Eng Int Syst (2023) 5: 379–387
© 2023 CRL Publishing Ltd

Engineering
Intelligent Systems

Key Technology Research on End-Side Arithmetic Network Based on Resource Virtualization for Multi-Terminal Systems

Fang Cui, Mao Ni,[*] Ting Zhou and Hengjiang Wang

China Mobile Group Device Co., Ltd., Beijing 100053, China

As technology continues to advance, intelligent terminal hardware has broken through technical and application barriers and is affecting traditional industries such as healthcare, logistics and energy in a variety of product forms, resulting in the rapid development of multi-terminal collaborative systems. The key technology of the end-side arithmetic network under this system has become the key to the improvement of terminal network resource utilisation. This study first optimises the virtual resource scheduling in the multi-terminal system by designing a VM migration algorithm based on the minimum load imbalance, and then proposes a self-coded data compression algorithm, which introduces feature reconstruction and a XGBoost classification model. Finally, simulation experiments are conducted on the proposed two methods. The outcomes demonstrate that by applying the proposed VM migration algorithm, the I/O, Memory and CPU load imbalance are lower than 0.1, 0.2 and 0.3 respectively, improving the resource utilization. With the proposed data compression algorithm, the classification precision of data compression reaches up to 91% and the running time is reduced by up to 82%, greatly improving the data compression efficiency and providing a new method reference for resource scheduling in end to end computing power networks.

Keywords: Multi-terminal; Resource virtualization; End-side arithmetic networks; Data compression; VM migration

1. INTRODUCTION

With the continuous expansion of the scale of the intelligent terminal industry, the intelligent transformation of non-intelligent devices and the information construction of traditional industries have gradually become the focus of widespread social attention, and intelligent information processing technologies for terminals and networks are in urgent need of further development [1]. Given the increase in end-side computing power, edge computing and chip technology have taken on a large number of computing tasks, requiring an enormous amount of computing power. As a vital element of AI and the digital economy, computing power plays a vital role in the processing of data in various industries [2]. The current terminal device is comparable to a sensor; i.e., it transmits captured data to the terminal for processing, which leads to low utilization of end-side and cloud computing power and intensifies the discrepancy between supply and demand. Hence, end-side computing power technology has gradually become an increasingly important aspect of the computing power domain. With the booming development of end-side arithmetic technology in China, the terminal ecology is becoming more and more active, the scale of devices is huge, and multi-terminal systems have become mainstream [3]. However, this situation has also increased the difficulty of and the time required for the unified nano-management of devices. Therefore, this study is aimed at optimising key

[*]E-mail: nimaoch@163.com

end-side arithmetic network technologies for multi-terminal systems in terms of virtual resource scheduling and data compression, with a view to providing technical support for the further development of smart terminals.

2. RELATED WORKS

In recent years, multi-terminal systems have developed rapidly and resource virtualization has become mainstream. Therefore, research on end-side computing networks has received extensive attention from many professionals, and a series of breakthroughs have been achieved. Researchers such as Wang [4] proposed edge computing technology in order to solve the huge pressure that data collection, transmission and computation in distribution IoT networks put on the communication channel and the storage and computation system of the master station, and the deployment of an edge computing platform. Wang's study demonstrates that the approach is more effective for network lightweight data processing. Sodhro et al. [5] addressed the problem of wireless channel fluctuations that degrade the performance of the entire network system. They implemented a new QoS optimisation strategy at V2V for multimedia transmission on an IoT-based edge computing platform. They used QoS metrics to analyse the performance of V2V networks. The outcomes demonstrate that the algorithm outperforms traditional techniques and is a potential candidate for V2V multimedia transmission on adaptive edge computing platforms. On the other hand, Wang et al. [5] addressed a problem common to traditional encryption-based techniques: these techniques are extremely demanding on the memory and computing power of network end-users. Wang et al. proposed an efficient proxy re-encryption approach used in the ICN framework of information-centric networks to help reduce encryption time. The study [6] demonstrates that the method is more efficient in terms of computational overhead and computational power. It is demonstrated that the method has good performance in terms of computational overhead and communication complexity. Wang C. [7] and other experts proposed a resource scheduling algorithm (GATS) based on a hybrid genetic algorithm and taboo search to address the problem that existing scheduling methods are barely able to meet the demand for low-latency mobile communication, and establish a dynamic bandwidth allocation strategy for virtual links using integer linear programming. Simulation outcomes demonstrate that the scheduling completion time of this algorithm is reduced by 17%, which can meet the 5G services with strict delay requirements. Qi et al. addressed the problem of low message delivery rate in deep space communication networks due to high latency and frequent topology switching. They proposed an improved algorithm that uses a flexible load balancing strategy to establish the relationship between the optimal threshold and the reliability of network performance. The simulation outcomes demonstrate that the algorithm effectively improves the network throughput and message delivery rate. It also reduces the bandwidth rejection rate by about 4.7%, which can better balance the impact of spatial communication network topology updates and resource consumption [8]. Ra et al. [9] proposed an adaptive scheduling algorithm based on a combined auction allocation mechanism with dynamic pricing to address the inefficiency of scheduling algorithms for cloud resources in order to categorise various virtual machine (VM) requests. The results of the simulations demonstrate that the method can significantly improve provider profit and the utilisation of allocated resources utilization.

Zhang et al. [10] proposed a Phasor Principal Component Analysis (PPCA) method in the complex domain to compress the simultaneous phases as a whole, in response to the problem that the existing eigenvalue-based criterion is not suitable for data compression. Furthermore, the proposed PPCA is enhanced by an iteration-based process to reduce the computation of PPCA. Experiments demonstrated that the method achieves higher compression ratios, better precision of reconstructed data, and improved real-time performance, as well as significantly reducing the computational effort required. Xia et al. [11] proposed an Absolute Moment Block Truncation Coding (AMBTC) compression technique and Reversible Data Hiding (RDH) method based on Hoffman coding for compressing raw grey-scale image data. Experimental outcomes demonstrate that this scheme has better hiding load compared to other methods, as well as acceptable image visual quality. Karthikeyan et al. [12] proposed and analysed an energy-efficient data compression and data aggregation algorithm, and also suggested a novel feedback mechanism for sensor data compression. A secure data aggregation algorithm is used for data aggregation to avoid additional transmission and computational overhead on sensor nodes to reduce the energy consumed by the network. Experimental outcomes demonstrate that the algorithm extends the lifetime of the whole network by 24%. Yang J et al. [13] proposed a probabilistic prediction model for neural networks based on the maximum entropy principle of textual information in order to optimise the efficiency of massive data transmission in practice. The model combines an optimised Hoffman coding algorithm to optimise the entire process from data exchange to data compression, transmission and decompression. The outcomes demonstrate that the algorithm optimises the data compression transmission algorithm to achieve effective data compression Research experts such as Zhao [14] introduced a gating mechanism and proposed a gated neural network based on the idea that linguistic data features such as lexical tags and dependency tags contribute to compression generation. They demonstrated that the proposed method achieves better compression performance in both automatic metrics and manual evaluation compared to previous competing compression methods Chen et al. [15] proposed an efficient parallel memory compression framework in order to reduce memory requirements and address the problem of time-consuming execution of encoding and decoding leading to a severe slowing down in the training of Deep Neutral Networks (DNNs). Experiments demonstrate that this framework can reduce the memory footprint during training by an average of 2.3 times without loss of precision, achieving a 2.2 times data compression ratio.

In summary, genetic algorithms and other applications have achieved good outcomes for virtual resource scheduling in multi-terminal systems. At the same time, data compression techniques including neural networks have performed well in

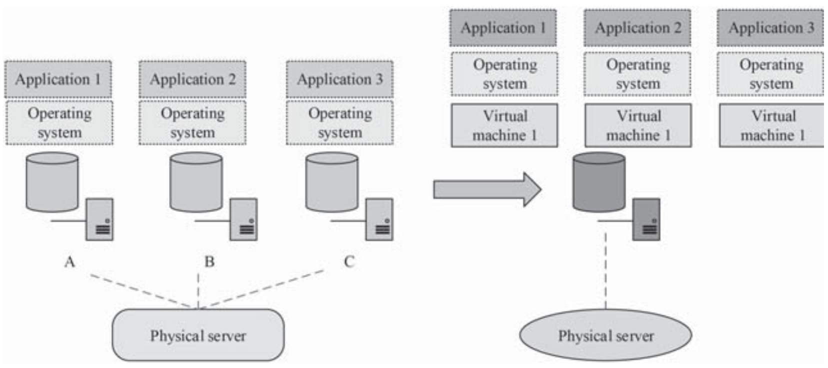

Figure 1 PM virtualisation diagram.

end-side arithmetic networks in terms of coping with the vast amounts of complex data being generated today. Therefore, this current study aims to optimise the virtual resource scheduling and data compression techniques in multi-terminal systems to further improve end-side arithmetic utilisation.

3. KEY TECHNOLOGIES FOR END-SIDE COMPUTING NETWORKS BASED ON RESOURCE VIRTUALISATION FOR MULTI-TERMINAL SYSTEMS

3.1 Optimization of Resource Virtualization Scheduling for Multi-Terminal Systems

An important feature of end-side computing networks is the processing of a task in concert with multiple end systems. Due to the differences in the hardware of various end devices, the abstraction of hardware capabilities is a necessary step towards improving efficiency, which is the main role of resource virtualisation [16]. Resource virtualisation enables different devices in a multi-terminal system to share the hardware capacity by creating a virtualised pool of resources, which overcomes the processing power deficit of a single terminal device. At the same time, the end-side arithmetic network can be matched to execution terminals with capabilities commensurate with user characteristics when faced with different services. The resultant change is that services flow seamlessly through the multi-terminal system; moreover, the combined capabilities of multiple devices including camera capabilities, display capabilities and sensor capabilities, are fully utilised [17]. When the user is operating, there may be a change in usage scenario that makes it difficult for the current device to meet the requirements of certain tasks, or the environment contains devices that are more suitable for processing [18]. In this case, the user can process the current task on the new device and achieve a better user experience, a process known as 'cross-terminology'. Also, cloud terminals allow multi-terminal systems to work better with the cloud edge. Because it is a terminal processed through the virtual resources of the cloud, the cloud terminal enables the interoperability of virtual resources between the cloud and the terminal, which is highly advantageous in terms of utilising virtual resources. By means of virtualisation technology, resources such as storage, networks and servers form virtual clusters, so users are able to have pools of virtualised resources, which is achieved through virtual machines (VMs). A physical machine (PM) virtualisation is illustrated in Fig 1.

In order to achieve load balancing and improve the quality of information services in multi-terminal systems, VMs need to be migrated from one PM to another in a timely manner, i.e., VM migration, which is an important aspect of virtual resource scheduling. The essence of VM migration is the migration of virtual resources, which is manifested as the migration of storage, computing and network resources [19]. The resource scheduling objective of VM migration is mainly load balancing, achieved by means of the multi-objective genetic algorithm. The purpose of VM migration is to ensure load balancing, Hence, this research introduces a multi-objective genetic algorithm in order to reduce the load imbalance as much as possible, and basing the VM on minimum load imbalance (MLI-VM). The multi-objective genetic algorithm is developed from the traditional genetic algorithm and is capable of solving multiple objective functions optimally, which is highly advantageous in handling multi-objective optimisation problems. The MLI-VM algorithm is based on the principle of multi-objective optimisation for the selection of the target PM, while the VM migration is not a mechanical migration of the current VM image file, but also a migration of peripherals, virtual runtime environment, etc. The VM migration process cannot be separated from the replication of the VM state, which inevitably produces a migration cost. The migration cost varies across VMs, and to quantify the migration cost, equation (1) is used.

$$P_i = \frac{M_i \cdot \mu_{mi}}{\mu_{mi} + \mu_{ci}} \tag{1}$$

In equation (1), P_i represents the migration cost, μ_{mi} is the memory utilisation, μ_{ci} is the physical CPU utilisation and M_i is the memory size. Therefore, the migration cost for a load-specific VM is directly affected by the amount of memory used. In order to achieve better migration and to ensure stable and reliable operation before and after migration, the minimum amount of memory migration must be one of the principles. In order to improve PM resource utilisation, VM migration must also follow complementary resource guidelines and take maximum account of the corresponding load imbalance during the migration process. Therefore,

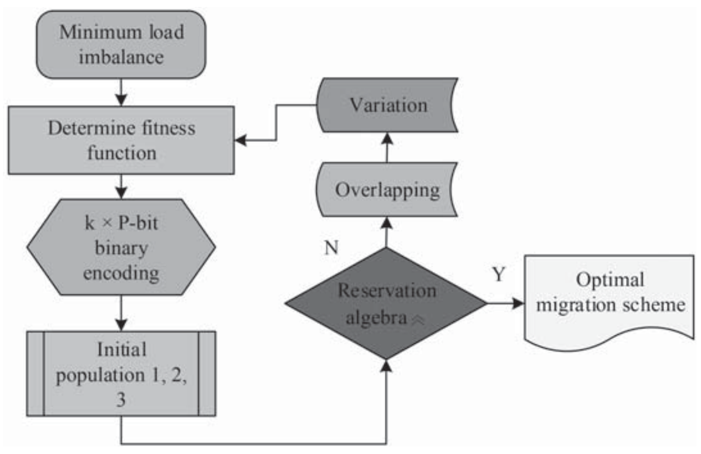

Figure 2 MLI-VM algorithm for target server selection process.

this study defines three dimensions for the proposed MLI-VM algorithm. The CPU load imbalance is demonstrated in equation (2).

$$r_c = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(a_i - E_c)^2} \tag{2}$$

In equation (2), r_c represents the CPU load imbalance, N represents the number of PMs, μ_{mi} is the CPU load of the ith PM, and E_c represents the average CPU load. The load imbalance of Memory is demonstrated in equation (3).

$$r_m = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(b_i - E_m)^2} \tag{3}$$

In equation (3), E_m is the Memory load average, which is equal to $\frac{1}{N}\sum_{i=1}^{N} b_i$ and r_m represents the Memory load imbalance. The I/O load imbalance is demonstrated in equation (3).

$$r_{i/o} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(c_i - E_{i/o})^2} \tag{4}$$

In Eq. (4), $r_{i/o}$ is the I/O load imbalance and $E_{i/o}$ represents the average I/O load value. Therefore, the target PM selection model based on the MLI-VM algorithm is demonstrated in Eq. (5).

$$\begin{cases} r_{i/o} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(c_i - E_{i/o})^2} \\ r_m = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(b_i - E_m)^2} \\ r_c = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(a_i - E_c)^2} \\ \min\{r_{i/o}, r_m, r_c\} \\ x = \{v_1, v_2, \ldots, v_k\}, v_i \in \{1, 2, \ldots, N\} \end{cases} \tag{5}$$

In Eq. (5), v_i represents the target PM number to be migrated, k represents the number of VMs to be migrated, and x represents the target PM number of all VMs, i.e. the VM migration solution. The MLI-VM algorithm solves Eq. (5) by a multi-objective genetic algorithm, which first uses the I/O load imbalance, Memory load imbalance and CPU load imbalance as the fitness functions, and the joint objective function The optimization is a population evolution process, and the three load imbalances correspond to the fitness functions of subgroup 1, subgroup 2 and subgroup 3 respectively. The three load imbalances correspond to the fitness functions of subpopulation 1, subpopulation 2 and subpopulation 3 respectively. For the three subpopulations corresponding to the three objective functions, the fitness of the individuals of the population is jointly found, while the selection operation is executed by the random competitive selection method. After the previous step, the three subpopulations are merged to form a single population and the next step of evolution is performed. In the selection operation, the execution is based on the relative fitness of individuals, which is obtained with equation (6).

$$P_{(x_i)} = \frac{f(x_i)}{\sum_{j=1}^{M} f(x_i)} \tag{6}$$

In equation (6), M is the number of individuals in the subpopulation, $P_{(x_i)}$ represents the relative fitness and $f(x_i)$ is the fitness of the individual. The greater the fitness of an individual, the higher the probability of selection, and conversely, the lower the relative fitness, the greater is the probability that an individual will be eliminated. The crossover operator of the algorithm is uniform crossover, which is used to perform genetic recombination, and the mutation probability is $0.001 \leq p_m \leq 0.1$, which is used to perform genetic mutation. The flow of the MLI-VM algorithm for target server selection is demonstrated in Fig 2.

4. END-SIDE ARITHMETIC NETWORK TECHNOLOGY BASED ON DATA COMPRESSION

Data compression is a key technology used for the low latency transmission of data in end-to-end arithmetic network multi-terminal systems as it removes redundant data features from large volumes of data. Data compression is used when storing information, mainly by removing redundant data that requires additional encoding, thus achieving a reduction in data volume [20]. When processing signals, data compression is the

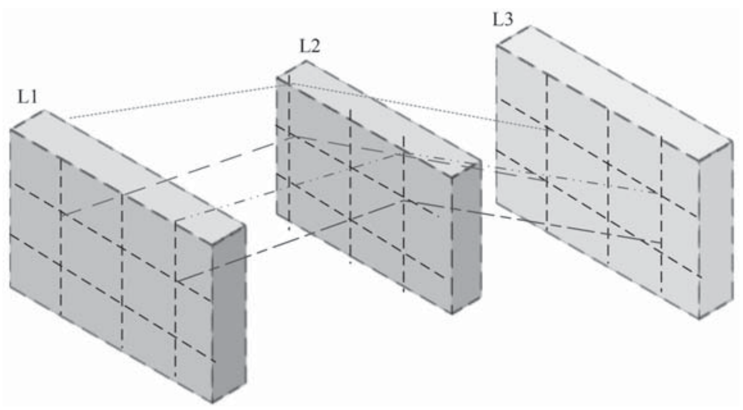

Figure 3 Basic structure of the proposed self-coded data compression process.

process of encoding information by means of fewer data bits than the original data. In general, the device that compresses the data is the encoder, and the decoder decompresses that data. In data transmission, the source encoding is encoded at the data source and the whole process needs to deal with the complexities of time and space [21]. In this study, a self-coding data compression algorithm is designed,which is able to collect data from the sensor receiver, build a self-coding model based on the input data, and compress large scale data sets using a coding and decoding network, thus improving data processing efficiency. The basic structure of the proposed self-encoding data compression process is shown in Fig 3.

In Fig 3, layers L1 to L2 are the encoding process of the proposed self-encoding data compression method, while the decoding process is from layers L2 to L3. The encoding function and decoding function are demonstrated in equation (7).

$$\begin{cases} h = f(x) = S_f(h_{A,b}(x)) \\ y = g(x) = S_g(h_{\overline{A},p}(h)) \end{cases} \tag{7}$$

In equation (7), h is the output layer expression, A is the mapping weight matrix between the L1 and L2 layers, $\overline{A}$ is the mapping weight matrix between the L2 and L3 layers, and $\overline{A}$ is the transpose matrix of A. f is the hidden layer expression. In constructing this data compression model, the study implemented training operations through the codec network to obtain the initial values of A in the neural network. $S_f(x)$ denotes the activation function of the encoder and S_g is the decoder activation function, using the Sigmoid function. A greater similarity between the output and input data than a given threshold means that the decoder-encoder network is able to retain most of the feature information in the original input data. The reconstruction error obtained with the Sigmoid function as the activation function is demonstrated in equation (8).

$$L(x, y) = \sum_{i=1}^{n} [(1 - x_i)\log(1 - y_i) + x_i \log y_i] \tag{8}$$

In equation (8), L represents the reconstruction error. The collective loss of the original input data training is calculated with Eq. (9).

$$J(\theta) = \sum_{x \in S} L[(x, g)g(f(x))] \tag{9}$$

In equation (9), J represents the collective loss outcome, S represents the original input data set, and θ is the parameter. The iterative training operation is then performed by the codec algorithm to obtain the minimum model parameters of the loss function$J(\theta)$, and the model is built. Since self-encoding data compression tends to reduce the precision, feature reconstruction is added to optimise the model. Feature reconstruction enables the conversion of the original data to a real matrix and, as a feature engineering technique, the conversion of the original data bit feature vectors is the purpose of the technique [22]. Feature reconstruction is an operation performed on the input data, which is then feature-compressed by self-encoding data compression to achieve a reduction in data size. The compression method which includes feature reconstruction is depicted in Figure 4.

After the data has been compressed, it is classified using the XGBoost classification model, which performs the classification of the compressed outcomes by means of integration trees. Let the compressed data in the compressed dataset be n and the number of attribute features be p, denoted as $\{(y_i, X_i)\}_{i=1}^{n}$. The integration tree construction is based on the regression tree, which performs the classification operation of y_i as demonstrated in equation (10).

$$\hat{y}_i = \sum_{m=1}^{M} f_m(X_i) \tag{10}$$

In equation (10), f_m denotes the expression of the function corresponding to the mth regression tree, as demonstrated in equation (11).

$$f_m(X_i) = \sum_{j=1}^{T} I[q(X_i)]w_j \tag{11}$$

In equation (11), $q(X_i)$ is the regression tree classification rule, T is the number of leaf nodes, $I(.)$ represents the schematic function, and w_j is the output value of the j leaf node. The objective function is defined as demonstrated in equation (12).

$$L = \sum_{k} \Omega(f_k) + \sum_{i} l(\hat{y}, y_i) \tag{12}$$

In equation (12), $\Omega(f_k)$ represents the regular term and $l(\hat{y}, y_i)$ represents the loss function. The compressed data

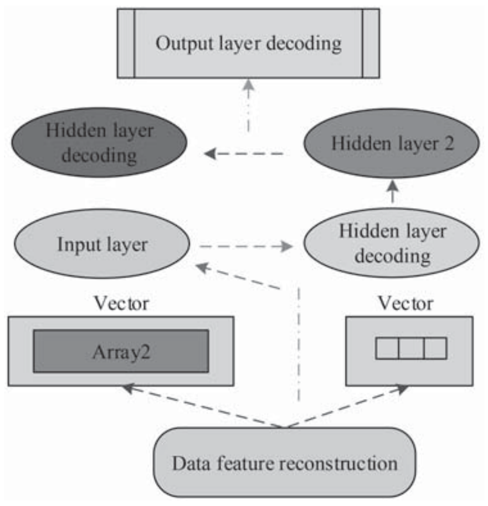

Figure 4 Self-coding data compression method including feature reconstruction.

Table 1 MLI-VM algorithm experimental simulation parameters.

Parameter	Description	Values
p_c	Crossover probability	0.6
p_m	Cross mutation probability	0.05
Q	Reservation evolution algebra	20
p	Binary code digits	7
k	Number of VMs to be migrated	3:1:7
k_0	Load level critical point	6
M	Individual number of Subpopulation	9
N	Number of PMs	65

model reduces the estimation error by adding a new regression tree, and the incoming objective function is demonstrated in Eq. (13) for a number of iterations of t

$$L^{(t)} = \sum_{i=1}^{n} l(f_t(X_i) + y_i + \hat{y}_i(t-1)) + \Omega(f_t) \qquad (13)$$

A second order Taylor expansion is performed on equation (13) at $\hat{y}_i^{(t-1)}$ and is defined as demonstrated in equation (14).

$$L(t) \cong \sum_{i=1}^{n} \left[\frac{1}{2} h_i f_t^2(X_i) + g_i f_t(X_i) \right] + \Omega(f_t) \qquad (14)$$

In equation (14), g_i represents the first order derivative and h_i is the second order derivative. The second-order Taylor expansion speeds up the optimisation of the objective function. As a classification rule and regression tree structure $q(X_i)$, the compressed dataset defined as I_j on the classification to leaf nodes, and the objective function value and optimal weights obtained when the number of iterations is t are demonstrated in Eq. (15).

$$\overline{L}^{(t)}(q) = -\frac{1}{2} \sum_{j=1}^{T} \frac{\left(\sum_{i \in I_j} g_i\right)}{\sum_{i \in I_j} h_i + \lambda} + \lambda_T \qquad (15)$$

Finally, the minimum value of $\overline{L}^{(t)}(q)$ is obtained using the optimal $q(\cdot)$, and the next iteration is executed directly after the t iteration until the stopping condition is met; the classification of the compressed outcomes is performed by the XGBoost model.

5. EFFECTIVENESS OF END-SIDE ARITHMETIC NETWORK TECHNOLOGY APPLICATION UNDER RESOURCE VIRTUALISATION

5.1 Virtual Resource Scheduling Outcomes

This study analyses the application effects of end-side arithmetic networks using resource virtualisation in multi-terminal systems, starting with the validation of the migration effects of the proposed MLI-VM algorithm. The simulation environment for this part of the experiment is MATLAB, and the genetic algorithm toolbox is also employed. The simulation parameters of the experimental procedure are given in Table 1.

Fig 5 demonstrates the outcomes of I/O, Memory and CPU load imbalance before and after migration using the MLI-VM algorithm. Fig 5 (a), (b) and (c) show the change curves of I/O, Memory and CPU load imbalance before and after migration respectively. Note that in Fig 5(a), before migration, the I/O load imbalance is stable at 0.15, while after

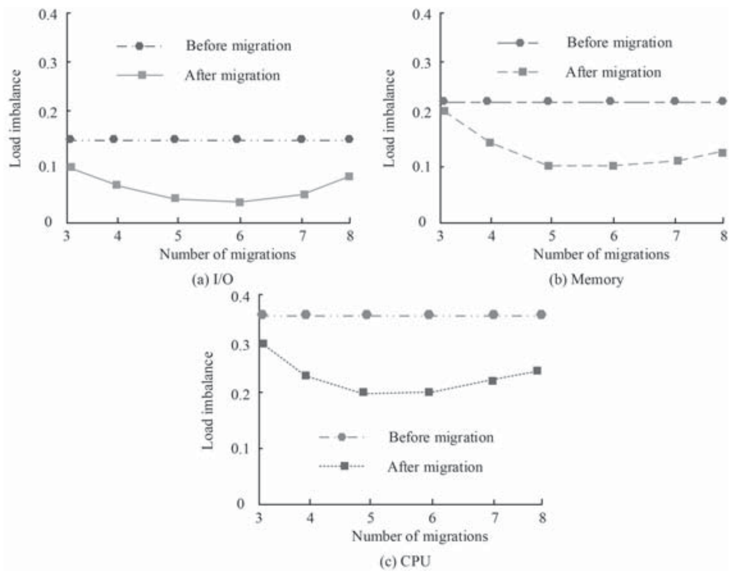

Figure 5 I/O, Memory and CPU load imbalance outcomes before and after migration using the MLI-VM algorithm.

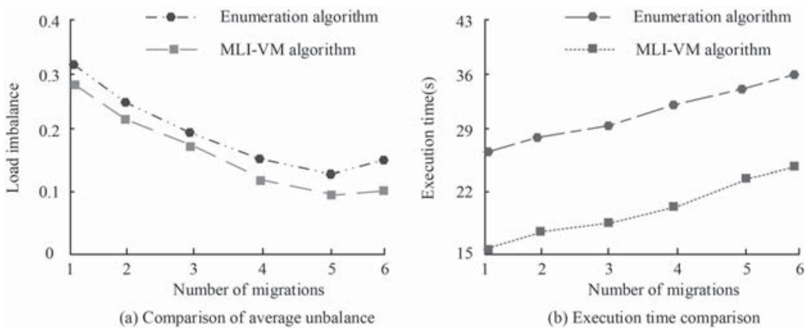

Figure 6 Comparison of migration effect and running time between MLI-VM migration algorithm and enumeration algorithm.

migration by the MLI-VM algorithm, the I/O load imbalance is highest at migration number 3, which is 0.1, and then as the migration number increases to 8, the I/O load imbalance tends to decrease and then increase, although all of them are below 0.1. As shown in Fig 5(b), the Memory load imbalance is 0.24 before migration and stays below 0.2 after migration, with the lowest being 0.1. In Fig 5(c), the CPU load imbalance is 0.35 before migration, and the highest value is 0.3 and the lowest is 0.2 after migration. After migration, the PM resource imbalance is significantly reduced and the resource utilisation is improved.

The results obtained with the MLI-VM migration algorithm are then compared with those of the enumeration algorithm. For comparison, the migration effects and running times are given in Fig 6. Fig 6(a) allows a comparison of the average imbalance outcomes between the MLI-VM migration algorithm and the enumeration algorithm, while Fig 6(b) shows the execution time achieved by each of the two methods. As seen in Fig 6(a), the average imbalance obtained by the enumeration method reaches its maximum value of 0.31 when the number of migrations is 1, and the lowest value is at the number of migrations is 5, but it is always above 0.1. The mean imbalance curve of the MLI-VM migration algorithm is always below that of the enumeration method, with a maximum value of 0.28 and a minimum value of 0.1, indicating that this method is able to use the global search of the multi-objective genetic algorithm to obtain an approximate = optimal solution, and outperforms the enumeration method. Fig 6(b) shows the difference between the two methods is greater in terms of execution time. The enumeration algorithm takes more time (up to 36s) as the number of migrations increases, while the execution time of the MLI-VM algorithm is always lower than that of the enumeration algorithm, with a maximum value of 23s, which is lower than the lowest value of the enumeration algorithm, indicating that the method has greater execution efficiency and performance.

5.2 Effectiveness of Self-Encoding Data Compression and Classification

This study proposes a self coding data compression method that combines the XGBoost classification model with the XGBoost classification model to achieve better compression and classification results. In this part of the experiment, the application effect of the proposed Self-Coding Compression based on Feature Rconstruction-XGBoost (FRSC-XGBoost) is analysed. The dataset was obtained from a multi-terminal system; its basic information is given in Table 1.

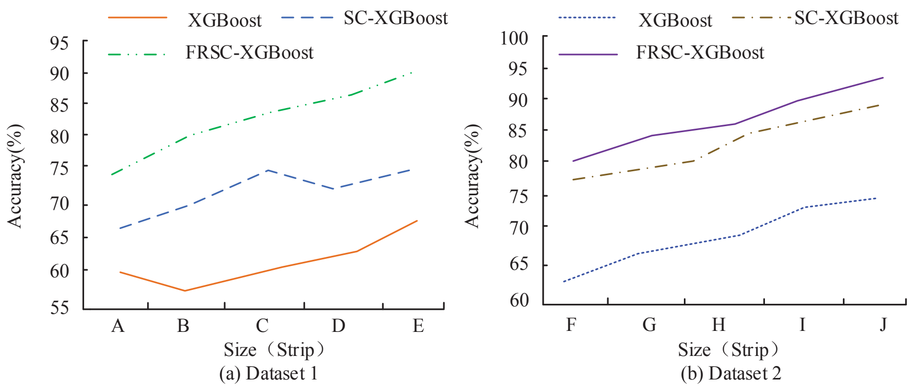

Figure 7 Comparison of operation precision of three methods applied to two datasets.

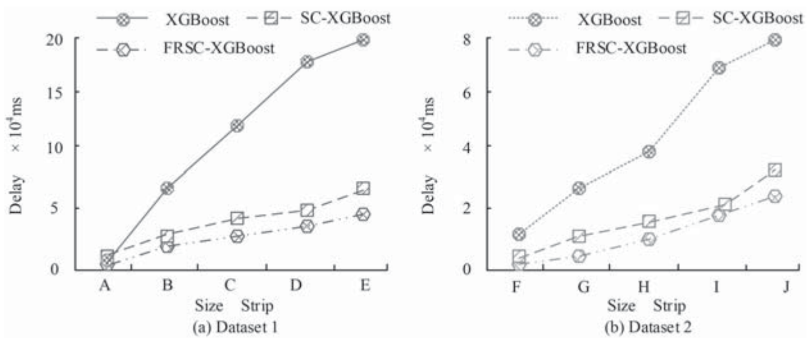

Figure 8 Comparison of running time of three methods in two selected data sets.

Dataset 1	Size (strip)	Dataset 2	Size (strip)
A	8000	F	5000
B	15000	G	9000
C	22,000	H	13000
D	29000	I	17000
E	36000	J	21000

The FRSC-XGBoost method proposed in this study was compared with the XGBoost and Self Coding Compression-XGBoost (SC-XGBoost) methods, and the running accuracies for the two datasets are shown in Fig 7. Fig 7 (a) and (b) give the running precision results of the three methods for dataset 1 and dataset 2 respectively. Fig 7(a) shows that the precision of the XGBoost and SC-XGBoost methods applied to dataset 1 generally tend to increase, with the highest values of the two being 65% and 71% respectively, a difference of 6%. However, the compression precision obtained by the FRSC-XGBoost method is always greater than that of the first two methods, with the maximum value close to 90%. Fig 7(b) shows that the precision of the XGBoost method applied to dataset 2 is still low, and increases as the number of data entries increases, but not by much, with a maximum value of around 70%. The difference in precision between the SC-XGBoost method and FRSC-XGBoost is smaller, but the proposed FRSC-XGBoost method still outperforms the first two methods, reaching a maximum precision rate of 91%.

For comparison, Fig 8 gives the runtimes of the three methods for the two selected datasets. The horizontal coordinates in Fig 8 are the data sizes with different numbers of entries, and the vertical coordinates represent the running times in ms. As indicated by Fig 8(a), the running times under the XGBoost method reach up to 20×10^4 ms, the SC-XGBoost method reaches over 5×10^4 ms, and the FRSC-XGBoost method tops out at about 3.5×10^4 ms, which is an improvement of 82% and 30% respectively compared to the first two methods. According to Fig 8(b), the XGBoost and SC-XGBoost methods have a maximum runtime of 3×10^4 ms and 8×10^4 ms, respectively, while the FRSC-XGBoost method has a maximum of 2×10^4 ms, which is an improvement of 75% and 33%, respectively, demonstrating greater efficiency and requiring a much lower classification time to compress the data.

6. CONCLUSION

The rapid development of smart services, digital twins and other services has given rise to more stringent requirements for intelligent information processing in terminals and networks. This has meant that key technologies for end-side arithmetic networks are in urgent need of development and updating. This study proposes a VM migration algorithm based on minimum load imbalance for scheduling virtual resources in multi-terminal systems, and introduces multi-objective genetic algorithms into it for the optimal solution of the objective function. Subsequently, a self-encoding data compression algorithm is proposed for the problem of processing massive environmental data in end-side arithmetic networks. With the proposed method, large-scale data sets are compressed by coding and decoding networks, while combining feature reconstruction and XGBoost classification models to optimise the compression effect. The outcomes demonstrate that before the migration of the MLI-VM algorithm, the I/O, Memory and CPU load imbalances were 0.15, 0.24 and 0.35 respectively; after the migration, the highest were all reduced by about 0.1. Compared with the enumeration method, this method requires an execution time of only 23s at most, which is more efficient. Meanwhile, the proposed FRSC-XGBoost method performs compressed classification in data sets collected from multi-terminal systems with close to 90% accuracy in both cases, and a running time of up to 3.5×10^4 ms and 2×10^4 ms respectively, which demonstrates much more efficient execution. However, the study has designed the data compression method in such a way that the pre-processing of the data collection may affect the integrity of the data features, and therefore further optimisation is needed to improve the precision.

REFERENCES

1. Chien H. T., Lin Y. D., Lai C. L., & Wang C. T. (2020). End-to-End slicing with optimized communication and computing resource allocation in multi-tenant 5G systems. *IEEE Transactions on Vehicular Technology, 69*(2):2079–2091.
2. Lu X., Ni Q., Zhao D., Cheng W., & Zhang H. (2019). Resource virtualization for customized delay-bounded QoS provisioning in uplink VMIMO-SC-FDMA systems. *IEEE Transactions on Communications, 67(*4):2951–2967.
3. Lv P., Pan S., Xu J. (2020). WiFi-Based virtual access network scheduling for downlink traffic dominated smart spaces. *Mobile Information Systems*, 2020: 8848558.
4. Wang Y., Cai D., Nian Y. (2020). Study of QoS-aware reliability transmission methods for edge computing networks in power distribution IoT. *Journal of Physics: Conference Series, 1650*(3):32112–32119.
5. Sodhro A. H., Obaidat M. S., Abbasi Q. H., Pace P., & Qaraqe M. (2019). Quality of service optimization in an IoT-Driven intelligent transportation system. *IEEE Wireless Communications, 26*(6):10–17.
6. Wang Q., Li W., Qin Z. (2019) Proxy re-encryption in access control framework of information-centric networks. *IEEE Access,* 7:48417–48429.
7. Wang C., Tang H., You W., Wang X., & Yuan Q. (2018). A resource scheduling algorithm with low latency for 5G networks. *Hsi-An Chiao Tung Ta Hsueh/Journal of Xi'an Jiaotong University, 52*(4):117–124.
8. Qi Y., Yang L., Pan C., & Li H. (2020). CGR-QV: A virtual topology DTN routing algorithm based on queue scheduling. *China Communications, 17*(7):113–123.
9. Ra A., Kk B. (2021). Resource allocation using dynamic pricing auction mechanism for supporting emergency demands in Cloud computing. *Journal of Parallel and Distributed Computing,* 158:213–226.
10. Zhang F. et al (2021). A synchrophasor data compression technique with iteration-enhanced phasor principal component analysis. *IEEE Transactions on Smart Grid, 12*(3):2365–2377.
11. Xia T. T., Lin J., Chang C. C., & Lu T. C. (2020). Reversible data hiding scheme based on the AMBTC compression technique and Huffman coding. *International Journal of Computational Science and Engineering, 22*(4):383–393.
12. Karthikeyan B., Kumar R., Inabathini S. R. (2018). Energy efficient data compression and aggregation technique for wireless sensor networks [TELOSB MOTES]. *International Journal of Reasoning-based Intelligent Systems,* 10:219–223.
13. Yang J., Zhang Z., Zhang N., Li M., & Zhang Y. (2019). Vehicle text data compression and transmission method based on maximum entropy neural network and optimized Huffman encoding algorithms. *Complexity, 2019*(4):5–9.
14. Zhao Y., Shen X., Senuma H., & Aizawa A. (2018). A comprehensive study: Sentence compression with linguistic knowledge-enhanced gated neural network. *Data & Knowledge Engineering,* 117:307–318.
15. Chen Z., Yang S., Liu C., Hu Y., Li K., & Li K. (2022). EPMC: efficient parallel memory compression in deep neural network training. *Neural Computing & Applications, 34*(1):757–769.
16. Borova M., Prauzek M., Konecny J., & Gaiova K. (2019). Environmental WSN edge computing concept by wavelet transform data compression in a sensor node. *IFAC-PapersOnLine, 52*(27):246–251.
17. Qin M., Chen L., Zhao N., Chen Y., & Wei G. (2020). Computing and relaying: Utilizing mobile edge computing for P2P communications. *IEEE Transactions on Vehicular Technology, 69*(2): 1582–1594.
18. Du J., Yu F. R., Chu X. (2018). Computation offloading and resource allocation in vehicular networks based on dual-side cost minimization. *IEEE Transactions on Vehicular Technology, 68(*2): 1079–1092.
19. Liu L., Chen C., Pei Q. (2021). Vehicular edge computing and networking: A survey. *Mobile Networks and Applications, 26*(3): 1145–1168.
20. Cappello F., Di S., Li S. (2019). Use cases of lossy compression for floating-point data in scientific data sets. *The International Journal of High-Performance Computing Applications, 33*(6): 1201–1220.
21. Wang X., Liu J., Wang Y., Chen X. & Chen L. (2020). Efficient tag grouping via collision reconciliation and data compression. *IEEE Transactions on Mobile Computing, 20*(5): 1817–1831.
22. Adedeji K. B. (2020). Performance evaluation of data compression algorithms for IOT-based smart water network management applications. *Journal of Applied Science & Process Engineering, 7*(2): 554–563.