COASTAL RESEARCH AND EDUCATION ACTIONS FOR TRANSPORTATION EQUITY
TIER 1 UNIVERSITY TRANSPORTATION CENTER
U.S. DEPARTMENT OF TRANSPORTATION

Automated Knowledge Graphs for Life-Cycle Management Of Coastal Bridge Networks

July 15, 2024

(1) Cheng, Minghui, Ph.D., Assistant Professor at University of Miami, Department of Civil & Architectural Engineering, College of Engineering, 1251 Memorial Dr, Coral Gables, FL 33146; ORCID https://orcid.org/0000-0002-8983-5148; Email: minghui.cheng@miami.edu
(2) Nanni, Antonio, Ph.D., P.E., Professor at University of Miami, Department of Civil & Architectural Engineering, College of Engineering, 1251 Memorial Dr, Coral Gables, FL 33146; ORCID https://orcid.org/0000-0003-2678-9268; Email: nanni@miami.edu
(3) Kulesza, Stacey, Ph.D., P.E., Professor at Texas State University, Ingram School of Engineering, 327 W Woods St, San Marcos, TX 78666; ORCID https://orcid.org/0000-0003-3283-6235; Email: sekulesza@txstate.edu

Final Research Report
Prepared for:
Coastal Research and Education Actions for Transportation Equity

ACKNOWLEDGMENT

This study was funded, partially or entirely, by the U.S. Department of Transportation through the Coastal Research and Education Actions for Transportation Equity University Transportation Center under Grant Award Number 69A3552348330.

DISCLAIMER

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. This document is disseminated under the sponsorship of the U.S. Department of Transportation's University Transportation Centers Program, in the interest of information exchange. The U.S. Government assumes no liability for the contents or use thereof.

ABSTRACT

With the ability to harness the power of big data, the digital twin (DT) technology has been increasingly applied to the modeling and management of structures and infrastructure systems, such as buildings, bridges, and power distribution systems. Supporting these applications, an important family of methods are based on graphs. For DT applications in modeling and managing smart cities, a large-scale knowledge graph (KG) is needed to represent the complex relationships and model the urban infrastructure as a system of systems. To this end, this paper develops a conceptual framework **Auto**mated knowledge **Gra**phs for **C**omplex **S**ystems (AutoGraCS). In contrast to existing KGs developed for DTs, AutoGraCS can support KGs accounting for statistical correlations and interdependencies within the complex systems. The established KGs from AutoGraCS can then be easily turned into Bayesian networks for probabilistic modeling and Bayesian analysis. Besides, AutoGraCS provides flexibility in support of users' need to implement the ontology and rules when constructing the KG. With the user-defined ontology and rules, AutoGraCS can automatically generate a KG to represent a complex system consisting of multiple systems. The bridge network in Miami-Dade County is used as an illustrative example to generate a KG that integrates multiple layers of data from the bridge network, traffic monitoring facilities, and flood water watch stations.

Keywords: System Digital Twin, Bayesian Network, Infrastructure Systems; Knowledge Graph

INTRODUCTION

The Digital twin (DT) technology has been increasingly applied to the modeling and management of structures and infrastructure systems (Broo et al. 2022; Errandonea et al. 2020; Pregnolato et al. 2022; Wagg et al. 2020). Example applications include individual structures, such as aircrafts (Kapteyn et al. 2021; Xiong and Wang 2022) and ships (Fonseca and Gaspar 2021; Groden and Collette 2017; Zhu and Collette 2017), and infrastructure, such as bridges (Lin et al. 2021; Pregnolato et al. 2022; Shim et al. 2019; Yoon et al. 2022) and industrial facilities (Guo et al. 2020; Jiang et al. 2023). The increase in popularity stems from the ability of DT to harness the power of big data to create a digital model that continuously represents/reports the conditions and performance the physical entity/asset over a long period time. More importantly, DT enables prediction and simulation of the future for the purpose of decision-making (Kunzer et al. 2022; Vieira et al. 2022; Wagg et al. 2020).

However, existing studies on DT rarely extend to urban infrastructure systems or even a whole city at a large scale because of their limitations in scalability, computational efficiency, and the capability to consider complex correlations and interdependencies across multiple layers of sub-systems.

Knowledge graphs (KGs) are widely used in search engines and recommendation systems to effectively store, manage, and query a large amount of data that represents prior knowledge (Chen et al. 2020). For infrastructure systems modeling and management, there exist multiple KGs and the corresponding ontologies. In (Austin et al. 2020), a KG is created for 1,788 buildings in Chicago to store data related to building energy consumption. In Coelho et al. (2017), a KG is established for a school system in Columbia-Clarksville Area to document the enrollment of the students in the region. For highway asset management, KGs are created to store the conditions of road sections and to prescribe maintenance actions (France-Mensah and O'Brien 2019; Le and Jeong 2016). For construction management, KGs with detailed ontologies are developed to monitor and manage the construction process (El-Gohary and El-Diraby 2010; Wu et al. 2021; Zhang et al. 2015). While these KGs represent the domain knowledge in detail via ontologies and efficiently store and query the large amounts of data associated with the knowledge, they focus on one specific domain and cannot support the modeling of system evolution and dynamics. Recently,

the World Avatar project was initiated and dynamic KGs are developed (Akroyd et al. 2021; Eibeck et al. 2019; Hofmeister et al. 2023; Savage et al. 2022). The World Avatar serves as a junction that stores and visualizes data from different domains, while the dynamic KG is the reasoning engine behind the project and can be updated by adding/deleting data and the instances. However, these existing KGs suffer from two limitations. First, the KGs cannot be used for probabilistic modeling and inference. Second, the existing works do not provide the capability/flexibility to define the complex and interdependent relationships across the systems. As urban infrastructure and cities are complex systems of systems, DTs should be able to (1) perform necessary statistical analysis and simulation such as spatio-temporal analysis and (2) model the correlations and interdependencies within and across the systems. Due to the above two limitations, the existing KGs are not yet suitable for DTs of urban infrastructure systems and smart cities.

Recently, the concept of performance-oriented system was proposed by Cheng and Gao (2024) to tackle the above challenges in building digital twins for system of systems. Given the desired performance objectives and available data, customized system digital twins for specific performance measures can be constructed from a large-scale knowledge graph (KG) that represents multiple urban infrastructure systems. The performance-oriented digital twins are flexible, manageable in size and computationally feasible. This work makes good headway towards construction of large-scale and complex KGs that are necessary for building SDTs in support of cross-sector infrastructure systems modeling/management to the city scale.

Extending our previous SDT research and KG generation method, in this paper we propose the framework of AutoGraCS, which automates knowledge graphs for complex systems, to address the aforementioned two limitations in existing KG methods. AutoGraCS can automate the generation of KGs from user inputs, including an ontology, a set of rules, and databases. The ontology reflects the multi-domain knowledge and represents the relationships within and across sub-systems/components, while the set of rules are used to determine whether the relationships exist between the individual variables of the KG. Each database represents a system, with each object of the system having multiple variables. Given the ontology and the set of rules, AutoGraCS can automatically create a KG by connecting the variables of the same object, the variables of different objects within the same database, and variables of different objects across the databases. The resulting AutoGraCS-generated KG can be easily turned into Bayesian network that supports

probabilistic modeling and inference, hence addressing the first limitation of existing KG methods. The user-defined ontology and set of rules tackle the second limitation as they give the flexibility to define complex relationships. For illustrative purpose, a hypothetical bridge network with a small size will be used to provide detailed information on how to construct the KG. After that, a KG that connects the bridges, river gages, and traffic monitoring sites in Miami-Dade County will be used to demonstrate the ability of AutoGraCS to create large-scale KGs.

BAYESIAN NETWORK (BN): AN ENABLER OF DT

BN serves as the bridges between KGs and digital twins. In this section, we will first provide an overview of how BN can enable digital twins and then introduce basic concepts of BN including a discussion of graphical structures that are used in our KGs.

BN-based DT

BN is a graphical model that can perform probabilistic modeling and Bayesian inference. Different types of BN have been applied to various problems. For example, BN has been used for probabilistic analysis of infrastructure systems such as transportation, water, and power networks (Applegate and Tien 2019; Byun and D'Ayala 2022; Cheng and Gao 2023; Johansen and Tien 2018). For another, variants of BN have been applied to real-time monitoring of structures (Groden and Collette 2017; Kapteyn et al. 2021) and life-cycle management of structures (Bismut and Straub 2021; Cheng and Frangopol 2021). Using BN, we can update any variables of interest when data is available for any variables in the BN. This characteristic aligns well with the objective of DTs because DTs utilize available data from the physical world to update the virtual model. Therefore, we focus on building DTs based on BN.

Previously, performance-oriented SDT (Cheng and Gao 2024) was proposed by the same authors for the modeling and management of infrastructure systems. Figure 1 shows the comparison of the conventional DT and performance-oriented SDT based on BN. For a conventional DT, a BN is created for the entire asset/entity to be modeled (e.g., a plane and a power distribution network). For performance-oriented SDT, the entire asset/entity is represented by a KG instead of a BN. When there is a need for constructing the DT, the user provides the performance objectives and the available data sources. Given these user's inputs, a custom subgraph is cut from the KG and transformed into a BN, which then serves as the DT. The

philosophies to design the performance-oriented SDT are twofold. First, for computational purposes, the size of a BN should be kept as small as possible. Urban infrastructure is large in scales. If we create a DT for it based on a substantially large graph, it is computationally infeasible to perform Bayesian analysis. However, to create a DT for a specific objective, we do not need every part of the city and all the available data. Therefore, we propose to store a KG to represent the urban infrastructure and only select the necessary subgraph to build the DT. Second, we process the data differently for different modeling objectives. For example, we want short-term average precipitation for flood warning and want annual precipitation for climate change modeling. The DT for flood warning and that for climate change modeling use the same variable (i.e., precipitation) but process it differently. Therefore, the DTs should function independently. More explanations of the design philosophies of performance-oriented SDT can be found in (Cheng and Gao 2024). To fully release the potential of performance-oriented SDT, a large-scale KG is necessary, which is the focus of this paper. Since the subgraphs of the KG will be transformed into BNs and then DTs, when the KG is generated, it should possess the properties of BNs.

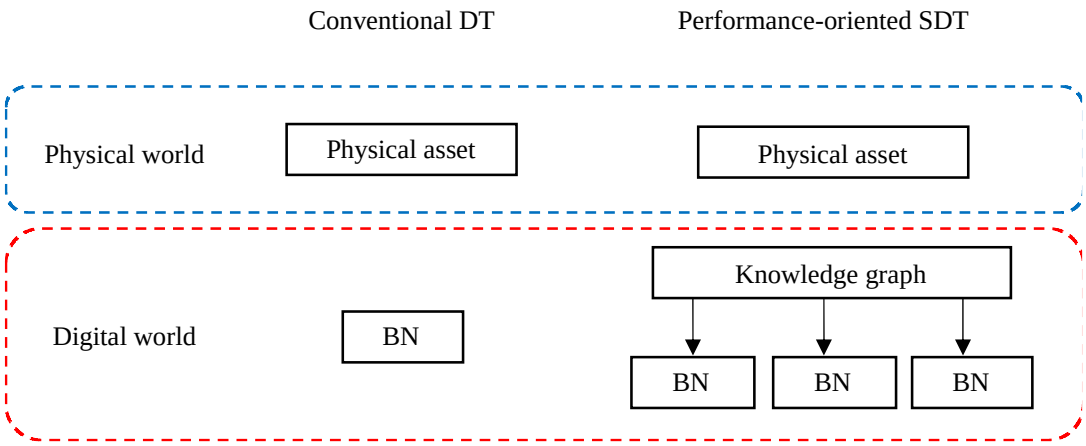


Figure 1. Comparisons between conventional DTs and performance-oriented SDT

Basic concepts of BN

BN is a directed acyclic graph consisting of a set of nodes and a set of edges to represent a joint probability distribution (Marcot and Penman 2019). Specifically, the nodes represent the random variables, while the edges represent the statistical relationships between the nodes. Figure 2 shows an illustrative BN. There are four nodes with X_1, X_2 as root nodes and X_3, X_4 as child nodes. X_1 and X_2 are described by marginal probability distributions $P(X_1)$ and $P(X_2)$, respectively, while X_3 and X_4 are described by conditional probability distributions $P(X_3|X_1, X_2)$

and $P(X_4|X_2)$, respectively. Given the graphical structure and the marginal and conditional probability distributions, the joint probability distribution of X_1 to X_4 is written as

$$P(X_1, X_2, X_3, X_4) = P(X_1)P(X_2)P(X_3|X_1, X_2)P(X_4|X_2) \quad (1)$$

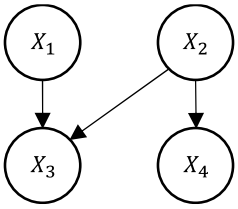


Figure 2. Example BN with Four nodes

BN can be used to perform Bayesian inference on any variables of interests when any data is available. For example, when we observe X_1 to be 1 and the variable of interest is X_3, BN can efficiently calculate the conditional probability $P(X_3|X_1 = 1)$. The functionality of a DT is to update the performance/conditions of the digital model given the available data, which is essentially calculating the conditional probability. Classical algorithms for Bayesian inference include the elimination algorithm and junction tree algorithm (Bensi 2010), while several methods have been proposed to improve the efficiency for inference with large BNs (Bensi et al. 2013; Byun et al. 2019; Tien and Der Kiureghian 2016).

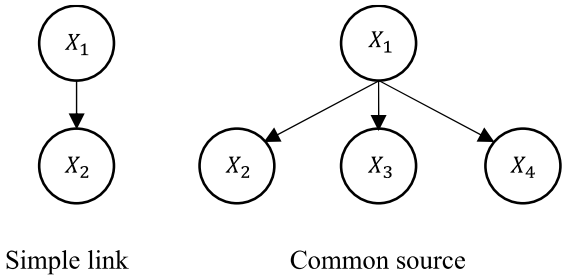


Figure 3. Typical graphical structures used in the KG

Since the KG should be easily turned into BN, when we generate the KG, we will follow the typical graphical structures of BN. Figure 3 shows two typical graphical structures that are used in the development of KGs. The left one shows a link between two nodes. There exist multiple interpretations: (1) X_1 is the cause and X_2 is the effect, (2) X_2 is a function of X_1, and (3) X_2 is the measurement of X_1. The right one shows a common source relationship. X_1 is the common source of X_2, X_3, and X_4. In terms of statistical modeling, we can use this graphical structure to model

correlated random variables (i.e., X_2, X_3, and X_4). More details on the graphical structure of modeling correlated random variables can be found in Bensi et al. (2011).

AUTOGRACS FRAMEWORK

AutoGraCS is developed to automatically generate KGs for complex systems. The user will provide three inputs: an ontology, a set of rules, and databases. The output is a KG that connects the variables of objects within and across the databases while considering statistical correlations.

Before going into more details on the methodology, the terminologies used in this paper are clarified. Each database is a class with each row and each column being an object and an attribute of the class, respectively. A variable is an attribute of a specific object and takes the format of "class[ID].attribute", where ID is the unique ID of the object in its class (i.e., database). An ontology is a graph that defines the relationships between the attributes with each node taking the format of "class.attribute". Each link of the ontology indicates that the two attributes are related. Yet, this does not mean that any two variables that match the attribute types can be connected to. To this end, for each link of the ontology, a rule is additionally defined to connect those that match the criteria stated in the rule. Since an ontology has multiple links, the user should provide a set of corresponding rules so that the number of rules equals the number of links. A KG, which is the output of AutoGraCS, consists of the selected variables from the databases.

In general, AutoGraCS has two phases. First, AutoGraCS will identify and connect all the variables. Then the statistically correlated variables are clustered and connected to a common source node. Please note that the specific graphical structure of a cluster of correlated variables should be determined given the data and the common source node only reflects that the variables are correlated.

First phase

Figure 4 illustrates the first phase of AutoGraCS using a simple and hypothetical example. On the left of Figure 4, it shows the three inputs: databases, an ontology, and a set of two rules, while on the right, it shows the KG. The simple example has two classes C and D whose attribute values are stored in two databases. Class C and D have three and two attributes, respectively and they both have attribute Y. The ontology has three nodes (i.e., C.A1, C.A2, and D.A3) and two

edges. The first edge links C.A2 and C.A1 and applies rule #1, while the second edge links D.A3 and C.A1 and applies rule #2. Specifically, rule #1 requires that the two variables should belong to the same object. Rule #2 states that if the two variables can be connected, the Y values of the two objects to which the two variables correspond must be the same. Evaluating these two rules will be introduced along with a detailed explanation of constructing the KG. Please note that Figure 4 is color coded. Particularly, if the color of the edge in the ontology matches that in the KG, this means the corresponding rule is satisfied. If the color of the node in the ontology matches that in the KG, this indicates that these nodes have the same attribute and class.

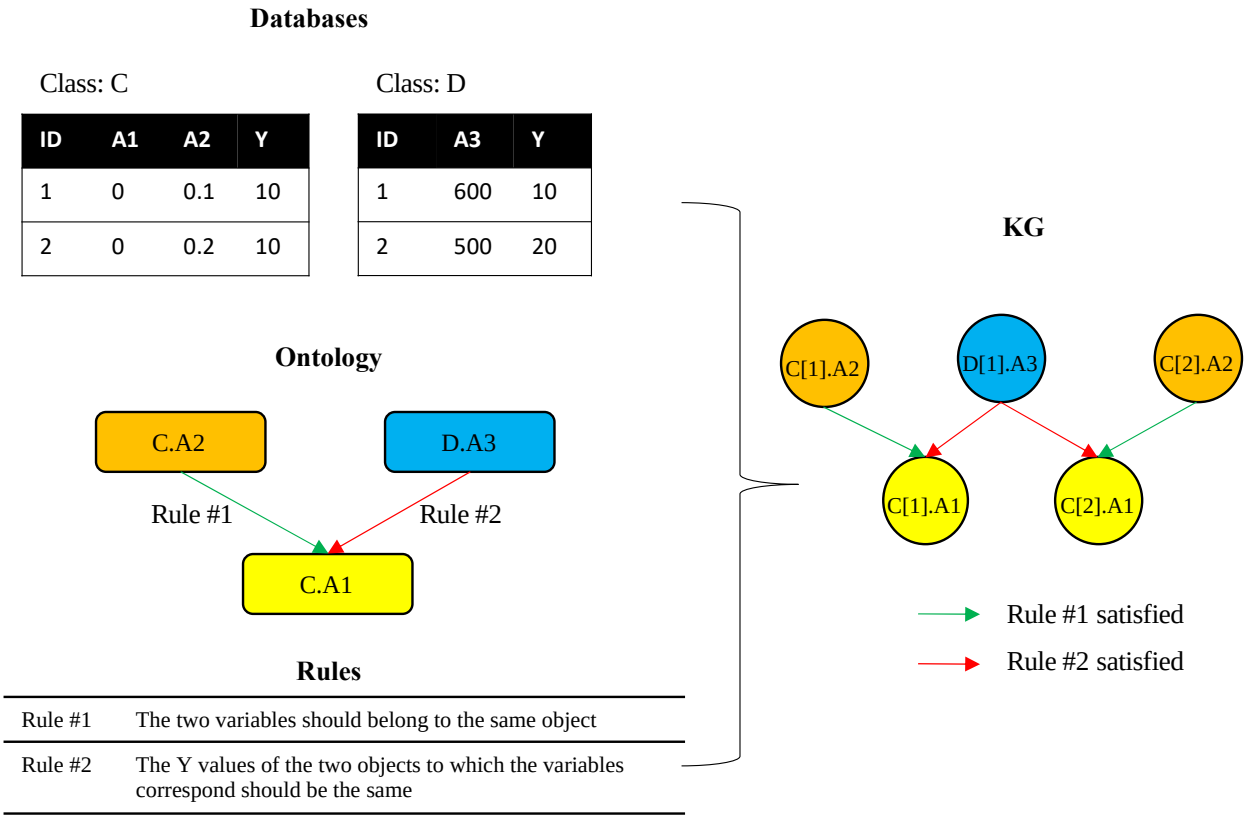


Figure 4. Illustration of the first phase of AutoGraCS

The number of steps to construct a KG in the first phase is the sum of the number of leaf nodes and edges in the ontology. For the simple example in Figure 4, the number of steps is three (i.e., one leaf node and two edges). The three steps are presented as follows:

1. We start from the leaf node of the ontology, which is C.A1. There are two objects in class C, so we add the two corresponding leaf nodes to the KG, which is C[1].A1 and C[2].A1.

2. In this step, rule #1 is evaluated. We start from finding the parent of variable C[1].A1. Based on the edge in the ontology, the parent of C.A1 is C.A2. Therefore, there are two candidates for this variable in the KG, which are C[1].A2 and C[2].A2. Since only C[1].A2 and C[1].A1 belongs to the same object C[1], based on rule #1, we add the variable C[1].A2. Similarly, variable C[2].A2 is added as the parent of C[2].A1.
3. In this step, rule #2 is evaluated. Again, we start from finding the parent of variable C[1].A1. Based on the edge of ontology, there are two candidates D[1].A3 and D[2].A3. For D[1].A3, the Y value of the object is 10 and equals the Y value of the object corresponding to C[1].A1. Since rule #2 satisfies, we add D[1].A3 and an edge between it and C[1].A1. However, rule #2 does not hold for D[2].A3. We then find the parent for variable C[2].A1. Similarly, rule #2 holds for D[1].A3 but not for D[2].A3. As a result, we add another edge between C[2].A1 and D[1].A3.

The three steps above illustrate the procedure to create a KG. The procedure can be repeated for problems with more and larger databases, a larger ontology, and a set of more complicated rules.

Second phase

Figure 5 shows the second phase of AutoGraCS. It starts from the KG generated in the first phase. Among the attributes of the ontology, the user first provides a list of attributes. For an attribute, if the variables of this attribute type can be correlated, the attribute will be called a self-correlated attribute. Each attribute in this list is a self-correlated attribute and this list is called a self-correlated list. The user then should provide a rule for each attribute and based on the rule, AutoGraCS can create clusters of variables with each cluster including correlated variables. For example, the substructure damage levels of different bridges in a region after an earthquake are statistically correlated and can be modeled as a random field (Bocchini and Frangopol 2011), so “bridge.sub_damage_level” is considered self-correlated and is added to the self-correlated list. When two bridges are too far away, their damage levels will have little correlation. Based on this,

the user defines a rule that stipulates the distance threshold for correlation and AutoGraCS generates a nested list of variables for this attribute. Each sublist represents a cluster in which the variables (i.e., substructure damage levels of different bridges) are correlated.

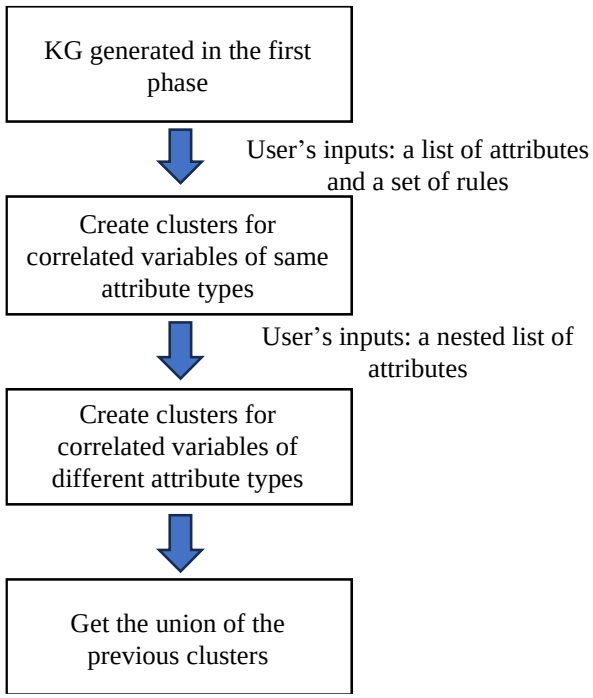


Figure 5. Illustration of the second phase of AutoGraCS

In the above step, the clusters of variables of the same attribute types are generated. Besides, variables of different attribute types can be correlated. For example, the superstructure damage level of a bridge is correlated to the substructure damage level of the same bridge. AutoGraCS also gives the flexibility to model this type of correlation, as shown in Figure 5. If variables of one attribute type can be correlated to variables of another attribute type, the attribute is called an inter-correlated attribute and the two attributes are inter-correlated. The user should provide a nested list of inter-correlated attributes with each sublist including inter-correlated attributes. For ease of implementation, AutoGraCS stipulates that the correlated variables of different attribute types should belong to the same object (i.e., ID should be the same). Using the bridge example, the output of this step is a nested list. Each sublist is a small cluster and has two variables: superstructure and substructure damage levels of the same bridge.

The final step of the second phase is to obtain the union of the clusters generated in the first and second steps. For example, there are two nearby bridges B1 and B2. The output of the first

step produces two clusters, which are superstructure damage levels of Bridges B1 and B2 (i.e., cluster #1) and substructure damage levels of Bridge B1 and B2 (i.e., cluster #2). The output of the second step also produces two clusters, which are superstructure and substructure damage levels of Bridge B1 (i.e., cluster #3) and those of Bridge B2 (i.e., cluster #4). After joining the clusters, the output becomes one big cluster that includes all four variables (i.e., superstructure and substructure damage levels of Bridges B1 and B2). While the variables in the clusters of the first two steps (e.g., cluster #1 to #4) generally have strong correlations, the variables in the joined clusters may not. However, the KG will represent the existence of correlations by assigning a common source node for each joined cluster after the final step. Future work is needed to better represent the correlations of variables with appropriate graphical structures. Some pioneering works have been done in Bensi et al. (2011).

IMPLEMENTATION OF AUTOGRACS

We aim at implementing a user-friendly AutoGraCS so that many operations can automate and the users can reduce the coding load. In this section, the structure of AutoGraCS platform is introduced and some implementation details are provided.

The platform has four major files, which include ontology, rules, graph construction functions, and main files. The contents of the four files are introduced as follows:

The ontology file is written by the user. In this file, the user should provide an ontology, a list of attributes to create clusters for correlated variables of same attribute types, and a nested list of attributes to create clusters for correlated variables of different attribute types. Please note that the ontology should be written as an edge list. For example, the ontology in Figure 4 is written as {(C.A2, C.A1), (D.A3, C.A1)}. Besides, for each edge, an additional rule indicator and a number of top candidates if necessary should be provided to call the correct type of graph construction functions and select the correct number of candidates.

The rules file is also written by the user. The user should provide rules for the first and second phase. For the first phase, each rule in the file corresponds to an edge in the ontology. The order of the rules should follow that of the edge lists of the ontology. Each rule evaluates a pair of variables and is used to determine whether an edge exists between the two variables. There are two

types of rules. One type (i.e., rule indicator = 0) returns true or false, while the other type (i.e., rule indicator = 1) returns a score. For the former one, if it returns true, then an edge will be created by a graph construction function. For the latter one, the score will be stored and the top candidates with the highest scores will be selected to establish the edges by a graph construction function. Please note that the number of top candidates is provided in the ontology file.

For the second phase, the user should define rules to create clusters for correlated variables of the same attribute types (i.e., self-correlated attributes) and different attribute types (i.e., inter-correlated attributes). For the first step of the second phase, each rule in the file corresponds to a self-correlated attribute in the list. The inputs of the rule are the potential candidate variables of the same attribute type and the database. The rule will access attribute values from the database and calculate a numeric affinity matrix for the candidate variables. For the second step of the second phase, a default rule corresponds to a sublist of inter-correlated attributes. Similarly, the inputs of rule are the potential candidate variables of different attribute types. In addition to the user-defined calculations, the default rule enforces that if the candidate variables are put into the same cluster, they should belong to the same object and will be assigned the same categorical value. After all the calculations, the output of the rule is a numeric array.

The graph construction functions file is written by the developers of AutoGraCS. In the first phase, a graph construction function (GCF) searches all possible candidate pairs of variables, calls the rule to evaluate all the candidate pairs, and creates edges for those pairs that satisfy the rule. In general, there are two types of GCFs. The first type corresponds to the rules that return true or false. The GCF loops over all candidate pairs and adds edges for the pairs whose rule evaluation returns true. The second type corresponds to the rules that return a score. A heap is used to maintain the top candidate pairs. Similarly, the GCF loops over all candidate pairs. However, after evaluating each candidate pair, the score and candidate pair are pushed to the heap. After all evaluations, edges are created for the top candidate pairs in the heap. In the second phase, a GCF searches all possible candidate variables that match the desired attribute type, calls the rule to obtain a numeric matrix, and uses agglomerative clustering in scikit-learn (Pedregosa et al. 2011) to cluster the candidate variables based on the matrix. After finishing the first two steps of the second phase, another GCF will be called to obtain the unions of the obtained cluster. Please note

that there is no need for the user to explicitly call GCFs because AutoGraCS can read the ontology file and automatically call GCFs.

In the main file, the user should read the databases and provide the selection criteria of leaf variables of the KG. This selection is generally simple, such as selecting the variables associated with the bridges in Miami-Dade County. AutoGraCS then automatically adds the leaf nodes of the KG and runs the remaining functions to add edges and establish clusters.

In addition to the four files, the user should provide the databases. The number of databases should equal the number of classes in the ontology file. Although the user can write complicated rules in the rules files to access the attribute values in the databases and perform operations, it is recommended that the user preprocesses the databases and writes simpler rules.

ILLUSTRATIVE EXAMPLES

Two illustrative examples are presented. One is a small and hypothetical bridge network and the other is the bridge network in Miami-Dade County. The former is smaller and is used to better explain the methodology. The two examples share the same data sources including the databases of bridges, river gages, and traffic monitoring sites. They also share the same ontology, which is shown in Figure 6. The ontology is established to monitor the risk of bridges. The bridge risk can be obtained by multiplying the failure probability and failure loss (Cheng and Frangopol 2023; Liu et al. 2020), so we link the two attributes Bridge.failure_probability (B.fp) and Bridge.failure_loss (B.fl) to the attribute Bridge.risk (B.r). The superstructure and substructure condition ratings are related to the reliability of a bridge (Cheng and Frangopol 2021), therefore, links are added from the two attributes Bridge.substruct_rating (B.subr) and Bridge.superstruct_rating (B.supr) to the attribute B.fp. For riverine bridges, scour is one of the important failure modes and the failure probability can be affected by the river flow and stage (Liu et al. 2020). To better monitor the failure probability, we link the River_gage.flow (RG.f) and River_gage.stage (RG.s) to B.fp. After a bridge fails, it will cause detour. Hence, the annual average daily traffic (AADT) of the nearby short term count sites (STCS) can be used to update the detour traffic and the failure consequences, which is also reflected in the ontology. After drawing the graphical structure of the ontology, we now designate the attributes for creating clusters. First, the flow of river gages at different locations may be correlated. In other words, the variables of type RG.f can be correlated. Similarly, the

stages of river gages at different locations may be correlated. Therefore, these two attributes (i.e., RG.f and RG.s) are designated as self-correlated attributes. Second, the flow and stage of the same river gage can be correlated. Therefore, the two attributes are inter-correlated.

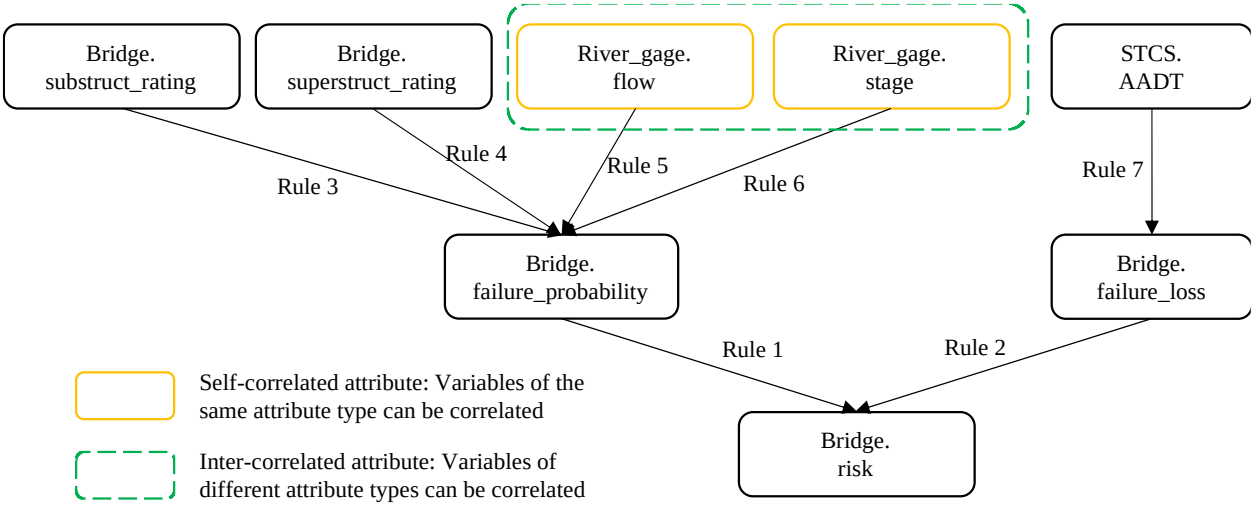


Figure 6. Ontology for illustrative examples

Example 1: small network

Figure 7 shows the locations of the bridges, river gages, and STCSs of the hypothetical network. More information is shown in Table 1. Please note that the failure probability, failure loss, and risk are calculated instead of data sources, therefore, the attribute values are not shown in Table 1. This small and hypothetical example is used to demonstrate how to generate a KG.

Table 1a. Information regarding the bridges

ID	County	Route	Under	Y axis	Substruct. rating	Superstruct. rating	Failure probability	Failure loss	Risk
11	Tompkins	13	A St	1	8	7	--	--	--
12	Tompkins	13	B St	3	9	6	--	--	--
13	Tompkins	13	C St	5	7	8	--	--	--
14	Tompkins	13	D St	7	6	9	--	--	--
15	Tompkins	13	E St	9	5	7	--	--	--
16	Tompkins	13	F St	13	7	5	--	--	--
17	Tompkins	13	G St	17	9	8	--	--	--
18	Tompkins	13	River 2	19	8	9	--	--	--
19	Cortland	81	S St	25	8	8	--	--	--
20	Tompkins	13	River 2	28	7	8	--	--	--
21	Tompkins	13	River 2	30	8	7	--	--	--

Table 1b. Information regarding river gages

ID	River	Y axis	Flow	Stage
1	River 1	9	30	3
2	River 2	18	8	3
3	River 2	19	9	4
4	River 2	27	13	5
5	River 2	29	13	7

Table 1c. Information regarding STCS

ID	Route	Y axis	AADT
1	13	2	5000
2	13	4	5200
3	13	6	4500
4	13	8	5500
5	81	15	8000
6	13	18	6000
7	13	20	7000
8	13	29	6000
9	13	31	5000

For the ontology in Figure 6, there exists one leaf node and seven edges. Therefore, the user should provide one rule for the leaf node and seven rules for the edges, which are shown as rules 1 to 7 in Figure 6. Herein, the rules are first presented. For the leaf node B.r, we stipulate that the bridge should be in Tompkins county. For rules 1 to 4, to connect two variables of different attribute types, they should belong to the same object. For rules 5 and 6, there are two criteria to connect two variables. First, what goes under the bridge should be the river that the river gage is measuring. Second, the distance between the bridge and the river gage should be equal to or smaller than 1. For rule 7, there are also two criteria to connect two variables. First, the bridge should be on the route that the STCS is measuring. Second, the distance between the bridge and the STCS should be equal to or smaller than 1.

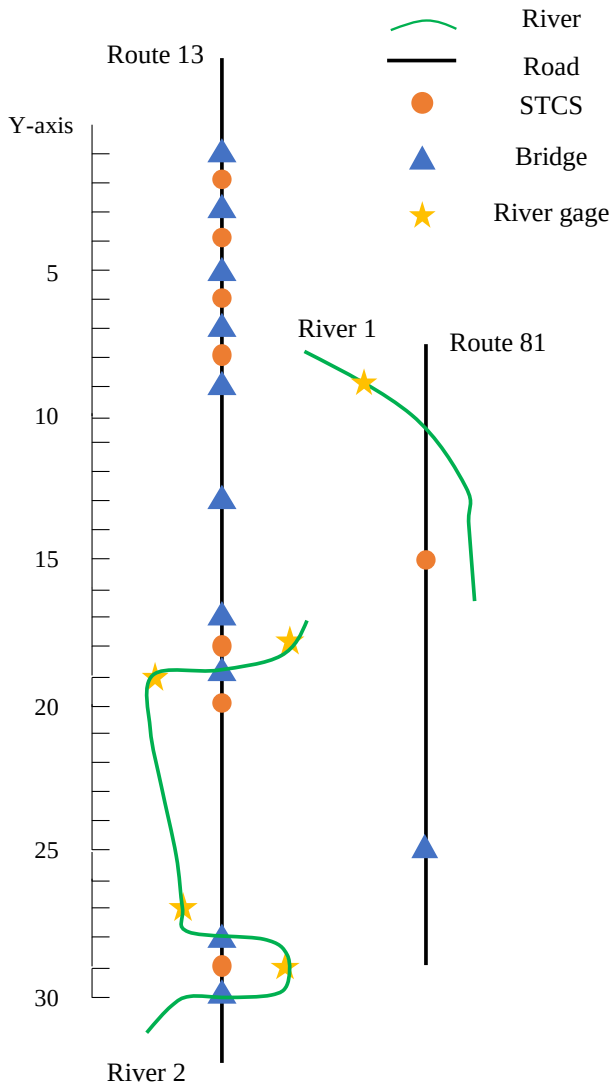


Figure 7. Map of the hypothetical network of Example 1

With the rules of the first phase, we generate a KG. Figure 8 shows the KG with each color representing one attribute type. Please note that to better explain how variables are connected, IDs of some variables are shown. Based on the rule for the leaf nodes and Table 1a, Bridge 19 (B[19]) does not match the criterion. Therefore, we only add the risk of the remaining 10 bridges to the KG, which are the 10 blue nodes at the bottom of Figure 8. The IDs of the variables are also shown. For rules 1 and 2, we connect the variables that are of the correct attribute types and belong to the same object. Using Bridge 21 as an example, B[21].fp and B[21].fl are linked to B[21].r. Same operations are performed for the other bridges, so 10 variables of B.fp and 10 variables of B.fl are connected to 10 variables of B.r, respectively. For rules 3 and 4, the same logic is applied to add

edges. For example, B[11].supr and B[11].subr are linked to B[11].fp. For rules 5 and 6, we find that five pairs of bridges and river gages exist, which are (B[18], RG[2]), (B[18], RG[3]), (B[20], RG[4]), (B[20], RG[5]), and (B[21], RG[5]). Please note that although the distance criterion is satisfied for B[17] and RG[2], this pair does not exist because river 2 does not go under B[17]. As shown in Figure 8, for each pair, the corresponding RG.s and RG.f will be linked to B.fp. For example, edges exist between RG[2].f and B[18].fp and between RG[2].s and B[18].fp. For rule 7, there exist 14 pairs of bridges and STCSs and we then connect the corresponding B.fl and STCS.AADT. For example, B[11] and B[12] share STCS[1] and edges exist between STCS[1].AADT and B[11].fl and between STCS[1].AADT and B[12].fl.

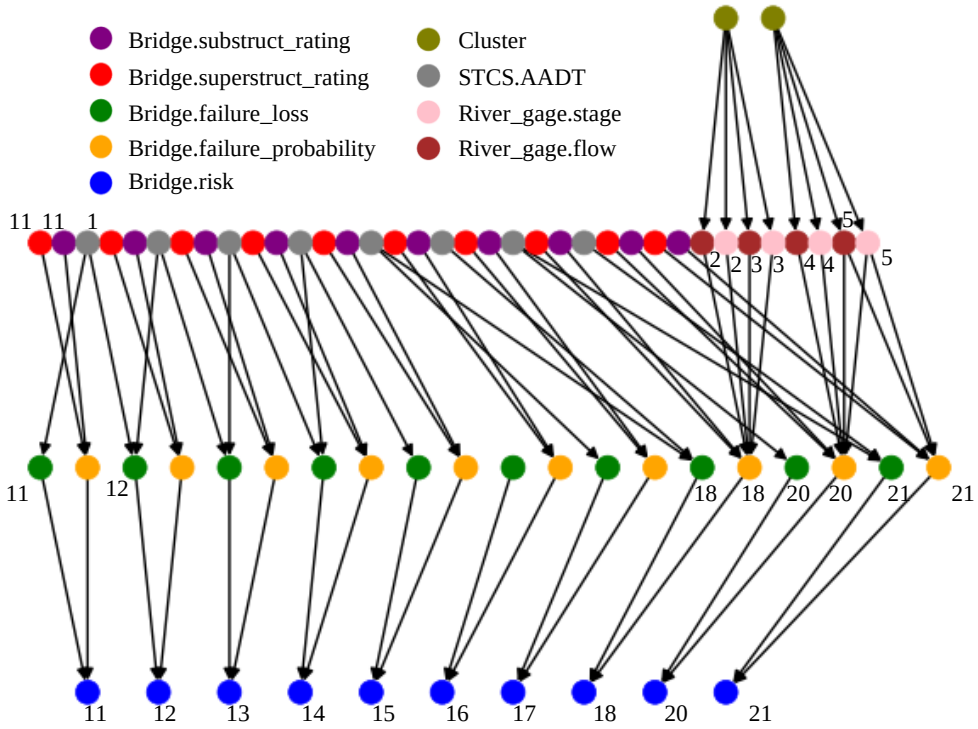


Figure 8. KG of Example 1

Now we move to the second phase. Since there are two self-correlated attributes and one sublist of inter-correlated attributes, the user should provide three rules. For RG.f, suppose there is a new variable and an existing cluster of variables. If there exists one variable in the cluster so that the distance between this variable and the new variable is equal to or smaller than 1, then the

new variable can be added to the cluster. The same rule is applied to RG.s. The sublist includes RG.f and RG.s. Herein, only the default rule is applied. The rule requires that the variables of the correct attribute types can be put into one cluster only if they belong to the same object. In this case, the variables should belong to the same river gage.

After applying the rules for the self-correlated attributes, four clusters are obtained: (RG[2].f, RG[3].f), (RG[2].s, RG[3].s), (RG[4].f, RG[5].f), and (RG[4].s, RG[5].s). Then the default rule is applied for the inter-correlated attributes and another four clusters are obtained: (RG[2].f, RG[2].s), (RG[3].f, RG[3].s), (RG[4].f, RG[4].s), and (RG[5].f, RG[5].s). Finally, the aforementioned clusters are unionized and the results are two clusters: (RG[2].f, RG[2].s, RG[3].f, RG[3].s) and (RG[4].f, RG[4].s, RG[5].f, RG[5].s). To represent the statistical correlations of the two clusters, additional variables are added to the link the clusters, as shown in Figure 8.

Example 2: big network

Example 2 uses the same ontology in Figure 6. In this example, a KG is created for Miami-Dade County. Similar to example 1, three databases are used. The bridge database is from National Bridge Inventory database (FHWA 1992), while the STCS database is from Florida Department of Transportation (FDOT) (FDOT 2023). When downloaded, the two databases are for the whole state, but we can select the bridges and STCSs in Miami-Dade County based on the county code. Please note that in Florida an STCS is called portable traffic monitoring site, but for consistency in this paper, we continue to use the term STCS. The database for river gages is from US Geological Survey (2016), however we cannot only select the river gages in the County due to the lack of a county code. Preprocessing of these databases is performed so that the longitudes and latitudes are consistent across the databases.

The attributes used in this example are shown in Table 2. Specifically, the attributes that are mentioned in the ontology will be added to the graph, while Table 2 also indicates the attributes that are used to define the rules. In general, the logic of rules is the same for examples 1 and 2. Rules 1 to 4 are the same for examples 1 and 2. Rules 5 and 6 are slightly different. There are two similar criteria. As in example 1, the first criterion requires that the river that intersects the bridge should be the one that the river gage is measuring. The second criterion requires that the distance between the bridge and the river gage should be equal to or smaller than 5km. The first criterion is satisfied when the attribute “FEATURES_DESC_006A” of a bridge and “STANAME” of a river

gage indicate the same river. The second one is satisfied when the distance calculated using the longitudes and latitudes of the bridge and river gage is smaller than 5km. Please note that errors can occur when automatically evaluating the first criterion because the river names are documented slightly differently in different databases. For rule 7, the 10 closest STCSs to the bridge are selected.

Table 2. Attributes used in example 2

Database	Attributes used in rules	Attributes in the Ontology
Bridge	FEATURES_DESC_006A; Longitude; Latitude	Substruct. rating; Superstruct. rating; Failure probability; Failure loss; Risk
River gage	STANAME; Longitude; Latitude	Flow; Stage
STCS	Longitude; Latitude	AADT

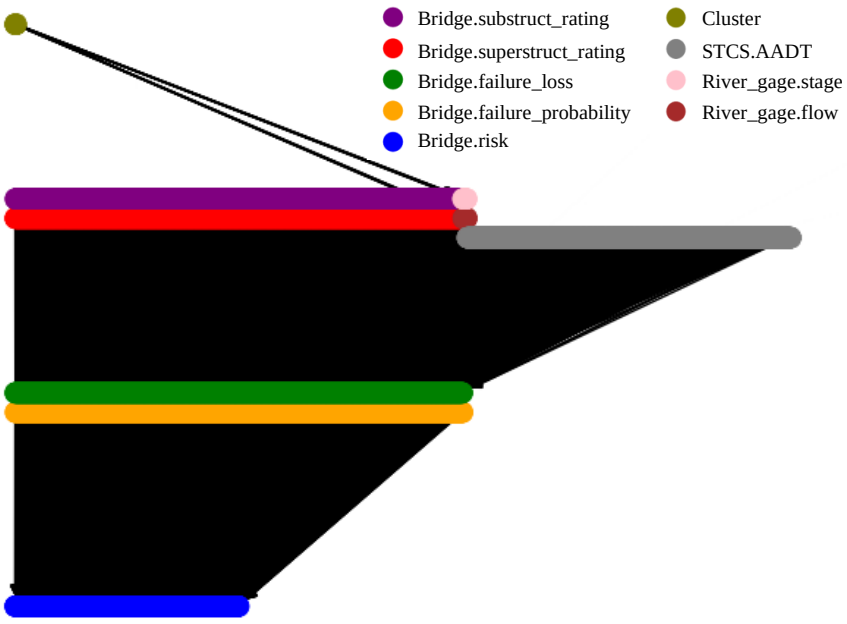


Figure 9. KG of Example 2: Monitor the risk of all bridges in Miami-Dade County.

Note: Data sources include bridge condition ratings from National Bridge Inventory, stage and flow data from river gages, and annual average daily traffic from traffic count stations

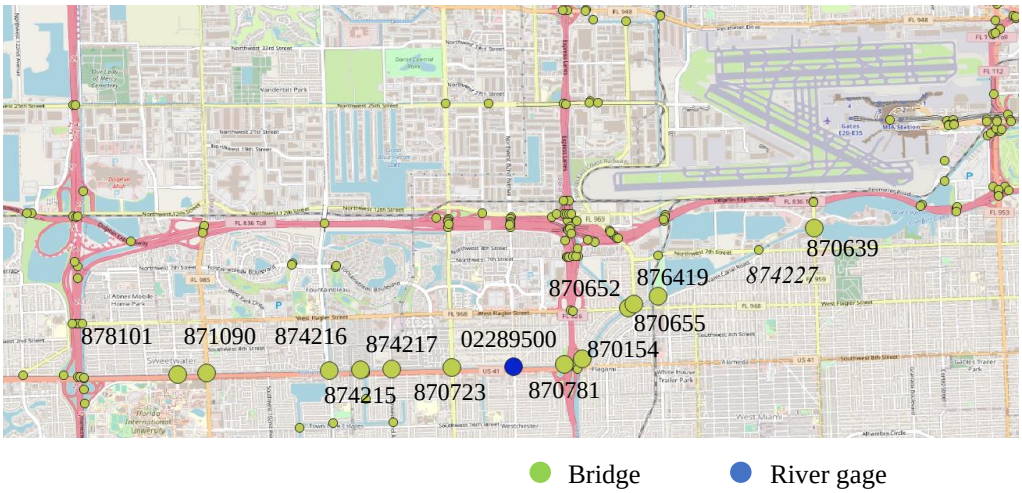
For self-correlated attributes, there are two criteria. The first one requires that the two river gages should measure the same river, while the second one requires that the two river gages should be within 2km. For inter-correlated attributes, only the default rule is used.

Figure 9 shows the KG of the whole Miami-Dade County. The KG has 6,692 nodes and 14,700 edges. It connects 1,034 bridges, 10 river gages, and 1,496 STCSs to monitor the risk of all bridges in the county. We verify the KG by looking at certain locations in the county. To illustrate the process, we look at the river gages, bridges, and STCSs near Miami Internation Airport. Figure 10(a) shows that river gage 02289085 is linked to bridges 878101, 871090, 874216, 874215, 874217, 870723, 870781, 870154, 870652, 870655, 876419, and 870639. These bridges are within 5km to the river gage and are on the same river which the river gage measures. There is only one exception with bridge 874227 which is also on the same river as we zoom in. The reason is that the river names in two databases do not match. Specifically, bridge 874227 is over "Canal C-4", while the name of the river is called "Tamiami Canal near Coral Gables". This shows the limitation of the framework. When the rules are complicated such as involving natural language processing, errors can occur. Figure 10(b) shows how bridges are linked to STCSs. We have verified that the STCSs in the blue polygon are the ten closest to the bridges. Although errors sometimes occur (e.g., in Figure 10(a)), AutoGraCS can efficiently generate a KG for a large region. The results from the two examples also show that once we develop an appropriate ontology and a set of rules, AutoGraCS can reuse them to generate the KG at an even larger scale.

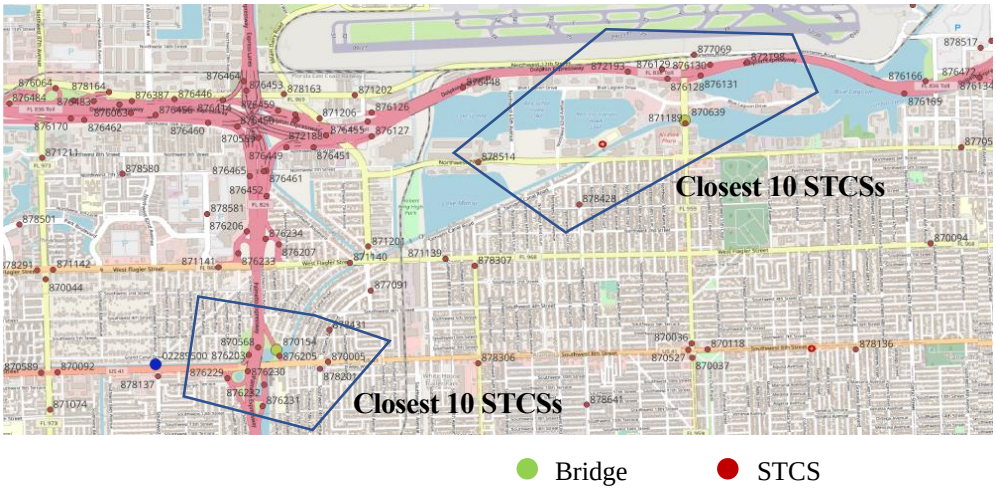
RECOMMENDATIONS AND CONCLUSIONS

This paper proposes a framework **Auto**mated Knowledge **Gra**ph for **C**omplex **S**ystems (AutoGraCS). Using AutoGraCS, knowledge graphs (KGs) can be automatically generated given three types of user inputs, including an ontology, a set of rules, and databases. The ontology representing the expert opinions identifies the attributes for the analysis and defines their relationships. The set of rules stipulate the conditions to determine whether variables of corresponding attributes in the ontology can be linked or clustered. The databases contain the objects of different systems that we analyze and must include the attributes shown in the ontology. AutoGraCS has two phases. The first phase is to connect all the variables that match the corresponding attributes in the ontology and the set of rules, while the second phase is to cluster the statistically correlated variables. The established KG can be easily turned into a Bayesian network to assist in the development of digital twins. Two numerical examples are presented and the following conclusions are drawn:

- The first example is a hypothetic and small bridge network. The KG generated by AutoGraCS connects and clusters the variables of 10 bridges, 10 STCSs, and 4 river gages. It matches what we expect, showing that AutoGraCS correctly generates the KG given the user's inputs.
- The second example is a big bridge network in Miami-Dade County. The KG generated by AutoGraCS has 6,692 nodes and 14,700 edges. It connects and clusters the variables of 1,034 bridges, 10 river gages, and 1,496 STCSs. We verify the KG by looking at certain locations. The verifications show that AutoGraCS follows the set of rules to correctly link the bridges with the river gages and identify the closest STCSs around the bridges. The second example is a KG for a whole county, indicating the scalability of AutoGraCS.
- The two examples share the same ontology and a similar set of rules. Using AutoGraCS, the expert opinions can be reused to repeat the same analysis for other regions. This implies that AutoGraCS can assist in extending the analysis to other counties or cities to cover a bigger region.
- Errors occur in the second example because the river names documented in different databases are different. Future research is needed to upgrade the reasoning ability of AutoGraCS so that it can detect the anomalies and make the necessary corrections.
- The computational time increases exponentially as the number of objects increases. Further research is necessary to improve the computational efficiency so that AutoGraCS can generate KGs for states or the nation.



(a)



(b)

Figure 10. Illustration and verification of the KG around Miami International Airport: (a) links between a river gage and its nearby bridges; (2) links between the bridges and the 10 closest traffic count stations

DATA AVAILABILITY STATEMENT

All source codes, data, and outputsthat support the findings of this study are in the Zenodo DataShare repository with the identifier https://doi.org/10.5281/zenodo.12741321. Data will be publicly accessible within one year of generation.

REFERENCES

Akroyd, J., S. Mosbach, A. Bhave, and M. Kraft. 2021. “Universal digital twin-a dynamic knowledge graph.” Data-Centric Engineering, 2: e24.

Applegate, C. J., and I. Tien. 2019. “Framework for probabilistic vulnerability analysis of interdependent infrastructure systems.” Journal of Computing in Civil Engineering, 33 (1): 04018058.

Austin, M., P. Delgoshaei, M. Coelho, and M. Heidarinejad. 2020. “Architecting smart city digital twins: Combined semantic model and machine learning approach.” Journal of Management in Engineering, 36 (4): 04020026.

Bensi, M., A. Der Kiureghian, and D. Straub. 2011. “Bayesian network modeling of correlated random variables drawn from a Gaussian random field.” Structural Safety, 33 (6): 317–332.

Bensi, M., A. Der Kiureghian, and D. Straub. 2013. “Efficient Bayesian network modeling of systems.” Reliability Engineering & System Safety, 112: 200–213.

Bensi, M. T. 2010. A Bayesian network methodology for infrastructure seismic risk assessment and decision support. University of California, Berkeley.

Bismut, E., and D. Straub. 2021. “Optimal adaptive inspection and maintenance planning for deteriorating structural systems.” Reliability Engineering & System Safety, 215: 107891.

Bocchini, P., and D. M. Frangopol. 2011. “A stochastic computational framework for the joint transportation network fragility analysis and traffic flow distribution under extreme events.” Probabilistic Engineering Mechanics, 26 (2): 182–193.

Broo, D. G., M. Bravo-Haro, and J. Schooling. 2022. “Design and implementation of a smart infrastructure digital twin.” Automation in Construction, 136: 104171.

Byun, J.-E., and D. D’Ayala. 2022. “Urban seismic resilience mapping: a transportation network in Istanbul, Turkey.” Scientific reports, 12 (1): 8188.

Byun, J.-E., K. Zwirglmaier, D. Straub, and J. Song. 2019. “Matrix-based Bayesian Network for efficient memory storage and flexible inference.” Reliability Engineering & System Safety, 185: 533–545.

Chen, X., S. Jia, and Y. Xiang. 2020. “A review: Knowledge reasoning over knowledge graph.” Expert Systems with Applications, 141: 112948.

Cheng, M., and D. M. Frangopol. 2021. “Optimal load rating-based inspection planning of corroded steel girders using Markov decision process.” Probabilistic Engineering Mechanics, 66: 103160.

Cheng, M., and D. M. Frangopol. 2023. “Efficient scenario analysis for optimal adaptation of bridge networks under deep uncertainties through knowledge transfer.” Structural Safety, 100: 102278.

Cheng, M., and H. O. Gao. 2023. “Data-driven life-cycle risk assessment of bridge networks using Bayesian network.” 8th International Symposium on Life-Cycle Civil Engineering, IALCCE 2023. CRC Press/Balkema.

Cheng, M., and H. O. Gao. 2024. “Performance-oriented system digital twin of infrastructure systems using Bayesian network.” Structure and Infrastructure Engineering. (submitted)

Coelho, M., M. A. Austin, and M. R. Blackburn. 2017. “Semantic behavior modeling and event-driven reasoning for urban system of systems.” International Journal on Advances in Intelligent Systems, 10 (3): 365–382.

Eibeck, A., M. Q. Lim, and M. Kraft. 2019. “J-Park Simulator: An ontology-based platform for cross-domain scenarios in process industry.” Computers & Chemical Engineering, 131: 106586.

El-Gohary, N. M., and T. E. El-Diraby. 2010. “Domain ontology for processes in infrastructure and construction.” Journal of construction engineering and management, 136 (7): 730–744.

Errandonea, I., S. Beltrán, and S. Arrizabalaga. 2020. “Digital Twin for maintenance: A literature review.” Computers in Industry, 123: 103316.

FDOT. 2023. FDOT Traffic Monitoring Handbook. FDOT, Tallahassee, FL.

FHWA. 1992. National Bridge Inventory (NBI). FHWA, Washington, DC.

Fonseca, Í. A., and H. M. Gaspar. 2021. “Challenges when creating a cohesive digital twin ship: a data modelling perspective.” Ship Technology Research, 68 (2): 70–83.

France-Mensah, J., and W. J. O’Brien. 2019. “A shared ontology for integrated highway planning.” Advanced Engineering Informatics, 41: 100929.

Groden, M., and M. Collette. 2017. “Fusing fleet in-service measurements using Bayesian networks.” Marine Structures, 54: 38–49.

Guo, D., R. Y. Zhong, P. Lin, Z. Lyu, Y. Rong, and G. Q. Huang. 2020. “Digital twin-enabled Graduation Intelligent Manufacturing System for fixed-position assembly islands.” Robotics and Computer-Integrated Manufacturing, 63: 101917.

Hofmeister, M., G. Brownbridge, M. Hillman, S. Mosbach, J. Akroyd, K. F. Lee, and M. Kraft. 2023. “Cross-domain flood risk assessment for smart cities using dynamic knowledge graphs.” Sustainable Cities and Society, 105113.

Jiang, Y., M. Li, W. Wu, X. Wu, X. Zhang, X. Huang, R. Y. Zhong, and G. G. Huang. 2023. “Multi-domain ubiquitous digital twin model for information management of complex infrastructure systems.” Advanced Engineering Informatics, 56: 101951.

Johansen, C., and I. Tien. 2018. “Probabilistic multi-scale modeling of interdependencies between critical infrastructure systems for resilience.” Sustainable and Resilient Infrastructure, 3 (1): 1–15.

Kapteyn, M. G., J. V. Pretorius, and K. E. Willcox. 2021. “A probabilistic graphical model foundation for enabling predictive digital twins at scale.” Nature Computational Science, 1 (5): 337–347.

Kunzer, B., M. Berges, and A. Dubrawski. 2022. “The Digital Twin Landscape at the Crossroads of Predictive Maintenance, Machine Learning and Physics Based Modeling.” arXiv preprint arXiv:2206.10462.

Le, T., and H. D. Jeong. 2016. “Interlinking life-cycle data spaces to support decision making in highway asset management.” Automation in construction, 64: 54–64.

Lin, K., Y.-L. Xu, X. Lu, Z. Guan, and J. Li. 2021. "Digital twin-based collapse fragility assessment of a long-span cable-stayed bridge under strong earthquakes." Automation in Construction, 123: 103547.

Liu, L., D. Y. Yang, and D. M. Frangopol. 2020. "Network-level risk-based framework for optimal bridge adaptation management considering scour and climate change." Journal of Infrastructure Systems, 26 (1): 04019037.

Marcot, B. G., and T. D. Penman. 2019. "Advances in Bayesian network modelling: Integration of modelling technologies." Environmental modelling & software, 111: 386–393.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, and V. Dubourg. 2011. "Scikit-learn: Machine learning in Python." the Journal of machine Learning research, 12: 2825–2830.

Pregnolato, M., S. Gunner, E. Voyagaki, R. De Risi, N. Carhart, G. Gavriel, P. Tully, T. Tryfonas, J. Macdonald, and C. Taylor. 2022. "Towards Civil Engineering 4.0: Concept, workflow and application of Digital Twins for existing infrastructure." Automation in Construction, 141: 104421.

Savage, T., J. Akroyd, S. Mosbach, N. Krdzavac, M. Hillman, and M. Kraft. 2022. "Universal Digital Twin: Integration of national-scale energy systems and climate data." Data-Centric Engineering, 3: e23.

Shim, C.-S., N.-S. Dang, S. Lon, and C.-H. Jeon. 2019. "Development of a bridge maintenance system for prestressed concrete bridges using 3D digital twin model." Structure and Infrastructure Engineering, 15 (10): 1319–1332.

Tien, I., and A. Der Kiureghian. 2016. "Algorithms for Bayesian network modeling and reliability assessment of infrastructure systems." Reliability Engineering & System Safety, 156: 134–147.

US Geological Survey, U. G. 2016. National water information system data available on the world wide web (USGS water data for the nation). USGS Surface-Water Annual Statistics for Wisconsin.

Vieira, J., J. Poças Martins, N. Marques de Almeida, H. Patrício, and J. Gomes Morgado. 2022. "Towards Resilient and Sustainable Rail and Road Networks: A Systematic Literature Review on Digital Twins." Sustainability, 14 (12): 7060.

Wagg, D. J., K. Worden, R. J. Barthorpe, and P. Gardner. 2020. "Digital twins: state-of-the-art and future directions for modeling and simulation in engineering dynamics applications." ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part B: Mechanical Engineering, 6 (3): 030901.

Wu, C., P. Wu, J. Wang, R. Jiang, M. Chen, and X. Wang. 2021. "Ontological knowledge base for concrete bridge rehabilitation project management." Automation in construction, 121: 103428.

Xiong, M., and H. Wang. 2022. "Digital twin applications in aviation industry: A review." The International Journal of Advanced Manufacturing Technology, 121: 5677–5692.

Yoon, S., S. Lee, S. Kye, I.-H. Kim, H.-J. Jung, and B. F. Spencer. 2022. "Seismic fragility analysis of deteriorated bridge structures employing a UAV inspection-based updated digital twin." Structural and Multidisciplinary Optimization, 65 (12): 346.

Zhang, S., F. Boukamp, and J. Teizer. 2015. "Ontology-based semantic modeling of construction safety knowledge: Towards automated safety planning for job hazard analysis (JHA)." Automation in Construction, 52: 29–41.

Zhu, J., and M. Collette. 2017. "A Bayesian approach for shipboard lifetime wave load spectrum updating." Structure and Infrastructure Engineering, 13 (2): 298–312.