Hindawi
Journal of Advanced Transportation
Volume 2018, Article ID 7284815, 11 pages
https://doi.org/10.1155/2018/7284815

Research Article

PULSim: User-Based Adaptable Simulation Tool for Railway Planning and Operations

Yong Cui[1], **Ullrich Martin**[1] **and Jiajian Liang**[2]

[1]*Institut fuer Eisenbahn- und Verkehrswesen der Universitaet Stuttgart, Pfaffenwaldring 7, 70569 Stuttgart, Germany*
[2]*China Academy of Railway Sciences, Daliushu Road 2, Haidian District, Beijing 100081, China*

Correspondence should be addressed to Yong Cui; yong.cui@ievvwi.uni-stuttgart.de

Received 13 October 2017; Revised 12 January 2018; Accepted 5 February 2018; Published 22 April 2018

Academic Editor: Lingyun Meng

Simulation methods are widely used in the field of railway planning and operations. Currently, several commercial software tools are available that not only provide functionality for railway simulation but also enable further evaluation and optimisation of the network for scheduling, dispatching, and capacity research. However, the various tools are all lacking with respect to the standards they utilise as well as their published interfaces. For an end-user, the basic mechanism and the assumptions built into a simulation tool are unknown, which means that the true potential of these software tools is limited. One of the most critical issues is the lack of the ability of users to define a sophisticated workflow, integrated in several rounds of simulation with adjustable parameters and settings. This paper develops and describes a user-based, customisable platform. As the preconditions of the platform, the design aspects for modelling the components of a railway system and building the workflow of railway simulation are elaborated in detail. Based on the model and the workflow, an integrated simulation platform with open interfaces is developed. Users and researchers gain the ability to rapidly develop their own algorithms, supported by the tailored simulation process in a flexible manner. The productivity of using simulation tools for further evaluation and optimisation will be significantly improved through the user-adaptable open interfaces.

1. Introduction

In order to improve the efficiency and effectiveness of railway planning and operations, the relationships between the critical components of railway systems and the behaviour of the studied system must be investigated. A specific investigated problem usually presents many possible alternatives for design. In order to enable evaluation and comparison among the various possible alternatives, their respective effects and outputs will be predicted through experiments. However, it is not practical to examine a design alternative directly based upon an actual system, because either such a system is not existing or experimentation upon it is prohibitively expensive.

For a simple railway network with a limited number of train runs, an analytical approach is practical in order to evaluate the output of a design alternative and even to find an optimal solution in a closed-form expression. For a large-scale network with a high density of train movements, the computational complexity of such an analytical model becomes considerably high. In this case, the simulation approach is often applied.

Several simulation tools are available for railway planning and operations purposes. The application of personal computer based simulation approaches for railway systems began in the 1990s, at which time most tools were originally developed as laboratory versions. Along with continuous development and applications, these tools have been widely accepted for research and commercial purposes. Meanwhile, a large number of data and models of railway systems have become available. In this paper, the existing simulation tools and applications will be presented in Section 2.

However, the potential of simulation approaches has not yet been sufficiently utilised. The limits of the current simulation tools, especially their lack of capability for extension and customisation, are discussed in Section 2. To overcome these limits, a user-based, adaptable simulation tool called PULSim has been developed in recent years. In Section 3,

the model, the workflow, and the main features of PULSim are introduced. The software's capability to provide users and third-party applications with an open interface for dynamic interaction and flexible extension is presented in Section 4, along with several use cases and scenarios. Finally, the perspective of further development is discussed in Section 6.

2. Applications of Railway Simulation and the Limitations of Existing Simulation Tools

For railway planning and operations, simulation approaches have been widely used by researchers, railway infrastructure companies, and railway operating companies. In German-speaking countries and Europe in general, the simulation software RailSys (developed by Rail Management Consultants GmbH, [1]), OpenTrack (from OpenTrack Railway Technology Ltd., [2]), and LUKS (by VIA Consulting & Development GmbH, [3]) are very practical for railway planning and operations.

The potential areas of application of simulation tools include technical feasibility studies, determination of conflicts, evaluation of the quality of a timetable, capacity research, and dispatching. Simulation tools can be used, for example, to build a test environment for the evaluation of the technical feasibility and the benefits of using the European Train Control System (ETCS) [4]. To investigate potential conflicts, a timetable simulation can be carried out, and the resulting waiting time can be determined as well [5, 6]. In certain situations, various optimisation functions can be added to a timetable simulation in order to reduce hindrances and waiting time of train runs. As a deterministic process in principle, the possible influence and variances during railway operations are not considered in a timetable simulation. If the robustness and stability of a timetable are matters of concern, a so-called operational simulation can be carried out, in which stochastic influences will be introduced. The robustness and stability of the investigated timetable can be derived according to statistical indicators, which are calculated based on the outputs of several rounds of simulation with different randomly generated influences [7, 8]. In the area of capacity research, system performance can be evaluated through simulating randomly generated timetables for a certain operating program, a process which can, in addition, identify bottlenecks and determine the quality of an operating program [9]. The approach of capacity research is implemented in the software tool PULEIV, developed by the IEV (Institut für Eisenbahn- und Verkehrswesen der Universitaet Stuttgart) for flexible analysis, evaluations and reports [10]. Simulation approaches can not only be used to identify conflicts, but also to generate a feasible dispatching timetable to resolve conflicts [11, 12], as well as to study the relationship between systems performance and dispatching measurements [13, 14].

At the beginning of development of a certain simulation tool, its further possible applications are often unknown by its developers. Therefore, it is difficult to foresee and to provide specifically desired functions and outputs, even if the simulation tool would theoretically able to fulfil said requirements. Hence, an open interface that enables customisation for user-based, adaptable simulation processes is a vital factor for the success of a simulation tool. Unfortunately, currently available simulation tools for railway planning and operations lack, in large part, transparent standards and open interfaces. Some internal workflows, such as the applied dispatching algorithm and the implementation of signalling systems, are not sufficiently documented for users and third party individuals and, as such, it is difficult for the end-user to understand the internal assumptions and simplifications made within these tools. The outputs of various simulation tools are insufficient and not standardised with insufficient documentation, which prevents users from carrying out further evaluation and optimisation. In addition, it is almost impossible to organise user-defined workflows and to integrate new functions into the existing simulation tools, which leads to constraints in their usability and applicability. The high efforts required for further evaluation decrease the efficiency and effectiveness of these simulation approaches. In addition, some well-known issues, such as deadlock problems in synchronous simulations, have not been addressed sufficiently. Once deadlocks take place, the simulation process must be cancelled or be solved manually, which greatly limits the usability of the simulation tool as well.

A user-based, adaptable simulation tool, PULSim, has been developed in recent years. In [15, 16], the tool was known as DoSim. The name has been changed to PULSim in order to conform to the naming convention of software tools in the IEV. PULSim provides a platform for railway simulation with a unified model, a transparent workflow, and open interfaces. Critically, PULSim provides the possibility for third-party applications to be flexibly integrated into the software. The functionality of PULSim can be extended thanks to its open interface, and the software has been tested with several railway networks in Germany. At the moment, the software is available for download from GitHub (https://github.com/herrcui/RailView/wiki), with the instructions for installing and using it provided with demo data. The user manual and the open interfaces are planned to be comprehensively documented and published in further development. The Continuous Integration (CI) for open access and updates will be implemented as well. In this paper, the model, the workflow, and the Graphical User Interface (GUI) are presented. Three special features of PULSim are outlined in particular:

(i) Dispatching mechanism

(ii) Deadlock avoidance

(iii) Open interfaces for user-based adaptable simulation.

An introduction of the model, the workflow, and the GUI of PULSim is provided in Section 3.1. In Sections 3.2 and 3.3, the design of the dispatching mechanism and deadlock avoidance are explained. The open interface for user-based adaptable simulation and its applications are introduced in Sections 4 and 5.

3. Introduction of Simulation Tool PULSim

3.1. The Model, the Workflow, and the GUI of PULSim. In PULSim, the infrastructure, rolling stocks, and railway

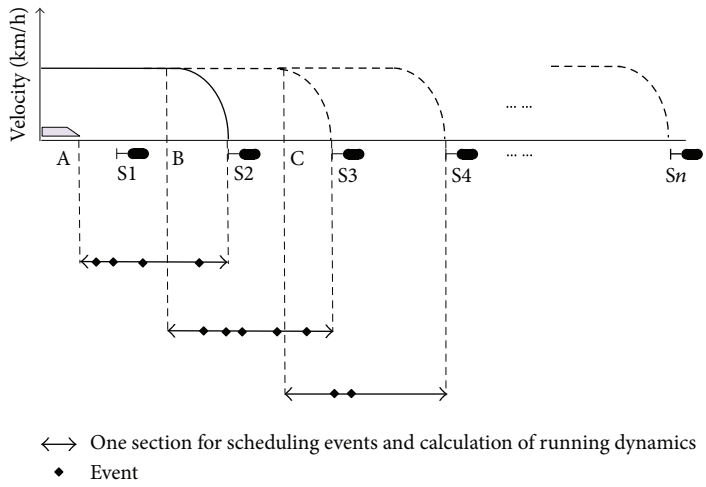

Figure 1: An example of event scheduling and calculation of running dynamics.

operations are modelled as components of railway systems. Attributes are used to describe the static and the dynamic information. The static information refers to the invariant values of the railway systems, for example, the length of a track, the configuration of a train, and the scheduled departure time of a train run. The dynamic information will vary along the duration of the simulation. For example, the aspect of a signal, the running dynamics of a train, and the occupancy situation of a track are updated continuously during the simulation. The model applied in PULSim provides a basis for multiscale simulation on the microscopic, mesoscopic, and macroscopic levels [17].

An internal class model of railway systems comes built-in with PULSim. This model is the further development and extension from the IEV core model [18]. The publication of the detailed design of this model, and its workflow, is planned for 2018. It will also consist of the details of the open interfaces for interacting with the model along with the software. Particularly, the basic structures and the path components based on the infrastructure element model are developed in the core model, so that the macroscopic, mesoscopic, and the microscopic models are fully integrated (the infrastructure element model is a microscopic model to model the basic infrastructure elements including tracks, turnouts, crossings, and single and double slips. A basic structure is defined as a basic occupancy element in which all parts should be equally occupied, regardless of the design of the operating program. A path component is a directed edge inside a block section used for train runs, which can be released separately as a directed occupancy element. Basic structures and path components are modelled for the mesoscopic level). This model can be used for various purposes, including capacity research, bottleneck analysis, and train dispatching [19]. It is independent of a concrete data format, which allows users to concentrate on the workflow and the business logic of railway planning and operations without requiring knowledge relating to a certain data format. Additional parsers are required to convert different data formats into the class model. At the moment, the data format used by RailSys [1] is supported by PULSim. The development of the parser to support railML [20] is in progress.

The workflow for an event-driven simulation is applied in PULSim. An event is defined as an occurrence that may change the attributes of the system at a certain point in time. During the simulation process, a series of time points for the occurrence of an event are identified and continuously updated. The system evolves over the series of time points. During a simulation process, the attributes (dynamic information) of the system (e.g., the position of a train) may be changed continuously. In PULSim, only that change of attribute, which causes interactions between two or more components or triggers another event, is considered a discrete event. Otherwise, the changes of the running dynamics, for example, the continuous variation of train velocity, will be handled at each discrete time point by the train itself, without requiring an explicit event. Hence, the event-driven process in PULSim can be organised efficiently with a limited number of events.

During the simulation process, a running time calculation is carried out for each individual train run. The change of position and velocity of each train will not be treated as an event. Instead, the events will be scheduled with respect to the signalling system. An example of organising an event-driven simulation is shown in Figure 1. After a train has received a Movement Authority (MA) at point A, the complete speed profile will be calculated until the End of Authority (EOA) at main signal S2. The corresponding events from point A to EOA S2 are scheduled. At point B, a new MA ending at main signal S3 is granted. The train movement from B to S3 will be calculated. Meanwhile, the events already scheduled between B and S2 will be discarded, since the running dynamics from B to S2 will be updated with the newly calculated speed profile. The events from B to S3 will be rescheduled and updated again.

Once a request of infrastructure resources is sent, the current operational situation will be observed by the dispatching module. The requested infrastructure resources can only be granted at a conflict-free and deadlock-free situation.

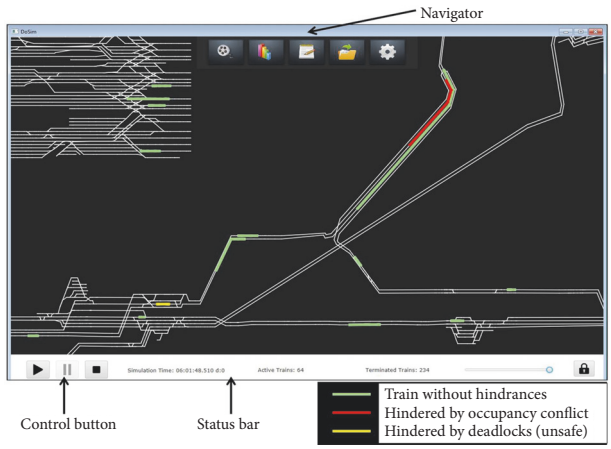

Figure 2: Screenshot of "Network/Train Run" view in PULSim.

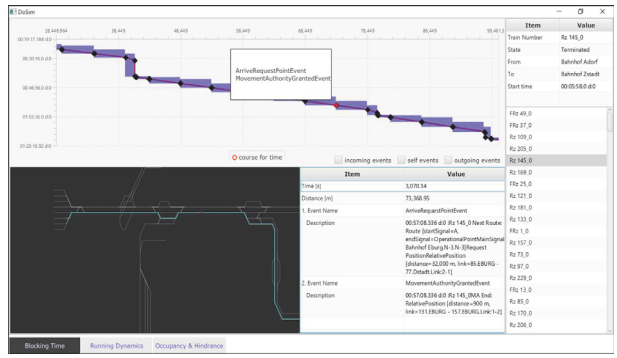

Figure 3: Screenshot of tracing events and train run.

A simple First Come First Served (FCFS) algorithm is provided by PULSim as a default module to solve occupancy conflicts. It can be replaced with a customised dispatching module provided by the user (see Section 3.2). A standard solution for deadlock avoidance, combined with the approach of searching for feasible resources to reduce false-positive situations, is provided in PULSim as well (see Section 3.3).

The design of the GUI in PULSim aims to provide the user with an insightful view during the process of railway simulation and to support efficient user interaction. Therefore, the concepts of visualisation and interaction are considered critical areas of focus. Various user interfaces are organised in different views. The movement of train runs in the railway network (the view of "Network/Train Run"), the information behind the scenes ("Analyse and Evaluation"), and the interaction between the simulation tool and users ("Open Interface") are provided.

In Figure 2, the screenshot of the "Network/Train Run" view is presented. This is a standard function provided by almost all railway simulation tools. A control panel can start, pause, or stop a simulation. Trains are marked in green, red, or yellow, which represent a train without hindrances, with hindrances due to occupancy conflicts, or with hindrances due to deadlocks, respectively. In order to enable a quick switch among different views, a floating navigator is set on top of the screen.

The function of tracing the occurrence of events is provided by PULSim. Within this view, users can observe the entire workflow of the simulation within the diagram of the blocking time stairway and running dynamics for each train run (Figure 3). Other evaluation views for the illustration of occupancy and hindrance of the running simulation are also provided. In addition, an "Open Interface" view to achieve a user-based, adaptable simulation is built into PULSim (see Section 4).

3.2. Dispatching Mechanism of PULSIM. With conventional dispatching approaches [21], the dispatching algorithm is activated once one or more potential conflicts are identified. However, it might be too late to start train dispatching at the time of the occurrence of conflicts. Furthermore, it is necessary to check for potential conflicts within the conventional dispatching systems periodically. The timing for identification of conflicts is critical. On one hand, some conflicts may be overlooked using a long time interval between two rounds of checks; on the other hand, overly frequent checks might lead to performance issues.

In PULSim, a new conflict identification/resolution mechanism is implemented. In contrast with other conventional dispatching approaches, the activation of a designed algorithm for train dispatching is not initiated at the time of facing conflicts, but at an earlier time, at which a train has the chance to occupy the requested infrastructure resources. For example, if the action to occupy the requested resources would produce significant hindrances on other trains, the train will give up the chance of occupancy. Therefore, potential conflicts from potential hindrances can be prevented in advance. In addition, this mechanism suits the workflow of event-driven simulation. A periodical examination of conflicts in a fixed time interval is not necessary; any potential conflicts can be identified and resolved at the time a Movement Authority is able to be granted.

In order to better illustrate this phenomenon, Figure 4 provides an example of giving up the opportunity of occupancy [15]. Within it, Train T1 is requesting occupancy of the route from signals S1 to S3. This route is not currently occupied by other trains (conflict-free), and its occupancy will not cause deadlocks (deadlock-free). Deadlocks may, indeed, be regarded as a type of conflict in train dispatching. In order to differentiate between occupancy conflicts and deadlocks, the term conflict-free and deadlock-free are used in this paper. There is a train T0 occupying the route from S3 to S5. Its occupancy time along this route is assumed to be very high due to a technical failure of train T0. If train T1 occupies the route from S1 to S3, trains T2 and T3, which will take the route from S2 to S4, must wait until the technical failure of T0 has been resolved. It would be more efficient if train T1 were to give up the chance of occupancy and wait before signal S2. Therefore, potential hindrances and conflicts can be reduced or avoided in advance.

The workflow of the dispatching mechanism is shown in Figure 5. As usual, a request will be pended in case of conflicts or deadlocks. A special feature for train dispatching

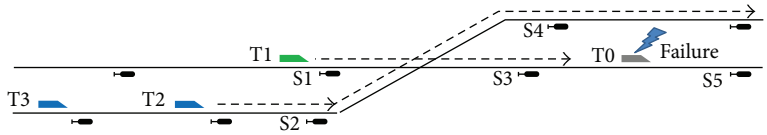

Figure 4: Example of giving up of occupancy [15].

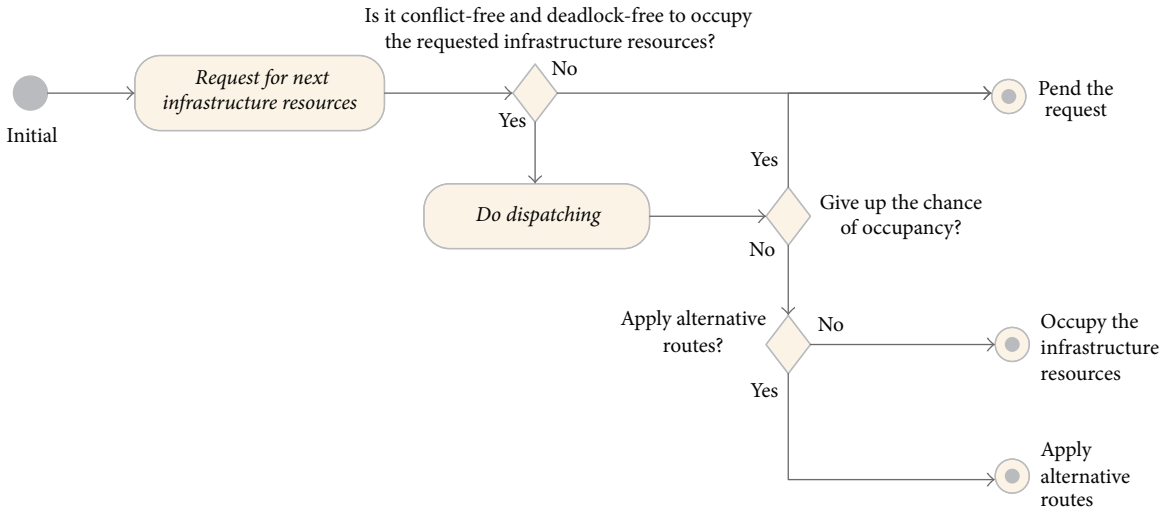

Figure 5: The dispatching mechanism in PULSim.

is designed when a request for infrastructure resources is conflict-free and deadlock-free. In this case, the train still has to evaluate the operational situation before it occupies the requested infrastructure resources. The decision of whether or not to give up the chance of occupancy depends on the applied dispatching algorithm, which is illustrated as the activity "do dispatching" (see Figure 5). On the basis of a FCFS strategy, the train will always take the chance to occupy the requested infrastructure resources. The FCFS strategy is implemented in PULSim by default, and if a dispatching strategy other than FCFS is applied, the train may give up the chance of occupancy according to certain dispatching objectives (e.g., to reduce total hindrances). The request will be then pended for the next round of dispatching. In addition, the dispatching mechanism also supports the implementation of applying alternative routes.

Although FCFS is currently the only strategy implemented in PULSim, users can customise and integrate their own dispatching algorithms into the dispatching process. This is achieved through the open interface and a standard workflow (see Section 4). Inside the workflow, users can define their own implementation to calculate the predicted effects for all possible dispatching actions (to occupy, to give up, or to choose an alternative route). The action with the best effects (e.g., with the minimal waiting time) will be chosen as the dispatching decision.

An example of using user-customised dispatching algorithms is published on GitHub (https://github.com/herrcui/RailView/blob/master/PULSimReleases/scripts/railapp.dispatching.services.ExternalDispatchingService.py). Within this example, the delays experienced by conflicting trains are retrieved from the simulation model by comparing the current departure/arrival times with the scheduled departure/arrival times. The example utilises a simple form of logic in order to illustrate the usage of the open interface. That is, the train experiencing the longest delay will receive the highest priority.

A Java interface is defined for users to provide customised train dispatching. Through the interface, a list of resulting states according to the generated possible actions will be given from PULSim as input. A state is a multidimensional list, which represents the resulting occupancy situation of an action for each train. The predicted occupancy situation includes the lists of the occupied infrastructure resources, as well as the start and end of the blocking time for each infrastructure resource. It should be noted that a state is not conflict-free. Users can implement their own algorithm to identify and to resolve conflicts. As a basis, all trains predict their further occupancy situation as if other trains do not exist. The predicted occupancy situation will then be shifted or changed according to a certain action. For the action to occupy the requested infrastructure resources by a train T, the blocking time of the infrastructure resources for other trains should be shifted until the infrastructure resources are released the train T. Similarly, if a train T gives up the chance of occupancy to wait for another train T', the blocking time of other related trains should also be shifted, until the infrastructure resources are released by T'. Hence, the current operational situation and the action to be taken are represented in the form of the resulting states.

Users can implement the method "determineAction" to determine a dispatching action. The method returns an

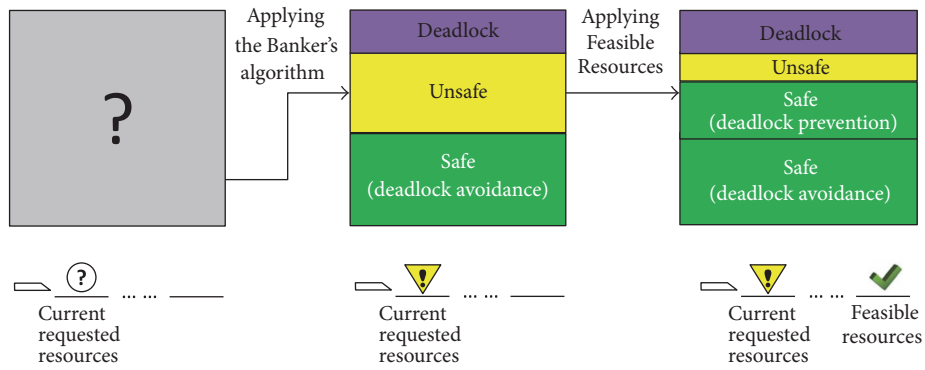

Figure 6: Searching feasible resources to reduce false-positive situations with the Banker's algorithm [15].

index of the given actions, which indicates the chosen action according to a specific dispatching objective (e.g., with minimal waiting time). At the moment, an algorithm based on reinforcement learning for train dispatching is being developed. A preliminary investigation to evaluate the effects for a certain state and a given action is described in Section 5.

3.3. Deadlock Avoidance Based on Banker's Algorithm. Deadlock problems [22] are well-known in the area of synchronous simulation. Some approaches of deadlock avoidance are proposed in [23–26]. PULSim applies a method based on the Banker's algorithm [27], which was initially used for the purposes of railway simulation in [11]. Once a train requests to occupy one or several infrastructure resources, a deadlock-free test will be carried out based on the Banker's algorithm. The workflow of this deadlock-free test is straightforward; if all the required infrastructure resources along the path of a train are available (in other words, not occupied by other trains), the train will be identified as a passed train. All the resources occupied by the passed train will be then returned to the system. The deadlock-free test will be executed iteratively for every train, until all trains have been identified as passed trains. In this case, a deadlock-free situation can be always guaranteed, since the feasibility of resource allocation has been proven.

If there are some trains remaining, whose required infrastructure resources are not available, the situation will be regarded as being in an unsafe state. It should be noted that an unsafe state does not necessarily result in a deadlock situation. A false-positive situation is defined in [11] as a deadlock-free test result that is read as positive but is actually negative. False-positive situations should be reduced to avoid unnecessary waiting time due to the overly strict rules in deadlock-free tests.

A method to search for feasible resources for reducing false-positive situations with the Banker's algorithm is implemented in PULSim (Figure 6). Feasible resources are defined as one or more connected and not yet granted infrastructure resources in the further path of a train. If the train occupies the feasible resources, a circular wait situation should not take place. The circular wait situation is one of the necessary conditions for deadlocks [25]. Once the circular wait situation is eliminated, the feasible resources can be granted to the train without deadlocks, even if the train has not passed the deadlock-free test through the Banker's algorithm. In [15], the approaches to identify feasible tracks as well as the effects of reducing false-positive situations are presented in detail.

In this method, a request for infrastructure resources will be at first analysed in accordance with the Banker's algorithm. Upon passing this test, the requested infrastructure resources can be safely granted to the requesting train. In case a request fails to pass the test with the Banker's algorithm, the method of searching for feasible resources will commence. If feasible resources are indeed available, the train's request may still be approved.

The aforementioned combination of the Banker's algorithm and the method of searching for feasible resources can ensure a high applicability and efficiency for deadlock avoidance. This method has been proven in PULSIM for several practical applications and case studies. For example, a real railway network in Germany is used for the case studies, consisting of 129 stations, including a large terminus station, and 2,388 train runs. Even with randomly shuffled departure times of train runs, deadlocks can be successfully avoided. Another extreme example is also available on GitHub. For a highly congested network with a high potential of deadlocks, the software PULSim is still able to simulate all the train runs without deadlocks.

4. User-Based Adaptable Simulation

4.1. Open Interface of Railway Simulation Tools. A simulation tool serves as the basis for further investigations into railway planning and operations, in which the output of a simulation can be used as the input of other applications. Evaluation of the simulation output is the most popular use case of these applications. Therefore, the interaction between a simulation tool and other applications for evaluation of simulation output should be enabled.

The interactions for evaluation can be categorised as either offline or online evaluations. In the case of offline evaluation, the data generated by the simulation tool will be initially stored and will be analysed afterwards. This process is suitable for a middle-term or long-term evaluation with a very large amount of data. The saved log files can be reused in the future for other purposes of evaluation without requiring

an additional run of simulation. With an online evaluation, the evaluation between a simulation tool and third-party applications can be achieved efficiently through exchanging real-time data. A built-in package in the simulation tool or an open interface supporting direct data access is required for online evaluation. Sometimes, the output of the third-party applications will be fed back to the simulation tool. In order to exchange information for further evaluation and analysis, open interfaces between a simulation tool and other third-party applications are provided in PULSim.

When conducting offline evaluation, it is very common to define the format and the schematic structure of the saved log files as an open interface. Popular formats of the log files include Extensible Markup Language (XML) files, Comma Separated Values (CSV), files or plain text files. As an open interface, the format and the schematic definition of the output files should be provided and published in advance. At the time that a simulation tool is developed, its possible usage cases may still be unknown. Therefore, it is a challenge to ensure that the tool provides open interfaces for further possible applications at an early stage. Currently, an open and standardised interface for the output of railway simulation is not available, with each simulation tool possessing its own definition for the output of simulation. Attempts to unify various outputs generated from different simulation tools require considerable effort. Certain simulation tools output log files without a published interface, which can be used in internal testing for various special purposes. It is not recommended to use these data due to the lack of documentation and support, although the information may be inferred from the contents of the files.

For online evaluation, the output of simulation can be retrieved and analysed directly through an open interface. The process of railway planning and operations can be evaluated and optimised by tuning the parameters of the simulation tools in real time. The open interface can be provided either from an open Application Programming Interface (API) or in the form of a scripting language.

An open API enables a dynamic data exchange via directly accessing the simulation tool from other third-party applications. In the simulation tool OpenTrack, the open API is provided in the form of web services [28], from which a third-party application can send messages as commands to OpenTrack and retrieve the output as status messages back to the application. The commands and the status messages are transferred in a machine readable format, according to the Simple Object Access Protocol (SOAP) over Hypertext Transfer Protocol (HTTP). An example use case of using the OpenTrack API for train dispatching is presented in [28]. New dispatching algorithms and prototypes are tested in a simulation environment, in which the realistic situation is replaced by the simulation tool OpenTrack. The commands and status messages in the same format used in reality are exchanged between the dispatching software and OpenTrack. Hence, the newly developed dispatching algorithms can be evaluated in an inexpensive experimental environment.

The system performance and efforts required for the development and maintenance of an open API solution should be considered. The amount of data in railway planning and operations is usually very large. Taking the example of using a web service with SOAP, the enveloped and transferred messages via XML data will cause a high overhead for communication, which will impact the performance of the system. Large software vendors and tools are often required for the implementation and development of web service solutions, which makes such a situation infeasible for researchers and institutes who are interested in rapid development with lightweight solutions. In addition, the complexity of learning the web service-based system and the induced efforts for debugging, testing, and maintaining the entire system are considerable.

Another option for an open interface is to use scripting language, which can combine the advantages of both offline evaluation and online evaluation. A scripting language is a programming language in the form of a series of imperative commands, which are executed in a certain run-time environment. These commands are interpreted during the run-time without needing to be compiled. Hence, sophisticated processes can be customised by users in an adaptable way. With scripting languages, the output of simulation can be either exported as log files for offline evaluation, or be accessed online via the provided interfaces (commands). Upon the demands and the required system performance, a user can flexibly decide the mode of interaction. A scripting language hides the internal structure and the implementation details of a complex system, which enables the user to concentrate on the core functions and the integration of the functions. The complexity of learning the system and the efforts for development, deployment, and maintenance are therefore significantly reduced.

Using open interfaces with scripting language can provide additional advantages in balancing the requirements of transparency and the complexity in development. It is important for users to understand the mechanism and the assumptions of a simulation tool. The open-source method would be an option to promote transparency and open collaboration. However, an excessive amount of implementation details of the complete source code will increase the complexity in comprehension. It should be noted that most users of railway simulation tools are interested principally in the domain logics related to railway planning and operations. Therefore, open interfaces will provide a moderate level of details for researchers to enable rapid development and integration of their own algorithm. As a precondition, the mechanism of the simulation tool should be published, so that users can customise their own logic based on the utilised mechanism. Hence, the internal assumptions and mechanism can be naturally understood by users according to the well-specified open interfaces. For example, the internal mechanism of dispatching systems should be specified in advance (see Section 3.2). As long as a customised dispatching algorithm is implemented, the mechanism of train dispatching is understood by users as well. In the future, the mechanism of running time calculations and signalling systems will be published in the form of open interfaces to users as well.

Other simulation tools also provide some open access to their internals. For example, the parameters used in RailSys and OpenTrack can be viewed and set through system

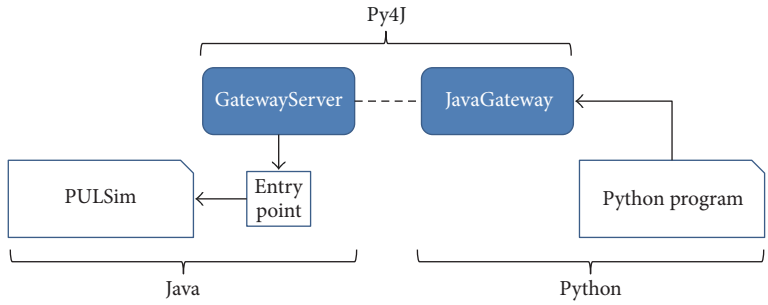

Figure 7: Py2J: accessing Java objects and services from Python.

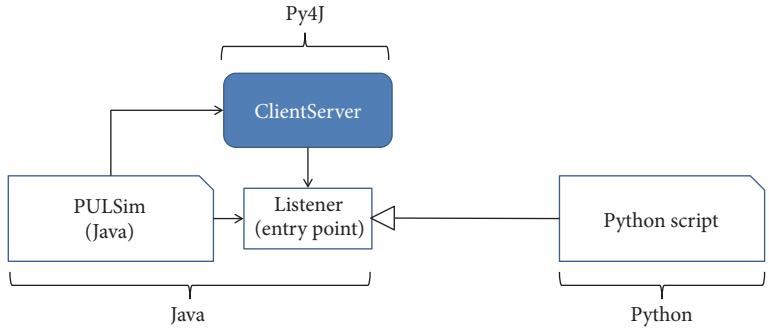

Figure 8: J2Py: accessing Python script from Java.

configuration. Users can also interact with OpenTrack through web services. PULSim provides additional support for customised simulation workflow and integration of their own algorithm. In Sections 4.2 and 5, the implementation of an open interface with scripting language and the applications of using the open interface provided by PULSim are presented.

4.2. The PULSim Implementation. In PULSim, a user-based adaptable interface for the simulation, evaluation, and optimisation for different applications is built using scripting language. Scripting languages can be categorised as either domain-specific languages or general-purpose languages. A domain-specific language is specifically designed for a certain application and platform, and an external parser is needed to interpret its commands. Additional efforts are required for users to learn the language. With general-purpose languages, and plenty of support, libraries and documentation are available, which usually allows users to obtain sufficient knowledge of general-purpose languages with lower learning efforts. In addition, most general-purpose languages are easily extended for certain special purposes.

Popular general-purpose scripting languages include Python, Perl, and Ruby. Among others, Python has been widely used for rapid and efficient development with plenty of extensions. Particularly, most implementations of machine learning in recent years have been written in Python. Therefore, Python is utilised as the scripting language in PULSim, and the open-source framework Py4J [29] is integrated within the software. Py4J enables Java programs to access Python code and also provides interfaces for Python programs to access Java objects. It works in a client-server mode for both directions.

In order to access Java objects from a Python program, the class GatewayServer should be at first initiated and started in Java code, taking an entry class as an entry point. The Python program will call the methods provided by the entry point through JavaGateway. Thereby, the Python program can access Java objects and services provided by PULSim. The mechanism of accessing Java objects and services from Python (Py2J) is shown in Figure 7.

To enable the Java code running in PULSim to access a Python script, a ClientServer class is initialised to start the Python script, which implements a listener interface defined in PULSim. Hence, the listener will be used as an entry point for Java code to access the functions provided by Python. The mechanism of accessing Python script from Java (J2Py) is shown in Figure 8.

The examples and the applications of using this bidirectional communication for user-based, adaptable simulation are described in Section 5.

5. Applications of User-Based Adaptable Simulation

A very common requirement for user-based, adaptable simulation is the ability to organise and customise several rounds of simulation as well as the desired output of simulation. For this purpose, PULSim provides an open interface for timetable simulation for Python. A simulator for timetable simulation can be obtained through the interface, in which users can designate the investigated data of infrastructure,

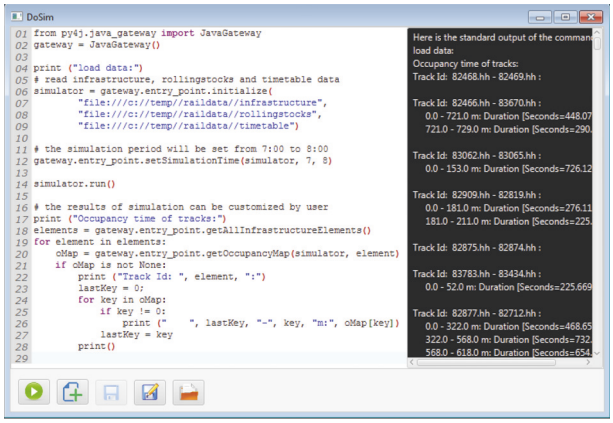

Figure 9: User interface for accessing Java services from Python in PULSim (Py2J).

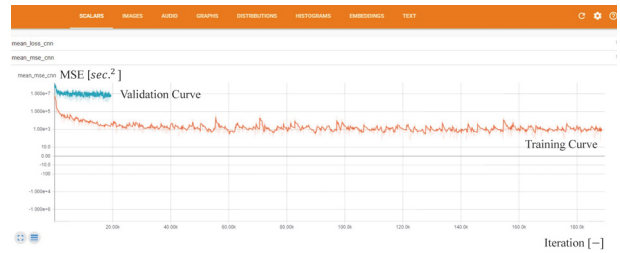

Figure 10: Prediction of total waiting time with TensorFlow (J2Py).

rolling stocks, and the timetable in the Python environment. Other configurations, for example, the time period of the simulation, can also be optionally specified for the simulator. With a few lines of code, a timetable simulation can be carried out in the Py2J mode (Figure 7).

The results of the simulation can be further analysed. For example, the occupancy and hindrance values during the simulation are matters of interest for purposes of capacity research. In Figure 9, the user interface for a timetable simulation with the outputs of occupancy and hindrance is shown. The editor for the Python program, the Py4J libraries, and the output window of the running Python program are integrated into PULSim as well.

In addition to being saved in log files for an offline evaluation, the output can also be directly retrieved by the Python program for further calculation and online evaluation. This is especially useful in order to customise a complicated workflow with many rounds of simulation. For example, to derive the recommended area of traffic flow within the scope of capacity research (source [10]) automatically, it is necessary to seamlessly integrate the software for capacity research with the simulation tools. Otherwise users would have to shift between different software tools manually to simulate several densified timetable variants. The advantage of the user-based, adaptable simulation can also be gained in the process of calibration of the parameters for operational simulation. In [16], an automatic process for calibration is implemented by the authors using PULSim. It is possible in this case, since the developers for the calibration system are also the developers of the simulation tool. However, if developers from a third party want to implement the calibration process, it is difficult if they do not have access and understanding of the source code of the simulation tools. A Py2J interface can provide the developer with the opportunity to adapt the disturbance parameters through many rounds of operational simulation iteratively, without the additional effort of interacting with the simulation tool.

Another advantage of user-based, adaptable simulation is the enhancement of the capabilities of the simulation tool through third-party software and algorithms. Today, machine learning plays an increasingly important role in learning and making predictions about data. Nowadays, many software packages for machine learning are implemented in Python, for example, Scikit-Learn [30], Theano [31], and Google TensorFlow [32]. PULSim is able to access these available Python-based software tools and to perform a learning process through J2Py communication (Figure 8). The desired functions for machine learning can be initially defined in a listener in PULSim. A Python script will implement the listener interface in order to carry out the learning process. PULSim will provide the training data for the Python-based software tools, which enables the learning process to be carried out either offline or online.

The authors are carrying out a case study, which integrates Google TensorFlow with PULSim to predict the total waiting time of a dispatching action. The case study is taken from a real railway network in Germany with 71 stations. There are in total 1350 trains running in the entire investigated day. The simulated timetable is not conflict-free; a dispatching action has to be taken for a certain state with occupancy conflicts. For a given state, there are several possible dispatching actions to be taken. The task of the dispatching system is to decide on a certain action as the dispatching decision, which results in the minimum possible waiting time. Therefore, the resulting total waiting time for a given state and a certain action has to be predicted exactly. This can be achieved through supervised learning. The object of learning is to minimise the Mean Squared Error (MSE) between the real waiting time and the predicted waiting time. During the simulation process, the states and the actions to be taken are recorded in the form of resulting states by PULSim as input (see Section 3.2), and the resulting total waiting time are learned as output. The total waiting time is learned and predicted by the Python script within the framework of TensorFlow.

With the case study, 12,000 resulting states and the total waiting time are fed into a Convolutional Neural Network (CNN) as training data, and 1,000 resulting states and the total waiting time are used as validation data. The CNN is at first trained with the training data and then validated with the validation data. The average value of the real waiting time for the training and validation data is 44.63 minutes. In Figure 10, the results are shown in Google TensorBoard, which is a tool provided for the visualisation of the learning process. The training curve represents the change of MSE for the training data along iterations, and the validation curve represents the MSE for the validation data along

iterations. After ten iterations of training on batch data, an epoch of validation is carried out. Hence, the iterations for training are ten times the iterations of validation. At the moment, the mean error of the predicted total waiting time from the training data is around 1 to 2 minutes, and the mean error of the predicted total waiting time from the validation data is around 8 minutes. The accuracy should be further improved through tuning the hyperparameter of the model.

6. Conclusion and Perspectives

In this paper, a user-based, adaptable simulation tool PULSim for railway planning and operations is presented. Supported by its open interface, users and researchers can rapidly develop their own algorithms to be integrated with PULSim. The productivity of using simulation tools for evaluation and optimisation of railway planning and operations will be improved significantly.

More interfaces will be provided in the further developments of PULSim. Special focus will be placed on the integration of varied signalling systems, dispatching algorithms, and the flexible configuration of operational simulation. The further developed interfaces will be built according to the demands and the feedback from users of the software. For example, in order to find an optimised setting of block sections, an interface to allow flexible placement of signals can be provided. The position of signals can be adjusted dynamically, so that the effects for different variants can be evaluated and optimised by the user-defined optimisation algorithm. In further development, additional dispatching actions, including the extension or shortening of train paths, along with the reordering or cancellation of train runs, should be integrated into PULSim for both railway simulation and real operations control.

Having the large amount of data generated by PULSim, various machine learning algorithms can be further applied to reveal and predict the system behaviour of complex railway systems. PULSim enables a simple integration of the simulation platform and popular machine learning software. Furthermore, it is a growing trend to carry out machine learning combined with the technology of big data. Since PULSim is developed in Java, it can be naturally integrated with Apache Hadoop [33], which is a Java-based software framework for managing and processing large amounts of data. The efficiency and the effectiveness of railway planning and operations will be continuously improved through integrating cutting-edge technologies with PULSim.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

During the work, Mr. Kai Chen made his contribution to GUI programming. The text and the English have been proofread and improved by Mr. Syed Murtaza Hasan.

References

[1] RMCON, "RailSys 7 User Manual. Preparations for Multiple Simulation," 2010.

[2] D. Hürlimann, *Open Track Betriebssimulation von Eisenbahnnetzen*, Zürich, Switzerland, 1.6 edition, 2010.

[3] VIA Con, LUKS Handbuch Version 2.1, 2011.

[4] B. Kogel, N. Nießen, and T. Büker, *Influence of the European Train Control Sys-tem (ETCS) on the Capacity of Nodes*, International Union of Railways, Paris, France, 2010.

[5] M. Luethi, D. Huerlimann, and A. Nash, "Understanding the timetable planning process as a closed control loop," in *Proceedings of the 1st International Seminar on Railway Operations Modelling and Analysis*, I. A. Hansen, F. M. Dekking, R. M. P. Goverde, B. Heidergott, and L. E. Meester, Eds., Delft, Netherlands, 2005.

[6] A. Radtke, *EDV-Verfahren zur Modellierung des Eisenbahnbetriebs*, vol. 64, Eurailpress Tetzlaff-Hestra (Wissenschaftliche Arbeiten für den Schienenver-kehr, 64), Hamburg, Germany, 2005.

[7] A. Radtke and J. Bendfeldt, "Handling of railway operation problems with RailSys," in *Proceedings of the 5th World Congress on Rail Research*, Cologne, Germany, 2001.

[8] F. Corman, A. D'Ariano, and I. A. Hansen, "Evaluating disturbance robustness of railway schedules," *Journal of Intelligent Transportation Systems: Technology, Planning, and Operations*, vol. 18, no. 1, pp. 106–120, 2014.

[9] U. Martin, Y. Cui, F. Hantsch, Z. Chu, and X. Li, "Knotenkapazität - Bewertungs-verfahren für das Mikroskopische Leistungsverhalten und die Engpasserkennung im Spurgeführten Verkehr (RePlan)," in *VWI Neues verkehrswissenschaftliches Journal – Band 8*, Books on Demand GmbH Norderstedt, Norderstedt, Deutschland, 2014.

[10] U. Martin, C. Schmidt, and Z. Chu, "PULEIV Anwendungsleitfaden zur PULEIV-Version 2.1," *Anleitung zur Ermittlung des Leistungsverhaltens von Eisenbahninfra-struktur mithilfe von Simulationsprogrammen*, 2011.

[11] Y. Cui, "Simulation-Based Hybrid Model for a Partially-Automatic Dispatching of Railway Operation," *Norderstedt: Books on Demand GmbH (Neues verkehrswissenschaftliches Journal, 4)*, 2010.

[12] A. D'Ariano, M. Pranzo, and I. A. Hansen, "Conflict resolution and train speed coordination for solving real-time timetable perturbations," *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, no. 2, pp. 208–222, 2007.

[13] I. A. Hansen and J. Pachl, *Railway Timetabling & Operations: Analysis - Modelling - Optimisation - Simulation - Performance Evaluation*, DW Media Group, Hamburg, Germany, 2014.

[14] J. Liang, *Metaheuristic-based Dispatching Optimization Integrated in Multi-scale Simulation Model of Railway Operation*, Institut für Eisenbahn- und Verkehrswesen der Universität Stuttgart, Stuttgart, Germany, 2017.

[15] Y. Cui, U. Martin, and J. Liang, "Searching feasible resources to reduce false-positive situations for resolving deadlocks with the Banker's algorithm in railway simulation," *Journal of Rail Transport Planning and Management*, vol. 7, no. 1-2, pp. 50–61, 2017.

[16] Y. Cui, U. Martin, and W. Zhao, "Calibration of disturbance parameters in railway operational simulation based on reinforcement learning," *Journal of Rail Transport Planning & Management*, vol. 6, no. 1, pp. 1–12, 2016.

[17] Y. Cui and U. Martin, "Multi-scale Simulation in Railway Planning and Operation," *PROMET - Traffic & Transportation*, vol. 23, no. 6, 2011.

[18] M. Wörner and Y. Cui, *Begriffslexikon - Datenmodell*, Institut für Eisenbahn- und Verkehrswesen der Universität Stuttgart, Stuttgart, Germany, 2008.

[19] U. Martin and J. Liang, "The Influence of Dispatching on the Relationship between Capacity and Operation Quality of Railway Systems (DFG Research Project MA 2326/15-1)," in *VWI Neues verkehrswissenschaftliches Journal – Band 20*, Books on Demand GmbH, Norderstedt, Deutschland, 2017.

[20] A. Nash, D. Huerlimann, J. Schütte, and V. P. Krauss, "RailML A Standard Data Interface for Railroad Applications," *WIT Transactions on The Built Environment*, vol. 74, 2004.

[21] W. Fang, S. Yang, and X. Yao, "A Survey on Problem Models and Solution Approaches to Rescheduling in Railway Networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 6, pp. 2997–3016, 2015.

[22] E. G. Coffman, M. Elphick, and A. Shoshani, "System Deadlocks," *ACM Computing Surveys*, vol. 3, no. 2, pp. 67–78, 1971.

[23] G. Mills and P. Pudney, "The Effects of Deadlock Avoidance on Rail Network Capacity and Performance," in *Proceedings of the 2003 Mathematics-in-Industry Study Group*, Brisbane, Australia, 2003.

[24] E. R. Petersen and A. J. Taylor, "Line block prevention in rail line dispatch and simulation models," *Information Systems and Operations Research*, vol. 21, no. 1, pp. 46–51, 1983.

[25] J. Pachl, "Deadlock Avoidance in Railroad Operations Simulations," in *Proceedings of the 90th Annual Meeting of Transportation Research Board*, Washington, DC, USA, 2011.

[26] R. Mittermayr, J. Blieberger, and A. Schöbel, "Kronecker algebra-based deadlock analysis for railway systems," *Promet - Traffic - Traffico*, vol. 24, no. 5, pp. 359–369, 2012.

[27] E. W. Dijkstra, "The mathematics behind the Banker's algorithm," in *Selected Writings on Computing: A Personal Perspective*, pp. 308–312, Springer-Verlag, New York, NY, USA, 1982.

[28] B. Seybold and D. Huerlimann, "OpenTrack - Simulation of railway systems. Presentation of the OpenTrack API. OpenTrack," http://www.opentrack.ch/opentrack/downloads/OpenTrack.API.pdf 2016.

[29] Py4J, "Py4J – A Bridge between Python and Java," https://www.py4j.org/, 2007.

[30] F. Pedregosa, G. Varoquaux, and A. Gramfort, "Scikit-learn: machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[31] J. Bergstra, O. Breuleux, P. Lamblin et al., "Theano: Deep learning on gpus with python," in *NIPS 2011, Big Learning Workshop*, vol. 3, Granada, Spain, 2011.

[32] M. Abadi, A. Agarwal, P. Barham et al., "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," https://arxiv.org/abs/1603.04467, 2016.

[33] J. Nandimath, E. Banerjee, A. Patil, P. Kakade, and S. Vaidya, "Big data analysis using Apache Hadoop," in *Proceedings of the 2013 IEEE 14th International Conference on Information Reuse and Integration, IEEE IRI 2013*, pp. 700–703, San Francisco, CA, USA, August 2013.