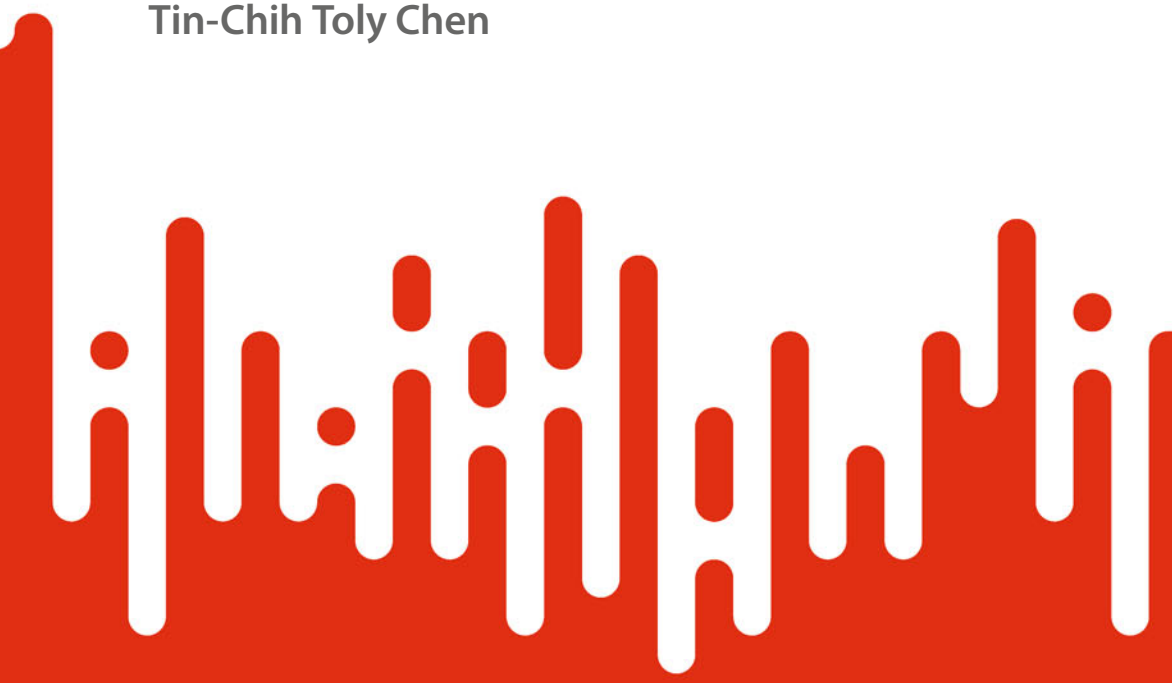
SpringerBriefs in Applied Sciences and Technology

Tin-Chih Toly Chen



Explainable Artificial Intelligence (XAI) in Manufacturing

Methodology, Tools, and Applications

Springer

SpringerBriefs in Applied Sciences and Technology

SpringerBriefs present concise summaries of cutting-edge research and practical applications across a wide spectrum of fields. Featuring compact volumes of 50 to 125 pages, the series covers a range of content from professional to academic.

Typical publications can be:

- A timely report of state-of-the art methods
- An introduction to or a manual for the application of mathematical or computer techniques
- A bridge between new research results, as published in journal articles
- A snapshot of a hot or emerging topic
- An in-depth case study
- A presentation of core concepts that students must understand in order to make independent contributions

SpringerBriefs are characterized by fast, global electronic dissemination, standard publishing contracts, standardized manuscript preparation and formatting guidelines, and expedited production schedules.

On the one hand, **SpringerBriefs in Applied Sciences and Technology** are devoted to the publication of fundamentals and applications within the different classical engineering disciplines as well as in interdisciplinary fields that recently emerged between these areas. On the other hand, as the boundary separating fundamental research and applied technology is more and more dissolving, this series is particularly open to trans-disciplinary topics between fundamental science and engineering.

Indexed by EI-Compendex, SCOPUS and Springerlink.

Tin-Chih Toly Chen

Explainable Artificial Intelligence (XAI) in Manufacturing

Methodology, Tools, and Applications

Springer

Tin-Chih Toly Chen
Department of Industrial Engineering
and Management
National Yang Ming Chiao Tung University
Hsinchu, Taiwan

ISSN 2191-530X ISSN 2191-5318 (electronic)
SpringerBriefs in Applied Sciences and Technology
ISBN 978-3-031-27960-7 ISBN 978-3-031-27961-4 (eBook)
https://doi.org/10.1007/978-3-031-27961-4

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Contents

Chapter 1
Explainable Artificial Intelligence (XAI) in Manufacturing

Abstract This chapter begins by defining explainable artificial intelligence (XAI). A procedure for implementing XAI was also established. Then, through literature analysis, the application of XAI in various fields such as medicine, service, education, finance, medical treatment, manufacturing, food, and military is compared. Some representative cases in these fields are also reported. Subsequently, several applications of XAI in the field of manufacturing are reviewed, including explaining the classification process and results of factory jobs, explaining artificial neural network (ANN)-based cycle time prediction methods, comparing the effect of alloy composition using Shapely additive explanation value (SHAP) analysis, etc.

Keywords Artificial intelligence · Explainable artificial intelligence · Artificial neural network · Shapely additive explanation value

1.1 Explainable Artificial Intelligence (XAI)

Artificial intelligence (AI) are technologies that enable computers to imitate human behavior [1]. The computing speed, storage capacity, reliability, and interconnectivity of computers combined with human reasoning patterns give AI the ability to solve complex and large-scale problems. **Explainable artificial intelligence (XAI)** is a new trend in AI [2]. At present, the development of XAI can be divided into the following two directions (see Fig. 1.1):

- XAI is to enhance the practicality of an existing AI technology by explaining its reasoning process and/or result [3]. For example, some studies have used heatmaps to show the parts of an image that a deep neural network (DNN) emphasizes when classifying or recognizing patterns [4, 5]. In addition, decision and regression rules have also been adopted to explain the reasoning mechanisms of DNNs [6, 7]. The concept of local interpretable model-agnostic explanation (LIME) is to explain the classification or regression result using a machine learning model by identifying critical features/predictors and fitting a simple and locally interpretable model [8]. In LIME, a synthetic dataset is generated, because raw data sometimes

T.-C.T. Chen, *Explainable Artificial Intelligence (XAI) in Manufacturing*,
SpringerBriefs in Applied Sciences and Technology,
https://doi.org/10.1007/978-3-031-27961-4_1

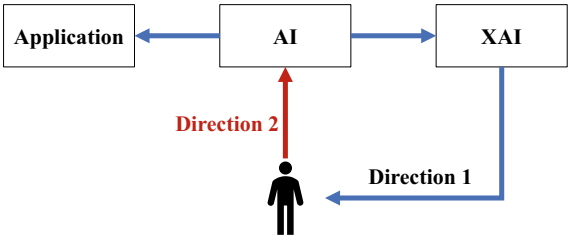


Fig. 1.1 Two development directions of XAI

contain exceptional cases and may not cover the entire sample space, which may complicate the explanation model. Furthermore, what kinds of AI technologies are explainable is inconclusive. In the view of McNamara [7], ensemble methods, random forecasts, decision trees, Bayesian networks, sparse linear models, and others are highly explainable, while artificial neural networks (ANNs), deep learning, type-*n* fuzzy logic [9, 10], support vector machines, and others are less explainable. Therefore, XAI techniques and tools developed following this direction are referred to as white boxes or the twins of AI applications that are often considered black boxes [11].

- XAI is to improve the effectiveness of existing AI technologies by incorporating easy-to-interpret visual features such as heatmaps [4, 5], decision (or regression) rules [12], decision trees [13, 14], scatter plots [15], etc., to help diagnose its reasoning mechanism. An example is attention-based DNNs that incorporates DNNs with heatmaps [16]. The linkage between the visual feature and the reasoning mechanism needs to be discovered.

1.2 Implementation Procedure of XAI

The implementation procedure of XAI is composed of several steps, as illustrated in Fig. 1.2. For an AI technology application, the reasoning process and/or result should be explainable and interpretable. In addition, the result needs to be validated. After these are done, the AI technology application can be said to be explainable, which contributes to its trustworthiness, fairness, accountability, and transparency [12, 17]. In addition, the explainability of AI technologies fosters improvement ideas based on which these AI technologies can be modified to enhance the effectiveness.

1.3 Basic Concepts in XAI

Some basic concepts in XAI are introduced as follows. A **local explanation** is to explain the rationale behind the result of a single example. A **global explanation** is to explain how the model arrives at its prediction/result and can be in the form of a visualization, a mathematical formula, or a diagram of the model. **Contrastive**

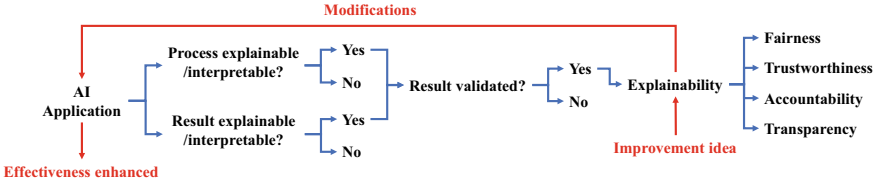


Fig. 1.2 Implementation procedure of XAI

explanations help us understand why a model generates a certain result/output for a given input and not another. Probably the most useful explanation technique is **what-if explanation**, which helps us understand the relationship between the model result and input features, similar to the concept of a parametric analysis. **Counterfactual explanations** tell us how changing assumptions about the inputs or parameters of a model can cause the model to arrive at a particular different result, similar to the concept of a sensitivity analysis. **Example-based explanations** are the simplest explanations in which the behavior of a model or underlying data distribution is simply explained by highlighting specific instances of the data.

A **model-agnostic XAI** method is that the computation required to explain an AI technology application is not based on the parameters of the AI technology application. **A post-hoc interpretability method** explains the result, while **a pre-hoc interpretability method** explains the process. Kamath and Liu [17] reviewed statistical and data analysis methods and tools suitable for the explanation of AI applications. In particular, they divided deep learning explanation methods into three categories: intrinsic methods, perturbation methods, and gradient/backpropagation methods.

Venugopal et al. [18] define the concept of **explainability failure** as a situation in which the output of an AI technology application is correct, but cannot be explained.

1.4 XAI Applications in Various Domains

Figure 1.3 provides statistics on the domains where XAI techniques and tools are most commonly applied. Most commonly applied domains include medicine, service, and education. In contrast, manufacturing is a domain where XAI is rarely applied. Therefore, the application of XAI techniques and tools is an urgent task and an opportunity for manufacturing.

Some XAI applications in various domains are reviewed as follows.

Lin and Chen [16] proposed a type-II fuzzy approach with XAI to overcome the difficulty for travelers to choose suitable nature-based leisure travel destinations during the coronavirus disease 2019 (COVID-19) pandemic. Several measures were taken by them to enhance the explainability of the selection process and result: color management, common expressions, annotated figures, traceable aggregation, and segmented distance diagrams. A segmented distance diagram was designed

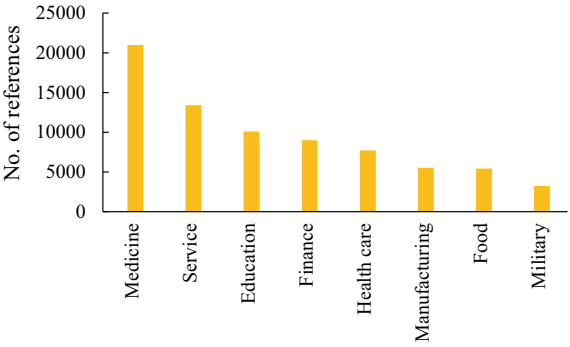


Fig. 1.3 Number of references about XAI applications in various domains from 2010 to 2022. *Data source* Google Scholar

for explaining the comparison result of alternatives using fuzzy Vise Kriterijumska Optimizacija I Kompromisno Resenje (VIKOR).

Panigutti et al. [13] constructed a decision tree to explain the classification result of patients according to their clinical histories.

Decision making seems to be the most pressing field where AI applications need to be properly explained, because in every decision-making task the stakeholders need to be explained somehow [19]. For a water quality conservation project, Dujmović and Allen III [19] evaluated and compared several alternatives using the logic scoring of preference (LSP) method. They identified three explanation issues: (1) explaining the composition and aggregation of LSP criteria, (2) explaining the evaluation result of each alternative, and (3) explaining the comparison and ranking results of alternatives.

Aghamohammadi et al. [20] constructed an adaptive neural fuzzy inference system (ANFIS) for heart attack prediction, in which three XAI tools and techniques were applied to explain the prediction mechanism and result: bar charts, line charts, and performance evaluation.

Chen and Chiu [21] proposed a hybridizing subjective and objective fuzzy group decision-making approach with XAI to evaluate the sustainability of smart technology applications in health care, in which three XAI techniques and tools were applied to enhance the understandability of the fuzzy group decision-making approach: color management technique, common expression technique, and traceable aggregation [15].

Liu and Liu used particle swarm optimization (PSO) to predict the permeability of tight sandstone reservoirs [22], in which eXtreme gradient boosting (XGBoost), also a decision tree method, was applied to explain the prediction process and result, so that the effect of each reservoir feature on the permeability could be accessed.

In sum, in these cases, XAI tools and techniques were applied for explaining the classification result, explaining the composition and aggregation of criteria, explaining the evaluation results of alternatives, explaining the comparison and ranking of alternatives, explaining the prediction mechanism and result, comparing the effects of inputs on the output, etc. (see Fig. 1.4).

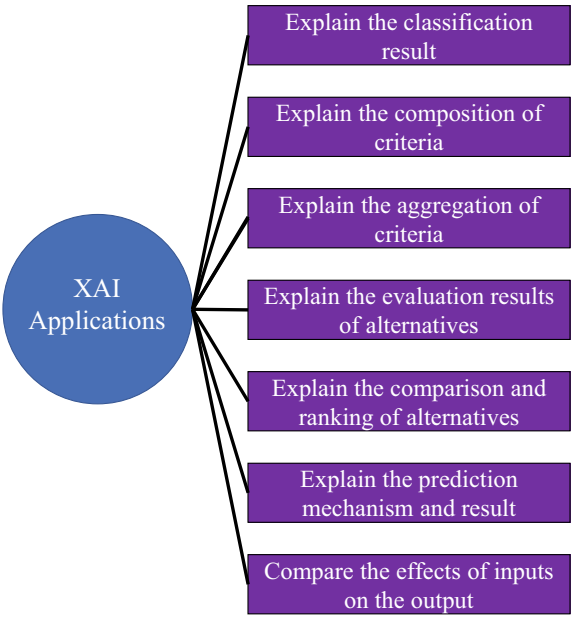


Fig. 1.4 Main purposes of XAI applications

1.5 XAI Applications in Manufacturing

Some XAI applications in manufacturing are reviewed as follows.

Chen and Wang [23] proposed a two-stage XAI approach to explain a classification-based cycle time prediction method. In their methodology, first, jobs are divided into several clusters. A scatter radar diagram is then designed to illustrate the classification result. Compared with existing XAI techniques, the scatter radar diagram meets more requirements for better interpretation. Subsequently, an ANN is constructed for each cluster to predict the cycle times of jobs in the cluster. A random forest (RF) is then constructed to approximate the prediction mechanism of the ANN. In existing practice, a RF generates many decision rules to predict the cycle time of a job, which may cause confusion for the user. To solve this problem, they established a systematic procedure to re-organize these decision rules. In this way, the first few decision rules can provide most of the information, and the user does not have to read all the rules.

Kong et al. [24] constructed an ANN to predict the properties of an alloy based on its composition, for which Shapely additive explanation value (SHAP) analysis was conducted to compare the effects of components.

A similar treatment was taken in Akhlaghi et al. [25], in which a DNN was constructed to predict the performance of a dew point cooler according to its features. SHAP was applied to compare the effects of these features.

1.6 Difficulties in Applying XAI in Manufacturing

So far, AI technologies have been widely applied in manufacturing [26, 27]. The result of an AI technology application in manufacturing may be directly put into practice without human intervention, which eliminates the need to explain it to the related people. In contrast, the result of an AI technology application in health care often requires human approval or decision making. As a consequence, analysts tend to apply AI technologies that are more complex but potentially more effective. This phenomenon may slow down the development of XAI technologies in the manufacturing domain.

In addition, AI applications in manufacturing typically pursue the optimization of the average performance. As a result, the failure in one case can be compensated for by the success in another. In contrast, AI applications in other fields may not, as they strive for acceptable performance in every case. Therefore, in manufacturing, the need to explain the overall (or global) performance will be much stronger than that to explain an individual (or local) case, which is different from the application of XAI in other fields (e.g., customer service) [13].

Further, a large part of AI technology applications in manufacturing is optimization [28, 29], while AI applications in other fields are mostly related to pattern recognition and classification [30]. For example, DNNs are often used to identify defect patterns. However, the inference process and results of such DNNs are difficult to understand. To interpret defect pattern recognition results, Chen et al. [30] represented a portion of an image with prototypes that were projected onto a representation of the training data for visualization and interpretation.

As stated above, statistical and data analysis methods and tools are widely used to explain AI applications. However, people in the manufacturing field may not be familiar with or interested in these statistical or data analysis methods and tools. In addition, concepts and tools familiar to those in the manufacturing field should be adopted to develop XAI methods in this domain. Furthermore, while a number of approaches have been developed for post-hoc explainability, only a few focus on how to use domain knowledge and how it influences the understandability of global explanations from the users' perspective, which is especially critical for manufacturing system.

1.7 Organization of This Book

This book is intended to provide technical details on the development and application of XAI to manufacturing, including methodologies, tools, system architectures, software and hardware, examples, and applications. XAI is currently the hottest topic, since the acceptability of overly complex AI techniques is usually questioned. Simple XAI techniques and tools for explaining these AI applications are pursued.

However, simple XAI techniques and tools often struggle to adequately explain an AI application. In other words, applying simple XAI techniques and tools to explain a complicated AI application is a challenge.

In manufacturing, employees often lack AI knowledge, so they especially look forward to understanding AI technologies. However, as long as an AI technology can effectively solve problems, employees are often willing to accept no matter how complicated the AI technology is [31]. Because of this, many studies or reports inevitably exaggerate the benefits of AI applications to manufacturing, while ignoring their understandability and acceptability.

In addition to introducing XAI tools and techniques applied in manufacturing, this book also discusses the management implications of the related applications. After all, innovative information, computer and computing technologies are constantly being introduced, not just for a single functionality, while how to improve the efficiency and even the competitiveness of a factory is still the most fundamental issue.

In specific, the outline of the present book is structured as follows.

In the current chapter, XAI is first defined. A procedure for implementing XAI is also established. Then, through a literature analysis, applications of XAI in various domains, such as medicine, service, education, finance, health care, manufacturing, food, and military, are compared. Some representative cases in these domains are also reported. Subsequently, several applications of XAI in the manufacturing domain are reviewed, including explaining the classification process and result of jobs in a factory, explaining an ANN-based cycle time prediction method, comparing the effects of the components of an alloy using SHAP analysis, etc.

Chapter 2, Applications of XAI for Forecasting in the Manufacturing Domain, focuses on forecasting, an important function of manufacturing systems. Many operation and production activities, such as cycle time forecasting [32], sales forecasting [33], unit cost reduction [34], predictive maintenance [35], yield learning [36], etc., are based on forecasting. This chapter takes job cycle time forecasting as an example. There are several applications of AI techniques for job cycle time prediction. Among these, ANN (or DNN) applications are the most effective, but very difficult for factory workers to understand or communicate. To address this issue, existing XAI techniques and tools for explaining the reasoning process and result of ANNs (or DNNs) are introduced. We first introduce XAI tools for visualizing operations in ANNs (or DNNs), such as ConvNetJS [37], TensorFlow [38], Seq2Seq [39], and MATLAB [40], and then mention XAI techniques for evaluating the effect, contribution, or importance of each input on the output, including partial derivation, odd ratio [41], out-of-bag (OOB) predictor importance [42], recursive feature elimination (RFE) [43], and SHAP [24]. Subsequently, XAI techniques for approximating the relationship between the inputs and output of an ANN (or DNN) [44, 45], especially simpler machine learning techniques like case-based reasoning (CBR) [11], classification and regression tree (CART) [46], RF [22], gradient boosted decision tree [47], eXtreme gradient boosting (XGBoost) [48], and RF-based incremental interpretation [22], are introduced. The application of each XAI technique is supplemented by simple examples and corresponding MATLAB codes, allowing readers to learn quickly.

Chapter 3, Applications of XAI for Decision Making in the Manufacturing Domain, deals with an important topic in factory management, namely improving the understandability of AI applications for group multi-criteria decision making in manufacturing systems. Decision making may be more critical to the competitiveness and sustainability of a manufacturing system than production planning and control because of its long-term and cross-functional impact [49–52]. This chapter uses the example of choosing suitable smart and automation technologies for factories during the COVID-19 pandemic. This topic is of particular importance as many factories are forced to close or operate on a smaller scale (using a smaller workforce), thus pursuing further automation. AI and Industry 4.0 technologies have many applications in this field, most of which can also be applied for other decision-making purposes in manufacturing systems. In the beginning, a systematic procedure is established for guiding the group multi-criteria decision-making process. Applications of AI and XAI in identifying targets are first reviewed. Subsequently, the applications of AI and XAI in selecting factors and developing criteria are presented. AI technologies are widely used to derive the priorities of criteria. Therefore, XAI techniques and tools for explaining such AI applications are particularly important. Aggregating the judgments of multiple decision makers is the next focus, followed by the introduction of AI and XAI applications to evaluate the overall performance of each alternative. Taking the fuzzy technique for order preference by similarity to the ideal solution (FTOPSIS) [11] as an example, the applications of XAI techniques and tools for explaining the comparison result using FTOPSIS are illustrated. Another AI technology for the same purpose is fuzzy VIKOR. XAI techniques and tools for explaining fuzzy VIKOR are also introduced. Finally, several metrics are proposed to evaluate the effectiveness of XAI techniques or tools for decision making in the manufacturing domain [53].

Chapter 4, Applications of XAI to Job Sequencing and Scheduling in Manufacturing, discusses a new field of applications of XAI in manufacturing—job sequencing and scheduling. It first breaks down job sequencing and scheduling into several steps and then mentions AI technologies applicable to some of these steps. It is worth noting that many AI applications are directed at the preparation of inputs required for scheduling tasks, rather than the processes of scheduling tasks, which is a distinctive feature of this field. Nevertheless, many AI technologies have been explained in other domains or fields. These explanations can provide a reference for explaining AI applications in job sequencing and scheduling. Therefore, some generic XAI techniques and tools for job sequencing and scheduling are reviewed, including

- Referring to the taxonomy of job scheduling problems;
- Tailoring dispatching rule;
- Textual description, pseudocode;
- Decision tree, flowchart.

In addition, job sequencing and scheduling problems are often formulated as mathematical programming (optimization) models to be optimized. AI technologies can be applied to find the optimal solutions to the models. Applications of GA are

of particular interest because such applications are most common in job scheduling. Moreover, XAI techniques and tools for explaining GA can be easily extended to account for other evolutionary AI applications such as artificial bee colony (ABC), ant colony optimization (ACO), and PSO in job scheduling. Applicable XAI techniques and tools include

- Flowchart, textual description;
- Chromosomal diagram;
- Dynamic line chart, bar chart with baseline.

Some novel XAI techniques and tools for explaining GA are also introduced:

- Decision tree-based interpretation;
- Dynamic transition and contribution diagram.

References

1. T.C.T. Chen, Y.C. Wang, AI applications to shop floor management in lean manufacturing, in *Artificial Intelligence and Lean Manufacturing* (2022), pp. 75–90
2. D. Gunning, M. Stefik, J. Choi, T. Miller, S. Stumpf, G.Z. Yang, XAI—Explainable artificial intelligence. Sci. Robot. **4**(37), eaay7120 (2019)
3. D. Gunning, D. Aha, DARPA's explainable artificial intelligence (XAI) program. AI Mag. **40**(2), 44–58 (2019)
4. D. Kumar, A. Wong, G.W. Taylor, Explaining the unexplained: a class-enhanced attentive response (clear) approach to understanding deep neural networks, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (2017), pp. 36–44
5. E. Tjoa, H.J. Khok, T. Chouhan, G. Cuntai, Improving deep neural network classification confidence using heatmap-based eXplainable AI (2022). https://doi.org/10.48550/arXiv.2201.000092022
6. A. Binder, G. Montavon, S. Lapuschkin, K.R. Müller, W. Samek, Layer-wise relevance propagation for neural networks with local renormalization layers, in *International Conference on Artificial Neural Networks* (2016), pp. 63–71
7. M. McNamara, Explainable AI: What is it? How does it work? And what role does data play? (2022). https://www.netapp.com/blog/explainable-ai/
8. M.T. Ribeiro, S. Singh, C. Guestrin, "Why should I trust you?" Explaining the predictions of any classifier, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016), pp. 1135–1144
9. T.-C.T. Chen, Type-II fuzzy collaborative intelligence for assessing cloud manufacturing technology applications. Robot. Comput. Integr. Manuf. **78**, 102399 (2022)
10. M.-C. Chiu, T. Chen, A ubiquitous healthcare system of 3D printing facilities for making dentures: application of type-II fuzzy logic. Digital Health **8**, 20552076221092540 (2022)
11. T. Chen, Y.-C. Wang, M.-C. Chiu, A type-II fuzzy collaborative forecasting approach for productivity forecasting under an uncertainty environment. J. Ambient. Intell. Humaniz. Comput. **12**, 2751–2763 (2021)
12. E.M. Kenny, M.T. Keane, Twin-systems to explain artificial neural networks using case-based reasoning: comparative tests of feature-weighting methods in ANN-CBR twins for XAI, in *Twenty-Eighth International Joint Conferences on Artificial Intelligence* (2019), pp. 2708–2715
13. C. Panigutti, A. Perotti, D. Pedreschi, Doctor XAI: an ontology-based approach to black-box sequential data classification explanations, in *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency* (2020), pp. 629–639

14. J. Souza, C.K. Leung, Explainable artificial intelligence for predictive analytics on customer turnover: a user-friendly interface for non-expert users, in *Explainable AI Within the Digital Transformation and Cyber Physical Systems* (2021), pp. 47–67
15. E. Choi, M.T. Bahadori, L. Song, W.F. Stewart, J. Sun, GRAM: graph-based attention model for healthcare representation learning, in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2017), pp. 787–795
16. Y.-C. Lin, T. Chen, Type-II fuzzy approach with explainable artificial intelligence for nature-based leisure travel destination selection amid the COVID-19 pandemic. Digital Health **8**, 20552076221106320 (2022)
17. U. Kamath, J. Liu, *Explainable Artificial Intelligence: an Introduction to Interpretable Machine Learning* (Springer, 2021)
18. V.K. Venugopal, R. Takhar, S. Gupta, V. Mahajan, Clinical explainability failure (CEF) & explainability failure ratio (EFR)–Changing the way we validate classification algorithms. J. Med. Syst. **46**(4), 1–5 (2022)
19. J. Dujmović, W.L. Allen III., Explainable decision-making for water quality protection. Information **13**(12), 551 (2022)
20. M. Aghamohammadi, M. Madan, J.K. Hong, I. Watson, Predicting heart attack through explainable artificial intelligence, in *International Conference on Computational Science* (2019), pp. 633–645.
21. T. Chen, M.-C. Chiu, Evaluating the sustainability of a smart technology application in healthcare after the COVID-19 pandemic: a hybridizing subjective and objective fuzzy group decision-making approach with XAI. Digital Health **8**, 20552076221136380 (2022)
22. J.J. Liu, J.C. Liu, Permeability predictions for tight sandstone reservoir using explainable machine learning and particle swarm optimization. Geofluids **2022**, 2263329 (2022)
23. T. Chen, Y.C. Wang, A two-stage explainable artificial intelligence approach for classification-based job cycle time prediction. Int. J. Adv. Manuf. Technol. **123**(5), 2031–2042 (2022)
24. B.O. Kong, M.S. Kim, B.H. Kim, J.H. Lee, Prediction of creep life using an explainable artificial intelligence technique and alloy design based on the genetic algorithm in creep-strength-enhanced ferritic 9% Cr steel. Metals Mater. Int. 1–12 (2022)
25. Y.G. Akhlaghi, K. Aslansefat, X. Zhao, S. Sadati, A. Badiei, X. Xiao, S. Shittu, Y. Fan, X. Ma, Hourly performance forecast of a dew point cooler using explainable artificial intelligence and evolutionary optimisations by 2050. Appl. Energy **281**, 116062 (2021)
26. B.H. Li, B.C. Hou, W.T. Yu, X.B. Lu, C.W. Yang, Applications of artificial intelligence in intelligent manufacturing: a review. Front. Inf. Technol. Electron. Eng. **18**(1), 86–96 (2017)
27. J.F. Arinez, Q. Chang, R.X. Gao, C. Xu, J. Zhang, Artificial intelligence in advanced manufacturing: current status and future outlook. J. Manuf. Sci. Eng. **142**(11), 110804 (2020)
28. A. Azizi, Hybrid artificial intelligence optimization technique. Appl. Artif. Intell. Tech. Ind. **4**, 27–47 (2019)
29. J.R. Rehse, N. Mehdiyev, P. Fettke, Towards explainable process predictions for industry 4.0 in the dfki-smart-lego-factory. KI-Künstliche Intelligenz **33**(2), 181–187 (2019)
30. C. Chen, O. Li, A. Barnett, J. Su, C. Rudin. This looks like that: deep learning for interpretable image recognition. arXiv preprint arXiv:1806.10574 (2018)
31. E. Daglarli, Explainable artificial intelligence (xAI) approaches and deep meta-learning models for cyber-physical systems, in *Artificial Intelligence Paradigms for Smart Cyber-Physical Systems* (2021), pp. 42–67
32. Y.-C. Wang, H.-R. Tsai, T. Chen, A selectively fuzzified back propagation network approach for precisely estimating the cycle time range in wafer fabrication. Mathematics **9**, 1430 (2021)
33. Q. Xu, V. Sharma, Ensemble sales forecasting study in semiconductor industry, in *Industrial Conference on Data Mining* (2017), pp. 31–44
34. T. Chen, H.-C. Wu, Forecasting the unit cost of a DRAM product using a layered partial-consensus fuzzy collaborative forecasting approach. Complex Intell. Syst. **6**, 497–492 (2020)
35. T.-C. T. Chen, Y.-C. Wang, AI applications to kaizen management, in *Artificial Intelligence and Lean Manufacturing*, pp. 37–52

36. T. Chen, Y.-C. Wang, Interval fuzzy number-based approach for modelling an uncertain fuzzy yield learning process. J. Ambient Intell. Humaniz. Comput. **11**, 1213–1223 (2020)
37. ConvNetJS, ConvnetJS demo: Toy 2d classification with 2-layer neural network (2022). https://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html
38. GitHub, Tensorflow (2022). https://github.com/tensorflow
39. Z. Li, J. Cai, S. He, H. Zhao, Seq2seq dependency parsing, in *Proceedings of the 27th International Conference on Computational Linguistics* (2018), pp. 3203–3214
40. S. Mantri, K. Bapat, Neural network based face recognition using MATLAB. Int. J. Comput. Sci. Eng. Technol. **1**(1), 6–9 (2011)
41. M. Green, U. Ekelund, L. Edenbrandt, J. Björk, J.L. Forberg, M. Ohlsson, Exploring new possibilities for case-based explanation of artificial neural network ensembles. Neural Netw. **22**(1), 75–81 (2009)
42. MathWorks, oobPermutedPredictorImportance (2022). https://www.mathworks.com/help/stats/classificationbaggedensemble.oobpermutedpredictorimportance.html?searchHighlight=oobPermutedPredictorImportance&s_tid=srchtitle_oobPermutedPredictorImportance_1
43. K. Yan, D. Zhang, Feature selection and analysis on correlated gas sensor data with recursive feature elimination. Sens. Actuators, B Chem. **212**, 353–363 (2015)
44. Y.-C. Lin, T. Chen, An intelligent system for assisting personalized COVID-19 vaccination location selection: Taiwan as an example. Digital Health **8**, 20552076221109064 (2022)
45. Y.-C. Wang, T. Chen, T.C. Hsu, A fuzzy deep neural network and simulation approach for enhancing cycle time range estimation precision in wafer fabrication. Decis. Anal. **1**, 100010 (2021)
46. H.-C. Wu, T. Chen, CART–BPN approach for estimating cycle time in wafer fabrication. J. Ambient. Intell. Humaniz. Comput. **6**, 57–67 (2015)
47. GoogleDevelopers, Gradient boosted decision trees|Machine learning (2022). https://developers.google.com/machine-learning/decision-forests/intro-to-gbdt
48. T. Chen, T. He, M. Benesty, V. Khotilovich, Y. Tang, H. Cho, K. Chen, Xgboost: extreme gradient boosting. R Package Version 0.4-2, 1(4), 1–4 (2015)
49. T. Chen, A FNP approach for evaluating and enhancing the long-term competitiveness of a semiconductor fabrication factory through yield learning modeling. Int. J. Adv. Manuf. Technol. **40**, 993–1003 (2009)
50. T. Chen, Establishing the optimal and efficient capacity re-allocation plans for enhancing the long-term competitiveness of a semiconductor product—a long-term trend viewpoint. Proc. Inst. Mech. Eng. Part B J. Eng. Manuf. **224**, 1295–1303 (2010)
51. T. Chen, A flexible way of modelling the long-term cost competitiveness of a semiconductor product. Robot. Comput. Integr. Manuf. **29**(3), 31–40 (2013)
52. T. Chen, Z. Mikoláš, Y.-C. Wang, Competitiveness assessment and enhancement for virtual organisations. Int. J. Technol. Manage. **70**(1), 1–3 (2016)
53. Y.-C. Wang, T.-C.T. Chen, M.-C. Chiu, An explainable deep-learning approach for job cycle time prediction. Decis. Anal. **6**, 100153 (2023)

Chapter 2
Applications of XAI for Forecasting in the Manufacturing Domain

Abstract This chapter focuses on forecasting, which is an important function of manufacturing systems. Many operations and production activities such as cycle time forecasting, sales forecasting, unit cost reduction, predictive maintenance, yield learning, etc. are based on forecasting. This chapter takes job cycle time forecasting as an example. Artificial intelligence (AI) techniques have many applications in job cycle time prediction. Of these, artificial neural network (ANN) (or deep neural network, DNN) applications are most effective, but are difficult for factory workers to understand or communicate. To address this issue, existing explainable AI (XAI) techniques and tools for explaining the inference process and results of ANNs (or DNNs) are introduced. We first introduce XAI tools for visualizing operations in ANNs (or DNNs), such as ConvNetJS, TensorFlow, Seq2Seq, and MATLAB, and then mention XAI techniques for evaluating the impact, contribution, or importance of each input on the output, including partial derivatives, odd ratio, out-of-bag (OOB) predictor importance, recursive feature elimination (RFE), Shapely additive explanation value (SHAP). Subsequently, XAI techniques for approximating the relationship between the inputs and output of an ANN (or DNN), especially simpler machine learning techniques such as case-based reasoning (CBR), classification and regression trees (CART), random forest (RF), gradient boosting decision trees, eXtreme gradient boosting (XGBoost), and RF-based incremental interpretation are introduced. The application of each XAI technique is supplemented with simple examples and corresponding MATLAB codes, allowing readers to get started quickly.

Keywords XAI · Forecasting · Job cycle time forecasting · Artificial neural network · Deep neural network · Partial derivatives · Odd ratio · OOB predictor importance · RFE · SHAP · Machine learning · CBR · CART · RF · Gradient boosting decision trees · XGBoost · RF-based incremental interpretation

T.-C.T. Chen, *Explainable Artificial Intelligence (XAI) in Manufacturing*,
SpringerBriefs in Applied Sciences and Technology,
https://doi.org/10.1007/978-3-031-27961-4_2

2.1 Applications of AI for Forecasting in the Manufacturing Domain

Artificial intelligence (AI) technologies have been widely used to support forecasting purposes in manufacturing systems, such as job cycle time forecasting [1], sales forecasting [2], unit cost reduction [3], predictive maintenance [4], yield learning [5], etc. AI technologies applied for different purposes may vary greatly. Therefore, in the following, job cycle time forecasting is taken as an example, because there are many AI technology applications in this field, and most of these AI technologies can also be applied to fulfill other forecasting purposes in a manufacturing system.

2.1.1 Applications of AI for Job Cycle Time Forecasting

The cycle time (or manufacturing lead time) of a job is the time it takes for the job to pass through a manufacturing system and can be calculated by subtracting the release time from the output/completion time [6]. The cycle time is also equal to the total processing time plus the waiting time. Predicting the cycle time of each job is an important task for manufacturing systems [7, 8]. After this task is fulfilled, the forecasted cycle times of jobs can be shortened by eliminating unnecessary waits and improving the responsiveness, which enhances the competitiveness of the manufacturing system [9]. However, accurately predicting the cycle time of a job is a difficult task if a manufacturing system has complex job routings, inconsistent human intervention, unreliable equipment, or unstable product quality [10].

So far, a variety of job cycle time prediction methods have been proposed, such as production simulation or digital twins [11, 12], case-based reasoning (CBR) [13], regression [14], artificial neural networks (ANNs) or deep neural networks (DNNs) [6, 10, 15–17], fuzzy inference systems (FISs) [10, 15, 16], hybrid methods [15, 16, 18], agent-based systems [19, 20], etc. Most of these methods are based on the application of AI technologies. However, advanced AI-based methods, such as ANNs, DNNs, FISs, hybrid methods, and agent systems, are not always easy to understand or communicate, especially for factory workers without sufficient background knowledge of AI, which limits the acceptability (or practicability) of these methods. The concept of explainable AI (XAI) is to cope with this problem [21], which aims to improve the interpretability or even effectiveness of an AI application by better explaining its reasoning mechanism and/or result [22].

Existing XAI techniques and tools for explaining ANN applications in job cycle time prediction are introduced as follows.

2.2 XAI Techniques and Tools for Explaining an ANN (DNN)

Existing XAI techniques for explaining ANNs are usually machine learning or AI methods that are simpler (i.e., easier to understand and communicate) than ANNs. Furthermore, these XAI techniques are designed to minimize the deviation between the approximate output and the network output, not the deviation between the network output and actual value. Furthermore, the approximated ANN is used for prediction, not for classification or decision making.

It is common to explain the operations in an ANN (or DNN) using textual descriptions and network configuration (or system architecture) diagrams [23, 24], as shown in Fig. 2.1. In addition, animation-based techniques, such as ConvNetJS for convolutional neural networks (CNNs) [25], TensorFlow (for ANNs with multiple hidden layers, DNNs) [26], Seq2Seq (for recurrent neural networks) [27], and MATLAB [28], have been applied to illustrate/animate the training process of an ANN or DNN (see Fig. 2.2). The comparison of these animation-based techniques refers to Table 2.1. Logarithmic sigmoid (log-sigmoid) function is a commonly used transformation/activation function in ANNs. Sudheer and Jain [29] developed a river model, which is actually a scatterplot with smooth lines by varying the parametric values of the log-sigmoid function to explain its behavior.

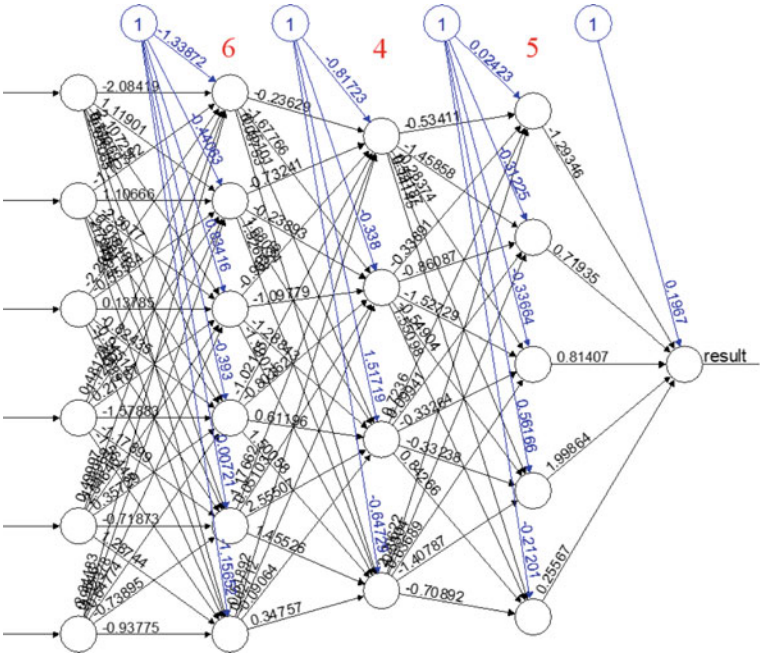


Fig. 2.1 Network architecture diagram of a DNN using R language

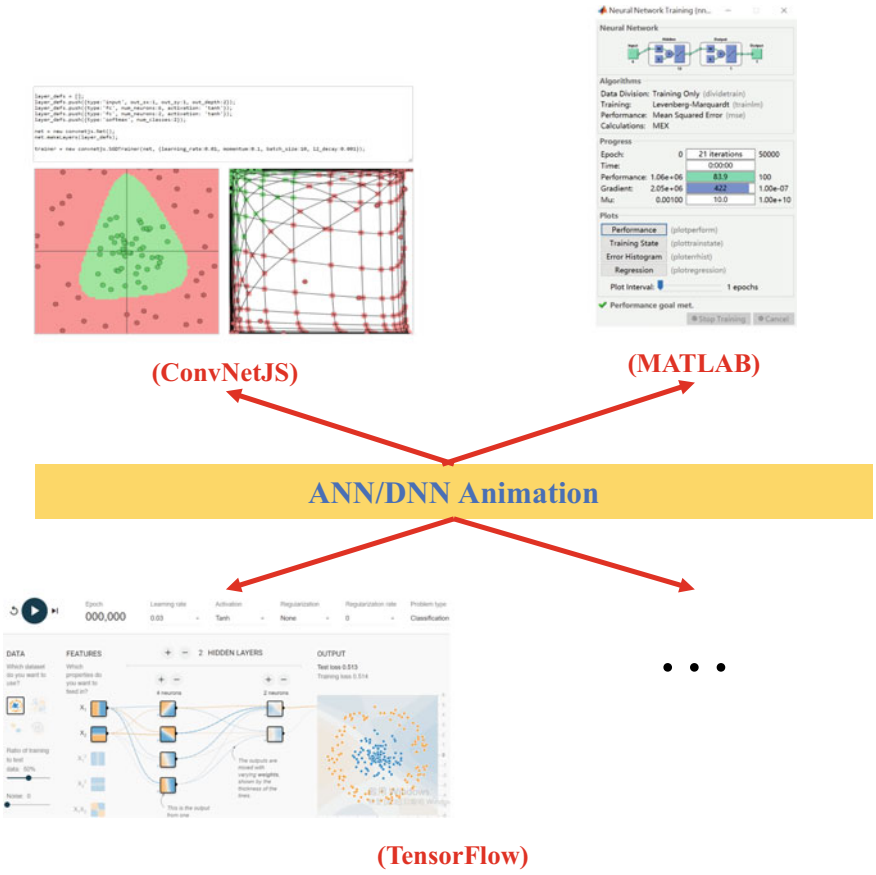


Fig. 2.2 Some existing animation-based tools for explaining an ANN/DNN

Table 2.1 Comparison of animation-based techniques

	R	ConvNetJS	MATLAB	TensorFlow	Seq2Seq
Showing the network architecture	Yes	Yes	Yes	Yes	Yes
Showing the values of network parameters	Yes	No	No	Yes	No
Showing the convergence process of network parameters	No	Yes	No	Yes	No
Showing the convergence process of results	No	Yes	No	Yes	No

For an XAI technique to be effective in approximating an ANN, the following conditions need to be satisfied:

- It is easy to understand.
- It is easy to communicate.
- It can illustrate the training process.
- No background knowledge is required.
- A direct relationship (a rule or a set of rules) between the inputs and output of the ANN is provided.
- The number of rules for explaining the ANN is not many.
- The contents of these rules are simple, such as including logic, constant, linear, and polynomial functions rather than inverse, exponential, logarithmic, trigonometric, and other advanced functions.
- The approximation deviation, measured in terms of mean absolute error (MAE), mean absolute percentage error (MAPE), or root mean squared error (RMSE), is small:

$$\text{MAE} = \frac{\sum_{j=1}^{n} \left|\hat{o}_j - o_j\right|}{n}; \quad k = 1{-}K \tag{2.1}$$

$$\text{MAPE} = \frac{\sum_{j=1}^{n} \left|\hat{o}_j - o_j\right|}{n}; \quad k = 1{-}K \tag{2.2}$$

$$\text{RMSE} = \sqrt{\frac{\sum_{j=1}^{n} \left(\hat{o}_j - o_j\right)^2}{n}}; \quad k = 1{-}K \tag{2.3}$$

where o_j is ANN output (i.e., the cycle time forecast of job j); $\hat{o}_j$ is the approximate value.

2.3 XAI Techniques and Tools for Explaining ANN Applications in Job Cycle Time Prediction

At first, the procedure of forecasting the cycle time of a job using an ANN is introduced.

2.3.1 Forecasting the Cycle Time of a Job Using an ANN

Taking the simplest feedforward neural network with three layers, the input layer, a hidden layer, and the output layer, as an example, is illustrated in Fig. 2.3.

Inputs to the ANN (indicated by x_{jp}; $p = 1{-}P$) are the attributes of example j. From the input layer to the hidden layer, the following operations are performed:

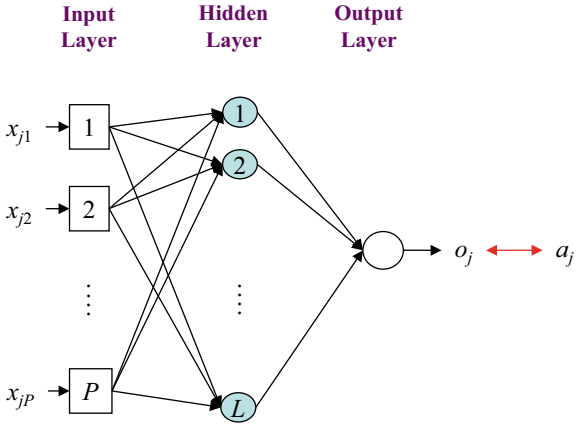


Fig. 2.3 Architecture of the ANN

$$I_{jl}^{h} = \sum_{p=1}^{P} (w_{pl}^{h} x_{jp}) \tag{2.4}$$

$$n_{jl}^{h} = I_{jl}^{h} - \theta_{l}^{h} \tag{2.5}$$

$$h_{jl} = f(n_{jl}^{h}) \tag{2.6}$$

where w_{pl}^{h} indicates the connection weight between the p-th input and the l-th node of the hidden layer. θ_{l}^{h} is the threshold on the hidden-layer node, and h_{jl} is the output from the node. $f()$ is the activation/transformation function. Outputs from the hidden layer are aggregated at the last layer as

$$I_{j}^{o} = \sum_{l=1}^{L} (w_{l}^{o} h_{jl}) \tag{2.7}$$

and then outputted as

$$o_j = I_{j}^{o} - \theta^{o} \tag{2.8}$$

where w_{l}^{o} is the connection weight between the l-th node of the hidden layer and the output node. θ^{o} indicates the threshold on the output node. o_j is to be compared with actual value a_j. If $f()$ is the log-sigmoid function, a direct relationship between inputs and the output can be derived as

$$o_j = \sum_{l=1}^{L} \left(\frac{2 w_{l}^{o}}{1 + \mathrm{e}^{-2\left(\sum_{p=1}^{P} (w_{pl}^{h} x_{jp}) - \theta_{l}^{h}\right)}} - w_{l}^{o} \right) - \theta^{o} \tag{2.9}$$

Table 2.2 Cycle time-related data of 120 jobs

j	x_{j1}	x_{j2}	x_{j3}	x_{j4}	x_{j5}	x_{j6} (%)	a_j (h)
1	24	1223	158	807	99	84.20	953
2	23	1225	164	665	142	94.80	1248
3	25	1232	154	718	373	88.40	1299
…							
120	22	1319	159	777	326	88.80	1285

The direct relationship is not easy to understand or communicate.

Example 2.1 Cycle time-related data have been collected for 120 jobs and are presented in Table 2.2 [30], in which the cycle time of job j, a_j, is to be predicted based on the values of six job attributes, x_{j1}–x_{j6}, using an ANN. The ANN has three layers: the input layer, a single hidden layer (with twelve nodes), and an output layer. The Levenberg–Marquardt (LM) algorithm [31] is applied to train the ANN using MATLAB. The maximum number of epochs is 20,000. The target for RMSE is less than 100 h, or mean squared error (MSE) < 10,000 h^2:

$$\text{RMSE} = \sqrt{\frac{\sum_{j=1}^{n} (o_j - a_j)^2}{n}} \tag{2.10}$$

Equation (2.10) is slightly different from Eq. (2.3). Data of the first 80 jobs are used as the training data, while the remaining data are left for testing. The MATLAB code is shown in Fig. 2.4. The forecasting results are summarized in Fig. 2.5.

2.3.2 Partial Derivation, Odd Ratio

There are several types of XAI techniques for explaining ANN applications in job cycle time prediction:

- XAI techniques for assessing the contribution, influence, effect, importance of each input (i.e., job attribute) in predicting the output (i.e., the cycle time);
- XAI techniques for approximating the relationship between inputs and the output, i.e., the trained ANN, with simpler and more understandable rules;
- XAI techniques to facilitate the understanding of the prediction process (i.e., the ANN) and result (i.e. a cycle time forecast).

First, to analyze the effect of an input on the output, a traditional way is to conduct a correlation analysis. For example, if a linear relationship exists between inputs and the output, the Pearson correlation coefficient [32] between each input and the output can be calculated. The input with the highest correlation coefficient is most influential

```
% Colleated data
training_x=[24 23 …; 1223 1225 …; 158 164 …; 807 665 …; 99 142 …; 0.842 0.948 …];
training_y=[953 1248 …];
test_x=[24 22 …; 1251 1249 …; 182 184 …; 785 766 …; 127 200 …; 0.852 0.827 …];
test_y=[1173 1008 …];

% ANN configuration
net=feedforwardnet([12]);
net.dividefcn='dividetrain';
net.trainParam.lr=0.1;
net.trainParam.epochs=20000;
net.trainParam.goal=10000;

% Training
net=train(net,training_x,training_y);

% Forecasting
training_est_ct=net(training_x);
test_est_ct=net(test_x);

% Accuracy evaluation
rmse1=mean((test_y-test_est_ct).^2)^0.5;
```

Fig. 2.4 MATLAB code for implementing the ANN

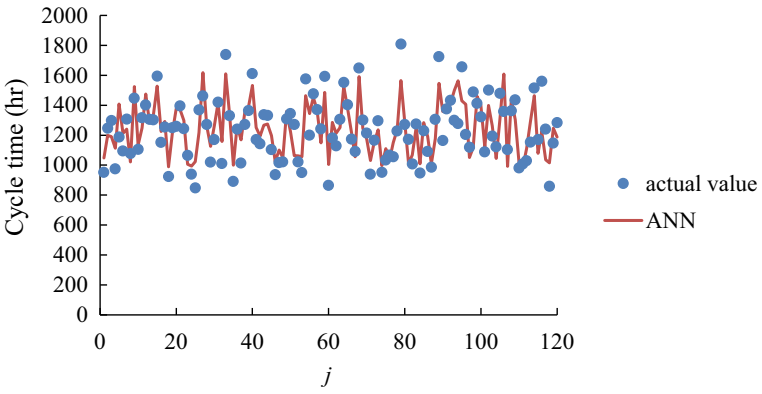


Fig. 2.5 Forecasting results

in determining the output. Otherwise, if the relationship between inputs and the output is nonlinear, Spearman's rank correlation [33] coefficient can be calculated instead.

To better interpret an ANN, Green et al. [34] evaluated the influence of each input/attribute on the output by taking the derivative of the output with respect to the input $\partial o_j / \partial x_{jp}$, where o_j is the output of example j and x_{jp} is attribute p of example j. The input with the highest partial derivative is the most influential attribute.

Example 2.2 In Example 2.1, the value of each attribute is varied by 1% for all jobs in test data to evaluate the average change of cycle time forecasts. The MATLAB code is presented in Fig. 2.6. The results are summarized in Fig. 2.7. Obviously, the second attribute has the greatest influence on cycle time forecasts.

To generate a local explanation, Green et al. calculated the odd ratio of each input:

$$\mathrm{OR}_p = \frac{o_j(x_{jp} + \sigma_p) \cdot (1 - o_j(x_{jp}))}{o_j(x_{jp}) \cdot (1 - o_j(x_{jp} + \sigma_p))} \tag{2.11}$$

```
pd3=zeros(1,6);
for p=1:6
   % Increase the value of the attribute by 1%
   test_x2=test_x;
   test_x2(p,:)=test_x2(p,:)*1.01;

   % Re-forecast cycle times
   test_est_ct2=net(test_x2);

   % Evaluate the average (absolute) change
   pd1=mean(abs(test_est_ct2-test_est_ct));

   % Decrease the value of the attribute by 1%
   test_x3=test_x;
   test_x3(p,:)=test_x3(p,:)*0.99;

   % Re-forecast cycle times
   test_est_ct3=net(test_x3);

   % Evaluate the average (absolute) change
   pd2=mean(abs(test_est_ct3-test_est_ct));

   % Average the evaluation results
   pd3(1,p)=(pd1+pd2)/2;
end
```

Fig. 2.6 MATLAB code for implementing partial derivation

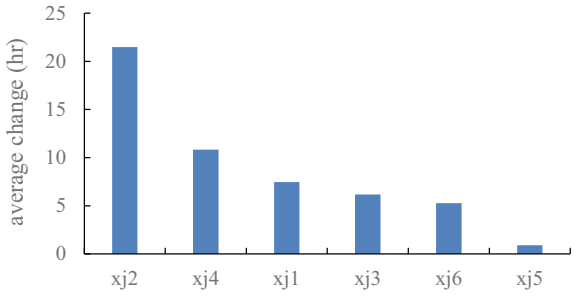


Fig. 2.7 Comparison of the influences of attributes on cycle time forecasts

where $o_j(\Omega)$ means the output of example j when $x_{jp} = \Omega$. $o_j(\Omega)$ may need to be normalized. The input with the highest odd ratio was the most influential attribute for this example.

Example 2.3 Following Example 2.2, the odd ratios of inputs for each job are calculated. Cycle time forecasts are divided by the maximum of cycle times, 1810 h, to normalize their values. The MATLAB code is presented in Fig. 2.8. Based on the results, the most influential attribute for each job can be determined. The results are summarized in Table 2.3. As expected, the second attribute is the most influential attribute for most jobs.

```
test_est_ct2=zeros(6,40);
test_est_ct3=zeros(6,40);
for p=1:6
   test_x2=test_x;
   test_x2(p,:)=test_x2(p,:)*1.01;
   test_est_ct2(p,:)=net(test_x2);
   test_x3=test_x;
   test_x3(p,:)=test_x3(p,:)*0.99;
   test_est_ct3(p,:)=net(test_x3);
end

or1=zeros(6,40);
or2=zeros(6,40);
or3=zeros(6,40);

for i=1:40
      for p=1:6
      % Calculate OR

or1(p,i)=(test_est_ct2(p,i)/1810*(1-test_est_ct(1,i)/1810))/((1-test_est_ct2(p,i)/1810)*test_est_ct(1,i)/1
810);

or2(p,i)=(test_est_ct3(p,i)/1810*(1-test_est_ct(1,i)/1810))/((1-test_est_ct3(p,i)/1810)*test_est_ct(1,i)/1
810);

      % Average the results
      or3(p,i)=(or1(p,i)+or2(p,i))/2;
   end
end

% Determine the most influential attribute for each job
c=ones(1,40);
for i=1:40
   c(1,i)=1;
   max1=-999;
   for p=1:6
      if or3(p,i)>max1
         c(1,i)=p;
         max1=or3(p,i);
      end
   end
end
```

Fig. 2.8 MATLAB code for calculating the odd ratios of inputs for each job

Table 2.3 Most influential attribute for each job

j	Most influential attribute
81	x_{j2}
82	x_{j2}
83	x_{j2}
…	
118	x_{j4}
119	x_{j2}
120	x_{j2}

2.3.3 *Out-of-Bag (OOB) Predictor Importance*

Similar to partial derivation, out-of-bag (OOB) predictor importance also compares the effects of varying inputs on the output [35]. OOB data are unlearned data (i.e., test data). Partial derivation evaluates the change in the value of the output, while OOB predictor importance considers the change in the forecasting error $e_j = a_j - o_j$:

Step 1. Consider the first input (attribute).
Step 2. (For each example in test data) Vary the value of the input by $q\%$, while fixing the values of the other inputs.
Step 3. (For each example in test data) Apply the trained ANN to predict the output.
Step 4. (For each example in test data) Calculate the change in the forecasting error $\varepsilon_{jp} = \left| e_{j\,(\text{before change input } p)} - e_{j\,(\text{after change input } p)} \right|$.
Step 5. The importance of input p can be evaluated as $\overline{\varepsilon}_{\cdot p}/s_{\varepsilon_{\cdot p}}$, where $\overline{\varepsilon}_{\cdot p}$ and $s_{\varepsilon_{\cdot p}}$ are the average and standard deviation of ε_{jp}, respectively.
Step 6. Consider the next input, and return to Step 2.

Example 2.4 Following Example 2.2, the importance of each feature is evaluated using OOB predictor importance instead. The required MATLAB code is provided in Fig. 2.9. The results are summarized in Fig. 2.10. Obviously, the second attribute is the most important feature.

2.3.4 *Recursive Feature Elimination (RFE)*

Recursive feature elimination (RFE), like backward elimination regression analysis [36], searches for a subset of features by starting with all features in the training dataset and then removing features until the required number of features remains [37]. The remaining features are considered more important than the others. In fact, some machine learning algorithms may be misled by irrelevant input features, resulting in worse predictive performance for unlearned data.

```
% Calculate forecasting error before change
ejb=test_y-test_est_ct;

importance1=zeros(1,6);
for p=1:6
   % Increase the value of the attribute by 1%
   test_x2=test_x;
   test_x2(p,:)=test_x2(p,:)*1.01;

   % Re-forecast cycle times
   test_est_ct2=net(test_x2);

   % Calculate forecasting error after change
   eja=test_y-test_est_ct2;

   % Evaluate the OOB predictor importance of the attribute
   ejp=abs(ejb-eja);
   importance1(1,p)=mean(ejp)/std(ejp);
end
```

Fig. 2.9 MATLAB code for evaluating the OOB predictor importance of each attribute

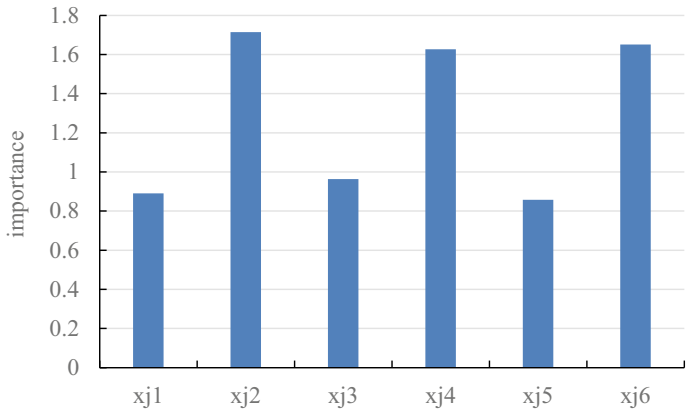


Fig. 2.10 OOB predictor importance of each feature

An example is given in Fig. 2.11. Originally, the ANN has five inputs. After training, the RMSE for the training data is 354.2. The required number of features is four. After removing one feature at a time, the ANN is retrained and the RMSE for the training data is reevaluated. Finally, by removing the fourth feature (x_{j4}), the RMSE is optimized. In addition, the required number of features is reached. As a result, x_{j1}, x_{j2}, x_{j3}, and x_{j5} are more important than x_{j4}. In addition, the contribution of each feature can be evaluated as:

$$\text{Contribution of a feature} = 1 - \frac{\text{RMSE (before removing the feature)}}{\text{RMSE (after removing the feature)}} \cdot 100\% \tag{2.12}$$

The evaluation results are summarized in Table 2.4.

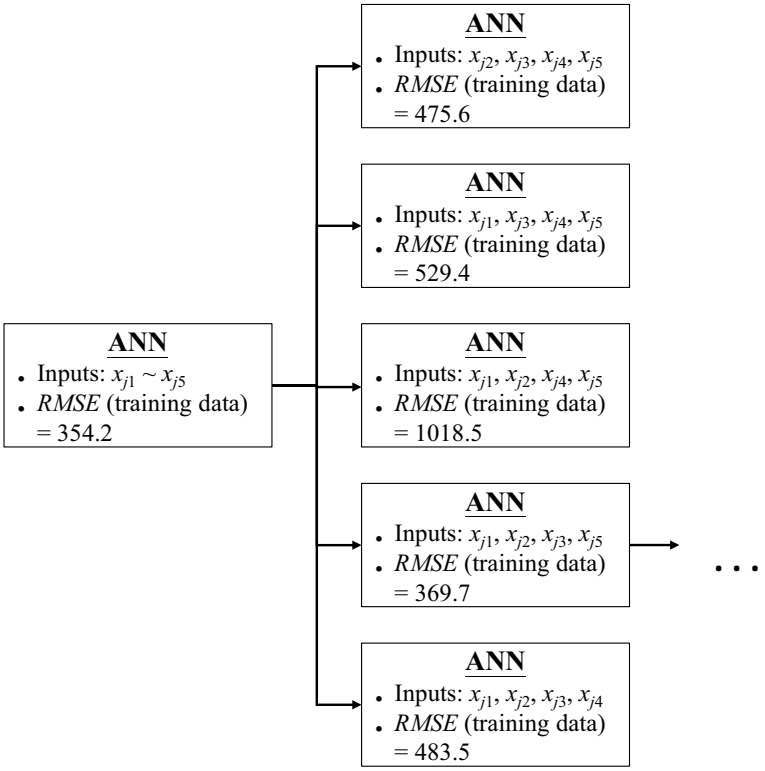


Fig. 2.11 Concept of RFE

Table 2.4 Contribution of each attribute

j	Contribution (%)
x_{j1}	26
x_{j2}	33
x_{j3}	65
x_{j4}	4
x_{j5}	27

2.3.5 Shapley Additive Explanations (SHAP)

Shapley additive explanations (SHAP) can be applied to make a local explanation [38]. The Shapley value of an input reflects the input's contribution and is determined as follows:

Step 1. Consider the first input.
Step 2. Fix the value of the input, and randomize the values of the other inputs to generate random data.
Step 3. Apply the trained ANN to predict the output of each randomly generated data.
Step 4. Calculate the average value of the outputs.
Step 5. Subtract the average output from the output of the example, and divide the result by P, where P is the number of inputs: The result gives the Shapley value of the input.
Step 6. Consider the next input, and return to Step 2.

Example 2.5 An ANN with three inputs $x_{j1}-x_{j3}$ is trained to predict the cycle time of a job o_j. For a job with inputs (16, 342, 0.85), the cycle time is predicted as 687 h. To determine the Shapley value of each input for the job, 15 random data (five for each input) are generated, as shown in Table 2.5. The trained ANN is applied to predict the cycle time of each randomly generated data. Then, the average cycle time forecasts of all randomly generated data are calculated as 776 h. The Shapley value of each input is determined as.

$$x_{j1}: (687 - 919)/3 = -77.3$$
$$x_{j2}: (687 - 640)/3 = 15.7$$
$$x_{j3}: (687 - 769)/3 = -27.2$$

The results are compared in Fig. 2.12. In this example, x_{j2} has the highest influence, while x_{j1} has the least. In other words, x_{j2} is the most influential input for this job, which is interpreted as "the value of x_{j2} lengthens the cycle time of the job most". In addition, the sum of all Shapley values explains the deviation between the average output and the cycle time forecast:

$$-77.3 + 15.7 + (-27.2) = 687 - 776$$

2.3.6 CBR

Kenny and Keane [39] established a CBR system to approximate the operations in an ANN. The CBR system was composed of cases $\{(\{r_{kp}|p = 1-P\}, s_k)|k = 1-K\}$

Table 2.5 Randomly generated data

x_{j1}	x_{j2}	x_{j3}	o_j (h)
16	961	0.91	1107
16	460	0.93	740
16	780	0.78	953
16	620	0.64	761
16	938	0.69	1034
		Average	919
x_{j1}	x_{j2}	x_{j3}	o_j (h)
18	*342*	0.92	715
12	*342*	0.78	584
15	*342*	0.51	580
20	*342*	0.65	637
13	*342*	0.82	684
		Average	640
x_{j1}	x_{j2}	x_{j3}	o_j (h)
20	693	*0.85*	950
19	375	*0.85*	720
15	219	*0.85*	610
16	706	*0.85*	883
18	291	*0.85*	681
		Average	769

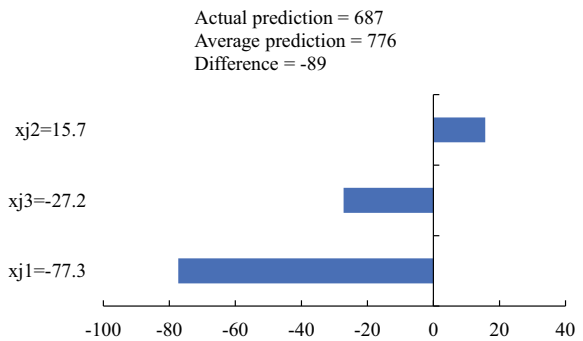


Fig. 2.12 Shapley values for the job

that were used in K rules

$$\text{“If } x_{j1} = r_{k1} \text{ and} \ldots \text{and } x_{jP} = r_{kP} \text{ then } \hat{o}_j = s_k\text{”}; k = 1-K \tag{2.13}$$

where $\hat{o}_j$ is the approximated output of example j. After training the ANN, synthetic examples were randomly generated and fed into the ANN to predict their outputs

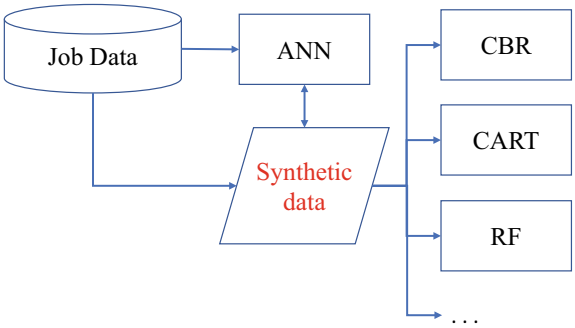


Fig. 2.13 Generation of synthetic data

(see Fig. 2.13). XAI techniques are then applied to fit the input–output relationship in synthetic data.

Then, k-means [40] was applied to extract cases from synthetic examples. To approximate the output of example j, the distance between example j and case k was measured:

$$d_{jk} = \sqrt{\sum_{p=1}^{P} (x_{jp} - r_{kp})^2} \tag{2.14}$$

If attributes were of unequal importance,

$$d_{jk} = \sqrt{\sum_{p=1}^{P} (w_p (x_{jp} - r_{kp})^2)} \tag{2.15}$$

where w_p was the importance/weight of attribute p. Then, the ANN output of this example was approximated as

$$\hat{o}_j = \sum_{k=1}^{K} \left(\frac{\frac{1}{d_{jk}}}{\sum_{l=1}^{K} \frac{1}{d_{jl}}} \cdot s_k \right) \tag{2.16}$$

Kenny and Keane [39] compared seven global and local methods to assign weights to various attributes: sensitivity (SENS), perturbation (PTB), connection weights (CW), local linear model (LLM), contributions oriented local explanations (COLE) C-LPR, C-IG, and C-DeepLIFT. Only the inputs and output of the ANN were considered in the rules. Therefore, the CBR approximation technique can be applied to more complicated ANNs such as DNNs (ANNs with multiple hidden layers, recurrent neural networks, fuzzy neural networks, pre-classifying or post-classifying ANNs, etc.). For example, Kenny and Keane [39] established a CBR system to approximate a CNN. However, the content of rules in such a CBR system is often too simplistic to provide an accurate approximation.

Table 2.6 Synthetic data

J	x_{j1}	x_{j2}	x_{j3}	x_{j4}	x_{j5}	x_{j6} (%)	a_j (h)
1	22	1320	165	748	421	85	1298
2	23	1261	169	734	198	81	1098
3	22	1187	159	635	287	93	1122
…							
200	24	1309	175	734	345	81	1276

Example 2.6 In Example 2.1, the train ANN is applied to generate 200 synthetic examples by randomizing the values of inputs and then forecasting the output. The results are summarized in Table 2.6. KM is then applied to cluster synthetic data into ten clusters. The required MATLAB code is provided in Fig. 2.14. The clustering results are summarized in Fig. 2.15. The centers of these clusters are used as cases. Based on these cases, the corresponding rules are generated, as summarized in Table 2.7. After applying all the ten rules, the ANN output is approximated, as summarized in Fig. 2.16. The effectiveness of CBR in approximating the ANN is evaluated in terms of the approximation accuracy:

$$\text{MAE} = 118.8 \text{ h}$$

```
synthetic_data=[22 1320 165 748 421 0.84; …; 24 1309 175 734 345 0.81]

% k-means
[idx, c, SSD]=kmeans(synthetic_data, 10);
```

Fig. 2.14 MATLAB code for implementing KM

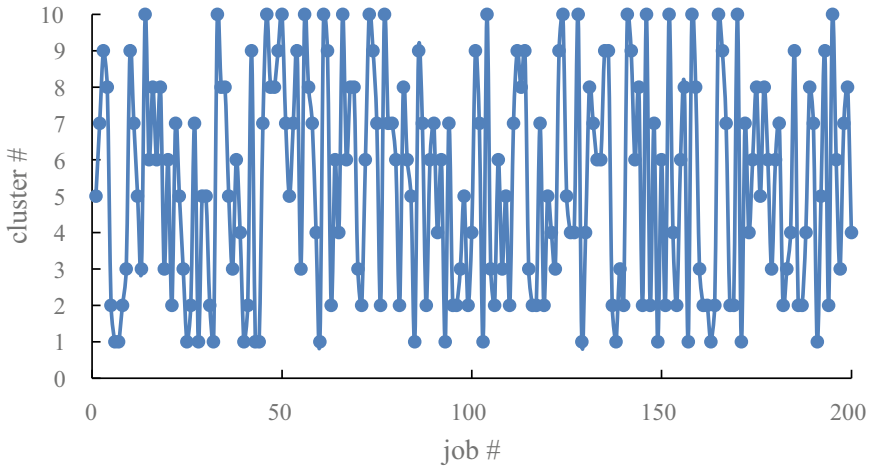


Fig. 2.15 Clustering results

Table 2.7 Generated CBR rules

Rule #	Rule content
1	If xj1 = 22.5 And xj2 = 1216.6 And xj3 = 169.6 And xj4 = 782 And xj5 = 307.5 And xj6 = 0.87 Then oj = 1067.3
2	If xj1 = 22.5 And xj2 = 1233 And xj3 = 169.1 And xj4 = 738.6 And xj5 = 97 And xj6 = 0.89 Then oj = 1066.2
3	If xj1 = 22.8 And xj2 = 1357.4 And xj3 = 172.8 And xj4 = 680.5 And xj5 = 391.5 And xj6 = 0.91 Then oj = 1528.5
4	If xj1 = 22.4 And xj2 = 1343 And xj3 = 171.5 And xj4 = 782.2 And xj5 = 344.4 And xj6 = 0.9 Then oj = 1299.4
5	If xj1 = 22.9 And xj2 = 1305.7 And xj3 = 167.5 And xj4 = 796.1 And xj5 = 436.6 And xj6 = 0.88 Then oj = 1238.1
6	If xj1 = 22.4 And xj2 = 1342.2 And xj3 = 172.3 And xj4 = 695.1 And xj5 = 105.4 And xj6 = 0.91 Then oj = 1323.6
7	If xj1 = 22.8 And xj2 = 1333.4 And xj3 = 168 And xj4 = 750.1 And xj5 = 229 And xj6 = 0.91 Then oj = 1289.4
8	If xj1 = 22.5 And xj2 = 1215.8 And xj3 = 168.7 And xj4 = 697.5 And xj5 = 428.8 And xj6 = 0.89 Then oj = 1139.9
9	If xj1 = 22.2 And xj2 = 1230.6 And xj3 = 169.4 And xj4 = 670.2 And xj5 = 236.3 And xj6 = 0.89 Then oj = 1229.5
10	If xj1 = 22.6 And xj2 = 1235 And xj3 = 171.8 And xj4 = 686.2 And xj5 = 340.4 And xj6 = 0.88 Then oj = 1244.7

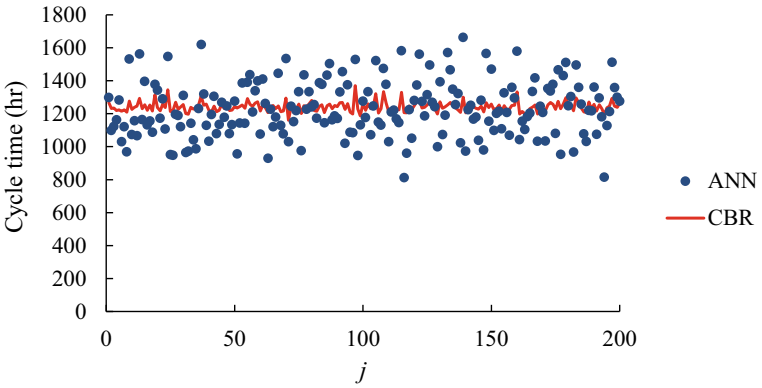


Fig. 2.16 Comparison of the ANN output and the approximated output using CBR

MAPE = 10.0%
RMSE = 146.0 h

It is noteworthy that the approximation accuracy may be enhanced using only the rules of clusters close to a job, or assigning unequal weights to different attributes.

Albawi et al. [41] gave a very detailed explanation of CNNs by

- Giving text descriptions;
- Introducing the matrix operations using filters;
- Providing formulas;
- Displaying the results of applying filters to an image;
- Displaying the part of the image where the filter takes effect;
- Comparing various transformation/activation functions.

However, as they mentioned, such an explanation assumes that the reader has sufficient knowledge of both machine learning and ANNs.

2.3.7 *Classification and Regression Tree (CART)*

CARTs have also been constructed to approximate ANNs with rules like [42]

$$\text{“If } x_{j(1)} \geq (\text{or } <) r_{k(1)} \text{ and} \ldots \text{and } x_{j(L)} \geq (\text{or } <) r_{k(L)} \text{ then } \hat{o}_j = s_k\text{”}; k = 1-K \tag{2.17}$$

Not all attributes will appear in such a rule. In addition, the order in which attributes appear in different rules may not be the same. The rule content of (2.17) is more complicated than that of (2.13). However, there is no aggregation mechanism like (2.16). The procedure for constructing a CART is composed of three stages: tree growing, stopping, and pruning. A tree is grown using a recursive partitioning technique that selects a variable and a split point according to the prespecified criterion. Common criteria include: Gini, towing, ordered towing, and maximum-deviance reduction [43].

Let Ω_i be the set of synthetic samples that belong to node i that is to be split into two branches with sets Ω_i^L and Ω_i^R. For either set, the predicted cycle times of all synthetic samples are averaged as the predicted value:

$$p_i^{L/R} = \frac{\sum_{j \in \Omega_i^{L/R}} o_j}{\left|\Omega_i^{L/R}\right|} \tag{2.18}$$

The split that minimizes the sum of squared errors (SSE) is chosen:

$$\text{SSE} = \sum_i \sum_{j \in \Omega_i} (o_j - p_i)^2 \tag{2.19}$$

Starting from the largest tree, the CART is pruned until the following objective function is minimized:

$$\text{Min } C = \sqrt{\frac{\text{SSE}}{n}} + \alpha |S| \tag{2.20}$$

where $|S|$ indicates the size of the CART (in terms of the number of nodes or branches); α is a positive constant.

Example 2.7 In Example 2.6, the synthetic data are fitted by a CART instead. The required MATLAB code is provided in Fig. 2.17. The construct CART contains 79 nodes, as illustrated in Fig. 2.18. The number of rules (branches) is 40. The rule contents are summarized in Table 2.8. The ANN output and the approximated output are compared in Fig. 2.19. The effectiveness of the XAI technique can be evaluated in terms of the approximation accuracy:

$$\text{MAE} = 33.2 \text{ h}$$
$$\text{MAPE} = 2.8\%$$
$$\text{RMSE} = 45.1 \text{ h}$$

```
y=[1298; 1097; …; 1275];

% CART
ct_tree=fitrtree([22 1320 165 748 421 0.85; …; 24 1309 175 734 345 0.81],y);

% Show the tree
view(ct_tree,'Mode','graph')

% Show the rules
view(ct_tree)

% Make prediction
yj=predict(ct_tree,[22 1320 165 748 421 0.85; …; 24 1309 175 734 345 0.81]);
```

Fig. 2.17 MATLAB code for constructing the CART

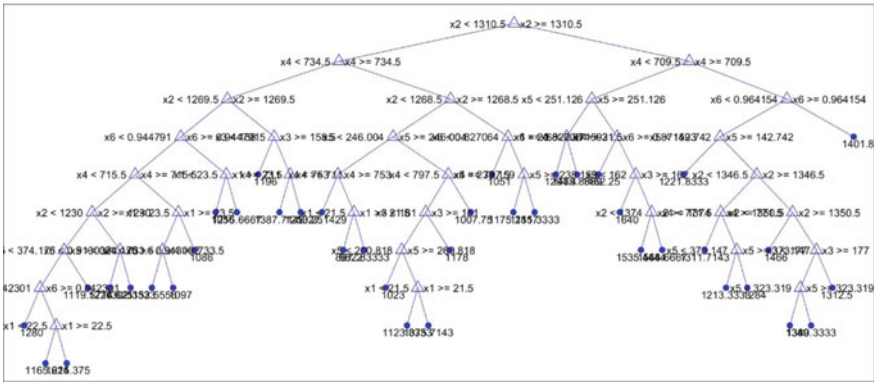


Fig. 2.18 CART for the synthetic data

Table 2.8 CART rules

Node #	Rule content
1	If xj1 < 22.5 And xj2 < 1230 And xj4 < 715.5 And xj5 < 374.175 And xj6 < 0.842301 Then oj = 1280
2	If xj1 < 22.5 And xj2 < 1230 And xj4 < 715.5 And xj5 < 374.175 And 0.842301 <= xj6 < 0.944791 Then oj = 1165.625
…	
40	If xj2 >= 1230 And xj4 >= 709.5 And xj6 >= 0.964154 Then oj = 1410.8

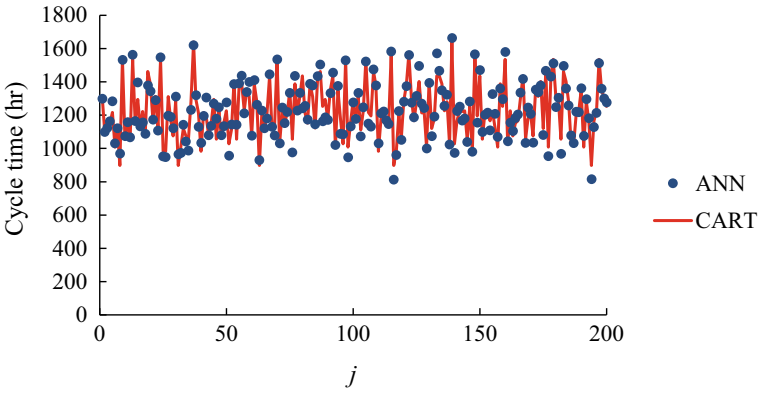


Fig. 2.19 Comparison of the ANN output and the approximated output using CART

2.3.8 Random Forest (RF)

A random forest (RF) is the ensemble of multiple trees that consider different parts of data and try various combinations of attributes [44]. For an example, a decision rule in each tree can be applied to predict the output of the example. Therefore, multiple decision rules can be applied to predict the output of an example. Then, the prediction results by all trees are averaged. Therefore, a RF is usually more accurate than a single CART, and is expected to approximate an ANN better. Using a RF, the ANN output of example j is predicted by a rule in each tree. As a result, multiple rules are used to approximate the ANN output for an example:

$$\begin{aligned} &\text{“If } x_{j(1)}(t) \geq (\text{or } <) r_{k(1)}(t) \text{ and} \ldots \text{and } x_{j(L)}(t) \geq (\text{or } <) r_{k(L)}(t) \\ &\quad \text{then } \hat{o}_j(t) = s_k(t)\text{”}; \; t = 1{-}T \end{aligned} \tag{2.21}$$

and

$$\hat{o}_j = \frac{\sum_{t=1}^{T} \hat{o}_j(t)}{T} \tag{2.22}$$

```
synthetic_x=[22 1320 165 748 421 0.85; …; 24 1309 175 734 345 0.81];
synthetic_y=[1298; …; 1275];

% Random forest
RF1=TreeBagger(10,synthetic_x,synthetic_y,Method="regression",OOBPrediction="on");

% Show the first tree
view(RF1.Trees{1},Mode="graph");

% Show rules in the first tree
view(RF1.Trees{1},Mode="text");

% Make prediction
synthetic_est_ct=predict(RF1,synthetic_x);
```

Fig. 2.20 MATLAB code for constructing the RF

where t is the index of a tree. More than one rule applies to a sample, which can be confusing for the user.

Example 2.8 Following Example 2.7, the synthetic data are now fitted by a RF instead. The required MATLAB code is provided in Fig. 2.20. The RF, containing ten CARTs, is shown in Fig. 2.21. Therefore, for each job, ten decision rules can be applied to approximate its cycle time forecasted by the ANN. Then, the results are averaged. The rule contents are summarized in Table 2.9. The ANN output and the approximated output using RF are compared in Fig. 2.22.

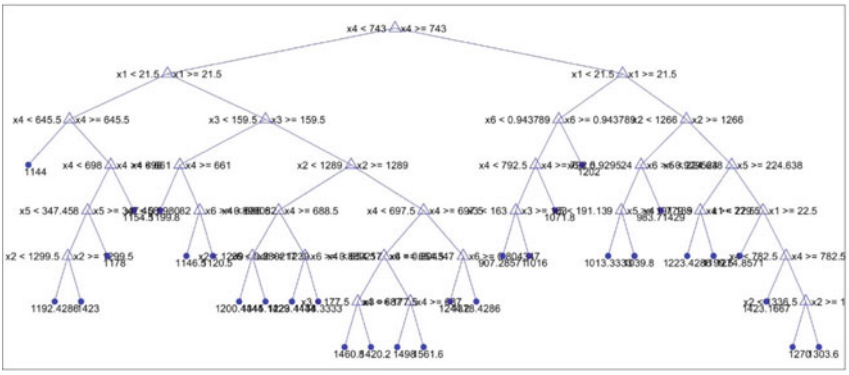
2.3.9 *Gradient Boosted Decision Trees, eXtreme Gradient Boosting (XGBoost)*

Gradient boosted decision trees [45] consist of multiple sequentially trained decision trees. Starting from the first decision tree, each subsequent decision tree improves the previous decision tree by predicting the prediction (approximation) error generated by the previous decision tree (the concept of boosting). The training of each decision tree is the same as described in the previous sections and will not be repeated here.

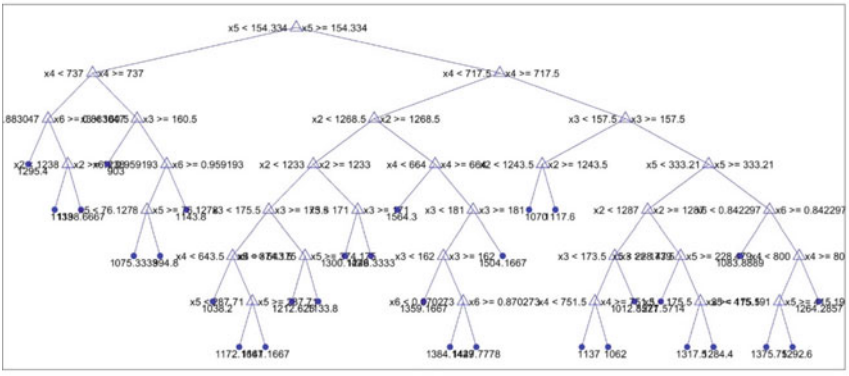
An example is given in Table 2.10, in which the ANN output of job j is $o_j = 2077$. From the first decision tree, a rule applies to job j and generates an approximation of o_j as $\hat{o}_j = 2050$. The approximation error for this job is $e_j^{(1)} = o_j - \hat{o}_j = 27$, which is to be predicted by the second decision tree. A rule from the second decision tree generates a prediction of $e_j^{(1)}$ as $\hat{e}_j^{(1)} = 36$. The approximation error is $e_j^{(2)} = e_j^{(1)} - \hat{e}_j^{(1)} = -9$, which is to be predicted by the third decision tree, giving $\hat{e}_j^{(2)} = -10$. Finally, the ANN output is approximated by $2050 + 36 - 10 = 2076$.

Gradient boosted decision trees can be enhanced by considering different parts of samples, especially samples with large prediction errors (the concept of gradient), and

(CART #1)



(CART #2)



…

(CART #10)

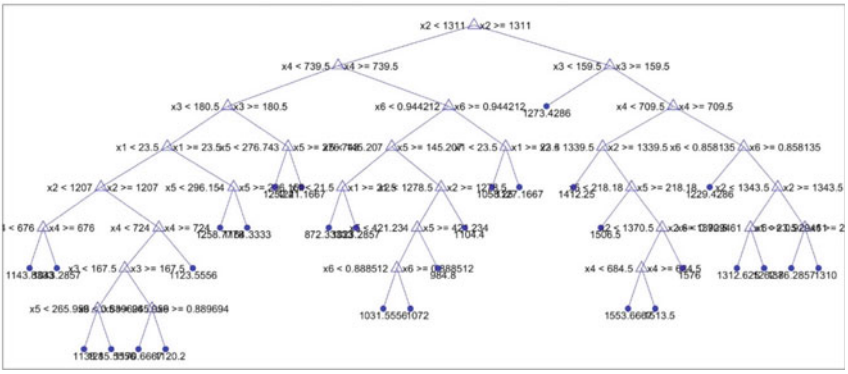


Fig. 2.21 Construct RF

Table 2.9 Rule contents

CART #	Rule contents
1	1 If xj4 < 743 Then node 2 Elseif xj4 >= 743 Then node 3 Else 1211.07 2 If xj1 < 21.5 Then node 4 Elseif xj1 >= 21.5 Then node 5 Else 1263.2 3 If xj1 < 21.5 Then node 6 Elseif xj1 >= 21.5 Then node 7 Else 1140.54 4 If xj4 < 645.5 Then node 8 Elseif xj4 >= 645.5 Then node 9 Else 1204.24 5 If xj3 < 159.5 Then node 10 Elseif xj3 >= 159.5 Then node 11 Else 1286.93 6 If xj6 < 0.943789 Then node 12 Elseif xj6 >= 0.943789 Then node 13 Else 1048.8 7 If xj2 < 1266 Then node 14 Elseif xj2 >= 1266 Then node 15 Else 1178.77 8 oj = 1144 9 If xj4 < 698 Then node 16 Elseif xj4 >= 698 Then node 17 Else 1223.52 10 If xj4 < 661 Then node 18 Elseif xj4 >= 661 Then node 19 Else 1152.32 11 If xj2 < 1289 Then node 20 Elseif xj2 >= 1289 Then node 21 Else 1327.52 12 If xj4 < 792.5 Then node 22 Elseif xj4 >= 792.5 Then node 23 Else 989.222 13 oj = 1202 14 If xj6 < 0.929524 Then node 24 Elseif xj6 >= 0.929524 Then node 25 Else 1015.83 15 If xj5 < 224.638 Then node 26 Elseif xj5 >= 224.638 Then node 27 Else 1280.05 16 If xj5 < 347.458 Then node 28 Elseif xj5 >= 347.458 Then node 29 Else 1256 17 oj = 1154.5 18 oj = 1199.8 19 If xj6 < 0.898082 Then node 30 Elseif xj6 >= 0.898082 Then node 31 Else 1135.36 20 If xj4 < 688.5 Then node 32 Elseif xj4 >= 688.5 Then node 33 Else 1227.77 21 If xj4 < 697.5 Then node 34 Elseif xj4 >= 697.5 Then node 35 Else 1424.16 22 If xj3 < 163 Then node 36 Elseif xj3 >= 163 Then node 37 Else 957.462 23 oj = 1071.8 24 If xj5 < 191.139 Then node 38 Elseif xj5 >= 191.139 Then node 39 Else 1029.88 25 oj = 983.714 26 If xj4 < 779.5 Then node 40 Elseif xj4 >= 779.5 Then node 41 Else 1212.38 27 If xj1 < 22.5 Then node 42 Elseif xj1 >= 22.5 Then node 43 Else 1316.71 28 If xj2 < 1299.5 Then node 44 Elseif xj2 >= 1299.5 Then node 45 Else 1288.5 29 oj = 1178 30 oj = 1146.5 31 oj = 1120.5 32 If xj2 < 1239 Then node 46 Elseif xj2 >= 1239 Then node 47 Else 1263.75 33 If xj6 < 0.880217 Then node 48 Elseif xj6 >= 0.880217 Then node 49 Else 1189.4 34 If xj4 < 654.5 Then node 50 Elseif xj4 >= 654.5 Then node 51 Else 1485.15 35 If xj6 < 0.804347 Then node 52 Elseif xj6 >= 0.804347 Then node 53 Else 1322.5 36 oj = 907.286 37 oj = 1016 38 oj = 1013.33 39 oj = 1039.8 40 oj = 1223.43 41 oj = 1199.5 42 oj = 1274.86 43 If xj4 < 782.5 then node 54 Elseif xj4 >= 782.5 then node 55 Else 1333.94 44 oj = 1192.43 45 oj = 1423 46 oj = 1200.44 47 oj = 1345.14 48 oj = 1223.44 49 oj = 1138.33

(continued)

Table 2.9 (continued)

CART #	Rule contents
	50 If xj3 < 177.5 Then node 56 Elseif xj3 >= 177.5 Then node 57 Else 1440.5 51 If xj4 < 687 Then node 58 Elseif xj4 >= 687 Then node 59 Else 1529.8 52 oj = 1244.2 53 oj = 1378.43 54 oj = 1423.17 55 If xj2 < 1336.5 Then node 60 Elseif xj2 >= 1336.5 Then node 61 Else 1285.27 56 oj = 1460.8 57 oj = 1420.2 58 oj = 1498 59 oj = 1561.6 60 oj = 1270 61 oj = 1303.6
2	1 If xj5 < 154.334 Then node 2 Elseif xj5 >= 154.334 Then node 3 Else 1226.97 2 If xj4 < 737 Then node 4 Elseif xj4 >= 737 Then node 5 Else 1130.26 3 If xj4 < 717.5 Then node 6 Elseif xj4 >= 717.5 Then node 7 Else 1252.68 4 If xj6 < 0.883047 Then node 8 Elseif xj6 >= 0.883047 Then node 9 Else 1229.1 5 If xj3 < 160.5 Then node 10 Elseif xj3 >= 160.5 Then node 11 Else 1031.43 6 If xj2 < 1268.5 Then node 12 Elseif xj2 >= 1268.5 Then node 13 Else 1315.58 7 If xj3 < 157.5 Then node 14 Elseif xj3 >= 157.5 Then node 15 Else 1186.51 8 oj = 1295.4 9 If xj2 < 1238 Then node 16 Elseif xj2 >= 1238 Then node 17 Else 1168.82 10 oj = 903 11 If xj6 < 0.959193 Then node 18 Elseif xj6 >= 0.959193 Then node 19 Else 1071.56 12 If xj2 < 1233 Then node 20 Elseif xj2 >= 1233 Then node 21 Else 1186.51 13 If xj4 < 664 Then node 22 Elseif xj4 >= 664 Then node 23 Else 1461.63 14 If xj2 < 1243.5 Then node 24 Elseif xj2 >= 1243.5 Then node 25 Else 1091.64 15 If xj5 < 333.21 Then node 26 Elseif xj5 >= 333.21 Then node 27 Else 1202.32 16 oj = 1133 17 oj = 1198.67 18 If xj5 < 76.1278 Then node 28 Elseif xj5 >= 76.1278 Then node 29 Else 1038.73 19 oj = 1143.8 20 If xj3 < 175.5 Then node 30 Elseif xj3 >= 175.5 Then node 31 Else 1148.03 21 If xj3 < 171 Then node 32 Elseif xj3 >= 171 Then node 33 Else 1275.31 22 oj = 1564.3 23 If xj3 < 181 Then node 34 Elseif xj3 >= 181 Then node 35 Else 1424.96 24 oj = 1070 25 oj = 1117.6 26 If xj2 < 1287 Then node 36 Elseif xj2 >= 1287 Then node 37 Else 1169.7 27 If xj6 < 0.842297 Then node 38 Elseif xj6 >= 0.842297 Then node 39 Else 1243.93 28 oj = 1075.33 29 oj = 994.8 30 If xj4 < 643.5 Then node 40 Elseif xj4 >= 643.5 Then node 41 Else 1121.82 31 If xj5 < 374.175 Then node 42 Elseif xj5 >= 374.175 Then node 43 Else 1182.31 32 oj = 1300.14 33 oj = 1246.33 34 If xj3 < 162 Then node 44 Elseif xj3 >= 162 Then node 45 Else 1403.36 35 oj = 1504.17

(continued)

Table 2.9 (continued)

CART #	Rule contents
	36 If xj3 < 173.5 Then node 46 Elseif xj3 >= 173.5 Then node 47 Else 1071.53 37 If xj5 < 228.479 Then node 48 Elseif xj5 >= 228.479 Then node 49 Else 1273.33 38 oj = 1083.89 39 If xj4 < 800 Then node 50 Elseif xj4 >= 800 Then node 51 Else 1315.95 40 oj = 1038.2 41 If xj5 < 287.71 Then node 52 Elseif xj5 >= 287.71 Then node 53 Else 1156.67 42 oj = 1212.62 43 oj = 1133.8 44 oj = 1359.17 45 If xj6 < 0.870273 Then node 54 Elseif xj6 >= 0.870273 Then node 55 Else 1419.94 46 If xj4 < 751.5 Then node 56 Elseif xj4 >= 751.5 Then node 57 Else 1105.75 47 oj = 1012.86 48 oj = 1227.57 49 If xj3 < 175.5 Then node 58 Elseif xj3 >= 175.5 Then node 59 Else 1302.45 50 If xj5 < 415.191 Then node 60 Elseif xj5 >= 415.191 Then node 61 Else 1343.77 51 oj = 1264.29 52 oj = 1172.17 53 oj = 1141.17 54 oj = 1384.14 55 oj = 1447.78 56 oj = 1137 57 oj = 1062 58 oj = 1317.5 59 oj = 1284.4 60 oj = 1375.75 61 oj = 1292.6
…	
10	1 If xj2 < 1311 Then node 2 Elseif xj2 >= 1311 Then node 3 Else 1213.8 2 If xj4 < 739.5 Then node 4 Elseif xj4 >= 739.5 Then node 5 Else 1109.39 3 If xj3 < 159.5 Then node 6 Elseif xj3 >= 159.5 Then node 7 Else 1395.44 4 If xj3 < 180.5 Then node 8 Elseif xj3 >= 180.5 Then node 9 Else 1172.05 5 If xj6 < 0.944212 Then node 10 Elseif xj6 >= 0.944212 Then node 11 Else 1024.69 6 oj = 1273.43 7 If xj4 < 709.5 Then node 12 Elseif xj4 >= 709.5 Then node 13 Else 1408.38 8 If xj1 < 23.5 Then node 14 Elseif xj1 >= 23.5 Then node 15 Else 1158.9 9 If xj5 < 276.743 Then node 16 Elseif xj5 >= 276.743 Then node 17 Else 1246.18 10 If xj5 < 145.207 Then node 18 Elseif xj5 >= 145.207 Then node 19 Else 1002.6 11 If xj1 < 23.5 Then node 20 Elseif xj1 >= 23.5 Then node 21 Else 1087.79 12 If xj2 < 1339.5 Then node 22 Elseif xj2 >= 1339.5 Then node 23 Else 1510.24 13 If xj6 < 0.858135 Then node 24 Elseif xj6 >= 0.858135 Then node 25 Else 1300.16 14 If xj2 < 1207 Then node 26 Elseif xj2 >= 1207 Then node 27 Else 1139.09 15 If xj5 < 296.154 Then node 28 Elseif xj5 >= 296.154 Then node 29 Else 1221 16 oj = 1252.2 17 oj = 1241.17 18 If xj1 < 21.5 Then node 30 Elseif xj1 >= 21.5 Then node 31 Else 938.375 19 If xj2 < 1278.5 Then node 32 Elseif xj2 >= 1278.5 Then node 33 Else 1045.42 20 oj = 1058.25

(continued)

Table 2.9 (continued)

CART #	Rule contents
	21 oj = 1127.17 22 oj = 1412.25 23 If xj5 < 218.18 Then node 34 Elseif xj5 >= 218.18 Then node 35 Else 1540.38 24 oj = 1229.43 25 If xj2 < 1343.5 Then node 36 Elseif xj2 >= 1343.5 Then node 37 Else 1319.96 26 If xj4 < 676 Then node 38 Elseif xj4 >= 676 Then node 39 Else 1089.69 27 If xj4 < 724 Then node 40 Elseif xj4 >= 724 Then node 41 Else 1157.97 28 oj = 1258.78 29 oj = 1164.33 30 oj = 872.333 31 oj = 1023.29 32 If xj5 < 421.234 Then node 42 Elseif xj5 >= 421.234 Then node 43 Else 1029.89 33 oj = 1104.4 34 oj = 1506.5 35 If xj2 < 1370.5 Then node 44 Elseif xj2 >= 1370.5 Then node 45 Else 1550.55 36 If xj6 < 0.929461 Then node 46 Elseif xj6 >= 0.929461 Then node 47 Else 1293.46 37 If xj1 < 23.5 Then node 48 Elseif xj1 >= 23.5 Then node 49 Else 1348.67 38 oj = 1143.83 39 oj = 1043.29 40 If xj3 < 167.5 Then node 50 Elseif xj3 >= 167.5 Then node 51 Else 1170.36 41 oj = 1123.56 42 If xj6 < 0.888512 Then node 52 Elseif xj6 >= 0.888512 Then node 53 Else 1046 43 oj = 984.8 44 If xj4 < 684.5 Then node 54 Elseif xj4 >= 684.5 Then node 55 Else 1533.58 45 oj = 1576 46 oj = 1312.62 47 oj = 1262.8 48 oj = 1376.29 49 oj = 1310 50 If xj5 < 265.959 Then node 56 Elseif xj5 >= 265.959 Then node 57 Else 1188.14 51 If xj6 < 0.889694 Then node 58 Elseif xj6 >= 0.889694 Then node 59 Else 1147.73 52 oj = 1031.56 53 oj = 1072 54 oj = 1553.67 55 oj = 1513.5 56 oj = 1138.8 57 oj = 1215.56 58 oj = 1170.67 59 oj = 1120.2

trying various combinations of attributes in subsequent decision trees (the concept of RF), which results in the eXtreme gradient boosting (XGBoost) method [46].

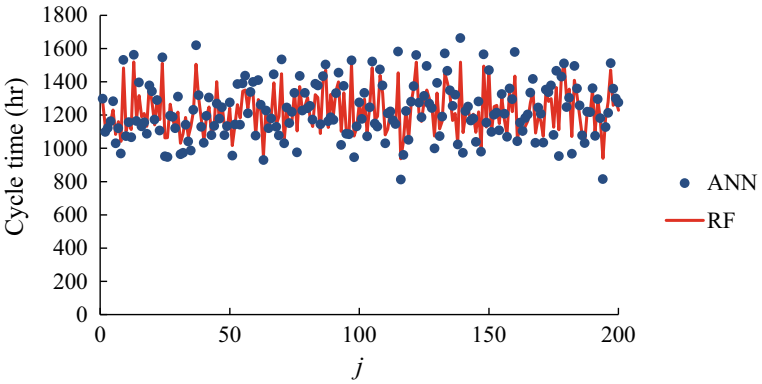


Fig. 2.22 Comparison of the ANN output and the approximated output using RF

Table 2.10 Rules contents

Decision tree #	Rule contents
1	If xj3 < 26 And xj2 < 1080 And xj4 < 43.3 Then oj = 2050
2	If xj1 < 22.5 And xj2 < 864 And xj4 < 36.7 Then ej1 = 36
3	If 19.5 < xj1 And xj3 < 22 Then ej2 = − 10

2.3.10 Random Forest-Based Incremental Interpretation

As mentioned previously, a RF is composed of a number of trees. From each tree, a decision rule can be applied to predict the cycle time of a job. As a result, multiple rules are simultaneously applicable, which may cause confusion for the user. To address this issue, Chen and Wang [47] proposed a variant of the RF method, RF-based incremental interpretation, to re-organize the decision rules derived using RF as follows:

Step 1. Average the prediction results by decision rules as $\hat{o}_j$.
Step 2. Set $q = 1$.
Step 3. Choose the decision rule with the closest prediction result to $\hat{o}_j$.
Step 4. Set the q-th incremental rule to the decision rule.
Step 5. Set $\hat{o}_j(current)$ to the prediction result of the decision rule.
Step 6. Remove the decision rule.
Step 7. Increase q by 1.
Step 8. If $q > T$, go to Step 14; otherwise, go to Step 9.
Step 9. Find the decision rule with the closest prediction result to $\hat{o}_j$.
Step 10. Subtract $\hat{o}_j(current)$ from the prediction result of the decision rule, and divide the result by T: The result is indicated by $\Delta\hat{o}_j(current)$.
Step 11. Set the premise of the q-th incremental rule to that of the decision rule.

Table 2.11 Rule contents

Decision tree #	Rule contents
1	If xj4 ≥ 651.5 Then oj = 1425.2
2	If xj2 ≥ 1322 Then oj = 1453.0
3	If xj2 ≥ 1316 Then oj = 1446.2

Step 12. Set the consequence of the q-th incremental rule to "add $\Delta\hat{o}_j(current)$ to $\hat{o}_j$" or "subtract $-\Delta\hat{o}_j(current)$ from $\hat{o}_j$".
Step 13. Return to Step 6.
Step 14. End.

Example 2.9 A RF is constructed to approximate the output of the ANN. Taking job j as an example. The decision rules applicable to this job are summarized in Table 2.11 [47]. The aggregation process is detailed in Table 2.12. Finally, the incremental rules are established as

"If $x_{j2} \geq 1316$ Then $\hat{o}_j = 1446.2$"
"If $x_{j2} \geq 1322$ Then add 2.3 to $\hat{o}_j$"
"If $x_{j4} \geq 651.5$ Then subtract 7.0 from $\hat{o}_j$"

The first incremental rule tells the user that the predicted value is around 1446.2 h. This rule is the most important rule. The user can stop reading if he/she doesn't want to continue reading. The user reads the rest of the rules from top to bottom, revising the predicted value successively, and finally learns that the predicted value is 1441.5 h.

2.3.11 Polynomial-Based Decision Tree (PBDT)

Nonlinear polynomial rules can also be applied to explain the operations in an ANN, e.g., PBDT. The procedure of the PBDT method includes the following steps:

Step 1. Use polynomials to approximate the transition/activation function in the ANN.
Step 2. Represent operations in the ANN as polynomial decision rules.
Step 3. Filter the polynomial-based decision rules.
Step 4. Propagate the polynomial decision rules through the ANN.
Step 5. Generate the network output as a polynomial decision rule.
Step 6. Extend polynomial decision rules if necessary.
Step 7. Evaluate the approximate accuracy using the PBDT method.

A flowchart is provided in Fig. 2.23 to illustrate the process.

The hyperbolic tangent sigmoid function is a popular transformation/activation function used in newer versions of some optimization packages such as MATLAB [48]:

Table 2.12 Aggregation process

Step #	Result
1	$\hat{o}_j = 1441.5$
2	$q = 1$
3	Rule #3 is chosen
4	Incremental rule #1 is "If $x_{j2} \geq 1316$ Then $\hat{o}_j = 1446.2$"
5	$\hat{o}_j(current) = 1446.2$
6	Rule #3 is removed
7	$q = 2$
8	$q \not> 3$; go to Step 9
9	Rule #2 is chosen
10	$\Delta\hat{o}_j(current) = 2.3$
11	The premise of incremental rule #2 is "$x_{j2} \geq 1322$"
12	The consequence of incremental rule #2 is "add 2.3 to $\hat{o}_j$"
13	Return to Step 6
6	Rule #2 is removed
7	$q = 3$
8	$q \not> 3$; go to Step 9
9	Rule #1 is chosen
10	$\Delta\hat{o}_j(current) = -7$
11	The premise of incremental rule #3 is "$x_{j4} \geq 651.5$"
12	The consequence of incremental rule #3 is "subtract 7 from $\hat{o}_j$"
13	Return to Step 5
6	Rule #1 is removed
7	$q = 4$
8	$q > 3$; go to Step 14
14	End

$$f(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{2.23}$$

as illustrated in Fig. 2.24.

It is difficult to accurately approximate the hyperbolic tangent sigmoid function with a single polynomial function. Therefore, the hyperbolic tangent sigmoid function is divided into four sections I–IV, and each section is approximated by a polynomial function [49, 50], as illustrated in Fig. 2.25.

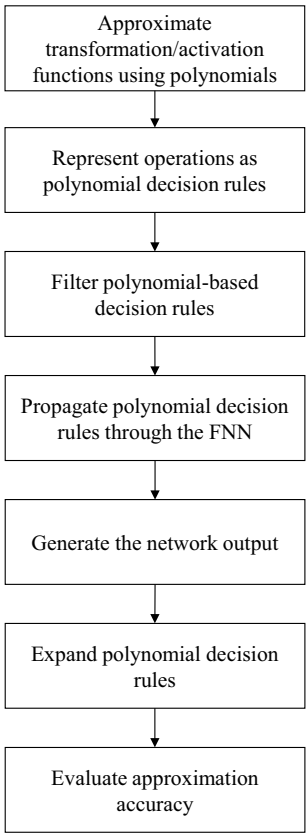


Fig. 2.23 Procedure of the PBDT method

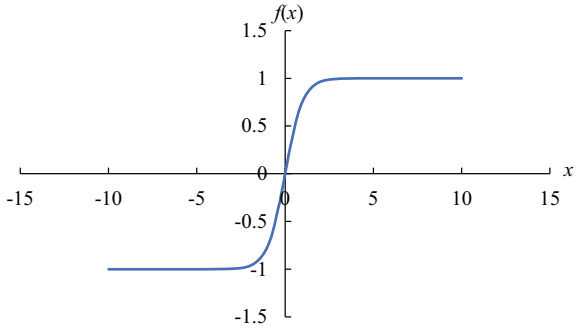


Fig. 2.24 Hyperbolic tangent sigmoid function

First, $-1 \leq f(x) < -0.99$ when $x < -2.64$. Therefore, by approximating $f(x)$ with $\hat{f}(x) = -1$, the absolute deviation $\left|\hat{f}(x) - f(x)\right|$ is less than 0.01 $\forall$ $x < -2.64$. Subsequently, for section II, $f(x)$ is fitted with a cubic polynomial $\hat{f}(x) = ax^3 + bx^2 + cx + d$, so that the sum of squared error (SSE) is minimized:

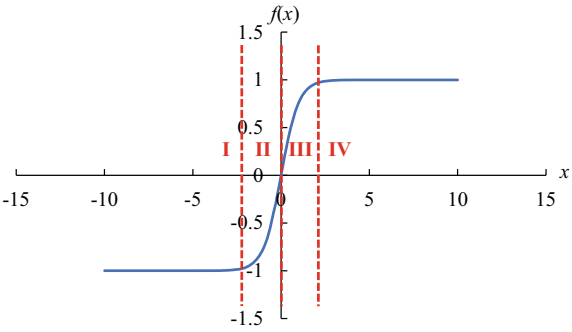


Fig. 2.25 Dividing the hyperbolic tangent sigmoid function into four sections

$$\begin{aligned}\text{SSE} &= \sum_{x=-2.64}^{0} \left(\hat{f}(x) - f(x)\right)^2 \\ &= \sum_{x=-2.64}^{0} \left(ax^3 + bx^2 + cx + d - \frac{2}{1+\mathrm{e}^{-2x}} - 1\right)^2 \end{aligned} \tag{2.24}$$

Taking the partial derivative of SSE with respect to each coefficient of $\hat{f}(x)$ and setting the result to zero gives

$$\begin{aligned}\frac{\partial \text{SSE}}{\partial a} &= \sum_{x=-2.64}^{0} 2\left(ax^3 + bx^2 + cx + d - \frac{2}{1+\mathrm{e}^{-2x}} - 1\right)x^3 \\ &= \sum_{x=-2.64}^{0} \left(2ax^6 + 2bx^5 + 2cx^4 + 2dx^3 - \frac{4x^3}{1+\mathrm{e}^{-2x}} - 2x^3\right) \\ &= 0 \end{aligned} \tag{2.25}$$

$$\begin{aligned}\frac{\partial \text{SSE}}{\partial b} &= \sum_{x=-2.64}^{0} 2\left(ax^3 + bx^2 + cx + d - \frac{2}{1+\mathrm{e}^{-2x}} - 1\right)x^2 \\ &= \sum_{x=-2.64}^{0} \left(2ax^5 + 2bx^4 + 2cx^3 + 2dx^2 - \frac{4x^2}{1+\mathrm{e}^{-2x}} - 2x^2\right) \\ &= 0 \end{aligned} \tag{2.26}$$

$$\begin{aligned}\frac{\partial \text{SSE}}{\partial c} &= \sum_{x=-2.64}^{0} 2\left(ax^3 + bx^2 + cx + d - \frac{2}{1+\mathrm{e}^{-2x}} - 1\right)x \\ &= \sum_{x=-2.64}^{0} \left(2ax^4 + 2bx^3 + 2cx^2 + 2dx - \frac{4x}{1+\mathrm{e}^{-2x}} - 2x\right) \\ &= 0 \end{aligned} \tag{2.27}$$

$$\begin{aligned}\frac{\partial \text{SSE}}{\partial d} &= \sum_{x=-2.64}^{0} 2\left(ax^3 + bx^2 + cx + d - \frac{2}{1+\mathrm{e}^{-2x}} - 1\right) \\ &= \sum_{x=-2.64}^{0} \left(2ax^3 + 2bx^2 + 2cx + 2d - \frac{4}{1+\mathrm{e}^{-2x}} - 2\right) \\ &= 0 \end{aligned} \tag{2.28}$$

Equations (2.25)–(2.28) are linear equations of variables a, b, c, and d, which can be easily solved to get

$$\hat{f}(x) = 0.0663x^3 + 0.4788x^2 + 1.1823x + 0.0133 \;\; \forall \;\; -2.64 \leq x < 0 \tag{2.29}$$

Similarly, the cubic polynomial for section III can be derived as

$$\hat{f}(x) = 0.0663x^3 + 0.4788x^2 + 1.1823x + 0.0133 \;\; \forall \;\; 0 < x \leq 2.64 \tag{2.30}$$

Finally, $\hat{f}(x) = 1$ when $x > 2.64$. The results are summarized by the following theorem.

Theorem 2.1

$$\hat{f}(x) = \begin{cases} \hat{f}_1(x) = (x) - 1 & \text{if } x < -2.64 \\ \hat{f}_2(x) = 0.0663x^3 + 0.4788x^2 + 1.1823x + 0.0133 & \text{if } -2.64 \leq x < 0 \\ \hat{f}_3(x) = 0.0663x^3 - 0.4788x^2 + 1.1823x - 0.0133 & \text{if } 0 \leq x < 2.64 \\ \hat{f}_4(x) = 1 & \text{if } x \geq 2.64 \end{cases} \tag{2.31}$$

The approximation result is illustrated in Fig. 2.26.

Substituting (2.4) and (2.5) into (2.6) gives

$$h_{jl} = f\left(\sum_{p=1}^{P} (w_{pl}^{h*} x_{jp}) - \theta_l^{h*}\right) \tag{2.32}$$

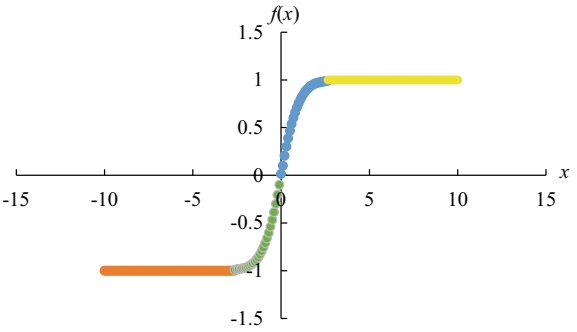


Fig. 2.26 Approximation result

Letting $c_1 = 0.0663$; $c_2 = 0.4788$; $c_3 = 1.1823$; $c_4 = 0.0133$. According to Theorem 2.1,

$$h_{jl} = \begin{cases} -1 & \text{if } \sum_{p=1}^{P} (w_{pl}^{h*} x_{jp}) - \theta_l^{h*} < -2.64 \\ \hat{f}_2\left(\sum_{p=1}^{P} (w_{pl}^{h*} x_{jp}) - \theta_l^{h*}\right) & \text{if } -2.64 \le \sum_{p=1}^{P} (w_{pl}^{h*} x_{jp}) - \theta_l^{h*} < 0 \\ \hat{f}_3\left(\sum_{p=1}^{P} (w_{pl}^{h*} x_{jp}) - \theta_l^{h*}\right) & \text{if } 0 \le \sum_{p=1}^{P} (w_{pl}^{h*} x_{jp}) - \theta_l^{h*} < 2.64 \\ 1 & \text{if } \sum_{p=1}^{P} (w_{pl}^{h*} x_{jp}) - \theta_l^{h*} \ge 2.64 \end{cases} \tag{2.33}$$

Substituting (2.33) into (2.7) gives

$$\hat{I}_j^o = \sum_{l=1}^{L} \begin{pmatrix} -w_l^{o*} & \text{if } \sum_{p=1}^{P} (w_{pl}^{h*} x_{jp}) - \theta_l^{h*} < -2.64 \\ w_l^{o*} \hat{f}_2\left(\sum_{p=1}^{P} (w_{pl}^{h*} x_{jp}) - \theta_l^{h*}\right) & \text{if } -2.64 \le \sum_{p=1}^{P} (w_{pl}^{h*} x_{jp}) - \theta_l^{h*} < 0 \\ w_l^{o*} \hat{f}_3\left(\sum_{p=1}^{P} (w_{pl}^{h*} x_{jp}) - \theta_l^{h*}\right) & \text{if } 0 \le \sum_{p=1}^{P} (w_{pl}^{h*} x_{jp}) - \theta_l^{h*} < 2.64 \\ w_l^{o*} & \text{if } \sum_{p=1}^{P} (w_{pl}^{h*} x_{jp}) - \theta_l^{h*} \ge 2.64 \end{pmatrix} \tag{2.34}$$

Finally, the network output can be approximated as

$$\hat{o}_j = \sum_{l=1}^{L} \begin{pmatrix} -w_l^{o*} & \text{if } \sum_{p=1}^{P} (w_{pl}^{h*} x_{jp}) - \theta_l^{h*} < -2.64 \\ w_l^{o*} \hat{f}_2\left(\sum_{p=1}^{P} (w_{pl}^{h*} x_{jp}) - \theta_l^{h*}\right) & \text{if } -2.64 \le \sum_{p=1}^{P} (w_{pl}^{h*} x_{jp}) - \theta_l^{h*} < 0 \\ w_l^{o*} \hat{f}_3\left(\sum_{p=1}^{P} (w_{pl}^{h*} x_{jp}) - \theta_l^{h*}\right) & \text{if } 0 \le \sum_{p=1}^{P} (w_{pl}^{h*} x_{jp}) - \theta_l^{h*} < 2.64 \\ w_l^{o*} & \text{if } \sum_{p=1}^{P} (w_{pl}^{h*} x_{jp}) - \theta_l^{h*} \ge 2.64 \end{pmatrix} - \theta^{o*} \tag{2.35}$$

Theorem 2.2

$$\hat{o}_j = \sum_{l=1}^{L} \begin{cases} -w_l^{o^*} & \text{if } \sum_{p=1}^{P} (w_{pl}^{h*} x_{jp}) - \theta_l^{h*} < -2.64 \\ c_1 w_l^{o*} \left(\sum_{p=1}^{P} (w_{pl}^{h*} x_{jp}) \right)^3 - 3c_1 w_l^{o*} \left(\sum_{p=1}^{P} (w_{pl}^{h*} x_{jp}) \right)^2 \theta_l^{h*} & \\ +3c_1 w_l^{o*} \sum_{p=1}^{P} (w_{pl}^{h*} x_{jp})(\theta_l^{h*})^2 - c_1 w_l^{o*} (\theta_l^{h*})^3 & \\ +c_2 w_l^{o*} \left(\sum_{p=1}^{P} (w_{pl}^{h*} x_{jp}) \right)^2 - 2c_2 w_l^{o*} \sum_{p=1}^{P} (w_{pl}^{h*} x_{jp}) \theta_l^{h*} & \text{if } -2.64 \le \sum_{p=1}^{P} (w_{pl}^{h*} x_{jp}) - \theta_l^{h*} < 0 \\ +c_2 w_l^{o*} (\theta_l^{h*})^2 + c_3 w_l^{o*} \sum_{p=1}^{P} (w_{pl}^{h*} x_{jp}) & \\ -c_3 w_l^{o*} \theta_l^{h*} + c_4 w_l^{o*} & \\ c_1 w_l^{o*} \left(\sum_{p=1}^{P} (w_{pl}^{h*} x_{jp}) \right)^3 - 3c_1 w_l^{o*} \left(\sum_{p=1}^{P} (w_{pl}^{h*} x_{jp}) \right)^2 \theta_l^{h*} & \\ +3c_1 w_l^{o*} \sum_{p=1}^{P} (w_{pl}^{h*} x_{jp})(\theta_l^{h*})^2 - c_1 w_l^{o*} (\theta_l^{h*})^3 & \\ -c_2 w_l^{o*} \left(\sum_{p=1}^{P} (w_{pl}^{h*} x_{jp}) \right)^2 + 2c_2 w_l^{o*} \sum_{p=1}^{P} (w_{pl}^{h*} x_{jp}) \theta_l^{h*} & \text{if } 0 \le \sum_{p=1}^{P} (w_{pl}^{h*} x_{jp}) - \theta_l^{h*} < 2.64 \\ -c_2 w_l^{o*} (\theta_l^{h*})^2 + c_3 w_l^{o*} \sum_{p=1}^{P} (w_{pl}^{h*} x_{jp}) & \\ -c_3 w_l^{o*} \theta_l^{h*} - c_4 w_l^{o*} & \\ w_l^{o*} & \text{if } \sum_{p=1}^{P} (w_{pl}^{h*} x_{jp}) - \theta_l^{h*} \ge 2.64 \end{cases} - \theta^{o*} \tag{2.36}$$

Proof Theorem 2.2 is the expansion of Eq. (2.35). It is noteworthy that Theorem 2.2 involves only polynomials.

2.3.12 Comparison of Various XAI Techniques

Decision tree-based XAI techniques for approximating an ANN are compared in Table 2.13.

Table 2.13 Comparison of various XAI techniques

	CART	RF	Gradient boosted decision trees	RF-based incremental interpretation	PBDT
Number of decision rules for an example	1	Many	Many	Many	1
Each decision rule directly approximates the ANN output	Yes	Yes	No	No	Yes
First decision rule is the most accurate	Yes	No	No	Yes	Yes

References

1. Y.-C. Wang, H.-R. Tsai, T. Chen, A selectively fuzzified back propagation network approach for precisely estimating the cycle time range in wafer fabrication. Mathematics **9**, 1430 (2021)
2. Q. Xu, V. Sharma, Ensemble sales forecasting study in semiconductor industry, in *Industrial Conference on Data Mining* (2017), pp. 31–44
3. T. Chen, H.-C. Wu, Forecasting the unit cost of a DRAM product using a layered partial-consensus fuzzy collaborative forecasting approach. Complex Intell. Syst. **6**, 497 (2020)
4. T.-C. T. Chen, Y.-C. Wang, AI applications to kaizen management, in *Artificial Intelligence and Lean Manufacturing*, pp. 37–52
5. T. Chen, Y.-C. Wang, Interval fuzzy number-based approach for modelling an uncertain fuzzy yield learning process. J. Ambient. Intell. Humaniz. Comput. **11**, 1213–1223 (2020)
6. J. Wang, J. Zhang, Big data analytics for forecasting cycle time in semiconductor wafer fabrication system. Int. J. Prod. Res. **54**(23), 7231–7244 (2016)
7. B.F. van Dongen, R.A. Crooy, W.M. van der Aalst, Cycle time prediction: when will this case finally be finished? in *OTM Confederated International Conferences on the Move to Meaningful Internet Systems* (2008), pp. 319–336
8. B.E. Ankenman, J.M. Bekki, J. Fowler, G.T. Mackulak, B.L. Nelson, F. Yang, Simulation in production planning: an overview with emphasis on recent developments in cycle time estimation, in *Planning Production and Inventories in the Extended Enterprise* (2011), pp. 565–591
9. T. Chen, Evaluating the mid-term competitiveness of a product in a semiconductor fabrication factory with a systematic procedure. Comput. Ind. Eng. **53**, 499–513 (2007)
10. T. Chen, Y.C. Wang, H.R. Tsai, Lot cycle time prediction in a ramping-up semiconductor manufacturing factory with a SOM–FBPN-ensemble approach with multiple buckets and partial normalization. Int. J. Adv. Manufact. Technol. **42**(11), 1206–1216 (2009)
11. F. Yang, B. Ankenman, B.L. Nelson, Efficient generation of cycle time-throughput curves through simulation and metamodeling. Nav. Res. Logist. **54**(1), 78–93 (2007)
12. T. Chen, Y.-C. Wang, A bi-criteria nonlinear fluctuation smoothing rule incorporating the SOM-FBPN remaining cycle time estimator for scheduling a wafer fab—a simulation study. Int. J. Adv. Manuf. Technol. **49**(5), 709–721 (2010)
13. C. Chiu, P.C. Chang, N.H. Chiu, A case-based expert support system for due-date assignment in a wafer fabrication factory. J. Intell. Manuf. **14**(3), 287–296 (2003)
14. B.R. Cobb, L. Li, Forward cycle time distributions for returnable transport items. J. Remanufact. **12**(1), 125–151 (2022)
15. T. Chen, An effective fuzzy collaborative forecasting approach for predicting the job cycle time in wafer fabrication. Comput. Ind. Eng. **66**(4), 834–848 (2013)
16. Y.C. Wang, T. Chen, T.C. Hsu, A fuzzy deep predictive analytics approach for enhancing cycle time range estimation precision in wafer fabrication. Decis. Anal. J. **1**, 100010 (2021)
17. T. Chen, H.C. Wu, A new cloud computing method for establishing asymmetric cycle time intervals in a wafer fabrication factory. J. Intell. Manuf. **28**(5), 1095–1107 (2017)
18. T. Chen, Y.-C. Wang, Hybrid big data analytics and Industry 4.0 approach for projecting cycle time ranges. Int. J. Adv. Manufact. Technol. **120**, 279–295 (2022)
19. T.C.T. Chen, Y.C. Wang, Fuzzy dynamic-prioritization agent-based system for forecasting job cycle time in a wafer fabrication plant. Complex Intell. Syst. **7**(4), 2141–2154 (2021)
20. T. Chen, Y.C. Wang, A nonlinearly normalized back propagation network and cloud computing approach for determining cycle time allowance during wafer fabrication. Robot. Comput. Integr. Manufact. **45**, 144–156 (2017)
21. D. Gunning, M. Stefik, J. Choi, T. Miller, S. Stumpf, G.Z. Yang, XAI—Explainable artificial intelligence. Sci. Robot. **4**(37), eaay7120 (2019)
22. D. Kumar, A. Wong, G.W. Taylor, Explaining the unexplained: a class-enhanced attentive response (clear) approach to understanding deep neural networks, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (2017), pp. 36–44

23. S. Plattner, D.M. Mason, G.A. Leshkevich, D.J. Schwab, E.S. Rutherford, Classifying and forecasting coastal upwellings in Lake Michigan using satellite derived temperature images and buoy data. J. Great Lakes Res. **32**(1), 63–76 (2006)
24. T. Chen, Y.-C. Lin, Enhancing the accuracy and precision of forecasting the productivity of a factory: a fuzzified feedforward neural network approach. Complex Intell. Syst. **7**, 2317–2327 (2021)
25. ConvNetJS, ConvnetJS demo: Toy 2d classification with 2-layer neural network (2022). https://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html
26. GitHub, tensorflow (2022). https://github.com/tensorflow
27. Z. Li, J. Cai, S. He, H. Zhao, Seq2seq dependency parsing, in *Proceedings of the 27th International Conference on Computational Linguistics* (2018), pp. 3203–3214
28. S. Mantri, K. Bapat, Neural network based face recognition using MATLAB. Int. J. Comput. Sci. Eng. Technol. **1**(1), 6–9 (2011)
29. K.P. Sudheer, A. Jain, Explaining the internal behaviour of artificial neural network river flow models. Hydrol. Process. **18**(4), 833–844 (2004)
30. T.C.T. Chen, *Production Planning and Control in Semiconductor Manufacturing: Big Data Analytics and Industry 4.0 Applications* (Springer Nature, 2022)
31. A. Ranganathan, The Levenberg-Marquardt algorithm. Tutorial on LM Algorithm **11**(1), 101–110 (2004)
32. P. Sedgwick, Pearson's correlation coefficient. Bmj 345 (2012)
33. P. Sedgwick, Spearman's rank correlation coefficient. Bmj 349 (2014)
34. M. Green, U. Ekelund, L. Edenbrandt, J. Björk, J.L. Forberg, M. Ohlsson, Exploring new possibilities for case-based explanation of artificial neural network ensembles. Neural Netw. **22**(1), 75–81 (2009)
35. MathWorks, oobPermutedPredictorImportance (2022). https://www.mathworks.com/help/stats/classificationbaggedensemble.oobpermutedpredictorimportance.html?searchHighlight=oobPermutedPredictorImportance&s_tid=srchtitle_oobPermutedPredictorImportance_1
36. T. Chen, A fuzzy back propagation network for output time prediction in a wafer fab. Appl. Soft Comput. **2**(3), 211–222 (2003)
37. K. Yan, D. Zhang, Feature selection and analysis on correlated gas sensor data with recursive feature elimination. Sens. Actuators, B Chem. **212**, 353–363 (2015)
38. C. Molnar, 9.5 Shapley values (2022). https://christophm.github.io/interpretable-ml-book/shapley.html#shapley
39. E.M. Kenny, M.T. Keane, Twin-systems to explain artificial neural networks using case-based reasoning: comparative tests of feature-weighting methods in ANN-CBR twins for XAI, in *Twenty-Eighth International Joint Conferences on Artificial Intelligence* (2019), pp. 2708–2715
40. A. Likas, N. Vlassis, J.J. Verbeek, The global k-means clustering algorithm. Pattern Recogn. **36**(2), 451–461 (2003)
41. S. Albawi, T.A. Mohammed, S. Al-Zawi, Understanding of a convolutional neural network, in *2017 International Conference on Engineering and Technology* (2017), pp. 1–6
42. T. Chen, A job-classifying and data-mining approach for estimating job cycle time in a wafer fabrication factory. Int. J. Adv. Manuf. Technol. **62**(1), 317–328 (2012)
43. W.Y. Loh, Classification and regression trees. Wiley Interdisc. Rev. Data Min. Knowl. Discovery **1**(1), 14–23 (2011)
44. J. Liu, Q. Huang, C. Ulishney, C.E. Dumitrescu, Comparison of random forest and neural network in modeling the performance and emissions of a natural gas spark ignition engine. J. Energy Res. Technol. **144**(3), 032310 (2022)
45. GoogleDevelopers, Gradient boosted decision trees | Machine learning (2022). https://developers.google.com/machine-learning/decision-forests/intro-to-gbdt
46. T. Chen, T. He, M. Benesty, V. Khotilovich, Y. Tang, H. Cho, K. Chen, Xgboost: extreme gradient boosting. R Package Version 0.4-2, **1**(4), 1–4 (2015)
47. T. Chen, Y.C. Wang, A two-stage explainable artificial intelligence approach for classification-based job cycle time prediction. Int. J. Adv. Manuf. Technol. **123**(5), 2031–2042 (2022)

48. MathWorks.com, Hyperbolic tangent sigmoid transfer function—MATLAB tansig (2022). https://www.mathworks.com/help/deeplearning/ref/tansig.html
49. T. Chen, Y.-C. Wang, Semiconductor yield forecasting using quadratic-programming based fuzzy collaborative intelligence approach. Math. Probl. Eng. **2013**, 672404 (2013)
50. T. Chen, M.C. Chiu, An improved fuzzy collaborative system for predicting the unit cost of a DRAM product. Int. J. Intell. Syst. **30**(6), 707–730 (2015)

Chapter 3
Applications of XAI for Decision Making in the Manufacturing Domain

Abstract This chapter discusses an important topic in factory management, that of improving the understandability of AI applications for group multi-criteria decision making in manufacturing systems. Due to its long-term and cross-functional impact, decision making may be more critical to the competitiveness and sustainability of manufacturing systems than production planning and control. This chapter uses the example of choosing the right smart and automation technologies for factories during the COVID-19 pandemic. This topic is of particular importance as many factories are forced to close or operate on a smaller scale (using a smaller workforce), thus pursuing further automation. Artificial intelligence and Industry 4.0 technologies have many applications in this area, most of which can also be applied for other decision-making purposes in manufacturing systems. First, a systematic procedure was established to guide the group multi-criteria decision-making process. Applications of AI and XAI to identify targets are first reviewed. Subsequently, the application of AI and XAI to selection factors and development of criteria is presented. Artificial intelligence techniques are widely used to derive criteria priorities. Therefore, it is particularly important to explain XAI techniques and tools for such AI applications. Aggregating the judgments of multiple decision makers is the next focus, followed by the introduction of AI and XAI applications to evaluate the overall performance of each alternative. Taking fuzzy ranking preference based on similarity to ideal solution (FTOPSIS) as an example, the application of XAI techniques and tools in explaining comparison results using FTOPSIS is illustrated. Another AI technology used for the same purpose is fuzzy VIKOR. XAI techniques and tools for interpreting fuzzy VIKOR are also presented. Finally, several metrics are proposed to evaluate the effectiveness of XAI techniques or tools for decision making in the manufacturing domain.

Keywords XAI · Group multi-criteria decision making · Smart and automation technologies · FTOPSIS · Fuzzy VIKOR

T.-C.T. Chen, *Explainable Artificial Intelligence (XAI) in Manufacturing*,
SpringerBriefs in Applied Sciences and Technology,
https://doi.org/10.1007/978-3-031-27961-4_3

3.1 Decision Making in the Manufacturing Domain

There are many decision-making processes in a manufacturing system. For example, equipment engineers need to choose the most suitable equipment among multiple options [1], production engineers need to determine the next job to be process on a machine [2], facilities engineers need to decide whether to perform periodic maintenance in advance to avoid unexpected machine down [3], etc. These decision-making processes are based on information that needs to be generated through in-depth analysis of relevant data and are subject to multiple criteria that may conflict with each other [4]. Artificial intelligence (AI) techniques have been widely used to address these issues and support decision making in manufacturing [5].

This chapter takes the selection of suitable smart and automation technologies for a factory during the COVID-19 pandemic as an example. This topic is of particular importance as many factories have been forced to close or operate on a small scale (using less workforce), hence pursuing further automation [6]. In addition, this is also a hot topic in the era of Industry 4.0 [7, 8]. There are many applications of AI technologies in this field, most of which can also be applied for other decision-making purposes in manufacturing systems.

3.1.1 Selection of Smart and Automation Technologies Within the COVID-19 Pandemic

Smart technologies are technologies that use electronic devices or systems that can be connected to the Internet, used interactively, and are to some extent intelligent [9–12], while automation technologies can make production smoother, faster, and more cost-effective by replacing or assisting manually operation [13]. In recent years, some advanced smart and automation technologies have been proposed in the domain of manufacturing, such as automatic inspection [14, 15], autonomous robots [16–18], additive manufacturing [19–23], ubiquitous manufacturing (UM) [20, 24, 25], cloud manufacturing [26–28], Internet of Things (IoT) [29, 30], cyber-physical systems [30, 31], etc., as shown in Fig. 3.1.

As shown in Table 3.1, some of these advanced smart and automation technologies aim to facilitate cooperation among factories by sharing manufacturing resources, which is difficult in the COVID-19 pandemic and may not help mitigate the impact, as shown in Table 3.1. In contrast, applying smart and automation technologies to assist workers is considered feasible [32–35].

For example, using voice commands or gestures to interact with machines could avoid spreading COVID-19 through touching machines [36, 37]. The same can be achieved by remotely controlling the machine using a smartphone [38]. In addition, since body temperature is one of the basic criteria for screening workers for possible infection, workers can wear smart bracelets or watches to detect body temperature [39]. Wearable sensors can also be used to measure the distance between workers

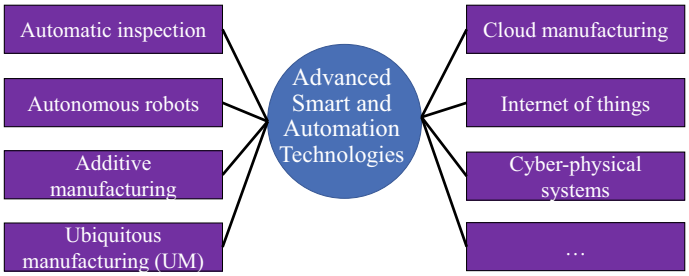


Fig. 3.1 Advanced smart and automation technologies

Table 3.1 Effects of existing smart and automation technologies on mitigating the impact of the COVID-19 pandemic

Smart and automation technology	Effect	Mechanism
Automatic inspection	Positive	Reducing workforce
Autonomous robots	Positive	Reducing workforce
Additive manufacturing	Positive	Reducing workforce
Ubiquitous manufacturing	Negative	Increasing the risk of cross-factory infection
Cloud manufacturing	Negative	Increasing the risk of cross-factory infection
Internet of Things	Positive	Reducing human–machine contact
Cyber-physical systems	Positive	Reducing human–machine contact

to ensure physical distancing [33]. However, such applications are time-consuming and laborious, and the effects are uncertain, which may not be acceptable to workers. In order to make full use of limited resources and time, it is necessary to establish a systematic procedure to compare various applications of smart and automation technologies.

In this chapter, the selection of suitable smart and automation technologies for a factory during the COVID-19 pandemic is modeled as a fuzzy group multi-criteria decision-making (MCDM) problem [40]. Multiple AI technologies are applied to help solve this problem, and appropriate explainable artificial intelligence (XAI) techniques and tools are applied to explain these AI technology applications.

3.2 Procedure for Selection of Smart and Automation Technologies Within the COVID-19 Pandemic

Selecting suitable smart and automation technologies for a factory during the COVID-19 pandemic is modeled as a fuzzy group MCDM problem, which is usually solved in seven steps [41] (Fig. 3.2):

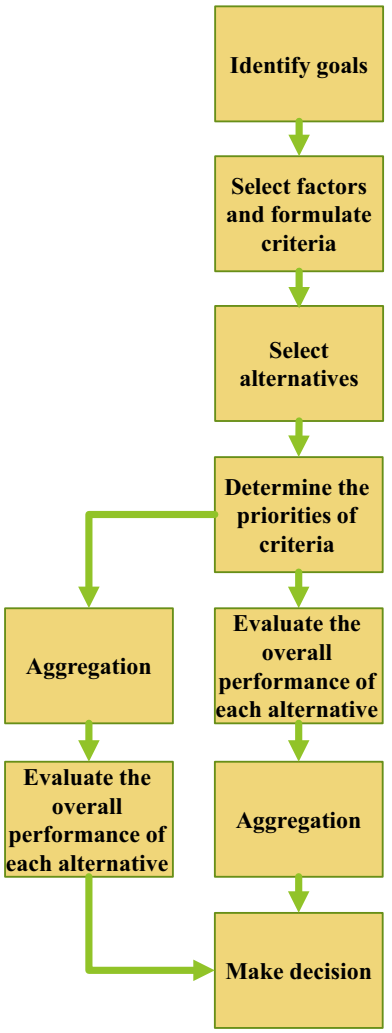


Fig. 3.2 Steps for solving a fuzzy group MCDM problem

- Identify goals;
- Select factors and formulate criteria;
- Select alternatives (options);
- Determine the priorities of criteria;
- Evaluate the overall performance of each alternative;
- Aggregation;
- Make decision.

The administrator can aggregate the fuzzy priorities of a criterion derived by all decision makers, or the overall performances of an alternative evaluated by them.

3.3 Identifying Goals

Selecting suitable smart and automation technologies for factories during the COVID-19 pandemic might have the following goals:

- Response to the shortage of manpower: Lockdowns and quarantines during the COVID-19 pandemic have reduced available manpower, and therefore present opportunities for factories to increase automation and remote service delivery [35]. Among existing advanced automation technologies, automated inspection, autonomous robotics, and additive manufacturing can mitigate the impact of the COVID-19 pandemic by reducing labor. As a result, more robots and automated systems are expected to be used earlier than planned [35, 42]. Canadian Plastics [32] describes this trend as "the enduring boom in factory robotics".
- Reduction of the risk of infection: Traditional automation focuses on low-level tasks. In the COVID pandemic, the targets of automation are shifting to tasks that are labor-intensive or difficult to maintain social distance [42]. Automated equipment such as computer numerical control (CNC) machine tools, controlled by minicomputers with interfaces such as keyboards and touchscreens, can easily spread COVID-19. To address this issue, such automated equipment can be operated by voice commands or gestures, or can be remotely controlled via an app on a smartphone [36–38]. The latter is a joint application of automation and smart technologies. IoT and cyber-physical systems can help prevent the spread of COVID-19 by reducing human–machine contact. Additionally, smart technologies can also be applied to improve operational efficiency and protect worker safety and health during the COVID-19 pandemic. For example, workers can wear smart wristbands or watches to check body temperature [39], while supervisors can wear smart helmets to monitor workers' body temperature [43]. In addition, wireless sensors can be worn or carried to measure the distance of workers to ensure physical distance or record their movements for tracking [33, 44]. If two wearable sensors get too close, they emit a warning signal. Contact times are also recorded. Indoor location technology can also be applied to screen workers who may have come into contact with infected workers [33].
- Cost-effectiveness: Cost-effectiveness is undoubtedly a critical performance measure to the management and operations of a manufacturing systems. However, whether automation technologies that have become popular in recent years, such as AI and cloud manufacturing, have brought benefits to factories during the COVID-19 pandemic has been questioned [32].

3.4 AI and XAI Applications in Selecting Factors and Formulating Criteria

After reviewing the relevant literature and practice, the following factors are considered to be critical for selecting suitable applications of smart and automation technologies amidst the COVID-19 pandemic:

- Low estimated total cost;
- Effective in preventing the spread of COVID-19;
- Little interference to existing operations;
- Ease of adoption; and
- High acceptance to workers.

Among the five factors, "effective in preventing the spread of COVID-19", "ease of adoption", and "high acceptance to workers" are the-higher-the-better performances, while others are the-lower-the-better performances. Therefore, the performances in optimizing the five aspects are evaluated according to the rules in Table 3.2, which can be visualized with an annotated line chart, as illustrated in Fig. 3.3, which is an improvement of a traditional line chart by applying three XAI techniques [45]:

- Color management: The color and shape of objects representing different concepts should be distinguished for easy comparison.
- Common expressions: Technical terms and variable names should be replaced with common expressions.
- Annotate figures: The legend for each object should be placed close to the object.

Similar performance evaluation rules have been proposed in previous studies [46, 47], and have the following advantages:

- The performances evaluated using the rules have the same ranges.
- These rules can be used to evaluate quantitative and qualitative performances.
- Fuzzy logic is applied to consider subjective and uncertain information, generate interval-valued evaluation results, thereby providing flexibility for decision making.

However, other forms of evaluation rules are possible.

3.5 AI and XAI Applications in Deriving the Priorities of Criteria

At first, each decision maker compares the relative priority of a factor to another in linguistic terms such as "as equal as", "weakly more important than", "strongly more important than", "very strongly more important than", and "absolutely more important than" [48]. These linguistic terms are usually mapped to triangular fuzzy numbers (TFNs) within [1, 9, 49]. For example, compared to factor #1 (baseline),

Table 3.2 Rules for evaluating performances

Factor	Rule
Low estimated total cost	$\tilde{p}_{l1}(x_l) = \begin{cases} (0,0,1) & \text{if } 0.1 \cdot \min\limits_r x_r + 0.9 \cdot \max\limits_r x_r \le x_l \text{ or data not available} \\ (0,1,2) & \text{if } 0.35 \cdot \min\limits_r x_r + 0.65 \cdot \max\limits_r x_r \le x_l < 0.1 \cdot \min\limits_r x_r + 0.9 \cdot \max\limits_r x_r \\ (1.5,2.5,3.5) & \text{if } 0.65 \cdot \min\limits_r x_r + 0.35 \cdot \max\limits_r x_r \le x_l < 0.35 \cdot \min\limits_r x_r + 0.65 \cdot \max\limits_r x_r \\ (3,4,5) & \text{if } 0.9 \cdot \min\limits_r x_r + 0.1 \cdot \max\limits_r x_r \le x_l < 0.65 \cdot \min\limits_r x_r + 0.35 \cdot \max\limits_r x_r \\ (4,5,5) & \text{if } x_l < 0.9 \cdot \min\limits_r x_r + 0.1 \cdot \max\limits_r x_r \end{cases}$ where x_l is the estimated total cost of alternative #l
Effective in preventing the spread of COVID-19	$\tilde{p}_{l2}(x_l) = \begin{cases} (0,0,1) & \text{if } x_l = \text{ineffective} \\ (0,1,2) & \text{if } x_l = \text{slightly ineffective} \\ (1.5,2.5,3.5) & \text{if } x_l = \text{quite effective} \\ (3,4,5) & \text{if } x_l = \text{highly effective} \\ (4,5,5) & \text{if } x_l = \text{very highly effective} \end{cases}$ where x_l is the effectiveness of alternative #l in preventing the spread of COVID-19

(continued)

Table 3.2 (continued)

Factor	Rule
Little interference to existing operations	$\tilde{p}_{l3}(x_l) = \begin{cases} (4,5,5) & \text{if } x_l = \text{very little} \\ (3,4,5) & \text{if } x_l = \text{little} \\ (1.5,2.5,3.5) & \text{if } x_l = \text{moderate} \\ (0,1,2) & \text{if } x_l = \text{much} \\ (0,0,1) & \text{if } x_l = \text{very much} \end{cases}$ where x_l is the interference of alternative #l to existing operations
Ease of adoption	$\tilde{p}_{l4}(x_l) = \begin{cases} (0,0,1) & \text{if } x_l = \text{very difficult} \\ (0,1,2) & \text{if } x_l = \text{difficult} \\ (1.5,2.5,3.5) & \text{if } x_l = \text{moderate} \\ (3,4,5) & \text{if } x_l = \text{easy} \\ (4,5,5) & \text{if } x_l = \text{very easy} \end{cases}$ where x_l is the easiness of adoption of alternative #l
High acceptability to workers	$\tilde{p}_{l5}(x_l) = \begin{cases} (0,0,1) & \text{if } x_l = \text{very low} \\ (0,1,2) & \text{if } x_l = \text{low} \\ (1.5,2.5,3.5) & \text{if } x_l = \text{moderate} \\ (3,4,5) & \text{if } x_l = \text{high} \\ (4,5,5) & \text{if } x_l = \text{very high} \end{cases}$ where x_l is workers' acceptability of alternative #l

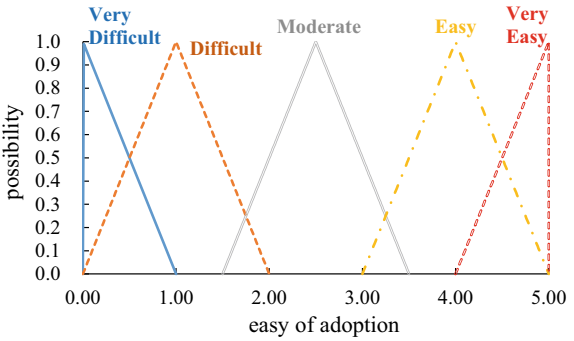


Fig. 3.3 Annotated line chart for visualizing the criteria

"factor #2 is weakly more important than factor #1" and "factor #1 is weakly more important than factor #3" can be visualized with a XAI tool, the gradient bar chart, as shown in Fig. 3.4, where the densities of colors reflect the different degrees of membership (see Fig. 3.5).

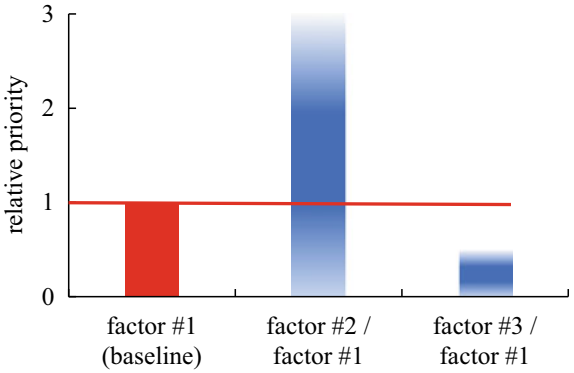


Fig. 3.4 Gradient bar chart for visualizing fuzzy pairwise comparison results

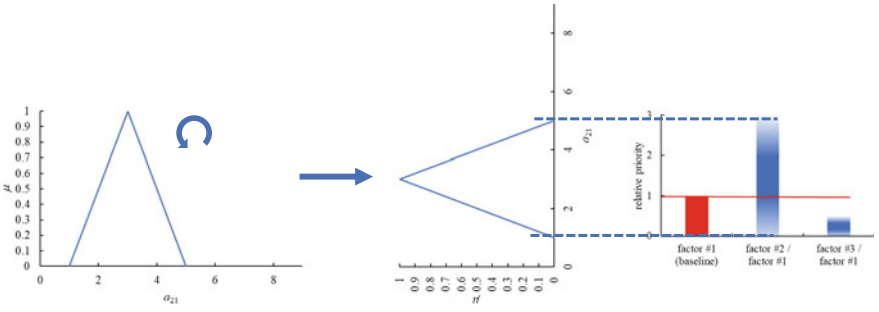


Fig. 3.5 Densities of colors reflect the different degrees of membership

According to the pairwise comparison results by decision maker #k, the fuzzy judgment (or pairwise comparison) matrix $\tilde{\mathbf{A}}(k) = [\tilde{a}_{ij}(k)]$ is constructed; $k = 1$–K. The fuzzy eigenvalue and eigenvector of $\tilde{\mathbf{A}}(k)$ are denoted by $\tilde{\lambda}(k)$ and $\tilde{\mathbf{x}}(k)$ respectively, satisfying [50]

$$\det\left(\tilde{\mathbf{A}}(k)(-)\tilde{\lambda}(k)\mathbf{I}\right) = 0 \tag{3.1}$$

and

$$\left(\tilde{\mathbf{A}}(k)(-)\tilde{\lambda}(k)\mathbf{I}\right)(\times)\tilde{\mathbf{x}}(k) = 0 \tag{3.2}$$

where $(-)$ and $(\times)$ denote fuzzy subtraction and multiplication, respectively. The fuzzy priorities of criteria can be derived by normalizing $\tilde{\mathbf{x}}(k)$:

$$\tilde{w}_i(k) = \frac{\tilde{x}_i(k)}{\sum_j \tilde{x}_j(k)} \tag{3.3}$$

The consistency between pairwise comparison results can be evaluated by fuzzy consistency ratio:

$$\widetilde{\mathrm{CR}}(k) = \frac{\frac{\tilde{\lambda}_{\max}(k)-n}{n-1}}{\mathrm{RI}} \tag{3.4}$$

where $\tilde{\lambda}_{\max}(k)$ is the fuzzy maximal eigenvalue; RI is the random consistency index [50]. $\widetilde{\mathrm{CR}}(k)$ should be less than 0.1–0.3, depending on the problem size. Subsequently, alpha-cut operations (ACO) [51] are applied to derive the values of $\tilde{\lambda}(k)$ and $\tilde{\mathbf{x}}(k)$ as follows below.

First, the fuzzy parameters and variables in Eqs. (3.1) and (3.2) are replaced by their α cuts:

$$\det(\tilde{\mathbf{A}}(\alpha)(k) - \tilde{\lambda}(\alpha)(k)\mathbf{I}) = 0 \tag{3.5}$$

$$(\tilde{\mathbf{A}}(\alpha)(k) - \tilde{\lambda}(\alpha)(k)\mathbf{I})\tilde{\mathbf{x}}(\alpha)(k) = 0 \tag{3.6}$$

If α takes 11 possible values (0, 0.1, … 1), Eqs. (3.5) and (3.6) must be solved $11 \cdot 2^{C_2^n}$ times using an eigen analysis to derive the α cuts of the fuzzy maximal eigenvalue and fuzzy eigenvector as [52]

$$\lambda^L(\alpha)(k) = \min_{\det([a_{ij}^*(\alpha)(k)]-\lambda_t(\alpha)(k)I)=0} (\lambda_t(\alpha)(k)) \tag{3.7}$$

$$\lambda^R(\alpha)(k) = \max_{\det([a_{ij}^*(\alpha)(k)]-\lambda_t(\alpha)(k)I)=0} (\lambda_t(\alpha)(k)) \tag{3.8}$$

$$\mathbf{x}^L(\alpha)(k) = \min_{([a_{ij}^*(\alpha)(k)]-\lambda_t(\alpha)(k)\mathbf{I})\mathbf{x}_t(\alpha)(k)=0} (\mathbf{x}_t(\alpha)(k)) \tag{3.9}$$

$$\mathbf{x}^R(\alpha)(k) = \max_{([a_{ij}^*(\alpha)(k)]-\lambda_t(\alpha)(k)\mathbf{I})\mathbf{x}_t(\alpha)(k)=0} (\mathbf{x}_t(\alpha)(k)) \tag{3.10}$$

where $* = L$ or R. $\lambda_t^L(\alpha)(k)$, $\lambda_t^R(\alpha)(k)$, $\mathbf{x}_t^L(\alpha)(k)$, and $\mathbf{x}_t^R(\alpha)(k)$ are the results from the t-th combination; $t = 1\text{–}11 \cdot 2^{C_2^n}$.

Example 3.1 Fuzzy judgment matrix of decision maker #k is as follows:

$$\tilde{\mathbf{A}}(k) = \begin{bmatrix} 1 & 1/(2,\ 4,\ 6) & (3,\ 5,\ 7) & (2,\ 4,\ 6) & (1,\ 3,\ 5) \\ (2,\ 4,\ 6) & 1 & (3,\ 5,\ 7) & (3,\ 5,\ 7) & (2,\ 4,\ 6) \\ 1/(3,\ 5,\ 7) & 1/(3,\ 5,\ 7) & 1 & 1/(2,\ 4,\ 6) & 1/(1,\ 2,\ 4) \\ 1/(2,\ 4,\ 6) & 1/(3,\ 5,\ 7) & (2,\ 4,\ 6) & 1 & 1/(1,\ 3,\ 5) \\ 1/(1,\ 3,\ 5) & 1/(2,\ 4,\ 6) & (1,\ 2,\ 4) & (1,\ 3,\ 5) & 1 \end{bmatrix}$$

MATLAB is used to implement ACO to derive the fuzzy maximum eigenvalue and fuzzy eigenvector of $\tilde{\mathbf{A}}(k)$. The required MATLAB code is shown in Fig. 3.6. The derived fuzzy maximum eigenvalue and fuzzy priorities are shown in Fig. 3.7 and Fig. 3.8, respectively.

A gradient bar chart can also be drawn to show the derived fuzzy priorities (Fig. 3.9).

3.6 AI and XAI Applications in Aggregation

There are two ways to aggregate the judgments of decision makers [40]. The first approach is to aggregate the pairwise comparison results of decision makers using fuzzy geometric mean (FGM):

$$\tilde{a}_{ij} = \sqrt[K]{\prod_{k=1}^{K} \tilde{a}_{ij}(k)} \tag{3.11}$$

The other approach is to use the fuzzy arithmetic mean (FAM) to aggregate the fuzzy priorities derived by decision makers:

$$\tilde{w}_i = \frac{\sum_{k=1}^{K} \tilde{w}_i(k)}{K} \tag{3.12}$$

Instead, if there is (overall) consensus among decision makers, fuzzy intersection (FI) can be applied to aggregate the priorities derived by them as follows [53].

```
stime=now       % start time
fl=zeros(22,1)      % fuzzy eigenvalue
fev=zeros(5,22)     % alpha cuts of fuzzy eigenvector; alpha = 0, 0.1, …, 1

for i0=1:11
   alpha0=i0*0.1-0.1        % alpha
   lmin=9999
   lmax=0
   evmin=[9999 9999 9999 9999 9999]
   evmax=[0 0 0 0 0]

   FA=zeros(5,5,3)      % fuzzy pairwise comparison matrix
   FA(2,1,:)=[2 4 6]
   FA(1,3,:)=[3 5 7]
   . . .
   FA(5,4,:)=[1 3 5]
   FA(:,:,1)=(1-alpha0)*FA(:,:,1)+alpha0*FA(:,:,2)      % calculate alpha cuts
   FA(:,:,3)=(1-alpha0)*FA(:,:,3)+alpha0*FA(:,:,2)

   A=zeros(5,5)      % crisp pairwise comparison matrix
   A(1,1)=1
   A(2,2)=1
   A(3,3)=1
   A(4,4)=1
   A(5,5)=1

   for i1=1:2
      for i2=1:2
         for i3=1:2
            for i4=1:2
               for i5=1:2
                  for i6=1:2
                     for i7=1:2
                        for i8=1:2
                           for i9=1:2
                              for i10=1:2

                                 % build crisp pairwise comparison matrix
                                 if FA(1,2,i1*2-1)==0
                                    A(1,2)=1/FA(2,1,i1*2-1)
                                 else
                                    A(1,2)=FA(1,2,i1*2-1)
                                 end
                                 A(2,1)=1/A(1,2)

                                 if FA(1,3,i2*2-1)==0
                                    A(1,3)=1/FA(3,1,i2*2-1)
                                 else
                                    A(1,3)=FA(1,3,i2*2-1)
                                 end
                                 A(3,1)=1/A(1,3)

                                 . . .
```

Fig. 3.6 MATLAB code for implementing ACO

```
                        if FA(4,5,i10*2-1)==0
                           A(4,5)=1/FA(5,4,i10*2-1)
                        else
                           A(4,5)=FA(4,5,i10*2-1)
                        end
                        A(5,4)=1/A(4,5)

                        % derive the eigenvalue and eigenvector
                        [E, V] = eig(A)

                        % update alpha cuts of fuzzy eigenvalue and eigenvector
                        if V(1,1)<lmin
                           lmin=V(1,1)
                        end
                        if V(1,1)>lmax
                           lmax=V(1,1)
                        end
                        ev=E(:,1)/sum(E(:,1))

                        for j=1:5
                           if ev(j)<evmin(j)
                              evmin(j)=ev(j)
                           end
                        end

                        for j=1:5
                           if ev(j)>evmax(j)
                              evmax(j)=ev(j)
                           end
                        end

                     end
                  end
               end
            end
         end
      end
   end

   % update fuzzy eigenvalue and eigenvector
   fl(i0,1)=lmin
   fl(23-i0,1)=lmax

   for j=1:5
      fev(j,i0)=evmin(j)
      fev(j,23-i0)=evmax(j)
   end
end

etime=now       % end time
(etime-stime)*60*60*24       % elapsed time
```

Fig. 3.6 (continued)

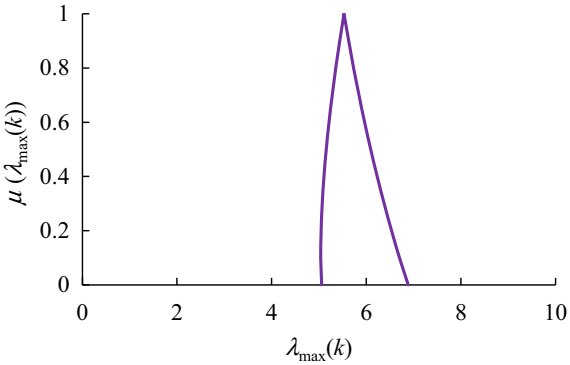


Fig. 3.7 Derived fuzzy maximum eigenvalue

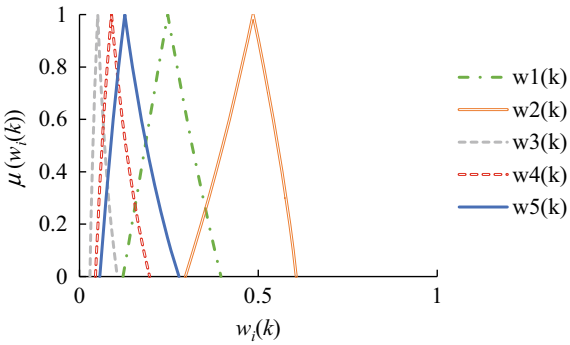


Fig. 3.8 Derived fuzzy priorities

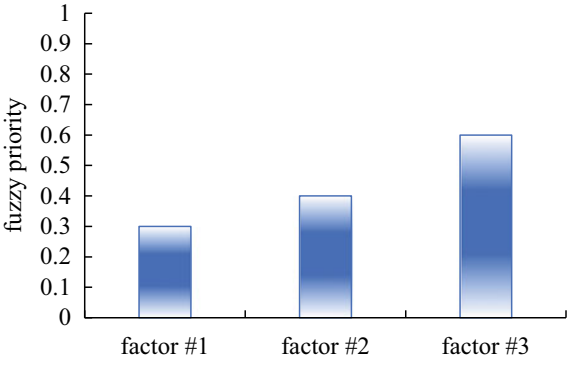


Fig. 3.9 Gradient bar chart for showing the derived fuzzy priorities

Definition 3.1. The fuzzy intersection (FI) of the fuzzy priorities evaluated by K decision makers on the i-th factor, indicated with $\tilde{w}_i(1)$–$\tilde{w}_i(K)$, is denoted by $\widetilde{\mathrm{FI}}(\{\tilde{w}_i(k)|k = 1-K\})$.

$$\mu_{\widetilde{\mathrm{FI}}(\{\tilde{w}_i(k)\})}(x) = \min_k \mu_{\tilde{w}_i(k)}(x) \tag{3.13}$$

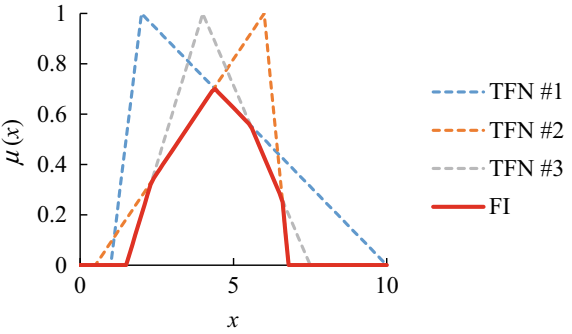


Fig. 3.10 FI of three TFNs

where $\mu_{\widetilde{\mathrm{FI}}(\{\tilde{w}_i(k)\})}(x)$ and $\mu_{\tilde{w}_i(k)}(x)$ denote the memberships of x in $\widetilde{\mathrm{FI}}(\{\tilde{w}_i(k)\})$ and $\tilde{w}_i(k)$, respectively.

Figure 3.10 provides an example showing the FI results of three TFNs.

In the view of Chen et al. [54], FI can generate a more reasonable aggregation result than FAM, FGM, and fuzzy weighted average (FWA).

Another XAI technique, traceable aggregation [45], can be used to illustrate the aggregation process using FI, as shown in Fig. 3.11. The original fuzzy priorities to be aggregated are shown on the left, connected by lines to the aggregation (FI) result on the right.

The FI result of fuzzy priorities can be easily obtained from their α-cuts:

$$\widetilde{\mathrm{FI}}(\{\tilde{w}_i(k)\}) = \{[\mathrm{FI}^L\{\tilde{w}_i(k)\}(\alpha),\ \mathrm{FI}^R\{\tilde{w}_i(k)\}(\alpha)] | \alpha = 0{-}1\} \tag{3.14}$$

where

$$\mathrm{FI}^L\{\tilde{w}_i(k)\}(\alpha) = \max_k(w_i^L(k)(\alpha)),\quad \alpha = 0{-}1, \tag{3.15}$$

$$\mathrm{FI}^R\{\tilde{w}_i(k)\}(\alpha) = \min_k(w_i^R(k)(\alpha)),\quad \alpha = 0{-}1 \tag{3.16}$$

if $\mathrm{FI}^L\{\tilde{w}_i(k)\}(\alpha) \le \mathrm{FI}^R\{\tilde{w}_i(k)\}(\alpha)$.

When other aggregators such as FGM and FAM are applied, traceable aggregation diagrams can also be used to show the aggregation process.

Example 3.2 The fuzzy priorities of criterion #1 derived by three decision makers are shown in Fig. 3.12. Table 3.3 summarizes their left and right α cuts. Equations (3.14)–(3.16) are used to derive the FI result. Table 3.4 shows the α cuts of the FI result. The FI result is illustrated in Fig. 3.13. The traceable aggregation diagram of this example is shown in Fig. 3.14.

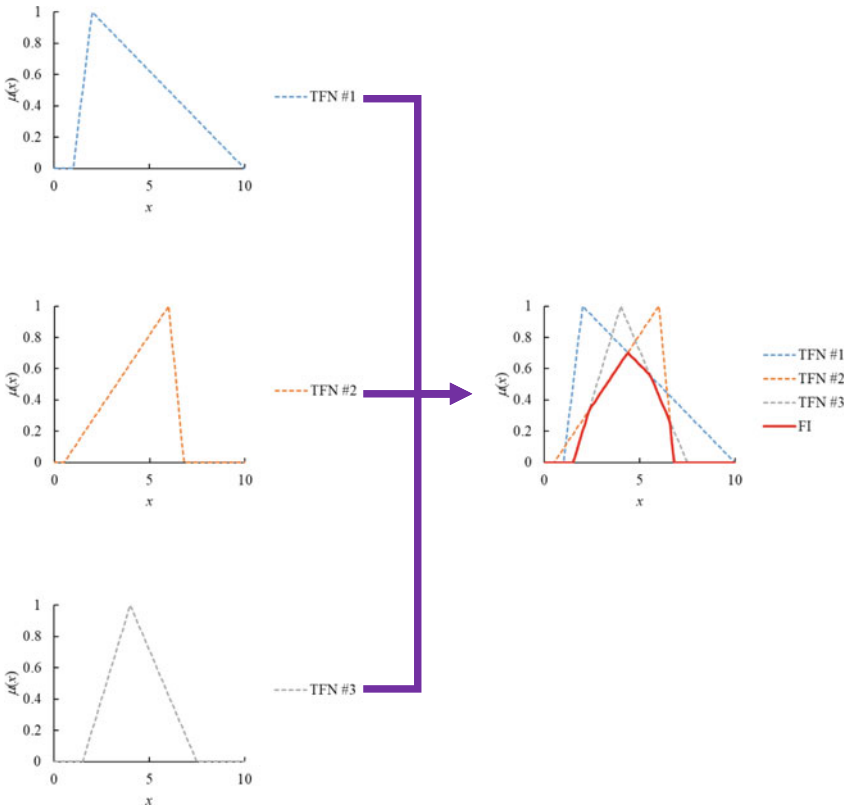


Fig. 3.11 Traceable aggregation for illustrating the aggregation process

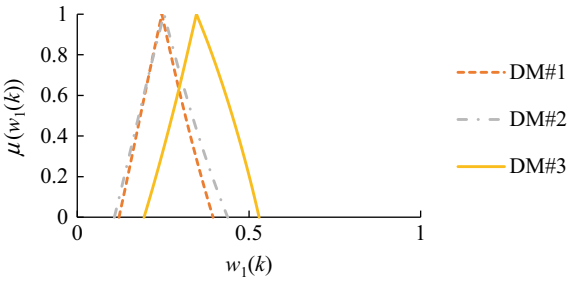


Fig. 3.12 Fuzzy priorities derived by three decision makers

Table 3.3 α-cuts of fuzzy priorities

Decision maker	α: $[w_1^L(k)(\alpha),\ w_1^R(k)(\alpha)]$
#1	0: [0.122, 0.396] 0.1: [0.134, 0.378] 0.2: [0.147, 0.362] 0.3: [0.159, 0.345] 0.4: [0.172, 0.33] 0.5: [0.184, 0.315] 0.6: [0.196, 0.301] 0.7: [0.209, 0.287] 0.8: [0.221, 0.273] 0.9: [0.234, 0.26] 1: [0.247, 0.247]
#2	0: [0.108, 0.438] 0.1: [0.123, 0.415] 0.2: [0.137, 0.394] 0.3: [0.151, 0.373] 0.4: [0.165, 0.354] 0.5: [0.179, 0.335] 0.6: [0.194, 0.317] 0.7: [0.208, 0.3] 0.8: [0.223, 0.284] 0.9: [0.238, 0.268] 1: [0.253, 0.253]
#3	0: [0.194, 0.53] 0.1: [0.212, 0.516] 0.2: [0.229, 0.501] 0.3: [0.245, 0.485] 0.4: [0.261, 0.469] 0.5: [0.277, 0.451] 0.6: [0.292, 0.433] 0.7: [0.306, 0.414] 0.8: [0.32, 0.393] 0.9: [0.333, 0.371] 1: [0.347, 0.347]

Table 3.4 α cuts of the FI result

α: $[\mathrm{FI}^L(\alpha),\ \mathrm{FI}^R(\alpha)]$
0: [0.194, 0.396]
0.1: [0.212, 0.378]
0.2: [0.229, 0.362]
0.3: [0.245, 0.345]
0.4: [0.261, 0.33]
0.5: [0.277, 0.315]
0.6: [0.292, 0.301]

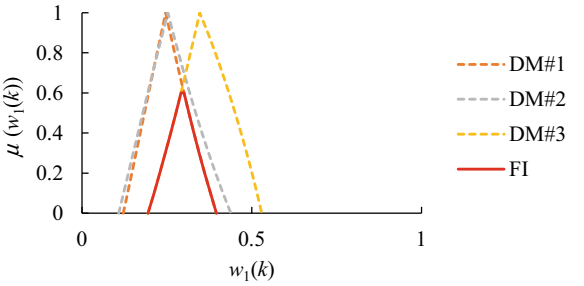


Fig. 3.13 FI result

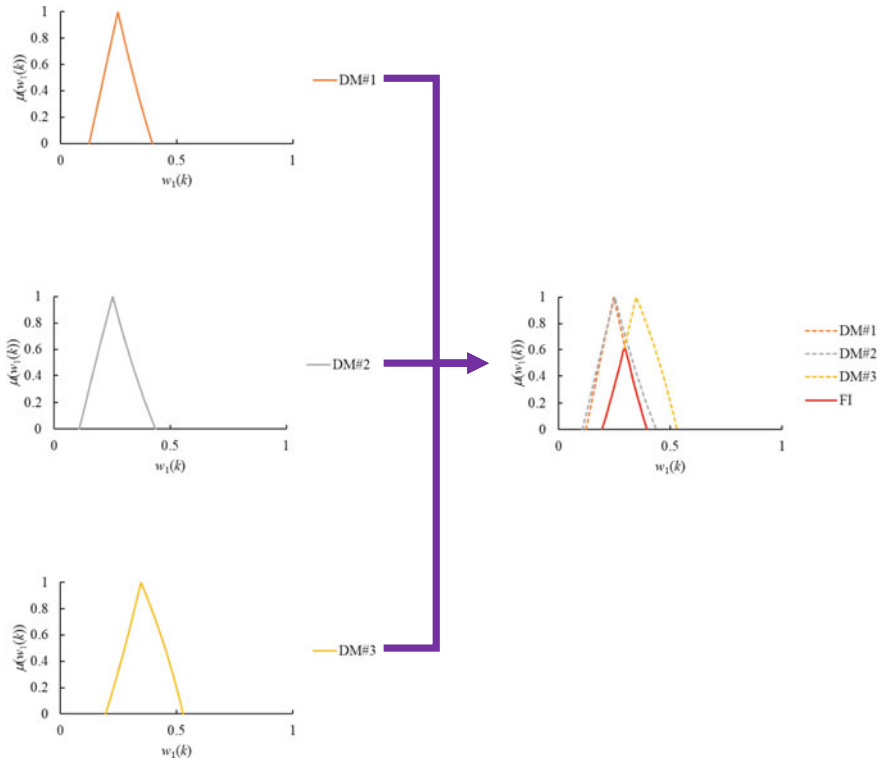


Fig. 3.14 Traceable aggregation diagram of this example

3.7 AI and XAI Applications in Evaluating the Overall Performance of Each Alternative

There are two popular AI methods for evaluating the overall performance of an alternative: fuzzy technique for order preference by similarity to the ideal solution (FTOPSIS) [55] and fuzzy Vise Kriterijumska Optimizacija I Kompromisno Resenje (fuzzy VIKOR) [56].

FTOPSIS is first introduced as follows. XAI tools and techniques applicable to FTOPSIS are also mentioned.

3.7.1 FTOPSIS

In FTOPSIS, first, the performance of an alternative in optimizing each factor is normalized using fuzzy distribution normalization [6] as

$$\begin{aligned}\tilde{\rho}_{li} &= \frac{\tilde{p}_{li}}{\sqrt{\sum_{m=1}^{M} \tilde{p}_{mi}^2}} \\ &= \frac{1}{\sqrt{1 + \sum_{m \neq l} \left(\frac{\tilde{p}_{mi}}{\tilde{p}_{li}}\right)^2}}\end{aligned} \tag{3.17}$$

where $\tilde{p}_{li}$ is the performance of the l-th alternative in optimizing the i-th factor; $\tilde{\rho}_{li}$ is the normalized performance.

Example 3.3 Four smart and automation technology applications will be evaluated and compared using FTOPSIS. Table 3.5 [34] summarizes the performances of each smart and automation technology application in optimizing the five factors. The performance of a smart and automation technology application in optimizing each factor is normalized using fuzzy distributive normalization. For example, the performance of alternative #2 in optimizing criterion #1 is normalized as

$$\begin{aligned}\tilde{\rho}_{21} &= \frac{1}{\sqrt{1 + \left(\frac{\tilde{p}_{11}}{\tilde{p}_{21}}\right)^2 + \left(\frac{\tilde{p}_{31}}{\tilde{p}_{21}}\right) + \left(\frac{\tilde{p}_{41}}{\tilde{p}_{21}}\right)^2}} \\ &= \frac{1}{\sqrt{1 + \left(\frac{(1.5,2.5,3.5)}{(3.0,4.0,5.0)}\right)^2 + \left(\frac{(1.5,2.5,3.5)}{(3.0,4.0,5.0)}\right)^2 + \left(\frac{(0.0,1.0,2.0)}{(3.0,4.0,5.0)}\right)^2}} \\ &\cong \frac{1}{\sqrt{1 + (0.3, 0.63, 1.17)^2 + (0.3, 0.63, 1.17)^2 + (0, 0.25, 0.67)^2}} \\ &\cong \frac{1}{\sqrt{1 + (0.09, 0.40, 1.37) + (0.09, 0.40, 1.37) + (0, 0.06, 0.45)}}\end{aligned}$$

$$= \frac{1}{\sqrt{(1.18, 1.86, 4.19)}}$$
$$\cong \frac{1}{(1.09, 1.36, 2.05)}$$
$$\cong (0.49, 0.74, 0.92)$$

the α-cuts of which can be easily derived (see Table 3.6). The normalization results are summarized in Table 3.7.

Subsequently, a fuzzy priority score $\tilde{s}_{li}$ is calculated based on the derived fuzzy priority:

Table 3.5 Performances of smart and automation technology applications in optimizing the five factors

l	Application	$\tilde{p}_{l1}$	$\tilde{p}_{l2}$	$\tilde{p}_{l3}$	$\tilde{p}_{l4}$	$\tilde{p}_{l5}$
1	Machine remote control	(1.5, 2.5, 3.5)	(4.0, 5.0, 5.0)	(0.0, 1.0, 2.0)	(0.0, 1.0, 2.0)	(3.0, 4.0, 5.0)
2	Worker's smart wristband	(3.0, 4.0, 5.0)	(0.0, 1.0, 2.0)	(4.0, 5.0, 5.0)	(4.0, 5.0, 5.0)	(4.0, 5.0, 5.0)
3	Worker's smart personal protection equipment	(1.5, 2.5, 3.5)	(3.0, 4.0, 5.0)	(3.0, 4.0, 5.0)	(3.0, 4.0, 5.0)	(1.5, 2.5, 3.5)
4	Smart warehouse	(0.0, 1.0, 2.0)	(4.0, 5.0, 5.0)	(0.0, 1.0, 2.0)	(1.5, 2.5, 3.5)	(3.0, 4.0, 5.0)

Table 3.6 α-cuts of $\tilde{\rho}_{21}$

α	α-cut
0	[0.49, 0.92]
0.1	[0.52, 0.90]
0.2	[0.54, 0.88]
0.3	[0.57, 0.87]
0.4	[0.59, 0.85]
0.5	[0.62, 0.83]
0.6	[0.64, 0.81]
0.7	[0.67, 0.79]
0.8	[0.69, 0.78]
0.9	[0.72, 0.76]
1	[0.74, 0.74]

Table 3.7 Normalized performances

l	Application	$\tilde{\rho}_{l1}$	$\tilde{\rho}_{l2}$	$\tilde{\rho}_{l3}$	$\tilde{\rho}_{l4}$	$\tilde{\rho}_{l5}$
1	Machine remote control	(0.18, 0.46, 0.72)	(0.48, 0.61, 0.71)	(0, 0.15, 0.37)	(0, 0.14, 0.36)	(0.36, 0.5, 0.69)
2	Worker's smart wristband	(0.49, 0.74, 0.93)	(0, 0.12, 0.3)	(0.57, 0.76, 0.86)	(0.53, 0.72, 0.83)	(0.45, 0.63, 0.74)
3	Worker's smart personal protection equipment	(0.23, 0.46, 0.73)	(0.38, 0.49, 0.66)	(0.46, 0.61, 0.78)	(0.42, 0.58, 0.76)	(0.17, 0.31, 0.51)
4	Smart warehouse	(0, 0.18, 0.49)	(0.48, 0.61, 0.71)	(0, 0.15, 0.37)	(0.2, 0.36, 0.57)	(0.36, 0.5, 0.69)

$$\tilde{s}_{li} = \tilde{w}_i(\times)\tilde{\rho}_{li} \tag{3.18}$$

In terms of the α-cuts of these fuzzy parameters, (3.18) can be expressed as

$$\begin{aligned} \tilde{s}_{li} &= [s_{li}^L(\alpha),\ s_{li}^R(\alpha)] \\ &= [w_i^L(\alpha)\rho_{li}^L(\alpha),\ w_i^R(\alpha)\rho_{li}^R(\alpha)] \end{aligned} \quad \forall\alpha \tag{3.19}$$

The fuzzy ideal (zenith) point $\tilde{\mathbf{\Lambda}}^+$ and fuzzy anti-ideal (nadir) point $\tilde{\mathbf{\Lambda}}^-$ are respectively specified as

$$\tilde{\mathbf{\Lambda}}^+ = [\tilde{\Lambda}_i^+] = [\max_l \tilde{s}_{li}] \tag{3.20}$$

$$\tilde{\mathbf{\Lambda}}^- = [\tilde{\Lambda}_i^-] = [\min_l \tilde{s}_{li}] \tag{3.21}$$

or

$$\begin{aligned} \tilde{\Lambda}_i^+ &= [\Lambda_i^{+L}(\alpha),\ \Lambda_i^{+R}(\alpha)] \\ &= [\max_l s_{li}^L(\alpha), \max_l s_{li}^R(\alpha)] \end{aligned} \tag{3.22}$$

$$\begin{aligned} \tilde{\Lambda}_i^- &= [\Lambda_i^{-L}(\alpha),\ \Lambda_i^{-R}(\alpha)] \\ &= [\min_l s_{li}^L(\alpha), \min_l s_{li}^R(\alpha)] \end{aligned} \tag{3.23}$$

Example 3.4 In the previous example, the fuzzy priority score of alternative #2 for optimizing criterion #1 is computed based on the aggregation result of the fuzzy priorities obtained by all experts using FI in Example 3.2:

$$\alpha = 0 : s_{21}^{L}(\alpha) = 0.19 \times 0.49 = 0.10; \quad s_{21}^{R}(\alpha) = 0.40 \times 0.93 = 0.36$$

$$\ldots$$

$$\alpha = 0.6 : s_{21}^{L}(\alpha) = 0.19; \quad s_{21}^{R}(\alpha) = 0.25$$

In sum, $\tilde{s}_{21} = \{\{0, [0.10, 0.36]\}, \ldots \{0.6, [0.19, 0.25]\}\}$.

Example 3.5 In the previous example, Table 3.8 summarizes the fuzzy priority scores for all alternatives to optimize each criterion. From this, the fuzzy ideal (zenith) point and the fuzzy anti-ideal (nadir) point can be established, as shown in Table 3.9.

The fuzzy distances from each alternative to the two reference points are measured as

Table 3.8 Fuzzy priority scores for all alternatives to optimize each criterion

l	$\tilde{s}_{l1}$ (α: α cut)	$\tilde{s}_{l2}$ (α: α cut)	$\tilde{s}_{l3}$ (α: α cut)	$\tilde{s}_{l4}$ (α: α cut)	$\tilde{s}_{l5}$ (α: α cut)
1	0: [0.04, 0.29] 0.1: [0.04, 0.26] 0.2: [0.05, 0.24] 0.3: [0.07, 0.22] 0.4: [0.08, 0.2] 0.5: [0.09, 0.19] 0.6: [0.1, 0.17]	0: [0.14, 0.34] 0.1: [0.16, 0.32] 0.2: [0.17, 0.31] 0.3: [0.19, 0.3] 0.4: [0.2, 0.29] 0.5: [0.22, 0.28]	0: [0, 0.04] 0.1: [0, 0.03] 0.2: [0, 0.03] 0.3: [0, 0.02] 0.4: [0, 0.02]	0: [0, 0.07] 0.1: [0, 0.06] 0.2: [0, 0.05] 0.3: [0, 0.04] 0.4: [0, 0.04] 0.5: [0.01, 0.03] 0.6: [0.01, 0.03] 0.7: [0.01, 0.02]	0: [0.02, 0.09] 0.1: [0.02, 0.08] 0.2: [0.03, 0.07] 0.3: [0.03, 0.07] 0.4: [0.03, 0.06] 0.5: [0.04, 0.05]
2	0: [0.1, 0.37] 0.1: [0.11, 0.35] 0.2: [0.12, 0.32] 0.3: [0.14, 0.3] 0.4: [0.15, 0.28] 0.5: [0.17, 0.26] 0.6: [0.19, 0.24]	0: [0, 0.14] 0.1: [0, 0.13] 0.2: [0.01, 0.12] 0.3: [0.01, 0.11] 0.4: [0.02, 0.1] 0.5: [0.02, 0.09]	0: [0.04, 0.09] 0.1: [0.04, 0.08] 0.2: [0.04, 0.07] 0.3: [0.04, 0.07] 0.4: [0.05, 0.06]	0: [0.03, 0.16] 0.1: [0.04, 0.15] 0.2: [0.04, 0.13] 0.3: [0.05, 0.12] 0.4: [0.05, 0.11] 0.5: [0.06, 0.1] 0.6: [0.06, 0.09] 0.7: [0.07, 0.08]	0: [0.03, 0.1] 0.1: [0.03, 0.09] 0.2: [0.03, 0.08] 0.3: [0.04, 0.07] 0.4: [0.04, 0.07] 0.5: [0.05, 0.06]
3	0: [0.04, 0.29] 0.1: [0.05, 0.27] 0.2: [0.06, 0.25] 0.3: [0.07, 0.23] 0.4: [0.08, 0.21] 0.5: [0.1, 0.19] 0.6: [0.11, 0.17]	0: [0.11, 0.32] 0.1: [0.12, 0.3] 0.2: [0.14, 0.29] 0.3: [0.15, 0.27] 0.4: [0.16, 0.26] 0.5: [0.17, 0.24]	0: [0.03, 0.08] 0.1: [0.03, 0.07] 0.2: [0.03, 0.07] 0.3: [0.04, 0.06] 0.4: [0.04, 0.05]	0: [0.03, 0.15] 0.1: [0.03, 0.13] 0.2: [0.03, 0.12] 0.3: [0.04, 0.11] 0.4: [0.04, 0.1] 0.5: [0.05, 0.09] 0.6: [0.05, 0.08] 0.7: [0.06, 0.07]	0: [0.01, 0.07] 0.1: [0.01, 0.06] 0.2: [0.01, 0.05] 0.3: [0.02, 0.05] 0.4: [0.02, 0.04] 0.5: [0.02, 0.04]
4	0: [0, 0.19] 0.1: [0, 0.17] 0.2: [0.01, 0.15] 0.3: [0.01, 0.14] 0.4: [0.02, 0.12] 0.5: [0.03, 0.11] 0.6: [0.03, 0.09]	0: [0.14, 0.34] 0.1: [0.16, 0.32] 0.2: [0.17, 0.31] 0.3: [0.19, 0.3] 0.4: [0.2, 0.29] 0.5: [0.22, 0.28]	0: [0, 0.04] 0.1: [0, 0.03] 0.2: [0, 0.03] 0.3: [0, 0.02] 0.4: [0, 0.02]	0: [0.01, 0.11] 0.1: [0.01, 0.1] 0.2: [0.02, 0.09] 0.3: [0.02, 0.08] 0.4: [0.02, 0.07] 0.5: [0.03, 0.06] 0.6: [0.03, 0.05] 0.7: [0.03, 0.05]	0: [0.02, 0.09] 0.1: [0.02, 0.08] 0.2: [0.03, 0.07] 0.3: [0.03, 0.07] 0.4: [0.03, 0.06] 0.5: [0.04, 0.05]

Table 3.9 Fuzzy ideal (zenith) point and fuzzy anti-ideal (nadir) point

l	$\tilde{\Lambda}_1^{+/-}$ (α: α cut)	$\tilde{\Lambda}_2^{+/-}$ (α: α cut)	$\tilde{\Lambda}_3^{+/-}$ (α: α cut)	$\tilde{\Lambda}_4^{+/-}$ (α: α cut)	$\tilde{\Lambda}_5^{+/-}$ (α: α cut)
$\tilde{\mathbf{\Lambda}}^+$	0: [0.1, 0.37] 0.1: [0.11, 0.35] 0.2: [0.12, 0.32] 0.3: [0.14, 0.3] 0.4: [0.15, 0.28] 0.5: [0.17, 0.26] 0.6: [0.19, 0.24]	0: [0.14, 0.34] 0.1: [0.16, 0.32] 0.2: [0.17, 0.31] 0.3: [0.19, 0.3] 0.4: [0.2, 0.29] 0.5: [0.22, 0.28]	0: [0.04, 0.09] 0.1: [0.04, 0.08] 0.2: [0.04, 0.07] 0.3: [0.04, 0.07] 0.4: [0.05, 0.06]	0: [0.03, 0.16] 0.1: [0.04, 0.15] 0.2: [0.04, 0.13] 0.3: [0.05, 0.12] 0.4: [0.05, 0.11] 0.5: [0.06, 0.1] 0.6: [0.06, 0.09] 0.7: [0.07, 0.08]	0: [0.03, 0.1] 0.1: [0.03, 0.09] 0.2: [0.03, 0.08] 0.3: [0.04, 0.07] 0.4: [0.04, 0.07] 0.5: [0.05, 0.06]
$\tilde{\mathbf{\Lambda}}^-$	0: [0, 0.19] 0.1: [0, 0.17] 0.2: [0.01, 0.15] 0.3: [0.01, 0.14] 0.4: [0.02, 0.12] 0.5: [0.03, 0.11] 0.6: [0.03, 0.09]	0: [0, 0.14] 0.1: [0, 0.13] 0.2: [0.01, 0.12] 0.3: [0.01, 0.11] 0.4: [0.02, 0.1] 0.5: [0.02, 0.09]	0: [0, 0.04] 0.1: [0, 0.03] 0.2: [0, 0.03] 0.3: [0, 0.02] 0.4: [0, 0.02]	0: [0, 0.07] 0.1: [0, 0.06] 0.2: [0, 0.05] 0.3: [0, 0.04] 0.4: [0, 0.04] 0.5: [0.01, 0.03] 0.6: [0.01, 0.03] 0.7: [0.01, 0.02]	0: [0.01, 0.07] 0.1: [0.01, 0.06] 0.2: [0.01, 0.05] 0.3: [0.02, 0.05] 0.4: [0.02, 0.04] 0.5: [0.02, 0.04]

$$\tilde{d}_l^+ = \sqrt{\sum_{i=1}^{n} (\tilde{\Lambda}_i^+(-)\tilde{s}_{li})^2} \tag{3.24}$$

$$\tilde{d}_l^- = \sqrt{\sum_{i=1}^{n} (\tilde{\Lambda}_i^-(-)\tilde{s}_{li})^2} \tag{3.25}$$

In their α-cuts,

$$\begin{aligned} \tilde{d}_l^+ &= [d_l^{+L}(\alpha), d_l^{+R}(\alpha)] \\ &= \left[\sqrt{\sum_{i=1}^{n} \max(\Lambda_i^{+L}(\alpha) - s_{li}^R(\alpha), 0)^2}, \sqrt{\sum_{i=1}^{n} (\Lambda_i^{+R}(\alpha) - s_{li}^L(\alpha))^2} \right] \end{aligned} \tag{3.26}$$

$$\begin{aligned} \tilde{d}_l^- &= [d_l^{-L}(\alpha), d_l^{-R}(\alpha)] \\ &= \left[\sqrt{\sum_{i=1}^{n} \max(s_{li}^L(\alpha) - \Lambda_i^{-R}(\alpha), 0)^2}, \sqrt{\sum_{i=1}^{n} (s_{li}^R(\alpha) - \Lambda_i^{-L}(\alpha))^2} \right] \end{aligned} \tag{3.27}$$

Finally, the fuzzy closeness of the alternative is evaluated as

$$\tilde{C}_l = \frac{\tilde{d}_l^-}{\tilde{d}_l^+(+)\tilde{d}_l^-} \tag{3.28}$$

or

$$\begin{aligned}\tilde{C}_l &= [C_l^L(\alpha), C_l^R(\alpha)] \\ &= \left[\frac{d_l^{-L}(\alpha)}{d_l^{+R}(\alpha)+d_l^{-L}(\alpha)}, \frac{d_l^{-R}(\alpha)}{d_l^{+L}(\alpha)+d_l^{-R}(\alpha)}\right]\end{aligned} \tag{3.29}$$

If the fuzzy closeness of an alternative is higher, its is more suitable. To obtain an absolute ranking, the COG method can be used to defuzzify the fuzzy closenesses of alternatives and then compare [57]:

$$D(\tilde{C}_l) = \frac{\int_0^1 \alpha\left(\frac{C_l^L(\alpha)+C_l^R(\alpha)}{2}\right)d\alpha}{\int_0^1 \alpha d\alpha} \tag{3.30}$$

Example 3.6 Following Example 3.5, each alternative is evaluated by comparing the distances to the two reference points, which are summarized in Table 3.10. Subsequently, the fuzzy closeness of each alternative is evaluated (see Table 3.11).

The evaluation mechanism in FTOPSIS is to compare the distances of alternatives to two reference points, which may not be very intuitive. Highlighting the difference between the two distances enhances the interpretability of FTOPSIS. To this end, this study proposes a new XAI tool, the bi-directional scatterplot, as shown in Fig. 3.15. In this figure,

Table 3.10 Fuzzy distances to the two reference points

l	1	2	3	4
$\tilde{d}_l^+(\alpha{:}\alpha$ cut)	0: [0, 0.44] 0.1: [0, 0.39] 0.2: [0.01, 0.34] 0.3: [0.02, 0.3] 0.4: [0.03, 0.25]	0: [0, 0.46] 0.1: [0.03, 0.42] 0.2: [0.05, 0.38] 0.3: [0.08, 0.34] 0.4: [0.1, 0.31]	0: [0, 0.43] 0.1: [0, 0.38] 0.2: [0, 0.34] 0.3: [0, 0.29] 0.4: [0, 0.25]	0: [0, 0.46] 0.1: [0, 0.42] 0.2: [0.01, 0.37] 0.3: [0.02, 0.34] 0.4: [0.04, 0.3]
$\tilde{d}_l^-(\alpha{:}\alpha$ cut)	0: [0, 0.46] 0.1: [0.03, 0.42] 0.2: [0.05, 0.39] 0.3: [0.08, 0.36] 0.4: [0.1, 0.33]	0: [0, 0.45] 0.1: [0, 0.41] 0.2: [0.01, 0.37] 0.3: [0.02, 0.34] 0.4: [0.04, 0.3]	0: [0, 0.47] 0.1: [0, 0.43] 0.2: [0.02, 0.39] 0.3: [0.04, 0.35] 0.4: [0.07, 0.32]	0: [0, 0.41] 0.1: [0.03, 0.38] 0.2: [0.05, 0.35] 0.3: [0.08, 0.33] 0.4: [0.1, 0.3]

Table 3.11 Fuzzy closeness of each alternative

l	1	2	3	4
$\tilde{C}_l$	0: [0, 1] 0.1: [0.06, 0.99] 0.2: [0.13, 0.97] 0.3: [0.21, 0.95] 0.4: [0.29, 0.92]	0: [0, 1] 0.1: [0.01, 0.94] 0.2: [0.03, 0.88] 0.3: [0.05, 0.81] 0.4: [0.12, 0.74]	0: [0, 1] 0.1: [0, 1] 0.2: [0.05, 1] 0.3: [0.12, 1] 0.4: [0.21, 1]	0: [0, 1] 0.1: [0.06, 0.99] 0.2: [0.12, 0.97] 0.3: [0.19, 0.95] 0.4: [0.26, 0.88]

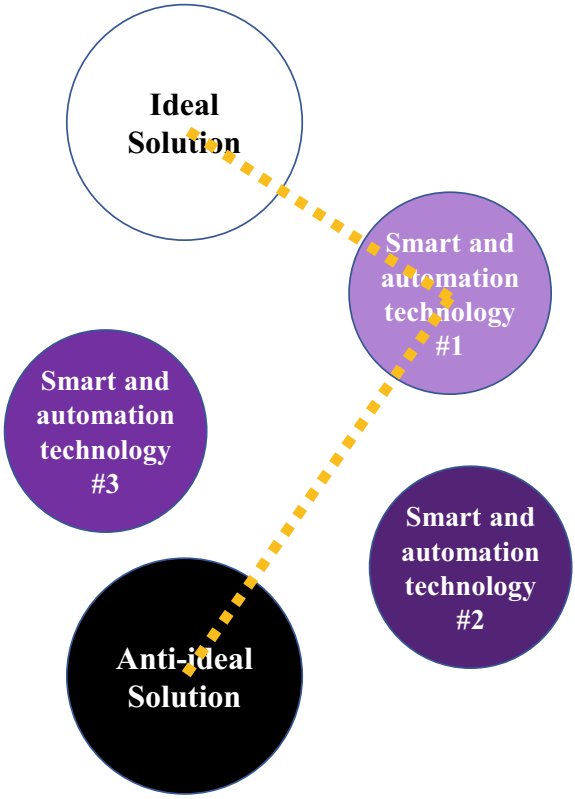


Fig. 3.15 Bidirectional scatterplot for illustrating the comparison mechanism in FTOPSIS

- The ideal solution is on the top, and the anti-ideal solution is on the bottom.
- The ideal solution is white, and the anti-ideal solution is black.
- All alternatives can be placed anywhere in the bi-directional scatterplot, as long as the following requirement is met. As shown, the distance of each alternative to two reference points was measured using the FTOPSIS method.
- The closer an alternative is to the ideal solution, the whiter (lighter) it will be. Conversely, if the alternative gets closer to the anti-ideal solution, it will be darker.

3.7.2 Fuzzy VIKOR

Subsequently, XAI tools and techniques for explaining fuzzy VIKOR [58] are introduced. The fuzzy VIKOR method can also be applied to evaluate the overall performance of an alternative. Several studies on fuzzy VIKOR applications exist in the literature. However, most past studies have been based on fuzzy sets with triangular or trapezoidal membership functions. In contrast, the lower and upper membership functions of the aggregation result may not belong to the two types.

The fuzzy VIKOR method comprises the following steps:

Step 1. Determine the ideal and anti-ideal performance in optimizing each factor:

$$\tilde{p}_i^* = \max_l \tilde{p}_{li} \tag{3.31}$$

$$\tilde{p}_i^- = \min_l \tilde{p}_{li} \tag{3.32}$$

Step 2. Compute the normalized fuzzy distance from the ideal performance [56]:

$$\tilde{d}_{li} = \frac{\tilde{p}_i^*(-)\tilde{p}_{li}}{p_{i3}^* - p_{i1}^-} \tag{3.33}$$

Step 3. Derive $\tilde{S}_l$ and $\tilde{R}_l$ [56]:

$$\tilde{S}_l = \sum_{i=1}^{n} (\tilde{w}_i(\text{all})(\times)\tilde{d}_{li}) \tag{3.34}$$

$$\tilde{R}_l = \max_i (\tilde{w}_i(\text{all})(\times)\tilde{d}_{li}) \tag{3.35}$$

The α cut can be derived using ACO as follows:

$$\begin{aligned} \tilde{S}_l &= \bigcup_\alpha \left[S_l^L(\alpha),\ S_l^R(\alpha)\right] \\ &= \bigcup_\alpha \left[\sum_{i=1}^{n} (w_i^L(\text{all})(\alpha) d_{li}^L(\alpha)),\ \sum_{i=1}^{n} (w_i^R(\text{all})(\alpha) d_{li}^R(\alpha))\right] \end{aligned} \tag{3.36}$$

$$\begin{aligned} \tilde{R}_l &= \bigcup_\alpha \left[R_l^L(\alpha),\ R_l^R(\alpha)\right] \\ &= \bigcup_\alpha \left[\max_i (w_i^L(\text{all})(\alpha) d_{li}^L(\alpha)),\ \max_i (\text{all})(w_i^R(\alpha) d_{li}^R(\alpha))\right] \end{aligned} \tag{3.37}$$

Step 4. Combine $\tilde{S}_l$ and $\tilde{R}_l$ into $\tilde{Q}_l$ as follows [56]:

$$\begin{aligned} \tilde{Q}_l &= \omega N(\tilde{S}_l)(+)(1-\omega)N(\tilde{R}_l) \\ &= \omega \cdot \frac{\tilde{S}_l(-)\min_r \tilde{S}_r}{\max\left(\max_r \tilde{S}_r\right) - \min\left(\min_r \tilde{S}_r\right)} (+)(1-\omega) \cdot \frac{\tilde{R}_l(-)\min_r \tilde{R}_r}{\max\left(\max_r \tilde{R}_r\right) - \min\left(\min_r \tilde{R}_r\right)} \end{aligned} \tag{3.38}$$

The α cut of $\tilde{Q}_l$ can be derived as follows:

$$\begin{aligned}\tilde{Q}_l &= \bigcup_{\alpha}\left[Q_l^L(\alpha),\ Q_l^R(\alpha)\right] \\ &= \bigcup_{\alpha}\left[\omega\cdot\frac{S_l^L(\alpha)-\min\limits_r S_r^R(\alpha)}{\max\limits_r S_r^R(\alpha)-\min\limits_r S_r^L(\alpha)}+(1-\omega)\cdot\frac{R_l^L(\alpha)-\min\limits_r R_r^R(\alpha)}{\max\limits_r R_r^R(\alpha)-\min\limits_r R_r^L(\alpha)},\right. \\ &\quad \left.\omega\cdot\frac{S_l^R(\alpha)-\min\limits_r S_r^L(\alpha)}{\max\limits_r S_r^R(\alpha)-\min\limits_r S_r^L(\alpha)}+(1-\omega)\cdot\frac{R_l^R(\alpha)-\min\limits_r R_r^L(\alpha)}{\max\limits_r R_r^R(\alpha)-\min\limits_r R_r^L(\alpha)}\right]\end{aligned} \tag{3.39}$$

Step 5. Defuzzify $\tilde{Q}_l$ using the center-of-gravity method [59]:

$$D(\tilde{Q}_l)=\frac{\sum_\alpha \alpha\left(\frac{Q_l^L(\alpha)+Q_l^R(\alpha)}{2}\right)}{\sum_\alpha \alpha} \tag{3.40}$$

Step 6. Choose the alternative that achieved the highest $D(\tilde{Q}_l)$ value.

Fuzzy VIKOR is less explainable than other more intuitive methods, such as fuzzy weighted average. The comparison mechanism in fuzzy VIKOR is based on the distance from an alternative to the ideal solution. Therefore, presenting the differences in the distances of alternatives from the ideal solution enhances the explainability of fuzzy VIKOR, as illustrated in Fig. 3.16, which is called the segmented distance diagram. In this figure,

- All possible alternatives surround the ideal solution.
- The distance between an alternative and the ideal solution is $\tilde{Q}_l$, which has two segments: the longest distance ($(1-\omega)N(\tilde{R}_l)$) and overall distance ($\omega N(\tilde{S}_l)$) in red and blue lines, respectively.

For example, in Fig. 3.16, Alternative 2 is the best choice because it is closest to the ideal solution. In addition, the advantage of Alternative 2 over the other alternatives lies in its overall distance, which is much shorter than the others.

3.8 Assessing the Effectiveness of XAI Applications for Decision Making in the Manufacturing Domain

For an XAI technique or tool to be effective for decision making in the manufacturing domain, the following conditions need to be met:

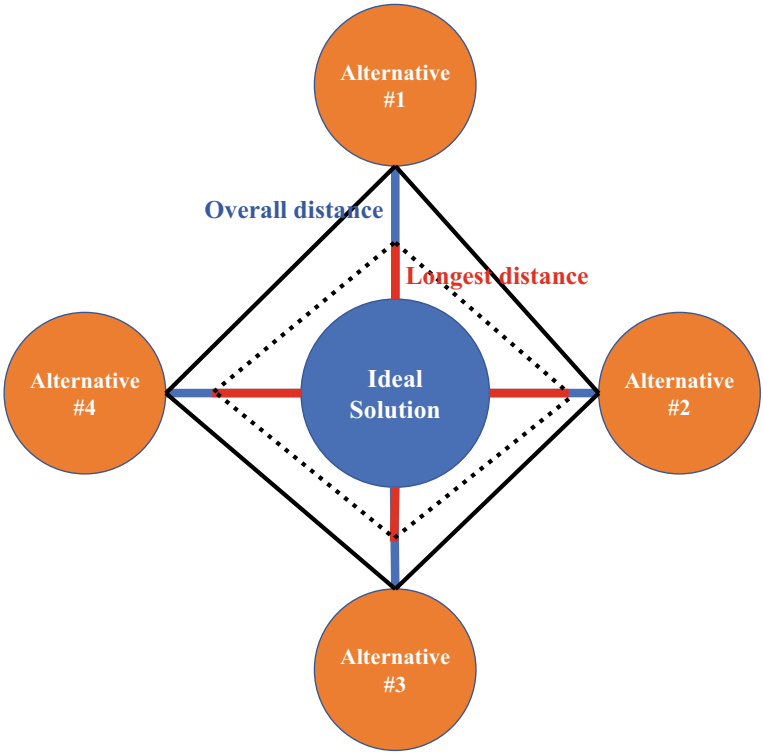


Fig. 3.16 Segmented distance diagram for illustrating the comparison mechanism in fuzzy VIKOR

- It is easy to understand.
- It is easy to communicate.
- It can explain the reasoning process.
- No background knowledge is required.
- It can provide an intuitive relationship between the attributes of an alternative and the overall performance.
- It facilitates the visual comparison of factors, alternatives, and/or decision makers.
- It can derive the fuzzy priorities of factors precisely from pairwise comparison results by minimize the sum of squared deviations (SSD):

$$\widetilde{\mathrm{SSD}} = \sum_{i=1}^{n} \sum_{j \neq i} \left(\frac{\tilde{w}_i}{\tilde{w}_j} (-) \tilde{a}_{ij} \right)^2 \tag{3.41}$$

References

1. M. Dağdeviren, Decision making in equipment selection: an integrated approach with AHP and PROMETHEE. J. Intell. Manuf. **19**(4), 397–406 (2008)
2. T. Chen, Y.-C. Wang, A fuzzy-neural approach for supporting three-objective job scheduling in a wafer fabrication factory. Neural Comput. Appl. **23**(1S), 353–367 (2013)
3. T.-C.T. Chen, Y.-C. Wang, AI applications to kaizen management, in *Artificial Intelligence and Lean Manufacturing* (2022), pp. 37–52
4. R. Venkata Rao, B.K. Patel, Decision making in the manufacturing environment using an improved PROMETHEE method. Int. J. Prod. Res. **48**(16), 4665–4682 (2010)
5. T.C.T. Chen, Y.C. Wang, *Artificial Intelligence and Lean Manufacturing* (Springer Nature, 2022)
6. T. Chen, A FAHP-FTOPSIS approach for choosing mid-term occupational healthcare measures amid the COVID-19 pandemic. Health Policy Technol. **10**(2), 100517 (2021)
7. T.C.T. Chen, *Production Planning and Control in Semiconductor Manufacturing: Big Data Analytics and Industry 4.0 Applications* (Springer Nature, 2022)
8. T. Chen, Y.-C. Wang, Hybrid big data analytics and Industry 4.0 approach for projecting cycle time ranges. Int. J. Adv. Manufact. Technol. **120**, 279–295 (2022)
9. M. van Doorn, What does SMART technology actually mean? (2015). http://labs.sogeti.com/wat-smart-technology-actually-mean/
10. A. Al-Refaie, T. Chen, M. Judeh, Optimal operating room scheduling for normal and unexpected events in a smart hospital. Oper. Res. Int. J. **18**(3), 579–602 (2018)
11. Y.C. Lin, Y.C. Wang, T.C.T. Chen, H.F. Lin, Evaluating the suitability of a smart technology application for fall detection using a fuzzy collaborative intelligence approach. Mathematics **7**(11), 1097 (2019)
12. M.C. Chiu, T.C.T. Chen, Assessing sustainable effectiveness of the adjustment mechanism of a ubiquitous clinic recommendation system. Health Care Manag. Sci. **23**(2), 239–248 (2020)
13. M. Dotoli, A. Fay, M. Miśkowicz, C. Seatzu, An overview of current technologies and emerging trends in factory automation. Int. J. Prod. Res. **57**(15–16), 5047–5067 (2019)
14. R.A. Velásquez, J.M. Lara, Robot unit for cost and time balance using automatic inspection on overhead lines, in *IEEE ANDESCON* (2016), pp. 1–4
15. Z. An, Y. Wang, L. Zheng, X. Liu, Adaptive recognition of intelligent inspection system for cable brackets in multiple assembly scenes. Int. J. Adv. Manufact. Technol. **108**(11–12), 3373–3389 (2020)
16. P.J. Costa, N. Moreira, D. Campos, J. Gonçalves, J. Lima, P.L. Costa, Localization and navigation of an omnidirectional mobile robot: The robot@ factory case study. IEEE Revista Iberoamericana de Tecnologias del Aprendizaje **11**(1), 1–9 (2016)
17. Y.C. Wang, T. Chen, H. Chiang, H.C. Pan, A simulation analysis of part launching and order collection decisions for a flexible manufacturing system. Simul. Model. Pract. Theory **69**, 80–91 (2016)
18. S. Huang, M. Ishikawa, Y. Yamakawa, A coarse-to-fine framework for accurate positioning under uncertainties—from autonomous robot to human–robot system. Int. J. Adv. Manufact. Technol. **108**(9–10), 2929–2944 (2020)
19. T. Chen, Y.C. Lin, Feasibility evaluation and optimization of a smart manufacturing system based on 3D printing: a review. Int. J. Intell. Syst. **32**(4), 394–413 (2017)
20. T. Chen, Y.C. Wang, An advanced IoT system for assisting ubiquitous manufacturing with 3D printing. Int. J. Adv. Manufact. Technol. **103**(5–8), 1721–1733 (2019)
21. Y.C. Wang, T. Chen, Y.L. Yeh, Advanced 3D printing technologies for the aircraft industry: a fuzzy systematic approach for assessing the critical factors. Int. J. Adv. Manufact. Technol. **105**(10), 4059–4069 (2019)
22. A.S. Yildiz, K. Davut, B. Koc, O. Yilmaz, Wire arc additive manufacturing of high-strength low alloy steels: study of process parameters and their influence on the bead geometry and mechanical characteristics. Int. J. Adv. Manufact. Technol. **108**(11–12), 3391–3404 (2020)

23. T.C.T. Chen, Y.C. Lin, Diverse three-dimensional printing capacity planning for manufacturers. Robot. Comput. Integr. Manufact. **67**, 102052 (2021)
24. Y.C. Lin, T. Chen, A ubiquitous manufacturing network system. Robot. Comput. Integr. Manufact. **45**, 157–167 (2017)
25. X.V. Wang, L. Wang, A. Mohammed, M. Givehchi, Ubiquitous manufacturing system based on cloud: a robotics application. Robot. Comput. Integr. Manufact. **45**, 116–125 (2017)
26. T. Chen, Y.C. Wang, Estimating simulation workload in cloud manufacturing using a classifying artificial neural network ensemble approach. Robot. Comput. Integr. Manufact. **38**, 42–51 (2016)
27. L. Ren, L. Zhang, L. Wang, F. Tao, X. Chai, Cloud manufacturing: key characteristics and applications. Int. J. Comput. Integr. Manuf. **30**(6), 501–515 (2017)
28. E.J. Ghomi, A.M. Rahmani, N.N. Qader, Cloud manufacturing: challenges, recent advances, open research issues, and future trends. Int. J. Adv. Manufact. Technol. **102**(9–12), 3613–3639 (2019)
29. F. Tao, Y. Zuo, L. Da Xu, L. Zhang, IoT-based intelligent perception and access of manufacturing resource toward cloud manufacturing. IEEE Trans. Industr. Inf. **10**(2), 1547–1557 (2014)
30. K. Thramboulidis, F. Christoulakis, UML4IoT—a UML-based approach to exploit IoT in cyber-physical manufacturing systems. Comput. Ind. **82**, 259–272 (2016)
31. J. Lee, B. Bagheri, H.A. Kao, A cyber-physical systems architecture for industry 4.0-based manufacturing systems. Manufact. Lett. **3**, 18–23 (2015)
32. Canadian Plastics, Will COVID-19 create a boom in factory automation? (2020). https://www.canplastics.com/features/will-covid-19-create-a-boom-in-factory-automation/
33. J. Miller, Basketball tech used to fight Covid-19 on factory floor (2020). https://www.ft.com/content/ffa5e644-e0b4-4656-b3d6-3069e16774ef
34. T. Chen, C.-W. Lin, Smart and automation technologies for ensuring the long-term operation of a factory amid the COVID-19 pandemic: an evolving fuzzy assessment approach. Int. J. Adv. Manuf. Technol. **111**, 3545–3558 (2020)
35. V.B. Ramirez, Coronavirus may mean automation is coming sooner than we thought (2020). https://singularityhub.com/2020/03/19/coronavirus-may-mean-automation-is-coming-sooner-than-we-thought/
36. C.Y. Kao, C.S. Fahn, A human-machine interaction technique: hand gesture recognition based on hidden Markov models with trajectory of hand motion. Proc. Eng. **15**, 3739–3743 (2011)
37. T. Kuremoto, T. Yamane, L. Feng, K. Kobayashi, M. Obayashi, A human-machine interaction system: A voice command learning system using PL-G-SOM, in *International Conference on Management and Service Science* (2011), pp. 1–4
38. C. Parga, X. Li, W. Yu, Smartphone-based human machine interface with application to remote control of robot arm, in *2013 IEEE International Conference on Systems, Man, and Cybernetics* (2013), pp. 2316–2321
39. A. Masood, K.B. Khan, T. Younas, A.R. Khalid, Design of wearable prototype smart wristband for remote health monitoring using Internet of things, in *International Conference on Intelligent Technologies and Applications* (2019), pp. 3–13
40. T.C.T. Chen, *Advances in Fuzzy Group Decision Making* (Springer Nature, 2022)
41. T.C.T. Chen, Introduction to fuzzy group decision making, in *Advances in Fuzzy Group Decision Making* (2022), pp. 1–7
42. M. Hamstra, Pandemic trend: Seeking efficiency and safety, businesses accelerate warehouse automation (2020). https://www.uschamber.com/co/good-company/launch-pad/warehouse-robotic-automation-coronavirus-pandemic
43. M. Borak, Chinese police now have AI helmets for temperature screening (2020). https://www.msn.com/en-sg/news/world/chinese-police-now-have-ai-helmets-for-temperature-screening/ar-BB10vUiN
44. E. Yu, Singapore issues COVID-19 contact tracing wearables to 'vulnerable seniors' (2020). https://www.zdnet.com/article/singapore-issues-covid-19-contact-tracing-wearables-to-vulnerable-seniors/

45. Y.-C. Lin, T. Chen, A type-II fuzzy approach with explainable artificial intelligence for nature-based leisure travel destination selection amid the COVID-19 pandemic. Digit. Health **8**, 1–15 (2022)
46. H.-C. Wu, Y.-C. Lin, T. Chen, Leisure agricultural park selection for traveler groups amid the Covid-19 pandemic. Agriculture **12**(1), 111 (2022)
47. T.-C.T. Chen, Type-II fuzzy collaborative intelligence for assessing cloud manufacturing technology applications. Robot. Comput. Integr. Manufact. **78**, 102399 (2022)
48. T. Chen, Assessing factors critical to smart technology applications to mobile health care—the fgm-fahp approach. Health Policy Technol. **9**, 194–203 (2020)
49. G. Zheng, N. Zhu, Z. Tian, Y. Chen, B. Sun, Application of a trapezoidal fuzzy AHP method for work safety evaluation and early warning rating of hot and humid environments. Saf. Sci. **50**(2), 228–239 (2012)
50. T.L. Saaty, Decision making with the analytic hierarchy process. Int. J. Serv. Sci. **1**(1), 83–98 (2008)
51. T. Chen, Evaluating the sustainability of a smart technology application to mobile health care—the FGM-ACO-FWA approach. Complex Intell. Syst. **6**, 109–121 (2020)
52. T. Chen, Y.C., Lin, M.C. Chiu, Approximating alpha-cut operations approach for effective and efficient fuzzy analytic hierarchy process analysis. Appl. Soft Comput. **85**, 105855 (2019)
53. T. Chen, Y.C. Lin, A fuzzy-neural system incorporating unequally important expert opinions for semiconductor yield forecasting. Int. J. Uncertain. Fuzziness Knowl. Based Syst. **16**(01), 35–58 (2008)
54. T. Chen, Y.-C. Wang, C.-W. Lin, A fuzzy collaborative forecasting approach considering experts' unequal levels of authority. Appl. Soft Comput. **94**, 106455 (2020)
55. F.R. Lima Junior, L. Osiro, L.C.R. Carpinetti, A comparison between Fuzzy AHP and Fuzzy TOPSIS methods to supplier selection. Appl. Soft Comput. **21**, 194–209 (2014)
56. J. Qin, X. Liu, W. Pedrycz, An extended VIKOR method based on prospect theory for multiple attribute decision making under interval type-2 fuzzy environment. Knowl. Based Syst. **86**, 116–130 (2015)
57. D. Huang, T. Chen, J.J. Wang, A fuzzy set approach for event tree analysis. Fuzzy Sets Syst. **118**, 153–165 (2001)
58. B. Sennaroglu, G.V. Celebi, A military airport location selection by AHP integrated PROMETHEE and VIKOR methods. Transp. Res. Part D: Transp. Environ. **59**, 160–173 (2018)
59. E. Van Broekhoven, B. De Baets, Fast and accurate center of gravity defuzzification of fuzzy system outputs defined on trapezoidal fuzzy partitions. Fuzzy Sets Syst. **157**(7), 904–918 (2006)

Chapter 4
Applications of XAI to Job Sequencing and Scheduling in Manufacturing

Abstract This chapter discusses a new application field of XAI in manufacturing—job sequencing and scheduling. It first breaks down job sequencing and scheduling into several steps and then mentions AI technologies applicable to some of these steps. It is worth noting that many AI applications focus on the preparation of inputs required for scheduling tasks, rather than the process of scheduling tasks, which is a distinctive feature of the field. Nonetheless, many AI techniques have already been explained in other fields or domains. These explanations can provide a reference for explaining the application of AI in job sequencing and scheduling. Therefore, some general XAI techniques and tools for job sequencing and scheduling are reviewed, including: referring to the classification of job scheduling problems; customizing scheduling rules; text description, pseudocode; decision trees, flowcharts. Furthermore, job sequencing and scheduling problems are often formulated as mathematical programming (optimization) models to be optimized. AI technologies can be applied to find the best solution for the model. Applications of genetic algorithm (GA) are of particular interest because such applications are most common in job scheduling. Furthermore, XAI techniques and tools for explaining GA can be easily extended to other evolutionary AI applications, such as artificial bee colony (ABC), ant colony optimization (ACO), and particle swarm optimization (PSO) in job scheduling. Applicable XAI techniques and tools include flowcharts, text description, chromosome maps, dynamic line charts, and bar charts with baseline. Some novel XAI techniques and tools for interpreting GAs are also introduced: decision tree-based interpretation and dynamic transition and contribution diagrams.

Keywords XAI · Job sequencing and scheduling · GA · Decision tree-based interpretation · Dynamic transition and contribution diagrams

4.1 Job Sequencing and Scheduling in the Manufacturing Domain

No matter how advanced a manufacturing system is, job sequencing and scheduling is a basic but critical task of the manufacturing system. In manufacturing, job

T.-C.T. Chen, *Explainable Artificial Intelligence (XAI) in Manufacturing*,
SpringerBriefs in Applied Sciences and Technology,
https://doi.org/10.1007/978-3-031-27961-4_4

sequencing and scheduling is a topic that has attracted many applications of artificial intelligence (AI) [1]. AI technologies can be used to build a job scheduling system or prepare the inputs required by a job scheduling system [2, 3]. Furthermore, job scheduling is usually formulated as a mathematical programming problem [4]. AI technologies are then applied to find the optimal (or near-optimal) solution to the problem [5]. In addition, artificial neural network (ANN), genetic algorithm (GA), particle swarm optimization (PSO), artificial bee colony (ABC), ant colony optimization (ACO), harmony search, and other AI technologies can be applied to analyze scheduling results to adjust/optimize scheduling rules [6–8].

However, many people related to this task do not have background knowledge of AI technologies, which hinders their understanding, communication, and acceptance of the application of AI technologies [9]. At the same time, job scheduling methods incorporating more sophisticated AI technologies are being proposed. As a result, there is a gap between research and practice. The emergence of explainable artificial intelligence (XAI) promises to fill this gap [10, 11]. However,

- The scheduling plan generated by an AI application is easy to verify its advantages over current practice, reducing the need to explain the reasoning process of the AI application.
- A scheduling plan involves the operations of many jobs on multiple machines, and it is difficult to explain the scheduling result of an operation (or job) locally.
- There has been little research on this topic in the past. Most existing XAI methods focus on pattern recognition, defect analysis [12, 13], and estimation and prediction [14, 15].
- XAI applications are more prevalent in other domains than in manufacturing: In domains such as medicine, services, and banking, the results of AI applications affect human–machine interactions, and thus need better explanations [16–19]. In contrast, in manufacturing, the results of AI applications are related to jobs or machines and can be used directly [20].

For these reasons, this chapter aims to discuss the application of XAI to job sequencing and scheduling in manufacturing systems.

4.2 AI Applications in Job Sequencing and Scheduling

Job sequencing and scheduling in a manufacturing system is an iterative process that includes the following steps [21], and AI technologies can be applied to several of these steps, as shown in Fig. 4.1:

Step 1. Select performance measures: Common job sequencing performance measures include makespan ($C_{\max}$), maximum lateness ($L_{\max}$), total completion time ($\sum C_j$), total weighted completion time ($\sum w_j C_j$), total (weighted) tardiness ($\sum (w_j) T_j$), (weighted) number of tardy jobs $\sum (w_j) U_j$, , mean cycle time ($\overline{C}$), cycle time standard deviation (s_C), etc. [22].

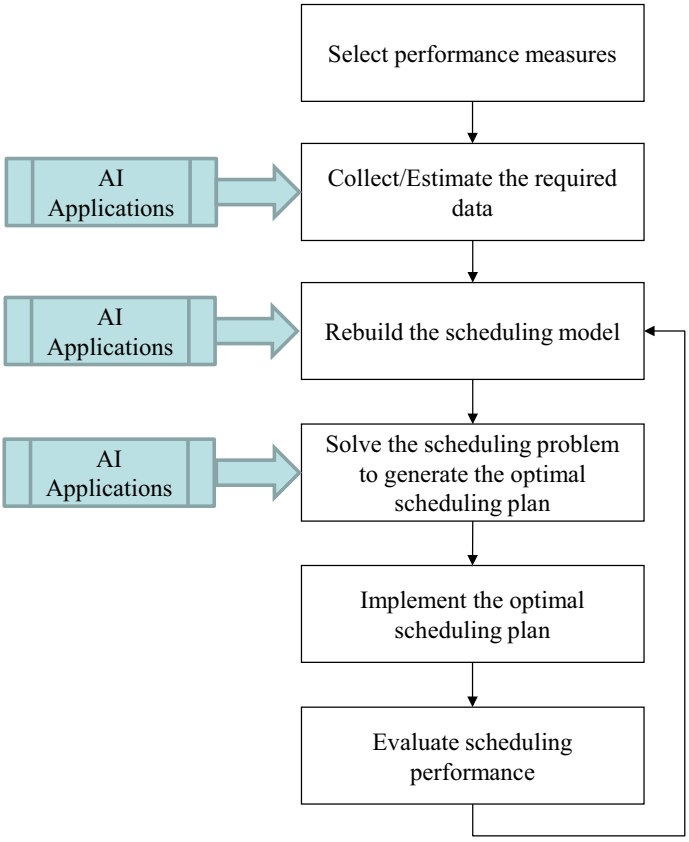


Fig. 4.1 Steps of job sequencing and scheduling

Step 2. Collect (or estimate) the required (input) data (see Table 4.1): Some data required for job scheduling can be extracted from the production plan and production management information systems (PROMIS) [6]. Other data need to be derived, estimated, or predicted, to which AI technologies are applicable [23].
Step 3. Build (or rebuild) the scheduling model: Job scheduling problems are usually solved by formulating and optimizing mathematical programming models [4] or by applying scheduling rules [6].
Step 4. Solve the scheduling problem to generate an optimal scheduling plan: AI technologies can be used to find optimal solutions to mathematical programming models, prepare inputs for dispatch rules, or optimize scheduling rules [24–27].
Step 5. Implement the optimal scheduling plan.
Step 6. Evaluate scheduling performance.
Step 7. Return to Step 3.

Table 4.1 Data required for job scheduling

Data type	Input data
Collected data	• Arrival time • Batch/Job size • Due date • Job no • Lateness penalty • Priority • Processing time • Product type • Release time • etc.
Derived data	• Bottleneck • Hourly capacity of the next machine • Minimum and maximum of raw/derived data • Next bottleneck • Number of visits • Processing time until the next bottleneck • Processing time until the next visit (by product type) • Remaining processing time (by product type) • Utilization of the next machine • WIP level in the queue of the next machine • etc.
Estimated/forecasted data	• Cycle time (by product type) • Minimum and maximum of estimated/forecasted data • Remaining cycle time (by product type) • etc.

4.3 Generic XAI Techniques and Tools for Job Sequencing and Scheduling

Many AI technologies have been explained in other domains or fields, as shown in Table 4.2. These explanations can provide a reference for explaining the application of AI applications in job scheduling. However, XAI applications for image analysis and pattern recognition [12, 13, 16, 17] are less useful for job scheduling and are therefore not included in this table. In addition,

- There are many types of AI technologies used in job scheduling, and existing explanations are limited to a few AI technologies (such as ANN, deep learning or deep neural networks (DNN), fuzzy analytic hierarchy process (FAHP), adaptive neuro-fuzzy inference systems (ANFIS), fuzzy Vise Kriterijumska Optimizacija I Kompromisno Resenje (VIKOR), k-means, eXtreme gradient boosting (XGBoost), recurrent neural network (RNN), etc.) [12–17]. The applications of other AI technologies (such as GA, ACO, PSO, ABC, etc.) are rarely explained.
- Existing explanations do not cover all steps of scheduling.

Table 4.2 Explained AI applications in other domains or fields and their references for job scheduling

Explained AI technology	References	XAI techniques	Applications	Reference for job scheduling
ANFIS	Aghamohammadi et al. [28]	• Textual description • Bar chart • Line chart • Performance evaluation	• Heart attack prediction	• Input forecasting
ANN	Chen and Wang [15], Kong et al. [29]	• Textual description • Shaply additive explanation value (SHAP) • Random forest (RF)	• Cycle time prediction • Alloy component effect evaluation	• Input forecasting • Job attribute prioritization
DNN	Akhlaghi et al. [30]	• Textual description • SHAP	• Dew point cooler feature comparison	• Input forecasting • Job attribute prioritization
FAHP	Chen and Wang [15], Chen and Chiu [19]	• Textual description • Gradient bar chart • Grouped bar chart • Traceable aggregation	• Factor prioritization • Travel destination selection	• Job attribute prioritization • Job sequencing
Fuzzy VIKOR	Chen and Wang [15]	• Textual description • Segmented distance diagram	• Travel destination selection	• Job selection • Multi-objective job scheduling
k-means	Chen and Wang [15]	• Textual description • Radar chart • Radar scatter diagram	• Job classification	• Job classification • Job prioritization
RNN	Panigutti et al. [31]	• Textual description • Decision trees	• Patient classification	• Job classification • Job prioritization
XGBoost	Liu and Liu [26]	• Textual description • SHAP	• Reservoir feature effect analysis	• Input forecasting • Job attribute prioritization

In estimation and prediction, complex AI applications are simulated/explained using simpler XAI techniques, and the average accuracy of the latter when simulating the former can be used to evaluate the effectiveness of XAI techniques [15]. In job scheduling, there is only one optimal solution (i.e., the optimal scheduling plan), and it is impossible to evaluate the effectiveness of an XAI technique in the same way. Alternatively, XAI techniques for interpreting AI applications in job scheduling are considered effective if the following requirements [18, 19] are met:

- Personnel responsible for job scheduling have the required background knowledge.
- A XAI technique can process high-dimensional scheduling data.
- A XAI technique is convenient for comparing the performances of various scheduling methods.
- Explanation formats are consistent in different applications.
- A XAI technique is easy to communicate.
- A XAI technique is simple and easy to understand.
- A XAI technique can visualize/compare feasible solutions.
- A XAI technique can visualize/track the evolution process.

4.3.1 Referring to the Taxonomy of Job Scheduling Problems

Referring a job scheduling problem to the taxonomy of job scheduling problems is the first way to help understand why existing scheduling rules perform well (or poorly).

A job scheduling problem can be described by the three-field notation $\alpha/\beta/\gamma$ where

- α: machine conditions;
- β: job characteristics;
- γ: scheduling performance (usually to be minimized).

Therefore, when dealing with a job scheduling problem, the job scheduling problem should be classified first. Once the type of the job scheduling problem is determined (i.e., the values of α, β, and γ are determined):

- Existing scheduling methods suitable for this type can be applied.
- The scheduler may further develop a more suitable and effective scheduling method based on these basic scheduling methods.

In other words, when explaining a new scheduling method, it is useful to refer to the taxonomy of job scheduling problems and to compare the new scheduling method with existing scheduling methods for this type of job scheduling problems.

4.3.2 Customizing Dispatching Rule

Dispatching (scheduling) rules are perhaps the most widely applied job sequencing and scheduling methods. Table 4.3 summarizes some common dispatching rules. Dispatch rules in the form of linear regression, polynomials, or simple inverse functions are easily understandable, so basically no more explanation is needed.

However, some dispatch rules are based on parameters that need to be predicted/estimated, such as the value of RCT_j in FSVCT. In this case, the explanation focuses on how to predict/estimate these parameters. If ANN or deep learning is applied, the XAI techniques introduced in Chap. 2 are applicable.

Researchers are constantly trying to propose new dispatching rules for different scheduling problems because they are easy to understand, communicate, and apply. For example, to simultaneously optimize two scheduling performance measures

Table 4.3 Some understandable dispatching rules for job scheduling

Dispatching rule	Abbreviation	Job priority evaluation formula[a]
First in first out	FIFO	r_j
Critical ratio	CR	$\frac{d_j - t}{\mathrm{RPT}_j}$
Cyclic scheduling	CYCLIC	(1) $\frac{1}{n_j \mod k}$ (2) FIFO
Earliest due date	EDD	d_j
First in first out plus	FIFO+	(1) NQ_j + FIFO (2) FIFO
Fluctuation smoothing policy for mean cycle time	FSMCT	$j/\lambda - \mathrm{RCT}_j$
Fluctuation smoothing policy for cycle time variation	FSVCT	$R_j - \mathrm{RCT}_j$
Longest processing time until the next visit	LTNV	(1) $1/\mathrm{PTN}_j$ (2) FIFO
Least amount of WIP in the next machine queue	LWNQ	NQ_j
Shortest processing time	SPT	p_j
Shortest remaining processing time	SRPT	RPT_j
Shortest remaining processing time plus	SRPT+	(1) RPT_j (2) NQ_j
Shortest processing time until the next visit	STNV	(1) $1/\mathrm{PTN}_j$ (2) FIFO

[a]Priority is the smaller the higher

j job no.; d_j due date; k cycle length; n_j number of visits to the current machine; p_j processing time; PTN_j processing time until the next visit; r_j arrival time; R_j release time; RCT_j remaining cycle time; RPT_j remaining processing time; t time; NQ_j the queue length of the next machine; λ release rate

(cycle time average and variation), Chen [6] proposes a new scheduling rule that combines the FSVCT rule and the FSMCT rule as

$$\mathrm{SK}_j = \left(\frac{\mathrm{SK}_j(\mathrm{FSMCT}) - \min\limits_l \mathrm{SK}_l(\mathrm{FSMCT})}{\max\limits_l \mathrm{SK}_l(\mathrm{FSMCT}) - \max\limits_l \mathrm{SK}_l(\mathrm{FSMCT})} \right)^{\alpha} \cdot \left(\frac{\mathrm{SK}_j(\mathrm{FSVCT}) - \min\limits_l \mathrm{SK}_l(\mathrm{FSVCT})}{\max\limits_l \mathrm{SK}_l(\mathrm{FSVCT}) - \max\limits_l \mathrm{SK}_l(\mathrm{FSVCT})} \right)^{1-\alpha} \tag{4.1}$$

where $\mathrm{SK}_j(\mathrm{FSMCT})$ and $\mathrm{SK}_j(\mathrm{FSVCT})$ are slack values of job j as defined in Table 4.3; $\alpha \in [0, 1]$. The design idea of the new dispatching rule is "when one measure is stronger, the other measure becomes weaker".

Another way to propose new scheduling rules is to use big data analytics to customize existing scheduling rules for the target manufacturing system. For example, Chen [6] tailored FSVCT rules for specific wafer fabs. He first proposed the nonlinear FSVCT rule as

$$\begin{aligned}
\mathrm{SK}_j &= \frac{N(R_j)}{N(\mathrm{RCT}_j)} \\
&= \frac{(R_j - \min\limits_k R_k)/(\max\limits_k R_k - \min\limits_k R_k)}{(\mathrm{RCT}_j - \min\limits_k \mathrm{RCT}_k)/(\max\limits_k \mathrm{RCT}_k - \min\limits_k \mathrm{RCT}_k)} \\
&= \frac{b}{a} \cdot \frac{R_j - \min\limits_k R_k}{\mathrm{RCT}_j - \min\limits_k \mathrm{RCT}_k} \\
&= \frac{b}{a} \cdot \frac{R_j - \min\limits_k R_k - \mathrm{RCT}_j + \mathrm{RCT}_j}{\mathrm{RCT}_j - \min\limits_k \mathrm{RCT}_k} \\
&= \frac{b}{a(\mathrm{RCT}_j - \min\limits_k \mathrm{RCT}_k)} \cdot (R_j - \mathrm{RCT}_j + \mathrm{RCT}_j - \min_k R_k) \\
&= \left(\frac{b(\mathrm{RCT}_j - \min\limits_k \mathrm{RCT}_k)}{a} \right)^{-1} \cdot (R_j - \mathrm{RCT}_j + (\mathrm{RCT}_j - \min_k R_k)^1)
\end{aligned} \tag{4.2}$$

where $N()$ is the normalization function; $a = \max\limits_k R_k - \min\limits_k R_k$; $b = \max\limits_k \mathrm{RCT}_k - \min\limits_k \mathrm{RCT}_k$.

The FSVCT rule can be re-written as

$$\begin{aligned}
\mathrm{SK}_j &= R_j - \mathrm{RCT}_j \\
&= \left(\frac{a(\mathrm{RCT}_j - \min\limits_k \mathrm{RCT}_k)}{b} \right)^{-0} \cdot (R_j - \mathrm{RCT}_j + (\mathrm{RCT}_j - \min_k R_k)^0)
\end{aligned} \tag{4.3}$$

These two formulas can be generalized as.

(Generalized FSVCT rule)

$$\mathrm{SK}_j = \left(\frac{a(\mathrm{RCT}_j - \min\limits_k \mathrm{RCT}_k)}{b} \right)^{-\xi} \cdot (R_j - \mathrm{RCT}_j + (\mathrm{RCT}_j - \min\limits_k R_k)^{\zeta}) \quad (4.4)$$

where ξ and ζ are positive real numbers satisfying the following constraints:

- $\xi = 0 \leftrightarrow \zeta = 0$
- $\xi = 1 \leftrightarrow \zeta = 1$

There are many possible models for choosing the values of ξ and ζ in Eq. (4.4). For example,

- Linear model: $\xi = \zeta$
- Nonlinear model: $\xi = \zeta^k, k \geq 0$
- Logarithmic model: $\xi = \ln(1 + \zeta) / \ln 2$.

The generalized FSVCT rule can be tailored to a specific wafer fab by selecting appropriate values of ξ and ζ using big data analytics.

Example 4.1 The most appropriate generalized FSVCT rule will be found to optimize the job scheduling performance in a fab with respect to cycle time standard deviation (σ_{CT_j}). For this purpose, 50 combinations of ξ and ζ were generated according to the logarithmic model. Then, production simulation is used to assess the effects of these combinations. The results are summarized in Table 4.4 [6].

Subsequently, a feed-forward neural network (FNN) is built to make predictions based on the values of ξ and ζ. The required MATLAB code is shown in Fig. 4.2. The predicted results are shown in Fig. 4.3.

To derive optimal values for both parameters, the range of ζ is divided into 1000 equal intervals. The value of ξ is derived from the logarithmic model. 1000 combinations are fed into the trained FNN for estimation. The combination that gives the smallest value of σ_{CT_j} is chosen:

$$\xi^* = 0.0128; \zeta^* = 0.0089; \sigma^*_{\mathrm{CT}_j} = 36\,\mathrm{h}$$

Namely, the most appropriate generalized FSVCT rule for the fab is

Table 4.4 Scheduling performances using various combinations of ξ and ζ

Combination #	ξ	ζ	Cycle time variation (h)
1	1.12	1.18	116
2	2.21	3.63	178
…			
50	2.74	5.7	353

```
GFSVCT_para=[1.12 2.21 … 2.74; 1.18 3.63 … 5.7];
ct_stdev=[116 178 … 353];
net=feedforwardnet(6);
net.dividefcn='dividetrain';
net.trainParam.lr=0.1;
net.trainParam.epochs=50000;
net.trainParam.goal=1;
net=train(net, GFSVCT_para, ct_stdev);
estimated_ct_stdev=net(GFSVCT_para);
```

Fig. 4.2 MATLAB code

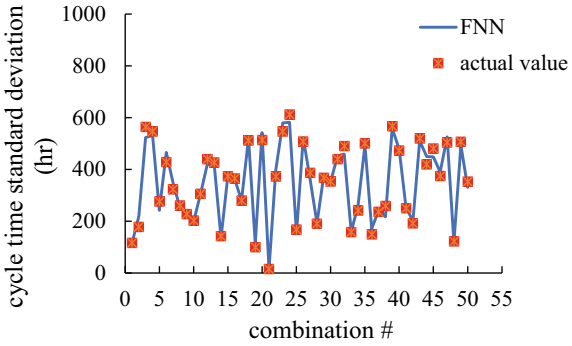
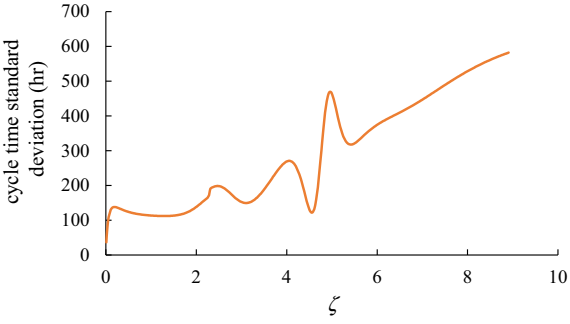


Fig. 4.3 Predicted scheduling performances

$$\mathrm{SK}_j = \left(\frac{a(\mathrm{RCT}_j - \min_k \mathrm{RCT}_k)}{b}\right)^{-0.0128} \cdot (R_j - \mathrm{RCT}_j + (\mathrm{RCT}_j - \min_k R_k)^{0.0089}) \tag{4.5}$$

The response surface method is not suitable for analyzing the effects of the two parameters on σ_{CT_j}, because the value of ξ depends on the value of ζ. In contrast, the effect of the value of ζ on σ_{CT_j} is shown in Fig. 4.4.

Fig. 4.4 Effect of the value of ζ on σ_{CT_j}



```
set r_min to a large value
set j_min to n+1 -> null job
for i=1 to n
   if r_i is smaller than r_min
      update r_min to r_i
      update j_min to j_i
   end
end
```

Fig. 4.5 Pseudocode of FIFO

4.3.3 Textual Description, Pseudocode

All job scheduling studies use the textual description technique [6] to explain the ideas behind scheduling methods. Compared with the formulas full of mathematical symbols in the previous section, textual descriptions are obviously easier to understand and communicate.

The following uses the dynamic bottleneck detection (DBD) method proposed by Zhang et al. [32] as an example. In DBD, different heuristics are used to sequence jobs before non-bottleneck and bottleneck workstations. First, jobs are divided into four categories. Then, jobs in these categories are sequenced using different dispatching rules [18]:

(1) First-priority category: CR is used to sequence jobs in this category first. Then, FIFO is applied to break possible ties (i.e., jobs with the same CRs).
(2) Second-priority category: The shortest processing time until the next bottleneck (SPNB) is applied to sequential jobs in this category. CR and FIFO are used in turn to break possible ties.
(3) Third-priority category: This category applies SPT, CR, and FIFO in turn.
(4) Fourth-priority category: CR and FIFO are used for this category.

Pseudocode is similar to textual descriptions in which the same ideas are presented informally, semi-colloquially, using a programming language. For system developers, pseudocode facilitates the coding of a scheduling method. However, for other stakeholders, pseudocode may not be easy to understand. Figure 4.5 provides a pseudocode example of FIFO.

4.3.4 Decision Tree, Flowchart

Some dispatching rules are compound because they contain multiple simple dispatching rules. For example, STNV consists of two simple rules: One selects the job with the shortest processing time until the next visit; the other is first in, first out. These simple dispatching rules are applied sequentially subject to different conditions being met, which can be better explained using a decision tree (see Fig. 4.6).

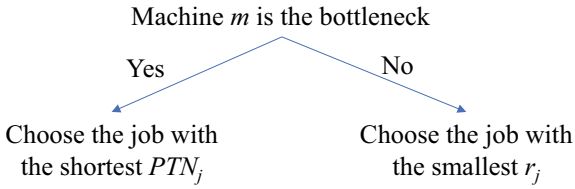


Fig. 4.6 Decision tree for explaining STNV

The second example is DBD, which consists of up to three simple dispatching rules per job category [33]. A decision tree is also built to explain (clarify) the sequencing mechanism of DBD in Fig. 4.7.

In many scheduling studies, flowcharts are drawn to explain the implementation/reasoning process of scheduling methods. A decision tree is a special flowchart with many conditional judgments. Figure 4.8 is a flowchart for explaining STNV.

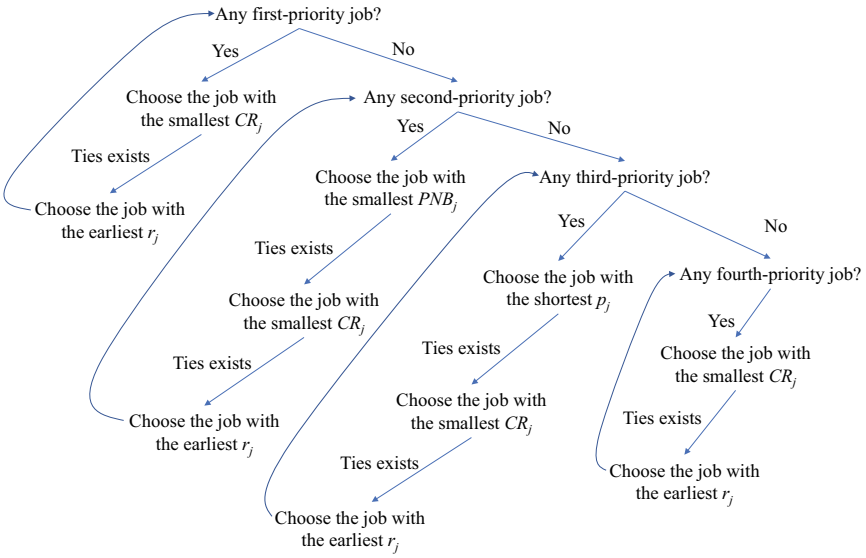


Fig. 4.7 Decision tree for explaining DBD

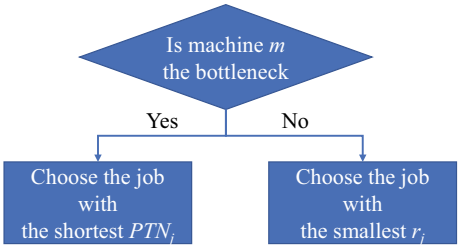


Fig. 4.8 Flowchart for explaining STNV

4.4 XAI Techniques and Tools for Genetic Algorithm Applications

Sequencing and scheduling of jobs is not difficult. The scheduler can arbitrarily determine the order of jobs and then schedule their start and finish times, just as is done in many traditional manufacturing systems. However, generating scheduling plans that optimize specific criteria is a challenging task. For this reason, job sequencing and scheduling problems are often formulated as mathematical programming (optimization) models to be optimized. AI technologies can be applied to find the optimal solutions to the models [34–41]. Applications of GA are of particular interest because such applications are most common in job scheduling [42]. Moreover, XAI techniques and tools for explaining GA can be easily extended to account for other evolutionary AI applications such as ACO, PSO, and ABC applications in job scheduling.

4.4.1 Flowchart, Textual Description

GA helps explain the process of solving complex job sequencing and scheduling problems by describing how to improve the feasible solution [43]. In other words, GA describes the evolution of feasible solutions to optimal solutions for mathematical programming problems of job sequencing and scheduling [44]. GAs have been widely used in job scheduling [24, 29, 45–47].

Figure 4.9 [6] is a flowchart used to illustrate the process of implementing GA.

Other generic XAI techniques have been applied to explain GA applications in job scheduling. For example, a scheduling system (method) including a GA application can be illustrated with a system architecture diagram [48]. Furthermore, textual descriptions are used to explain the reasoning mechanism of GA [2], which is supported by pseudocode [2, 49].

4.4.2 Chromosomal Diagram

Chromosomal diagrams (or graphs) are special XAI tools used to illustrate chromosomal encoding (i.e., permutation representation), crossovers, and mutations [2, 49], in which solutions to a job sequencing and scheduling problem are represented as bit strings called chromosomes (or individuals) and then evolved. Figure 4.10 gives an example showing the sequence of processing five jobs to minimize the makespan $C_{\max}$. In this example, the sequence of processing jobs is job #1 → job #3 → job #5 → job #2 → job #4, which is a feasible solution to the scheduling problem.

Different job scheduling problems require different encodings, and some may be complex.

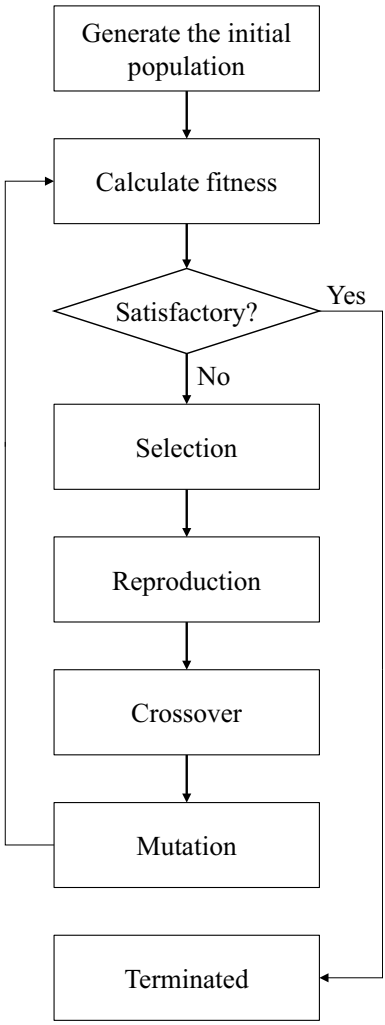


Fig. 4.9 Process of implementing GA

#1	#3	#5	#2	#4

Fig. 4.10 Sequence of processing jobs as a chromosome (permutation representation)

Example 4.2 The case discussed by Pezzella et al. [50] is used as an example, where the operations of three jobs on four machines were scheduled. The manufacturing system is a flexible job shop where the numbers of operations for different jobs are not equal. Different jobs on the same machine have different processing times too. Furthermore, each operation can be performed on different machines with unequal processing times. Optimizing the makespan of such manufacturing systems has been

(1, 1, 3)	(1, 2, 1)	(2, 1, 3)	(2, 2, 4)	(3, 1, 3)	(1, 3, 2)	(3, 2, 1)	(2, 3, 4)

Fig. 4.11 Array string for representing a chromosome

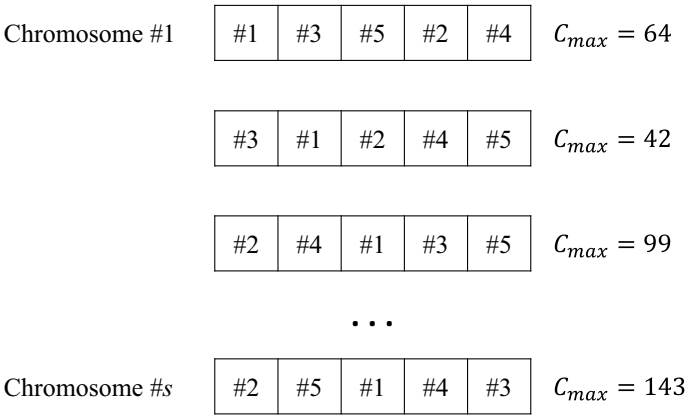


Fig. 4.12 Initial population (first generation)

shown to be an NP-hard problem [51]. Theoretically, the number of possible permutations for this problem is at most $4^{3+3+2} \cdot 8! = 2.64 \times 10^9$. For this reason, GA is applied to help solve the job scheduling problem.

Each chromosome (i.e., job permutation or feasible solution) is represented as an array string, as shown in Fig. 4.11, where the array (a, b, c) indicates that the bth operation of job a is performed on machine c. For example, the rightmost array (2, 3, 4) means that the 3rd operation of job #2 is executed on machine #4.

In the beginning, an initial population is created consisting of s individuals, called the first generation, as shown in Fig. 4.12. The fitness (scheduling performance) of each individual is also evaluated.

Individuals (parents) with better fitness are chosen to generate their offspring. Other selection methods include random selection and tournament selection.

Individuals are crossed over and mutated to create the next generation, as shown in Fig. 4.13. The crossover point "X" is chosen randomly. The jobs of the two parents following the crossover point are swapped with a given probability. However, this can cause the same job to appear twice, which is then resolved by randomly replacing the job with a non-occurring one. Furthermore, jobs to be mutated are randomly selected, and these jobs have a certain chance of exchanging their positions.

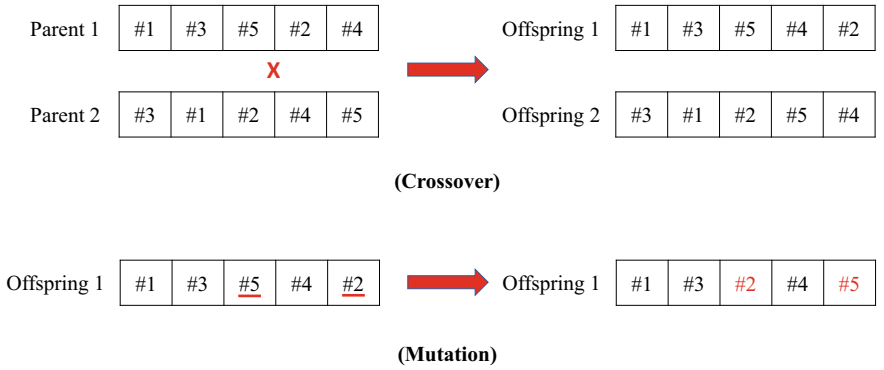


Fig. 4.13 Crossover and mutation

4.4.3 Dynamic Line Chart, Bar Chart with Baseline

During the evolution, a dynamic line chart can be used to track the convergence of the scheduling performance [50] (see Fig. 4.14), where the best and average performances of each generation are monitored. After optimization, a bar chart with baseline can be plotted to compare the scheduling performance using GA with other scheduling methods (see Fig. 4.15) [33, 52, 53].

Existing XAI techniques for explaining the application of GA in job scheduling face the following problems:

- In the application of GA for job scheduling, the inputs of GA are usually feasible solutions rather than job attributes, and the output is the optimal solution [49]. However, there is no direct relationship between feasible and optimal solutions, i.e., there is no intuitive reasoning from precedent [54], because if the initial feasible solutions are different, the optimal solution is still the same. Therefore, popular methods for fitting input–output relationships, such as decision trees [31], classification and regression trees (CART) [15], RFs [15], fuzzy inference rules

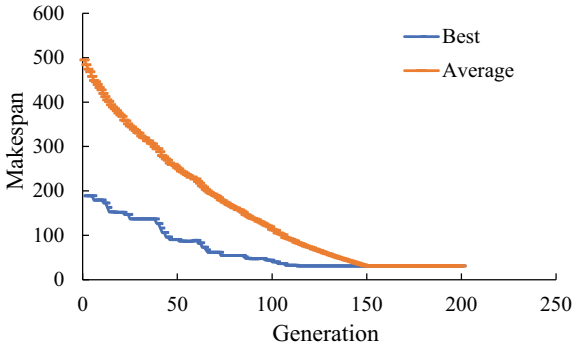


Fig. 4.14 Dynamic line chart for tracking the convergence of the scheduling performance

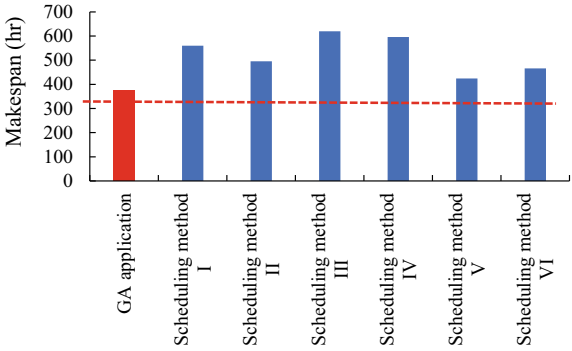


Fig. 4.15 Bar chart with baseline for comparing the scheduling performances of various scheduling methods

(or systems), SHAP [29], local interpretable model-agnostic explanation (LIME) [55], etc. are not suitable for explaining the application of GA in job scheduling.

- In addition, most of the popular input–output relationship fitting methods are difficult to handle a large number of inputs [56], while GA has many inputs (i.e., feasible solutions) [57].
- Furthermore, in previous studies, chromosome maps are usually used to analyze the evolution of feasible solutions. How these changes lead to the optimal solution has not yet been visualized.

4.5 Other XAI Techniques and Tools for Explaining GA

4.5.1 Decision Tree-Based Interpretation

It is interesting to explain the operations in GA by applying other XAI techniques such as decision trees. This is a meaningful attempt, since decision trees have been used to explain the application of AI to other types of problems such as estimation, prediction, and pattern recognition.

In GA, the selection mechanism can be explained by a decision tree, as shown in Fig. 4.16. $C_{\max,q}$ is the makespan (or fitness) associated with chromosome q; $q = 1$–Q. The set of candidate chromosomes is $\{\mathbf{x}_s\}$; the summary of selection results is $\{\mathbf{x}_p\}$. i is the index of a gene. Decision trees are a highly interpretable XAI technique and thus are applied [9, 10]. Other selection mechanisms can be explained similarly.

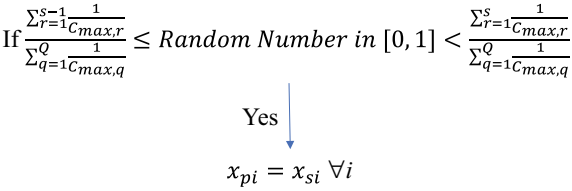


Fig. 4.16 Decision tree for explaining the selection mechanism

The crossover operation shown in Fig. 4.13 can also be interpreted in terms of decision trees, as shown in Fig. 4.17:

- p and p' are the two parental chromosomes to be crossed over, and o and o' represent the two offspring chromosomes.
- $\mathbf{x}_o$ indicates the oth offspring chromosome.

In this example, for most genes, only one level of the decision tree is required. Decision trees for other genes have more levels. Other crossover mechanisms can be explained similarly.

Subsequently, the mutation operation in Fig. 4.13 is explained using another decision tree, as shown in Fig. 4.18, where the genes at positions k and l are swapped. Other mutational mechanisms can be explained similarly.

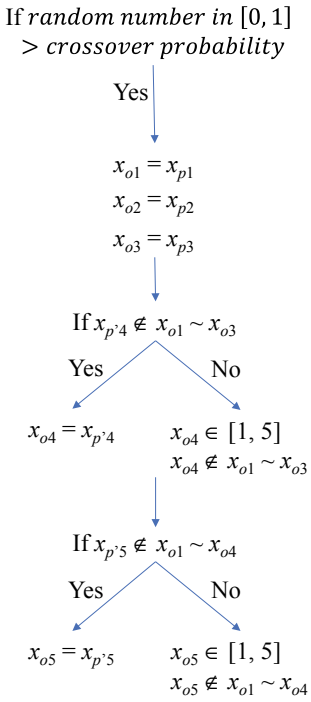


Fig. 4.17 Decision tree for explaining the crossover mechanism

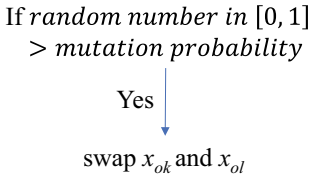


Fig. 4.18 Decision tree for explaining the mutation mechanism

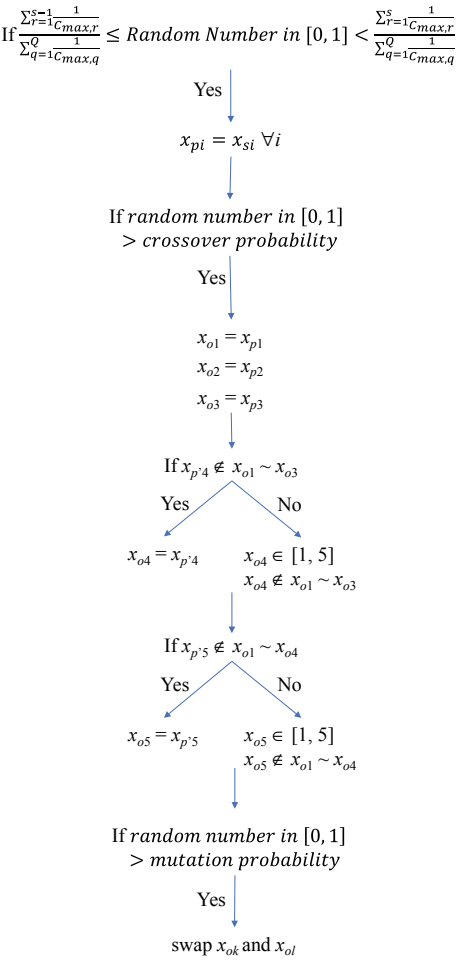


Fig. 4.19 Decision tree for explaining the complete process of generating offspring chromosomes

These two diagrams can be combined to explain the complete process of generating offspring chromosomes. Figure 4.19 gives an example.

4.5.2 Dynamic Transition and Contribution Diagram

Furthermore, in order to visualize how the change of feasible solutions leads to the optimal solution, this section borrows the concept of scatter radar diagrams [15] and proposes dynamic transition and contribution diagrams. In a dynamic transition and contribution diagram,

- The data of all feasible solutions (i.e., chromosomes) of each population and the optimal solution are presented in the form of radar diagrams.
- The optimal solution is placed in the center.
- If a population contains numerous feasible solutions, only the feasible solutions that are closer to the optimal solution will be liberated around the optimal solution.
- Directional arrows link each feasible solution to the optimal solution.
- The thicker the arrow, the closer the feasible solution is to the optimal solution. In other words, the feasible solution contributes more to the optimal solution.

The dynamic transition and contribution diagram at the end of the evolutionary process is shown in Fig. 4.20. According to Fig. 4.20,

- The optimal solution is job #5 → job #2 → job #7 → job #1 → job #4 → job #3 → job #6.
- The top five chromosomes of each generation are shown.

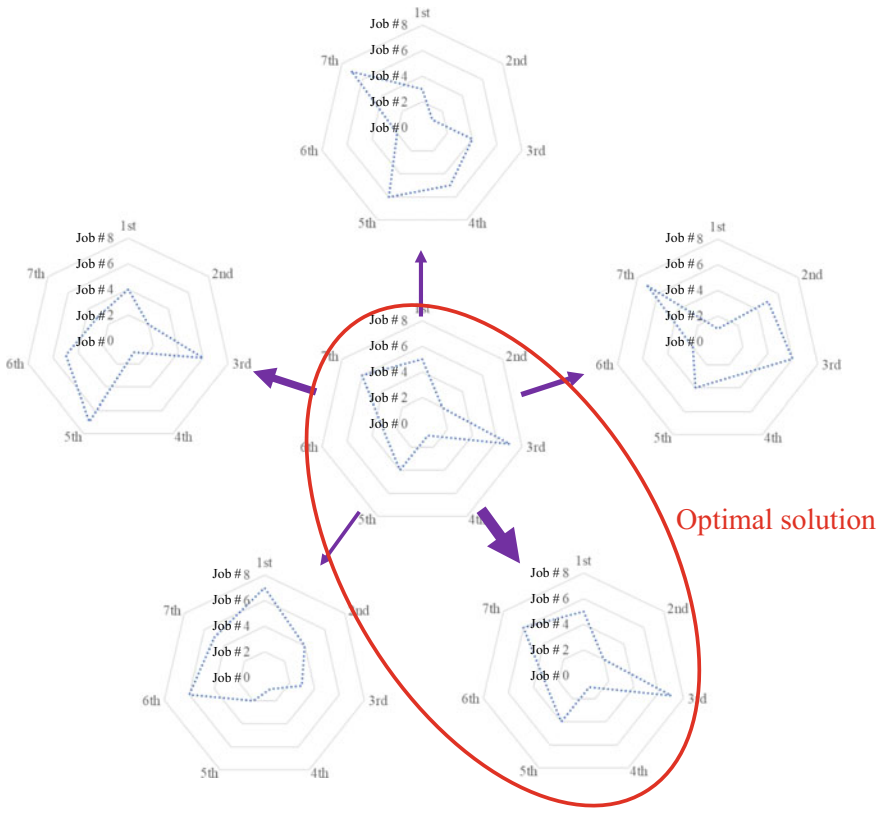


Fig. 4.20 A dynamic transition and contribution diagram

References

1. H. Wang, Flexible flow shop scheduling: optimum, heuristics and artificial intelligence solutions. Expert. Syst. **22**(2), 78–85 (2005)
2. A. Seker, S. Erol, R. Botsali, A neuro-fuzzy model for a new hybrid integrated process planning and scheduling system. Expert Syst. Appl. **40**(13), 5341–5351 (2013)
3. T. Chen, Y.C. Wang, A fuzzy-neural approach for supporting three-objective job scheduling in a wafer fabrication factory. Neural Comput. Appl. **23**(1), 353–367 (2013)
4. G. El Khayat, A. Langevin, D. Riopel, Integrated production and material handling scheduling using mathematical programming and constraint programming. Eur. J. Oper. Res. **175**(3), 1818–1832 (2006)
5. E.G. Talbi, Combining metaheuristics with mathematical programming, constraint programming and machine learning. Ann. Oper. Res. **240**(1), 171–215 (2016)
6. T.C.T. Chen, Job sequencing and scheduling, in *Production Planning and Control in Semiconductor Manufacturing: Big Data Analytics and Industry 4.0 Applications* (2020), pp. 77–99
7. Z.W. Geem, Optimal scheduling of multiple dam system using harmony search algorithm, in *International Work-Conference on Artificial Neural Networks* (2007), pp. 316–323
8. X. Yuan, L. Wang, Y. Yuan, Application of enhanced PSO approach to optimal scheduling of hydro system. Energy Convers. Manage. **49**(11), 2966–2972 (2008)
9. D. Gunning, M. Stefik, J. Choi, T. Miller, S. Stumpf, G.Z. Yang, XAI—explainable artificial intelligence. Sci. Robot. **4**(37), eaay7120 (2019)
10. A. Das, P. Rad, Opportunities and challenges in explainable artificial intelligence (xai): a survey. arXiv preprint arXiv:2006.11371 (2020)
11. S. Meister, M. Wermes, J. Stüve, R.M. Groves, Investigations on explainable artificial Intelligence methods for the deep learning classification of fibre layup defect in the automated composite manufacturing. Compos. B Eng. **224**, 109160 (2021)
12. C. Ieracitano, N. Mammone, A. Paviglianiti, F.C. Morabito, Toward an augmented and explainable machine learning approach for classification of defective nanomaterial patches, in *International Conference on Engineering Applications of Neural Networks* (2021), pp. 244–255
13. L.C. Brito, G.A. Susto, J.N. Brito, M.A. Duarte, An explainable artificial intelligence approach for unsupervised fault detection and diagnosis in rotating machinery. Mech. Syst. Signal Process. **163**, 108105 (2022)
14. H. Nasiri, A. Homafar, S.C. Chelgani, Prediction of uniaxial compressive strength and modulus of elasticity for Travertine samples using an explainable artificial intelligence. Results Geophys. Sci. **8**, 100034 (2021)
15. T. Chen, Y.C. Wang, A two-stage explainable artificial intelligence approach for classification-based job cycle time prediction. Int. J. Adv. Manuf. Technol. **123**(5), 2031–2042 (2022)
16. S. L'Yi, B. Ko, D. Shin, Y.J. Cho, J. Lee, B. Kim, J. Seo, XCluSim: a visual analytics tool for interactively comparing multiple clustering results of bioinformatics data. BMC Bioinf. **16**(11), 1–15 (2015)
17. J. Zhang, H. Wang, H. Zhu, Increase the classification and expression ability and visualize the decision through a novel deep neural network model for the diagnosis of glaucoma. Invest. Ophthalmol. Vis. Sci. **59**(9), 4079–4079 (2018)
18. Y.C. Lin, T.C.T. Chen, Type-II fuzzy approach with explainable artificial intelligence for nature-based leisure travel destination selection amid the COVID-19 pandemic. Digital Health **8**, 20552076221106320 (2022)
19. T. Chen, M.-C. Chiu, Evaluating the sustainability of a smart technology application in healthcare after the COVID-19 pandemic: a hybridizing subjective and objective fuzzy group decision-making approach with XAI. Digital Health **8**, 20552076221136380 (2022)
20. H. Na, J. Park, Multi-level job scheduling in a flexible job shop environment. Int. J. Prod. Res. **52**(13), 3877–3887 (2014)
21. L.P. Michael, *Scheduling: Theory, Algorithms, and Systems* (Springer, 2018)

22. R.K. Suresh, K.M. Mohanasundaram, Pareto archived simulated annealing for job shop scheduling with multiple objectives. Int. J. Adv. Manuf. Technol. **29**(1), 184–196 (2006)
23. T. Chen, Job remaining cycle time estimation with a post-classifying fuzzy-neural approach in a wafer fabrication plant: a simulation study. Proc. Inst. Mech. Eng. Part B J. Eng. Manuf. **223**(8), 1021–1031 (2009)
24. S.S. Sana, H. Ospina-Mateus, F.G. Arrieta, J.A. Chedid, Application of genetic algorithm to job scheduling under ergonomic constraints in manufacturing industry. J. Ambient. Intell. Humaniz. Comput. **10**(5), 2063–2090 (2019)
25. R.G. Babukartik, P. Dhavachelvan, Hybrid algorithm using the advantage of ACO and Cuckoo Search for job scheduling. Int. J. Inf. Technol. Convergence Serv. **2**(4), 25 (2012)
26. J.J. Liu, J.C. Liu, Permeability predictions for tight sandstone reservoir using explainable machine learning and particle swarm optimization. Geofluids **2022**, 2263329 (2022)
27. D. Thiruvady, A.T. Ernst, G. Singh, Parallel ant colony optimization for resource constrained job scheduling. Ann. Oper. Res. **242**(2), 355–372 (2016)
28. M. Aghamohammadi, M. Madan, J.K. Hong, I. Watson, Predicting heart attack through explainable artificial intelligence, in *International Conference on Computational Science* (2019), pp. 633–645
29. B.O. Kong, M.S. Kim, B.H. Kim, J.H. Lee, Prediction of creep life using an explainable artificial intelligence technique and alloy design based on the genetic algorithm in creep-strength-enhanced ferritic 9% Cr steel. Metals Mater. Int. 1–12 (2022)
30. G. Akhlaghi, K. YAslansefat, X. Zhao, S. Sadati, A. Badiei, X. Xiao, S. Shittu, Y. Fan, X. Ma, Hourly performance forecast of a dew point cooler using explainable Artificial Intelligence and evolutionary optimisations by 2050. Appl. Energy **281**, 116062 (2021)
31. C. Panigutti, A. Perotti, D. Pedreschi, Doctor XAI: an ontology-based approach to black-box sequential data classification explanations, in *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency* (2020), pp. 629–639
32. H. Zhang, Z. Jiang, C. Guo, Simulation-based optimization of dispatching rules for semiconductor wafer fabrication system scheduling by the response surface methodology. Int. J. Adv. Manuf. Technol. **41**(1–2), 110–121 (2009)
33. T. Chen, A fuzzy-neural DBD approach for job scheduling in a wafer fabrication factory. Int. J. Innov. Comput. Inf. Control **8**(6), 4024–4044 (2012)
34. T. Chen, T. Wang, Enhancing scheduling performance for a wafer fabrication factory: the bi-objective slack-diversifying nonlinear fluctuation-smoothing rule. Comput. Intell. Neurosci. **13**, 13 (2012)
35. T. Chen, An optimized tailored nonlinear fluctuation smoothing rule for scheduling a semiconductor manufacturing factory. Comput. Ind. Eng. **58**, 317–325 (2010)
36. T. Chen, Y.-C. Wang, Y.-C. Lin, A bi-criteria four-factor fluctuation smoothing rule for scheduling jobs in a wafer fabrication factory. Int. J. Innov. Comput. Inf. Control **6**(10), 4289–4303 (2010)
37. H.-C. Wu, T. Chen, A fuzzy-neural ensemble and geometric rule fusion approach for scheduling a wafer fabrication factory. Math. Probl. Eng. **2013**, 956978 (2013)
38. T. Chen, Y.-C. Wang, A bi-criteria nonlinear fluctuation smoothing rule incorporating the SOM-FBPN remaining cycle time estimator for scheduling a wafer fab—a simulation study. Int. J. Adv. Manuf. Technol. **49**(5), 709–721 (2010)
39. T. Chen, Y.-C. Wang, Y.-C. Lin, A fuzzy-neural system for scheduling a wafer fabrication factory. Int. J. Innov. Comput. Inf. Control **6**(2), 687–700 (2010)
40. T. Chen, A tailored nonlinear fluctuation smoothing rule for semiconductor manufacturing factory scheduling. Proc. Inst. Mech. Eng. Part I J. Syst. Control Eng. **223**, 149–160 (2009)
41. T. Chen, Dynamic fuzzy-neural fluctuation smoothing rule for jobs scheduling in a wafer fabrication factory. Proc. Inst. Mech. Eng. Part I J. Syst. Control Eng. **223**, 1081–1094 (2009)
42. T. Chen, C.-W. Lin, Smart and automation technologies for ensuring the long-term operation of a factory amid the COVID-19 pandemic: an evolving fuzzy assessment approach. Int. J. Adv. Manuf. Technol. **111**, 3545–3558 (2020)

43. B.M. Baker, M.A. Ayechew, A genetic algorithm for the vehicle routing problem. Comput. Oper. Res. **30**(5), 787–800 (2003)
44. Y.-C. Wang, T.-C.T. Chen, M.-C. Chiu, An explainable deep-learning approach for job cycle time prediction. Decis. Anal. **6**, 100153 (2023)
45. B. Skinner, S. Yuan, S. Huang, D. Liu, B. Cai, G. Dissanayake, H. Lau, A. Bott, D. Pagac, Optimisation for job scheduling at automated container terminals using genetic algorithm. Comput. Ind. Eng. **64**(1), 511–523 (2013)
46. C. Shen, L. Wang, Q. Li, Optimization of injection molding process parameters using combination of artificial neural network and genetic algorithm method. J. Mater. Process. Technol. **183**(2–3), 412–418 (2007)
47. J. Xia, Y. Yan, L. Ji, Research on control strategy and policy optimal scheduling based on an improved genetic algorithm. Neural Comput. Appl. **34**(12), 9485–9497 (2022)
48. T. Chen, A self-adaptive agent-based fuzzy-neural scheduling system for a wafer fabrication factory. Expert Syst. Appl. **38**(6), 7158–7168 (2011)
49. C.R. Reeves, A genetic algorithm for flowshop sequencing. Comput. Oper. Res. **22**(1), 5–13 (1995)
50. F. Pezzella, G. Morganti, G. Ciaschetti, A genetic algorithm for the flexible job-shop scheduling problem. Comput. Oper. Res. **35**(10), 3202–3212 (2008)
51. Y. Demir, S.K. İşleyen, Evaluation of mathematical models for flexible job-shop scheduling problems. Appl. Math. Model. **37**(3), 977–988 (2013)
52. T. Chen, An effective dispatching rule for bi-objective job scheduling in a wafer fabrication factory—considering the average cycle time and the maximum lateness. Int. J. Adv. Manuf. Technol. **67**(5–8), 1281–1295 (2013)
53. T. Chen, Intelligent scheduling approaches for a wafer fabrication factory. J. Intell. Manuf. **23**(3), 897–911 (2012)
54. E.M. Kenny, M.T. Keane, Twin-systems to explain artificial neural networks using case-based reasoning: comparative tests of feature-weighting methods in ANN-CBR twins for XAI, in *Twenty-Eighth International Joint Conferences on Artificial Intelligence* (2019), pp. 2708–2715
55. M.T. Ribeiro, S. Singh, C. Guestrin, "Why should i trust you?" Explaining the predictions of any classifier, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016), pp. 1135–1144
56. O. Abedinia, N. Amjady, H. Zareipour, A new feature selection technique for load and price forecast of electrical power systems. IEEE Trans. Power Syst. **32**(1), 62–74 (2016)
57. S. Zhang, C. Wang, A. Zomaya, Multi-level explanation of deep reinforcement learning-based scheduling. arXiv preprint arXiv:2209.09645 (2022)