On graph models in knowledge engineering

Antoni Ligęza[1], Weronika T. Adrian[1], Marek Adrian[1], Krzysztof Kluza[1], Krystian Jobczyk[1], Piotr Wiśniewski[1], Mateusz Ślażyński[1], Paweł Jemioło[1], Dominik Sepioło[1], Bernadetta Stachura-Terlecka[1], Mateusz Zaremba[1], Mateusz Szymkowski[1], Anna Suchenia[2], Tomasz Potempa[3]

[1] AGH University of Krakow, Faculty of Electrical Engineering, Automatics, Computer Science and Biomedical Engineering, Krakow
[2] Cracow University of Technology, Faculty of Electrical and Computer Engineering, Krakow
[3] University of Applied Sciences, Polytechnic Faculty in Tarnow, Tarnów

Abstract: The paper presents selected applications of graph models in knowledge engineering. Starting from the basic definition of a graph as a set of nodes connected by edges, the article presents possible extensions of this concept aimed at increasing the power of expression and the ability to process knowledge. In particular, the work focuses on selected applications of graph models in research areas explored by the members of the KRaKEn research team.

Keywords: graph, graph models, multigraphs, hypergraphs, causal graphs, knowledge graphs, BPMN, DMN, constraint representation graphs

RZECZ O MODELACH GRAFOWYCH W INŻYNIERII WIEDZY

Streszczenie: W pracy przedstawiono wybrane zastosowania grafowych modeli w inżynierii wiedzy. Wychodząc od bazowej definicji grafu jako zbioru węzłów połączonych krawędziami, zaprezentowano możliwe rozszerzenia tego pojęcia ukierunkowane na zwiększenie siły ekspresji i możliwości przetwarzania wiedzy. W szczególności praca koncentruje się na wybranych zastosowaniach modeli grafowych w obszarach eksplorowanych przez członków grupy badawczej KRaKEn.

Słowa kluczowe: graf, modele grafowe, multigrafy, hipergrafy, grafy przyczynowo-skutkowe, grafy wiedzy, BPMN, DMN, grafy reprezentacji ograniczeń

https://doi.org/10.7494/978-83-67427-74-6_8

1. Introduction

Over the past several decades, graph models have significantly gained in importance and popularity in technical and scientific applications. Such models, apart from numerical, algebraic and logical models, have a wide range of information representation possibilities and significantly facilitate both the visual and effective presentation of data, information and knowledge, as well as human-computer communication. Moreover, current computer systems enable effective processing of knowledge represented by graph models.

2. A note on evolution of graph definition

The first definition of 'graph' was introduced in 1878 by James Joseph Sylvester. He used the word to describe the similarity of the relation between mathematics and chemical structure. Some definitions of graphs that are currently used in different areas include: a simple graph which is a pair $G = (V, E)$, where V is a set of vertices and E is a set of paired vertices (edges). A directed graph is a type of graph in which the edges are directed, a multigraph extends the above definition by allowing multiple edges to have the same pair of endpoints, while a hypergraph allows the edges to join any number of vertices. A conceptual graph is a relatively new definition introduced in 1976 by John F. Sowa to describe schemas used in databases (Sowa 1976). Later, he described various applications of his definition in different fields of science: cognitive science, computer science, and artificial intelligence. The concept has recently been adapted for the knowledge reasoning and representation domain as knowledge graph. It serves as a way to integrate data using a data model based on graph structure. It is used widely in various search engines or linked open data projects. Causal graph, in turn, represents another subtype of directed graphs that are used in statistics and modeling to represent assumptions about the data-generating process.

3. Graphs in knowledge representation and reasoning

Representing knowledge about the world in terms of objects and relations between them is intuitive for humans, so multiple models, of various levels of formality, have been proposed, including informal "mind maps" and various semantic networks, taxonomies (with a defined semantics for "is-a" relationship), thesauri (with relations such as synonyms, antonyms, etc.), up to the most expressive and universal models that are called ontologies. Ontologies, defined in a selected language (ideally based on frag-

ments of formal logic), can express *any* relationships among single objects (instances, individuals) or classes (categories, concepts). In the context of the so-called Semantic Web, which aims to bring formal knowledge representation and reasoning to the Web, Web Ontology Language (OWL) has been adopted as a *de facto* standard for expressing ontologies. Carefully defined variants of OWL allow users to formally represent and process knowledge using description logics. However, the adoption of formal KRR methods is not trivial in the wider community, so a new wave of graph-oriented databases that are simpler, yet easier to use, has gained popularity in recent years. Among the modern implementations of NoSQL databases, Neo4j is probably the most popular and widely-used database system based on graphs. It features intuitive modeling and visualization interface (see Fig. 1) and provides a wide selection of libraries and extensions for data science and graph operations. No wonder it is currently used in numerous business and academic projects.

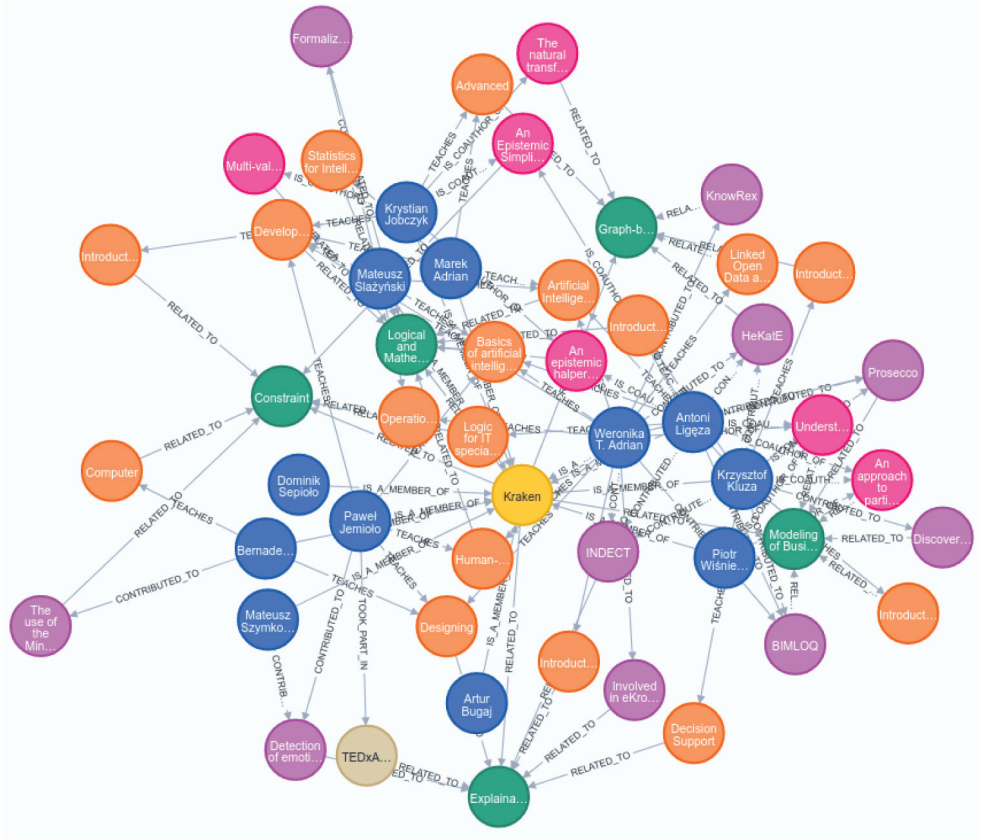

Fig. 1. Neo4j is a modern graph database with multiple extensions for visualization, data science, and knowledge processing. It represents knowledge with labeled property graphs
Source: Pękala (2022)

Graphs in knowledge representation allow for smooth navigation and facilitate knowledge discovery (see Fig. 2). Moreover, a flexible data model allows to pose intuitive queries based on graph patterns and discover paths that represent connections between the entities of the universe of discourse. In order to perform more advanced reasoning tasks, we proposed an extended knowledge graphs model (Adrian et al. 2020), in which the nodes can be of three kinds: agents, objects, and transformations, and within which several "levels" of operations, from simple CRUD tasks up to deductive and inductive reasoning can be performed.

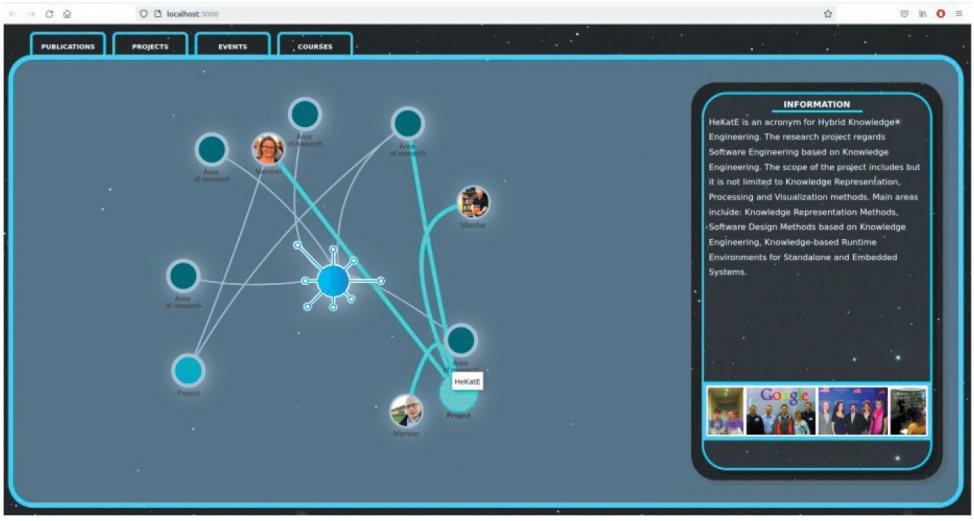

Fig. 2. Intuitive knowledge browsing and discovery of connections between real-world entities. A case of a research group knowledge representation
Source: Pękala (2022)

4. Causal graphs and Bayes nets

Bayesian networks are a well-established tool for reasoning over uncertain information that can have known and/or latent dependencies between each other. To this end, it uses concepts from two very well-established mathematical domains, namely probability and graph theory. From the first, it takes the notions of independent events and the Bayes theorem. The first concept allows us to directly calculate whether two kinds of observations (for example smoking and developing lung cancer) have any sort of dependency between them. The Bayes theorem, on the other hand, gives us a formula to find how knowledge of one event occurring affects our beliefs about another event, given the inverse dependency between them is known (as an example, we can calculate the probability of developing lung cancer if we are smoking from the information about how many lung cancer patients have been smokers).

Having the necessary tools to calculate possible dependencies between events, we also need to put some structure on them, and for that end, we use graph theory. We construct a graph by assigning a vertex to each event considered in our model. We draw an edge between two vertices if, and only if, we believe to be a direct dependency between those events (as an example, abusing alcohol would be connected by an edge to developing heart disease, but would not be connected to the inevitable heart surgery that comes afterward). This edge would be directed from cause to effect to reflect the casual nature of events in real life. Due to this cause-effect mindset, it is also obvious why no cycles would appear in such representation. Thus, any form of dependencies between events can be represented by a directed acyclic graph.

The Bayesian network model has been studied for a long time by now and many of its strengths, but also many weaknesses have been identified. One of the more significant drawbacks is that, in general, running interference on a Bayesian network is an NP-complete problem, and moreover, any known methods of approximating the results of interference can give results far from useful. Thus, if the model becomes large enough, it is unfeasible to use this particular model of reasoning.

But with some control over the size of the model, we can use Bayesian networks to draw significant conclusions in practical domains, like medicine, as can be seen in the work of Fuster-Parra et al. (2015), where a Bayesian network was implemented to discover relationships among thirteen epidemiological features of the heart age and to analyze their connection to a new concept called cardiovascular lost years. The relationship between smoking, gender, blood pressure and cholesterol to the faster aging of the heart was established (especially for the relation to smoking), while other factors have been found to be independent.

Furthermore, different heuristics can be tried to improve the results given by Bayesian networks. One such attempt was made by Staniec (2020) in his master thesis, where he inquired if clustering the data before running it through a Bayesian network could improve the results of interference. Unfortunately, no improvement was noted, and even degradation in the quality of the interference was observed. This could be the result of the clustered data being too similar, and further inquiries into ways to improve the interference are yet to be tested.

5. Graphs in business processes modeling

Models of business processes constitute graph knowledge representation that help organizations visualize and optimize their processes, what allows for achieving their business goals more effectively. There are various notations for representing these concepts (Kluza et al. 2017, Suchenia et al. 2019). Such models might be modeled manually or generated automatically (Wiśniewski et al. 2019), e.g., from natural

language description (Honkisz et al. 2018) or tasks specification in other form (Wiśniewski et al. 2018).

Since the manual creation of process models usually occurs in a dedicated editor, the use of graph-based models is related to the emerging area of automated modeling. Two main graph representations of a business process are activity graphs and business process graphs.

5.1. Activity graph

An activity graph is a directed graph where each vertex represents a process activity, and each edge corresponds to an admissible precedence relation between two activities (Yan et al. 2019). Activity graphs may also contain nodes that determine the start and end of the workflow. What remains undetermined in such graphs are split and merge flows that represent activities executed in parallel or alternatively. In process models, gateways are used to control such constructs. Although activity graphs provide a general overview of tasks in a process and their sequence, they do not explicitly represent gateways that are an essential part of every process model. As a consequence, additional information about relations between activities is needed to transform an activity graph into a process model.

5.2. Identification of logical gateways in an activity graph

One of the possibilities to determine alternative and parallel gateways in an activity graph is the identification of gateway structures. Having the activity graph as input and knowing which process activities can be executed concurrently, it is possible to define a set of constraints that need to be satisfied by the resulting structure. To obtain a correct graph-based representation of a business process, the formulated constraint satisfaction problem must be resolved for every activity graph node whose degree is greater than one.

5.3. Business process graph

Business process graphs (BP graphs) extend the concept of activity graphs by including nodes that correspond to process flow objects other than activities and start or end events (Heinrich et al. 2020). Assuming that there are no limitations on cyclicity, such graphs may serve as exact copies of process models under the condition that vertices of the graph are annotated with the activity properties, event state descriptions or gateway conditions. Figure 3 shows an example activity graph and the result of its transformation into a BP graph.

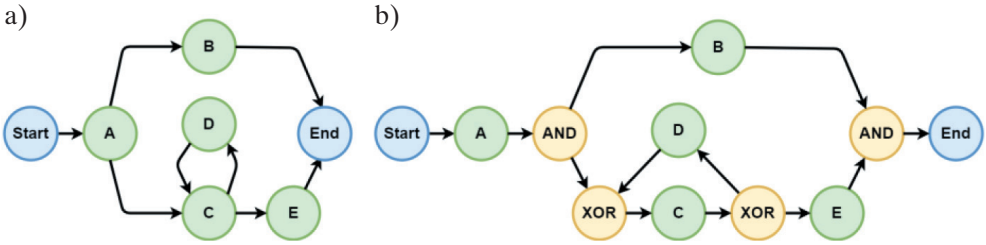

Fig. 3. An activity graph (a) and its corresponding business process graph (b)

Besides the use of process graphs in automated process modeling, such representations are present in process discovery, which consists in extracting the process model from existing event logs. Dymora et al. (2019) used the concept of a BPMN graph as a representation of a business process graph with a graphical distinction of BPMN flow objects.

6. Graphs in logic

The so-called Kripke frame is one of the most fundamental graph structures – exploited in non-classical logic. A single Kripke frame forms a semantic 'surrogate' for interpreting the modal-type logic systems, such as temporal, epistemic, deontic, or alethic logic systems.

A single Kripke frame is a tuple $F = <U, R>$, where U is a non-empty set of the so-called states, and R is a relation between states. The Kripke frame F is said to be a Kripke model if it is equipped with a standard valuation function, say V, which interprets a given modal logic language as subsets of U.

In order to reconcile a syntax of a given propositional modal language, say L, with its semantic interpretations in terms of Kripke frame-based models, a variety of completeness theorems is formulated.

Example: *Modal logic* **K** *(a propositional calculus in a modal language with Kripke axiom) is complete with respect to all Kripke models with an arbitrary relation R in them. Modal logic* **T** *(system* **K** *with axiom* $\varphi \rightarrow \Diamond \varphi$*) is complete with respect to Kripke models with reflexive relation R in them.*

Putting aside the details of semantic interpretations of modal languages in their Kripke frame-based models, we focus our attention on elucidating Kripke frames (in a more general n-dimension depiction) as a graph-type structure be its similarity to such typical graph structures as trees or irreflexive trees. The construction refers to ideas from (Gabbay et al. 2003).

Having defined a rooted (Kripke) n-frame F with root w_0, we are in a position to construct another n-frame $U = (W, R_1, \ldots, R_n)$ by taking W as the set consisting of $\langle w_0 \rangle$ and all the tuples $\langle w_0, R_1, w_1, \ldots, R_k, w_k \rangle$, for $k > 0$ of points of W, and each accessibility relation R_j from the set $\{R_1, \ldots, R_n\}$, for $j < k$, and such that:

- $w_j R_j w_{j+1}$, and,
- for any two points $\langle w_1, \ldots, w_k \rangle$ and x, the following relation S_j holds:

$$\langle w_0, ..., w_k \rangle S_j x \Leftrightarrow \exists w \in W \left(x = \langle w_0, ..., w_k \rangle R_j w \right) \tag{1}$$

Informally speaking, S_j creates new states from the previous ones by extending them at the end. The frame U, just introduced, is said to be the *unravelling* of F. The visual effect of this algebraic process is depicted in Figure 4. The left part of it delivers an exemplary Kripke frame, and the right side – its unravelling (Jobczyk 2019).

Let us assume now that R is the smallest reflexive and transitive relation containing the sum relation:

$$R = 1 \leq i \leq n R_i \tag{2}$$

If a rooted n-frame $F = (W, R_1, ..., R_n)$ of pairwise disjoint relations R_i $(1 \leq j \leq n)$ R_i is such that the set $\{y: yRx\}$, for every $x \in W$, is finite and linearly ordered by R restricted to this set, then the frame F is said to be a *tree*. A tree, which forms an *intransitive* n-frame is simply called an *intransitive tree*.

This algebraic reasoning elucidates the graph nature of general Kripke frames and illustrates the method of an intermediate transforming the ordinary (even n-dimensional) Kripke frame to the tree structures (see Fig. 4).

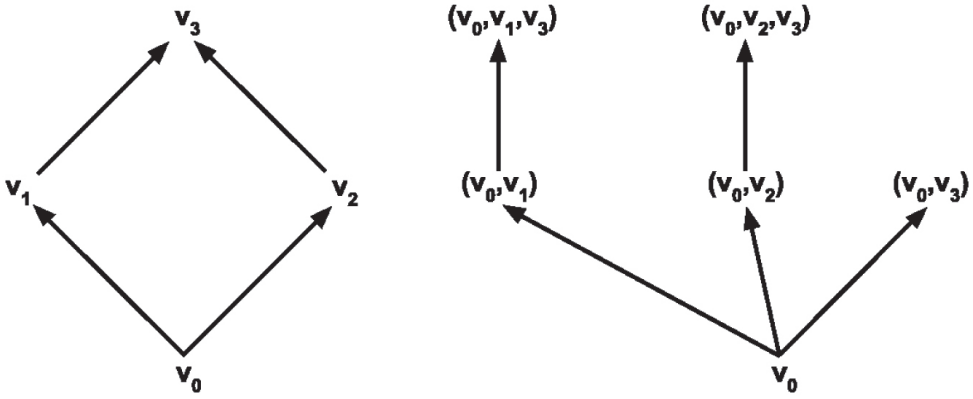

Fig. 4. The graph nature of general Kripke frames

7. Constraint problem solving: hypergraphs for constraint modeling and execution planning

A constraint satisfaction problem (CSP) is one of the classical tasks of artificial intelligence and knowledge engineering. The generic problem statement is relatively simple yet universal; in fact, solving CSP found numerous practical applications. A simple statement of CSP can be formulated as follows: given a *finite set of discrete variables* with *finite domains*, find *an assignment of domain values to all these variables*, so that *a set of predefined constraints is satisfied*. Such problems are solved with advanced constraint programming tools (CP solvers). The core idea of the search is to apply the depth-first search (DFS) with backtracking and extensive constraint propagation. Constraint networks (Lecoutre 2009), hypergraph-based representations are extensively used for modeling and analysis of CSP with the main focus on constraint propagation. However, exploring the problem structure can also be very useful in specific cases. Below, we shall illustrate the use of hypergraphs for *constraint modeling* and *planning the execution of the search*; the discussion is based on Ligęza (2011, 2012), where further details can be found.

Consider the following, well-known cryptarithmetic puzzle:

```
  SEND
+ MORE
-------
 MONEY
```

The eight variables – S, E, N, D, M, O, R, Y – are to be assigned digits from the domain 0, 1, 2, ..., 9, so that the above constraint is satisfied. Different variables are to be assigned different digits. Leading digits (in our example S and M) must be different from 0.

A naive approach may consist in examining as many as 8^{10} possible assignments; in fact, such problems suffer from a very strong combinatorial explosion. Using contemporary CP methods and tools allows us to easily model the problem and find the only admissible solution in milliseconds. However, exploring the problem structure with a dose of planning leads to the unexpectedly efficient execution of a solution plan.

Consider any of the five columns of the addition – in fact, it is a constraint of the form:

$$X + Y + C_{i-1} = Z + 10 \cdot C_i \tag{3}$$

where X and Y are the respective digits summed up in column i, C_{i-1} (if exists) is the carry digit (0 or 1) from the previous column to the left, and C_i is the carry digit to be passed to the next column.

In the presented example we have four such constraints plus one extra of the form $M = C_4$. The constraints are modeled as a hypergraph (Fig. 5) with the hyperlinks: M, SMO, EON, NRE, and DEY (represented with ovals in the middle line), and vertices: C4, C3, C2, C1, M, S, O, N, E, R, D, Y (represented with small circles).

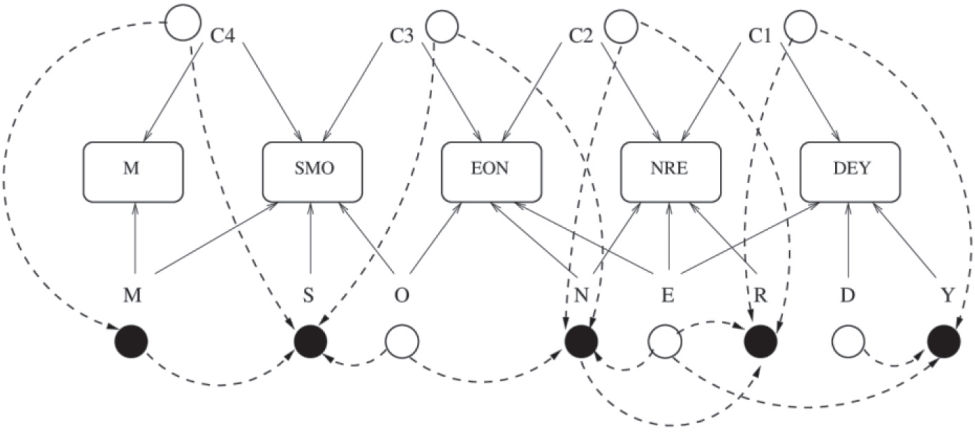

Fig. 5. A hypergraph model of constraints and variables exploration plan

Here, a very efficient plan, exemplified with the dashed line, is as follows: start with C4/2, M, C3/2, O/9, S, C2/2, E/8, N, C1/2, R, D/7, Y. The notation X/N says that we explore variable X with N values in the current domain; X alone means that at the specific point the variable value is calculated in a unique way. Variables, where selection and backtracking are needed, are represented with circles, while variables that can be calculated are represented with black-filled circles.

As for the results, a simple, pure SWI-Prolog program working according to the predefined plan finds the solution in less than 300 steps (logical inferences) and in the time less than 0.000 s CPU on a standard PC/Linux, comparing to around 13,500 steps and 0.005 s CPU when using the clp(fd) library, and around 0.075 s CPU in MiniZinc. Although planning for efficient solving of CSP remains an open issue, this section puts forward some promising ideas of the exploration of hypergraph constraint modeling.

8. Graphs in metaheuristic algorithms

As shown in the previous section, graphs (i.e., constraint hypergraphs) are a very versatile representation language for combinatorial problems. Such formally specified structure may be used to infer an efficient solving strategy tailored to the problem. While the CP solvers, in general, construct the solution by consecutively assigning val-

ues to the variables, many optimization techniques are search based – instead of building a solution, they explore all the possible solutions in an intelligent and often stochastic manner. This family of algorithms is known as metaheuristics and includes such methods as: hill climbing, simulated annealing, and tabu search.

In our work (Ślażyński et al. 2019b), we propose a hyper-heuristic method to impose a structure on the problem search space. Formally, the task involves finding a neighborhood relation: $N : S \to S$, defined over all the solutions in the search space (S in the formula). The neighborhood relation defines what solutions are considered *accessible* from the given solution. It is used in the search-based optimization to explore the search space, e.g., starting from the solution s_1 algorithm can only move to its neighbors: $s \in N(s_1)$. Given the neighborhood, the search space itself can be viewed as a graph with nodes corresponding to the solutions and edges representing the neighborhood relation itself. The choice of the neighborhood is independent of the problem and may have a crucial impact on the algorithm performance. When the neighborhood is too much connected, the cost of the exploration is prohibitively high. Also, the neighborhood should connect similar solutions, otherwise the search would be effectively random. Finally, the neighborhood may impose intelligent restrictions on the search process by excluding not interesting parts of the search space.

The proposed method exploits a constraint programming model to find an efficient neighborhood for the given combinatorial problem. There are two types of graphs involved in the process. First, there is a typed constraint network – a constraint hypergraph enriched with annotations designed to preserve structural features of a constraint programming model. Secondly, there is the neighborhood itself which (due to the size) is not representable extensionally (i.e., by enumerating all the edges). To bypass this issue, we have defined a formal language called Neighborhood Definition Language (NDL) designed to define the neighborhood intentionally, i.e., it provides a set of neighbors for the given solution. The scope of our work involved the well-specified task: creating an NDL definition of the neighborhood given the TCN representation of the combinatorial problem. Our experiments (Ślażyński et al. 2019a) have shown that such inference is possible, and the neighborhood graph may be found using evolutionary algorithms using the TCN as the source of knowledge and the basis for evaluation of various neighborhood structures.

9. Graphical models in XAI

Graphical models are an efficient form for expressing explanations as they can capture complex dependencies between facts whilst avoiding redundancy (Saha et al. 2021). Graphical models can perform both classification and regression tasks.

Moreover, the structure of tree-based models gives a promise of making a step towards efficient generation of explanations (Sepioło and Ligęza 2022). Several commonly used machine learning models are based on the graph structure. Some of them are compact and interpretable by humans (e.g., decision trees or Bayesian networks). Every prediction generated by those models can be fully explained thanks to their simulatability (the ability of a model to be simulated or thought about strictly by a human (Barredo-Arrieta et al. 2020)), as users can follow and understand the graphical representation of the model's structure.

However, human capacities limit the complexity of those transparent graphical models, which is usually connected with their weaker predictive power. More complex, opaque models are being implemented. Random forests (RF) are among the most prominent ones. RF is a collection of explainable-by-itself decision (Fig. 6).

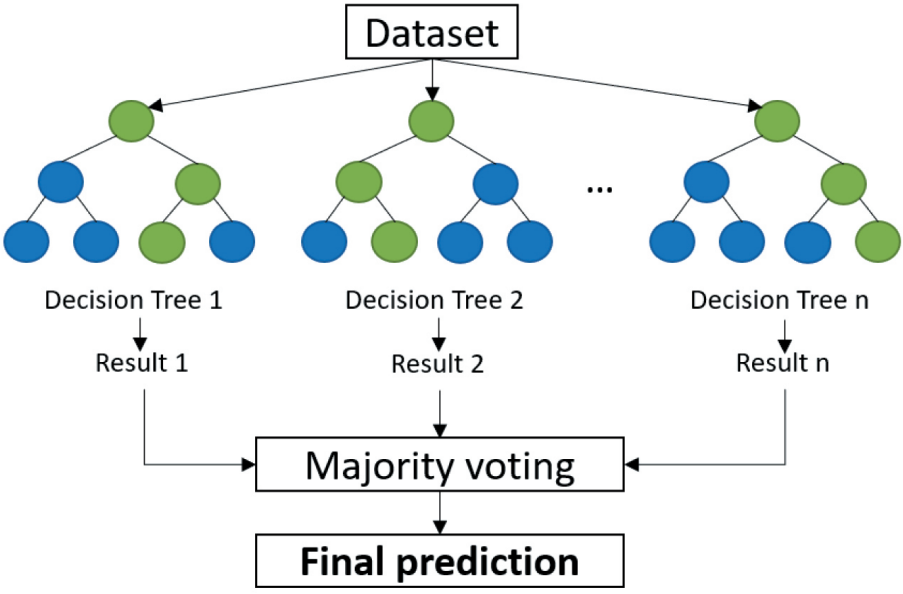

Fig. 6. A random forest operating scheme

The predictions are generated independently by each tree in the collection and then combined into the final decision, usually through majority voting. RF provides better prediction accuracy and is generally more resistant to overfitting than a single decision tree. However, complex ensemble models cannot be easily interpreted. This is the reason why various model-specific (methods that utilize assumptions regarding model structure) and model-agnostic (methods that can be applied to any class of machine learning models) explainability techniques are being developed as part of explainable artificial intelligence (XAI) toolkit.

Nowadays, more and more systems incorporate deep learning techniques. Taking into account their performance, it is not surprising. However, the accuracy is often connected with high complexity. Following this, trust in systems involving deep learning remains low. Therefore, researchers came up with explainable artificial intelligence, which goal is to provide insight into how deep learning models operate (Gunning et al. 2019). Such insight should be comprehensible for users, who often are not experts in a specific field.

In our work (Mróz et al. 2020), we focused on incorporating graphs into the process of explanations generation. More specifically, we used knowledge graphs with the significance of two features presented as the level of thickness of edges and the features encoded as nodes. We believe that presenting the results in the form of graphs may increase the readability and comprehensibility of the model structure (Adrian et al. 2022).

10. Summary

In this overview paper, we provided an outline of basic definitions of various types of graphs and summarized possible extensions of graph concepts and applications of graph-based technologies. Specifically, we presented various applications of graph models in the knowledge engineering area explored by the members of the KRaKEn research group, such as knowledge representation, knowledge discovery, business process modeling, logic reasoning, constraint modeling and execution planning, metaheuristic algorithms, as well as explainable artificial intelligence methods.

References

Adrian W.T., Adrian M., Kluza K., Stachura-Terlecka B., Ligęza A., 2020, *Extended knowledge graphs: A conceptual study*, [in:] Aveiro D., Dietz J., Filipe J. (eds.), *IC3K 2020: Proceedings of the 12th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management: November 2–4, 2020. Vol. 2, KEOD: 12th International Conference on Knowledge Engineering and Ontology Development*, SciTePress – Science and Technology Publications, pp. 173–180. https://doi.org/10.5220/0010111601730180.

Adrian W.T., Kluza K., Zaremba M., Jemioło P., Adrian M., Pękala A., Florek K., Ligęza A., 2022, *Selected applications of graph-based knowledge representation*, [in:] *PP-RAI'2022: Proceedings of the 3rd Polish Conference on Artificial Intelligence: April 25–27, 2022, Gdynia, Poland*, Gdynia Maritime University, Gdynia, pp. 172–175.

Barredo Arrieta A., Díaz-Rodríguez N., Del Ser J., Bennetot A., Tabik S., Barbado A., Garcia S., Gil-Lopez S., Molina D., Benjamins R., Chatila R., Herrera F., 2020, *Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI*, Information Fusion, vol. 58, pp. 82–115. https://doi.org/10.1016/j.inffus.2019.12.012.

Dymora P., Koryl M., Mazurek M., 2019, *Process discovery in business process management optimization*, Information, vol. 10(9), 270. https://doi.org/10.3390/info-10090270.

Fuster-Parra P., Tauler P., Bennasar-Veny M., Ligęza A., Lopez-Gonzalez A.A., Aguiló A., 2016, *Bayesian network modeling: A case study of an epidemiologic system analysis of cardiovascular risk*, Computer Methods and Programs in Biomedicine, vol. 126, pp. 128–142. https://doi.org/10.1016/j.cmpb.2015.12.010.

Gabbay D., Kurucz A., Wolter F., Zakharyashev M. (eds.), 2003, *Many-Dimensional Modal Logic: Theory and Applications*, Studies in Logic and the Foundations of Mathematics, 148, Elsevier, Amsterdam.

Gunning D., Stefik M., Choi J., Miller T., Stumpf S., Yang G.Z., 2019, *XAI – Explainable artificial intelligence*, Science Robotics, vol. 4(37), eaay7120. https://doi.org/10.1126/scirobotics.aay7120.

Heinrich B., Schiller A., Schön D., Szubartowicz M., 2020, *Adapting process models via an automated planning approach*, Journal of Decision Systems, vol. 29(4), pp. 223–259. https://doi.org/10.1080/12460125.2020.1800976.

Honkisz K., Kluza K., Wiśniewski P., 2018, *A concept for generating business process models from natural language description*, [in:] Liu W., Giunchiglia F., Yang B. (eds.), *Knowledge Science, Engineering and Management: 11th International Conference, KSEM 2018: Changchun, China, August 17–19, 2018: Proceedings, Pt. 1*, Lecture Notes in Computer Science, vol. 11061, Springer, Cham, pp. 91–103. https://doi.org/10.1007/978-3-319-99365-2_8.

Jobczyk K., 2019, *Multi-valued deontic Halpern–Shoham logic for fuzzy deontic-temporal expressions*, Journal of Intelligent and Fuzzy Systems, vol. 36(5), pp. 5091–5103. https://doi.org/10.3233/JIFS-179054.

Kluza K., Wiśniewski P., Jobczyk K., Ligęza A., Suchenia A., 2017, *Comparison of selected modeling notations for process, decision and system modeling*, [in:] Ganzha M., Maciaszek L., Paprzycki M. (eds.), *FedCSIS: Proceedings of the 2017 Federated Conference on Computer Science and Information Systems: September, 3–6, 2017, Prague, Czech Republic*, IEEE, Piscataway, pp. 1111–1114. https://doi.org/10.15439/2017F454.

Lecoutre Ch., 2009, *Constraint Networks: Targeting Simplicity for Techniques and Algorithms*, ISTE, Wiley, London, UK, Hoboken, USA.

Ligęza A., 2011, *Models and tools for improving efficiency in constraint logic programming*, Decision Making in Manufacturing and Services, vol. 5(1–2), pp. 68–78.

Ligęza A., 2012, *Improving Efficiency in Constraint Logic Programming Through Constraint Modeling with Rules and Hypergraphs*, [in:] Ganzha M., Maciaszek L., Paprzycki M. (eds.), *FedCSIS: Proceedings of the Federated Conference on Computer Science and Information Systems 2012: September 9–12, 2012 Wrocław, Poland*, Polskie Towarzystwo Informatyczne, Warsaw; IEEE Computer Society Press, Los Alamitos, pp.101–107.

Mróz P., Quemy A., Ślażyński M., Kluza K., Jemioło P., 2020, *GBEx – towards graph-based explanations*. [in:] Alamaniotis M., Pan S. (eds.), *ICTAI 2020: IEEE 32nd International Conference on Tools with Artificial Intelligence: 9–11 November 2020: Proceedings*, IEEE, Piscataway, pp. 112–117. https://doi.org/10.1109/ICTAI50040.2020.00028.

Pękala A., 2022, *Strona internetowa zespołu badawczego zaprojektowana w oparciu o graf wiedzy*, AGH University of Science and Technology [engineering thesis, unpublished].

Saha S., Yadav P., Bauer L., Bansal M., 2021, *ExplaGraphs: An Explanation Graph Generation Task for Structured Commonsense Reasoning*, [in:] Moens M.-F., Huang X., Specia L., Wen-tau Yih S. (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: EMNLP 2021 Virtual Event: Punta Cana, Dominican Republic, 7–11 November, 2021*, Association for Computational Linguistics, pp. 7716–7740. https://doi.org/10.18653/v1/2021.emnlp-main.609.

Sepioło D., Ligęza A., 2022, *Towards explainability of tree-based ensemble models: A critical overview*, [in:] Wojciech Zamojski et al. (eds.), *New Advances in Dependability of Networks and Systems: Proceedings of the Seventeenth International Conference on Dependability of Computer Systems DepCoS-RELCOMEX: June 27–July 1, 2022, Wrocław, Poland*, Lecture Notes in Networks and Systems, vol. 484, Springer, Cham, pp. 287–296. https://doi.org/10.1007/978-3-031-06746-4_28.

Sowa J.F, 1976, *Conceptual graphs for a data base interface*, IBM Journal of Research and Development, vol. 20(4), pp. 336–357. https://doi.org/10.1147/rd.204.0336.

Staniec A., 2020, *Wybrane problemy odkrywania modeli przyczynowo-skutkowych z użyciem sieci bayesowskich*, Akademia Górniczo-Hutnicza, Kraków [M.Sc. thesis].

Suchenia A., Łopata P., Wiśniewski P., Stachura-Terlecka B., 2019, *Towards UML representation for BPMN and DMN models*, MATEC Web of Conferences, vol. 252, 02007. https://doi.org/10.1051/matecconf/201925202007.

Ślażyński M., Abreau S., Nalepa G., 2019a, *Generating local search neighborhood with synthesized logic programs*, Electronic Proceedings in Theoretical Computer Science, vol. 306, pp. 168–181. https://doi.org/10.4204/EPTCS.306.22.

Ślażyński M., Abreau S., Nalepa G., 2019b, *Towards a formal specification of local search neighborhoods from a constraint satisfaction problem structure*, [in:] *GECCO 2019: The Genetic and Evolutionary Computation Conference: A Recombination of the 28th International Conference on Genetic Algorithms (ICGA) and the 24rd Annual Genetic Programming Conference (GP): July 13th–17th 2019, Prague, Czech Republic*, ACM, USA, pp. 137–138. https://doi.org/10.1145/3319619.3321968.

Ślażyński M., Abreau S., Nalepa G., 2019c, *Generating local search neighborhood with synthesized logic programs*, Electronic Proceedings in Theoretical Computer Science [35th International Conference on Logic Programming (ICLP'19): Las Cruces, USA, September 20–25, 2019], vol. 306, pp. 168–181. http://doi.org/10.4204/EPTCS.306.22.

Wiśniewski P., Kluza K., Ligęza A., 2018, *An approach to participatory business process modeling: BPMN model generation using constraint programming and graph composition*, Applied Sciences, vol. 8(9), 1428. https://doi.org/10.3390/app8091428.

Wiśniewski P., Kluza K., Jobczyk K., Stachura-Terlecka B., Ligęza A., 2019, *Overview of generation methods for business process models*, [in:] Douligeris Ch., Karagiannis D., Apostolou D. (eds.), *Knowledge Science, Engineering and Management: 12th International Conference, KSEM 2019, Athens, Greece, August 28–30, 2019: Proceedings, Pt. 2*, Lecture Notes in Computer Science, vol. 11776, Springer, Cham, pp. 55–60. https://doi.org/10.1007/978-3-030-29563-9_6.

Yan Z., Sun B., Chen Y., Wen L., Hu L., Wang J., Yang M., Wang L., 2019, *Decomposed and parallel process discovery: A framework and application*, Future Generation Computer Systems, vol. 98, pp. 392–405. https://doi.org/10.1016/j.future.2019.03.048.