

An open-source and extensible platform for general infrastructure asset management system

Vahid Asghari¹ and Shu-Chien Hsu^{1,*}

¹Department of Civil and Environmental Engineering, The Hong Kong Polytechnic University, Hong Kong.

*Corresponding Author; E-mail: mark.hsu@polyu.edu.hk

Abstracts: Given the importance of maintaining infrastructure assets, asset management at both project- and network-levels has been the focus of hundreds of studies. Although these studies and previously developed asset management systems have pushed the boundaries of this field, these advances have been made separately without the ability to build upon one another. This paper puts forward the first Python-based, open-source, extensible, freely accessible, and modular platform, GIAMS, paving the way for researchers to easily collaborate and contribute to future related research. Drawing upon related literature, this paper describes modules of GIAMS and illustrates their use with bridge components and models. One project-level lifecycle optimization and one network-level project-selection based on the Indiana, US, bridge network with more than 4,600 bridges are provided to demonstrate the applicability of GIAMS. The modular and open-source nature of GIAMS makes it readily capable of further extension in a variety of asset management topics.

Keywords: asset management, life cycle optimization, network-level optimization, open-source

1 Introduction

Reliable, well-functioning, and durable community infrastructure assets are requirements for economic growth and prosperity in the modern world [1]. The condition of these assets is constantly altered by aging, loading, environmental agents, or even natural and man-made hazards. Agencies, asset managers, and decision-makers use asset management systems to maintain their assets in acceptable conditions. Asset management refers to the process of condition monitoring, collecting data, analyzing, and decision making about resource allocation to the maintenance of assets [2]. Asset management systems (AMS) must answer the following questions: 1) which components need to be maintained, 2) when, and 3) to what extent the intervention should be carried out [3] (i.e., regular maintenance, rehabilitation, or reconstruction(MRR)). Optimized and powerful asset management systems that incorporate up-to-date and advanced methodologies could reduce agency costs and benefit the applicable community. This underscores the importance of improving asset management systems globally [4,5]. However, asset management is a difficult task for stakeholders mainly because of insufficient high-quality data, infrastructure vulnerabilities, varying levels of asset response to future hazards, and restoration challenges after damage [6]. To tackle these challenges, hundreds of research papers over the past decades have tried to provide solutions to various problems and improve AMS [7,8]. Apart from the research papers, commercial computer programs [9,10] have also been developed for the maintenance of infrastructure assets.

The abovementioned asset management systems are associated with different strengths and limitations. However, there are generally three main limitations shared by all these AMSs. First and foremost, they are not extensible. This means one cannot easily update a component of others' asset management systems for further research and analysis. As a result, whoever

intends to conduct a study in the asset management field must develop AMS almost from scratch. Second, the programming languages of these AMSs that are being used in different departments of transportations are not flexible enough. Consequently, it is too difficult, and sometimes impossible, to apply new tools and models, such as machine learning models, on them. Third, some of the AMSs that are used by agencies are licensed and expensive. Therefore, researchers in some parts of the world do not have access to the currently used AMSs. In addition, municipalities in underprivileged countries will never be able to use up-to-date AMSs for their assets due to the above-described constraints. Given these limitations, there is a need for an open-source asset management platform built with widely used and flexible programming languages such as Python.

The idea of open collaboration in asset management could unleash new opportunities. Therefore, the major objective of this paper is to present an open-source asset management platform that is available for further development through open collaboration among researchers and programmers from both academic and practice settings. This platform is called the “general infrastructure asset management system” (GIAMS). Using bridges as an example asset, this article introduces the major blocks of the python-based platform for both project and network level asset management. In addition to a technical overview, guidelines for the future development of these blocks from a programming perspective are also provided. These modular blocks are written in Python with an objective-oriented mindset and the intention of facilitating the extension of GIAMS. Given the nature of the asset management problems from a programming perspective, the builder method has been chosen as the design pattern of GIAMS. The builder method is a design pattern that is mostly used when a final product (e.g., an AMS) aggregates several other blocks and modules to share various tasks and responsibilities among them [11]. The design and development of these blocks are mostly based on previously used methods and algorithms of asset management systems (AMS). Currently, GIAMS can provide optimized maintenance, rehabilitation, and reconstruction plans for the life cycle of an asset. It can also yield a prioritized portfolio of assets that need to be maintained or rehabilitated in a network. With minor modifications, it can provide optimized structural health monitoring schedules. The Python code of GIAMS is freely accessible online in a GitHub repository (<https://github.com/vd1371/GIAMS>) [12], and ready for use and further development. Given the inextensibility and licensed nature of previously designed AMSs written in less flexible programming languages, the main contribution of this study is to provide a freely accessible asset management platform for researchers focusing on this area of research. This platform can eventually accelerate studies on asset management globally and could also be used in under-privileged communities to support more informed, methodical, sustainable, and profitable decision making.

The applicability of GIAMS is illustrated through two examples based on the Indiana, US, bridge network derived from the National Bridge Inventory (NBI). Realistic models and data from NBI, IBMS [2], HAZUS [13], and USGS [14] were used to analyze these two examples. In the first example, the life cycle cost of one bridge in a 20-year management horizon was optimized by the genetic algorithm and Monte Carlo simulation. In the second example, a portfolio of suggested MRR plans for all bridges, which is a multichoice multidimensional knapsack problem, was optimized using the incremental utility cost (IUC) heuristic method. While these examples are focused on bridges and bridge networks, GIAMS could be further developed and expanded to be used in other asset management areas such as pavement management. The rest of this paper is structured as follows. In section 3, a relatively thorough description of the modules and sub-models usually used in asset management systems as developed in the GIAMS structure is provided. In section 4, the illustrative examples based on the Indiana bridge network are described to show a part of the current capabilities of GIAMS.

In section 5, the results of the illustrative examples are provided, and section 6 is allocated to the discussion and summary. Technical aspects of open collaboration, key terminologies, and main processes in open collaboration for software development are briefly summarized in Appendix A. Appendix B provides examples for extending this framework using three cases: 1) development of a new deterioration model, 2) prioritizing seismic retrofit of structure, and 3) maintenance optimization with structural health monitoring results. The location and role of all classes and objects which will be discussed later in the manuscript can be found in the directory tree in Appendix C.

2 Background research

In this section, an overview of previous studies on open-source computer programs in civil engineering, open collaboration ethos, and bridge management systems are provided.

2.1 Open-source civil engineering computer programs

Several computational programs for civil engineering problems have been developed in the past decade. For example, Mahsuli and Haukaas [15] developed *Rt*, a freely available computer program, for reliability analysis and probabilistic modeling with a focus on the seismic analysis of different structures. This computer program has been since under development and upgrading. Pagani et al. [16] developed an open-source platform called *OpenQuake* engine for risk calculation and hazard analysis of earthquakes globally. *OpenQuake* is also currently under development and extension with open collaboration on GitHub. *Optimist* is another open-source Python library that was developed by Raso et al. [17] to ease stochastic dual dynamic programming in water systems operation and analysis. Warren et al. [18] developed another open-source software for simulating the electromagnetic wave propagation in soil by numerical models. Gadi et al. [19] proposed a Python program for image processing to evaluate soil moisture content. *OpenSeesPy* was developed by Zhu et al. [20] for finite element analysis of structural and geotechnical systems and their response to seismic hazards in Python. Guan et al. [21] developed another computational Python-based platform called *AutoSDA* to facilitate the seismic design, modeling, and analysis of steel moment frames. Similarly, attempts have been made and other Python-based platforms have been developed in the structural engineering area (e.g., structural health monitoring [22], materials resistance analysis [23]). *PySWMM* was developed by McDonnell et al. [24] to model and analyze stormwater and related phenomena in Python. Zeraoui et al. [25] developed another program for the optimization of various processes in managing dredged sediments. Developing computer programs, and more specifically open-source and Python-based platforms have gained more attention in the past few years. This philosophy, however, is yet to become popular and widely used in civil engineering, and more specifically in the construction management field.

2.2 Open collaboration

The open collaboration approach to software engineering has been in practice for years. Perhaps one of the first and greatest examples of open-source projects is the Linux kernel [26], which is used in billions of devices around the world. More than ten thousand collaborators, most of whom have no direct acquaintance with each other, have contributed to this project since the early 90s. In open collaboration, contributors develop valuable features, reuse, and share each other's output, work purposefully toward a common goal, and permit anyone to contribute and consume [27]. Such an ethos enables collaborators around the world to directly build upon each other's work, interact with each other, and share their knowledge and resources. Consequently, it accelerates the speed of developing any item, lowers costs, and

140 opens the gate for new advancements. A similar approach has already been adopted in most
141 Python libraries and packages for scientific and industrial purposes [28,29]. So far, however,
142 there has been little effort devoted to providing an open-source asset management system to be
143 employed both in research and practice. The developed framework in this study (GIAMS)
144 intends to fill this gap. This paper uses a bridge as an example asset to describe the main
145 modules and components of asset management systems currently provided in GIAMS.

146 **2.3 Example asset: Bridge**

147 Bridges, as one of the most important components of transportation systems, have gained
148 particular attention in the asset management literature. A famous bridge management system,
149 PONTIS [9], was developed in the 1990s under the sponsorship of the Federal Highway
150 Administration and has been widely used in the US. It provides a network-level bridge
151 management strategy and optimizes a multitude of maintenance, rehabilitation, and
152 reconstruction (MRR) projects. During the same period, and for the same type of asset,
153 BRIDGIT [10] was developed for smaller networks. It could provide the decision-makers both
154 network-level and project-level analysis of the assets. PONTIS and BRIDGIT have long been
155 used in several departments of transportation in the US and have been shown to be very
156 practical. In the absence of highly sophisticated computers, these asset management systems
157 could provide rational strategies based on theoretically-well founded methodologies. However,
158 bridge management systems (BMSs) have been advancing both in practice and in the literature.
159 Almost ten years later, for example, the Indiana bridge management system (IBMS) received
160 a major update [2]. In this BMS, user costs, agency costs, MRR costs, deterioration models,
161 etc. were developed in a detailed manner. Similarly, PONTIS has been replaced by
162 AASHTOWare in recent years. Several studies [30–33] have focused on the methodologies
163 and shortcomings of BMSs. Similar research has also been conducted on other assets such as
164 pavement [34], geotechnical assets [35], and sewage systems [36]. But, there is currently no
165 open-source and freely accessible platform written by a flexible programming language for
166 asset management.

167 **3 Modules of the framework**

168 In this section, an overview of the main modules of GIAMS's management framework is
169 provided. Four major modules shape the current structure of this framework. First is the asset
170 (e.g., bridge) module, which consists of several models to explicitly take into account various
171 characteristics and natural phenomena governing the asset. Second is the network module,
172 which aggregates the assets of a network, delineates the limitations (e.g., budget limitation),
173 and yields the objectives for the following modules. The third is the life cycle analyzer module,
174 which evaluates the costs (user costs and agency costs) and performance of an asset (based on
175 the utility theory) in a life cycle with the help of a simulator. Fourth is the optimizer module,
176 which provides an optimal or near-optimal MRR plan for agencies and decision-makers. Fig.
177 1 depicts these modules and the design pattern of this framework.

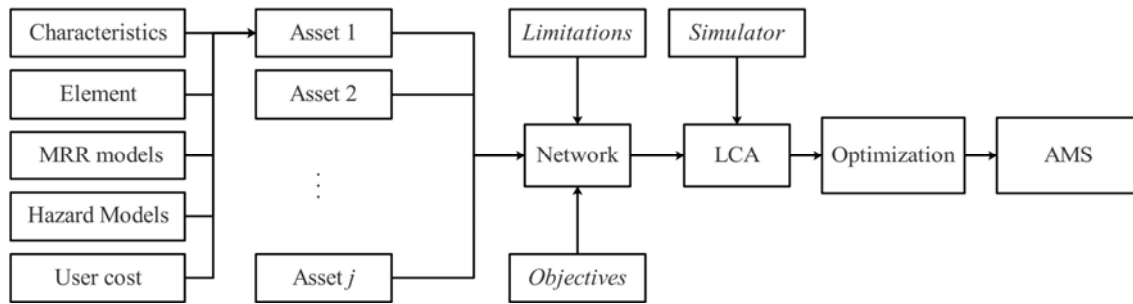


Fig. 1. The logical flow of the modules of the proposed framework

In programming, a design pattern is a solution to a general problem intended to reduce implementation time and increase the efficiency of developing and extending a software program. The design pattern has three major categories (i.e., creational patterns, structural patterns, and behavioral patterns), each of which is suitable for handling a specific problem. The builder method is a subtype of the creational pattern. This method is usually used when a system of multiple components is independent of how the products are created and composed. It is also used when a family of related product objects should be aggregated to create a final product. To further explain the logical flow of **Fig. 1**, the relationships between these blocks are explained in reverse order in the following. The ultimate purpose of an asset management system is to optimize the life cycle costs or utility of a network of assets. The optimization is usually conducted on the results of the life cycle costs analysis (LCA) (e.g., costs and utility) of a network of assets or one asset. Since different practices and simulation technics might be required in the LCA to get different outcomes, a simulator module should be placed in the LCA block. Simulator module makes random realizations of probabilistic events in the life cycle of an asset. Another main component of the LCA is the network of the assets. Each network of assets has its limitations (e.g., monetary limitations) and objectives (e.g., maintaining the condition rating of 90% assets above a certain level). One or some assets could be a part of the network module. Optimization and LCA modules cannot directly receive the limitations and objectives of a network since they are general mathematical and simulation tools for general purposes. Being subject to different hazards in the life cycle, each asset has several elements, characteristics, MRR plan, and a user cost module. The asset's components are thoroughly discussed in section 3.1. It should be noted that different assets experience different magnitudes of hazards that occur in a network. Similarly, other components of the hazard module such as response, loss costs, and recovery plans are also asset-specific models. Therefore, the hazard module is considered as a building block of each asset. Further explanations of these modules can be found in section 3.1.3. Both programming and technical concerns have been considered in designing the relationships between the blocks and the logical flow of GIAMS. Extension and development of this framework consist of writing new blocks guided by a given pattern, performing analysis, validating the code, and sharing with others. Intending to maintain the integrity of future developed modules and codes with objective-oriented programming mindset, a *Base* model is defined for all models. For example, all current and future developed assets must inherit from the *BaseAsset* model. As a result, attributes and methods of future developed objects must use similar terminologies to that of the *Base* models and overwrite them. Similarly, other modules and models should inherit from their corresponding *Base* model in the repository to maintain the integrity in this framework. The idea of inheritance from the *Base* models and using similar terminologies will simplify modification and update of the framework. In other words, the new blocks should imitate the currently developed blocks to perform similar tasks based on similar inputs and return similar outputs. GIAMS' blocks, as

well as their inputs and outputs, are described both from technical and programming perspectives in the following sections.

3.1 Asset

The asset is the principal component of any asset management system. Consisting of several elements, it is affected by several natural or man-made phenomena and incurs costs on the community. In the context of this study and without loss of generality, a bridge, as a transportation lifeline, has been chosen to demonstrate the flow of constructing an asset management system. The asset module has several characteristics and is mainly built upon four sub-models to capture real incidents and phenomena in the modeling procedure. Asset characteristics are attributes that are set in the asset class, while sub-models are instantiated objects assigned to the asset. The characteristics and sub-models employed in this platform are discussed in the following sections.

3.1.1 Asset characteristics

Some parameters and characteristics are shared among various types of assets while some only belong to one type of asset. This makes it necessary to develop an independent sub-model to carefully address the needs for analyzing the intended asset of the study. To illustrate, the seismic response model of bridges differs from that of buildings. Each bridge has traffic-related characteristics (e.g., average daily traffic and road class), seismic characteristics (e.g., site class and HAZUS classification), and other characteristics such as the number of spans, ID, and skew angle. The full characteristics of this module could be found in the repository of GIAMS.

3.1.2 Elements

Assets are typically built upon several elements. For example, bridges have three main sets of elements [37], which one or some of these elements have been used in asset management studies previously. For instance, Yang and Frangopol [38] merely used the structure element, while Sinha *et al.* [2] used deck, substructure, superstructure, and wearing surfaces in their studies. These elements deteriorate over time and need to be monitored, maintained, rehabilitated, or reconstructed. These improvement actions incur costs that are taken on by the relevant agencies based on elements' attributes. In GIAMS, condition monitoring, deterioration models, and agency costs are the input objects/models of the element class. An element object is ultimately assigned to the asset module to be used in the next modules (e. g., LCA module). Fig. 2 depicts the components of a sub-model that should be aggregated to create the element.

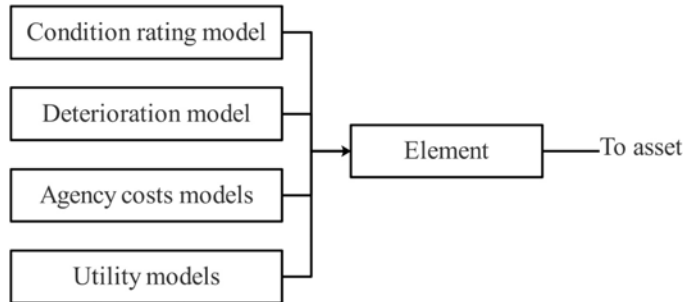


Fig. 2. Elements' sub-models

3.1.2.1 Condition rating

An asset's elements deteriorate over time due to constant exposure to natural agents, chemicals, and loading. Several condition rating schemes have been proposed to monitor the state of an

asset in the face of deterioration. These ratings could be continuous (e.g., the continuous space in [0,1] interval) or discrete (good, fair, poor, severe [37]). In the case of bridges, discrete condition ratings have been used in several famous bridge management systems such as PONTIS [9], BRIDGIT [10], and IBMS [2]. Table 1. summarizes the common discrete numbering in the US, China, Korea, and Japan[39].

Table 1. Common condition rating schemes around the world

NBI rating description	NBI [40]	PONTIS[9]	China	Korea	Japan
As new	9	1	A	A	I
No problems noted	8				
Some minor problems noted	7	2	B	B	II
Structural elements show some minor deterioration	6				
All primary structural elements are sound but may have minor section loss, deterioration, spalling, or scour	5	3	C	C	
Advanced section loss, deterioration, spalling, scour	4				
Loss of sections etc. has affected primary structural components. Local failures are possible. Fatigue cracks in steel or shear cracks in concrete may be present	3	4	D	D	III, IV
Advanced deterioration of primary structural elements. Fatigue cracks in concrete may be present or scour may have removed structural support. Unless closely monitored it may be necessary to close the bridge until corrective action is taken	2				
Major deterioration or loss of section in critical structural components or obvious vertical or horizontal movement affecting structural stability. Bridge is closed to traffic, but corrective action may put back in light service	1	5	E	E	
Out of service, beyond corrective action	0				

Condition rating class, which inherits from *BaseConditionRating*, is a set of predefined rating models and is used as the input of the element model in the present study. Similar to the parent object, future developed condition rating modules must contain a “ratings” attribute that contains the condition ratings of the assets. Although several condition rating models for bridges (based on NBI, PONTIS, and systems used in other countries) are currently provided in GIAMS, similar models could be later incorporated into the platform. It should be noted that due to technical concerns regarding the implementation of these condition ratings, it was decided to use a numeric system starting from 0 in increasing order from best to worst condition. For example, 0 instead of 9 and 9 instead of 0 should be used in the NBI rating. Similarly, the Korean rating would turn to the 0-4 rating system.

3.1.2.2 Deterioration

Deterioration is a stochastic process affected by several factors including but not limited to weather conditions, age, and construction quality. In one of the earliest studies of asset management systems, Golabi *et al.* [41] used the Markov chain to model the transition between

one state to another. This approach has been adopted in various asset management systems [2,9,10]. In this context, Markov chain models have a limited number of states (i.e., condition ratings). It is assumed that there is a fixed transition probability P_{ij} , from one state, i , to the next, j . Besides, a state can only change one level at a time, i.e., the state can change from 2 to 3, but not from 2 to 4. Fig. 3 depicts an example of a Markov chain matrix with 5 states that could be used in asset management.

$$\begin{bmatrix} P_{11} & 1 - P_{11} & 0 & 0 & 0 \\ 0 & P_{22} & 1 - P_{22} & 0 & 0 \\ 0 & 0 & P_{33} & 1 - P_{33} & 0 \\ 0 & 0 & 0 & P_{44} & 1 - P_{44} \\ 0 & 0 & 0 & 0 & P_{55} \end{bmatrix}$$

Fig. 3. A Markov chain matrix with 5 states

Although this phenomenon is stochastic, several studies [2,38,42] have proposed empirical methods based on available data to predict the asset's condition in time that could be used in this regard. Regardless of the formula, the deterioration object inherits from the *BaseDeterioration* object. Similarly, deterioration models developed in the future should follow this pattern. The *predict_method* of the deterioration model provides a probabilistic condition of an element after a time step given a previous condition. Its results will be used in the simulator model of the LCA module.

3.1.2.3 Agency costs

Implementing MRR incurs direct and inevitable costs. Given the type of actions, be it maintenance, rehabilitation, or reconstruction, the responsible agencies are required to provide the financial resources for carrying out MRR plans. Therefore, it is essential to formulate these costs effectively. Linear models, Wiener process [43], Geometric Brownian motion [43], Cobb-Douglas models [2,44], and an aggregation of several expert judgments and historical data [45] are some examples of methods that could be used to model agency costs. Any agency cost model to be developed in GIAMS in the future must inherit from the *BaseAgencyCost*. The *predict_series* of the agency cost objects returns the values required for life cycle analysis in the simulator of the life cycle analysis.

3.1.2.4 Utilities

Each MRR action has a different utility for the agencies. As the ultimate goal of agencies is to maximize utility and minimize costs at the same time, utility functions need to be properly estimated. This estimation could be performed via surveys and calibration of a function to the survey results via regression [2]. Li and Sinha [46] provided several utility functions for the case of bridges that have been used in the literature [33]. The utility of elements could be congregated and form the utility of assets with equal or different weights [33]. For future developments in other problems, the utility object of each element should be inherited from the *BaseUtility*. Its *get* method returns the utility of any conducted action based on a *utility_function* method for the element.

3.1.3 Hazard model

The response of assets in the face of probable hazards, as well as the incurred loss costs and post-hazard recovery strategies, should be modeled meticulously to represent various aspects

of hazards both before and after the occurrence. The hazard model consists of four main sub-models in this regard, shown in Fig. 4.

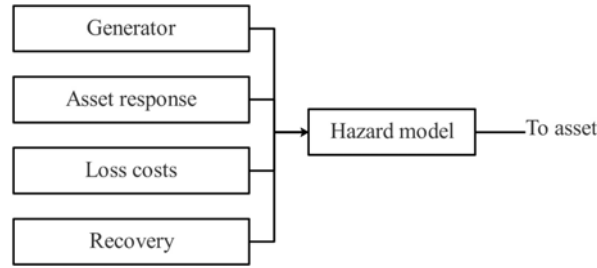


Fig. 4. Sub-models of the hazard model

3.1.3.1 Generator models

Hazards such as earthquakes are unpredictable events that could happen at different times with different magnitudes. Despite their unpredictable nature, the distribution and occurrence rate of a given hazard can be estimated based on historical data. Such data enables random sampling of hazards based on the distributions of occurrences and magnitudes [47]. The Poisson point process (PPP) is a method that can be used to model random phenomena [43]. The interval between occurrences is modeled by the Poisson process, Eq. (1), which gives the probability of n occurrences with a rate of λ during period T :

$$p(n) = \frac{(\lambda T)^n e^{-\lambda T}}{n!} \quad (1)$$

After generating the occurrence times of the hazards, the magnitude of the hazard is sampled based on its magnitude distribution. The distribution and occurrence rate of the hazards are the input parameters of the currently implemented generator class, PPP. Other hazard generator models that must inherit from the *BaseGenerator* object could have different input parameters. In any case, the hazard generator model must consist of a *generate_one_lifecycle* method that will be called during the simulation. This method returns the time and magnitude of a generated sample of hazards on a horizon.

3.1.3.2 Response models

Different assets behave in different ways towards hazards. Response models are required to be designed to reflect these variations. HAZUS – MH2.1 [13] is a technical manual with a focus on hazard responses of assets. The general approach of this manual is to use a type of probabilistic model called fragility curves, shown in Eq. (2). The purpose of using these curves is to find the probability of exceedance from a certain damage state given a hazard magnitude and the asset characteristics. In the case of bridges, these models are a function of bridge classification, spectral accelerations at 0.3 sec ($S_{a0.3}$), and 1.0 sec ($S_{a1.0}$), and peak ground acceleration.

$$P_{S \geq s_i | IM} = \Phi \left\{ \frac{1}{\beta_{s_i}} \ln \left(\frac{IM}{m_{s_i}} \right) \right\} \quad (2)$$

where $P_{S \geq s_i | IM}$ is the probability of exceedance of the state of assets from s_i , Φ is the standard normal cumulative distribution function, IM is the ground motion intensity measure, m_{s_i} is the median value of ground motion intensity with damage state i , and $\beta_i(s_i)$ is the dispersion factor of damage state i . A fragility-based response model and a pre-tuned model for different bridge classifications are currently provided in the proposed platform. Future response models

must inherit from the *BaseResponse*, in which its *get* method returns the response of the asset given a specific hazard. The results of the response model are twofold in GIAMS and include the damage state and the mapped condition based on the damage state. These output parameters will be used in the lifecycle analysis model.

3.1.3.3 Loss models

The occurrence of a hazard not only disrupts the functionality of an asset or in some cases leads to its collapse, but also can cause fatalities and casualties. These casualties and fatalities differ for different assets. In other words, the fatalities of buildings are different from those of bridges. These values are also provided by the HAZUS – MH2.1 [13] in a probabilistic manner. Although the manual does not provide bridge casualty data the option to assess losses due to a hazard is provided in GIAMS. Any future designed and developed loss object inheriting from the *BaseLoss* must have a *predict_series* method that returns the loss costs of an asset given a certain damage state in the management horizon. Similar to other models, the output of this model will be used in the life cycle analysis model.

3.1.3.4 Recovery models

After the occurrence of a disastrous hazard, affected assets must be recovered to maintain the asset's functionality for the community. Agencies and decision-makers should have pre-determined plans for various after-hazard circumstances to evaluate the life cycle costs and conduct optimization. In other words, they should determine the course of actions to be taken after hazards if the asset would be in intact, slight, moderate, extensive, or collapsed damage states. This option has also been provided in this framework to make simulation and analysis closer to reality. Inheriting from the *BaseRecovery*, any recovery object must contain a *get* method that returns an after hazard recovery plan given a certain condition rating.

3.1.4 MRR models

Typically, four different types of actions are considered in asset management: maintenance, rehabilitation, reconstruction, and do nothing [2,9,10]. Two different types of encoding, binary and numerical, have been used in optimization and life cycle analysis, shown in Table 2.

Table 2. MRR actions encoding

Action	Abbreviation	Numerical encoding	Binary encoding
Do nothing	DONOT	0	00
Maintenance	MAINT	1	01
Rehabilitation	REHAB	2	10
Reconstruction	RECON	3	11

To implement a vectorized MRR, this framework uses a 3-dimensional array in its computation core. The first dimension corresponds to the asset, the second dimension corresponds to elements, and the third dimension corresponds to the year in which the MRR action should take place. Illustrating the vectorized form of the MRR in this framework, **Fig. 5** shows an example biennial 20-year MRR plan for a network with 2 bridges with 2 elements.

	0-2	2-4	4-6	6-8	8-10	10-12	12-14	14-16	16-18	18-20
Bridge 1 – Element 1	[[0	2	0	0	0	1	0	0	0	0]
Bridge 1 – Element 2	[0	1	0	0	0	3	0	0	0	0]]
Bridge 2 – Element 1	[[0	0	0	3	0	0	0	0	0	1]
Bridge 2 – Element 2	[0	0	0	3	0	0	0	0	0	0]]

Fig. 5. An example of vectorized MRR for 2 bridges with 2 elements

Some asset management studies also focus on retrofit planning [47] of structure. These areas of study usually focus on binary choices (Do Nothing and Take Action). To meet this need, GIAMS also contains a two action MRR module for binary choices that could be used for similar purposes. The developed MRR models for this platform are the least flexible module as most of the analysis modules (e.g., optimization modules) rely on the assumption of two or four different actions. However, future MRR models could be implemented in GIAMS with various substantial changes and revisions in the modules of the platform.

3.1.4.1 MRR effectiveness

Another noteworthy issue regarding the MRR plans is the effectiveness of actions. Effectiveness refers to the degree that an MRR action would ameliorate the condition of the asset. For instance, maintenance of an element with major deterioration or section loss would not improve its condition greatly. On the other hand, reconstruction of any element would change its condition to an as-built state. Inspired by their use in PONTIS [9], Markov chain models have been adopted in GIAMS to inform the model regarding the extent of improvement. An example of such a model for rehabilitation is shown in Table 3.

Table 3. An example of the transition probabilities as a result of MRR actions and deterioration for an asset with 5 condition ratings

State	Action	1	2	3	4	5
1	DONOT	0.98	0.02	0	0	0
	MAINT	1.00	0	0	0	0
	REHAB	1.00	0	0	0	0
	RECON	1.00	0	0	0	0
2	DONOT	0	0.95	0.03	0	0
	MAINT	0.96	0.04	0	0	0
	REHAB	1.00	0	0	0	0
	RECON	1.00	0	0	0	0
3	DONOT	0	0	0.88	0.12	0
	MAINT	0.63	0.27	0.10	0	0
	REHAB	0.90	0.10	0	0	0
	RECON	1.00	0	0	0	0
4	DONOT	0	0	0	0.87	0.13
	MAINT	0.50	0.15	0.15	0.20	0
	REHAB	0.80	0.15	0.05	0	0
	RECON	1.00	0	0	0	0
5	DONOT	0	0	0	0	1.00
	MAINT	0	0	0	0.02	0.98
	REHAB	0.03	0.10	0.17	0.30	0.40
	RECON	1.00	0	0	0	0

Any further developed effectiveness model must inherit from the *BaseEffectiveness* module and must contain a *get* method. This method receives the condition of the asset and the proposed action as input and returns the condition of the asset as output. This output of the effectiveness models is then used in the life cycle analysis.

3.1.5 User cost models

MRR actions usually result in a heavy burden placed on the stakeholding community. This cost, which is usually called user costs, could be many times greater than the agency costs [48].

For example, if an agency wants to rehabilitate the deck of a bridge, it will typically need to close one lane or the whole bridge. Drivers will then have to reduce their speed because of a reduced number of lanes, or they must choose a detour to get to their destinations. Detours are longer than the route that includes the bridge and have less maximum travel speed. This means thousands of additional hours and gallons of gas consumed because of one bridge rehabilitation. This concept holds for other types of assets. User costs are another parameter that should be minimized in the optimization process. Several methods have been described to estimate the user costs [2,49], details of which are beyond the scope of this article. A user costs model based on the Texas Department of transportation is provided in the framework of this study. Future user cost models will have to inherit from the *BaseUserCost* that makes having a *predict_series* method in it necessary. The output of this model will also be used in the life cycle analysis.

3.2 Network

Asset managers and agencies are in charge of maintaining several assets in a network. Developing a network module follows developing asset modules and sub-models. The network module in this framework plays the role of forming a network by aggregating all assets, yielding the network objectives in the optimization process, and imposing the network constraints. The aggregation part has the responsibility of loading the assets from a data file and converting them in a way so that the framework can understand and conduct analysis upon it. An example of the network module has been provided in the repository for further modifications. Modifying the network module is one of the earliest steps in using this platform for other problems. Inheriting from the *BaseNetwork*, any network object must contain a *load_network* method that loads all the assets and assign them to the network as the *asset* attributes. The network object itself will be assigned to the LCA module for further analysis.

Intending to optimize the MRR plans, the network module also yields objective values, which can be generated in a minimization or maximization form. The objective function of this analysis could be:

$$\begin{aligned} & \text{minimize } f_1(x) \\ & \text{maximize } f_2(x) \\ & \dots \\ & \text{minimize } f_m(x) \end{aligned} \tag{3}$$

In the case of having more than one objective function, which is called multi-objective optimization, a widely used approach is to transform all objective functions into a single objective using a multiattribute utility function method [33]. Maximizing the final utility function, Eq. (4), would then be the new objective.

$$\text{maximize } U(f_1(x), f_2(x), \dots, f_m(x)) \tag{4}$$

The utility of a network could be the sum of the utility of its assets. However, Bai et al. [33] proposed the concept of Holism and raised the additivity problem of the non-linear utility functions. Instead, they proposed applying the utility function to the average of the network condition rating. Both of these methods for calculating the utility of a network are readily provided in GIAMS. These objective values (utilities) are optimized under several constraints in asset management. These constraints are usually in two forms [33]: budget constraints and

performance constraints. Budget constraints represent the maximum amount of money that agencies are capable of investing in the MRR of the assets and can be formulated as shown in:

$$\sum_{i=1}^n x_i c_i \leq B \quad (5)$$

where x_i is a binary value indicating the presence of project i in a portfolio, c_i is the cost of project i , and B is the budget. Similar constraints could be imposed on the performance level of a network. Performance constraints are vital to decision-makers who want to keep their network at an acceptable level of performance. In the case of bridges, for example, the upper bound of performance would be the maximum number of assets in a poor condition and the lower band of performance would be the average of the assets' condition rating in the network. The performance constraint can be described as:

$$f^L \leq f(x) \leq f^U \quad (6)$$

where f^L and f^U are the lower and upper bands of constraints, and $f(x)$ is an objective function.

3.3 Life cycle analysis

The life cycle in this framework refers to the investment and analysis horizon of an asset. This decision horizon could vary for different assets of a network [2] or could be a fixed value [10]. The range and period of the asset management horizon could also vary drastically. It could be as low as 20 years [10] up to several decades [48]. Decisions are taken periodically based on new information gathered by professional inspectors. In the case of bridges, for example, the condition and other data on bridges are gathered every 24 months [40]. Example costs that could be placed upon an agency and community given the condition of a bridge and MRR plans are depicted as a cash flow diagram in Fig. 6.

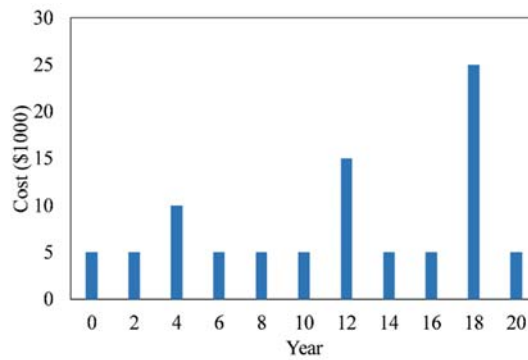


Fig. 6. Example of cash flow and MRR for an asset

To compare these costs, they can be discounted by Eq. (7) to the present time with a discount factor:

$$NVP_{Costs} = \sum_i^n \sum_{j=1}^{n_p} \frac{C_{ij}}{(1+r)^n} \quad (7)$$

where C_{ij} is the cost of project j in the i^{th} year, r is the discount factor, n_p is the number of projects, and n is the number of steps in the investment horizon.

The life cycle analysis could be conducted using simple calculations if all models are deterministic [9], or using a Monte Carlo simulation to get the expected results if uncertainty lies within the models [45]. In such cases, the life cycle analysis module requires a simulator to create samples of all possible incidents in a life cycle. Fig. 7 shows an example of all incidents that have happened to an asset in a 20-years horizon.

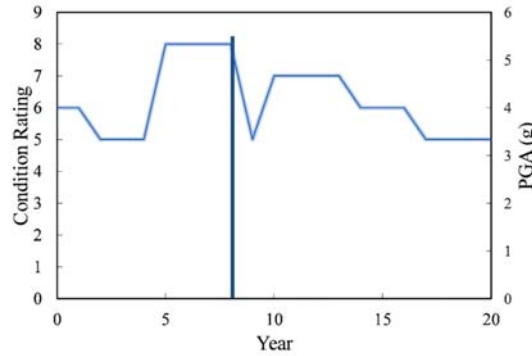


Fig. 7. An example of life cycle incidents for an asset

In this example scenario, the initial condition rating (CR) of one element of an asset is 6 on the NBI scale. Later, deterioration alters its CR resulting in a CR of 5. Rehabilitation in the 4th year improves its condition, but an earthquake with a PGA of approximately 5.5 during the 8th year results in a CR of 5. Then recovery actions restore its condition to a CR of 7. The element's condition rating continues to change, affected by deterioration until the end of its management horizon. These events, including sampling hazards, response to hazards, and deterioration, are probabilistic phenomena. A Monte Carlo simulation should be used to calculate the expected life cycle results of one MRR strategy under various uncertainties.

The simulator object serves as the computation core of the proposed platform. It generates samples for each asset in the life cycle and returns the analysis results based on the MRR of the asset. Although most common incidents and phenomena governing the assets are currently incorporated into GIAMS, other factors could be added through further development. The simulator object inherits from the *BaseSimulator* and has a *get_one_instance* method. The outputs of this method, i.e. the user costs, agency costs, and utilities, will be used in the LCA module. The LCA class receives a network and a simulator as input objects and performs a life cycle analysis using the simulator on the network. The results of the LCA usually fall into three main categories: 1) user costs, 2) agency costs, 3) utilities of the actions. However, the option to get other user-defined types of results from the simulator is also provided in the GIAMS repository. These results, be it deterministic or probabilistic, net present value, or stepwise, are passed on to the optimization module to determine optimal plans.

3.4 Optimization

In this section, the concept and theory of two optimization algorithms that could be suitable for the project-level and network-level asset management are briefly discussed.

3.4.1 Project-level

The purpose of project-level asset management is to inform decision making about prospective MRR actions concerning the elements of an asset in a life cycle or a management horizon in

advance. Hazards, deterioration patterns, as well as different costs, are probabilistic phenomena that could occur in the life cycle. Therefore, an optimized MRR strategy should be selected given various uncertainties in the management horizon. This binary decision-making process could be formulated in a variety of forms. Utility maximization with respect to budgetary limitations and costs minimization with respect to performance limitations are examples of such forms. From another perspective, the process could be approached as a single objective optimization (SOO) or a multi-objective optimization (MOO) problem. Given the complexity of MOO problems, a multi-attribute utility function method could be used as an alternative [33]. This method consists of converting various objectives into dimensionless utilities and then combining them into a single utility. As a result, the original MOO problem will turn into an SOO. All in all, an example of a general form of utility maximization could be:

$$\text{maximize } E \left[\sum_{i=0}^n \sum_{p \in P_i} U_{pi} X_{pi} \right] \quad (8)$$

With respect to:

$$\begin{aligned} \sum_{p \in P_i} AC_{pi} X_{pi} &\leq B_i \\ \sum_{p \in P_i} CR_{pi} X_{pi} &\leq CR_{min} \\ \sum_{p \in P_i} X_{pi} &= 1, \quad \text{for each year } i \\ X_{pi} &\in \{0, 1\} \end{aligned}$$

where U_{pi} , AC_{pi} , CR_{pi} are utility, agency cost, and condition rating after the project p is conducted on the i^{th} year, X_{pi} is a binary value if project p is selected on the i^{th} year, CR_{min} is the minimum of the asset (or element) condition, and B_i is the maximum agency budget on the i^{th} year. Given the complexity of this optimization formula, heuristic algorithms are recommended [38] to solve the optimization problem.

The genetic algorithm (GA) [50] is a heuristic search method that is used to find the maximum or minimum of an objective function. Although GA is computationally expensive and perhaps impractical for big networks, its merits make it desirable for project-level life cycle optimization of assets [38]. First, it can provide near-optimal MRR strategies with the highest expected utility, though it does not guarantee to identify the optimal solution. Second, GA can be easily understood, implemented, and used by practitioners and researchers. Finally, GA is flexible for use in optimization problems with a variety of forms. Consequently, it could satisfy three out of four conditions, namely accuracy, computation time, robustness, and simplicity, of an appropriate optimization method proposed by Patidar *et al.* [51]. Presumably, due to these reasons, GA has been the most popular optimization algorithm in the asset management literature as of the late 2020s [8].

Inspired by natural evolution, this algorithm seeks a solution in several evolving generations of individuals until the search algorithm meets certain criteria. The common steps of a GA optimization are provided in Fig. 8.

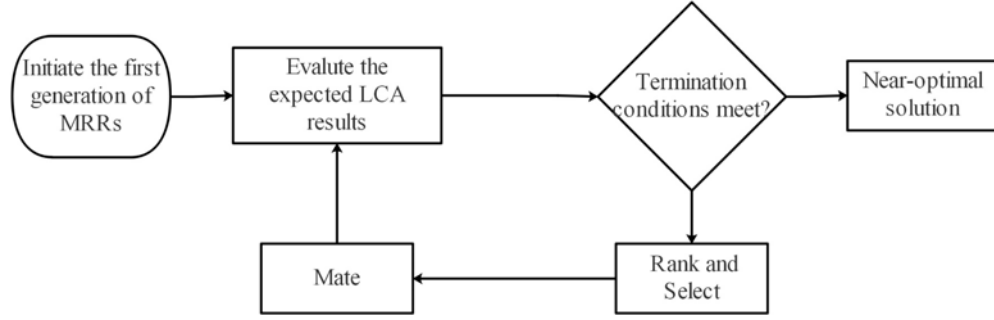


Fig. 8. Genetic algorithm flowchart

Individuals contain a chromosome which represents a point in the search space. In the context of asset management, the binary representation of MRR plans is its chromosome. To illustrate, this **framework**.

	0-2	2-4	4-6	6-8	8-10	10-12	12-14	14-16	16-18	18-20
Bridge 1 – Element 1	[[0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0]]									
Bridge 1 – Element 2	[0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0]]									
Bridge 2 – Element 1	[[0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1]]									
Bridge 2 – Element 2	[0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0]]									

Fig. 9 shows the binary representation of the MRR plan in this framework.

	0-2	2-4	4-6	6-8	8-10	10-12	12-14	14-16	16-18	18-20
Bridge 1 – Element 1	[[[0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0]]									
Bridge 1 – Element 2	[0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0]]									
Bridge 2 – Element 1	[[0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1]]									
Bridge 2 – Element 2	[0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0]]									

Fig. 9. The binary representation of an MRR plan

At first, these chromosomes are initiated to breed the first generation. The performance of individuals indicates their chance of passing their genes to the next generation. The objective function of the optimization problem (e.g., Eq. (10)) is used to represent this performance. The selection and mating steps follow the performance evaluation step. Intuitively and similar to natural selection, stronger genes are passed on to the next generation and weaker ones are eliminated. Among several selection methods such as random selection and roulette wheel, the rank-based selection was chosen because of its simple adaptability to both minimization and maximization problems. In rank-based selection, individuals are sorted based on their objective function value and the optimization type. Then they are selected with the probability of:

$$p(j) = \frac{R_j}{\sum_{i=1}^n i} \quad (9)$$

where $p(j)$ is the probability of selection of individual j , R_j is the rank of individual j , and n is the number of individuals. After selecting two parents, they will mate and create two new

offspring. The mating consists of two phases, namely crossover and mutation, shown in Fig. 10.

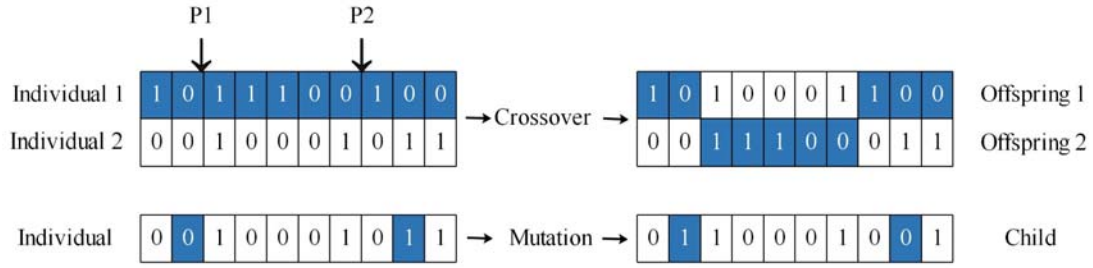


Fig. 10. Crossover and bit-flip mutation

Some of the top solutions are directly transferred to the next generation to avoid losing the optimal solution during the optimization. They are called the elites of each generation. Some of the hyperparameters of the genetic algorithm are crossover strategies, crossover probability, mutation strategies, mutation probability, number of elites, the population size in each generation, number of generations, number of elites, and termination conditions. For details on these hyperparameters, please refer to [52–55].

3.4.2 Network-level

The ultimate purpose of network-level asset management is allocating the available and limited budget to certain decisions and tasks to improve the overall conditions of an asset or a network, reduce the user costs, and sustain the community. The budget allocation for different projects of a portfolio is also a binary choice. This problem is defined as a multichoice multidimensional knapsack problem (MCMDKP) [51]. Multichoice in this formulation means that exactly one choice, including do-nothing, must be selected for each project. Multidimensional refers to different budget or performance constraints. Although this problem could be formulated in many ways, a general example is provided:

$$\text{maximize } \sum_{b=1}^n \sum_{p \in P_b} U_{pb} X_{pb} \quad (10)$$

With respect to:

$$\begin{aligned} & \sum_{b=1}^n \sum_{p \in P_b} AC_{pb} X_{pb} \leq B \\ & \frac{1}{n} \left(\sum_{b=1}^n \sum_{p \in P_b} CR_{pb} X_{pb} \leq CR_{min} \right) \\ & \sum_{p \in P_b} X_{pb} = 1, \quad \text{for each } b \\ & X_{pb} \in \{0, 1\} \end{aligned}$$

where U_{pb} , AC_{pb} , CR_{pb} are utility, agency cost, and condition rating after the project p is selected for bridge b , respectively, X_{pb} is a binary value if project p is selected for bridge b , CR_{min} is the minimum of the average of network condition, and B is the maximum agency

budget. Exact methods could yield a solution to an MCMDKP problem. Nevertheless, the computation time for these polynomial time-hard optimization problems grows exponentially with the number of decision parameters. Acknowledging the undesirability of high computational costs [38,51,56,57], a heuristics method could provide near-optimal solutions more quickly, though at the expense of losing the exact solution.

Incremental utility cost (IUC) ratio is a heuristic algorithm that could be used to approximate the result of the knapsack optimization, Eq. (10). Another version of IUC, incremental benefit-cost ratio (IBC) has been used in previous asset management systems [9,51] for the same purpose. The IUC optimization algorithm is appealing for several reasons. First, it is a greedy heuristic algorithm that yields a near-optimal solution benefiting the project with the highest IUC [51]. Second, the computation time of IUC optimization is considerably less than that of deterministic methods or some other heuristic methods (e.g., Lagrangian relaxation). Third, it is simple and understandable by prospective users. Finally, it is easily adaptable to variations in the optimization problem. Further discussion on methods for asset management can be found in [51].

Within the IUC heuristic optimization algorithm, the potential projects are sorted based on their utility cost ratio, Eq. (11) in descending order:

$$UC(p_j) = \frac{\Delta U_{p_j}}{C_{p_j}} \quad (11)$$

where ΔU_{p_j} is the difference in the utility if the project p_j is used at the cost of C_{p_j} . Then the available budget is allocated to the remaining projects with the highest IUC. This allocation is continued until the agency's budget is exhausted (Algorithm 1).

Algorithm 1. IUC heuristic

```

1:  $P = \{\}$ 
2:  $Q = \{\text{All projects sorted based on IUC in decreasing order}\}$ 
3: for each  $p_j$  in  $Q$ :
4:   if  $C(P) + C(p_j) > B$ :
5:     break
6:   else:
7:      $P = P \cup \{p_j\}$ 
8: return  $P$ 

```

Note: P is the selected portfolio of projects, Q is all projects sorted based on IUC, and C is the cost of the projects.

4 Illustrative examples

Since the 1980s, several bridge management systems, updates, and research methods have been proposed for the Indiana bridge network [2,33]. The network in Indiana consists of more than 4,600 state bridges [33], with their related information such as structural types, average daily traffic, and condition rating stored in the national bridge inventory of the US. For illustration, three main components of bridges, namely deck, superstructure, and substructure, were used for life cycle optimization and network budget allocation. In addition, agency costs, user costs, utilities, and deterioration models for each bridge were adopted from IBMS [2] and the Texas Department of Transportation [49]. Seismic characteristics such as site class, HAZAS classification, response models, and loss models were derived from the HAZUS manual [13]. Earthquake history data for the surrounding region were also collected from the USGS

earthquake database [14]. Other models such as effectiveness models and recovery models were inspired by previous BMSs (e.g., BRIDGIT [10]) or rationally assumed (i.e., the condition rating of the asset after recovery actions will be similar to that of NBI rating 8). These assumptions and the models can be readily found in the repository of GIAMS [12]. Given that a number of previously published articles in the asset management area by the late 2020s were at the project-level [8], GIAMS was designed so that it could search for near-optimal life cycle MRR strategies by GA. However, the majority of previous studies have focused on network-level optimization [8]. Therefore, IUC has been developed in GIAMS to meet this need across different asset management areas.

Using the bridge network of Indiana, two illustrative examples (one project-level and one network-level) are provided to demonstrate the applicability of the current components of GIAMS. Being derived by genetic algorithm and Monte Carlo simulation, the first example (“Example1” in the GIAMS repository) is dedicated to the optimization of the utility over a fraction of all costs of one asset. The limitations of this optimization are the agency’s annual budget in the management horizon and a maximum of the net present value of all costs. In addition, one element cannot experience two consecutive MRR actions, be replaced more than twice, be rehabilitated more than 4 times, or be maintained more than 6 times in the management horizon. The second example is dedicated to network-level optimization of Indiana’s bridge upkeep using IUC. The pool of candidate projects for network-level optimization could be populated according to expert judgments and decision trees (e.g., DTREE [2]). However, a naïve approach has been currently adopted in GIAMS. In this approach, the projects with the highest utility/cost (U/C) ratios out of all possible combinations of MRR actions for all bridge elements at the decision-making time are considered potential projects to be carried out for that bridge.

5 Results

The results of the two illustrative examples are briefly summarized in this section.

5.1 Example 1: Life cycle optimization of one asset

The first example focuses on finding an optimized MRR plan for the bridge with structure number 10 in the NBI inventory. The purpose of this optimization is to maximize the utility of the MRR actions over a fraction of all costs in a 20-year management horizon under budget limitations. The optimization is formulated as follows:

$$\text{maximize } E \left[\sum_{i=0}^n \sum_{p \in P_i} (U_{pi} X_{pi} / C_{pi}^{0.2}) \cdot e^{-\lambda i} \right] \quad (12)$$

where U_{pi} and C_{pi} are utility and all costs (i.e., agency cost + user costs) after the project p is conducted on the i^{th} year, X_{pi} is a binary value if project p is selected on the i^{th} year, and λ is the discount rate. The discount rate was assumed to be 3% in this study. Table 4 provides a summary of the hyperparameters used in this example.

Table 4. Summary of GA hyperparameters

Hyperparameter	Value	Hyperparameter	Value
Optimization type	Maximize	Crossover probability	0.75
Population size	100	Crossover method	Two-point
Number of generations	200	Mutation probability	0.03
Number of elites	5	Mutation method	Bit-flip

Random initialization and preference initialization could be used for this first-generation initiation. However, preference initialization has a higher convergence speed [58] and is used in this example. In this approach, the probability of each bit of the MRR binary array to be 1 could be 0.1, 0.2, 0.3, 0.4, or 0.5. This analysis was conducted by an Intel® CORE™ i7-8700T, 2.40 GHz, 12 computational cores, and 8GB RAM in parallel in approximately 492 minutes. The results of the optimization, (i.e., the utility versus generation number) are depicted in Fig. 11.

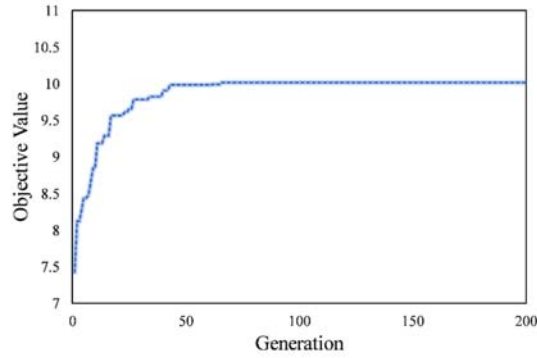


Fig. 11. The utility of the asset in generations

The results of the optimization suggest a strategy regarding the MRR of the studied bridge, which is depicted in **Fig. 12**. The description of the numerical encoding can be found in Table 2.

		0-2	2-4	4-6	6-8	8-10	10-12	12-14	14-16	16-18	18-20	
Deck	[[[3	0	1	0	0	0	1	0	0	1]
Superstructure	[2	0	1	0	0	0	0	0	0	1]]
Substructure	[[2	0	1	0	0	0	0	0	0	1]]]

Fig. 12. The MRR strategy in the management horizon of the example bridge

Given the underlying uncertainties in genetic algorithm optimization, it is sometimes necessary to validate its results. To this end, different options are provided in GIAMS:

- 1- **Brute force algorithm:** Given the complexity of the designed problem and access to sufficient computational resources, GIAMS can enumerate and analyze all possible combinations of MRR plans. To reduce the computation time, the developed brute force algorithm takes advantage of parallel computing [59] being able to analyze in parallel on any arbitrary number of computational cores.
- 2- **Benchmarking functions:** The performance of a developed genetic algorithm can be evaluated on some benchmarking functions [60]. Although several benchmarking functions (e.g., De Jong's function, Rosenbrock's valley function, and Schwefel's function) are currently implemented in GIAMS for validation, the design of GIAMS provides the opportunity for adopting any other benchmarking functions to validate the results of the optimization.
- 3- **Optimizing several rounds:** Another approach to validate the results of the genetic algorithm is to conduct the optimization several times to ensure not converging to a

- sub-optimal point [61]. This option is currently provided in GIAMS to automatically perform the optimization for a user-defines number of rounds.
- 4- Other heuristic algorithms: Although genetic algorithm is the proposed optimization algorithm for project-level asset management and has been the most used optimization algorithm in the asset management literature [8], other heuristic algorithms (i.e., particle swarm optimization and hill climbing) are designed and developed in GIAMS for further research. Nevertheless, the comparison of the performance of these optimization algorithms with the genetic algorithm is not in the scope of this study.

5.2 Example 2: Network-level project selection

IUC heuristics analysis was conducted by the same computational platform used in the previous example in 8.4 minutes. The output of this analysis is a ranked portfolio of possible projects for bridges in the network at the time of analysis. Table 5 presents the top 20 bridges with the highest U/C ratio and their chosen MRR strategies.

Table 5. Top 20 bridges with highest U/C ratio and corresponding MRR actions

Rank	Bridge ID	Deck	Superstructure	Substructure
1	18841	DONOT	DONOT	RECON
2	80132	DONOT	MAINT	MAINT
3	80362	DONOT	MAINT	MAINT
4	80126	DONOT	MAINT	MAINT
5	43020	DONOT	MAINT	MAINT
6	75240	DONOT	MAINT	MAINT
7	37410	DONOT	MAINT	MAINT
8	70520	DONOT	MAINT	MAINT
9	6002	DONOT	DONOT	MAINT
10	31130	MAINT	MAINT	MAINT
11	75200	DONOT	MAINT	MAINT
12	80294	DONOT	MAINT	DONOT
13	33165	DONOT	MAINT	MAINT
14	26570	DONOT	DONOT	MAINT
15	70250	DONOT	MAINT	MAINT
16	76270	DONOT	MAINT	MAINT
17	37300	DONOT	MAINT	MAINT
18	25510	MAINT	MAINT	MAINT
19	80348	DONOT	DONOT	MAINT
20	6003	DONOT	DONOT	MAINT

6 Discussions and summary

This paper introduced the first open-source Python-based platform for asset management, GIAMS, which could be used as a stepping-stone in a variety of asset management research areas. First, the major building blocks of the asset module consisting of asset type, elements, hazard models, MRR, and user costs were discussed. Then the network module as a holder of the assets, limitations, and objectives was introduced followed by a life cycle analyzer block that simulates different possible events affecting an asset. Finally, two heuristic optimization algorithms, namely a genetic algorithm and an incremental utility cost algorithm, were

described. The utilization of GIAMS was illustrated through two examples based on real data for bridges in Indiana, US, and relevant practical models. Key concepts for open collaboration in software development are provided in Appendix A, while Appendix B briefly discusses three examples for the development of GIAMS. In these examples, the simplicity of using GIAMS, testing a new idea, and contributing updates to the source code are shown. Moreover, extending GIAMS to a new area of asset management (i.e., prioritizing seismic retrofit of building structures [47]) and employing the concept of structural health monitoring in asset management [62–65] are briefly discussed. All previously developed AMSs to date have been inextensible, costly, and written in less flexible programming languages. GIAMS is different from and superior to these asset management systems in some respects:

- 1- GIAMS is open-source. This crucial characteristic of GIAMS can greatly accelerate progress in the field of asset management by enabling future researchers to build upon each other's work. Being open-source also facilitates the use of the latest updates for researchers and practitioners around the world.
- 2- GIAMS is written in Python. Python language makes adopting advanced models related to asset management possible (e.g., using a trained deterioration model by machine learning algorithms). The popularity of Python among researchers also paves the way for researchers to collaborate more efficiently.
- 3- GIAMS is freely accessible. Researchers can clone the latest updates of the proposed GitHub repository. In addition, underprivileged communities and agencies that are incapable of affording licensed and expensive asset management systems could take advantage of this system.

GIAMS is currently in its embryonic stage. The main limitation of this platform is its relatively nascent models in various asset management areas for representing real-world phenomena as close as possible to reality. Besides, the implementation of GIAMS is described here through a bridge management example. Future research is required to implement models describing other phenomena governing different assets such as pavement and sewage systems that could be developed and used in this platform. In other words, the proposed platform can also be further developed and extended to be applied to other types of assets that need to be managed by agencies, municipalities, and other decision-makers. Another possible area of future research would be to investigate and develop other optimization algorithms (e.g., Lagrangian relaxation) to be used in GIAMS. There is abundant room for further research in the implementation of advanced models (e.g., machine learning models for predicting deterioration) and developing them in GIAMS.

The extensibility and open-source nature of GIAMS promote the ability of researchers to directly draw upon previous work when developing models for analyzing and optimizing the project-level life cycle of assets and network-level budget allocation for MRR actions. Be it implementation of advanced models or new optimization algorithms, more easily and rapidly conducted research in the asset management area is greatly enabled by the extensibility of the proposed platform. It is expected that through the open collaboration of researchers over time, this software could be turned into an advanced and mature platform in the asset management research area.

References

- [1] F. Biondini, D.M. Frangopol, Life-cycle performance of deteriorating structural systems under uncertainty: Review, *Journal of Structural Engineering*. 142 (2016) pp.F4016001. [https://doi.org/10.1061/\(ASCE\)ST.1943-541X.0001544](https://doi.org/10.1061/(ASCE)ST.1943-541X.0001544).
- [2] K.C. Sinha, S.A. Labi, B.G. McCullough, A. Bhargava, B. Qiang, Updating and enhancing

- the Indiana bridge management system (IBMS), Publication FHWA/IN/JTRP-2008/30. Joint Transportation Research Program, Indiana Department of Transportation and Purdue University. (2009). <https://doi.org/10.5703/1288284314306> (accessed January 6, 2021).
- [3] Z. Lounis, T.P. McAllister, Risk-based decision making for sustainable and resilient infrastructure systems, *Journal of Structural Engineering*. 142 (2016) pp.F4016005. [https://doi.org/10.1061/\(ASCE\)ST.1943-541X.0001545](https://doi.org/10.1061/(ASCE)ST.1943-541X.0001545).
 - [4] O. Kaganova, J. Telgarsky, Management of capital assets by local governments: An assessment and benchmarking survey, *International Journal of Strategic Property Management*. 22 (2018) pp.143–156. <https://doi.org/10.3846/ijspm.2018.445>.
 - [5] C. Zhong, S.T. Ng, M. Skitmore, An interdependent infrastructure asset management framework for high-density cities, in: *Proceedings of the Institution of Civil Engineers - Municipal Engineer*, (2019) pp. 1–11. <https://doi.org/10.1680/jmuen.18.00053>.
 - [6] Y. Yang, S.T. Ng, F.J. Xu, M. Skitmore, S. Zhou, Towards resilient civil infrastructure asset management: An information elicitation and analytical framework, *Sustainability*. 11 (2019) pp.4439. <https://doi.org/10.3390/su11164439>.
 - [7] N.A. Khalifa, A. Zulkiple, A. Gheit, A. Alfughi, I. Ali, S.A. Ghammeid, Critical review of flexible pavement maintenance management systems, *International Journal of Engineering & Technology*. 8 (2019) pp.95–101. <https://doi.org/10.13140/RG.2.2.18001.68967>.
 - [8] L. Chen, Q. Bai, Optimization in decision making in infrastructure asset management: A review, *Applied Sciences*. 9 (2019) pp.1380. <https://doi.org/10.3390/app9071380>.
 - [9] P.D. Thompson, E. P.Small, M. Johnson, A. R.Marshall, The Pontis bridge management system, *Structural Engineering International*. 8 (1998) pp.303–308. <https://doi.org/10.2749/101686698780488758>.
 - [10] H. Hawk, E.P. Small, The BRIDGIT bridge management system, *Structural Engineering International*. 8 (1998) pp.309–314. <https://doi.org/10.2749/101686698780488712>.
 - [11] E. Gamma, R. Helm, R. Johnson, John M. Vlissides, *Design patterns: elements of reusable object-oriented software*, Pearson Education India, (1995), ISBN:9780321700698.
 - [12] V. Asghari, S.-C. Hsu, GIAMS, (2020). <https://github.com/vd1371/GIAMS> (accessed January 6, 2021).
 - [13] FEMA-NIBS, Hazus–MH 2.1: Technical manual, (2015). www.fema.gov/plan/prevent/hazus (accessed January 6, 2021).
 - [14] USGS, United States Geological Survey, (2020). <https://earthquake.usgs.gov/earthquakes/search/> (accessed January 6, 2021).
 - [15] M. Mahsuli, T. Haukaas, Computer program for multimodel reliability and optimization analysis, *Journal of Computing in Civil Engineering*. 27 (2013) pp.87–98. [https://doi.org/10.1061/\(asce\)cp.1943-5487.0000204](https://doi.org/10.1061/(asce)cp.1943-5487.0000204).
 - [16] M. Pagani, D. Monelli, G. Weatherill, L. Danciu, H. Crowley, V. Silva, et al., OpenQuake engine: An open hazard (and risk) software for the global earthquake model, *Seismological Research Letters*. 85 (2014) pp.692–702. <https://doi.org/10.1785/0220130087>.
 - [17] L. Raso, D. Dorchies, J. Kwakkel, P.O. Malaterre, Optimist: A python library for water system optimal operation and analysis using SDDP, in: *International Congress on Environmental Modelling and Software, IEMSs*, (2016) pp. 926–932. <https://scholarsarchive.byu.edu/iemssconference/2016/Stream-D/26/> (accessed January 6, 2020).
 - [18] C. Warren, A. Giannopoulos, I. Giannakis, gprMax: Open source software to simulate

- electromagnetic wave propagation for Ground Penetrating Radar, *Computer Physics Communications*. 209 (2016) pp.163–170. <https://doi.org/10.1016/j.cpc.2016.08.020>.
- [19] V.K. Gadi, D. Alybaev, P. Raj, A. Garg, G. Mei, S. Sreedeeep, et al., A novel python program to automate soil colour analysis and interpret surface moisture content, *International Journal of Geosynthetics and Ground Engineering*. 6 (2020) pp.1–8. <https://doi.org/10.1007/s40891-020-00204-3>.
- [20] M. Zhu, F. McKenna, M.H. Scott, OpenSeesPy: Python library for the OpenSees finite element framework, *SoftwareX*. 7 (2018) pp.6–11. <https://doi.org/10.1016/j.softx.2017.10.009>.
- [21] X. Guan, H. Burton, T. Sabol, Python-based computational platform to automate seismic design, nonlinear structural model construction and analysis of steel moment resisting frames, *Engineering Structures*. 224 (2020) pp.111199. <https://doi.org/10.1016/j.engstruct.2020.111199>.
- [22] U.T. Jagadale, C.B. Nayak, A. Mankar, S.B. Thakare, W.N. Deulkar, An experimental-based python programming for structural health monitoring of non-engineered RC frame, *Innovative Infrastructure Solutions*. 5 (2020) pp.1–10. <https://doi.org/10.1007/s41062-020-0260-x>.
- [23] J.A. Rivera S, A.F. Estupiñán L, Development of a computational software in Python, used to study the materials resistance in beams, *ArXiv E-Prints*. (2020) pp.arXiv:2009.09448v2. <http://arxiv.org/abs/2009.09448v2> (accessed January 6, 2021).
- [24] B.E. McDonnell, K. Ratliff, M.E. Tryby, J.J.X. Wu, A. Mullapudi, PySWMM: The Python interface to stormwater management model (SWMM), *Journal of Open Source Software*. 5 (2020) pp.2292. <https://doi.org/10.21105/joss.02292>.
- [25] A. Zeraoui, M. Benzerzour, W. Maherzi, R. Mansi, N.E. Abriak, New software for the optimization of the formulation and the treatment of dredged sediments for utilization in civil engineering, *Journal of Soils and Sediments*. 20 (2020) pp.2709–2716. <https://doi.org/10.1007/s11368-020-02605-3>.
- [26] L. Torvalds, Linux [Operating System], *GitHub*. (2020). <https://github.com/torvalds/linux> (accessed January 6, 2021).
- [27] S.S. Levine, M.J. Prietula, Open collaboration for innovation: principles and performance, *Organization Science*. 25 (2014) pp.1414–1433. <https://doi.org/10.1287/orsc.2013.0872>.
- [28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, et al., Scikit-learn: machine learning in Python, *Journal of Machine Learning Research*. 12 (2011) pp.2825–2830. <http://jmlr.org/papers/v12/pedregosa11a.html>.
- [29] F. Chollet, KERAS, *GitHub*. (2015). <https://github.com/keras-team/keras> (accessed January 6, 2021).
- [30] Y. Li, Y. Dong, D.M. Frangopol, D. Gautam, Long-term resilience and loss assessment of highway bridges under multiple natural hazards, *Structure and Infrastructure Engineering*. (2020) pp.626–641. <https://doi.org/10.1080/15732479.2019.1699936>.
- [31] M. Cheng, D.Y. Yang, D.M. Frangopol, Investigation of effects of time preference and risk perception on life-cycle management of civil infrastructure, *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part A: Civil Engineering*. 6 (2020) pp.04020001. <https://doi.org/10.1061/AJRUA6.0001039>.
- [32] F. Ghodoosi, S. Abu-Samra, M. Zeynalian, T. Zayed, Maintenance cost optimization for bridge structures using system reliability analysis and genetic algorithms, *Journal of Construction Engineering and Management*. 144 (2018) pp.04017116. [https://doi.org/10.1061/\(ASCE\)CO.1943-7862.0001435](https://doi.org/10.1061/(ASCE)CO.1943-7862.0001435).
- [33] Q. Bai, S. Labi, K.C. Sinha, P.D. Thompson, Multiobjective optimization for project

- selection in network-level bridge management incorporating decision-maker's preference using the concept of holism, *Journal of Bridge Engineering*. 18 (2013) pp.879–889. [https://doi.org/10.1061/\(ASCE\)BE.1943-5592.0000428](https://doi.org/10.1061/(ASCE)BE.1943-5592.0000428).
- [34] G. Lamptey, S. Labi, Z. Li, Decision support for optimal scheduling of highway pavement preventive maintenance within resurfacing cycle, *Decision Support Systems*. 46 (2008) pp.376–387. <https://doi.org/10.1016/j.dss.2008.07.004>.
- [35] P.D. Thompson, D. Beckstrand, A. Mines, M. Vessely, D. Stanley, B. Benko, Geotechnical asset management plan: Analysis of life-cycle cost and risk, *Transportation Research Record*. 2596 (2016) pp.36–43. <https://doi.org/10.3141/2596-05>.
- [36] M.A. Cardoso, M.C. Almeida, M. Santos Silva, Sewer asset management planning – implementation of a structured approach in wastewater utilities, *Urban Water Journal*. 13 (2016) pp.15–27. <https://doi.org/10.1080/1573062X.2015.1076859>.
- [37] AASHTO, Manual for bridge element inspection, American association of state highway and transportation officials, (2015), ISBN:9781560516224.
- [38] D.Y. Yang, D.M. Frangopol, Life-cycle management of deteriorating bridge networks with network-level risk bounds and system reliability analysis, *Structural Safety*. 83 (2020) pp.101911. <https://doi.org/10.1016/j.strusafe.2019.101911>.
- [39] Y. Jeong, W.S. Kim, I. Lee, J. Lee, Bridge inspection practices and bridge management programs in China, Japan, Korea, and U.S., *Journal of Structural Integrity and Maintenance*. 3 (2018) pp.126–135. <https://doi.org/10.1080/24705314.2018.1461548>.
- [40] FHWA, Recording and coding guide for the structure inventory and appraisal of the nation's bridges, Federal Highway Administration, Washington, DC, (2012), ISBN:978-1249169666.
- [41] K. Golabi, R.B. Kulkarni, G.B. Way, A statewide pavement management system, *INFORMS Journal on Applied Analytics*. 12 (1982) pp.5–21. <https://doi.org/10.1287/inte.12.6.5>.
- [42] B.R. Ellingwood, Risk-informed condition assessment of civil infrastructure: state of practice and research issues, *Structure and Infrastructure Engineering*. 1 (2005) pp.7–18. <https://doi.org/10.1080/15732470412331289341>.
- [43] S.M. Ross, Introduction to probability models, 10th ed., Academic Press, (2010), ISBN:9780123756862.
- [44] W. Nicholson, S. Christopher, Microeconomic theory: Basic principles and extensions, South-Western College Pub, (2011), ISBN:978-1111525538.
- [45] L. Liu, D.M. Frangopol, A. Mondoro, D.Y. Yang, Sustainability-informed bridge ranking under scour based on transportation network performance and multiattribute utility, *Journal of Bridge Engineering*. 23 (2018) pp.04018082. [https://doi.org/10.1061/\(ASCE\)BE.1943-5592.0001296](https://doi.org/10.1061/(ASCE)BE.1943-5592.0001296).
- [46] Z. Li, K.C. Sinha, Methodology for multicriteria decision making in highway asset management, *Transportation Research Record*. 1885 (2004) pp.79–87. <https://doi.org/10.3141/1885-12>.
- [47] H. Talebiyan, M. Mahsuli, Risk-based prioritization of a building portfolio for retrofit, *Journal of Structural Engineering*. 144 (2018) pp.04017181. [https://doi.org/10.1061/\(ASCE\)ST.1943-541X.0001927](https://doi.org/10.1061/(ASCE)ST.1943-541X.0001927).
- [48] D.Y. Yang, D.M. Frangopol, Risk-based portfolio management of civil infrastructure assets under deep uncertainties associated with climate change: a robust optimisation approach, *Structure and Infrastructure Engineering*. 16 (2020) pp.531–546. <https://doi.org/10.1080/15732479.2019.1639776>.
- [49] TexasDOT, Estimate of road user cost for personal vehicles and commercial trucks,

- (2020). <https://www.txdot.gov/inside-txdot/division/construction/road-user-costs.html> (accessed January 6, 2021).
- [50] D. Whitley, A genetic algorithm tutorial, *Statistics and Computing*. 4 (1994) pp.65–85. <https://doi.org/10.1007/BF00175354>.
- [51] V. Patidar, S. Labi, T. Morin, P.D. Thompson, K.C. Sinha, Evaluating methods and algorithms for multicriteria bridge management at the network level, *Transportation Research Record*. 2220 (2011) pp.38–47. <https://doi.org/10.3141/2220-05>.
- [52] J.J. Grefenstette, Optimization of control parameters for genetic algorithms, *IEEE Transactions on Systems, Man, and Cybernetics*. 16 (1986) pp.122–128. <https://doi.org/10.1109/TSMC.1986.289288>.
- [53] R. Edward Best, Modeling, optimization, and decision support for integrated urban and infrastructure planning, Stanford University, (2016). <https://www.proquest.com/docview/2454193414/977DFBDF0BAB429APQ/1?accountid=16210> (accessed January 6, 2021).
- [54] F. Chicano, A.M. Sutton, L.D. Whitley, Fitness probability distribution of Bit-Flip mutation, *Evolutionary Computation*. 23 (2015) pp.217–248. https://doi.org/10.1162/EVCO_a_00130.
- [55] C. Witt, Tight bounds on the optimization time of a randomized search heuristic on linear functions, *Combinatorics, Probability & Computing*. 22 (2013) pp.294–318. <https://doi.org/10.1017/S0963548312000600>.
- [56] D.M. Frangopol, Life-cycle performance, management, and optimisation of structural systems under uncertainty: accomplishments and challenges, *Structure and Infrastructure Engineering*. 7 (2011) pp.389–413. <https://doi.org/10.1080/15732471003594427>.
- [57] S. Kim, D.M. Frangopol, Decision making for probabilistic fatigue inspection planning based on multi-objective optimization, *International Journal of Fatigue*. 111 (2018) pp.356–368. <https://doi.org/10.1016/j.ijfatigue.2018.01.027>.
- [58] P. Xuan, M.Z. Guo, J. Wang, C.Y. Wang, X.Y. Liu, Y. Liu, Genetic algorithm-based efficient feature selection for classification of pre-miRNAs, *Genetics and Molecular Research*. 10 (2011) pp.588–603. <https://doi.org/10.4238/vol10-2gmr969>.
- [59] I.T. Yang, Y.M. Hsieh, L.O. Kung, Parallel computing platform for multiobjective simulation optimization of bridge maintenance planning, *Journal of Construction Engineering and Management*. 138 (2012) pp.215–226. [https://doi.org/10.1061/\(ASCE\)CO.1943-7862.0000421](https://doi.org/10.1061/(ASCE)CO.1943-7862.0000421).
- [60] J.G. Digalakis, K.G. Margaritis, On benchmarking functions for genetic algorithms, *International Journal of Computer Mathematics*. 77 (2001) pp.481–506. <https://doi.org/10.1080/00207160108805080>.
- [61] Q. Chen, Y. Chen, W. Jiang, Genetic particle swarm optimization-based feature selection for very-high-resolution remotely sensed imagery object change detection, *Sensors (Switzerland)*. 16 (2016). <https://doi.org/10.3390/s16081204>.
- [62] S. Thöns, On the value of monitoring information for the structural integrity and risk management, *Computer-Aided Civil and Infrastructure Engineering*. 33 (2018) pp.79–94. <https://doi.org/10.1111/mice.12332>.
- [63] D. Straub, M.H. Faber, Risk based inspection planning for structural systems, *Structural Safety*. 27 (2005) pp.335–355. <https://doi.org/10.1016/j.strusafe.2005.04.001>.
- [64] A.D. Orcesi, D.M. Frangopol, Optimization of bridge maintenance strategies based on structural health monitoring information, *Structural Safety*. 33 (2011) pp.26–41. <https://doi.org/10.1016/j.strusafe.2010.05.002>.
- [65] W.J. Klerk, T. Schweckendiek, F. Den Heijer, M. Kok, Value of information of

structural health monitoring in asset management of flood defences, Infrastructures. 4
(2019) pp.1–20. <https://doi.org/10.3390/infrastructures4030056>.
[66] Github, (2020). <https://github.com> (accessed January 6, 2021).

Appendix A

This section defines basic terminology and concepts for open collaboration in further developing the platform by volunteer researchers. Open collaboration in software development has been enabled by using distributed version-control systems that facilitate software development among developers. Git is one of the most popular version control systems and through the help of online and free access repositories (e.g., GitHub [66]), which are usually called remote repositories, forms the foundation of numerous projects. Collaborators will need to have the git installed on their systems and to own a personal GitHub page. Four common steps executed in git and GitHub, namely fork, clone, edit, and pull request, involve contributing and developing with an open collaboration mindset. These steps will be summarized in the following paragraphs:

Step 1: Fork

Forking refers to the process in which one collaborator makes a copy of others' repositories (e.g., GIAMS repository) to her Github page. This step can be easily carried out using features of GitHub available on the repository page.

Step 2: Clone

Cloning refers to the process of downloading the source code to a folder on the local machine for further development. This is usually performed as the second step in contributing to others' work.

Step 3: Edit

The editing process involves developing code, committing the code to the version control system (git), and pushing the updated code to the remote repository. In short, saving the changes in the git is called committing, and uploading the code to the remote repository is called pushing the code. Git enables developers to change and update files in branches instead of directly changing the code. The main branch, which is called the master branch in git, usually remains untouched during the development. Other user-defined branches will be assigned for further development of the framework. These branches will finally be merged to the master branch by the principal contributors. Figure A.1 depicts the notion of branches in software development in an abstract form.

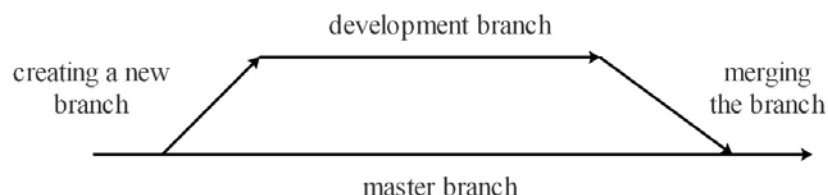


Fig. A.1. Git branches in abstract form

Step 4: Pull request

After editing and pushing the changes, the contributor should inform the main contributor to the project about the updates on a project. The submission of a pull request refers to the process

of asking the repository owner to merge the updated code to the original source code. This task is initiated on the contributor's own GitHub page after pushing the code to the remote repository. The main or other contributors will then review the updates, and, in case of validity, merge them to the source code.

Apart from directly contributing to the source code, researchers could raise issues or make feature requests in the *Issues* section of the Github repositories. The conventional response of the contributors of a repository is replying to communications regarding issues or feature requests and updating the code.

Appendix B

This section briefly discusses three examples that are other extensions for GIAMS. These examples could be used as simple guides for future developments and extensions.

Example1: A new deterioration model

This example illustrates the process of implementing a new deterioration models and contributing to the development of GIAMS. In this example, an imaginary researcher called "GIAMS Researcher" has developed a deterioration model that can predict the future condition rating of bridge elements located in Indiana as a function of bridges' latitude (Lat.) and longitude (Long.) as well as previous condition of elements. Table B.1 describes all the required steps for contributing her model, referred to here as *LatLongFunc*, to the GIAMS repository to be accessed and used publicly.

Table B.1. The steps for adding a new deterioration model to GIAMS

	Task	Code
1	Forking the GIAMS repository to her page	
2	Cloning the GIAMS in a folder	In git bash: <i>git clone https://github.com/giams_researchers/GIAMS.git</i>
3.1	Adding the Lat. And Long. of bridges to Indiana .CSV file to columns 21 and 22	In Network/Networks/INDIANA2019.csv
3.2	Adding a method to set these attributes to the Bridge	In Asset/AssetTypes/Bridge.py: def set_latitude_longitude(self, lat, long): self.lat, self.long = lat, long In Asset/Elements/Deterioration:
3.3	Adding the deterioration model	- Copying and renaming one of the previously provided models to <i>LatLong.py</i> - Changing the name of the class to <i>LatLong</i> - Changing the <i>predict_condition</i> of the <i>LatLong.py</i> : def predict_condition(previous_condition): return LatLongFunc (self.asset.lat, self.asset.long, previous_condition) In the <i>load_asset</i> method of Network/IndianaNetwork.py:
3.4	Updating the network loader	lat, long = asset_info[21:23] asset.set_latitude_longitude(lat, long)
3.5	Pushing the changes to the remote repository	*element.set_deterioration_model(LatLong()) In git bash: - git branch langlat_branch

- git add .
- git commit
- git push origin langlat_branch

4 Making a pull request on her repository branch

The main contributors will then evaluate the pull request and in case of validity will merge it with the source code. In this example, the “GIAMS Researcher” could use the readily developed platform in several simple steps without writing another asset management system from scratch. This would save enormous time and effort expended by asset management researchers. Although not all developments and modifications are as simple as this example, the general procedure remains intact and the proposed framework remains applicable.

Example2: Prioritizing seismic retrofit of building structures

Taking structural seismic retrofit planning of a portfolio of buildings in a state/province as an example, this section discusses the extent of required modifications for new area applications to be developed in GIAMS. The modification of GIAMS’ modules varies depending on the nature of the problem. Analyzing new problems would require more rigorous modifications in comparison to the example 1. A number of new modules should be developed for this purpose while the rest of GIAMS’ modules could be directly or with minor modifications to be used. For example, a new module for buildings should be developed, while some other modules like hazard generators could be readily used with minor or no modifications. Table B.2 summarizes the extent of modification and development of modules that are required for this example.

Table B.2. The extent of changes required for GIAMS’ modules for structural seismic retrofit

New modules required		Minor modifications/No Change	
Asset/AssetTypes	/Building	Asset/Elements	/Condition rating /Deterioration models
Asset/Elements	/Structure /AgencyCost /Utility	Asset/HazardModels	/Generator /Recovery
Asset/HazardModels	/Loss /Response	Asset/MRRModels	/MRRTwoActions /MRREffectiveness
Asset/UserCostModels	/UserCosts	LifeCycleAnalyzer	/Simulator /LCA
Network	/BuildingPortfolio	Optimizer	/GA /IUC
LifeCycleAnalyzer/Simulator Optimizer	/RiskAnalyzer /Rank		

The developed dummy models for the abovementioned problem could be readily found in the GIAMS repository. During first time extension of GIAMS for retrofit prioritization of buildings, several new modules should be developed for this purpose, though the majority of GIAMS’ modules could be used directly or with minor modifications. After the first-time implementation of retrofit prioritization of buildings, the extension of GIAMS for similar purposes would be merely limited to minor modifications of its modules. As a result, the efforts of researchers and collaborators that will be built upon each other will make future modifications and extensions of GIAMS easier.

Example3: Maintenance optimization by considering structural health monitoring

Maintenance optimization of assets based on structural health monitoring (SHM) results is another common task in asset management. Similar to example 2, this section provides a brief

guideline for adapting GIAMS to consider SHM updates in the life cycle of assets with the focus on bridge structures. SHM results usually affect the previously designated MRR decisions, deterioration models, and future inspection planning [62–65]. Therefore, a few new modules such as a new simulator should be written from scratch, a few modules should be modified, while the rest of the framework could be directly used. Table B.3 provides a summary of the extent of GIAMS’ modifications that are required for this example.

Table B.3. The required modifications for GIAMS’ modules for adopting SHM

New modules required		Minor modifications/No Change	
Asset/MRRModels	/SHMActions	Asset/Elements	/Deterioration models
	/SHMEffectiveness		
Asset/Elements	/SHMCost	Asset/UserCostModels	/UserCosts
LifeCycleAnalyzer	/SHMSimulator	Asset/HazardModels	/Generator
			/Recovery
			/Loss
			/Response
Network	/SHMNetwork	LifeCycleAnalyzer	/LCA
		Optimizer	/GA
			/IUC

Dummy models are used in the of the abovementioned modules, which could be found in the GIAMS repository, to illustrate the extensibility of GIAMS. Similar to the example 2, after the first-time adopting structural health monitoring in GIAMS, the extension of GIAMS for more advanced implementations of structural health monitoring will be limited to minor modifications. Although the current version of GIAMS is not designed to cover all domains of infrastructure asset management, its design pattern and accessibility will make further developments simpler than developing new AMS from scratch.

Appendix C

The directory tree of the GIAMS repository is summarized as follows:

GIAMS	.
---Asset/	
---AssetTypes/	---LifeCycleAnalyzer/
---Bridge	---Simulators/
---other assets...	---BaseSimulator
---Elements/	---other models...
---AgencyCost/	---BaseLCA
--- BaseAgencyCost	---LCA
--- other models...	---other models...
---ConditionRating/	---Network/
--- BaseConditionRating	---Networks/
--- other models...	---network csv files
---Deterioration/	---BaseNetwork
--- BaseDeterioration	---other models...
--- other models...	---Optimizer/
---Utility/	---GA
--- BaseUtility	---IUC
--- other models...	---reports/
---BaseElement	---utils/
---BridgeElement	
---HazardModels/	
---Generator/	

```

|
|      |--- BaseHazard
|      |--- other models...
|---Loss/
|      |--- BaseHazardLoss
|      |--- other models...
|---Recovery/
|      |--- BaseRecovery
|      |--- other models...
|---Response/
|      |--- BaseResponse
|      |--- other models...
|---HazardModel
|---MRRModels/
|      |---EffectivenessModels/
|      |      |--- BaseMRREffectiveness
|      |      |--- other models...
|      |---BaseMRRPlan
|      |---other models
|---UserCostModels/
|      |--- BaseUserCost
|      |--- other models...
.
.
.

```