Edge-Shared GraphSAGE: A New Method of Buffer Calculation for Parallel Management of Big Data Project Schedule

Yawei Zhao[1(✉)], Yangyuanxiang Xu[1], and Zhiwei Wang[1,2]

[1] University of Chinese Academy of Sciences, Beijing, China
zhaoyw@ucas.ac.cn, xuyangyuanxiang20@mails.ucas.ac.cn
[2] Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
wangzhiwei20s@ict.ac.cn

Abstract. Schedule network is essential for project schedule management. Critical Chain Method (CCM) is the most commonly used method on a schedule network to avoid project extension. The key to CCM lies in setting the proper buffer size. However, little work has considered the interdependence of nodes into buffer size calculating. In this paper, we present Edge-shared GraphSAGE, a model based on Graph Neural Network (GNN) for improving the result of buffer size prediction. Edge-shared GraphSAGE constructs undirected edges between schedule networks of projects sharing resources with each other. Fed by historical data of previous projects, the model predicts Safe-time Utilization Rate of each node of current project, so as to calculate the predicted size of the buffer. To the best of our knowledge, this is the first time that GNN is used in calculating buffer size. In several real projects, the proposed method outperforms Rule-based method and Machine Learning method.

Keywords: Schedule management · Project network · Edge-shared GraphSAGE · Buffer calculation · Parallel big data project

1 Introduction

Research shows that 65–100% of big data projects end in failure. Gartner [1] believes that 60% of BDA projects fail for being out of the budget or the plan. Therefore, in the actual implementation of big data projects, not only advanced technology is needed to complete some challenging tasks, but also the management of project process should be paid attention to. Generally, the project is broken down into some processes. Each process can be seen as nodes, and the sequence relationship between the processes can be abstracted into edges. Thus, a huge schedule network is formed, on which the process management of the project is carried out.

As shown in Fig. 1, the network shows a complete big data project, including the logical relationship order for all processes, in which a node represents a

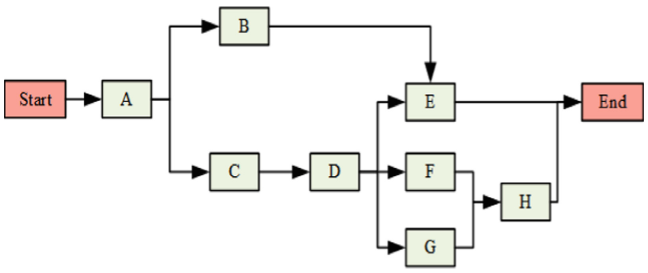


Fig. 1. Schematic diagram of network progress

process. Such network is called an Activity-on-node (AON) network. AON networks are particularly critical to the management of big data projects, especially the optimization of project progress. There is a problem in big data projects, that some operations often have a higher risk of delay due to their high level of uncertainty, which requires more buffer time to prevent such processes from affecting the completion of the entire project.

For the management of big data projects, the Critical Chain Method (CCM) in the management method can usually be used for analysis and optimization. The key of CCM is to find a suitable critical chain and set a reasonable buffer size. Traditional buffer size prediction methods are usually based on rules, which are too subjective and lack the learning of historical data. Besides, most rule-based methods consider the processes to be independent of each other, ignoring the successive logical relationship among processes.

We propose a new method based on graph neural network (GNN) to predict the buffer size in the process, which achieves better results. We also compared our method with the rule-based method and the regression methods of machine learning to verify its effectiveness.

The contributions of our paper are as follows:

(1) We construct feature indicators of different nodes through expert experience using the historical network data, and define Safe-time Utilization Rate θ_i as the dependent variable.
(2) According to the resource sharing of nodes, we merge the network of these projects to construct a parallel project schedule heterogeneous network.
(3) We propose Edge-shared GraphSAGE method based on graph neural network (GNN) to predict the Safe-time Utilization Rate of each node in the network, so as to indirectly calculate the buffer size of each process. Compared with other methods(include Rule-Based method and Machine Learning method), this method shows the superiority.

2 Related Work

In the field of schedule management, the Critical Chain Method has been widely used in recent years. For the setting of the buffer size, there are some methods that many scholars have proposed.

For the calculation of the Project Buffer, the most typical methods are the cut and paste method (C&PM) and the root square error method (RSEM) [2]. The shortcomings of C&PM include that if there are more processes in a project, the buffer that needs to be set will also be enlarged. For RSEM, its priori assumption is that each process is independent, but in fact it is difficult to strictly hold, which will make the buffer setting too small. Ashtiani B (2008) [3] used the Log-normal distribution to improve the deficiencies of the RSEM method, integrated the risk of the task, and calculated the parameters of the distribution. Gong Jun (2019) [4] proposed the concept of using information entropy and introduced an interval intuitionistic trapezoidal fuzzy number algorithm to quantify the impact of uncertain factors on the process, thereby modifying the root variance method. Zhou Yaoyao (2020) [5] proposed a calculation method for the critical chain buffer based on comprehensive constraints, using a piecewise fuzzy function to determine specific resource constraint coefficients. Xu Ye (2014) [6] used the linear regression equation method to predict the time schedule of different processes in the project, and optimized the schedule and construction period of the project by importing buffers, thereby shortening the construction period. Yadav S (2020) [7] used the principal component analysis (PCA) method to improve the root variance method, reduced the dimension of some indicators, merged them into the buffer, and gave actual cases to shorten the construction period.

However, for the setting of buffer size, most methods consider that each node in the network is independent, and a few researchers have only conducted the study of the previous and the latter projects. Especially for the schedule management of parallel big data projects, there are huge differences in their uncertainties, which is usually closely related to the position of the node in the entire graph. For example, there are some process nodes with large uncertainties, which are at the beginning and end of the network, also have significantly different buffer sizes.

In recent years, researches on graph analysis method are emerging in the computer field. The mainstream methods are a series of derivative and extended methods including GCN [8,12], such as GraphSAGE [9], GAT [10], etc. Some authors applied the GNN model to the prediction of traffic flow [11,13,14], which has a very good prediction effect. Aiming at the optimization of the schedule network, we try to apply GNN to the field of progress management for the first time. For buffer calculation, we consider the impact of the overall network on the buffer size of each process, and propose a new method of buffer size calculation based on Edge-shared GraphSAGE. In addition, we compare this method with the Rule-based method and the regression methods of Machine Learning, and the result demonstrates that Edge-shared GraphSAGE performs well to some degree.

3 Background

This section introduces theories of schedule network which is used in the modeling process.

3.1 Schedule Network

In a complete project process, it is necessary to describe the logical sequence of each activity. We define each activity of the project as a process, and define the logical sequence structure of a project as edges in the network. For two adjacent nodes, each node represents the operation of a process, and each edge represents the completion of the previous process and the beginning of the next process.

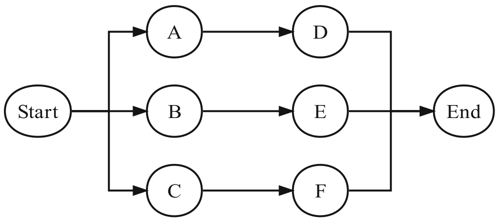


Fig. 2. Network diagram of a project

As shown in Fig. 2, each node represents each processing procedure in the big data project. For this network, the former procedure must be completed before the latter procedure can be executed, and there is a strong logical dependence in it.

3.2 Critical Chain Method

At the beginning of the project, the schedule network needs to be drawn based on the duration estimation, the given dependencies and constraints. Then the critical path is calculated. After the critical path is determined, the availability of resources is considered, so as to develop a resource-constrained schedule, in which the critical path is usually different from the previous ones. And the resource-constrained critical path in the schedule is called the critical chain [11].

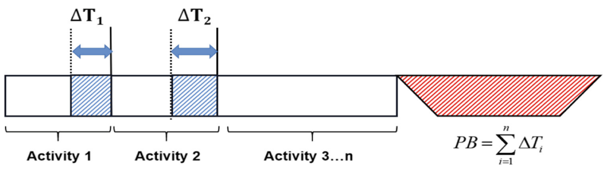


Fig. 3. Schematic diagram of the Project Buffer

The method above is called Critical Chain Method (CCM), which is based on Parkinson's law. CCM is a resource-constrained schedule network analysis tool. By setting a buffer to ensure network progress, it can effectively solve the situation of work delays. As shown in Fig. 3, through CCM, the accumulated buffer time will be eventually placed at the end of the project to form a Project Buffer. Import Buffer ensures that the critical chain will be successfully imported without affecting the process.

3.3 Buffer Size

In a large number of previous studies, there is a lack of constructing comprehensive quantitative indicators to analyze the size of the buffer, and also short of some indicators to measure the specific impact of process delaying. So it is necessary to set relevant feature indicators for the buffer of each process. The 90% of probability estimate of the *i-th* process duration is noted as S_{i1}, and the 50% of probability estimate is noted as S_{i2}. For the *i-th* process, the safe time of the process is noted as $\Delta S = S_{i1} - S_{i2}$. In a big data project, different processes have different actual buffer size, which is noted as ΔT.

Define θ_i as the Safe-time Utilization Rate of the *i-th* process:

$$\theta_i = \frac{\Delta T_i}{\Delta S_i} \tag{1}$$

where ΔT_i is the buffer time applicable to the *i-th* process in a big data project, and ΔS_i is the safe time of the *i-th* process. Because the uncertainty is different in different processes of real implementation of the project, θ_i is also different [12]. If the uncertainty of the sub-process is higher, the corresponding θ_i is larger and the greater ΔT_i is needed to ensure the completion of the process, otherwise, the θ_i is smaller.

Set $X = [x_1, x_2, x_3, \cdots, x_n]$ as the impact factor of the buffer size, which is defined as an independent variable. θ is the Safe-time Utilization Rate, which is defined as a dependent variable. Therefore, a nonlinear regression relationship between X and θ can be established, namely $\theta_i = f(x_1, x_2, x_3, \cdots, x_n)$.

3.4 Evaluation Index

The predicted value of the buffer size of each node in the process network is a continuous value. Therefore, we use R^2 to calculate the effectiveness of fitting results between the predicted value and the true value:

$$R^2 = \frac{\sum(\widehat{y}_i - \overline{y})^2}{\sum(y_i - \overline{y})^2} \tag{2}$$

where y_i represents the true value of the model, $\overline{y}$ represents the average of true values, and $\widehat{y}_i$ represents the predicted value.

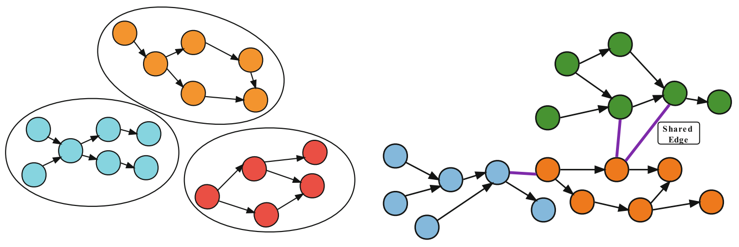
4 Edge-Shared GraphSAGE

4.1 Global Network Without Resource Sharing

Define the network topology graph set $G = \{G_1, G_2, G_3, \cdots, G_n, G_s\}$, which means a collection of $(n+1)$ schedule network of projects. As shown in Fig. 4a, this set contains n schedule networks G_j, $j \in [1, 2, 3, \cdots, n]$ of projects already done before, whose nodes contain numerical labels, and a network G_s whose nodes need to be predicted.

The projects are not connected to each other, i.e. they are independent of each other. We call the individual elements G_j, $j \in [1, 2, 3, \cdots, n]$ a sub-network of G. Each process node in G has its specific attribute value X as an independent variable. Besides, all nodes on G have a corresponding Safe-time Utilization Rate θ as a dependent variable. The labels of all nodes in sub-network G_s are unknown and need to be predicted.

For G_i and G_j, there will be resource sharing in some processes. If there are nodes sharing the same resources in G_i and G_j, define these nodes as shared nodes, and their connecting edges as shared edges e_{ij}.



(a) Without resource sharing (b) With resource sharing

Fig. 4. Global network with/without resource sharing

4.2 Global Network with Resource Sharing

For the whole network, the edge whose ends belonging to the same project schedule sub-network, is called a directed edge, otherwise, an undirected edge which connects two different sub-networks. As shown in Fig. 4b, edges in brown are shared edges.

$$e_{ij} = \begin{cases} G_i \rightarrow G_j, & if\ \ (i \in G_i) \cap (j \in G_j) \cap (i = j) \\ G_i \leftrightarrow G_j, & if\ \ (i \in G_i) \cap (j \in G_j) \cap (i \neq j) \end{cases} \tag{3}$$

4.3 Features of the Node

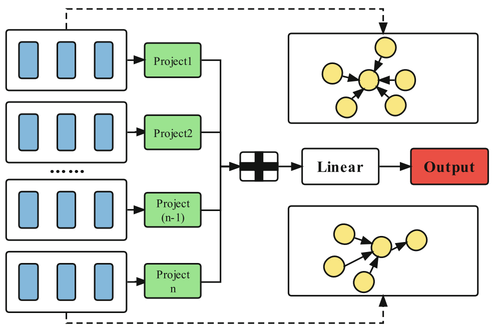
The attributes of each node in G are noted as $X = \{x_1, x_2, x_3, x_4, x_5\}$, where x_1 represents resource tensity of data, x_2 represents implementation difficulty of the process, x_3 represents connection level of data, x_4 represents changing probability of the process result, and x_5 represents importance of the process. These 5 features above are core factors affecting the prediction of buffer size.

Let experts score each process according to the indicators mentioned above, and the score ranges from *1* to *5*. *5* means the risk level of the indicator is *extremely high*, *4* means *high*, *3* means *fair*, *2* means *low*, and *1* means *extremely low*.

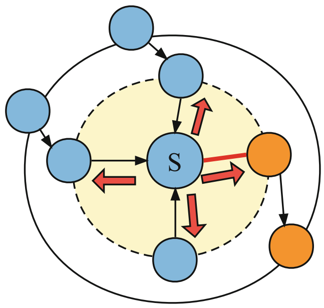
We conduct a number of project managers with rich experience to perform quantitative evaluations on the historical projects of the company to obtain features of historical data. We use the historical buffer time, the expected probability of 50% and of 90% to calculate the utilization rate of safe time θ.

4.4 Edge-Shared GraphSAGE

For most tasks, GNN is used to solve classification problems requiring some labels of nodes which are generally discrete values. We use GraphSAGE [12] as backbone to build a regression model that outputs continuous values for prediction, and we only consider the use of *Mean aggregation* to reduce the complexity of our model.



(a) GraphSAGE regression network structure



(b) Node aggregation

Fig. 5. Analytical diagram of the algorithm architecture (Color figure online)

As shown in Fig. 5a, different schedule sub-networks are input as nodes, and they are connected and aggregated through sharing edges when they share the same resources in the schedule, thereby forming a big network with historical data and current data. The schedule network is input to GraphSAGE for prediction. In order to fit continuous values, on the top layer, we replaced the Softmax layer with the Linear layer, which played an important role in predicting continuous data.

For the entire network, there will be a clear distinction between the internal connections of the sub-network and the connections between the sub-networks. This is because the networks of different schedules are heterogeneous networks, there is a distinct difference between their characteristics, and it cannot be treated the same during aggregation. We add parameters λ and fine-tuned the weights of these two parts above for trade-off to better adapt to the current regression prediction. The improved model is named Edge-shared GraphSAGE. The aggregation of the model is shown as Fig. 5b. The center node is the target aggregation node, noted as S. This figure contains two types of schedule networks, which are in blue and in yellow respectively. Similar to GraphSAGE, the neighbors of S are aggregated.

The aggregation function in GraphSAGE is expressed as:

$$h^l_{N(v)} \leftarrow Aggregate_l(\{h^{l-1}_u, \forall u \in N(v)\}) \tag{4}$$

We use the Mean aggregation function here, namely:

$$h^l_{N(v)} \leftarrow \sigma(W \cdot mean(\{h^{l-1}_u\} \cup \{h^{l-1}_u, \forall u \in N(v)\})) \tag{5}$$

Notes: $N(v)$ represents nodes connected to v

During aggregation, different projects connect to each other with shared edges, so as to form a big network. A project contributes differently to aggregation than its neighbor project, so we set weight λ for trade-off.

Define $u \in N(v)$. If u,v belong to the same project progress subgraph, mark u as $N^+(v)$. If u, v don't belong to the same subgraph, mark u as $N^-(v)$. The resulting aggregation function of Edge-shared GraphSAGE is denoted as:

$$h^l_{N(v)} \leftarrow \sigma(W \cdot mean(h^{l-1}_v \cup \{h^{l-1}_{u_1}, \forall u_1 \in N^+(v)\} \cup \{h^{l-1}_{u_2}, \forall u_2 \in N^-(v)\})) \tag{6}$$

λ is a hyper-parameter, which controls the degree of connectivity between different sub-networks and can be optimized through experimental tuning. When $\lambda = 1$, the weight between sub-networks is considered to be the same as the weight into the sub-network, which is to say, this structure is the Mean aggregation form of GraphSAGE. The pseudo code of the method is as Algorithm 1.

Algorithm 1: Edge-shared GraphSAGE: node value prediction (forward propagation) algorithm

```
Input: Graph $G(V,E) = \{G_1, G_2, \cdots, G_n\}$; input features $x_v, \forall v \in V$; depth
       $K$; weight matrices $W^k, \forall k \in \{1, \cdots, K\}$; non-linearity function $\sigma$;
       linear function $Linear$; mean aggregator function $MEAN$; positive
       neighborhood function $N^+ : v \to 2^v$; negative neighborhood function
       $N^- : v \to 2^v$; hyperparameter $\lambda$
Output: Predicted value $z_v$ for all $v \in V$
1  $h^0_v \leftarrow x_v, \forall v \in V$;
2  for $k = 1, \cdots, K$ do
3      for $v \in V$ do
4          $h^k_{N^+(v) \cup N^-(v)} \leftarrow MEAN(\{h^{k-1}_{u_1}, \forall u_1 \in N^+(v)\} \cup \{\lambda \cdot h^{k-1}_{u_2}, \forall u_2 \in N^-(v)\})$;
5          $h^k_v \leftarrow \sigma(W^k \cdot CONCAT(h^{k-1}_v, h^k_{N^+(v) \cup N^-(v)}))$;
6      end
7      $h^k_v \leftarrow h^k_v / \|h^k_v\|_2, \forall v \in V$;
8  end
9  $z_v \leftarrow Linear(h^K_v), \forall v \in V$;
10 return $z_v$;
```

4.5 Calculate the Project Buffer and Import Buffer

The Safe-time Utilization Rate is used to calculate the real buffer time of each process. The Project Buffer(PB) is the sum of all buffer times in the critical chain, namely:

$$\Delta T_i = \Delta S_i \cdot \theta_i; PB = \sum_{i=1}^{n} \Delta T_{i \in CriticalChain} \tag{7}$$

When the increasing of the Import Buffer does not affect the critical chain, let the Import Buffer be the sum of all buffers on the non-critical chain.

5 Experiment

5.1 Data

We investigate a company's implementation of big data projects in the past 4 years, and conduct two experiments on its relative schedule networks. One uses the existing data (5 schedule networks with known node labels, and a schedule network with unknown node labels) for comparative analysis, and let the invited experts to score the attributes of nodes in the network. The other uses similar pattens to increase random items of data, and automatically generates 50 similar projects for analysis and comparison.

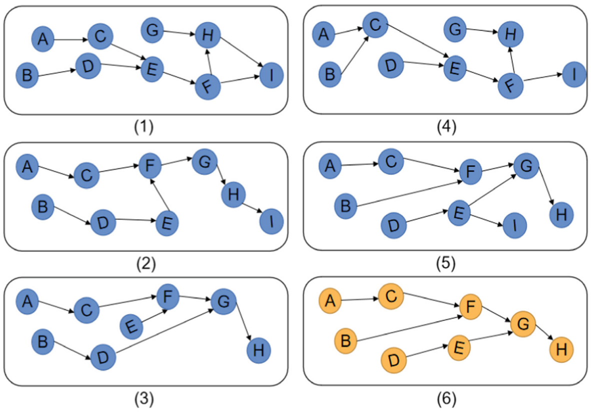


Fig. 6. A set of project network diagrams (Color figure online)

Figure 6 shows the topological structure of the project schedule network. Networks in blue are historical projects. This data set has a total of 44 nodes and 40 directed edges. The network in yellow is current project which is marked as true dataset. In addition, an analog dataset of 50 similar projects is generated, who has a total of 498 nodes and 477 edges. The historical data set is divided into 60% and 40% as training set and testing set, respectively. Compare the results of different methods in the testing set.

5.2 Method

Rule-Based Method. When setting up the critical chain for schedule management, the 50% shearing method is usually used to directly determine the size of each buffer. For these historical projects, each node has a corresponding buffer size. The essence of the Rule-Based method (Shearing method) is to directly take 50% of the process as the size of the safety time, which θ is always equal to 0.5. The R^2 of this method in the testing set is only 0.319, that is to say, the prediction accuracy of the buffer size for each node is low.

However, the advantage of this method is that it is relatively simple and intuitive. Because there is no reference to relevant historical data, the safety time prediction of each node cannot reflect the characteristics of the project itself, and the prediction is poor.

Machine Learning Method. For the nodes of each schedule network, suppose the nodes are independent of each other, we use three machine learning models for training, including: (1) Decision Tree Regression (2) Random Forest Regression (3) GBDT Regression. They are verified in the testing set, and measured by R^2. Although the original data form is graph, there is no node relationship information used in the calculation, which only uses the own information of each node. The results are shown in Table 1.

Table 1. Comparison of experimental results of different methods

Method	Dataset 1			Dataset 2		
Results	Test 1	Test 2	Test 3	Test 1	Test 2	Test 3
Decision Tree	0.613	0.628	**0.632**	0.593	0.608	**0.637**
Random Forest	0.707	0.695	**0.713**	0.683	**0.693**	0.686
GBDT	**0.643**	0.623	0.637	**0.607**	0.585	0.586

We conduct 10 tests on each dataset of each model and take down the best three results after adjusting parameters. Table 1 (Dataset 1) gives the prediction of three models using the historical data of the company. It can be seen that the best result is from *Random Forest*, whose R^2 reaches 0.713, which is much better than the Rule-based method. The results of both *Decision Tree* and *GBDT* are not as good as that of *Random Forest*. Table 1 (Dataset 2) shows the result on an expended data set of generated 50 similar projects. The overall R^2 is slightly lower than that in Dataset 1(historical data). The results indicate that when data expands, the data is less accidental, the result of prediction is worse.

Edge-Shared GraphSAGE. In previous sections, the topological structure of the network is not considered in experiments. In this section, we consider the topological information, so the prediction θ learns the relationship between nodes. In addition, the resources between projects are usually shared (e.g. people involved in one project are involved in other projects at the same time). As a result, we add edges between nodes where resources are shared and established strong correlation between projects.

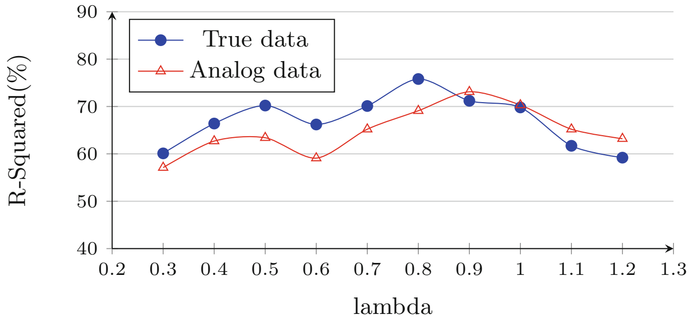


Fig. 7. The test results of Weight-GraphSAGE (Color figure online)

As show in Fig. 4b, node A in both two sub-networks is the process in which the same project manager participates. Two schedule networks share the same resource, so they are defined as shared edges. For some projects that don't have any shared resource, they are excluded from the calculation. Thus, a connected graph G can be obtained. We use Edge-shared GraphSAGE to make predictions on G. When $\lambda = 1$, the model degrades to GraphSAGE. We conduct experiments on the true dataset and simulated dataset separately, and select 9 groups of parameters where $\lambda \in \{0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2\}$, for analysis and comparison. We use R^2 as the evaluation metrics to measure the fit of the model. The results of Edge-shared GraphSAGE are shown in Fig. 7. Points in blue are the results of the real data, and points in red are the results of the expanded analog data. Relatively speaking, the real data is sparse, the quantity is small, so the stability is poor. We test the model using different value of λ, When $\lambda = 0.8$, R^2 is 0.758, the result is better than which of Machine Learning methods and GraphSAGE ($\lambda = 1$). When λ diminishes, the accuracy decreases, which may be caused by the reduced features that learned from adjacent sub-network. For what kind of parameters to choose is highly depend on the characteristics of the pattern and the distribution of degrees of schedule networks.

Therefore, we could not give the optimal parameters for all network to predict buffer size accurately. However, what is certain is that when the topology information and associated network information are added, the prediction results of each sub-network is improved. (e.g. R^2 of the true dataset is increased from 0.713 to 0.758, and which of the simulation dataset is increased from 0.693 to 0.731.) It shows that the model can learn information from the adjacent nodes, and leads to the best prediction of Safe-time Utilization Rate for each node.

5.3 Comparision

For real projects, the rule-based method (shearing method) is usually very convenient and has certain practical effects. But it doesn't take the mutual influence between nodes into account, and it also ignores the historical information of nodes. Therefore, we introduce historical data and use Machine Learning method to capture the characteristics of buffer settings in different nodes.

Table 2. Comparison of experimental results of different methods

Method	Best R^2 of true data	Best R^2 of analog data
Rule-Based Method	0.319	0.407
Machine Learning Method (RF)	0.713	0.693
GraphSAGE ($\lambda = 1$)	0.698	0.703
Edge-shared GraphSAGE ($\lambda = 0.8$)	**0.758**	**0.731**

As shown in Table 2, Machine Learning methods have improved the prediction greatly compared with the Rule-based method, and make more targeted predictions for different types of schedules. However, the information of the adjacent nodes is not used, so the prediction is not complete in theory. So we propose Edge-shared GraphSAGE method to capture the information of adjacent nodes, and the effect has been greatly improved.

Through networks, Edge-shared GraphSAGE learns the internal connection of nodes sharing the same resources, which will make the regression fit better. However, if the parameters are not selected properly, each project would take a negative impact on its adjacent projects. Therefore, it is necessary to constantly optimize the parameters to ensure the best results of the model.

6 Conclusion

We discussed how to optimize the buffer size in schedule management of parallel big data projects, so as to improve the ability to resist uncertainty of the project. The traditional method had the problem of excessive subjectivity and poor fit with real data. Therefore, we proposed Edge-shared GraphSAGE, a GNN-based model for regression analysis, which models the correlation between nodes, further improved the result of buffer size prediction, and provided a reference for the node regression problem of heterogeneous graphs.

For big data parallel projects, due to the large number of resources and the serious problem of resource conflication, if buffer settings were unreasonable, the process would be seriously affected. Introducing the Graph Neural Network prediction technology into the Critical Chain method, we could reduce the error of the schedule forecast of real project to a certain extent, avoided the delay of the big data project, and reduced the cost of the project. Therefore, the model was worthy of application and recommendation in the engineering field.

Acknowledgement. This work was supported in part by the National Key Research and Development Program of China under Grants 2020YFC1807104.

References

1. Toku, A.A., Uran, Z.E., Tekin, A.T.: Management of Big Data Projects: PMI Approach for Success, pp. 279–93. IGI Global (2019)
2. Jiang, h.: Summary and prospect of research on critical chain project management method. Build. Constr. **41**(09), 1764–1769 (2019)
3. Fallah, M., Ashtiani, B., Aryanezhad, M.B.: Critical chain project scheduling: utilizing uncertainty for buffer sizing. Int. J. Res. Rev. Appl. Sci. **3**(3), 280–289 (2010)
4. Gong, J., Hu, T., Yao, L.: Buffer setting method of critical chain based on information entropy. Acta Automatica Sinica **45**(x), 1–10 (2019)
5. Xiaoxiao, Z., Mengrui, L., Xunguo, Z., et al.: Critical chain size calculation method based on comprehensive resource constraints. J. Civ. Eng. Manag. **37**(06), 145–51 (2020)
6. Xu, Y.: Research on the application of critical chain multi-project schedule management in mobile phone projects. Shanghai Jiaotong University (2014)
7. Yadav, S., Singh, S.P.: Blockchain critical success factors for sustainable supply chain. Resour. Conserv. Recycl. **152**, 104505 (2020)
8. Abu-El-Haija, S., Kapoor, A., Perozzi, B., et al.: N-GCN: multi-scale graph convolution for semi-supervised node classification. In: Uncertainty in Artificial Intelligence, pp. 841–851. PMLR (2020)
9. Xiao, L., Wu, X., Wang, G.: Social network analysis based on graph SAGE. In: 2019 12th International Symposium on Computational Intelligence and Design (ISCID), vol. 2, pp. 196–199. IEEE (2019)
10. Veličković, P., Cucurull, G., Casanova, A., et al.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)
11. Msahli, M., Qiu, H., Zheng, Q., et al.: Topological graph convolutional network-based urban traffic flow and density prediction. IEEE Trans. Intell. Transp. Syst. **22**, 4560–4569 (2020)
12. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems, pp. 1024–1034 (2017)
13. Yu, B., Yin, H., Zhu, Z.: Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. arXiv preprint arXiv:1709.04875 (2017)
14. Wu, Z., Pan, S., Long, G., et al.: Graph WaveNet for deep spatial-temporal graph modeling. arXiv preprint arXiv:1906.00121 (2019)