Predicting milling overload based on sensor data: a graph-based approach

Roy Krumpak
Jožef Stefan Institute
Ljubljana, Slovenia
krumpak.roy@gmail.com

Jože M. Rožanec
Jožef Stefan Institute
Ljubljana, Slovenia
joze.rozanec@ijs.si

Dunja Mladenić
Jožef Stefan Institute
Ljubljana, Slovenia
dunja.mladenic@ijs.si

Zhenyu Guo
BGRIMM Technology Group
Beijing, China
guozhenyu@bgrimm.com

Tao Song
BGRIMM Technology Group
Beijing, China
songtao@bgrimm.com

Dumitru Roman
SINTEF Digital
Oslo, Norway
titi.roman@sintef.no

Inna Novalija
Jožef Stefan Institute
Ljubljana, Slovenia
inna.koval@ijs.si

Xiang Ma
SINTEF Industry
Oslo, Norway
xiang.ma@sintef.no

ABSTRACT

In this paper, we present an approach to predict milling overload that leverages time series-to-graph transformations, which, along with domain data encoded as a graph, are fed to predictive machine learning models. Additionally, we compared the performance of the graph-based approach with the TS2Vec foundational model, regarded as the State-Of-The-Art. Our results show that TS2Vec performed best across all time windows. While combining TS2Vec and graph embeddings resulted in reduced performance compared to TS2Vec, it enhanced the outcomes when compared to the sole use of graph embeddings. Furthermore, combining Ordinal Partition Graph and TS2Vec embeddings resulted in more stable performance across predictive time windows.

KEYWORDS

Time series, graphs, mining, milling, predictive maintenance, sensor data

1 INTRODUCTION

Milling, central to mineral processing, involves breaking down ores into smaller particles, but is prone to abnormal behavior due to material properties and upstream steps (Hodouin et al. 2001 [3]; Galán et al. 2002 [2]). While traditional control relied on operators, advances in machine learning (ML) have enabled data-driven optimization and predictive maintenance (Mobley 2002 [6]). Graph-based methods are increasingly applied to time series to capture temporal and structural relations (Silva 2021 [8]). Variants include Natural Visibility Graphs (NVG) to capture the time series topology (Lacasa et al. 2008 [4]; Stephen et al. 2015 [10]), Quantile Graphs for time series values' transitions (Silva et al. 2024 [9]), and Ordinal Partition Graphs to capture regular temporal patterns and their transitions.

The contributions of this paper include the use of multiple graph representations (not just one) to capture the structure of a time series and evaluation of the described approach on a real-world dataset.

2 USE-CASE DESCRIPTION

BGRIMM Technology Group is a Chinese leader in mining and mineral processing solutions, focusing on automation and intelligent control, with grinding optimization as a core area. Grinding is both the most energy-intensive step in mineral processing—accounting for 40% of total energy costs—and a key determinant of downstream recovery and product quality (Zhou et al. 2009 [11]; Lessard et al. 2016 [5]; Groenewald et al. 2006 [1]). At a 10,000 ton/day copper plant in Anhui Province using a SAG–ball–pebble (SABC) circuit, BGRIMM is developing intelligent control strategies to maximize throughput while preventing SAG mill overload. Central to this effort is accurate SAG power prediction, which serves as a feedforward signal to improve feed regulation and overall process efficiency.

3 DATASET

The dataset used in this article was collected and provided by BGRIMM Technology Group. The data consists of various sensor measurements from the machines used in their mine's ore processing plant, accounting for a total of 42 columns. One column stores the date and time of the measurement, while the rest contain numerical values. The sensor data was sampled every two seconds and compiled across a hundred days from January 1^{st} 2019, to April 12^{th} 2019, excluding the first two days of April, resulting in 4.32 million rows in the data. Besides the raw data, a description of an overload state was also provided. A column named `SAG_2201.power`, which represents the power of the SAG mill, is used to decide whether there is an anomaly in the data. If the column reaches a value above 4700 [kW] and has an upward trend or whenever it surpasses the value of 4800, this is considered an overload of the system, and a supervisor might take appropriate actions to stop the overloading.

Jože M. Rožanec and Roy Krumpak are co-first authors with equal contribution and importance.
Corresponding author: Jože M. Rožanec: joze.rozanec@ijs.si.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
Information Society 2025, 6–10 October 2025, Ljubljana, Slovenia
© 2025 Copyright held by the owner/author(s).
https://doi.org/10.70314/is.2025.sikdd.21

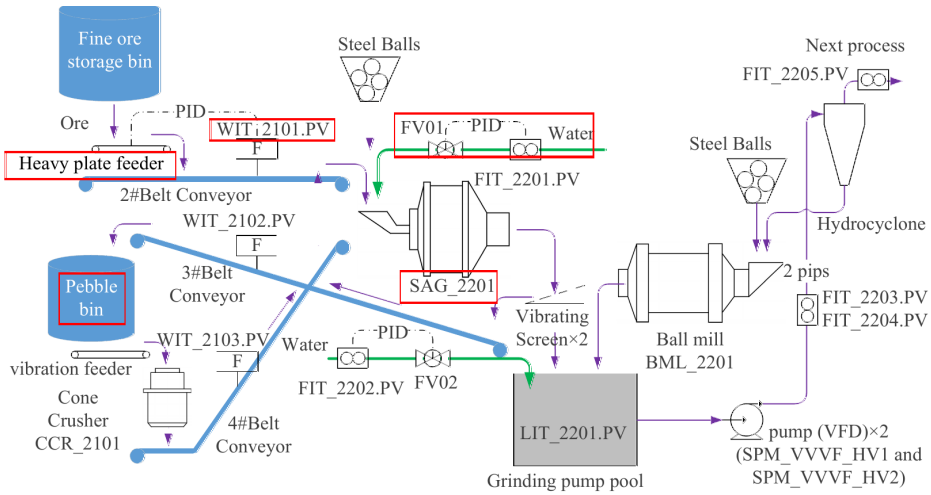


Figure 1: The diagram depicts the milling plant components and how they are connected. The components of interest are highlighted with red rectangles.

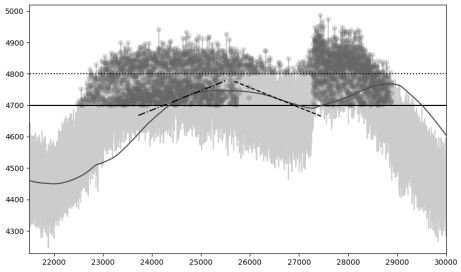


Figure 2: `SAG_2201.power` column (light gray), where anomalies (gray dots) are annotated based on the moving average (gray), the automatic anomaly label threshold (dotted black), the possible anomaly label threshold (solid black), and linear regression slope (positive - dashed and dotted, negative - dashed).

4 METHODOLOGY

4.1 Data preparation

Based on experts' input, the samples with `SAG_2201.power` < 4700 were labeled 0 (no anomalous event), others with 1 (milling overload). A 1-hour (1800-sample) moving average with linear regression checked for upward trends; if none, the label was reset to 0 (see Fig. 2). Next, we selected a subset of columns to be used in the analysis, utilizing expert knowledge to choose only those columns that are measured in the workflow before `SAG_2201.power` column. The resulting columns are `LIT_2103A.PV`, `FCV_2201.PID_SP`, `SAG_2201.Press_Ziyouduangaoya2`, `Feeder_Control.SP`, `SAG_2201.power` and `WIT_2101.PV`.

4.2 Feature engineering

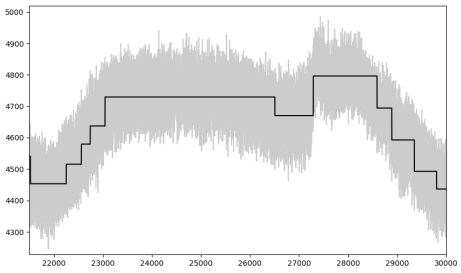
The raw data from the selected columns was first checked for any missing values, which were not present. In the next step, we detected changes in the columns and then replaced the values in the samples between two such changes with the mean value of that segment (see Fig. 3a). This data was further simplified with the help of a k-bins discretizer, which was used to encode each column with seven values based on the quantile into which each sample fell (see Fig. 3b). The column named `WIT_2101.PV` was excluded from the first step of data simplification and graph representations and was processed separately because its values did not appear to have distinct oscillating levels and did not benefit from such processing. After discretization, every column had an integer value between zero and six, and with each row being then interpreted as a state. The average state duration is 42 seconds. Repeated states (duplicate rows) were dropped, decreasing the size of the dataset (see Fig. 3c). For a visual representation of these steps, see Fig. 3, where the data from one picture is used, and, where important, also noted in the next one. The data here include raw data in Fig. 3a, the 'means' data in Fig. 3a and Fig. 3b, simplified data in Fig. 3b, and unique sample data in Fig. 3c. The annotated plot in Fig. 3c is used as the base data for an example NVG generation in Fig. 4. The numbers represent the same data point, one in the plot and one in the graph representation.

4.3 Modeling the data as graphs

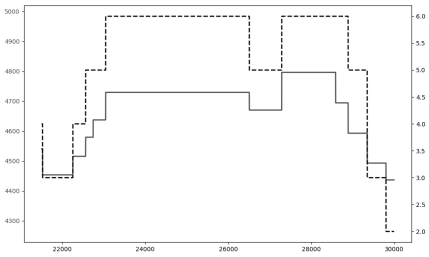
We employ three strategies for converting time series into graphs: Natural Visibility Graphs (NVG), Ordinal Partition Graphs (OPG), and Quantile Graphs (QG). We used the time series to graph and back library[1] to achieve this.

For each sample in the data, we built a graph representation of it by looking at the samples within a selected window w_s preceding it and applying the described time series to graph strategies on each column, apart from `WIT_2101.PV`, separately. Such graphs, called subgraphs, were bound to a default graph structure that presents which columns are neighboring in the plant process (see Fig. 1) by connecting a node which represents the `SAG_2201.power` column to every other column. The result of this step was a larger type of graph called a state graph (see Fig. 5). The black nodes represent nodes for a particular column, while gray nodes represent the subgraphs created from the time series. The subgraphs are connected to the column nodes via the node that corresponds to the first instance from the timeseries. Depending on the experiment, we made an additional step of joining w_0 many of the state graphs into a larger graph, which was used to generate embeddings.

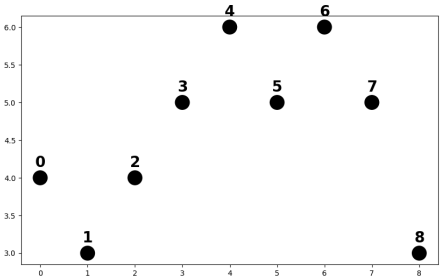
[1] https://timeseriestographs.com/



(a) `SAG_2201.power` column (light gray), where a threshold change detection was used to detect changes and to replace in-between values with the mean value (black).



(b) Result (dashed black) of applying a k-bins discretizer model on the previously simplified data (solid black) from Fig. 3a. Note the different y-axis scales of the overlaid graphs.



(c) A representation of the simplified column data from Fig. 3b, considering only the unique consecutive values.

Figure 3: Pipeline of transformations on the `SAG_2201.power` column.

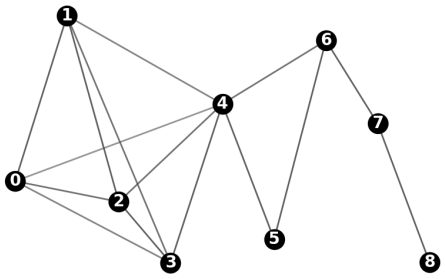


Figure 4: The Natural Visibility Graph representation of the data in Fig. 3c.

A Graph2Vec model from the karateclub library [7] was used to generate graph embeddings, with an embedding size of 250.

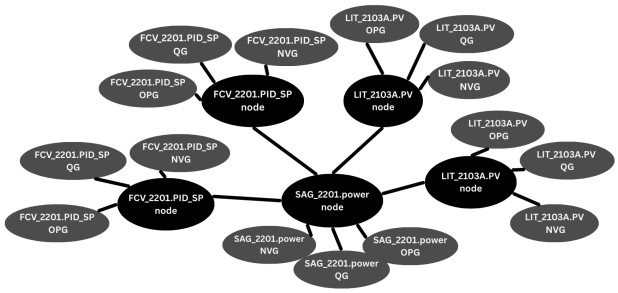


Figure 5: Example of a state graph.

We chose this model for its ease of use and performance reasons. Column `WIT_2101.PV` was also transformed into an embedding form by using a TS2Vec model[2]. The embedding output size was set to 40, as this is approximately the size of features proportional to the number of columns in the graph embeddings.

4.4 Model training and evaluation

An initial subset of the data, which included the data from the first available day, was used to test the performance of different graph embeddings. This was done to reduce the time and memory consumption for the first assessment. A CatBoost model was used, where it was trained for 800 iterations, with a learning rate equal to 0.03 and the Cross Entropy loss function, as well as the leaf regularization parameter set to 0.3. To assess our model's ability to predict anomalous states, we also tried to fit the model on the same data, but with the target column shifted accordingly. This was done for up to 90 shifts, which is equivalent to predicting 63 minutes in advance. When we selected the best graph embeddings, we built and tested the model on the entire data set.

5 EXPERIMENTS

We conducted three experiments, all of which follow the same template, where we tested how the structure of a graph affects the end model's ability to predict anomalies. This includes first creating subgraphs as NVG, OPG and QG representations of the columns with window size w_s and joining them into the state graph representation (see Fig. 5). Finally, w_0 many of these state graphs are joined sequentially according to the order given by the time at which the represented states appear in the data. The experiments differ in the window sizes w_s and w_0. Experiment A used $w_s = 50, w_0 = 1$, Experiment B used $w_s = 15, w_0 = 20$, lastly Experiment C used $w_s = 15, w_0 = 40$. If we take the average state duration of 42 seconds into account, we see that in Experiment A, data from the last 35 minutes is used, in Experiment B, 15 minutes, and finally in Experiment C, 28 minutes.

We carried out experiments similar to Experiment B, where the state graphs were structured based only on one specific type of subgraph. Furthermore, the impact of the separately processed `WIT_2101.PV` was also tested, by repeating the same experiments, with the difference being that this column's embeddings were excluded when training the final model. These experiments do not have a mark in the 'WIT' column of the resultst Table 3.

[2]https://github.com/zhihanyue/ts2vec

Time to predict[min]				
7	21	35	49	63
0.9905	0.9528	0.8929	0.8235	0.7623

Table 1: ROC AUC results of the experiment where all data was embedded with TS2Vec models.

	Time to predict[min]				
Experiment	7	21	35	49	63
A	0.6083	0.5763	0.5356	0.5333	0.4945
B	**0.6943**	**0.6698**	**0.6364**	0.6184	**0.6128**
C	0.5897	0.5688	0.6109	**0.6417**	0.5910

Table 2: ROC AUC results of the three experiments with respect to how far ahead the model is predicting. The best results are marked in bold text.

Lastly, a separate experiment was carried out, in which all raw data were processed using the TS2Vec model. Each column had its own TS2Vec model, which was used to embed the data associated with that column. Then, a CatBoost model with the same configuration as in the previous experiments was used in combination with TS2Vec joined embeddings to predict the anomalies. These results are gathered in Table 1.

6 RESULTS

The results of the three experiments, which tested the informativeness of the graph structure, as well as the experiments designed to determine which type of data is the most predictable, are summarized in the following tables.

As can be seen in Table 2, Experiments A and C have lower scores than Experiment B. However, Experiment C approaches the performance of Experiment B at the maximum predicting shift. For this reason, and because the types of graphs in Experiment B are smaller compared to those in Experiment C, the experiments that tested the impact of different types of data used Experiment B-type graphs. The best results for the final model were obtained from the data, where all columns were embedded using TS2Vec models, as shown in Table 1. Similarly, the results in table 3 show that when we predict anomalies from only the TS2Vec embeddings of the column `WIT_2101.PV`, the performance is the best.

Additionally, if we compare the experiments with `WIT_2101.PV` embeddings to the ones without them, we can see that the latter perform worse. This suggests that the TS2Vec embeddings are more informative than the graph embeddings. Nevertheless, when comparing different types of graphs used in the final graph, we can see that OPGs alone yield the best performance.

A few possible explanations for the difference in performance between the graph-based and time series-based approaches are possible. First, when working with graphs, there are more parameters that need to be optimized, such as window sizes and parameters for constructing graphs from time series. Another reason might be that NVGs have approximately thirty times more edges and eight times more nodes compared to OPGs and QGs, which makes them disproportionately large. Additionally, the construction of state graphs has repeated structures, which is inefficient. Lastly, the TS2Vec embeddings do not have these limitations, and embeddings can be made from the entirety of the data, as opposed to the simplified ones when not using TS2Vec.

type of data used				Time to predict[min]				
NVG	OPG	QG	WIT	7	21	35	49	63
✓	✓	✓	✓	0.6558	0.6418	0.6251	0.6402	0.6184
✓	✓	✓		0.5938	0.6257	0.5831	0.5882	0.5725
	✓		✓	0.7427	<u>0.7146</u>	<u>0.6930</u>	<u>0.6853</u>	<u>0.6719</u>
		✓	✓	0.7265	0.6959	0.6586	0.6502	0.6365
	✓			<u>0.7452</u>	0.6978	0.6838	0.6734	0.6578
		✓		0.7219	0.6866	0.6643	0.6416	0.6096
			✓	**0.9292**	**0.9025**	**0.8893**	**0.8004**	**0.7042**

Table 3: ROC AUC results of the models trained on different types of graphs and data for Experiment B across all days. The best results are written in bold text, while the second best are underlined.

7 CONCLUSIONS

In this paper, we discuss the use of graph-based time series representations for training machine learning models. Our experiments suggest that while this approach has potential, it did not outperform the TS2Vec foundational model and was unable to yield superior results when combined with it. Future work will explore alternative graph representations and utilize GNNs to integrate topological, semantic, and time series information directly into a single machine learning model, aiming to achieve superior results.

ACKNOWLEDGEMENTS

The Slovenian Research Agency supported this work. It was also developed as part of the Graph-Massivizer project (grant agreement No. 101093202), the enRichMyData project (grant agreement No. 101070284), and the DataPACT project (grant agreement No. 101189771), all funded by the Horizon Europe research and innovation program of the European Union.

REFERENCES

[1] J.W. de V. Groenewald, L.P. Coetzer, and C. Aldrich. 2006. Statistical monitoring of a grinding circuit: an industrial case study. *Minerals Engineering*, 19, 11, 1138–1148. DOI: 10.1016/j.mineng.2006.05.009.

[2] O. Galán, G.W. Barton, and J.A. Romagnoli. 2002. Robust control of a sag mill. *Powder Technology*, 124, 3, 264–271. DOI: 10.1016/S0032-5910(02)00021-9.

[3] D. Hodouin, S.-L Jämsä-Jounela, M.T. Carvalho, and L. Bergh. 2001. State of the art and challenges in mineral processing control. *Control Engineering Practice*, 9, 9, 995–1005. DOI: 10.1016/S0967-0661(01)00088-0.

[4] L. Lacasa, B. Luque, F. Ballesteros, J Luque, and J.C. Nuño. 2008. From time series to complex networks: the visibility graph. *Proceedings of the National Academy of Sciences*, 105, 13, 4972–4975. DOI: 10.1073/pnas.0709247105.

[5] J. Lessard, W. Sweetser, K. Bartram, J. Figueroa, and L. McHugh. 2016. Bridging the gap: understanding the economic impact of ore sorting on a mineral processing circuit. *Minerals Engineering*, 91, 5, 92–99. DOI: 10.1016/j.mineng.2015.08.019.

[6] R. Keith Mobley. 2002. 4 - benefits of predictive maintenance. In *An Introduction to Predictive Maintenance (Second Edition)*. Plant Engineering. (Second Edition ed.). R. Keith Mobley, editor. Butterworth-Heinemann, Burlington, 60–73. ISBN: 978-0-7506-7531-4. DOI: 10.1016/B978-075067531-4/50004-X.

[7] B. Rozemberczki, O. Kiss, and R. Sarkar. 2020. Karate Club: An API Oriented Open-source Python Framework for Unsupervised Learning on Graphs. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20)*. ACM, 3125–3132. DOI: 10.1145/3340531.3412757.

[8] V.F. Silva, M.E. Silva, P. Ribeiro, and F. Silva. 2021. Time series analysis via network science: concepts and algorithms. *WIREs Data Mining and Knowledge Discovery*, 11, 3, 1–39. DOI: 10.1002/widm.1404.

[9] V.F. Silva, M.E. Silva, and P. Ribeiroand F. Silva. 2024. Multilayer quantile graph for multivariate time series analysis and dimensionality reduction. *International Journal of Data Science and Analytics*, 1–13. DOI: 10.1007/s41060-024-00561-6.

[10] M. Stephen, C. Gu, and H. Yang. 2015. Visibility graph based time series analysis. *PloS one*, 10, 11, e0143015. DOI: 10.1371/journal.pone.0143015.

[11] P. Zhou, T. Chai, and H. Wang. 2009. Intelligent optimal-setting control for grinding circuits of mineral processing process. *IEEE Transactions on Automation Science and Engineering*, 6, 4, 730–743. DOI: 10.1109/TASE.2008.2011562.