



RESEARCH-ARTICLE

# A Parallel World Framework for scenario analysis in knowledge graphs

Andreas Eibeck<sup>1</sup>, Arkadiusz Chadzynski<sup>1</sup>, Mei Qi Lim<sup>1</sup>, Kevin Aditya<sup>2</sup>, Laura Ong<sup>1</sup>, Aravind Devanand<sup>3</sup>, Gourab Karmakar<sup>1</sup>, Sebastian Mosbach<sup>4</sup> , Raymond Lau<sup>2</sup>, Iftekhar A. Karimi<sup>3</sup>, Eddy Y. S. Foo<sup>1,5</sup> and Markus Kraft<sup>1,2,4</sup> 

<sup>1</sup>Cambridge Centre for Advanced Research and Education in Singapore (CARES), Singapore, Singapore

<sup>2</sup>School of Chemical and Biomedical Engineering, Nanyang Technological University, Singapore, Singapore

<sup>3</sup>Department of Chemical and Biomolecular Engineering, National University of Singapore, Singapore, Singapore

<sup>4</sup>Department of Chemical Engineering and Biotechnology, University of Cambridge, Cambridge, United Kingdom

<sup>5</sup>School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, Singapore

\*Corresponding author. E-mail: [mk306@cam.ac.uk](mailto:mk306@cam.ac.uk)

(Received 02 March 2020; revised 03 June 2020; accepted 08 June 2020)

**Keywords:** Interoperability; knowledge graph; parallel world; scenario analysis; superstructure versioning

## Abstract

This paper presents Parallel World Framework as a solution for simulations of complex systems within a time-varying knowledge graph and its application to the electric grid of Jurong Island in Singapore. The underlying modeling system is based on the Semantic Web Stack. Its linked data layer is described by means of ontologies, which span multiple domains. The framework is designed to allow what-if scenarios to be simulated generically, even for complex, inter-linked, cross-domain applications, as well as conducting multi-scale optimizations of complex superstructures within the system. Parallel world containers, introduced by the framework, ensure data separation and versioning of structures crossing various domain boundaries. Separation of operations, belonging to a particular version of the world, is taken care of by a scenario agent. It encapsulates functionality of operations on data and acts as a parallel world proxy to all of the other agents operating on the knowledge graph. Electric network optimization for carbon tax is demonstrated as a use case. The framework allows to model and evaluate electrical networks corresponding to set carbon tax values by retrofitting different types of power generators and optimizing the grid accordingly. The use case shows the possibility of using this solution as a tool for CO<sub>2</sub> reduction modeling and planning at scale due to its distributed architecture.

## Impact Statement

The methodology developed in this paper allows simulation of complex systems that consist of many inter-dependent parts, such as an industrial park, as well as variations thereof, referred to as parallel worlds. In addition to the ability to consider different scenarios, a key distinguishing feature of our approach, which is based on a generic all-purpose design that enables interoperability between heterogeneous software and, as a consequence, cross-domain applications, is its employment of knowledge graphs and autonomous software agents. As such, the methodology presented here allows city planners and policy makers to ask what-if questions or explore alternatives—a process that can play an important role in decision-making. As an example, optimizing the electrical grid of Jurong Island in Singapore is considered, for two different levels of carbon tax, thus demonstrating how the methodology can assist planning for carbon footprint reduction.

## 1. Introduction

The fact that simulation plays an important role in the process industry acted as a motivator for design and development of the Parallel World Framework presented in this paper. The timescale of simulation can vary: from near real-time optimization (for automated decision-making) to long-term policy tests (strategic, support of human decision-making, etc.). Scenario analysis is a technique used to analyze potential events by examining possible outcomes under given starting conditions. The outcomes are sometimes called “alternative worlds.” Scenarios can be characterized as processes of varying some initial settings (configuration, conditions, parameters, and representations forming parts of the knowledge graph) and then assessing results of simulation and optimization. The framework was developed within the J-Park Simulator (JPS), an ontology-based system for cross-domain scenarios for the process industry (Kleinelanghorst et al., 2017), which is part of [theworldavatar.com](http://theworldavatar.com) effort which is a general dynamic knowledge graph not restricted to process industry. The main ingredients are a knowledge graph built on semantic technologies and intelligent agents performing operations on it.

The main challenges addressed by the framework are as follows:

- A need to allow for parallel existence of relevant entities within the knowledge graph forming a semantic representation of a “parallel world.” At the same time, not allowing for interference with other parallel worlds and agent’s activities.
- Management of initial settings, intermediate changes, and results as well as storing, visibility, finding, and reusing (e.g., in more complex scenarios or archiving for digital twins life-cycle).

There are several aspects, which are strongly related to information persisting, annotating, and versioning. First of all comes persistence. This layer is concerned with reading, querying, and writing new or updating existing resources. They come in different shapes of formats as well as storage implementations. Second is an annotation layer. Annotations provide additional information about resources. An example is provenance data, which consist of information about when was a resource created, which agent created or updated a resource, and so on. It can also be used to find resources or to distinguish between different versions of them. The third aspect under consideration in this paper is versioning. By setting up an initial configuration or as result of simulation and optimization, some parts of the knowledge graph change. To prevent overwriting the original state by a new state, there is a need to keep track of different versions for these parts.

One can also find a multitude of versioning strategies proposed in the literature. Many of them are built on top of existing solutions for persistence and annotation, and are tailored to their specific area of application. There are a number of criteria used to characterize existing versioning strategies, which will be referred to in later sections of the paper:

- Granularity level: triple, RDF document (set of triples).
- Reification strategy: for example, standard reification and named graph.
- Storage strategy: independent copies, change-based, and timestamp-based.
- Change tracking: support for branching and merging or only linear change tracking.

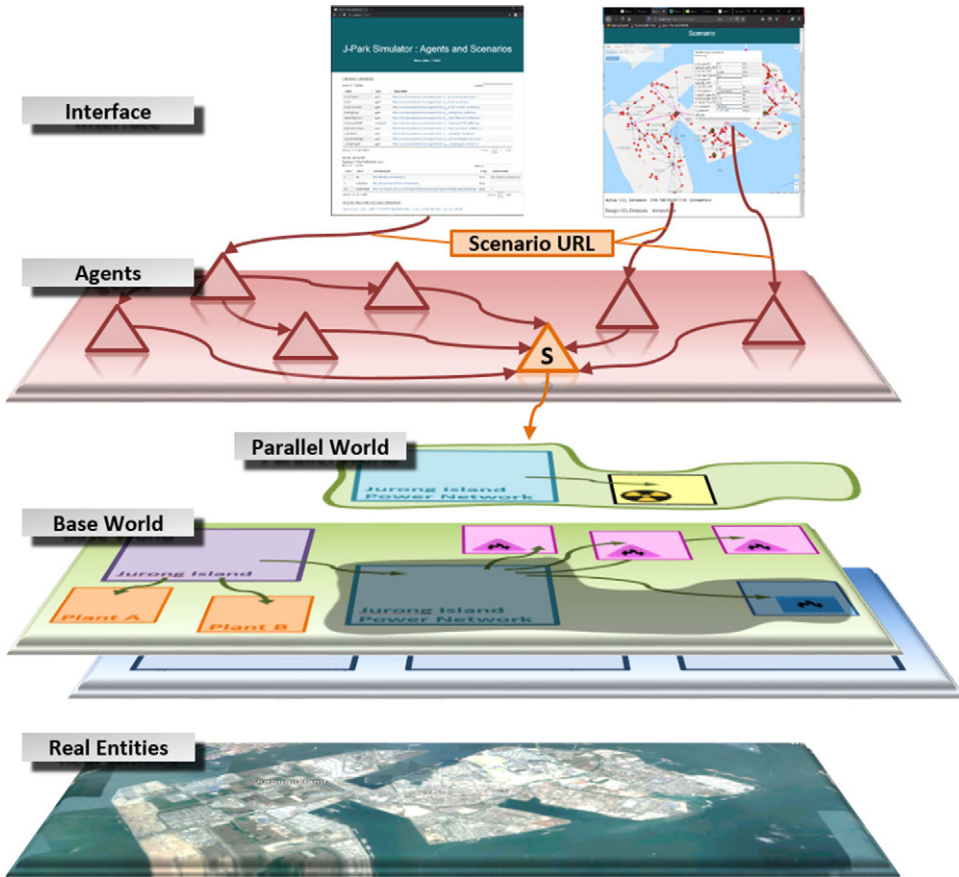
Existing solutions are not applicable to the distributed nature of both knowledge graphs and agents. We discuss this here along two versioning tools, Git and the Memento Framework. As pointed out by Arndt et al. (2019), distributed version control systems, and Git in particular, could be employed while designing change tracking and versioning solutions for linked data. Operations implemented in Git are transformable to corresponding operations on RDF datasets. It is possible to build systems that support collaboration on distributed resources and heterogeneous data repositories (Arndt et al., 2016) on top of it. However, in case of JPS, distributed nature of Git would make it too complex to make sure that changes in the knowledge graph are visible to all participating agents immediately and without additional push and pull operations. Although agents could still make their changes and read them back on a

dedicated URL service via REST API, without extending bare Git, the volume of provenance data is likely to grow beyond the size of the described data (Arndt et al., 2017). Undoubtedly, among its advantages are structural merging capabilities, with which archiving and adding new scenarios on top of the other ones could be easily created. Concerning JPS cross-domain linking requirements and operating under the open world assumption, solutions based on Git have some disadvantages. One can only pull what is in the repository but cannot put the entire world into this repository. The importance of such multi-scale modeling is more extensively elaborated on by Kraft and Mosbach (2010).

A slightly different approach is taken by the Memento Framework (Van De Sompel et al., 2009). Instead of minting a new URL for every new version, URL is kept stable and new URLs are minted only for old versions. This approach has become rather widespread. The main idea behind it is to balance quality and scalability by creating traces of interactions with web pages. The created trace also indicates the URL pattern to which the trace applies to as well as provenance information, including the resource on which the trace was created (Klein et al., 2019). JPS is designed as a system based on the principle of autonomous intelligent agents operating on a knowledge graph. Although it addresses issues of temporal coherence (Ainsworth et al., 2014), Memento Framework is not capable of tracing interactions of autonomous agents at the moment.

The purpose of this paper is to present how reduction of CO<sub>2</sub> emissions is possible by the means of the Parallel World Framework, implemented within the JPS. It was envisioned in mind with addressing problems related to versioning and archiving of resources, which belong to representations of complex cross-domain superstructures. Capabilities of the framework enable generic simulation of what-if scenarios, even for complex, inter-connected, cross-domain applications. More specifically, it allows simulating different versions of the world's emission sources and picks the one with the optimal configuration with regard to CO<sub>2</sub> emissions. Other ideas, such as multidimensional and temporal versioning at the RDF level plus filtering conditions in SPARQL queries (Tappolet and Bernstein, 2009), are designed to address only relatively narrow problem spaces. One of the advantages of the solution presented in this paper being implemented within the JPS is that it is suitable for a distributed environment. Data are stored on different hosts, while incoming and outgoing links are preserved. A disadvantage of the solution is that it relies on named graphs, which contain only small numbers of triples. Due to the versioning level being very coarse, operations on named graphs can be very time and memory consuming. Moreover, some additional calls to resolve URLs are needed. This approach was chosen in order to pave the way for relatively easy parallel worlds annotations. Annotation is a prerequisite to track modifications (i.e., versioning with identifiers) as well as to provide metadata such as time stamp, creators, and scenario. It is possible to annotate RDF resources at different levels. Parallel worlds originate from the base world and can be regarded as different versions of it. Therefore, within JPS, versioning occurs at the level of named graphs.

JPS and the Parallel World Framework rely heavily on the Semantic Web Stack. Its main architectural building blocks are detailed by Eibeck et al. (2019). In this paper, Section 2 provides an overview of the JPS system as well as its knowledge graph technology. It includes information about relations between high-level concepts, such as ontologies and their description language as well as their realizations in the form of named graphs corresponding to real-world entities and their representations in the form of Subject-Predicate-Object triples. The use of ontologies and related semantic technologies in JPS for multilevel and cross-domain modeling as well as decentralized management of data and knowledge is presented by Zhou et al. (2017). This paper introduces more specific technological choices made within the system to handle data serialization, storage, and communication protocols. The Parallel World Framework is introduced in Section 3, which describes information retrieval and manipulation mechanisms for individual scenarios, identified by unique URLs. It is shown how those operations are realized within the JPS with intelligent autonomous agent-driven system architecture on a couple of intuitive examples. The section also describes how those mechanisms allow to organize such information into parallel world containers. Section 4 shows an application of the knowledge graph-based Parallel World Framework to the scenario analysis-based simulations of the power network, modeled within the JPS system. Results of such an analysis are applied to the power grid optimization for carbon tax by retrofitting



**Figure 1.** Breakdown of the World representation layers in the J-Park Simulator system (map data ©Google).

nuclear energy generators into the simulated power grid. It also includes detailed descriptions of interactions between all of the previously introduced components and their concrete implementations.

## 2. JPS and Knowledge Graph

The JPS uses the Web Ontology Language (OWL) to model a variety of real-world entities on Jurong Island, such as chemical plants, as well as Jurong Island in its entirety as an eco-industrial park (EIP; Pan et al., 2015). While OWL is a very rich and expressive language, we restrict ourselves here to only those aspects that are relevant to describing how the Parallel World Framework takes advantage of semantic technologies. It allows to define classes (types, concepts), individuals (instances of a given class), and properties (relations) and uses URLs as globally unique identifiers for them. It can be illustrated by means of Subject-Predicate-Object triples. For instance, the triple consisting of the following absolute URLs defines a class for power generators:

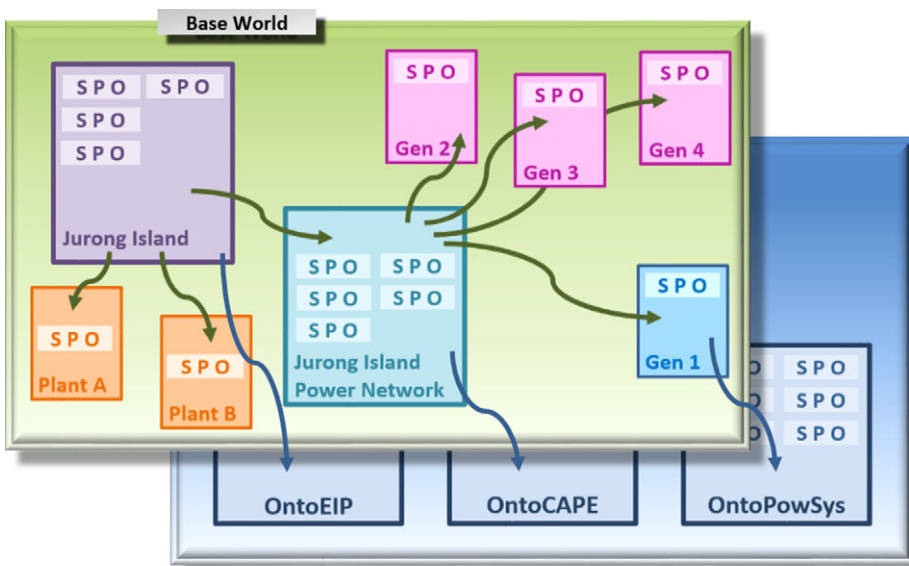
- <http://www.theworldavatar.com/ontology/ontopowsys/PowSysRealization.owl#PowerGenerator>
- <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
- <http://www.w3.org/2002/07/owl#Class>

This triple may as well be expressed in a more compact way with the help of namespaces. However, to keep it more understandable, short and human-readable triple notation is frequently used in this paper. For

instance, the triples “System is a Class” and “PowerGenerator is a Class” define two classes. Triples “JurongIslandPowerNetwork is a System” and “Gen1 is a PowerGenerator” define two corresponding individuals. The triple “hasSubsystem is an ObjectProperty” defines a property and the triple “JurongIslandPowerNetwork hasSubsystem Gen1” links the previously defined instances.

Entities described this way form the JPS knowledge graph. Each triple may be represented by two nodes labelled with the URLs of the subject and the object, respectively, and a directed edge in between, labelled with the URL for the predicate. Consequently, a set of triples can be represented by a directed graph, which does not necessarily have to be completely connected. A knowledge graph usually denotes a large graph (i.e., a large set of triples) where the number of instances is much larger than the number of classes (Paulheim, 2016). This is the case of the JPS knowledge graph. In principle, the JPS knowledge graph is not restricted to the data generated for JPS but may also integrate triples from other sources. In the same way, other applications may make use of the JPS knowledge graph. A plain, large set of triples as a graph without any clear substructures can be hard to navigate. The question of how to organize these triples and subdivide them into suitable subsets is of high importance for maintainability, performance, distribution, and discovery and also for the Parallel World Framework.

Figure 2 shows a few entities from the power network scenario in a very simplified manner. The box with the label “Gen1” contains only a single triple “Gen1 is a PowerGenerator.” Meanwhile, a power generator in JPS is modeled with the help of over one hundred triples that, among other things, specify appropriate electrical parameters and the bus node it is connected to. Different colors, in Figure 2, illustrate distinguished classes of individuals: Jurong Island EIP, power plants A and B, power network, and two different types of power generators. A triple set may be serialized in RDF/XML syntax<sup>1</sup> and stored as a file. The triple set of the file content may be requested by a specific URL.<sup>2</sup> At the same time, it is



**Figure 2.** Representation of the Base World fragment by the means of semantic technologies: concrete realizations of domain ontologies as named graphs correspond to real entities and are described by the Subject-Predicate-Object triples.

<sup>1</sup> <https://www.w3.org/TR/rdf-syntax-grammar/>

<sup>2</sup> Requesting the resource <http://www.jparksimulator.com/kb/sgp/jurongisland/jurongislandpowernetwork/EGen-001.owl> will return the triple set for “Gen1” in RDF/XML syntax. However, the URL of individual Gen1 itself is given by adding #EGen-001 to the previous URL. These URLs with fragments are used frequently.

important to distinguish the URL that identifies the physical entity as an individual in the sense of OWL from this specific URL that identifies the triple set describing the model associated with that entity.

Alternatively, the triple set may be stored as a named graph forming a part of a dataset in a triple store. Roughly speaking, a named graph denotes a pair consisting of an URL and a triple set.<sup>3</sup> Both ways are similar with respect to packaging triple sets, and indeed, the implementation of the Parallel World Framework is able to switch between them easily. For this reason, we apply the term “named graph” to both alternatives going forward. However, when querying a huge triple set or a large collection of named graphs, a triple store would provide a much better performance.

JPS separates the class level and the instance level in accordance with the best practices. Classes and properties are defined in modular, reusable domain ontologies. [Figure 2](#) shows three domain ontologies as boxes in the lower blue layer. Together with other domain ontologies, they have been used in previous scenarios in JPS and are also relevant to the power network scenario:

- [OntoCAPE \(Marquardt et al., 2009\)](#): a domain ontology for chemical process engineering based on more general upper ontology for engineering.
- [OntoEIP \(Zhou et al., 2019\)](#): ontological description for EIPs.
- [OntoPowerSys \(Devanand et al., 2020\)](#): a domain ontology for power systems.

In fact, the blue box representation can be still regarded as a dramatic oversimplification: for example, [OntoCAPE](#) contains over 50 submodels, [OntoPowerSys](#) around 6, each of which can be represented as a named graph on its own. Therefore, each of the blue boxes could be seen as representing rather a dataset consisting of named graphs. An arrow pointing from one box to another box in [Figure 2](#) reflects the fact that there is at least one triple in the first box that refers to an individual or class (either as a subject or as an object of that triple) or to a property defined in the second box.

JPS is built upon an agent-driven architecture ([Zhou et al., 2020](#)). Intelligent and autonomous agents are the components that operate on its knowledge graph by retrieving inputs from it and writing outputs back to it—either adding new nodes or modifying existing ones. The agents have a representation in the knowledge graph themselves, as service instances governed by the [OntoAgent](#) ontology ([Zhou et al., 2020](#)). They exchange information by updating and querying the knowledge graph with the help of SPARQL,<sup>4</sup> a semantic query language which is part of the Semantic Web Stack. Those interactions can be realized in one of the following two ways. Agents may request a named graph by its URL in order to retrieve information via a SPARQL query or perform an update. The retrieval or update query is sent to a SPARQL endpoint, a service that performs the query or update on a triple store. As described by [Eibeck et al. \(2019\)](#), agents in JPS also communicate with each other directly. HTTP<sup>5</sup> is used as a communication protocol for this purpose by the agents, which use the GET<sup>6</sup> method to exchange data, as well as the POST<sup>7</sup> and PUT<sup>8</sup> methods to update each other about any changes in their states. All data exchanged between agents in this way, including input and output parameters, travel serialized in JSON<sup>9</sup> format, either as a URL query component<sup>10</sup> or request message body.<sup>11</sup> The input and output quantities are provided in the HTTP requests and responses, respectively. URLs passed via JSON serve as starting points for navigating, querying, and updating parts of the knowledge graph.

<sup>3</sup> <https://www.w3.org/TR/rdf11-concepts>

<sup>4</sup> <https://www.w3.org/TR/sparql11-query/>

<sup>5</sup> <https://tools.ietf.org/html/rfc2616>

<sup>6</sup> <https://tools.ietf.org/html/rfc2616#section-9.3>

<sup>7</sup> <https://tools.ietf.org/html/rfc2616#section-9.5>

<sup>8</sup> <https://tools.ietf.org/html/rfc2616#section-9.6>

<sup>9</sup> <https://tools.ietf.org/html/rfc8259>

<sup>10</sup> <https://tools.ietf.org/html/rfc3986#section-3.4>

<sup>11</sup> <https://tools.ietf.org/html/rfc2616#section-4.3>

### 3. The Parallel World Framework

As explained in the previous section, the agents' queries and updates are performed on the knowledge graph directly, that is, by default on the main part of the knowledge graph that one may consider as the base world. The Parallel World Framework changes this behavior by identifying each parallel world by its own unique scenario URL. Agents that participate in the same scenario propagate the corresponding URL as an additional input parameter while communicating with each other. The framework introduces a separate scenario agent that is associated with the particular parallel world and works as a mediator, which performs retrieval and update queries on the knowledge graph on behalf of the other participating agents.

The current implementation of the Parallel World Framework in JPS for simplicity works at the granularity level of named graphs, although this approach is not without its own specific drawbacks (Tappolet and Bernstein, 2009). As noted by Frey et al. (2018), this approach is relatively easy to understand as well as allows for reuse of existing data queries. At the same time, it also provides a very compact representation for data and queries. In JPS, the scenario agent keeps track of named graphs that have been changed by any agents participating in the same scenario. Any other agents can request the scenario agent's functionality by reading the scenario URL from the input parameters and using it for HTTP requests. If no scenario URL is given as an input parameter, agents fall back to the default mode and operate directly on the base world representation instead.

Because of the above, operations supported by JPS scenario agents can be seen as ones modeled on those described in the W3C recommendations for SPARQL 1.1 Protocol<sup>12</sup> and Graph Store HTTP Protocol.<sup>13</sup> Therefore, GET and PUT operations for entire named graphs are supported with the key "resource." A separate DELETE method is not yet implemented as it is in the case of the mentioned protocols. However, it could be easily mimicked using SPARQL updates as, in essence, every update could be broken down into delete and subsequent insert operation on data. The main idea in the Parallel World Framework is that DELETE in the base world leads to deleting the appropriate named graph physically. In contrast to that, DELETE in a parallel world leads to deleting of the copy in the parallel world and marking of the named graph as deleted with an annotation triple in the meta dataset. After that a scenario agent should respond to requests for it with the same HTTP status code as it would if there was no such resource. Similarly to the SPARQL 1.1 Protocol, GET and POST can be combined with SPARQL retrieval and update queries. In addition to that, GET and PUT can be also used to read from and write to non-RDF resources. This closely corresponds to the Semantic Document Model (Nešić, 2009). The raw scenario URL may also be used to request a semantic description of the parallel world itself by issuing HTTP GET request with <scenario URL> without keys into the JPS.

Figure 3 illustrates some of the agents involved in a sample scenario as red triangles and their calls to each other as red arrows. Their functionality is described in detail in Section 4. This section illustrates only some of their interactions with the knowledge graph via the scenario agent. The agent, represented by the orange triangle in Figure 3, may be deployed to any host. In JPS, it is currently deployed at [www.theworldavatar.com](http://www.theworldavatar.com). It can be requested by a scenario URL of the form:

`http://www.theworldavatar.com/jps/scenario/<scenario name>`

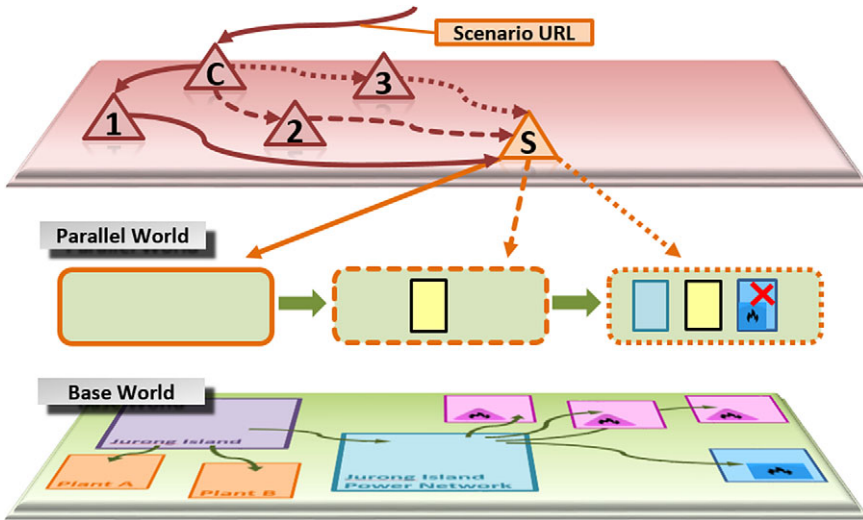
The coordination agent C is called with input parameters. It has to be able to identify the part of the knowledge graph representing the part of the base world, which would be altered in the parallel world. In our use case, those parameters consist of "Jurong Island Power Network" URL and a new scenario URL of the above form (e.g., "pownet1") as a parallel world name. The Agent C calls Agents 1, 2, and 3 one by another, providing them with all of the necessary information.

First of all, Agent 1 queries the power network to obtain a list of connected entities, which will be subject to alteration in the parallel world. Without the Parallel World Framework, the default HTTP request

`GET /kb/sgp/jurongisland/jurongislandpowernetwork`

<sup>12</sup> <https://www.w3.org/TR/sparql11-protocol/>

<sup>13</sup> <https://www.w3.org/TR/sparql11-http-rdf-update/>



**Figure 3.** Activities of the J-Park Simulator agents construct and manipulate entities placed in Parallel Worlds' Containers and accessible by individual URLs to the scenario agent S.

`/JurongIslandPowerNetwork.owl HTTP/1.1`

Host: [www.jparksimulator.com](http://www.jparksimulator.com)

would return the triple set corresponding to the named graph for “Jurong Island Power Network” as it is stored in the base world. However, since Agent 1 is called with a scenario URL, it switches the roles between the named graph URL and the scenario URL and sends the HTTP request

```
GET /jps/scenario/pownet1?query=
{"resource": "www.jparksimulator.com/kb/sgp/...
/JurongIslandPowerNetwork.owl"} HTTP/1.1
```

Host: [www.theworldavatar.com](http://www.theworldavatar.com)

instead. The original URL becomes the input parameter for the scenario agent.<sup>14</sup> Any agent may interact with JPS and use its Parallel World Framework just by switching the roles between the named graph URL and the scenario URL as described.

Upon the first HTTP request for a new scenario, the scenario agent creates a new container for storing modified named graphs of the particular parallel world. In Figure 3, the boxes under the parallel world label sketch different states of this container. The empty container on the left side stands for the new container. Upon any of the other consecutive requests, the scenario agent always looks up whether a copy of the named graph already exists in the container. If the appropriate named graph could not be found there, the agent reads the corresponding graph from the base world instead.

Agent 1 performs further queries, for example, to get information about each connected entity. All queries are redirected to the scenario agent in the same way as they are for “Jurong Island Power Network” representation. Since Agent 1 does not update any of these named graphs, the parallel world container is kept empty during such interactions with the knowledge graph.

Next, Agent 2 performs several queries and creates—as a result of its optimization—new individuals and their models as triple sets. Because of that it is called with the same scenario URL, it delegates the requests for storing the named graphs to the same scenario agent. The only difference is that it sends a HTTP PUT request for each named graph instead of a HTTP GET request. The request contains the URL

<sup>14</sup> Following the software design principle “separation of concerns,” the JPS implementation separates the handling of the scenario URL and the redirection to the scenario agent from the actual functionality of all other agents. This keeps their program code clean.



of the named graph as an input parameter and the triple set as a message body. Figure 3 shows the state of the container (middle box with dashed frame) after storing the named graph for one altered entity.

Finally, Agent 3 updates the whole parallel world structure, which is taken under consideration, and replaces some of its original entities by new and altered instances (i.e., it deletes the *hasSubsystem* triples for these entities and adds *hasSubsystem* triples for the new entities in the named graph of the whole structure). The present implementation of the update operation in JPS does not yet mark named graphs for the replaced entities as deleted in the meta dataset. They are also not deleted “physically” in the base world. They are only removed in the copy placed in the data container belonging to the particular parallel world. Therefore, comparing parallel world with the base world in order to find out what has changed there would involve comparison of the whole datasets. This may be time and resource consuming operation, in case of more complex configurations running for certain periods of time. The problem of tracing parallel worlds origins, so to say, will be addressed in the future versions of JPS. This is one of the ways in which the current Parallel World Framework itself is going to evolve and get enriched in the future. This may be also necessary for regulatory reasons as data traceability issues are looked into by the EU (Frey et al., 2018).

As illustrated in Figure 1, agents’ activities within parallel worlds overlay their activities on the particular parts of the knowledge graph. Modified parts of it get accentuated and gain prominence within contexts of their activities. Meanwhile, their origins within the base world remain unmodified in the background. They may pose as subjects of activities of the same or different agents operating within the base world or independent parallel worlds. At the same time, all corresponding named graphs are independent of each other and evolve in parallel. Knowledge graph parallelization via copies of named graphs allows JPS agents to operate on a shared world representations without interference. However, the results of their operations could be still eventually tracked back to the base world representation, which always stays untouched in the shadow of their current activities.

Several agents, which collaborate with each other within one parallel world, have to share the same view of the data in the knowledge graph. They shall not interfere with other agents’ activities, and their operations on the knowledge graph must be separated. This raises a necessity to manage different versions of named graphs (i.e., of changed entities). Different versions must be resolved in the same way for all agents participating in the same view. Problems of co-evolution (Schlobach and Beek, 2013) and identification (Sompel et al., 2010) are solved within JPS by associating parallel worlds with unique scenario URLs, which are shared between agents participating in activities related to the particular parallel world. Agents are able to find appropriate parts of the knowledge graph and collaboratively work on them via resolving those unique URLs. Since all of the named graphs for each parallel world are kept in separate containers, all of them can co-evolve independently as well. This idea resembles to large extent the concept of “context specifiers” from the Multidimensional RDF framework (Gergatsoulis and Lilis, 2005).

Entities within the higher-level structures and their subsystems are resolvable by their unique URLs. Although they share some of the generic parts with the identifiers of the base world entities, the specific parallel world information is embedded within them as well. Therefore, an agent that aggregates a certain information concerning all entities within such structure is able to resolve every single entity by its unique URL and query it for the individual property values. Mentioned agent takes for a starting point the string identifying the structure as an input parameter (i.e., “Jurong Island”). Next, it “navigates along” the knowledge graph via queries to its subsets and their interconnected entities. Bearing in mind that the scenario modifies a particular structure during the parallel world’s specific alteration process (i.e., optimization), the aggregation agent needs some functionality that allows it to resolve the “right” version of the entity. It does so by traversing the knowledge graph through unique URLs, which point to the resources stored within the particular parallel world container.

As pointed out by Fernández et al. (2015), minimization of redundant information and respect of the original modeling as well as provenance information of archives is one of the main research challenges for structured interlinked data representation systems. Described isolation of JPS agents’ operations on the knowledge graph within separate parallel worlds motivated the choice of “Independent Copies” as a

default Archiving Policy. Certainly, from the pure data storage point of view, there are possible more efficient solutions, such as those based on theory of patches (Frommhold et al., 2016) and delta calculations (Berners-Lee and Connolly, 2004) or even multi-indexed and compressed delta chains (Taelman et al., 2019). However, they may not always be a first choice from the version creation and retrieval timing point of view, which is regarded as another key performance aspect of version management systems (Tzitzikas et al., 2008). This is especially true with regard to complex objects, evolving over time and accumulating many changes building upon one another. Parallel worlds definitely show characteristics of such superstructures. Moreover, storage minimization oriented solutions may also need additional annotation of resources (Frommhold et al., 2016), which is often mentioned as a more general issue for ontology-based knowledge representation systems (Lopes et al., 2010).

Within JPS architecture, agents taking part of a scenario interactions as well as their associated non-RDF resources (such as configuration files) are annotated using RDF and OWL. Therefore, a scenario, with all of the corresponding resources, can be viewed as an RDF dataset. This makes scenarios themselves becoming parts of the JPS knowledge graph. The same holds for all of the modified named graphs, which belong to parallel worlds.

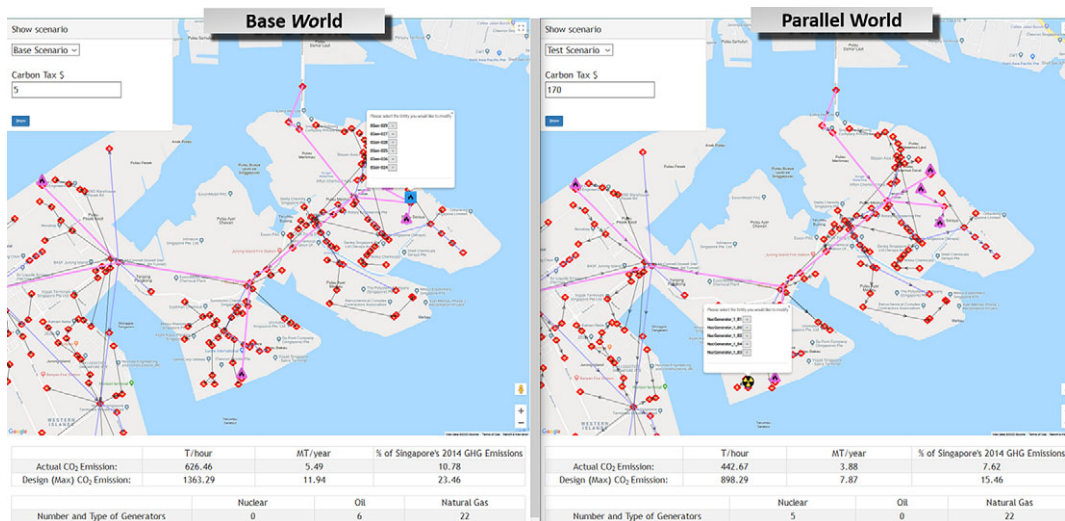
All the above does not mean that the problem of storage space was completely disregarded within JPS. Contrary to that, it has been addressed in a form of providing two different copy strategies for parallel world data. Copy-On-Write—the default option—makes sure that only those parts of the knowledge graph that are modified in a parallel world are copied over to a scenario data container. On the other hand, selecting Copy-On-Read strategy makes agents to copy all data, which they interact with in any way within a parallel world and regardless of that whether they modify it or not. JPS, as a cross-domain modeling and simulation system, integrates different types of software that produce heterogeneous data formats. Therefore, its knowledge base manager currently supports Blazegraph, Fuseki, and Rdf4j storage back-ends, apart from file-based back-end used for RDF/OWL. There are also appropriate abstractions in place, which would allow for potential implementations of cloud-based knowledge graph solutions and parallel world data containers in the future.

At the same time, a scenario can be also looked at from beyond a mere persistence layer (Christ and Nagel, 2011) perspective. It can be viewed as an activity or context, which encapsulates other activities occurring in a particular parallel world. In fact, there is an implementation of a specific `JPSContext` object, which is used by the `ScenarioAgent` objects in JPS. The context object holds all of the necessary parallel world information at run-time. This information is passed around between agents, which participate in a particular scenario. This way all agents work within one and the same scenario, where a dedicated scenario agent manages its context and acts on behalf of all the other participating agents.

#### 4. Application to the Power Network Scenario

The power network scenario presented here is based on the `OntoPowSys` ontology and the model for Jurong Island that is described in detail by Devanand et al. (2020). The model consists of over 200 bus nodes, 200 electrical lines, 5 plants with 22 natural gas generators in total, and 1 plant with 6 fuel oil generators. This section describes an application of the Parallel World Framework to this scenario as well as discusses some of the aspects of the framework itself. A more detailed discussion focused on the quantifiable outputs of the power network scenario (carbon tax amount, different configuration, etc.) will be part of another paper.

Figure 4 presents results of applying the Parallel World Framework to optimizing the power network on Jurong Island. The original and unmodified power grid, which contains six oil generators, is presented on the left side. The right side shows a modified network, where five nuclear generators replaced the oil generators in a parallel world with a simulated carbon tax value of \$170. The total number and type of generators is summarized at the bottom. Due to the level of detail of the map, generators are visualized as grouped in one location. The blue square visualizes the six oil generators, whereas the yellow and black circle groups the mentioned five nuclear generators. The bottom section shows the original and optimized



**Figure 4.** Result of a scenario-based analysis retrofitting altered and optimized power network into a Parallel World (map data ©Google).

emission details. They are displayed in tonnes per hour and mega-tonnes per year as well as the percentage of the total of Singapore's greenhouse gas emissions.

The following capabilities of the Parallel World Framework, summarized in Figure 1, are exemplified via the simulation of the alternate power network scenario:

- The lower layer corresponds to a section of the physical real world (in this case Jurong Island).
- The middle layer models the appropriate section semantically (power network, but in addition: carbon tax, etc.). It is a part of the knowledge graph persistence layer.
- The upper level contains the agents that operate on the knowledge graph and collaborate with each other. This layer can also be regarded as a “business logic” layer.

As explained in the previous section, the coordination agent C makes requests to the Agents numbered 1–3 one after another. The coordination agent can be started by a script providing the required input parameters. It is responsible for the overall service composition and makes sure that all relevant agents are called in a correct order. Intelligent automation of the agent composition process is a work in progress and is considered to be one of the essential elements of the overall JPS system architecture (Zhou et al., 2019).

The coordination agent C calls Agent 1 with a carbon tax amount and the URL that identifies any power network as its input arguments. In the presented example, the Jurong Island Power Network is taken under consideration. As an instance of the OntoPowSys (Devanand et al., 2020) ontology, it forms a part of the JPS knowledge graph. Agent 1 uses this URL to query for generators that are attached to the power network. It combines returned information with data about generating cost, CO<sub>2</sub> emissions, and the target carbon tax value in order to make a decision whether replacement of some of the existing generators by small modular reactors (SMRs) makes sense from the economic point of view. Agent 1 does not update the knowledge graph in any way. It only reads it and merely returns a serialized, and possibly empty, list of replaceable generators back to the coordination agent, which in turn passes over the list to Agent 2.

Agent 2 optimizes the number and capacities of substitutional SMRs of different types and their location on Jurong Island. It takes into account the amount of power delivered to the network as well as costs associated with the whole setup. To do so, it queries the JPS knowledge graph for the list of land lots on Jurong Island and the design capacity of the replaceable generators. Geographical coordinates of load points of the electrical network as well as potential sites are used as input parameters for the optimization

model. Different types of costs and risk as well as the overall project lifespan complete the parameter list. The model aims to balance out corresponding risks, costs, and power transition losses for the entire network.<sup>15</sup> Its work results in the creation of new named graphs for the SMRs, which will substitute the replaceable generators in the parallel world. Underlying modeling system makes sure that they are placed in the most optimal places, while taking into account all of the optimization factors listed above.

Both Agents 1 and 2 make use of the General Algebraic Modeling System (GAMS)<sup>16</sup> to solve the optimization problems via Mixed Integer Nonlinear Programming (MINLP). Due to the complexity of the model required by Agent 2, GAMS computations require around 36 hr to complete. This constraint motivated a development of an asynchronous watching service for JPS. It was designed as an autonomous and reusable part of the system, capable of serving agents operating on different use cases and parallel worlds in multiple threads of execution. The service constantly monitors the modeling system's output directories in different parallel worlds and informs instances of Agent 2 whenever the results of GAMS calculations are ready for them to be picked up and processed any further. Mentioned agents' instances themselves are not blocked by waiting for the appropriate models' convergence. The watching service is capable of keeping track of that which agent it has to forward the response back to. Upon their requests, it informs the agents about its own waiting state immediately, so they can carry on with performing any other functions or accept other requests.

Agent 3 uses the list of replaceable generators from Agent 1 and the optimization result of Agent 2 to retrofit the power network from the base world: it connects the SMRs to the closest proper buses and replaces the `hasSubsystem` relations for the replaceable generators by new relations between the power network and the SMRs. This is an example of resolving and altering individual instances of entities within a subset of a higher-level parallel world structure, described in the previous section. As already mentioned, the retrofitted network is shown as a parallel world on the left side of [Figure 4](#).

So far, the separation of modeling activities between agents relates to different phases of the overall simulation:

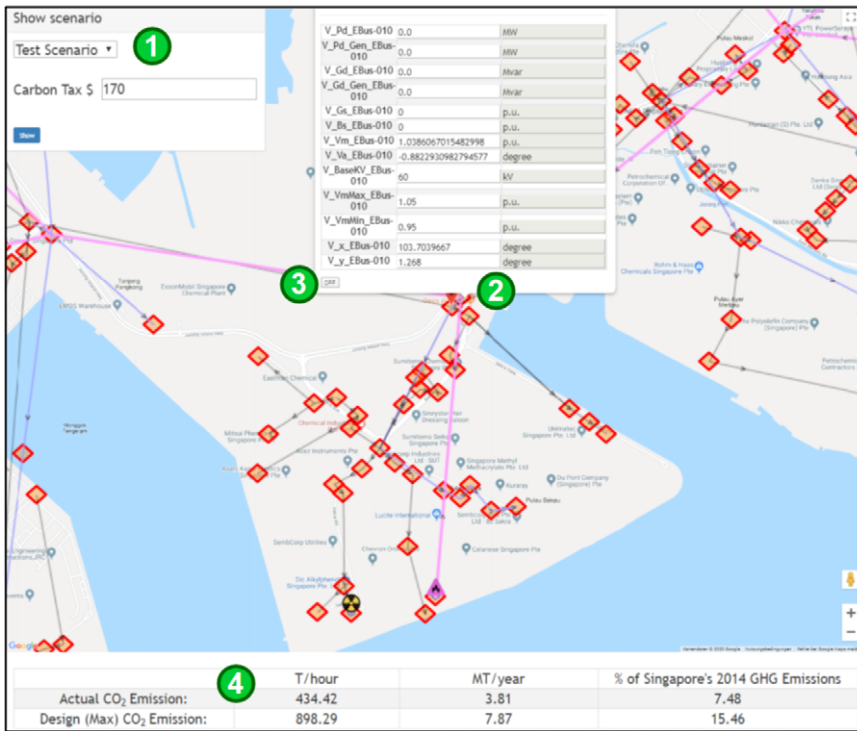
- economic analysis and policy testing;
- design phase; and
- retrofitting and installation.

The scenario of the retrofitted power network could be also extended to the operational phase, by adding the optimal power flow (OPF) agent to the base world scenario. The agent was already used by Devanand et al. (2020). Its goal is to optimize load supply with the minimum generation cost within the power grid constrains (power line limits, bus voltages, etc.). At this point, the network must be already well designed. Otherwise, the OPF will not converge. Solving it requires a well-designed grid topology, which contains information about the connections between the bus nodes and the attached equipment (power loads, power generators, etc.). The topology needs to be modified every time a change within the grid occurs. Electrical networks are modeled as sets of buses interconnected by branches. The first represent physical points of interconnection among power systems equipment. The second model paths for the flow of electrical current (Frank and Rebennack, 2016). Inputs to the OPF are matrices for buses, lines, and generators. Optimized outputs are line current and power losses, bus voltages, and generated powers. Both, inputs and outputs, are read from and written back to the knowledge graph. Some of the inputs are updated along the way of computation and are present in the output value set. More detailed description of the power network ontology and OPF implementation within JPS is presented by Devanand et al. (2020).

All agents that are called with the same scenario URL have the same view on the knowledge graph, corresponding to the particular parallel world. [Figure 1](#) illustrates this in a way of “overlying” the base world by the parallel one and making it go to the background. [Figure 5](#) shows how the optimization results

<sup>15</sup> Please refer to Devanand et al. (2019) for a more detailed description of the parameters and optimization goals.

<sup>16</sup> <https://www.gams.com/>



**Figure 5.** Parallel World user interface of J-Park Simulator for scenario listing, component alteration, and analysis (map data ©Google).

(both the retrofitted power network and the OPF calculation) are visualized in a browser. The user selects the scenario (Step 1) to visualize the corresponding version of the power network. Upon a click on one of the network entities, a pop-up window presents the current property values of the selected entity (Step 2). The user can change the values and call the OPF agent by pressing the OPF button (Step 3). After the OPF simulation has finished, another agent aggregates the actual CO<sub>2</sub> emissions of all generators. The aggregated values are displayed at the bottom of Figure 5 (Step 4).

If the user selects the base scenario, the OPF agent is requested without a scenario URL and the browser will visualize the model from the base world, otherwise the model from the corresponding parallel world. Effects of starting simulation this way are isolated solely into visualization. It operates only on the client-side copy of the visualized knowledge graph and is not persistent in any way. This ephemeral simulation could be started from the base world as well as parallel world in order to estimate and visualize any further changes to electrical networks. Adding the OPF calculation on top of the previous optimization result (retrofitted power network) demonstrates the ease of combining different aspects from different domains within JPS as well as its capabilities of keeping track of complex scenarios by the Parallel World Framework.

## 5. Conclusions

This paper presents the Parallel World Framework implemented within the JPS. It extends its capabilities by allowing to evaluate multiple versions of modeled complex superstructures during optimization process. The electrical network on Jurong Island, represented within the system, is taken as a use case. It is demonstrated that the described framework is capable of optimizing the network for carbon tax. Therefore, this part of JPS could be used as a CO<sub>2</sub> reduction planning and modeling tool. At the same time, its distributed architecture allows for potential superstructure optimizations at scale.

The framework is designed to keep documents, models, and results for the entire life-cycle together as well as to use them for optimization, prognosis, and policy testing without interference with the real world. It is thus closely related to the concept of digital twins. One of the drawbacks of this approach is its demand for data storage. Keeping large volumes of data in always-on data warehouses itself contributes to energy consumption and CO<sub>2</sub> emissions. Although it could be regarded as a sacrifice necessary to improve the present state of affairs, there is possibly yet another, slightly more sophisticated, approach. Namely, one can think of building small models that are able to generate data on demand, instead of keeping the original volumes always available, even if rarely used.

A subsequent paper will present a deeper analysis of carbon tax price for the power network scenario. It will also explore the use of the framework for hyperparameter optimization together with annotation for querying time and location of entities subject to atmospheric dispersion modeling in JPS. There is potential for using the framework in cases where parallel in-memory optimization for different initial conditions is required. It is possible to use it for spinning off multiple parallel world simulations based on different initial conditions and selecting the world which reaches optimization goals first as the end result. Furthermore, an evaluation of performance, investigating for example how the computational requirements scale with the number of agents, scenarios, and so forth, remains to be conducted.

**Funding Statement.** This project is funded by the National Research Foundation (NRF), Prime Minister's Office Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) program. Markus Kraft gratefully acknowledges the support of the Alexander von Humboldt Foundation.

**Competing Interests.** The authors declare no competing interests exist.

**Authorship Contributions.** Conceptualization, A.E., A.C., M.K., M.Q.L., and S.M.; Data curation: A.D. and G.K.; Investigation: A.E.; Methodology: A.E. and A.C.; Software: A.E., A.C., K.A., L.O., A.D., and G.K.; Visualization: A.E. and K.A.; Writing—original draft: A.E. and A.C.; Writing—review & editing: S.M. and M.K.; Supervision: R.L., I.A.K., E.Y.S.F., M.K., and M.L.; Funding acquisition: M.K., I.A.K., R.L., and E.Y.S.F. All authors approved the final submitted draft.

**Data Availability Statement.** All data involved in this study are publicly available at <http://www.theworldavatar.com>.

## References

- Ainsworth SG, Nelson ML and Van de Sompel H (2014) A framework for evaluation of composite memento temporal coherence arXiv preprint: <https://arxiv.org/abs/1402.0928>.
- Arndt N, Naumann P and Marx E (2017) Exploring the evolution and provenance of Git versioned RDF data. In Debattista J, Umbrich J and Fernández JD (eds), *Proceedings of the 3rd Workshop on Managing the Evolution and Preservation of the Data Web (MEPDaW)*, Vol. 1824 of *CEUR Workshop Proceedings*. Portorož, Slovenia, pp. 12–27.
- Arndt N, Naumann P, Radtke N, Martin M and Marx E (2019) Decentralized collaborative knowledge management using Git. *Journal of Web Semantics* 54, 29–47.
- Arndt N, Radtke N and Martin M (2016) Distributed collaboration on RDF datasets using Git: Towards the quit store. In Fensel A, Zaveri AJ, Hellmann S and Pellegrini T (eds), *SEMANTICS 2016: Proceedings of the 12th International Conference on Semantic Systems*. New York, NY: Association for Computing Machinery, pp. 25–32.
- Berners-Lee T and Connolly D (2004) Delta: An ontology for the distribution of differences between RDF graphs. <https://www.w3.org/DesignIssues/Diff.html>
- Christ F and Nagel B (2011) A reference architecture for semantic content management systems. In Nüttgens M, Thomas O and Weber B (eds), *Enterprise Modelling and Information Systems Architectures: Proceedings of the 4th International Workshop on Enterprise Modelling and Information Systems Architectures, EMISA 2011*, Vol. P-190 of *LNI*. Hamburg, Germany: Gesellschaft für Informatik (GI), pp. 135–148.
- Devanand A, Karmakar G, Krdzavac N, Aditya K, Rigo-Mariani R, Krishnan A, Eddy YF, Karimi IA and Kraft M (2020) OntoPowSys: A power system ontology for cross domain interactions in an eco industrial park. *Energy and AI*, 1. 100008 (<https://doi.org/10.1016/j.egyai.2020.100008>).
- Devanand A, Kraft M and Karimi I (2019) Optimal site selection for modular nuclear power plants. *Computers & Chemical Engineering*, 125, 339–350.
- Eibeck A, Lim MQ and Kraft M (2019) J-Park Simulator: An ontology-based platform for cross-domain scenarios in process industry. *Computers & Chemical Engineering*, 131, 106586.

- Fernández JD, Polleres A and Umbrich J** (2015) Towards efficient archiving of dynamic linked open data. In Debattista J, d'Aquin M and Lange C (eds), *Proceedings of the First DIACHRON Workshop on Managing the Evolution and Preservation of the Data Web*, Vol. 1377 of *CEUR Workshop Proceedings*. Portorož, Slovenia, pp. 34–49.
- Frank S and Rebennack S** (2016) An introduction to optimal power flow: Theory, formulation, and examples. *IIE Transactions*, 48 (12), 1172–1197.
- Frey J, Mueller K, Hellmann S, Rahm E and Vidal M-E** (2018) Evaluation of metadata representations in RDF stores. *Semantic Web*, 10(2), 205–229.
- Frommhold M, Navarro Piris R, Arndt N, Tramp S, Petersen N and Martin M** (2016) Towards versioning of arbitrary RDF data. In Fensel A, Zaveri AJ, Hellmann S and Pellegrini T (eds), *SEMANTICS 2016: Proceedings of the 12th International Conference on Semantic Systems*. New York, NY: Association for Computing Machinery, pp. 33–40.
- Gergatoulis M and Lilis P** (2005) Multidimensional RDF. In Meersman R and Tari Z (eds), *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE*, Vol. 3761 of *Lecture Notes in Computer Science*. Heidelberg, Germany: Springer, pp. 1188–1205.
- Klein M, Shankar H, Balakireva L and Van de Sompel H** (2019) The Memento Tracer Framework: Balancing quality and scalability for web archiving. In Doucet A, Isaac A, Golub K, Aalberg T and Jatowt A (eds), *Digital Libraries for Open Knowledge*, Vol. 11799 of *Lecture Notes in Computer Science*. Cham, Switzerland: Springer International Publishing, pp. 163–176.
- Kleinlanghorst MJ, Zhou L, Sikorski J, Foo Yi Shyh E, Aditya LK, Mosbach S, Karimi IA, Lau R and Kraft M** (2017) J-Park Simulator: Roadmap to smart eco-industrial parks. In *Proceedings of the Second International Conference on Internet of Things, Data and Cloud Computing (ICC 17)*, ACM, New York. pp. 1–10. 107, <https://doi.org/10.1145/3018896.3025155>.
- Kraft M and Mosbach S** (2010) The future of computational modelling in reaction engineering. *Philosophical Transactions*, 368, 3633–3644.
- Lopes N, Zimmermann A, Hogan A, Lukácsy G, Polleres A, Straccia U and Decker S** (2010) RDF needs annotations. Technical report as part of W3C Workshop—RDF Next Steps.
- Marquardt W, Morbach J, Wiesner A and Yang A** (2009) *OntoCAPE: A Re-usable Ontology for Chemical Process Engineering*. Heidelberg, Germany: Springer.
- Nešić S** (2009) Semantic Document Model to enhance data and knowledge interoperability. In Devedžić V and Gašević D (eds), *Web 2.0 & Semantic Web*, Vol. 6 of *Annals of Information Systems*. New York, NY: Springer, pp. 135–160.
- Pan M, Sikorski J, Kastner C, Akroyd J, Mosbach S, Lau R and Kraft M** (2015) Applying Industry 4.0 to the Jurong Island eco-industrial park. *Energy Procedia*, 75, 1536–1541.
- Paulheim H** (2016) Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic Web*, 8, 489–508.
- Schlobach S and Beek W** (2013) Pragmatic semantics for the web of data. *Semantics for Big Data*, AAAI Technical Report FS-13-04, pp. 58–62.
- Sompel H, Sanderson R, Nelson M, Balakireva L, Shankar H and Ainsworth S** (2010) An HTTP-based versioning mechanism for linked data. In Bizer C, Heath T, Berners-Lee T and Hausenblas M (eds), *Proceedings of the WWW2010 Workshop on Linked Data on the Web*, Vol. 628 of *CEUR Workshop Proceedings*. Raleigh, USA. [http://ceur-ws.org/Vol-628/ldow2010\\_paper13.pdf](http://ceur-ws.org/Vol-628/ldow2010_paper13.pdf).
- Taelman R, Sande MV, Van Herwegen J, Mannens E and Ruben V** (2019) Triple storage for random-access versioned querying of RDF archives. *Journal of Web Semantics*, 54, 4–28.
- Tappolet J and Bernstein A** (2009) Applied temporal RDF: Efficient temporal querying of RDF data with SPARQL. In Aroyo L, Traverso P, Ciravegna F, Cimiano P, Heath T, Hyvönen E, Mizoguchi R, Oren E, Sabou M and Simperl E (eds), *The Semantic Web: Research and Applications*, Vol. 5554 of *Lecture Notes in Computer Science*. Heidelberg, Germany: Springer, pp. 308–322.
- Tzitzikas Y, Theoharis Y and Andreou D** (2008) On storage policies for semantic web repositories that support versioning. In Bechhofer S, Hauswirth M, Hoffmann J and Koubarakis M (eds), *The Semantic Web: Research and Applications*, Vol. 5021 of *Lecture Notes in Computer Science*. Heidelberg, Germany: Springer, pp. 705–719.
- Van De Sompel H, Nelson ML, Sanderson R, Shankar H, Balakireva L and Ainsworth S** (2009) Memento: Time travel for the web. arXiv preprint: <https://arxiv.org/abs/0911.1112>
- Zhou L, Pan M, Sikorski JJ, Garud S, Aditya LK, Kleinlanghorst MJ, Karimi IA and Kraft M** (2017) Towards an ontological infrastructure for chemical process simulation and optimization in the context of eco-industrial parks. *Applied Energy*, 204, 1284–1298.
- Zhou L, Zhang C, Karimi IA and Kraft M** (2018) An ontology framework towards decentralized information management for eco-industrial parks. *Computers & Chemical Engineering*, 118, 49–63.
- Zhou X, Eibeck A, Lim MQ, Krdzavac NB and Kraft M** (2019) An agent composition framework for the J-Park Simulator—A knowledge graph for the process industry. *Computers & Chemical Engineering*, 130, 106577.
- Zhou X, Lim MQ and Kraft M** (2020) A smart contract-based agent marketplace for the J-Park Simulator—A knowledge graph for the process industry. *Computers & Chemical Engineering*, 139, 106896.

**Cite this article:** Eibeck, A. Chadzynski, A. Lim, M. Q. Aditya, K. Ong, L. Devanand, A. Karmakar, G. Mosbach, S. Lau, R. Karimi, I. A. Foo, E. Y. S. and Kraft, M. 2020. A Parallel World Framework for scenario analysis in knowledge graphs. *Data-Centric Engineering*, 1: e6. doi:<https://doi.org/10.1017/dce.2020.6>