIS Data," *Atmospheric Environment*, Vol. 91, No. 7, pp. 1-14, 2014
4. L. Cheng, Z. Yuan, and J. Ou, "An AIS-based High-Resolution Ship Emission Inventory and its Uncertainty in Pearl River Delta Region, China," *Science of the Total Environment*, Vol. 573, pp. 1-10, 2016
5. B. Goldsworthy, "Spatial and Temporal Allocation of Ship Exhaust Emissions in Australian Coastal Waters using AIS Data: Analysis and Treatment of Data Gaps," *Atmospheric Environment*, Vol. 163, pp. 77-86, 2017
6. J. Pan, Q. Jiang, and J. Hu, "An AIS Data Visualization Model for Assessing Maritime Traffic Situation and its Applications," *Procedia Engineering*, Vol. 29, pp. 365-369, 2012
7. M. Fiorini, A. Capata, and D. D. Bloisi, "AIS Data Visualization for Maritime Spatial Planning," *International Journal of E-Navigation and Maritime Economy*, Vol. 5, pp. 45-46, 2016
8. W. Zhang, F. Goerlandt, and J. Montewka, "A Method for Detecting Possible Near Miss Ship Collisions from AIS Data," *Ocean Engineering*, Vol. 107, pp. 60-69, 2015
9. W. Zhang, F. Goerlandt, and P. Kujala, "An Advanced Method for Detecting Possible Near Miss Ship Collisions from AIS Data," *Ocean Engineering*, Vol. 124, pp. 141-156, 2016
10. A. N. Skauen, "Quantifying the Tracking Capability of Space-based AIS Systems," *Advances in Space Research*, Vol. 57, pp. 527-542, 2016
11. X. Wu, A. L. Mehta, and V. A. Zaloom, "Analysis of Waterway Transportation in Southeast Texas Waterway based on AIS Data," *Ocean Engineering*, Vol. 121, pp. 196-209, 2016
12. S. Li, L. Chen, and X. Chen, "Long-Range AIS Message Analysis based on the TianTuo-3 Micro Satellite," *Acta Astronautica*, Vol. 136, pp. 159-165, 2017
13. ITU, "Technical Characteristics for an Automatic Identification System using Time Division Multiple Access in the VHF, Maritime Mobile Frequency Band," REC. ITU-RM. 1371-5, ITU, Geneva, 2014
14. IEC, "Maritime Navigation and Radio Communication Equipment and Systems Digital Interfaces Part 1: Single Talker and Multiple Listeners," IEC61162-1, IEC, Geneva, 2010
15. C. Xu and L. L. Kong, "AIS Development in Recent Years and Message Decoding," *Applied Mechanics and Materials*, Vol. 3138, No. 536, pp. 776-781, 2014
16. L. Zhang, Q. Meng, and T. F. Fwa, "Big AIS Data based Spatial-Temporal Analyses of Ship Traffic in Singapore Port Waters," *Transportation Research Part E*: *Logistics and Transportation Review*, Article in Press, 2017
17. L. Q. Huang, W. C. Hu, and Z. P. Shao, "Study on Techniques of Decoding Output Data Packages from AIS," *Journal of Jimei University* (*Natural Science*), Vol. 10, No. 1, pp. 37-41, March 2005

Xiangkun Zeng is a Ph.D. candidate and lecturer in the College of Information Engineering at Shanghai Maritime University. His current research interests include intelligent transportation, data mining, and machine learning.

Huaran Yan is a Ph.D. candidate and lecturer in the Merchant Marine College at Shanghai Maritime University. His current research interests include traffic information engineering and control and traffic flow theory.

Yingjie Xiao is a professor and captain in the Merchant Marine College at Shanghai Maritime University. His current research interests include information engineering and control and ship and ocean engineering.

Xiaoming Yang is a Ph.D. He is currently working in the Shanghai offshore Engineering Research Institute in Shanghai Maritime University. His research interests include operation research, simulation and optimizations.