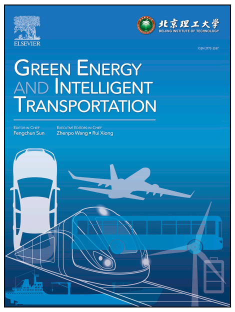


Extracting Multi-objective Multigraph Features for the Shortest Path Cost Prediction: Statistics-based or Learning-based?

Songwei Liu, Xinwei Wang, Michal Weiszer, Jun Chen

PII: S2773-1537(23)00065-8

DOI: https://doi.org/10.1016/j.geits.2023.100129

Reference: GEITS 100129

To appear in: *Green Energy and Intelligent Transportation*

Received Date: 29 April 2023

Revised Date: 30 August 2023

Accepted Date: 31 August 2023

Extracting Multi-objective Multigraph Features for the Shortest Path Cost Prediction: Statistics-based or Learning-based?

Songwei Liu, **Xinwei Wang**, **Michal Weiszer**, **Jun Chen**[1]

School of Engineering and Materials Science

Queen Mary, University of London, Mile End Road, London E1 4NS, UK

Email: {songwei.liu | xinwei.wang | m.weiszer | jun.chen}@qmul.ac.uk

CRediT Authorship Contribution Statement

Songwei Liu: Conceptualization, Methodology, Software, Formal analysis, Investigation, Writing - Original Draft, Visualization. **Xinwei Wang**: Software, Resources, Writing - Review & Editing. **Michal Weiszer**: Conceptualization, Writing - Review & Editing. **Jun Chen**: Conceptualization, Methodology, Writing - Review & Editing, Supervision, Project administration, Funding acquisition.

Data Availability

The data and materials used to support the findings of this study are available from the corresponding author upon reasonable request.

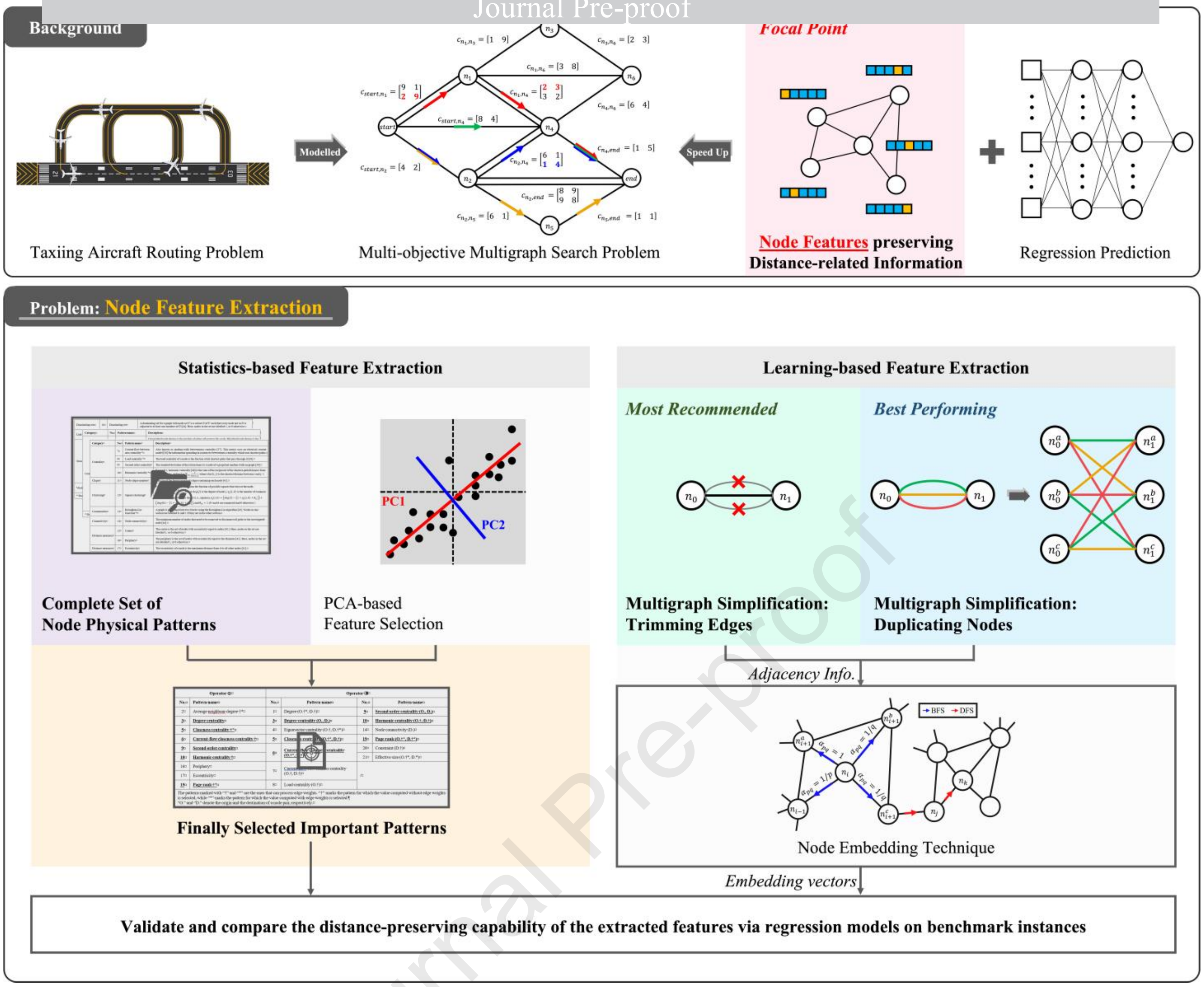
Acknowledgements

This work was supported by the UK Engineering and Physical Sciences Research Council (grant no. EP/N029496/1, EP/N029496/2, EP/N029356/1, EP/N029577/1, and EP/N029577/2) and the joint scholarship of the China Scholarship Council and Queen Mary, University of London (grant no. 202006830015).

Declaration of Generative AI and AI-assisted Technologies in the Writing Process

During the preparation of this work the authors used ChatGPT (24 May 2023 Version) in order to improve readability. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

[1] Corresponding author


Background
Taxiing Aircraft Routing Problem
Modelled
Multi-objective Multigraph Search Problem
Speed Up
Focal Point
Node Features preserving Distance-related Information
Regression Prediction
Problem: Node Feature Extraction
Statistics-based Feature Extraction
Complete Set of Node Physical Patterns
PCA-based Feature Selection
PC1
PC2
Finally Selected Important Patterns
Learning-based Feature Extraction
Most Recommended
Best Performing
Multigraph Simplification: Trimming Edges
Multigraph Simplification: Duplicating Nodes
Adjacency Info.
BFS
DFS
Node Embedding Technique
Embedding vectors
Validate and compare the distance-preserving capability of the extracted features via regression models on benchmark instances

Abstract—Efficient airport airside ground movement (AAGM) is key to successful operations of urban air mobility. Recent studies have introduced the use of multi-objective multigraphs (MOMGs) as the conceptual prototype to formulate AAGM. Swift calculation of the shortest path costs is crucial for the algorithmic heuristic search on MOMGs, however, previous work chiefly focused on single-objective simple graphs (SOSGs), treated cost enquires as search problems, and failed to keep a low level of computational time and storage complexity. This paper concentrates on the conceptual prototype MOMG, and investigates its node feature extraction, which lays the foundation for efficient prediction of shortest path costs. Two extraction methods are implemented and compared: a statistics-based method that summarises 22 node physical patterns from graph theory principles, and a learning-based method that employs node embedding technique to encode graph structures into a discriminative vector space. The former method can effectively evaluate the node physical patterns and reveals their individual importance for distance prediction, while the latter provides novel practices on processing multigraphs for node embedding algorithms that can merely handle SOSGs. Three regression models are applied to predict the shortest path costs to demonstrate the performance of each. Our experiments on randomly generated benchmark MOMGs show that (i) the statistics-based method underperforms on characterising small distance values due to severe overestimation, (ii) a subset of essential physical patterns can achieve comparable or slightly better prediction accuracy than that based on a complete set of patterns, and (iii) the learning-based method consistently outperforms the statistics-based method, while maintaining a competitive level of computational complexity.

Keywords—multi-objective multigraph, feature extraction, shortest path cost prediction, node patterns, node embeddings, regression

1. Introduction

Urban air mobility aims to revolutionize urban transportation by incorporating advanced air traffic systems. Emerging concepts, such as air taxis, on-demand aircraft, and large unmanned aerial vehicles, are gradually becoming integral to daily life. To ensure the long-term sustainability of the civil aviation industry, most of these novel operations need to seamlessly integrate with existing conventional airport infrastructures. Consequently, in future airport environments characterised by complex mixed traffic modalities, the harmonisation of airside ground movement with the well-established safety culture developed over decades in aviation becomes an imperative. A promising way to achieve this is to adopt taxiing operations based on four-dimensional-trajectories (4DTs, latitude/longitude/altitude/time) [1, 2], which facilitates the coordinated management of taxiing traffic, ensuring precise positioning at designated locations and predetermined times. The resulting benefits include savings in energy consumption, along with heightened operational safety and efficiency [3].

The introduction of 4DTs into airport airside ground movement (AAGM) formulates airport taxiways into a multi-objective multigraph (MOMG) [1, 4]. Within this modelling framework, multiple objectives necessitate trade-offs, including minimising taxiing time, fuel consumption and pollution emissions. Moreover, parallel edges between two adjacent nodes in MOMG correspond to multiple 4DTs assigned to each taxiway segment. The management of AAGM then becomes to determine the optimal traversal order of 4DTs over the taxiway layout to find the "shortest" paths with Pareto-front costs for both manned and unmanned aircraft. However, the MOMG shortest path search is NP-hard [5], posing considerable challenges to search algorithms that support efficient operations. Note that in this paper, our focus is entirely on the conceptual prototype MOMG. To provide generalised analyses, we utilise randomly generated benchmark MOMGs instead of specific real-world AAGM cases.

Among algorithms for solving the MOMG search problem, heuristic A* methods have been shown capable of achieving high optimality without overloading computing power [6], inclusive of airport multi-objective A* (AMOA*) [4] and multi-objective multigraph A* (MOMGA*) [6]. This advanced capability is attributed to the guidance of the algorithms' cost function $f(\cdot) = g(\cdot) + h(\cdot)$, where $g(\cdot)$ represents the exact cost of reaching the current node and $h(\cdot)$, heuristic information, represents the estimated cost toward the destination. Powered by the prior knowledge of cost attributes in $h(\cdot)$, the algorithms can discard some nodes and edges that do not belong to the optimal paths, thus saving running time [7]. On multi-objective graphs, $h(\cdot)$ is defined as the combination of the shortest distances for each objective between the residing node and the destination [8], however, these minimum values are not readily available and there has been a paucity of relevant research conducted on multigraphs.

The approaches to obtaining $h(\cdot)$, $i.e.$, calculating the shortest distances on single-objective simple graphs (SOSGs), include exact search methods, and approximation methods mainly composed of landmark-based and learning-based categories. Exact search methods deliver not only costs but also detailed traversal plans. However, the traversal plan is not necessary as heuristic information. Furthermore, exact search is of expensive computation costs and excessive memory requirements [6], $e.g.$, both Dijkstra's algorithm [9] and single-objective A* [10] have a time complexity of $O(|V|\log|V| + |E|)$, the time complexity of the Floyd-Warshall algorithm [11] is $O(|V|^3)$, and these three algorithms all require $O(|V|^2)$ for storage space [12], where $|V|$ and $|E|$ denote the number of nodes and edges in the graph respectively. Landmark-based approximation methods [13-15] select k nodes as landmarks ($k \ll |V|$) and other nodes hold the shortest distances to the landmarks as labels. When the shortest path costs are enquired, the labels of two investigated nodes are scanned, distances to the same landmarks are summed up and the minimum is the approximation. Despite turning the time and storage complexity down to $O(k)$ and $O(k|V|)$ respectively, landmark-based methods lack theoretical guarantees on the solution quality, and this quality heavily depends on the landmark selection, which is NP-hard [16]. Learning-based approximation methods [12, 17-19] predict the shortest path costs via machine learning, which is more assured to hit a good accuracy compared to landmark-based methods [20]. Besides, due to the low computational and storage complexity of learning-based methods, combing heuristic information derived from them with A* search is well-suited for online implementation. The feature-based calculation of learning-based methods is relatively effortless, node features to be stored are linearly related with the number of nodes, and predictive model parameters are independent of the graph size. $E.g.$, feedforward neural network takes $O(1)$ time and $O(d|V|)$ memory space, in which d represents the feature dimension. Although learning-based approximation on SOSGs has overestimation and/or underestimation because of the nature of regression, it has been proven empirically that the heuristic information generated by them performs well in MOMG shortest path search [6].

Motivated by the merits of learning-based shortest path cost approximation methods, we aim to further investigate their applications in MOMGs, and tackle the primary challenge for the learning model exploitation, namely feature engineering to characterise graph nodes. To the best of our knowledge, the learning-based shortest path cost approximation has exclusively relied on learning-based node feature extraction methods, which are inspired by an emerging node characterising technique called node embedding [21]. Node embedding automatically learns to encode non-Euclidean graph particulars into low-dimensional discriminative node vectors [22], and three approaches, GraRep [23], node2vec [24] and ProNE [25], have been illustrated to effectively preserve distance-relevant information within embeddings [12, 19, 6]. Since almost all embedding approaches primarily rely on graph adjacency information and are typically applied to SOSGs [22], they are not suitable for handling adjacency data of node pairs in MOMG, which is in matrix format rather than single values. Then a problem arises how to describe MOMG by SOSG(s), with the aim of facilitating the applicability of node embedding algorithms on MOMGs. As the sole related practice, [6] directly simplifies a MOMG into several SOSGs by splitting objectives and pruning parallel edges except the edge with the minimum cost between two connected nodes. Exploring alternative methods for (i) extracting node features and (ii) expressing adjacency information in MOMGs might yield improvements in prediction performance and should be considered.

In view of these discussions, to broaden the frontier of feature-based prediction for multigraph shortest path costs, this paper firstly proposes a statistics-based node feature extraction method for comparison with the learning-based extraction. We present a novel summary of 22 node physical patterns and employ the principal component analysis (PCA) [26] to select the vital ones. Secondly, new simplification of MOMG is devised to eliminate the multigraph nature, by duplicating nodes and maintaining their original connection relationships. We adopt the graph adjacency information obtained via this new simplification method, as well as that obtained through the method in [6] mentioned above, to generate embeddings that further offer insights into the impact of simplification methods on prediction. Thirdly, three regression models, multi-layer perceptron (MLP) [27], polynomial regression (PR) [28] and gradient boosted regression trees (GBRT) [29], are tested to show their abilities. Finally, experiments are performed on randomly generated benchmark instances, which can serve as templates for any real-life MOMG-based applications.

The main contributions are three-fold. (i) A collection of node physical patterns in benchmark multigraphs for predicting the shortest path costs is presented, which to our knowledge has not been done before. The most salient 16 patterns are identified, and these important patterns are found to offer similar or slightly improved prediction results compared to the full set of patterns. (ii) Two MOMG simplification methods to enable the MOMG handleability of node embedding algorithms are put forward and compared. And (iii) for the first time, we compare the prediction accuracy of shortest distances using the statistics-based and learning-based node feature extraction methods, and the results can potentially aid in the design of MOMG heuristic search algorithms.

The rest of the paper is organised as follows: Section 2 reviews learning-based node feature extraction methods for SOSGs, and their applications in the shortest path cost estimation. Section 3 sorts out node physical patterns and the feature importance selection method. Section 4 presents the two established MOMG simplification methods and outlines the utilised node embedding approach. Section 5 describes three regression models, which are then tested alongside the two proposed node feature extraction methods on benchmark MOMGs in Section 6. Lastly, the article ends with conclusions in Section 7.

2. Literature Review

Learning-based node feature extraction methods for predicting shortest distances in SOSGs draw inspiration from node embedding technique. These methods either utilise node embedding algorithms directly [12, 19], or exploit customised backpropagation learning methods that resemble node embedding to extract feature vectors in a supervised manner [17, 18].

When applied to a SOSG with a node set V and an edge set E, node embedding always employs graph adjacency information to map each node onto a vector $\psi \in \mathbb{R}^d$ that recapitulates the node's structural position, where d denotes the dimensionality of the embedding space and $d \ll |V|$. The key components of node embedding include (i) a pairwise similarity function s_G, which is defined over graph and quantifies the similarity between two nodes, (ii) an encoder function ENC that generates embeddings, (iii) a decoder function DEC, which reconstructs pairwise similarity based on the embeddings provided by ENC, and (iv) a loss function ℓ that assesses the difference between the similarity values obtained from s_G and DEC [22], according to which the ENC performance is optimised.

Table I ***Binary operators for node embedding vectors ψ_ϱ and ψ_σ***

Operator	Symbol	Definition
Addition †	$\oplus$	$\psi_\varrho + \psi_\sigma$
Concatenation	◨	$[\psi_\varrho, \psi_\sigma]$
Hadamard	$\odot$	$\psi_\varrho \cdot \psi_\sigma$
Subtraction	$\ominus$	$\psi_\varrho - \psi_\sigma$
† In [12], "Addition" is defined as $(\psi_\varrho + \psi_\sigma) \cdot 0.5$.		

The paper [12] pioneered the prediction of shortest path costs based on node embeddings. It innovatively adopted node2vec [24] and a hierarchical structure representation method Poincaré [30] to effectively incorporate distance-related information into node representations. Embedding dimensions of 32 and 128 were tested, with the latter outperforming in the majority of cases. The embedding vectors of an origin and destination pair ($a.k.a.$, O.D. pair) on the predicted path are integrated together via binary operators listed in Table 1. In the end, MLP networks estimate the shortest distances based on the integrated feature vector, and two metrics, mean absolute error (MAE) and the mean of relative error (MRE), measure the predictive performance. Results indicate that (i) binary operators have obvious but various impacts on different datasets and (ii) node2vec exhibits better predictive accuracy than Poincaré. Although comprehensively displaying a feasible framework for predicting shortest distances using node embeddings for the first time, there is an issue with the datasets, which consist of four social media user databases. The shortest path cost to be predicted refers to the minimum number of users that need to pass through to establish a connection between two unfamiliar users. The path length falls largely into the range of 2 to 4, with the longest distance being only 8 for one database and 6 for other databases. The prediction is more like a classification task where O.D. pairs are to be labelled with category numbers rather than a regression task, which renders the framework performance uncertain for long-distance-value regression. [19] further employed other two embedding approaches, GraRep [23] and ProNE [25], and verified their capability on the large-scale road networks of four metropolitan cities.

Compared to node embedding algorithms, customised backpropagation learning methods [17, 18] disregard the information contained within the original graph, as evidenced by their absence of pairwise similarity, encoder, and decoder functions. Though these customised methods obtain a $|V| \times d$ node representation matrix like node embedding does, what they exactly do is translate shortest distances to uninterpretable vectors of node pairs. In [17], the node representation matrix is initialised randomly, and the two corresponding row vectors of an O.D. pair are integrated in a "concatenation" way as Table 1 shows. The integrated vector is then fed to a MLP network, and the two rows in the matrix are gradually optimised backward based on the mean square error (MSE) between the actual and estimated distances. Evaluated on city road networks with thousands of nodes, this bespoke method reduces prediction errors compared to node2vec. Building upon the framework of [17], [18] enhanced the configuration of the backpropagation neural network and received the improved predictive accuracy. Nonetheless, an underlying reason for the inferior performance of node embedding algorithms compared to the customised backpropagation learning methods is the usage of default embedding parameters. Some of these embedding parameters control the node sampling scale, but the default values are considerably smaller than the diameter of city road networks, leading to undersampling, the inability to adequately capture node location properties, and the incapability of preserving distance-relevant information. Moreover, the customised backpropagation requires a large volume of training data. Node embedding algorithm encodes graph structural information, enabling the predictive model to establish correlations between prediction outcomes and the inputs with similar representations. However, in contrast, the customised backpropagation learning methods optimise representation vectors independently, necessitating a high amount of training data to enhance their discriminative power. Both studies [17] and [18] entail the complete set of node pairs. Although [18] claims that it reduces the data amount demand, by selecting some nodes as landmarks and using only node pairs consisting of landmarks and non-landmarks, the complete set of node pairs is still necessary for the first training epoch.

In addition, [19] also proposed the utilisation of graph neural networks (GNNs) [31] for generating node features. Even though the features obtained by GNNs offered competitive predictive results than the employed node embedding algorithms node2vec, GraRep and ProNE, it is important to note that the success of this GNN-based method for benchmark instances cannot be guaranteed. This research converts two-dimensional coordinates (latitude and longitude) into d-dimensional ($d = 32$ or 128) feature vectors via GNNs, however, benchmark instances may not be expressible in terms of coordinates. Besides, since real-life road networks were employed as test scenarios, and in this case the coordinates themselves can be used straightforward to calculate approximate shortest distances, the GNN-based conversion does not make intense sense. Considering the divergent working principle of this GNN-based method, it is presented separately from the aforementioned two categories.

Given above descriptions, while we can reason intuitively that the customised backpropagation learning is practicable for MOMGs, there is still a gap in processing multigraph format to fulfill the requirement of node embedding algorithms. Therefore, one of our focuses is to develop the MOMG processing method and then extract MOMG features via node embedding technique.

3. Statistics-based Feature Extraction

3.1 Node physical patterns

The lack of inherent node and edge attributes in the benchmark MOMG poses a challenge for feature-based shortest distance prediction. Graph adjacency information is the only available data in our MOMGs. This section provides a summary of existing node physical patterns that can be derived from graph adjacency information.

In Appendix, the table displays the collection of 22 node physical patterns that are applicable to multigraphs. To minimise the required storage space, only individual node patterns are enumerated, while the patterns of node pairs are neglected. The patterns are divided into ten groups, each serving a distinct purpose. The five groups, "centrality", "connectivity", "link analysis", "structural holes", and "vitality", explore the interactions between nodes. The group "clique" identifies complete subgraphs known as cliques, wherein all nodes are mutually adjacent [49]. The group "clustering" quantifies the tendency of nodes in a graph to cluster together [50, 51]. The group "communities" computes and evaluates community structures. The group "distance measure" demonstrates properties such as graph diameter, radius, and eccentricity [41]. Finally, the group "dominating sets" distinguishes a clique of nodes, where all nodes outside the clique have at least one neighbour within the clique [44]. Additionally, the Appendix table includes 13 patterns labelled with asterisks, where the corresponding pattern values differ depending on whether edge weights are included or not. In total, 35 physical patterns are taken into account for each node.

It is worth noting that certain physical patterns are associated with noticeable computational complexity. For instance, the calculation of "(5) closeness centrality" requires determining the shortest path distances from a target node to all other nodes in the graph. Similarly, "(8) load centrality" measures the number of shortest paths that pass through the investigated node. Furthermore, "(17) eccentricity" represents the longest path from the target node to any other nodes, and "(22) closeness vitality" reflects the change in path distances when the investigated node is removed from the graph. In addition to computational complexity, some physical patterns exhibit high correlations, $e.g.$, "(1) degree" and "(14) node connectivity", which are identical in unweighted graphs. As a consequence, identifying the importance of features is necessary to optimise computational resources and reduce coupling.

As stated earlier, heuristic information for multi-objective A* search is obtained by simply combining the shortest path cost of each objective [8]. In this section, the preprocessing of a MOMG involves partitioning the graph into several single-objective multigraphs, from which corresponding pattern values are derived. Afterward, the feature vectors of O.D. pairs are integrated using four binary operators listed in Table 1 for the subsequent feature selection and cost prediction.

3.2 Feature importance

To avoid any potentially confounding effects of the predictive model, this study adopts PCA [26] as the feature importance measure, which is a standalone technique operating independently of the predictive model. In contrast, numerous frequently used feature selection methods, such as permutation importance [52] and SHapley [53], perceive feature contributions in terms of predictive model outcomes, thereby relying to some extent on the performance of specific machine learning predictive models.

It is widely recognised that PCA is a popular transformation method to convert samples into a low-dimensional space and capture the most variable data components [54]. While the paper [26] introduced a new possibility, namely feature selection, for the application of PCA. The basic concept behind PCA-based feature selection is to assess the importance of features based on their weights (or, coefficients) on eigenvectors/components. Let us suppose that x is an eigenvector of the covariance matrix obtained by PCA, then its corresponding principal component can be expressed in Eq.(1).

$$a^T x = \sum_{i=1}^{d} a_i x_i \tag{1}$$

where a denotes an arbitrary sample vector, $a = [a_1, \cdots, a_d]^T$, $x = [x_1, \cdots, x_d]^T$, and d is the feature dimensionality. Obviously, the absolute value of x_i, where $i \in [1, d]$, can be used to statistically evaluate the contribution of the i-th feature in constructing this principal component. The smaller the absolute value, the less the contribution of the i-th feature. Specifically, if the absolute value is extremely small, eliminating $a_i x_i$ from $\sum_{i=1}^{d} a_i x_i$ will not influence the representational ability of this principal component. By extension, when a feature is not important for composing the principal component, we can speculate that this feature can be ignored in the original data space as well.

There are four steps to execute PCA feature selection. (i) Calculate the PCA covariance matrix using the training samples from original data. Compute all the necessary eigenvectors and eigenvalues (note that the number of eigenvectors/eigenvalues represent the number of principal components). (ii) Select m eigenvectors, denoted by $x_1, \cdots, x_m$, with the top largest eigenvalues. (iii) Calculate the i-th feature's contribution to the selected eigenvectors/components as: $c_i = \sum_{\eta=1}^{m} |x_{\eta i}|$, where $x_{\eta i}$ stands for the i-th element/weight/coefficient of eigenvector x_η, $i \in [1, d]$, $\eta \in [1, m]$, and $|x_{\eta i}|$ means the absolute value of $x_{\eta i}$. And (iv) sort c_i in the descending order and the descending feature importance is obtained.

In the implementation of this paper, there are two questions worth thinking about: (i) how to determine the number of principal components ($i.e.$, the second step of PCA feature selection), and (ii) how to summarise the set of important features when facing potential disturbances caused by the binary operators and graphs, particularly the bi-objectives that may also produce varying results. Regarding the first question, we compare the feature selection results obtained from taking only the first principal component and from exploiting multiple components, please see detailed experiments in Section 6.2. With regard to the second question, we initially select a relatively large number of features, and then take the intersection of the results obtained via different operators and graphs.

4. Learning-based Feature Extraction

This section describes the process of extracting features from multigraph nodes based on node embedding. Due to the limitation that most embedding approaches can only deal with SOSGs [22], where the entire graph adjacency relations can be expressed in matrix format, a problem arises how to describe MOMG by SOSG(s), as MOMG adjacency information is difficult to directly represent in the same format. Thus, two MOMG simplification methods are proposed in Section 4.1 to represent MOMG adjacent data by SOSGs. And Section 4.2 presents node2vec [24] as an example of the node embedding approach adopted in this paper.

In later narratives, we denote an undirected weighted MOMG as $MOMG = (V_{MOMG}, E_{MOMG})$, where $|V_{MOMG}|$ is the number of nodes and $|E_{MOMG}|$ is the number of edges. In the graph, there are at most Δ parallel edges existing between two adjacent nodes and the objective number is Ω.

4.1 Multi-objective multigraph simplification

The general idea of simplifying a MOMG involves two steps: first, splitting objectives, and second, eliminating parallel edges. The anticipated outcomes of the simplification process are several adjacency matrices, equal in number to the objectives Ω. We propose two methods for eliminating parallel edges: duplicating nodes and trimming edges.

4.1.1 Duplicating nodes

Algorithm I. ***Multi-objective Multigraph Simplification via Duplicating Nodes***

```
        /* EDGE SUPPLEMENT                                                                 */
 1      for $Parallel\ edge\ sets\ E_{m,n}\ of\ all\ node\ pairs$ do
 2          if $The\ number\ of\ parallel\ edges\ between\ nodes\ m\ and\ n\ is\ not\ \Delta$ then
 3              Add edge(s) to the node pair $(m,n)$ to make the number of parallel edges be equal to $\Delta$;
 4              Assign cost vector(s) randomly selected from the existing edge(s) between $m$ and $n$ to the new added edge(s);
 5          endif
 6      endfor
        /* NODE DUPLICATION                                                                */
 7      for $All\ objective\ IDs\ j$ do
            CREATE:
 8              one zero matrix $A$ with the dimensionality $\Delta|V_{MOMG}| \times \Delta|V_{MOMG}|$;
                /* Both the rows and columns of $A$ are indexed from 0.                        */
 9          for $All\ row\ IDs\ u\ in\ A$ do
10              for $All\ column\ IDs\ v\ in\ A$ do
11                  $A(u,v) \leftarrow COST\_exd^{(v\%\Delta+u\%\Delta)\%\Delta,j}_{u//\Delta,v//\Delta}$;
12              endfor
13          endfor
            OUTPUT:
14              $A$, to represent the adjacency matrix of the simplified graph with respect to objective $j$;
15      endfor
```

Algorithm I outlines the pseudocode for eliminating parallel edges via node duplication. A single node in the MOMG is duplicated into Δ new nodes, with the adjacent relationships of the newly generated nodes being consistent with those of the original node in the graph. In case where the number of parallel edges between a node pair is less than the maximum number Δ, new edges are added before the nodes are duplicated, and the costs of the newly added edges are determined by randomly selecting the existing costs of this node pair.

Each of the newly generated nodes holds the equal status as the original node, which means that in the weighted graph, each duplicated node should contain the same weighted adjacency information as the original node. Line 11 of Algorithm I achieves this by using the mapping calculation $A(u,v) \leftarrow COST\_exd^{(v\%\Delta+u\%\Delta)\%\Delta,j}_{u//\Delta,v//\Delta}$ to assign the original complete adjacency data to every new node. Here, $A(u,v)$ denotes the element in the u-th row and v-th column in the adjacent matrix of the simplified graph with dimension $\Delta|V_{MOMG}| \times \Delta|V_{MOMG}|$, $COST\_exd^{(v\%\Delta+u\%\Delta)\%\Delta,j}_{u//\Delta,v//\Delta}$ denotes the weight assigned to the j-th objective of the $[(v\%\Delta + u\%\Delta)\%\Delta]$-th parallel edge between nodes, whose IDs are $(u//\Delta)$ and $(v//\Delta)$, and this weight is picked from the extended costs $COST\_exd$ obtained via the "edge supplement" process of Algorithm I. The operator symbolised by "%" represents taking the reminder after division and "//" represents taking the quotient after division.

In more detail, as for the parallel edge ID mapping $(v\%\Delta + u\%\Delta)\%\Delta$, let us illustrate by the following example. Supposing the original node IDs are 0 and 1, $\Delta = 3$, and the edge costs are $[cost_0, cost_1, cost_2]$. Then the newly generated nodes should have IDs [0,1,2] and [3,4,5], which correspond to the original node IDs 0 and 1, respectively. The expected adjacent matrix after duplicating nodes is shown in Eq.(2). Parallel edge ID, and row and column in the adjacent matrix are all indexed from 0.

$$A = \begin{bmatrix} cost_0 & cost_1 & cost_2 \\ cost_1 & cost_2 & cost_0 \\ cost_2 & cost_0 & cost_1 \end{bmatrix} \tag{2}$$

When assigning value to the element $A(0,1)$, $v\%\Delta = 1\%3 = 1$ already locates the second parallel edge with cost $cost_1$. For $A(1,1)$, $u\%\Delta = 1$ should be added to $v\%\Delta = 1$ to locate the third edge with $cost_2$. As for $A(2,2)$, $v\%\Delta + u\%\Delta = 4$, which is beyond the edge index range. Therefore, this value needs to be taken as the reminder of its division by Δ, to locate the second edge cost. Fig.1 exemplifies this process of node duplication and cost assignment.

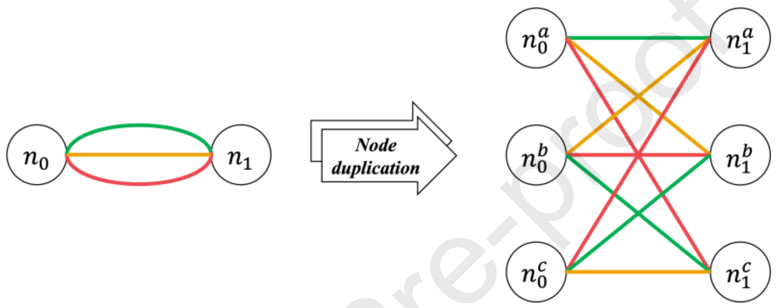


Figure 1 ***Example of node duplication and cost assignment for an undirected weighted node pair with three parallel edges****. The costs of the edges are indicated by different colours, each representing a different advantage. After duplication, every individual node is equipped with edges that have all three colours, preserving the original adjacent relationships within the node pair.*

4.1.2 Trimming edges

The pseudocode in Algorithm II for trimming edges is a more detailed version of the method introduced in [6]. After splitting objectives, this method prunes parallel edges between two nodes, only retaining the edge with the minimum cost.

Algorithm II. ***Multi-objective Multigraph Simplification via Trimming Edges***

```
     /* EDGE ELIMINATION                                                              */
1    for $All\ objective\ IDs\ j$ do
         CREATE:
2            one zero matrix $A$ with the dimensionality $|V_{MOMG}| \times |V_{MOMG}|$;
             /* Both the rows and columns of $A$ are indexed from 0.                  */
3        for $Parallel\ edge\ sets\ E_{m,n}\ of\ all\ node\ pairs$ do
4            Identify the minimum weight $cost_{min}$ among parallel edges between nodes $m$ and $n$ with respect to objective $j$;
5            $A(m,n) \leftarrow cost_{min}$;
6        endfor
         OUTPUT:
7            $A$, to represent the adjacency matrix of the simplified graph with respect to objective $j$;
8    endfor
```

4.2 Node embedding approach node2vec

Node2vec is a random-walk-based node embedding approach, which is inspired by the word2vec [55] developed for learning sentence-based word representations. Node2vec learns embeddings by predicting nearby nodes that co-occur in random walks, using an encoder called SkipGram [55]. In the context of the SOSG, node2vec treats each node as a "word" in a language modelling problem, and then generates a random walk for the node, resulting in a sampled node sequence analogous to a "sentence" in nature language processing. Eq.(3) describes the possibility to travel between two nodes, where r_t denotes the t-th sampled node on the random walk, π_{ij} denotes the unnormalised transition probability between nodes n_i and n_j, and Z is a normalising constant [24].

$$p(r_t = n_j | r_{t-1} = n_i) = \begin{cases} \frac{\pi_{ij}}{Z}, & if\ n_i\ and\ n_j\ are\ adjacent, \\ 0, & otherwise. \end{cases} \tag{3}$$

Especially, node2vec utilises a biased way to sample both local and global information in the graph. The walking direction is guided by two hyperparameters p and q, where p is the return parameter that indicates the probability of returning to a previously visited node, and q is the in-out parameter that indicates the probability of passing through an unseen part of the graph. These two hyperparameters allow the walk to interpolate between a pure breadth-first-search (BFS) that models structural equivalence by visiting a node's immediate neighbours and a pure depth-first-search (DFS) that reflects homophily by exploring nodes far away. The unnormalised transition probability π mentioned in Eq.(3) is calculated as Eq.(4), which employs a bias factor α_{pq} in conjunction with edge costs.

$$\pi_{i\delta} = \alpha_{pq}(n_{i-1}, n_\delta) \cdot cost_{i\delta}, \text{where } \alpha_{pq}(n_{i-1}, n_\delta) = \begin{cases} \frac{1}{p}, if\ Hop(n_{i-1}, n_\delta) = 0, \\ 1, if\ Hop(n_{i-1}, n_\delta) = 1, \\ \frac{1}{q}, if\ Hop(n_{i-1}, n_\delta) = 2. \end{cases} \tag{4}$$

where the random walk is assumed to have arrived at node n_i from previously visited node n_{i-1} and be deciding which nodes n_δ to visit, $\pi_{i\delta}$ denotes the unnormalised transition probability from n_i to n_δ, $cost_{i\delta}$ denotes the cost/weight of edge (n_i, n_δ), and $Hop(n_{i-1}, n_\delta)$ represents the minimum number of edges that need to be traversed from n_{i-1} to n_δ. This biased random walk schema is shown in Fig.2, in which n_δ may be one of four nodes n_{i+1}^a, n_{i+1}^b, n_{i+1}^c and n_{i-1}.

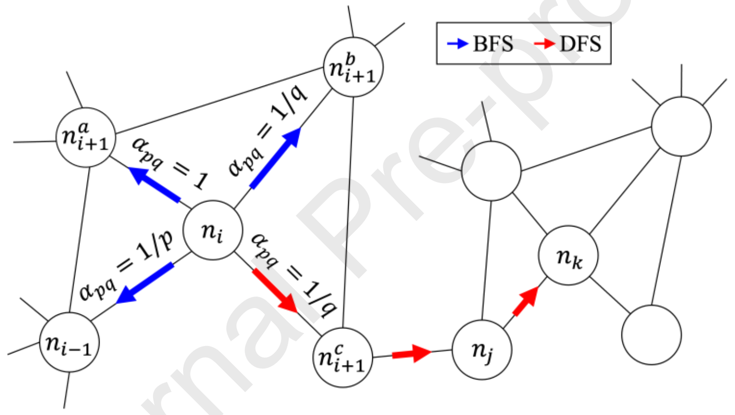


Figure 2 ***Biased random walk schema of node2vec****. The BFS cares more about the surrounding nodes (n_{i-1}, n_{i+1}^a, n_{i+1}^b, and n_{i+1}^c) of the currently residing node n_i, while the DFS enables the expansion of node n_k which has similar structure role as node n_i.*

5 Predictive Models and Benchmark Instance Generation

5.1 Regression predictive models

The model selection can impact the predictive accuracy. In this paper, three regression models are exploited, including MLP, PR and GBRT. Table 2 demonstrates the general comparison among these three models.

Table 2 ***Regression model comparison***

Models	**Parameter/coefficient interpretability**	**Handling non-linearity**	**Handling interactions between features**
MLP	Low. Often regarded as black boxes due to the complex structure. It can be challenging to interpret the learned weights and biases and understand the underlying relationships between the input features and output variable.	Yes, by employing activation functions that introduce non-linear transformations to inputs.	Yes, via hidden layers that allow for learning complex feature representations.
PR	High. The relationship between the input features and the target variable is expressed through polynomial terms with explicit coefficients. The coefficients represent the impact of each input feature on the target variable.	Yes, by introducing higher-degree polynomial terms, the model fits non-linear patterns.	Yes, by including interaction terms that multiply the input features together.
GBRT	Low. The ensemble of decision trees makes it challenging to interpret the overall model. However, feature importance measures can be derived to understand the relative importance of different input features.	Yes, by combining multiple decision trees.	Yes, by constructing a series of decision trees.

5.1.1 Multi-layer perceptron

Artificial neural networks (ANNs) are brain inspired models that enable a machine learning from available data [56]. MLP is a class of feedforward ANNs, consisting of at least three neuron layers, namely an input layer, one or several hidden layer(s), and an output layer. Neurons are connected between each pair of adjacent layers with different weights. Except for the input layer, each neuron has a nonlinear activation function. A three-layer MLP network can be expressed as Eq.(5).

$$\hat{y} = (XW_h + b_h)W_0 + b_0 \tag{5}$$

where $\hat{y}$ is the predicted value, X represents the input vector, W_h and b_h are the weights and biases of the hidden layer respectively, and W_0 and b_0 represent the weights and biases of the output layer.

5.1.2 Polynomial regression

PR is a statistical technique used in regression analysis to model relationships between a dependent variable and one or more independent variables by fitting a polynomial function of a specified degree to the data. Linear regression is a special case of PR, where the polynomial order is one. PR is useful for modelling curvilinear relationships, but it is susceptible to overfitting. Thus, it is important to select the appropriate degree of polynomial. The model with the N-th degree polynomial regression is:

$$\hat{y} = \beta + \sum_{i=1}^{N} \alpha_i X^i \tag{6}$$

where β is the regression coefficient, α_i is the corresponding coefficient vector to the i-th polynomial degree, and X^i is the i-th polynomial input features.

5.1.3 Gradient boosting regression tree

GBRT is a flexible non-parametric learning method for regression. It builds the model in a stage-wise fashion and allows optimisation of an arbitrary differentiable cost function [29]. Similar to other boosting techniques, gradient boosting combines weak leaners into a strong learner through an iterative process. At each stage s, a current predictive model F_s is updated by adding a new estimator $H_{estimator}$. The improved model F_{s+1} in the next stage can be constructed as $F_{s+1}(X) = F_s(X) + H_{estimator}$.

5.1.4 Performance metrics

Four metrics are provided to evaluate and compare prediction performance obtained by different feature extraction methods and predictive models. In the following equations, y denotes the actual value and $\hat{y}$ denotes the predicted value.

The coefficient of determination R^2. This metric indicates the dependent variable's fluctuations that can be explained by the independent variables. $R^2 \leq 1$, where 1 represents the best possible accuracy case. When the value is negative, the predictive model is attempting to fit nonlinear functions to the sampled data.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}\left(y_i - \frac{1}{n}\sum_{i=1}^{n} y_i\right)^2} \tag{7}$$

MAE. This metric measures the average absolute deviations between the predicted and actual shortest path costs.

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n}|y_i - \hat{y}_i|}{n} \tag{8}$$

Root mean squared error (RMSE). This metric is like MAE but assigns a greater weight to data points with larger errors.

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}} \tag{9}$$

MRE. As reported in [12], this metric is extensively used in the study of shortest path evaluation.

$$\mathrm{MRE} = \frac{\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|}{n} \tag{10}$$

The closer the last three metrics are to zero, the higher accuracy the embedding method or predictive model shows.

5.2 Benchmark instance generation

Six undirected weighted MOMGs are generated as benchmark instances. Node self-loops are not considered with the aim of conforming to the patterns observed in real-world airport taxiways. Each instance is a bi-objective search task, with up to 10 parallel edges for a pair of connected nodes (the node pairs have randomly different numbers of parallel edges). The six instances can be grouped according to the number of nodes: 500 nodes, 750 nodes, and 1000 nodes. Although two graphs in the same group have the same number of nodes, the way their nodes are connected and how their objectives are correlated is varied.

The objective correlation means that, for bi-objective costs in a set of parallel edges, objectives are positively correlated if the values of one objective vary positively with the other objective across all parallel edges, and vice versa for the negatively correlated. As for two graphs belonging to the same group in our case, one graph has objectives of positive correlation while the other graph has the negatively correlated objectives.

Benchmark instances are produced in two steps. First, an unweighted simple graph is generated using the "WAXMAN" graph tool of the Python module "networkX" [41], given the number of nodes. Second, a cost matrix is assigned to the edges of the unweighted simple graph, representing the weights of parallel edges connecting two nodes. Each row of this cost matrix corresponds to a parallel edge and has nondominated [1] costs compared to other rows. To generate costs for a row vector $(cost_1, cost_2)$ with positive correlation, $cost_1$ is randomly selected from a uniform distribution of the range $[cost_{\min}, cost_{\max}]$, and $cost_2$ is computed using Eq.(11), which is a convex combination of $cost_1$ and a randomly selected value $cost_2^*$ from the same distribution, where $\rho \in [0,1]$ is a correlation multiplier.

$$cost_2 = \rho cost_1 + (1-\rho) cost_2^* \tag{11}$$

Eq.(12) is used to calculate the $cost_2$ value that is negatively correlated with $cost_1$.

$$cost_2 = cost_{\min} + cost_{\max} - [\rho cost_1 + (1-\rho) cost_2^*] \tag{12}$$

The origin and destination nodes are designated as the two endpoints of the longest hop-count in the graph, in order to cover more potentially interesting cases. Between two nodes, the hop-count is the minimal number of edges that have to be traversed.

6 Experiments and Discussion

In this section, the complete results of predicting shortest path costs based on node physical patterns are presented in Section 6.1. In Section 6.2, we ran PCA on two benchmark instances as examples of feature selection, and then the effectiveness of the selected important features is presented on all the six benchmark instances. Section 6.3 shows the prediction performance comparison of two MOMG simplification methods.

We conducted experiments on a macOS 13 laptop with an Apple M2 chip and 16 GB of RAM. The codes were programmed in Python 3.8. Physical pattern values were computed via the "networkX" module [41]. We designed MLP as a network with two hidden layers, with the configurations suggested in [6]. The numbers of neurons in the hidden layers are 1024 and 64 respectively. The activation function of hidden layers is the rectified linear unit [57] that returns $f(x) = \max(0, x)$, and the output layer uses the "SoftPlus" [58] as its activation function that returns $f(x) = \log(\exp(x) + 1)$. The loss function employs MSE, and the solver is a stochastic gradient-based optimiser ADAM [59]. For PR, we set the polynomial degree to 2 and kept the remaining hyperparameters as the default values in the Python module "sklearn" [60]. For GBRT, the default hyperparameters in the "sklearn" module were used. 70% of all O.D. pairs are used for training, other 20% for validation and the remaining 10% for testing.

As a reminder, two feature vectors of an O.D. pair were first integrated by the binary operators shown in Table 1 before being used for the shortest path cost prediction.

6.1 Predicting based on node physical patterns

In the four graphs with 750 nodes and 1000 nodes, the calculation result of pattern "closeness vitality" contains infinite values, indicating that removing certain nodes causes the sum of the shortest distances between all other nodes to become excessively large. Since infinite values are not usable for regression, only the two graphs with 500 nodes include the "closeness vitality".

Table 3 shows the overall prediction results using all node physical patterns. Among the four binary operators, "addition" and "concatenation" consistently produce better performance. Additionally, we can observe that when using these two operators, the

[1] For two d-dimensional cost vectors $\overrightarrow{cost}$ and $\overrightarrow{cost}'$, $\overrightarrow{cost}$ is said to dominate $\overrightarrow{cost}'$, denoted as $\overrightarrow{cost} \prec \overrightarrow{cost}'$, when $\forall\, \overrightarrow{cost}, \overrightarrow{cost}' \in \mathbb{R}^d \quad \overrightarrow{cost} \prec \overrightarrow{cost}' \Leftrightarrow \forall j \quad cost_j \le cost_j' \wedge \overrightarrow{cost} \neq \overrightarrow{cost}'$, where $cost_j$ denotes the j-th element of $\overrightarrow{cost}$. By extension, a cost vector $\overrightarrow{cost}$ is nondominated in set Υ when $\nexists \overrightarrow{cost}' \in \Upsilon \quad \overrightarrow{cost}' \prec \overrightarrow{cost}$.

results obtained via the three predictive models do not exhibit much significant differences. Some negative R^2 values reflect that model fits nonlinear functions, in which cases we primarily evaluate the performance based on the other three metrics.

Table 3 ***Complete results for predicting shortest path costs based on all node physical patterns****. The results of the best performing binary operators are underlined, and the results of the best performing predictive models are both underlined and bolded.*

Graph nodes	Obj. cor relation	Obj. ID	Opr.	MLP				PR				GBRT			
				R^2	MAE	RMSE	MRE	R^2	MAE	RMSE	MRE	R^2	MAE	RMSE	MRE
500	Positive	0	⊕	**0.422**	**27.07**	**35.22**	**0.193**	0.406	27.41	35.48	0.196	0.382	27.42	35.53	0.197
			◨	0.494	29.63	38.11	0.201	0.420	27.38	35.31	0.195	0.405	27.32	35.30	0.196
			⊙	0.386	27.39	35.49	0.197	0.290	28.69	37.10	0.204	0.380	27.46	35.51	0.197
			⊖	0.219	30.81	39.60	0.217	-0.247	33.51	43.26	0.234	-0.631	33.48	43.10	0.234
		1	⊕	-0.908	32.00	41.35	0.192	**-1.15**	**31.78**	**41.07**	**0.191**	-1.33	31.89	41.24	0.192
			◨	-1.57	32.63	41.75	0.194	-1.06	31.85	41.12	0.191	-1.35	31.95	41.39	0.193
			⊙	-1.37	32.35	41.63	0.194	-0.935	32.53	42.20	0.196	-1.38	32.02	41.47	0.193
			⊖	-1.49	33.21	42.75	0.198	-2.16	33.73	43.45	0.201	-4.47	34.42	44.24	0.206
500	Negative	0	⊕	0.039	27.49	35.67	0.209	0.097	27.46	35.53	0.208	0.071	27.46	35.59	0.208
			◨	0.131	31.28	39.94	0.221	0.111	27.57	35.58	0.208	**0.093**	**27.45**	**35.48**	**0.208**
			⊙	-0.069	28.07	36.05	0.211	0.046	27.91	36.06	0.210	0.070	27.59	35.69	0.209
			⊖	-0.135	30.90	39.76	0.226	-0.947	32.32	41.63	0.239	-1.81	32.97	42.21	0.243
		1	⊕	-0.209	37.25	48.18	0.189	**-0.484**	**36.79**	**47.31**	**0.187**	-0.565	36.85	47.43	0.188
			◨	-0.302	40.49	52.16	0.203	-0.467	37.03	47.50	0.188	-0.539	36.99	47.55	0.189
			⊙	-0.414	36.90	47.83	0.190	-0.398	37.81	48.66	0.192	-0.574	37.03	47.67	0.189
			⊖	-0.784	39.26	50.52	0.199	-1.55	40.43	51.75	0.204	-3.23	41.67	53.03	0.210
750	Positive	0	⊕	0.682	35.61	46.45	0.180	0.660	35.80	46.76	0.182	0.644	36.15	47.37	0.184
			◨	0.643	39.57	51.04	0.191	**0.677**	**35.32**	**45.95**	**0.180**	0.671	35.34	46.03	0.180
			⊙	0.673	36.69	47.92	0.185	0.612	37.34	48.88	0.187	0.649	36.09	47.18	0.183
			⊖	0.503	44.36	57.61	0.217	0.068	49.46	64.56	0.240	-0.071	49.05	63.86	0.236
		1	⊕	-0.106	53.90	68.87	0.173	-0.115	53.50	69.12	0.175	-0.182	53.53	69.20	0.176
			◨	0.080	58.21	75.89	0.189	**-0.084**	**53.18**	**68.63**	**0.174**	-0.165	53.57	69.29	0.177
			⊙	-0.087	54.70	70.75	0.178	-0.172	53.81	69.63	0.177	-0.180	53.69	69.44	0.177
			⊖	-0.390	58.03	74.23	0.187	-1.14	61.31	78.76	0.198	-1.80	61.05	78.34	0.198
750	Negative	0	⊕	0.317	23.89	30.86	0.187	0.413	23.51	30.66	0.186	0.398	23.50	30.71	0.186
			◨	0.401	24.89	32.43	0.194	0.436	23.34	30.38	0.185	**0.430**	**23.27**	**30.32**	**0.185**
			⊙	0.400	24.11	30.75	0.185	0.359	24.07	31.59	0.189	0.409	23.45	30.62	0.186
			⊖	0.176	27.01	35.00	0.208	-0.448	29.98	39.01	0.228	-0.666	29.58	38.43	0.225
		1	⊕	-0.553	31.64	40.98	0.178	**-0.711**	**31.46**	**40.81**	**0.178**	-0.787	31.47	40.86	0.179
			◨	-0.602	33.40	43.44	0.187	-0.684	31.50	40.84	0.178	-0.755	31.50	40.90	0.179
			⊙	-0.657	31.32	41.31	0.181	-0.778	31.59	41.01	0.179	-0.777	31.50	40.91	0.179
			⊖	-0.939	34.68	44.68	0.193	-2.46	35.12	45.26	0.196	-3.98	35.18	45.26	0.197
1000	Positive	0	⊕	0.778	36.21	46.94	0.171	0.769	36.65	48.28	0.174	0.745	37.14	49.50	0.175
			◨	0.778	36.84	48.26	0.176	0.790	35.58	46.41	0.171	**0.791**	**35.20**	**46.04**	**0.170**
			⊙	0.789	35.92	47.51	0.173	0.678	39.99	55.24	0.183	0.770	36.42	47.93	0.173
			⊖	0.678	42.39	55.14	0.194	0.336	54.54	72.90	0.245	0.276	50.91	68.94	0.227
		1	⊕	0.558	36.95	48.34	0.162	0.486	37.11	48.21	0.162	0.472	37.11	48.22	0.163
			◨	0.498	38.46	50.17	0.167	**0.497**	**36.86**	**47.79**	**0.161**	0.478	37.06	48.17	0.163

Table 3 ***Continued***

Graph nodes	Obj. cor relation	Obj. ID	Opr.	MLP				PR				GBRT			
				R^2	MAE	RMSE	MRE	R^2	MAE	RMSE	MRE	R^2	MAE	RMSE	MRE
1000	Positive	1	⊙	0.517	36.77	48.43	0.163	0.409	40.18	52.87	0.175	0.473	37.17	48.32	0.163
			⊖	0.402	40.02	51.20	0.171	0.151	43.47	56.29	0.187	0.022	42.95	55.61	0.185
	Negative	0	⊕	0.707	34.62	45.28	0.169	0.677	34.99	45.97	0.172	0.663	35.19	46.32	0.172
			◨	0.677	35.88	46.80	0.172	**0.697**	**34.20**	**44.75**	**0.169**	0.692	34.25	44.92	0.169
			⊙	0.696	34.26	45.33	0.170	0.607	38.20	50.74	0.183	0.672	34.99	45.93	0.172
			⊖	0.560	41.00	54.00	0.197	0.005	50.19	65.80	0.233	-0.114	48.89	64.60	0.226
		1	⊕	**0.515**	**26.68**	**34.64**	**0.164**	0.436	26.96	35.01	0.167	0.416	26.90	34.90	0.167
			◨	0.468	27.59	36.01	0.169	0.454	26.66	34.48	0.165	0.434	26.79	34.72	0.167
			⊙	0.445	26.83	34.89	0.167	0.349	27.85	36.61	0.171	0.425	26.91	34.90	0.167
			⊖	0.273	28.82	37.28	0.177	0.035	31.33	40.56	0.190	-0.114	31.08	40.32	0.189

6.2 Node physical pattern selection

According to Table 3, only the data generated via the operators "addition" and "concatenation" were selected to reveal the importance of the integrated features. For the sake of reducing workload, we randomly chose the graph with 500 nodes and negatively correlated objectives (denoted as 500_Nega) and the graph with 750 nodes and positively correlated objectives (denoted as 750_Posi) to operate PCA.

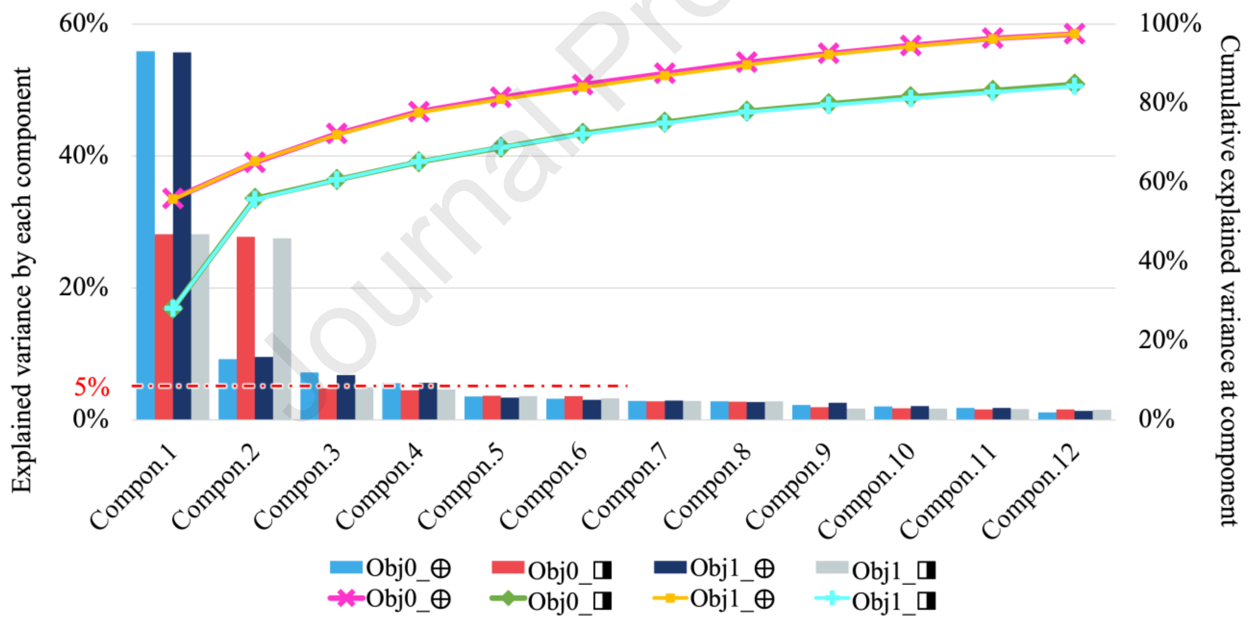


Figure 3 ***Percentage of variance explained by each component of PCA: an example in the benchmark MOMG with 500 nodes and negatively correlated objectives****. Node physical patterns of O.D. pairs integrated via "addition" and "concatenation" operators are evaluated, and twelve components are selected. The bars demonstrate the percentage of variance explained by each selected component while the lines show the cumulative percentage of explained variance at each component.*

Fig. 3 presents the PCA-explained variance against different number of components on graph 500_Nega. The contributions mainly come from the first 4 components, where explained variance is more than 5% out of the whole data.

Table 4 demonstrates the prediction performance of features roughly selected via three methods of combining components. As discussed in the end of Section 3.2, a challenge in PCA-based feature selection is determining the number of components. In Table 4, we compare the prediction results obtained from considering only the first component versus using multiple components. Note that, to avoid interference from less important components, we only take components with explained variance larger than 5% into account. Results show that features selected via considering multiple components predict better than single component.

Table 4 ***MLP prediction results for roughly selected features of two example graphs****. "First component" means that the features are selected based on their coefficients on the first component, "components added equally" denotes that the components with an explained variance larger than 5% are added equally to obtain the final feature coefficients for further selection, and "components added with weights" indicates that the feature coefficients on the components with explained variance larger than 5% are added with weights (weights are the explained variance of components). The results of the best performing component combination are underlined. The percentage values below the underlined values show the performance differences compared to the initial prediction performance generated by using all physical patterns in Table 3.*

Graph nodes	Obj. cor relation	Obj. ID	Opr.	First component				Components added equally				Components added with weights			
				R^2	MAE	RMSE	MRE	R^2	MAE	RMSE	MRE	R^2	MAE	RMSE	MRE
500	Negative	0	⊕	0.041	27.77	36.02	0.210	0.0165	27.41 0.291%↑	36.08 1.15%↓	0.213 1.91%↓	-1.67	34.12	43.60	0.251
			◨	-0.266	30.01	38.34	0.223	-0.124	29.76	37.76	0.218	-0.038	28.65 8.41%↑	36.83 7.79%↑	0.214 3.17%↑
		1	⊕	-1.17	37.78	48.71	0.194	-0.739	37.52 0.725%↓	47.93 0.519%↑	0.190 0.529%↓	-1.17	37.78	48.71	0.194
			◨	-1.45	40.57	51.45	0.202	-0.692	37.65 7.01%↑	48.47 7.07%↑	0.193 4.93%↑	-0.866	37.78	48.62	0.194
750	Positive	0	⊕	0.643	36.92	47.85	0.184	0.668	35.98 1.04%↓	47.49 2.24%↓	0.184 2.22%↓	0.0539	49.79	63.79	0.240
			◨	0.644	38.34	49.03	0.189	0.675	35.14 11.2%↑	46.18 9.52%↑	0.181 5.24%↑	0.675	35.14	46.18	0.181
		1	⊕	-0.173	53.97	69.87	0.178	-0.332	53.61 0.538%↑	69.27 0.581%↓	0.177 2.31%↓	-0.173	53.97	69.87	0.178
			◨	-0.095	54.21	70.44	0.179	-0.058	54.19	70.05	0.178	-0.044	53.39 8.28%↑	69.43 8.51%↑	0.176 6.88%↑

During tests, we noticed that different objectives and graphs yielded varying results in feature selection. Therefore, Table 4 represents the first selection step to produce rough results, where the top 20 important features were selected when using the operator "addition", while the top 40 were selected regarding the operator "concatenation". Table 5 displays the interactions among the features that were roughly selected in Table 4. There are the 12 most important features finally selected when using the operator "addition" and the 30 most important features regarding the "concatenation". It is worth focusing that some selected features in Table 5 are of high computational complexity, such as "closeness centrality", "load centrality" and "eccentricity".

Table 5 ***Selected important features****. Features are sorted by their ID order, consistent with Table 2. The features selected both by operators ⊕ and ◨ are underlined and bolded. Note that when integrating feature vectors of a node pair via ◨, the node with the larger ID is placed at the tail.*

Operator ⊕		Operator ◨			
No.	Pattern name	No.	Pattern name	No.	Pattern name
2	Average neighbour degree †*	1	Degree (O.†*, D.†)	**9**	**Second order centrality (O., D.)**
3	**Degree centrality**	**3**	**Degree centrality (O., D.)**	**10**	**Harmonic centrality (O.†, D.†)**
5	**Closeness centrality †***	4	Eigenvector centrality (O.†, D.†*)	14	Node connectivity (D.)
6	**Current-flow closeness centrality †**	**5**	**Closeness centrality (O.†*, D.†)**	**19**	**Page rank (O.†*, D.†*)**
9	**Second order centrality**	**6**	**Current-flow closeness centrality (O.†*, D.†)**	20	Constraint (D.†)
10	**Harmonic centrality †**			21	Effective size (O.†*, D.*)
16	Periphery	7	Current-flow betweenness centrality (O.†, D.†)	/	
17	Eccentricity				
19	**Page rank †***	8	Load centrality (O.†)		

The physical patterns marked with "†" and "*" are the ones that can process edge weights. "†" marks the pattern for which the value computed without edge weights is selected, while "*" marks the pattern for which the value computed with edge weights is selected.
"O." and "D." denote the origin and the destination of a node pair, respectively.

Using the finally selected features in Table 5 for prediction, the results generated by MLP are listed in Table 6. Compared to the initial prediction performance shown in Table 3, the selected features can achieve nearly equivalent or even slightly better accuracy levels.

*Table 6 **MLP prediction results of finally selected features**. Those percentage values in the second lines in MAE, RMSE and MRE are the performance differences compared to the initial prediction performance generated by using all physical patterns in Table 3.*

Graph nodes	Objective Correlation	Obj. ID	Operator ⊕				Operator ◨			
			R^2	MAE	RMSE	MRE	R^2	MAE	RMSE	MRE
500	Positive	0	0.4194	27.89 3.029% ↓	36.34 3.180% ↓	0.2000 3.627% ↓	0.3034	29.02 2.059% ↑	37.30 2.125% ↑	0.2051 2.040% ↓
		1	-1.248	32.01 0.0313% ↓	41.85 1.209% ↓	0.1951 1.615% ↓	-2.668	33.81 3.616% ↓	43.44 4.048% ↓	0.2028 4.536% ↓
	Negative	0	0.05075	27.61 0.4365% ↓	36.17 1.402% ↓	0.2119 1.388% ↓	-0.009177	29.31 6.298% ↑	37.29 6.635% ↑	0.2151 2.670% ↑
		1	-0.6245	37.57 0.8591% ↓	47.98 0.4151% ↑	0.1898 4.233% ↓	-0.7861	37.73 6.816% ↑	48.88 6.288% ↑	0.1949 3.990% ↑
750	Positive	0	0.6299	36.81 3.370% ↓	48.04 3.423% ↓	0.1855 3.05% ↓	0.6291	36.76 7.101% ↑	47.84 6.270% ↑	0.1862 2.513% ↑
		1	0.00722	53.75 0.2783% ↑	70.57 2.468% ↓	0.1791 3.526% ↓	-0.2418	54.16 6.958% ↑	70.57 7.010% ↑	0.1800 4.762% ↑
	Negative	0	0.4637	23.58 1.298% ↑	31.12 0.8425% ↓	0.1872 0.1070% ↓	0.3515	24.60 1.165% ↑	31.75 2.097% ↑	0.1922 0.9278% ↑
		1	-0.6378	31.33 0.9798% ↑	41.04 0.1464% ↓	0.1795 0.8427% ↓	-0.7757	32.13 3.802% ↑	41.78 3.821% ↑	0.1823 2.513% ↑
1000	Positive	0	0.7454	37.12 2.513% ↓	48.88 4.133% ↓	0.1750 2.339% ↓	0.7640	37.09 0.6786% ↓	48.41 0.3108% ↓	0.1775 0.8523% ↓
		1	0.4862	38.09 3.085% ↓	48.73 0.8068% ↓	0.1637 1.049% ↓	0.4508	37.41 2.730% ↑	49.07 2.193% ↑	0.1666 0.2395% ↑
	Negative	0	0.6721	35.25 1.820% ↓	46.66 3.048% ↓	0.1735 2.663% ↓	0.6716	36.32 1.226% ↓	46.73 0.1496% ↑	0.1737 0.9884% ↓
		1	0.3807	27.40 2.699% ↓	35.35 2.050% ↓	0.1687 2.866% ↓	0.4703	27.68 0.3262% ↓	35.73 0.7776% ↑	0.1700 0.5917% ↓

6.3 Predicting based on node embeddings

*Table 7 **Results for predicting shortest path costs based on node embeddings using the MOMG simplification method trimming edges***

Graph nodes	Obj. relation	Obj. ID	MLP with the operator ⊕				PR with the operator ⊕				GBRT with the operator ⊕			
			R^2	MAE	RMSE	MRE	R^2	MAE	RMSE	MRE	R^2	MAE	RMSE	MRE
500	Positive	0	0.7719	20.95	26.46	0.1215	0.5456	25.88	33.11	0.1716	-8.535	38.99	50.98	0.2688
		1	0.5897	24.24	30.64	0.1226	0.2267	26.75	33.68	0.1459	-14.56	35.56	45.69	0.2110
	Negative	0	0.7402	19.44	24.95	0.1213	0.3603	25.83	33.18	0.1817	-8.191	35.06	45.29	0.2589
		1	0.6244	28.80	36.38	0.1233	0.2833	32.37	40.81	0.1525	-10.90	43.08	55.23	0.2164
750	Positive	0	0.8770	24.19	31.63	0.0991	Out of memory, killed by Python.				-8.227	60.11	82.59	0.2829
		1	0.7536	36.07	45.80	0.1007					-9.114	65.33	84.89	0.2096
	Negative	0	0.7838	17.68	22.54	0.1148					-10.30	34.30	45.28	0.2584
		1	0.5915	24.88	31.43	0.1202					-26.53	37.60	48.35	0.2090
1000	Positive	0	0.9129	24.98	32.52	0.0992					-7.932	66.13	98.03	0.2843
		1	0.8436	24.61	31.51	0.0923					-13.57	50.92	75.36	0.2101
	Negative	0	0.8845	23.59	30.46	0.0949					-6.949	60.54	81.26	0.2748
		1	0.8278	17.89	22.98	0.0930					-11.84	36.23	52.76	0.2122

Tables 7 and 8 show the results of predicting shortest path costs based on node embeddings using the proposed two MOMG simplification methods. To obtain the best embeddings-based prediction performance, we followed the experiment settings reported in [6], which involves not utilising edge weights for node2vec, setting node2vec hyperparameters $p = 0.3$ and $q = 2$, and employing the operator "addition". The results indicate that (i) the simplification method duplicating nodes results in better predictive accuracy on all benchmark instances, and (ii) MLP always outperforms. And the comparison of Tables 6, 7, and 8 reveals that when using MLP, the node features extracted using the learning-based extraction method consistently yield better prediction results than the statistics-based extraction method.

Table 8 ***Results for predicting shortest path costs based on node embeddings using the MOMG simplification method duplicating nodes***

Graph nodes	**Obj. relation**	**Obj. ID**	**MLP with the operator ⊕**				**PR with the operator ⊕**				**GBRT with the operator ⊕**			
			R^2	**MAE**	**RMSE**	**MRE**	**R^2**	**MAE**	**RMSE**	**MRE**	**R^2**	**MAE**	**RMSE**	**MRE**
500	Positive	0	0.7940	19.63	25.04	0.1192	0.5606	25.65	32.74	0.1683	-7.445	38.82	50.97	0.2666
		1	0.6579	22.05	27.67	0.1104	0.3560	25.15	31.78	0.1352	-18.99	35.90	46.04	0.2119
	Negative	0	0.7573	18.86	24.14	0.1176	0.3954	25.46	32.48	0.1759	-9.064	35.40	45.78	0.2603
		1	0.6853	26.08	32.90	0.1124	0.3745	30.88	38.98	0.1438	-11.94	43.08	55.24	0.2156
750	Positive	0	0.8932	23.01	29.51	0.0958	Out of memory, killed by Python.				-7.358	59.95	82.25	0.2823
		1	0.7838	33.88	42.90	0.0947					-11.32	65.63	85.23	0.2098
	Negative	0	0.7959	17.30	22.02	0.1124					-11.08	34.47	45.84	0.2597
		1	0.6421	23.42	29.70	0.1111					-22.96	37.51	48.16	0.2080
1000	Positive	0	0.9142	24.32	31.58	0.0947					-7.251	66.39	97.83	0.2855
		1	0.8596	23.56	30.12	0.0881					-14.17	51.61	75.64	0.2128
	Negative	0	0.8993	22.64	29.07	0.0909					-7.345	61.09	82.12	0.2776
		1	0.8453	16.85	21.51	0.0879					-12.85	36.38	53.05	0.2127

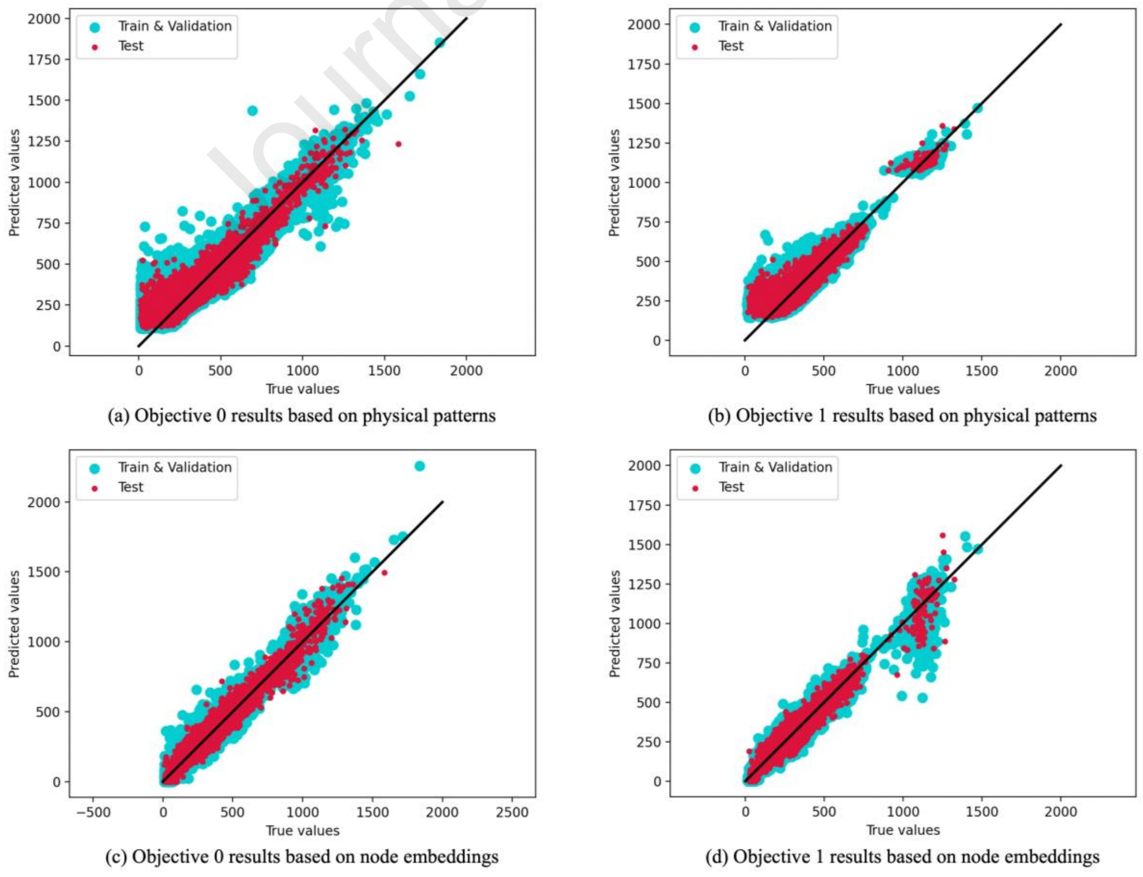


Figure 4 ***MLP-predicted shortest path costs against the actual values: an example in the benchmark MOMG with 1000 nodes and positively correlated objectives****. Subgraphs (a) and (b) depict the predicted results based on the finally selected important physical patterns,*

while (c) and (d) depict the results based on the node embeddings using the MOMG simplification method that trims edges. Overestimation happens severer in small-distance areas in subgraphs (a) and (b).

Fig. 4 takes the benchmark instance 1000_Posi as an example to illustrate the predicted shortest path costs plotted against the actual costs. Note that, according to Table 6, the graph 1000_Posi provides the best R^2 when using the MLP prediction based on the finally selected physical patterns, where the linear relationship between the predicted and actual values is fitted the best. However, even in this case, overestimation is obviously more pronounced in small distance values when using physical patterns compared to using node embeddings. We observed that overestimation is much worse when predicting based on physical patterns as for other benchmark instances.

7 Conclusions

This paper paid attention to the MOMG search problem, which serves as a conceptual prototype for AAGM. To efficiently obtain single-objective shortest path costs for each objective in MOMG, which are heuristic information for MOMG heuristic search algorithms, we proposed to predict the costs via regression models and investigated two node feature extraction methods: statistics-based extraction and learning-based extraction. For the former extraction method, we provided a novel collection of node physical patterns in benchmark multigraphs, and identified the top important patterns that are able to deliver the nearly equivalent or slightly better predictive accuracy compared to the total set of patterns. For the latter extraction method, we proposed two methods to simplify MOMGs to satisfy the requirements of node embedding technique, since almost all node embedding algorithms can only deal with the adjacency information in the format of SOSGs.

Our experiment results suggest that the learning-based node feature extraction with the simplification method that trims edges is the most appropriate approach, as it can reach a balance between predictive accuracy and computational complexity. Compared to the finally selected important node physical patterns generated by the statistics-based extraction, the recommended approach offers significantly outperforming prediction results, besides, it is noteworthy that some selected node physical patterns still have undeniably high computational complexity. Compared to the simplification method that duplicates nodes, the distance prediction results of the recommended approach are slightly worse. However, when the simplification method that trims edges is employed, the number of nodes in the simplified graphs remains the same as the original MOMG. This characteristic is advantageous for the efficient processing of node embedding algorithms, particularly in scenarios where the maximum number of parallel edges in MOMG is extremely large.

For the future work, these directions deserve more focuses: (i) the exploration of additional node physical patterns. (ii) the development of mechanism to handle the overestimation of small distances when using node physical patterns to predict shortest path costs. (iii) the fine-tuning of hyperparameters for PR and GBRT. (iv) the conduct of further research and experimentation on more regression models to evaluate their performance of predicting shortest path costs. (v) the investigation of the node2vec hyperparameter that controls the number of random walks generated for each node, particularly in relation to the simplification method that duplicates nodes. Even though the duplicated nodes have the same adjacent status as the initial nodes in the graph and do not cause undersampling, increasing the number of walks may help node2vec capture the location property of duplicated nodes more accurately. And (vi) the application of the proposed methods to real-world AAGM cases, incorporating techniques to handle constraints encountered in actual operations, such as the time-varying availability of airport taxiways.

Appendix

Table ***Node physical patterns***

Category	No.	Pattern name	Description
Centrality	1	Degree *	Unweighted node degree is the number of edges adjacent to this node. Weighted node degree is the sum of the edge weights for edges incident to this node.
	2	Average neighbour degree *	In an unweighted graph, average neighbour degree of node i is defined as: $\frac{1}{\lvert N(i)\rvert}\sum_{j\in N(i)} k_j$, where $N(i)$ are node i's neighbours and k_j is node j's degree. While in a weighted graph [32], that for node i is defined as: $\frac{1}{s_i}\sum_{j\in N(i)} w_{ij}k_j$, where s_i denotes the weighted degree of node i and w_{ij} is the weight of edge connecting nodes i and j.
	3	Degree centrality	The degree centrality for a node is the fraction of nodes it is connected to.
Centrality	4	Eigenvector centrality *	Eigenvector centrality [33] computes the centrality for a node based on the centrality of its neighbours. The eigenvector centrality for node i is the i-th element of the vector x, which is defined as $Ax = \lambda x$, where A is the adjacency matrix of the graph with eigenvalue λ.
	5	Closeness centrality *	Closeness centrality [34] of a node i is the reciprocal of the average shortest path distance to i over all $n-1$ reachable nodes, defined as: $\frac{n-1}{\sum_{j=1}^{n-1} dist(i,j)}$, where $dist(i,j)$ is the shortest-path distance between i and j, and $n-1$ is the number of nodes reachable from i. The higher values of closeness indicate higher centrality.
	6	Current-flow closeness centrality *	Current-flow closeness centrality is variant of closeness centrality based on effective resistance between nodes in a network [35]. This metric is also known as information centrality [36].
	7	Current flow betweenness centrality *	Also known as random-walk betweenness centrality [37]. This metric uses an electrical current model [38] for information spreading in contrast to betweenness centrality which uses shortest paths.
	8	Load centrality *	The load centrality of a node is the fraction of all shortest paths that pass through it [38].
	9	Second order centrality	The standard deviation of the return times to a node of a perpetual random walk on graph [39].
	10	Harmonic centrality *	For node i, harmonic centrality [40] is the sum of the reciprocal of the shortest-path distances from all other nodes, defined as $\sum_{i\neq j}\frac{1}{dist(i,j)}$, where $dist(i,j)$ is the shortest distance between i and j.
Clique	11	Node clique number	The size of the largest maximal clique containing each node [41].
Clustering	12	Square clustering	For node j, this metric [42] returns the fraction of possible squares that exist at the node, $\frac{\sum_{i=1}^{deg(j)}\sum_{k=i+1}^{deg(j)} q_j(i,k)}{\sum_{i=1}^{deg(j)}\sum_{k=i+1}^{deg(j)}[a_j(i,k)+q_j(i,k)]}$, where $deg(j)$ is the degree of node j, $q_j(i,k)$ is the number of common neighbours of i and k other than j ($i.e.$, squares), $a_j(i,k) = \big(deg(i) - (1 + q_j(i,k) + \theta_{ij})\big) + \big(deg(k) - (1 + q_j(i,k) + \theta_{ik})\big)$, and $\theta_{ik} = 1$ if i and k are connected and 0 otherwise.
Communities	13	Kernighan-Lin bisection *	A graph is partitioned into two blocks using the Kernighan-Lin algorithm [43]. Nodes in one section are labelled 0, and 1 if they are in the other section.
Connectivity	14	Node connectivity	The minimum number of nodes that need to be removed to disconnect all paths to the investigated node [44].
Distance measures	15	Centre	The centre is the set of nodes with eccentricity equal to radius [41]. Here, nodes in the set are labelled 1, or 0 otherwise.
	16	Periphery	The periphery is the set of nodes with eccentricity equal to the diameter [41]. Here, nodes in the set are labelled 1, or 0 otherwise.
	17	Eccentricity	The eccentricity of a node is the maximum distance from it to all other nodes [41].
Dominating sets	18	Dominating set	A dominating set for a graph with node set V is a subset D of V such that every node not in D is adjacent to at least one member of D [44]. Here, nodes in the set are labelled 1, or 0 otherwise.
Link analysis	19	Page rank *	This metric computes a ranking of nodes based on the structure of the incoming links. It was originally designed as an algorithm to rank web pages [45].
Structural holes	20	Constraint *	This metric quantifies the extent that node i is invested in the nodes [46], which are themselves invested in the neighbours of i. The constraint on node i is defined as: $\sum_{j\in N(i)\setminus\{i\}} \ell oc(i,j)$, where $N(i)$ is the subset of the neighbours of i that are either predecessors or successors of i and $\ell oc(i,j)$ is the local constraint [46] on i with respect to j.
	21	Effective size *	The effective size of a node's ego network is based on the concept of redundancy. A person's ego network has redundancy to the extent that her/his contacts are connected to each other as well. The nonredundant part of a person's relationships is the effective size of her/his ego network [47]. Then, the effective size of node i is defined as: $\sum_{j\in N(i)\setminus\{i\}}\big(1 - \sum_{k\in N(j)} p_{ik}m_{jk}\big)$, where $N(i)$ is the set of neighbours of i, p_{ik} is the normalised mutual weight of i and k by i highest mutual weight with any of its neighbours, and m_{jk} is the mutual weight of j and k. The mutual weight of two nodes is the sum of the weights of edges joining them.
Vitality	22	Closeness vitality *	The closeness vitality of a node is the change in the sum of distances between all node pairs when excluding that node [48].

* In calculation, edge weights can be used or not, $i.e.$, these asterisked metrics have two potential values: one computed with weights while the other not.

References

[1] J. Chen, M. Weiszer, P. Stewart, et al., “Towards a more realistic, cost effective, and greener ground movement through active routing: part 1 - Optimal speed profile generation,” *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 5, pp. 1196-1209, 2016. DOI: 10.1109/TITS.2015.2477350

[2] J. Chen, M. Weiszer, P. Stewart, et al., “Towards a more realistic, cost-effective, and greener ground movement through active routing: part 2 - A multi-objective shortest path approach,” *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 12, pp. 3524-3540, 2016. DOI: 10.1109/TITS.2016.2587619

[3] X. Wang, A. E. I. Brownlee, M. Weiszer, et al, “An interval type-2 fuzzy logic-based map matching algorithm for airport ground movements,” *IEEE Trans. Fuzzy Syst.*, vol. 31, no. 2, pp. 582-595, 2023. DOI: 10.1109/TFUZZ.2022.3221793

[4] M. Weiszer, E. K. Burke, J. Chen, “Multi-objective routing and scheduling for airport ground movement,” *Transp. Res. C: Emerg. Technol.*, vol. 119, pp. 102734-102756, 2020. DOI: 10.1016/j.trc.2020.102734

[5] P. Serafini, “Some considerations about computational complexity for multi-objective combinatorial problems,” in *Proc. Int. Conf. on Vector Optimisation*, Aug. 1986, pp. 222-232. DOI: 10.1007/978-3-642-46618-2_15

[6] S. Liu, J. Chen, M. Weiszer, “Multi-objective multigraph A* search with learning heuristics based on node metrics and graph embedding,” in *Proc. IEEE 11th Int. Conf. Intell. Syst.*, Oct. 2022, pp. 186-193. DOI: 10.1109/IS57118.2022.10019653

[7] L. Fu, D. Sun, L. R. Rilett, “Heuristic shortest path algorithms for transportation applications: State of the art,” *Comput. Oper. Res.*, vol. 33, pp. 3324-3343, 2006. DOI: 10.1016/j.cor.2005.03.027

[8] C. T. Tung, K. L. Chew, “A multicriteria pareto-optimal path algorithm,” *Eur. J. Oper. Res.*, vol. 62, no. 2, pp. 203-209, 1992. DOI: 10.1016/0377-2217(92)90248-8

[9] E. W. Dijkstra, “A note on two problems in connexion with graphs,” *Numerische Mathematik*, vol. 1, no. 1, pp. 269-271, 1959. DOI: 10.1007/BF01386390

[10] P. E. Hart, N. J. Nilsson, B. Raphael, “A formal basis for the heuristic determination of minimum cost paths,” *IEEE Trans. Syst. Sci. Cybern.*, vol. 4, no. 2, pp. 100-107, 1968. DOI: 10.1109/TSSC.1968.300136

[11] R. W. Floyd, “Algorithm 97: shortest path,” *Commun. ACM*, vol. 5, no. 6, pp. 345, 1962. DOI: 10.1145/367766.368168

[12] F. S. Rizi, J. Schloetterer, M. Granitzer, “Shortest path distance approximation using deep learning techniques,” in *2018 IEEE/ACM Int. Conf. Adv. Soc. Netw. Anal. and Min. (ASONAM)*, Aug. 2018, pp. 1007-1014. DOI: 10.1109/ASONAM.2018.8508763

[13] S. Chechik, “Approximate distance oracles with improved bounds,” in *Proc. 47th Annu. ACM Symp. Theory Computing*, Jun. 2015, pp. 1-10. DOI: 10.1145/2746539.2746562

[14] J. Sankaranarayanan, H. Samet, “Distance oracles for spatial networks,” in *IEEE 25th Int. Conf. Data Engineering*, Mar. 2009, pp. 652-663. DOI: 10.1109/ICDE.2009.53

[15] M. Thorup, U. Zwick, “Approximate distance oracles,” *J. ACM*, vol. 52, no. 1, pp. 1-24, 2005. DOI: 10.1145/1044731.1044732

[16] M. Potamias, F. Bonchi, C. Castillo, et al.,“Fast shortest path distance estimation in large networks,” in *Proc. 18th ACM Conf. Info. Knowl. Mgmt.*, Nov. 2009, pp. 867-876. DOI: 10.1145/1645953.1646063

[17] J. Qi, W. Wang, R. Zhang, et al., “A Learning Based Approach to Predict Shortest-Path Distances,” in *Proc. Int. Conf. Extd. Database Technol.*, 2020, pp. 367-370.

[18] X. Chen, S. Wang, H. Li, et al., “Ndist2vec: node with landmark and new distance to vector method for predicting shortest path distance along road networks,” *ISPRS Int. J. Geo-Inf.*, 11, 514, pp. 1-12, 2022. DOI: 10.3390/ijgi11100514

[19] D. Brunner, “Distance preserving graph embedding,” B.S. thesis, ETH Zürich, Zurich, 2021. [Online]. Available: https://pub.tik.ee.ethz.ch/students/2021-FS/BA-2021-17.pdf

[20] L. H. S. Lelis, R. T. Stern, A. Felner, et al., “Predicting optimal solution cost with conditional probabilities,” *Ann. Math. Artif. Intell.*, vol. 72, pp. 267-295, 2014. DOI: 10.1007/s10472-014-9432-8

[21] P. Goyal, E. Ferrara, “Graph embedding techniques, applications, and performance: a survey,” *Knowl. Based Syst.*, vol. 151, pp. 78-94, 2018. DOI: 10.1016/j.knosys.2018.03.022

[22] W. L. Hamilton, R. Ying, J. Leskovec, “Representation learning on graphs: methods and applications,” *ArXiv* abs/1709.05584, 2017, n. pag. DOI: /10.48550/arXiv.1709.05584

[23] S. Cao, W. Lu, Q. Xu, “GraRep: learning graph representations with global structural information,” in *Proc. 24th ACM Int. Conf. Info. Knowl. Mgmt.*, Oct. 2015, pp. 891-900. DOI: 10.1145/2806416.2806512

[24] A. Grover, J. Leskovec, “Node2vec: Scalable feature learning for networks,” in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, Aug. 2016, pp. 855-864. DOI: 10.1145/2939672.2939754

[25] J. Zhang, Y. Dong, Y. Wang, et al., “ProNE: fast and scalable network representation learning,” in *Proc. 28th Int. Joint Conf. Artif. Intell.*, 2019, pp. 4278-4284. DOI: 10.24963/ijcai.2019/594

[26] F. Song, Z. Guo, D. Mei, “Feature selection using principal component analysis,” in *Proc. Int. Conf. Syst. Sci., Eng. Design Manuf. Informatisation*, Nov. 2010, pp. 27-30. DOI: 10.1109/ICSEM.2010.14

[27] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Hoboken, New Jersey: Prentice Hall PTR, 1994. Accessed: Mar. 30, 2023. [Online]. Available: https://dl.acm.org/doi/abs/10.5555/521706

[28] X. Wang, A. E. I. Brownlee, J. R. Woodward, et al., “Aircraft taxi time prediction: feature importance and their implications,” *Transp. Res. C: Emerg. Technol.*, vol. 124, pp. 102892-102914, 2021. DOI: 10.1016/j.trc.2020.102892

[29] J. H. Friedman, “Stochastic gradient boosting,” *Compt. Stat. Data Anal.*, vol. 38, no. 4, pp.367-378, 2002. DOI: 10.1016/S0167-9473(01)00065-2

[30] M. Nickel, D. Kiela, “Poincaré embeddings for learning hierachical representations,” in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, Dec. 2017, pp. 6341-6350. DOI: 10.48550/arXiv.1705.08039

[31] J. Zhou, G. Cui, S. Hu, et al., “Graph neural networks: a review of methods and applications,” *AI Open*, vol. 1, pp. 57-81, 2020. DOI: 10.1016/j.aiopen.2021.01.001

[32] A. Barrat, M. Barthélemy, R. Pastor-Satorras, et al., “The architecture of complex weighted netwroks,” *PNAS*, vol. 101, no. 11, pp. 3747-3752, 2004. DOI: 10.1073/pnas.0400087101

[33] P. Bonacich, “Power and centrality: a family of measures,” *Am. J. Sociol.*, vol. 92, no. 5, pp. 1170–1182, 1986.

[34] L. C. Freeman, “Centrality in social networks conceptual clarification,” *Soc. Netw.*, vol. 1, no. 3, pp. 215-239, 1979. DOI: 10.1016/0378-8733(78)90021-7

[35] U. Brandes, D. Fleischer, “Centrality measures based on current flow,” in *Proc. Ann. Symp. Theor. Aspects Comput. Sci.*, 2005, pp. 533-544. DOI: 10.1007/978-3-540-31856-9_44

[36] K. Stephenson, M. Zelen, “Rethinking centrality: methods and examples,” *Soc. Netw.*, vol. 11, no. 1, pp. 1-37, 1989. DOI: 10.1016/0378-8733(89)90016-6

[37] M. E. J. Newman, “A measure of betweenness centrality based on random walks,” *Soc. Netw.*, vol. 27, no. 1, pp. 39-54, 2005. DOI: 10.1016/j.socnet.2004.11.009

[38] K.-I. Goh, B. Kahng, D. Kim, “Universal behaviour of load distribution in scale-free networks,” *Phys. Rev. Lett.*, vol. 87, no. 27, pp. 1-4, 2001. DOI: 10.1103/PhysRevLett.87.278701

[39] A.-M. Kermarrec, E. L. Merrer, B. Sericola, et al., “Second order centrality: distributed assessment of nodes criticity in complex networks,” *Comput. Commun.*, vol. 34, no. 5, pp. 619-628, 2011. DOI: 10.1016/j.comcom.2010.06.007

[40] P. Boldi, S. Vigna, “Axioms for centrality,” *Internet Math.*, vol. 10, no. 3-4, pp. 222-262, 2014. DOI: 10.1080/15427951.2013.865686

[41] A. Hagberg, D. Schult, P. Swart, “Exploring network structure, dynamics, and function using NetworkX,” in *Proc. 7th Python Sci. Conf.*, 2008, pp. 11-16.

[42] P. Zhang, J. Wang, X. Li, et al., “Clustering coefficient and community structure of bipartite networks,” *Phys. A: Stat. Mech. Appl.*, vol. 387, no. 27, pp. 6869-6875, 2008. DOI: 10.1016/j.physa.2008.09.006

[43] B. W. Kernighan, S. Lin, “An efficient heuristic procedure for partitioning graphs,” *Bell Syst. Tech. J.*, vol. 49, no. 2, pp. 291-307, 1970. DOI: 10.1002/j.1538-7305.1970.tb01770.x

[44] A.-H. Esfahanian, “Connectivity algorithms.” cse.msu.edu. http://www.cse.msu.edu/~cse835/Papers/Graph_connectivity_revised.pdf (accessed Apr. 6, 2023).

[45] A. N. Langville, C. D. Meyer, “A survey of eigenvector methods for web information retrieval,” *SIAM Rev.*, vol. 47, no. 1, pp. 135-161, 2005. DOI: 10.1137/S0036144503424786

[46] R. S. Burt, “Structural holes and good ideas,” *Am. J. Sociol*., vol. 110, no. 2, pp. 349-399, 2004. DOI: 10.1086/421787

[47] R. S. Burt, *Structural Holes: The Social Structure of Competition*. Cambridge: Harvard University Press, 1995.

[48] U. Brandes, *Network Analysis: Methodological Foundations*. New York: Springer, 2005. Accessed: Apr. 6, 2023. [Online]. Available: http://books.google.com/books?id=TTNhSm7HYrIC

[49] R. D. Luce, A. D. Perry, “A method of matrix analysis of group structure,” *Psychometrika*, vol. 14, no. 2, pp. 95–116, 1949. DOI:10.1007/BF02289146

[50] P. W. Holland, S. Leinhardt, “Transitivity in structural models of small groups,” *Comparative Group Studies*, vol. 2, no. 2, pp. 107–124, 1971. DOI: 10.1177/104649647100200201

[51] D. J. Watts, S. Strogatz, “Collective dynamics of ‘small-world’ networks,” *Nature*, vol. 393, no. 6684, pp. 440–442, 1998. DOI: 10.1038/30918

[52] A. Altmann, L. Toloşi, O. Sander, et al., “Permutation importance: a corrected feature importance measure,” *Bioinformatics*, vol. 26, no. 10, pp. 1340-1347, 2010. DOI: 10.1093/bioinformatics/btq134

[53] S. M. Lundberg, S. Lee, “A unified approach to interpretin model predictions,” in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, Dec. 2017, pp. 4768-4777. DOI: 10.48550/arXiv.1705.07874

[54] Y. Xu, D. Zhang, J. Y. Yang, “A feature extraction method for use with bimodal biometrics,” *Pattern Recognit.*, vol. 43, no. 3, pp. 1106-1115, 2010. DOI: 10.1016/j.patcog.2009.09.013

[55] T. Mikolov, K. Chen, G. Corrado, et al., “Efficient estimation of word representations in vector sapce,” arXiv preprint, 2013. DOI: 10.48550/arXiv.1301.3781

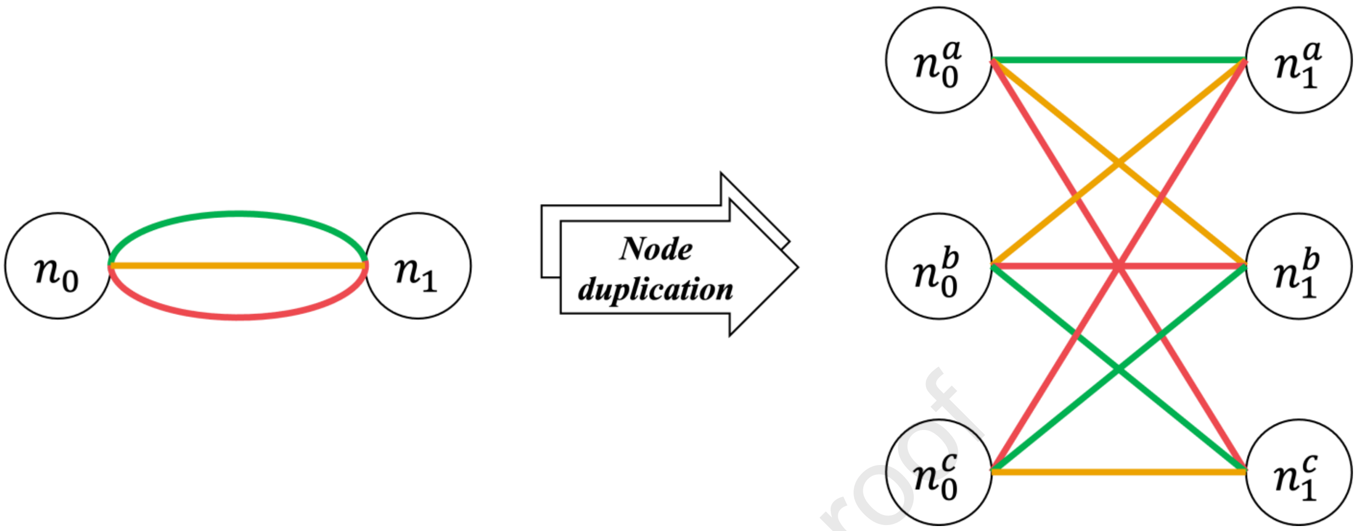
[56] M. Kubat, “Neural networks: a comprehensive foundation by Simon Haykin, Macmillan, 1994,” *Knowl. Eng. Rev.*, vol. 13, no. 4, pp. 409-412, 1995. DOI: 10.1017/s0269888998214044

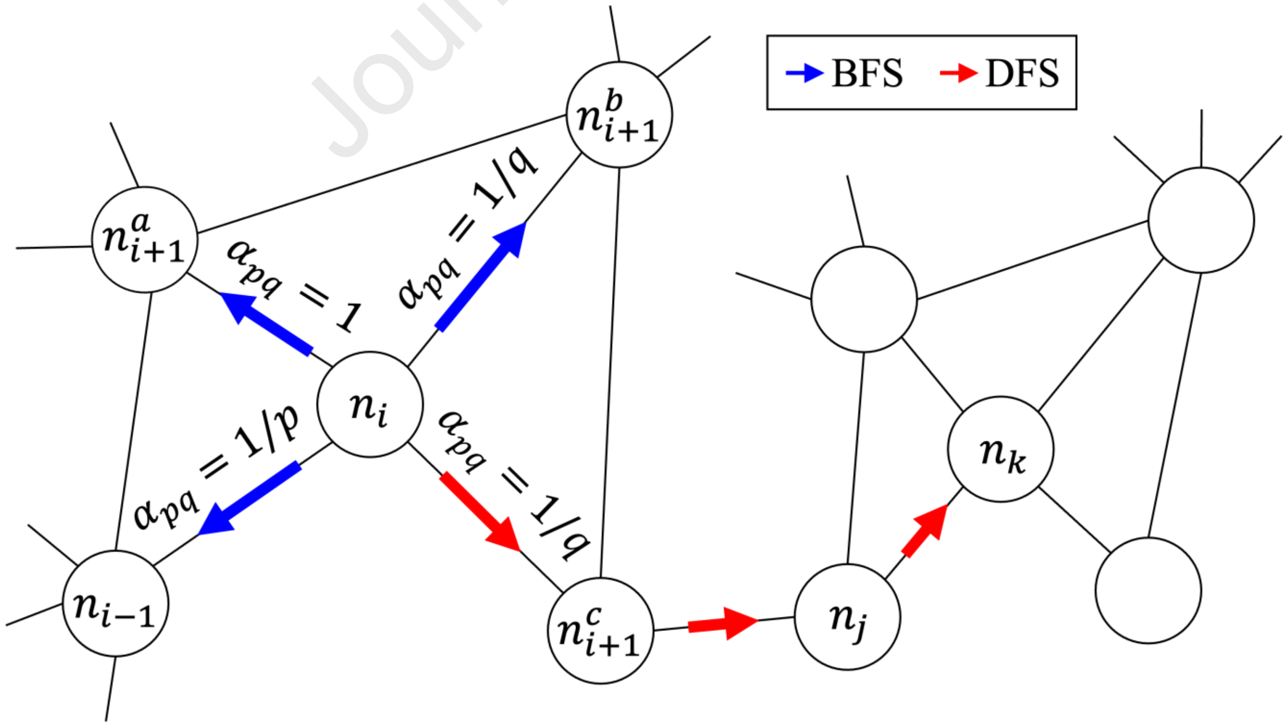
[57] V. Nair, G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” in *Proc. 27th Int. Conf. Mach. Learn. (ICML-10)*, 2010, pp. 807–814.

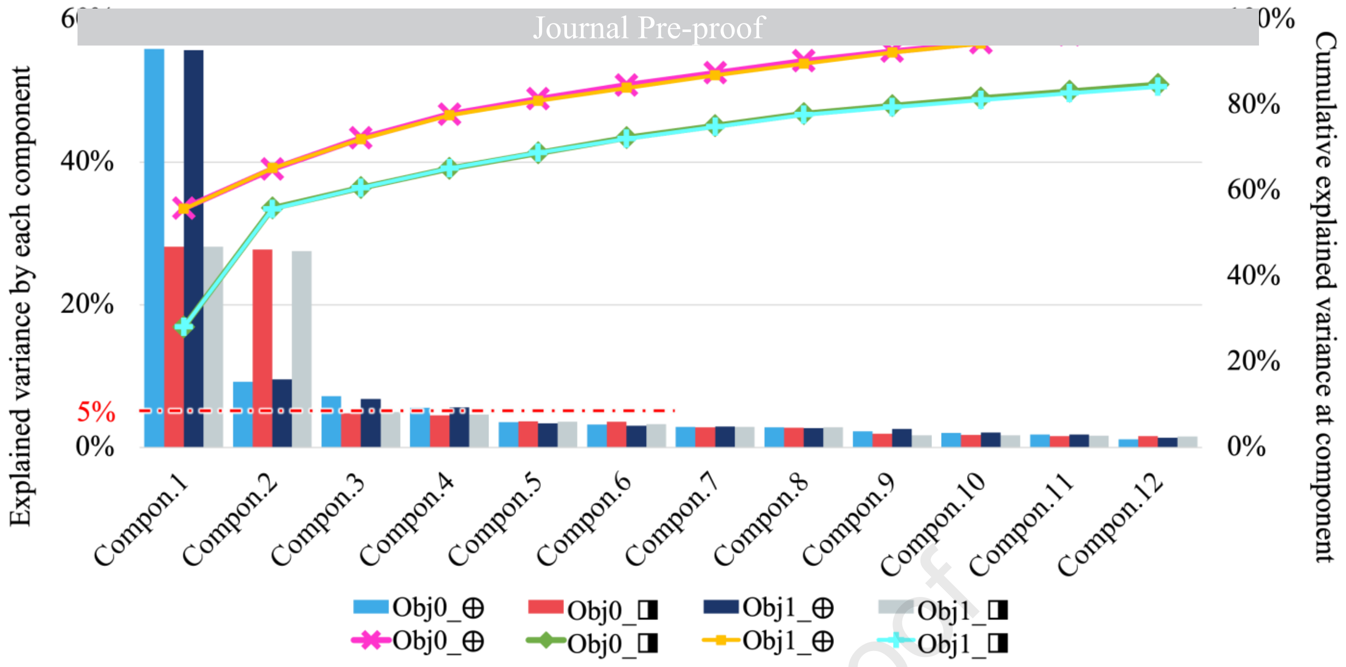
[58] X. Glorot, A. Bordes, Y. Bengio, “Deep sparse rectifier neural networks,” in *Proc. 14th Int. Conf. Artif. Intell. Stats.*, 2011, pp. 315–323. DOI: 10.1.1.208.6449

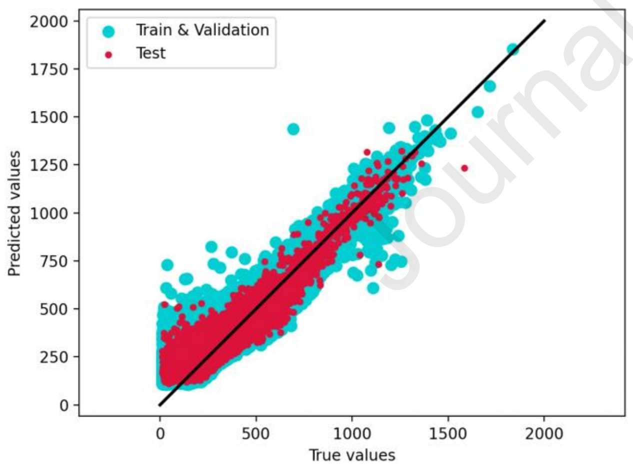
[59] D. P. Kingma, J. L. Ba, “Adam: a method for stochastic optimisation,” *CoRR* abs/1412.6980, 2014, n. pag. DOI: 10.48550/arXiv.1412.6980

[60] F. Pedregosa, G. Varoquaux, A. Gramfort, et al, “Scikit-learn: machine learning in Python,” arXiv preprint, 2012. DOI: 10.48550/arXiv.1201.0490

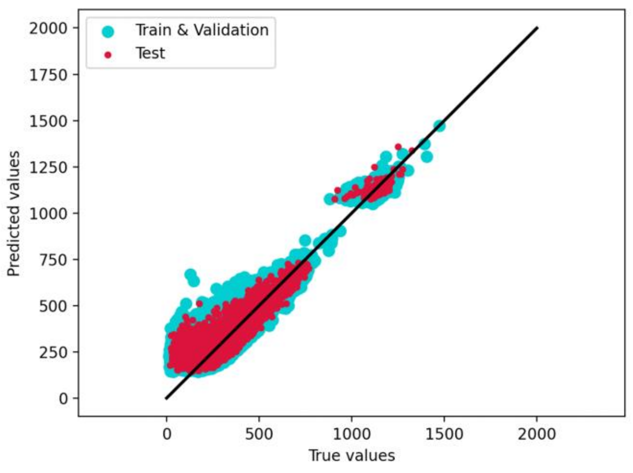

n_0
n_1
Node duplication
n_0^a
n_0^b
n_0^c
n_1^a
n_1^b
n_1^c


BFS
DFS
n_{i+1}^a
n_{i+1}^b
n_{i+1}^c
n_i
n_{i-1}
n_j
n_k
$\alpha_{pq} = 1$
$\alpha_{pq} = 1/q$
$\alpha_{pq} = 1/p$
$\alpha_{pq} = 1/q$

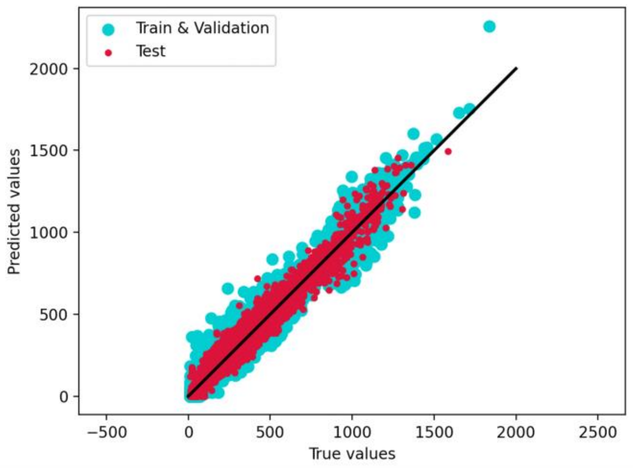




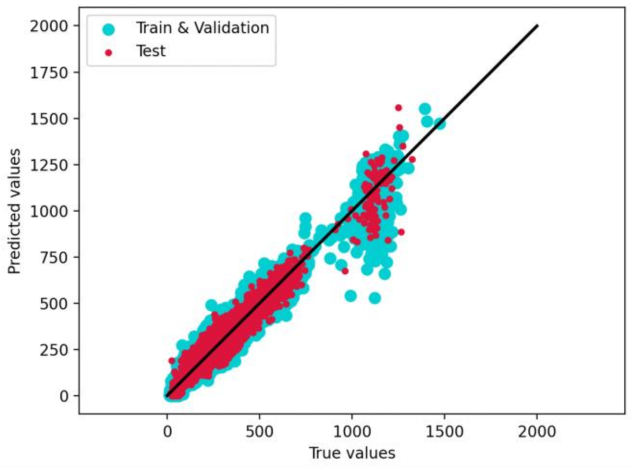
(a) Objective 0 results based on physical patterns



(b) Objective 1 results based on physical patterns



(c) Objective 0 results based on node embeddings



(d) Objective 1 results based on node embeddings

Highlights (for review)

- Node physical patterns are offered for multigraph shortest path cost prediction
- Important node physical patterns are identified
- Multigraph simplification methods are proposed for node embedding algorithm usability
- Trimming parallel edges is the most recommended method to simplify multigraphs
- Simplifying by duplicating multigraph nodes produces embeddings that predict the best

CONFLICT OF INTEREST

For Green Energy and Intelligent Transportation

Manuscript title:
Extracting Multi-objective Multigraph Features for the Shortest Path Cost Prediction: Statistics-based or Learning-based?

The authors whose names are listed immediately below certify that they have NO affiliations with or involvement in any organization or entity with any financial interest (such as honoraria; educational grants; participation in speakers' bureaus; membership, employment, consultancies, stock ownership, or other equity interest; and expert testimony or patent-licensing arrangements), or non-financial interest (such as personal or professional relationships, affiliations, knowledge or beliefs) in the subject matter or materials discussed in this manuscript.

Songwei Liu
Xinwei Wang
Michal Weiszer
Jun Chen

This statement is signed by all the authors to indicate agreement that the above information is true and correct:

Author's name (typed)	Author's signature	Date
Songwei Liu	Songwei Liu	25 Apr 2023
Xinwei Wang	Xinwei Wang	26 Apr 2023
Michal Weiszer	[illegible]	26 Apr 2023
Jun Chen	[illegible]	26 Apr 2023