

This work was submitted to:

Chair of Process and Data Science (PADS - Informatik 9), RWTH Aachen University

OCEAn: A Web Application for Object-Centric Emission Analysis Using Process Mining

Master's Thesis

Author: **Raimund Hensen**

Student ID: **366920**

Supervisor: Nina Graves, M. Sc.

Examiners: Prof. Wil M. P. van der Aalst
Prof. Sander J. J. Leemans

Registration Date: 2024-03-12

Submission Date: 2024-09-12

Abstract

Organizations are facing increasing pressure to report Greenhouse Gas (GHG) emissions and optimize their processes for sustainability. Object-Centric Process Mining (OCPM) provides insight into business processes from multiple perspectives based on automatically logged event data. However, the application of OCPM for emission analysis is limited by the general unavailability of emission data within Object-Centric Event Logs (OCELs).

In this work, we present a method for 1) estimating carbon emissions based on data in the OCEL 2.0 format, 2) determining total event emissions to provide event-driven sustainability insights, and 3) allocating event emissions to a set of objects using an allocation rule.

We present a selection of allocation rules and evaluate their performance on simulated event data. A graph-based allocation rule is found to produce the best results. Moreover, dropping resource objects with long lifecycles from the graph further refines the results and significantly reduces runtime. The OCEAn web application (short for Object-Centric Emission Analysis) is developed based on these findings and allows for the practical application of our methods to enable further research on process mining for sustainability by enriching OCELs with emission data.

Contents

Abstract	ii
1 Introduction	1
1.1 Motivation	2
1.2 Research Questions and Goals	4
2 Related Work	6
3 Preliminaries	9
3.1 Mathematical Foundations	9
3.2 Object-Centric Event Logs	11
3.2.1 Example Log	12
3.2.2 Object Interaction Graphs	13
4 Method	15
4.1 Emission Estimation	16
4.2 Emission Allocation	18
4.2.1 Object Allocation Framework	18
4.2.2 Allocation Rules	20
4.2.3 Handling Units and Resources	22
4.2.4 Object Interaction Graph Modifications	23
4.2.5 Allocation Properties	25
5 Evaluation	27
5.1 Setup	27
5.1.1 Target Emission Distribution	28

5.1.2	Runtime Analysis	29
5.2	Input Data	29
5.3	Results	31
5.3.1	Order Management	33
5.3.2	Container Logistics	34
5.3.3	Procure-to-Pay (P2P)	35
5.3.4	Hinge Manufacturing	36
6	Discussion	38
7	OCEAn: Object-Centric Emission Analysis Tool	42
7.1	Architecture	42
7.2	User Journey	43
8	Conclusion	46
	Bibliography	51
	Acknowledgements	52

Chapter 1

Introduction

In times of battling the climate crisis, organizations are challenged with optimizing their processes in order to drive sustainable development. The implementation of emissions trading hereby requires reporting of Greenhouse Gas (GHG) emissions, referred to as Carbon Accounting (CA).

Employing Object-Centric Process Mining (OCPM) offers a holistic view of the as-is process utilizing event data associated with objects of different types [1]. OCPM is enabled by extracting such event data from information systems that support the process. However, these information systems typically do not contain data on sustainability indicators such as carbon footprints [2]. Assuming such data were available, OCPM could provide multi-perspective sustainability insights, such as identifying emission-driving activities or object types within the process [2, 3]. Moreover, it enables calculating object carbon footprints. This is closely related to the concept of Life-Cycle Assessment (LCA) [3], that involves identifying environmental impacts associated with a product over the course of its lifecycle [4].

The recent OCEL 2.0 standard provides an interchangeable format for Object-Centric Event Logs (OCELs) [5]. However, there are currently no approaches addressing the aforementioned tasks based on these data. In general, applying OCPM for sustainability assessment has been encouraged, but contributions are mostly limited to theoretical frameworks [3, 4].

In this work, we propose a method for 1) deriving emissions from an OCEL based on hand-selected emission factors, 2) aggregating emissions to total event emissions, and 3) allocating emissions from events to objects. To make this practically applicable, the OCEAn web application (short for **O**bject-**C**entric **E**mission **A**nalysis) is implemented, allowing for the application of OCPM in CA and for exporting resulting data to form a basis for further research.

Hereby, emission estimation is achieved by allowing the user to define and apply emission rules, combining hand-selected emission factors with event data including attribute values from an OCEL. The resulting carbon footprints are then integrated into the log, allowing for a file export for further use. The second and third contribution demonstrate OCPM's capabilities of offering multi-dimensional views to the process – and in this case, to the emissions associated with its events. After selecting the perspective, i.e., an object type,

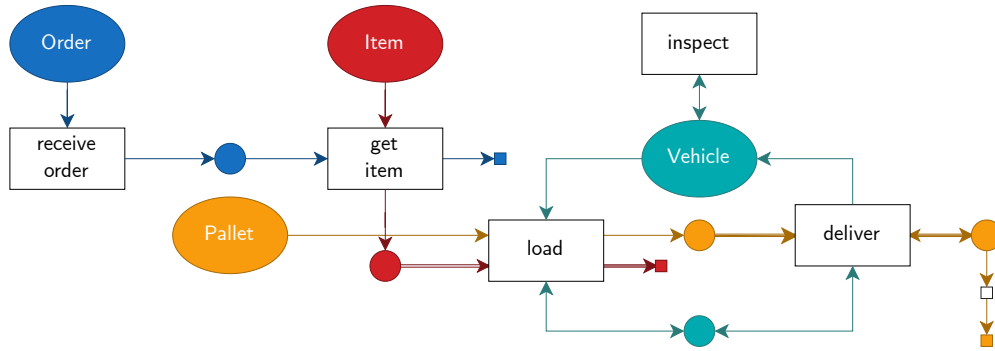


Figure 1.1: Process model of an example logistics process. Orders consist of multiple items, that are loaded onto pallets and into vehicles. A vehicle delivers the pallets to customers and needs to undergo regular inspection.

the event emissions are allocated to the objects of that type, resulting in object carbon footprints, thus addressing LCA. Object allocation involves the use of an allocation rule that selects the objects related most directly to each event. We propose three allocation rules of different complexities. A special treatment of process resources like machines, employees or vehicles is advised in multiple OCPM approaches [3, 6]. Resources are found to have longer lifecycles and more interacting objects than objects of other types, referred to as Handling Units (HUs). Object allocation considers this distinction, optionally disregarding interactions with resource objects.

An evaluation is conducted in order to compare the allocation rules proposed w.r.t. their resulting object emission distributions using suitable measures. Furthermore, runtime of the most complex allocation rule is recorded, examining potential speedups while considering implications on result preservation. Based on these findings, the OCEAn application is equipped with functionality to execute object allocation employing the best-performing rule.

1.1 Motivation

In the following, the potential of OCEL 2.0 data for the aforementioned emission analysis tasks is illustrated. To this end, consider an example process of an imaginary logistics provider. The company operates *vehicles* for delivering large goods, leading to carbon emissions. The process starts with receiving an *order* that consists of one or more *items*. Items are then loaded onto vehicles using *pallets*. *Pallets* can contain multiple items of the same order. Pallets of multiple orders may be delivered together, unloading one after the other from the vehicle. Next to the *deliver* activity, the vehicles need to undergo regular inspection, involving additional journeys. A model of this process is shown in Figure 1.1.

We assume an OCEL of the given process is extracted from an information system, containing records of events and objects together with attributes such as journey distances and average fuel consumption. However, emission data are not captured in the information system. Therefore, they are also missing in the OCEL. Still, the company wants to report the emissions caused by the process. More precisely, the goal is to

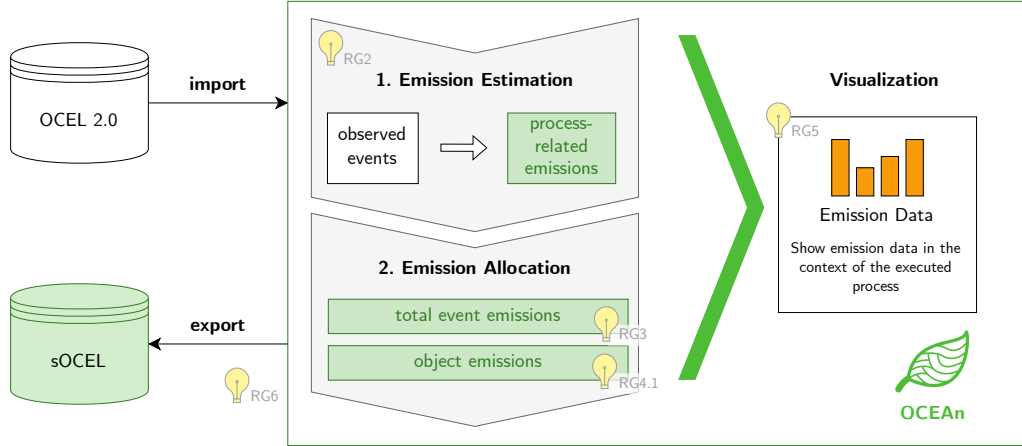


Figure 1.2: Contributions of the OCEAn application. First, event emissions are estimated by defining emission rules for different activities. Second, two perspectives on these emissions are offered in order to identify impacts of different activities or objects.

- estimate event-level emissions in a bottom-up fashion leveraging the event data,
- detect emission drivers based on these emission data such as specific activities,
- allocate all process emissions to objects of a given type, e.g., orders or items.

For simplicity, we assume that only the activities *deliver* and *inspect* cause emissions due to driving the vehicle. In order to estimate vehicle emissions, the fuel consumption of each journey is approximated by multiplying an average fuel consumption with the journey distance. Finally, an emission factor database is searched to obtain a CO₂ equivalent. Assuming the vehicle consumes 12.5 L/100 km and travels 20 km, applying an appropriate emission factor ¹ yields the event emission estimate

$$3.055 \text{ kg CO}_2\text{e/L} \cdot 12.5 \text{ L/100 km} \cdot 20 \text{ km} = 7.638 \text{ kg CO}_2\text{e}.$$

Performing this computation for each event in the OCEL offers process insights in the next step: For example, aggregating all event emissions by the event’s activity, the overall emission share of inspection journeys can be computed ².

Moreover, the company wants to assess the emissions caused by each order and each item. This adds two additional perspectives in which overall process emissions are fully distributed among these target objects. To this end, all emissions along an object’s lifecycle within the process are added to obtain a carbon footprint. This way, LCA is performed on the selected objects, with the system boundaries here defined by the scope of the process data, i.e., within the logistics company. Upstream emissions from producing an item are not included, neither are emissions caused by the use phase or disposal of the product. After computing carbon footprints on either *order* or *item* level, more process insights are gained. For example, the impact of merging multiple deliveries can be assessed.

¹“Emission intensity of diesel (without biofuel content) per liter used in vehicle” – source: GEMIS (<https://www.climatiq.io/data/emission-factor/d56128ea-76e5-4a31-a6fe-b5e1ed855e52>)

²In this example, just considering distances allows for the same comparison. Emission estimation only adds value as soon as different activities employ different emission factors.

The OCEAn application enables solving all three previously mentioned problems. Figure 1.2 depicts these use cases and the methods proposed, divided in two phases: First, emission estimation allows bottom-up event emission computation using an emission factor selected by the user. Second, emission allocation offers different perspectives on emission data, including the event perspective, and the object perspective.

Object allocation involves identifying relations between events and objects. OCEs contain such relations (objects participating in events), however, these data are not sufficient: In the above example, neither *orders* nor *items* participate in a delivery. Therefore, we propose an *object allocation framework* employing different allocation rules in order to find more links between events and the objects in question.

Apart from possible applications like in the above example, OCEAn aims at providing the research community with emission-annotated datasets and thus enabling further research. To this end, it is built to work with event data in the OCEL 2.0 file format [5], and allows integrating the computed emission values into that format in order to obtain a sustainability-enriched OCEL (sOCEL).

1.2 Research Questions and Goals

To structure the work, the problem statement from the previous section is divided in seven research questions (RQs).

- RQ1 How are techniques from PM used to carry out carbon emission analysis? How does OCPM improve applicability?
- RQ2 How can process-related emissions be estimated using the data available in an OCEL?
- RQ3 How can the emissions recorded in the event log be aggregated and distributed to gain sustainability insights w.r.t. the different events and activities?
- RQ4a How can the emissions recorded in the event log be aggregated and distributed to gain sustainability insights w.r.t. a specific subset of objects as required by LCA?
- RQ4b What characteristics of the event log impact the results of different allocation techniques in terms of performance and emission distribution?
- RQ5 What process-related sustainability insights can be directly gained by considering emission data in the context of an OCEL?
- RQ6 How can the integration of emission data serve as a first step for a practical application of PM for sustainability?

In the following, research goals are defined in order to answer the previously defined research questions.

- RG1 Investigation of previous works employing process mining for carbon emission analysis, as well as advances in OCPM influencing this.
- RG2 Development and implementation of a method that uses the expressiveness of an OCEL to estimate emissions caused by behavior observed in the event log.

- RG3 Development and implementation of a method to support the aggregation and distribution of the process emissions to events and activities.
- RG4a Development and implementation of methods for the allocation of process-related emissions to a subset of objects.
- RG4b Identification of OCEL characteristics that impact the results of allocation techniques in terms of performance and emission distribution.
- RG5 Visualization of emission data derived from an OCEL to support the identification of basic process-related sustainability insights.
- RG6 Provide a web application supporting the estimation and integration of process-related emissions as well as the export of an OCEL enhanced by these data.

The remainder of this thesis is structured as follows: Chapter 2 addresses related work in order to pursue RG1. Following this, mathematical backgrounds and OCEs are introduced. Chapter 4 establishes the methods used in both phases, including emission estimation (RG2) and emission allocation (RG3, RG4a). After evaluating the allocation framework in Chapter 5 to address RG4b, results and possible extensions are discussed in Chapter 6. Chapter 7 gives implementation details of the OCEAn app and shows a potential user journey, summarizing the contributions to RG5 and RG6. The thesis is concluded with Chapter 8.

Chapter 2

Related Work

This chapter addresses RG1 by outlining the fundamentals and advances in topics related to our work. These include the basics of CA and emission factors, the application of PM for sustainability, and OCPM techniques relevant for our method.

Carbon Accounting

The GHG Protocol from 2004 remains the de-facto standard for carbon emission reporting [7]. Other standards focus on particular sectors. For example, in logistics, the GLEC framework provides reporting standards and emission factors for different modes of transport [8]. The target quantity in CA is the Global Warming Potential (GWP), measured by the weight of carbon dioxide equivalents (e.g., kg CO₂e). This allows summarizing the contribution to global warming in just one quantity while considering not just carbon dioxide, but a variety of GHGs, such as methane and nitrous oxide [9]. Publicly available emission factors used for CA mostly follow this principle.

In CA, as well as in emission factor databases, two fundamental patterns are visible [7]: The top-down approach and the bottom-up approach. The top-down approach employs Environmentally Extended Input-Output Analysis (EEIOA) [10], aggregating emissions and spendings of whole sectors and countries. The outcome is spend-based emission factors. Such factors are offered by databases such as EXIOBASE [11] or WIOD [12].

In the bottom-up approach, emission data are determined for particular process activities based on direct measurement, scientific studies or industry standards. These factors are also called activity-based emission factors and map quantities in different physical units to carbon equivalents [7]. Major databases providing activity-based emission factors include ecoinvent [13] and the UK government conversion factors [14].

While spend-based factors are universally applicable due to the macroeconomic fashion of input-output analysis, they lack precision when applied to granular process data. Therefore, the use of activity-based factors is preferable in the context of event logs [15]. The aforementioned databases exhibit a variety of data formats and access interfaces, which presents a challenge when attempting to combine them for use in CA for heterogeneous processes. APIs like climatq solve this problem by bundling emission factors from currently 39 sources including all of the aforementioned vendors [16].

Process Mining for Sustainability

PM itself is noted for not being capable of generating emission data, as this requires access to emission factors and the selection of those appropriate to the event or object in question [2]. Therefore, applications of PM for sustainability purposes are still rare and assume availability of event data with sustainability-related KPIs. Given that these data are available, PM can unveil emission drivers within the process, such as different activities or trace variants [2, 3]. OCPM is considered especially useful as emission-intense objects can be identified and LCA is enabled [3].

In LCA, environmental impacts of a product or process output are analyzed, rather than those of single events. LCA considers specific parts of a product’s lifecycle, like *cradle-to-grave* or *gate-to-gate* [4]. PM’s potential to aid the implementation of LCA has been acknowledged, with process discovery being employed in order to relate environmental impacts to different process variants, saving time and costs compared to classical LCA [4]. Again, this approach requires integration of all impact factors needed into the event log. Also, as classical PM is used, the objectives can only be determined at a single case level [4].

A recent approach to LCA in electronics manufacturing represents environmental impacts as activities in a Petri net [17]. This allows performing LCA by replaying the model, making the approach both visual and scalable to large data. However, the model represents a pre-defined control flow within the process and cannot handle outliers correctly.

Activity-Based Costing (ABC) is an accounting technique for assigning indirect costs to services or products based on the different activities they require. Though originally intended for purely monetary costs, ABC has also been used for GHG emission allocation, distributing an overall carbon footprint among products [18]. The high complexity of ABC lead to the proposal of Time-Driven ABC (TDABC), considering activity duration as the only cost driver [19]. PM is used in ABC mostly to capture timespans between events, serving as input for TDABC. Classical PM is used to discover BPMN models annotated with waiting time [20]. Another approach does not rely on process discovery but makes use of event logs containing start and end timestamps for each event [21]. Classical event logs in the XES format support cost annotations by means of the XES cost extension [22]. However, a similar concept for OCEs is still missing.

Celonis, a major PM vendor, recently presented their “Material Emissions” app in cooperation with climatiq. The app facilitates reporting of scope 3 emissions from procurement based on event data, and helps identifying emission reduction potentials, for example by changing vendors [23, 24]. Another Celonis app allows for freight emission calculation, enabling reporting and optimization in logistics processes [25]. Horsthofer-Rauch et al. [26] utilize the Celonis platform for sustainability enhancement of generic production processes. To this end, they employ an ontology of common entity relations for data extraction, which includes object types such as machines, products, energy, and waste. OCPM is considered especially useful in this context as it captures production steps and product composition as inter-object relations [26]. The aforementioned commercial solutions all employ PM for sustainability assessment. However, there is currently no tool that is both platform-independent and sector-agnostic.

Object-Centric Process Mining

OCPM is a recent concept based on the approach to associate events with multiple objects instead of one unique case identifier. This way, the single-case assumption from classical PM is dropped, allowing for a more holistic view of the process [1]. However, with the new event log format, new challenges arise: Many existing process mining techniques are no longer applicable and need flattening of the data to again extract a classical event log. In process discovery, the Object-Centric Petri Net (OCPN) is proposed, with the mining algorithm based on existing techniques applied on flattened logs for each object type [27]. Regarding data formats, the OCEL standard is developed in an attempt to provide an interoperable, scalable file format. The first version of the standard [28] is extended by the current version OCEL 2.0 that introduces object-to-object (O2O) relations, qualifiers for event-to-object (E2O) and O2O relations, dynamic object attributes and a new, more efficient way of storing the data using an `sqlite` database file [5].

Many OCPM techniques involve the use of object interaction graphs, in which two objects are connected if they interact in some event. The graph is used in order to create a concept similar to cases in classical process mining, with *process executions* either defined as connected components or sets of those objects closest to a representative of a selected type. Isomorphic process executions are then considered *variants* [29]. Another approach directly views an OCEL as an *event knowledge graph* made from events and objects together with their interactions and directly-follows relations [6]. A new graph-based process model is presented with the TOTeM model [30], extracting temporal relations between object lifespans, grouping by object types, and visualizing relations between the types including cardinalities.

Resources in classical PM are entities like machines or employees that execute events or are used by an event. In this context, resources are often included in event logs as an additional attribute, and used in tasks like social network discovery [31]. In OCPM, resources are naturally represented as objects instead of event attribute values, facilitating the inclusion of further properties of the resource as object attributes [3], and allowing for multiple resource links per event. A special role of resources in event knowledge graphs [6] is brought up as well as their importance for applications for sustainability [3].

Viewing processes from the perspective of different object types is noted to be especially valuable in sustainability assessment [3, 24]. When applying LCA or ABC, employing OCPM allows selecting different object types as targets for assessing impacts like GHG emissions caused by the respective objects instead of being limited to a one-dimensional case output perspective.

Chapter 3

Preliminaries

This chapter introduces all concepts and notation that are used in the remainder of this thesis. In Section 3.1, basic mathematical foundations are introduced, followed by object-centric event logs (OCELs) in Section 3.2.

3.1 Mathematical Foundations

- $\mathbb{N} = \{1, 2, 3, \dots\}$ denotes the set of *natural numbers*, $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$ the set of *natural numbers including zero*, \mathbb{Z} the set of *integers* and \mathbb{R} the set of *real numbers*. *Infinity* is denoted by ∞ and is not contained in any of the aforementioned sets.
- For a set X , $\mathcal{P}(X)$ denotes the *powerset* of X , i.e., the set of subsets of X . A *partition* of X is a set of subsets $P \subseteq \mathcal{P}(X)$ s.t. the subsets in P are pairwise disjoint ($X' \cap X'' = \emptyset \ \forall X', X'' \in P$) and their union is X , i.e., $(\bigcup_{X' \in P} X' = X)$.
- A *function* $f: X \rightarrow Y$ assigns a value $f(x) \in Y$ to each $x \in X$. A *partial function* $f: X \not\rightarrow Y$ does not necessarily map all $x \in X$ to a value in Y . For those $x \in X$ where $f(x)$ is undefined, we write $f(x) = \perp$.

For a function or partial function $f: X \rightarrow Y$, the *range* of f is given by $\text{rng}(f) = \{f(x) \mid x \in X\} \setminus \{\perp\}$. The *domain* of f is denoted by $\text{dom}(f) = \{x \in X \mid f(x) \neq \perp\}$. The *inverse image* of a set $Y' \subseteq Y$ under f is defined as $f^{-1}(Y') = \{x \in X \mid f(x) \in Y'\}$, making f^{-1} a function $\mathcal{P}(Y) \rightarrow \mathcal{P}(X)$.

- We denote the *minimum* of a finite set $Y \subseteq \mathbb{R} \cup \{\infty\}$ by $\min Y$. We assume $\min \emptyset = \infty$.

For sets X and $Y \subseteq \mathbb{R} \cup \{\infty\}$, a function $f: X \rightarrow Y$ and a finite set $X' \subseteq X$, we write

$$\arg \min_{x \in X'} f(x) := f^{-1}\left(\left\{\min\{f(x) \mid x \in X'\}\right\}\right).$$

- A *sequence* of length $n \in \mathbb{N}_0$ over a domain set X is an ordered list containing elements $x_1, \dots, x_n \in X$ and is denoted by $\sigma = \langle x_1, \dots, x_n \rangle$. The length of a sequence is $|\sigma| := n$. The set of sequences over X is denoted by X^* . The set of

sequences of a specific length $n \in \mathbb{N}_0$ over X is denoted by X^n . Sequences can be empty, with $X^0 = \{\langle \rangle\}$.

- An *undirected graph* is a tuple $G = (V, E)$ where $V(G) := V$ is the set of vertices and $E(G) := E \subseteq \{\{u, v\} \subseteq V \mid u \neq v\}$ is the set of edges.
 - Two vertices $u, v \in V$ are *connected* in G if $\{u, v\} \in E$.
 - A *path* of length $n \in \mathbb{N}_0$ from v_0 to v_n in G is a sequence of vertices $\langle v_0, \dots, v_n \rangle \in V^{n+1}$ where $\{v_i, v_{i+1}\} \in E$ for all $0 \leq i \leq n$.
 - A vertex $v \in V$ is *reachable* in G from $u \in V$ if there exists a path from u to v in G .
 - A *connected component* of G is a maximal set $V' \subseteq V$ s.t. all vertices in V' are mutually reachable. The set of connected components of G is a partition of V and is denoted by $C(G)$.
 - The *distance* of u and v in G is the minimal $n \in \mathbb{N}_0$ s.t. there exists a path of length n from u to v in G and is denoted by $\text{dist}_G(u, v) := n$. If v is not reachable from u in G , then $\text{dist}_G(u, v) := \infty$. This makes dist_G a function $V \times V \rightarrow \mathbb{N}_0 \cup \{\infty\}$. Note that $\text{dist}_G(u, v) = 0$ if and only if $u = v$.
 - The *neighborhood* of $v \in V$ is the set of vertices v is connected to. The *degree* of v is the size of the neighborhood of v and is denoted by $\deg_G(v) := |\{u \in V \mid \{u, v\} \in E(G)\}|$.
 - A *clique* in G is a set of vertices that are mutually connected.

For data visualization and evaluation purposes, some descriptive statistics are used. These measures are defined for a given sequence of numbers $x = \langle x_1, \dots, x_n \rangle \in \mathbb{R}^n$, $n \in \mathbb{N}$.

- The *mean* of x is defined as $\text{mean}(x) := \frac{1}{n} \sum_{i=1}^n x_i$.
- For an ordering $x^{(1)} \leq \dots \leq x^{(n)}$ of x , the *median* of x is denoted by

$$\text{median}(x) := \begin{cases} x^{((n+1)/2)} & \text{if } n \text{ is odd,} \\ \frac{1}{2}(x^{(n/2)} + x^{(n/2+1)}) & \text{if } n \text{ is even.} \end{cases}$$

- For $n > 1$, the *standard deviation* of x is defined as

$$\text{std}(x) := \sqrt{\frac{1}{n-1} \sum_{i=1}^n (x_i - \text{mean}(x))^2}.$$

The *variation coefficient* is a relative version of the standard deviation and is denoted by $\text{cv}(x) := \frac{\text{std}(x)}{\text{mean}(x)}$.

In the remainder of the thesis, the compact notation “ $m[a - b]$ ” is used to summarize a number series $x \in \mathbb{R}^*$ with $\text{median}(x) = m$, $\min(x) = a$ and $\max(x) = b$.

3.2 Object-Centric Event Logs

As mentioned in Chapter 2, there have been two versions of the OCEL standard as of now. This project's concept is built around the most recent one of them, the OCEL 2.0 standard [5]. In the following, the formal definition of an OCEL according to the standard is given.

First, universe notation is used to denote a set of unique identifiers for events, objects and other entities. The following universes are defined as pairwise disjoint sets:

- \mathbb{U}_{ev} is the universe of events,
- \mathbb{U}_{etype} is the universe of event types,
- \mathbb{U}_{obj} is the universe of objects,
- \mathbb{U}_{otype} is the universe of object types,
- \mathbb{U}_{attr} is the universe of attribute names,
- \mathbb{U}_{qual} is the universe of qualifiers.

Additionally,

- \mathbb{U}_{val} is the universe of attribute values,
- \mathbb{U}_{time} is the universe of timestamps, including $0, \infty \in \mathbb{U}_{time}$, totally ordered with $0 \leq t \leq \infty$ for all $t \in \mathbb{U}_{time}$.

Definition 3.1: An Object-Centric Event Log (OCEL) is a tuple

$$L = (E, O, EA, OA, evtype, time, objtype, eatype, oatype, eaval, oaval, E2O, O2O)$$

where

- $E \subseteq \mathbb{U}_{ev}$ is the set of events,
- $O \subseteq \mathbb{U}_{obj}$ is the set of objects,
- $evtype: E \rightarrow \mathbb{U}_{etype}$ assigns event types to events,
- $time: E \rightarrow \mathbb{U}_{time}$ assigns timestamps to events,
- $objtype: O \rightarrow \mathbb{U}_{otype}$ assigns object types to objects,
- $EA \subseteq \mathbb{U}_{attr}$ is the set of event attributes,
- $OA \subseteq \mathbb{U}_{attr}$ is the set of object attributes,
- $eatype: EA \rightarrow \mathbb{U}_{etype}$ assigns event types to event attributes,
- $oatype: OA \rightarrow \mathbb{U}_{otype}$ assigns object types to object attributes,
- $eaval: (E \times EA) \not\rightarrow \mathbb{U}_{val}$ assigns values to event attributes,
- $oaval: (O \times OA \times \mathbb{U}_{time}) \not\rightarrow \mathbb{U}_{val}$ assigns values to object attributes,
- $E2O \subseteq E \times \mathbb{U}_{qual} \times O$ are the qualified event-to-object relations,
- $O2O \subseteq O \times \mathbb{U}_{qual} \times O$ are the qualified object-to-object relations.

Table 3.1: An OCEL from a logistics process, represented by three tables: On the left, events and objects are given with identifiers, activity resp. object type, and further attributes. On the right, the E2O relations are listed, i.e., participations of objects in events.

Event	Activity	Distance	Timestamp	Event	Object
receive_o1	receive order	—	05-07 21:00	receive_o1	o1
get_i1	get item	—	05-10 18:00	get_i1	o1
receive_o2	receive order	—	05-14 10:30	get_i1	i1
load_p1	load	—	05-14 12:00	receive_o2	o2
deliver_1	deliver	52.00	05-14 13:00	load_p1	i1
get_i3	get item	—	05-15 13:00	load_p1	p1
get_i2	get item	—	05-20 13:00	load_p1	v1
get_i4	get item	—	05-20 13:10	deliver_1	p1
load_p2	load	—	05-20 14:00	deliver_1	v1
load_p3	load	—	05-20 14:05	get_i3	o2
deliver_p2p3	deliver	20.00	05-20 15:00	get_i3	i3
deliver_p3	deliver	40.00	05-20 15:42	get_i2	o1
inspect_v1	inspect	118.00	05-22 15:00	get_i2	i2
inspect_v2	inspect	115.00	05-26 16:30	get_i2	i2
				get_i4	o2
Object	Obj. type	Consumption		get_i4	i4
o1	Order	—		load_p2	i2
o2	Order	—		load_p2	p2
i1	Item	—		load_p2	v2
i2	Item	—		load_p3	i3
i3	Item	—		load_p3	i4
i4	Item	—		load_p3	p3
p1	Pallet	—		load_p3	v2
p2	Pallet	—		deliver_p2p3	p2
p3	Pallet	—		deliver_p2p3	p3
v1	Vehicle	10		deliver_p2p3	v2
v2	Vehicle	12.5		deliver_p3	p3
				deliver_p3	v2
				inspect_v1	v1
				inspect_v2	v2

In the following, we define auxiliary notation for an OCEL L . Parts of these are originally proposed by the OCEL 2.0 standard:

- $ET(L) = \{evtype(e) \mid e \in E\}$ is the set of event types (activities),
- $OT(L) = \{objtype(o) \mid o \in O\}$ is the set of object types,
- For an event $e \in E$, the set of objects participating in e is denoted by

$$obj_L(e) := \{o \in O \mid \exists q \in \mathbb{U}_{qual}: (e, q, o) \in E2O\}.$$

3.2.1 Example Log

Table 3.1 shows an exemplary OCEL. The dataset corresponds to the same logistics process introduced in Section 1.1: The company receives *orders* consisting of multiple *items*. These items get loaded onto *pallets* and are then delivered using *vehicles*.

The data are shown by means of three tables: events, objects, and E2O relations. As opposed to other formats like OCEL 1.0 [28], the second version of the standard uses this

relational data model to efficiently store object-centric process data in an `sqlite` database file, avoiding sparse tables.

The events (top left table) are represented by an identifier and are assigned an activity, a timestamp, and more attributes, in this case a distance (for the activities *deliver* and *inspect*). Objects (bottom left) have an identifier and a type, with *vehicles* having the attribute *consumption*. E2O relations (right) are pairs of one event and one object. For simplicity, the example does not contain qualified relations.

Chapter 4 introduces the method employed for emission analysis of OCELs. For this, the logistics event log is used as a minimal example to demonstrate different parts of the method.

3.2.2 Object Interaction Graphs

In OCPM, the object interaction graph captures relations between pairs of objects extracted from the log. Two objects have are connected in this graph if they share some common event [6, 29]. With the advent of OCEL 2.0 [5], object interaction graphs have been enriched with the newly added O2O relations, or vice versa [30]. This concept is used in this work as well, allowing to explicitly add more edges to the graph if two objects do not interact in an event.

Object interaction graphs are used in this work for allocating emissions to a subset of the objects, called target objects. To this end, shortest paths from other objects to target objects are computed. As edges between pairs of target objects are not relevant for this use case, they are removed in order to reduce graph size.

Definition 3.2 (Object Interaction Graph): For an OCEL L and a set of target objects $\Omega \subseteq O$ with $\Omega \neq \emptyset$, the *object interaction graph* $OG^\Omega(L)$ is an undirected graph defined by

$$\begin{aligned} V(OG^\Omega(L)) &= O, \\ E(OG^\Omega(L)) &= \left(\left\{ \{o_1, o_2\} \subseteq O \mid o_1 \neq o_2 \wedge \exists e \in E: o_1, o_2 \in \text{obj}_L(e) \right\} \right. \\ &\quad \cup \left. \left\{ \{o_1, o_2\} \subseteq O \mid o_1 \neq o_2 \wedge (o_1, q, o_2) \in O2O \right\} \right) \\ &\quad \setminus \mathcal{P}(\Omega) \end{aligned}$$

Figure 3.1 shows the object interaction graph $OG^\Omega(L)$ extracted from the example log from Table 3.1, with $\Omega := \{\text{i1}, \text{i2}, \text{i3}, \text{i4}\}$ defined as the set of target objects. As the OCEL does not contain O2O relations, the edges all correspond to object interactions. There is no edge between the *item* objects *i3* and *i4* as they are both target objects. When instead labeling the *pallet* objects as targets, the items will be connected, and the edge between *p2* and *p3* will be removed.

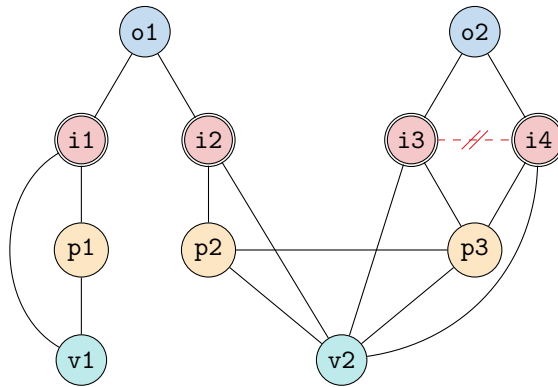


Figure 3.1: Object interaction graph of the logistics example log. For visualization, objects are colored depending on their type, target objects (here all *items*) are highlighted with double borders. The *items* **i3** and **i4** are **not** connected as they are both target objects.

Chapter 4

Method

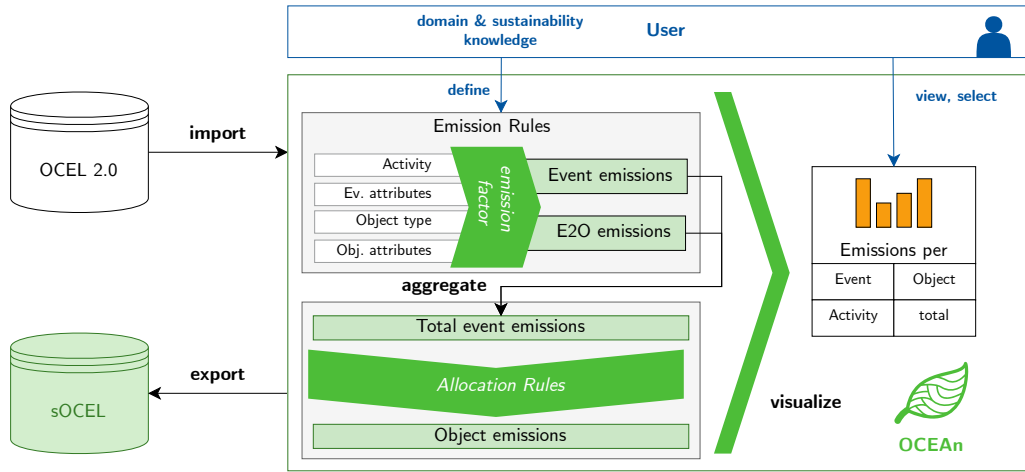


Figure 4.1: Schematic view of our method. After importing an OCEL, emission rules are defined by the user and applied to estimate emissions on event or event-to-object level (phase 1). Following this, emissions are aggregated for events, or for objects using the object allocation framework (phase 2). Results are visualized to the user, and integrated into the (s)OCEL when exporting.

The overall goal of our work is to develop a tool enabling emission analysis leveraging data contained in an event log in the OCEL 2.0 [5] standard. Figure 4.1 shows how the methods used to achieve this are employed by the OCEAn application. After importing an OCEL, the procedure is split into two phases.

Phase 1 first determines emission data based on the OCEL. The data is obtained by applying emission rules, using a specified emission factor and data from the OCEL to compute emission values on event or event-to-object (E2O) level. Section 4.1 further elaborates on this emission estimation process.

Phase 2 further processes emissions specified on event or E2O level to provide different perspectives. First, an event perspective is obtained by performing a simple aggregation. This provides insights like emission-driving activities. Second, the *object allocation* framework is proposed, mapping emissions on event and E2O level to objects. At its core, the framework uses different *allocation rules* that identify those objects related most

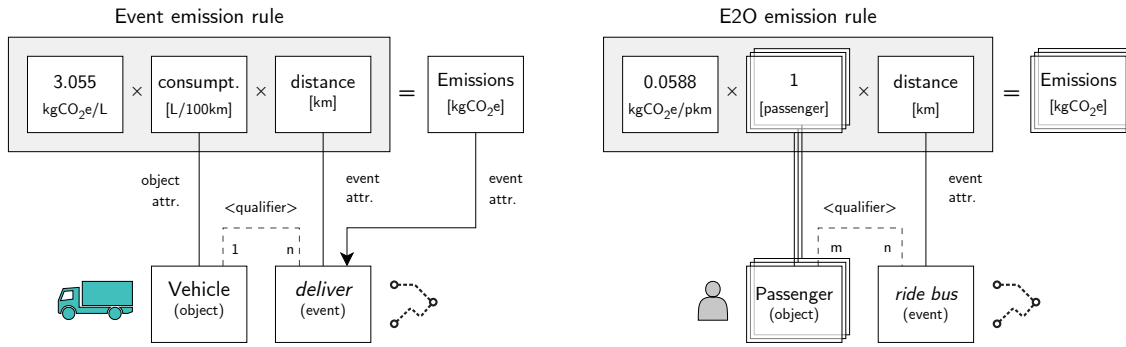


Figure 4.2: Emission rules using attributes of an event and associated objects. Left: Event emission rule, computing fuel-based emissions in a logistics process. Right: E2O emission rule, computing emission contribution per passenger in a bus transport.

directly with each event. Details of the framework and the emission rules are given in Section 4.2.

The OCEAn application provides a frontend to control the import and export of OCELs as well as the execution of both phases. It also visualizes the results obtained from executing both phases. Further implementation details are given in Chapter 7. In the following, both parts of the method are addressed.

4.1 Emission Estimation

In order to compute emissions on event or E2O level, we propose *emission rules* that extract different types of data from an OCEL, and apply a selected emission factor to determine concrete emissions.





Emission factors are real numbers stating a conversion of a given quantity to CO₂ equivalents. These quantities can be of various units, depending on the activity. Examples include

- kWh – electricity from grid,
- L – fuel combustion,
- pkm (*passenger-kilometers*) – bus transportation,
- tkm (*tonne-kilometers*) – freight.

In OCELs, events and objects can be annotated with attribute values. Each attribute is associated with a specific activity or object type. The same is adopted for emission rules, leading to emissions only being computed for events of a given activity. To compute emissions for multiple activities in the process, multiple emission rules are defined and their results combined. Moreover, OCELs can contain both attributes with numerical and categorical values. Here, we only refer to numerical attributes to be used in emission rules.

The units pkm and tkm mentioned above are commonly used for computing transport emissions [32]. As they are composed of two quantities by multiplication, emission rules also support multiplying values of different attributes.

Table 4.1: OCEL data usable in emission rules, depending on the level on which emissions are computed. Constant values, attributes of events and those object types uniquely associated with the activity are usable on both levels. E2O rules can additionally include attributes of the object type it is associated with. For the logistics example, the unique types per activity are listed on the right.

Data type	Usable in ...		Activity	Unique types			
	Event em. rule	E2O em. rule		 ord	 item	 pllt	 vhcl
Constant	✓	✓	<i>receive order</i>	✓			
Event attr.	✓	✓	<i>get item</i>	✓	✓		
Object attr. (of E2O object)	✗	✓	<i>load</i>		✗	✓	✓
Object attr. (of unique type)	✓	✓	<i>deliver</i>			✗	✓
Object attr. (others)	✗	✗	<i>inspect</i>				✓

In Section 1.1, an example for computing emissions delivery events was given, involving journey distance [km] and average fuel consumption [L/100 km], with example values:

$$3.055 \text{ kg CO}_2\text{e/L} \cdot 12.5 \text{ L/100 km} \cdot 20 \text{ km} = 7.638 \text{ kg CO}_2\text{e} \text{ } ^{(1)}$$

Figure 4.2 (left) shows how this computation is represented by an emission rule: Attribute values linked to the *vehicle* object (consumption) and the *deliver* event (distance) are multiplied with the emission factor, yielding an emission value for the event. The result is later integrated into the OCEL as a new event attribute on the *deliver* event. This is part of the emission allocation framework introduced in Section 4.2.

In this example, emissions are computed on event level. Still, a vehicle’s object attribute value is used. The value is well-defined because exactly one vehicle participates in the *deliver* event. Note that attributes of other objects from the logistics process (Table 3.1), such as pallets, cannot be used this way: The event `deliver_p2p3` is linked to multiple pallets (p2 and p3), making *pallet* a *non-unique type* of the activity *delivery*.

In object-centric Petri nets, a process model introduced in OCPM, this concept is represented by *variable arcs* [27]. When an object type is not unique for a given activity, the arc between that activity’s transition and a place associated with the object type is shown as a double arrow when visualizing the model. For the logistics OCEL, the different activities’ unique and non-unique object types are shown on the right side of Table 4.1. Object attributes of non-unique types can only be used by the second type of emission rules, *E2O emission rules*, as shown on the left side of the table.

E2O emission rules compute emissions on the level of E2O relations, with each result linked to an event and an object. Therefore, the rule is defined for a specific activity and object type. Figure 4.2 (right) depicts such an allocation rule. The example is from a different process, where multiple *passengers* (represented as objects) are transported by a bus. To employ an emission factor of the unit kg CO₂e/pkm, the passengers are usually counted.

¹ “Emission intensity of diesel (without biofuel content) per liter used in vehicle” – source: GEMIS (<https://www.climatiq.io/data/emission-factor/d56128ea-76e5-4a31-a6fe-b5e1ed855e52>)

E2O emission rules work differently: A count of 1 is used in order to compute individual emission shares per *passenger* object. Thus, E2O emissions are obtained by

$$0.059 \text{ kg CO}_2\text{e/pkm} \cdot 1 \text{ passenger} \cdot 20 \text{ km} = 1.176 \text{ kg CO}_2\text{e} \text{ (}^2\text{)}.$$

The same computation is carried out for each passenger linked to the *ride bus* event. In the following step, these values are aggregated in order to obtain total event emissions. This forms one part of *emission allocation*, presented in the following section.

4.2 Emission Allocation

The previous section showed how emissions can be determined using an OCEL. In this section, these emissions are used to determine the total emissions associated with an event or an object. To this end, the emissions are

1. aggregated on event level in order to gain event and activity-based emission insights,
2. allocated to a set of target objects, accumulating object carbon footprints.

The input to both tasks consists of event emissions and event-to-object (E2O) emissions, both possible outcomes of applying emission rules as previously defined. Note that all emission values are denoted as real numbers. We assume all of these quantities to be specified in kilograms of CO₂ equivalents (kg CO₂e).

To solve the first task, aggregation on event level is directly performed on the input data.

Definition 4.1 (Total Event Emissions): Let L be an OCEL, $em^E: E \rightarrow \mathbb{R}$ a function specifying every event’s emissions, and $em^{E2O}: E \times O \rightarrow \mathbb{R}$ a function determining the event-to-object emissions where for all $e \in E$, $o \in O$: $o \notin obj_L(e) \implies em^{E2O}(e, o) = 0$. For any $e \in E$ the total event emissions are denoted by $em_{tot}^E(e) := em^E(e) + \sum_{o \in O} em^{E2O}(e, o)$.

Computing em_{tot}^E enables emission comparison on event level. The results can be further summarized by the event’s activities. This allows identifying different activities from the process as emission drivers, and is visualized in the OCEAn application using a bar plot. For the second task, object allocation, a more sophisticated framework is proposed.

4.2.1 Object Allocation Framework

In order to perform LCA, emissions have to be aggregated along a product’s lifecycle. This is enabled by the object allocation framework: Given emission data on event and E2O level, they are allocated to objects.

For this, we assume a given non-empty set of objects labeled as *target objects* $\Omega \subseteq O$.

When distributing the event emissions among these target objects, the overall emissions have to be preserved: All emissions associated with the process captured in the event log

² “Emission intensity of diesel urban bus” – source: UBA (<https://www.climateq.io/data/emission-factor/06c42852-3f50-4f31-a86c-503d84b3eef5>)

should be assigned to some object, and no additional emissions should be introduced or counted multiple times.

Definition 4.2 (Valid Object Emissions): Given an OCEL L , a set of target objects $\Omega \subseteq O$ with $\Omega \neq \emptyset$ and $em_{\text{tot}}^E: E \rightarrow \mathbb{R}$, an object emission function $em^O: \Omega \rightarrow \mathbb{R}$ is *valid* if and only if the following invariant holds:

$$\sum_{o \in \Omega} em^O(o) = \sum_{e \in E} em_{\text{tot}}^E(e)$$

Object emissions are composed of two intermediate components:

- Target E2O emissions: E2O emissions that are directly associated with a target object.
- Allocated object emissions: emissions distributed among one or more objects via an allocation rule.

The second is directly obtained from input E2O emissions.

Definition 4.3 (Target E2O Emissions): Let L be an OCEL, $\Omega \subseteq O$ a set of target objects with $\Omega \neq \emptyset$ and $em^{E2O}: E \times O \rightarrow \mathbb{R}$ the E2O emissions over L . For $o \in O$ and $e \in E$, the *target E2O emissions of o in e* are given by

$$em^{E2\Omega}(e, o) := \begin{cases} em^{E2O}(e, o) & \text{if } o \in \Omega \\ 0 & \text{else.} \end{cases}$$

Target E2O emissions form the only possibility how input emissions of the framework are directly connected to a single target object. All remaining emissions are distributed to a set of target objects determined by applying an *allocation rule*. Here, event emissions and (non-target) E2O emissions are no longer distinguished. We denote the remaining event emissions of an event $e \in E$ by $em_{\text{rem}}^E(e) = em^E(e) + \sum_{o \in O \setminus \Omega} em^{E2O}(e, o)$.

Definition 4.4 (Allocated Object Emissions): Given an OCEL L , a set of target objects $\Omega \subseteq O$ with $\Omega \neq \emptyset$, the remaining event emissions $em_{\text{rem}}^E(e): E \rightarrow \mathbb{R}$, and an allocation rule $\alpha: E \rightarrow \mathcal{P}(\Omega) \setminus \{\emptyset\}$, for $o \in O$ and $e \in E$, the *allocated object emissions of o in e* are given by

$$em^\alpha(e, o) := \begin{cases} \frac{em_{\text{rem}}^E(e)}{|\alpha(e)|} & \text{if } o \in \alpha(e) \\ 0 & \text{else.} \end{cases}$$

An object's total emissions are determined by combining its target E2O emissions and the emissions allocated to it by applying the allocation rule α .

Definition 4.5 (Object Emissions): Given an OCEL L , a set of target objects $\Omega \subseteq O$ with $\Omega \neq \emptyset$, and an allocation rule $\alpha: E \rightarrow \mathcal{P}(\Omega) \setminus \{\emptyset\}$, the target object emissions of $o \in \Omega$ are computed as

$$em^O(o) := \sum_{e \in E} (em^{E2\Omega}(e, o) + em^\alpha(e, o))$$

It can easily be shown that this definition always yields valid object emissions:

Proof. Let L be an OCEL, $\Omega \subseteq O$, $\Omega \neq \emptyset$ a set of target objects, $\alpha: E \rightarrow \mathcal{P}(\Omega) \setminus \{\emptyset\}$ an allocation rule, em^E the event emissions and em^{E2O} the E2O emissions. Then for any $e \in E$,

$$\sum_{o \in \Omega} em^\alpha(e, o) = |\alpha(e)| \cdot \frac{em_{\text{rem}}^E(e)}{|\alpha(e)|} = em_{\text{rem}}^E(e),$$

so

$$\begin{aligned} \sum_{o \in \Omega} em^O(o) &= \sum_{o \in \Omega} \sum_{e \in E} (em^{E2O}(e, o) + em^\alpha(e, o)) \\ &= \sum_{e \in E} \left(\sum_{o \in \Omega} em^{E2O}(e, o) + \sum_{o \in \Omega} em^\alpha(e, o) \right) \\ &= \sum_{e \in E} \left(\sum_{o \in \Omega} em^{E2O}(e, o) + em_{\text{rem}}^E(e) \right) \\ &= \sum_{e \in E} \left(\sum_{o \in \Omega} em^{E2O}(e, o) + em^E(e) + \sum_{o \in O \setminus \Omega} em^{E2O}(e, o) \right) \\ &= \sum_{e \in E} \left(em^E(e) + \sum_{o \in O} em^{E2O}(e, o) \right) = \sum_{e \in E} em_{\text{tot}}^E(e) \quad \square \end{aligned}$$

4.2.2 Allocation Rules

The previous section introduced the procedure used to distribute emissions among objects. The set of objects a given event's emissions are assigned to is determined using an allocation rule $\alpha: E \rightarrow \mathcal{P}(\Omega) \setminus \{\emptyset\}$.

This section provides a selection of such allocation rules. To give examples, the rules are applied to some events of the logistics event log (Table 3.1). Emission-driving activities are ignored, assuming each event causes emissions of 1 kg CO₂e, leading to $em_{\text{rem}}^E(e) = 1$ for all events $e \in E$. This is assumed for simplicity and to better demonstrate the impact of structures contained in the OCEL. Emissions are to be allocated to the set of *items*, so

$$\Omega = \{\text{i1}, \text{i2}, \text{i3}, \text{i4}\}.$$

AllTargets

The trivial allocation rule assigns each event's emissions to all target objects, distributing the overall emissions evenly among them.

Definition 4.6 (ALLTARGETS): Given an OCEL L and a set of target objects $\emptyset \neq \Omega \subseteq O$, the ALLTARGETS allocation rule is defined by

$$\alpha_{\text{AT}}(e) := \Omega.$$

In the example, this distributes all emissions evenly among all items, disregarding the set of objects an event is related to. For the first two events, this leads to

$$\begin{aligned} \alpha_{\text{AT}}(\text{receive_o1}) &= \Omega = \{\text{i1}, \text{i2}, \text{i3}, \text{i4}\} & em^{\alpha_{\text{AT}}}(\text{receive_o1}, o) &= \frac{1}{4} \forall o \in \Omega \\ \alpha_{\text{AT}}(\text{get_i1}) &= \Omega = \{\text{i1}, \text{i2}, \text{i3}, \text{i4}\} & em^{\alpha_{\text{AT}}}(\text{get_i1}, o) &= \frac{1}{4} \forall o \in \Omega \end{aligned}$$

ParticipatingTargets

Another rather obvious way to allocate an event's emissions considers those target objects directly participating in the event, distributing the emissions among them. In case no target object participates in the event (like `inspect_v2`), the emissions are distributed among all target objects, like in the `ALLTARGETS` rule.

Definition 4.7 (PARTICIPATINGTARGETS): Given an OCEL L and a set of target objects $\emptyset \neq \Omega \subseteq O$, the PARTICIPATINGTARGETS allocation rule is defined by

$$\alpha_{PT}(e) := \begin{cases} obj_L(e) \cap \Omega & \text{if } obj_L(e) \cap \Omega \neq \emptyset \\ \Omega & \text{else.} \end{cases}$$

Using this, emissions from the logistics process are allocated to those items related to each event, or to all items when there are none (like in `inspect_v2`).

$$\begin{aligned} \alpha_{PT}(\text{load_p3}) &= \{i3, i4\} & em^{\alpha_{PT}}(\text{load_p3}, -) &= (i3 \mapsto \frac{1}{2}, i4 \mapsto \frac{1}{2}) \\ \alpha_{PT}(\text{get_i4}) &= \{i4\} & em^{\alpha_{PT}}(\text{get_i4}, i4) &= 1 \\ \alpha_{PT}(\text{inspect_v2}) &= \Omega = \{i1, i2, i3, i4\} & em^{\alpha_{PT}}(\text{inspect_v2}, o) &= \frac{1}{4} \forall o \in \Omega \end{aligned}$$

ClosestTargets

The previous rule only considers direct E2O relations for allocation. Any events without relation to a target object are allocated to the complete set of target objects, not leveraging any process information further.

The CLOSESTTARGETS rule uses the object interaction graph to derive distances between pairs of objects. Event emissions are then allocated to those target objects that have a minimal distance to an object related to the event.

Definition 4.8 (CLOSESTTARGETS): Given an OCEL L and a set of target objects $\emptyset \neq \Omega \subseteq O$, the CLOSESTTARGETS allocation rule is defined by

$$\alpha_{CT}(e) := \arg \min_{o \in \Omega} \min_{o' \in obj_L(e)} \text{dist}_{OG^\Omega(L)}(o, o').$$

The definition implies the following for any event e : In case e has participating objects reachable from a target object, $\alpha_{CT}(e)$ yields the closest target objects. Otherwise, $\text{dist}(o, o') = \infty$ for all $o \in \Omega$ and $o' \in obj_L(e)$, leading to $\alpha_{CT}(e) = \Omega$.

By definition, the object interaction graph $OG^\Omega(L)$ is constructed by object interactions, i.e., pairs of objects that participate in some shared event. For the logistics process (Table 3.1), this graph is shown in Figure 4.3. As the objects `i2`, `p2` and `v2` are all related to the event `load_p2`, they have pairwise edges. Items that get loaded onto the same pallet are not connected in the graph. This is due to the fact that an extra edge between two target objects does not reduce the distance between a non-target and a closest target object. As items are set as targets, this makes all of them unconnected even though `i3` and `i4` both participate in the event `load_p3`.

Some events, including `receive_o1` and `deliver_p3` are not directly related to any *item*. They are included in Figure 4.3 and connected to their participating objects. Their emissions are allocated as follows:

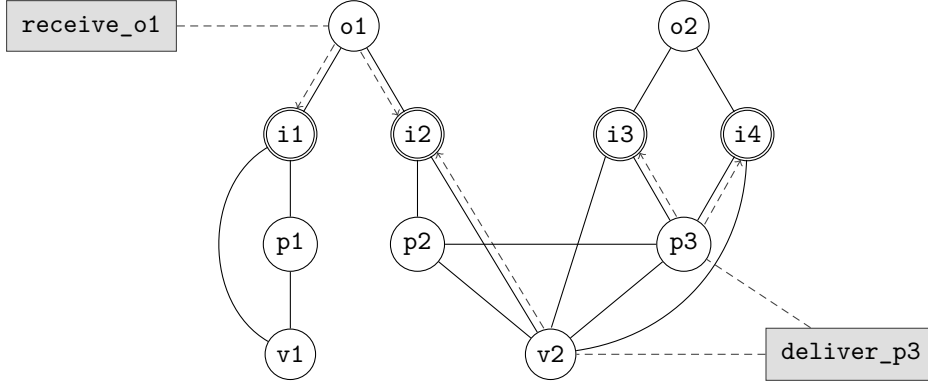


Figure 4.3: Object interaction graph $OG^\Omega(L)$ of the logistics event log. Two objects are connected when they share some event. There are no edges between *item* objects as they are all target objects (double borders). Two events are included as squares and connected to their participating objects, with the paths used for allocation depicted by arrows.

$$\begin{aligned} \alpha_{CT}(\text{receive_o1}) &= \{i1, i2\} & em^{\alpha_{CT}}(\text{receive_o1}, -) &= (i1 \mapsto \frac{1}{2}, i2 \mapsto \frac{1}{2}) \\ \alpha_{CT}(\text{deliver_p3}) &= \{i2, i3, i4\} & em^{\alpha_{CT}}(\text{deliver_p3}, -) &= (i2 \mapsto \frac{1}{3}, i3 \mapsto \frac{1}{3}, i4 \mapsto \frac{1}{3}) \end{aligned}$$

Compared to the previous rules, this allocation rule allocates more event emissions to proper subsets of the target objects. In fact, whenever $\alpha_{PT}(e) \subsetneq \Omega$, CLOSESTTARGETS returns the same result, because the participating target objects are at the minimal distance zero from themselves. In Section 4.2.5, this is proven formally, along with other statements about allocation rule.

Still, CLOSESTTARGETS allocates to large sets of target objects for some events, as seen in the example of `deliver_p3`: As the *item* `i2` is connected to the *vehicle* `v2`, it receives emissions from a transport it is not involved in. In the following, this issue is addressed by deriving a new object interaction graph based on a distinction between object types.

4.2.3 Handling Units and Resources

In an OCEL, objects are the central elements connecting events with each other. As a contrast, in classical event logs, case IDs play a more well-defined role, representing instances of a process and at the same time a sequence of events. While the lifecycle concept can be translated directly to an OCEL, an object is not necessarily equivalent to a process instance. This gets clear when considering a few examples of certain object types in a process:

- In the example logistics process (Table 3.1), an *order*, as well as an *item* or *pallet* is a suitable case notion. *Vehicles* cannot be considered process instances, as their traces will grow large in a real-world dataset.
- In a sales process, a quotation or invoice can be considered a process instance. Products can be included as objects and linked to different quotations even when not translating to tangible objects. Assuming a product is offered permanently, its traces will contain events related to many different instances of that product.

- In a P2P process, an employee can be represented as an object and be related to different kinds of events they execute. However, an employee can handle multiple purchase requisitions at a time. Also, throughout their time at the company, they will handle a lot of purchase requisitions that might not be related at all.

Based on the above examples, we distinguish between

- *Handling units (HUs)*: Objects that can be considered *cases* in the underlying process. A handling unit mostly has a limited lifetime within the process boundary, and may be an input or an output of the process. When two objects interact with the same handling unit in two different events, they are considered related to each other.

Common examples of HUs are orders, invoices, packages, and documents.

- *Resources*: Objects that can **not** be considered *cases* in the underlying process. A resource mostly has a long lifetime or even exist throughout all the event log's timespan. Two objects interacting with the same resource are not considered to be related with each other.

Common examples of resources are machines, employees, departments, and vehicles.

The object's role in the process provides another distinction: A process works *on* HUs and *with* resources. Again considering the logistics example, *orders*, *items* and *pallets* should be considered HUs, while a *vehicle* is a resource as it is worked *with*. This distinction can be made under the assumption that the number of events linked to a vehicle will be high in a larger dataset: The number of vehicles is constant, while each vehicle makes a number of rides each day over a long period of time. As *item* is an HU, two *pallets* containing *items* of the same *order* are considered related, while two *pallets* just loaded into the same *vehicle* are not.

There is no exact definition in form of a quantitative requirement an object type has to satisfy to distinguish between HUs and resources. In the following, this is assumed user input and denoted as O_{HU} and O_{resource} , two sets forming a partition of the set of objects.

4.2.4 Object Interaction Graph Modifications

Section 4.2.2 introduced the CLOSESTTARGETS rule that is using the object interaction graph $OG^{\Omega}(L)$. This graph contains all objects of the event log as vertices, and every two objects that share some common event or an O2O relation are connected by an edge.

As resources have long lifecycles with many events, they also interact with many other objects, leading to a high degree in the object interaction graph. Also, by definition, two objects interacting with the same resource are not considered related with each other. Therefore, a shortest path used for allocation should not contain a resource, unless it is a target object itself.

When using the graph for allocating emissions from the logistics event log (Table 3.1) to the *items*, all items ever loaded to a *vehicle* are at distance 1 from that vehicle. If an event of that vehicle is not directly related to any item, all those items at distance 1 will be assigned the event's emissions. Here, the differentiation between HUs and resources comes

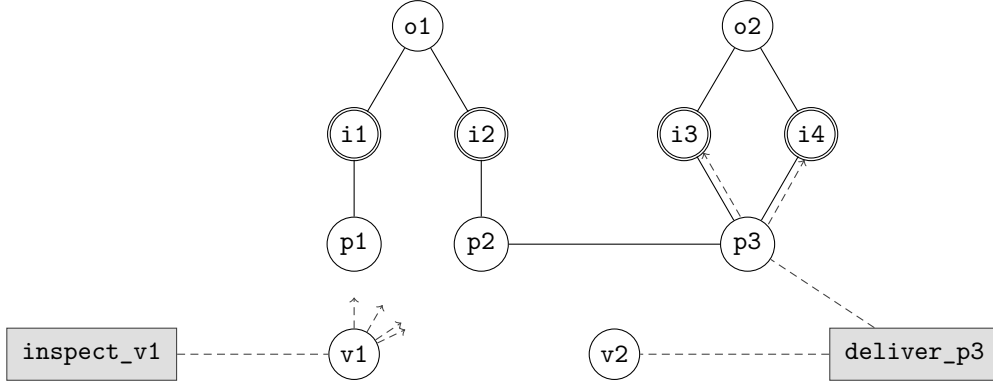


Figure 4.4: HU interaction graph $OG_{\text{HU}}^{\Omega}(L)$ of the logistics event log. *Vehicles* do not have any edges as they are labeled as resources. Allocation paths for two events are highlighted, with inspect_v1 now allocated to all four target objects.

into play. *Vehicles* are resources and *orders*, *items* and *pallets* are HUs. Interactions of *items* and *pallets* should be prioritized over those involving a *vehicle*. This is why another object interaction graph is proposed, just containing HUs.

Definition 4.9 (HU Interaction Graph): For an OCEL L with HUs $O_{\text{HU}} \subseteq O$ and a set of target objects $\Omega \subseteq O$ with $\Omega \neq \emptyset$, the *HU interaction graph* is the undirected graph $OG_{\text{HU}}^{\Omega}(L)$ with

$$\begin{aligned} V(OG_{\text{HU}}^{\Omega}(L)) &= O, \\ E(OG_{\text{HU}}^{\Omega}(L)) &= E(OG^{\Omega}(L)) \cap \mathcal{P}(O_{\text{HU}} \cup \Omega). \end{aligned}$$

The CLOSESTTARGETS allocation based on distances in $OG_{\text{HU}}^{\Omega}(L)$ is denoted by $\alpha_{\text{CT,HU}}$.

Figure 4.4 shows the HU interaction graph of the logistics event log. *Vehicles* are considered resources, therefore they do not have any neighboring vertices. The edge set is reduced to interactions between HUs (orders, items and pallets). Again, there are no edges between *items* according to Definition 3.2, as they are all labeled target objects.

As $p3$ is the only HU the event deliver_p3 is related to, its emissions are allocated differently, this time not including the item $i2$ that belongs to the other order. inspect_v1 relates to no HUs at all, so its emissions are distributed among all *items*:

$$\begin{aligned} \alpha_{\text{CT,HU}}(\text{deliver_p3}) &= \{i3, i4\} & em^{\alpha_{\text{CT,HU}}}(\text{deliver_p3}, -) &= (i3 \mapsto \tfrac{1}{2}, i4 \mapsto \tfrac{1}{2}) \\ \alpha_{\text{CT,HU}}(\text{inspect_v1}) &= \{i1, i2, i3, i4\} & em^{\alpha_{\text{CT,HU}}}(\text{inspect_v1}, o) &= \tfrac{1}{4} \forall o \in \Omega \end{aligned}$$

In the previous example, the *items* are labeled target objects. Therefore, they are mutually unconnected in both graphs (see Definition 3.2). Edges between objects of the same type are only present in *pallets* that are delivered together. In general, the non-target objects participating in the same event always form a clique in $OG^{\Omega}(L)$. In events with many objects, this leads to a high number of edges.

To reduce the edge number, another version of the object interaction graph is introduced, where edges between objects of the same type are always removed, regardless of whether they are target objects or not.

Definition 4.10 (Object Interaction Graphs without Object Type Self-Loops): For an OCEL L with HUs $O_{\text{HU}} \subseteq O$ and a set of target objects $\Omega \subseteq O$ with $\Omega \neq \emptyset$, the *object interaction graph without object type self-loops* is the undirected graph $OG^{\Omega, \times}(L)$ with

$$\begin{aligned} V(OG^{\Omega, \times}(L)) &= O, \\ E(OG^{\Omega, \times}(L)) &= E(OG^{\Omega}(L)) \setminus \left\{ \{o_1, o_2\} \subseteq O \mid \text{objtype}(o_1) = \text{objtype}(o_2) \right\}. \end{aligned}$$

Analogously, the *HU interaction graph without object type self-loops* is the undirected graph $OG_{\text{HU}}^{\Omega, \times}(L)$ with

$$\begin{aligned} V(OG_{\text{HU}}^{\Omega, \times}(L)) &= O, \\ E(OG_{\text{HU}}^{\Omega, \times}(L)) &= E(OG_{\text{HU}}^{\Omega}(L)) \setminus \left\{ \{o_1, o_2\} \subseteq O \mid \text{objtype}(o_1) = \text{objtype}(o_2) \right\}. \end{aligned}$$

The CLOSESTTARGETS allocation rules based on distances in $OG^{\Omega, \times}(L)$ and $OG_{\text{HU}}^{\Omega, \times}(L)$ are denoted by $\alpha_{\text{CT}}^{\times}$ and $\alpha_{\text{CT, HU}}^{\times}$, respectively.

Using these graphs, the question arises whether allocation results are preserved after removing object type self-loops. This is assessed in Chapter 5 using large-scale event logs. Also, the number of edges removed and the runtime reduction in distance computation is analyzed.

4.2.5 Allocation Properties

In order to compare the results obtained from object allocation, appropriate measures are required. These measures are later used for the evaluation of the object allocation framework (Chapter 5). In the following, we assume an OCEL L , a set of target objects $\emptyset \neq \Omega \subseteq O$, and an allocation rule $\alpha: E \rightarrow \mathcal{P}(\Omega) \setminus \{\emptyset\}$.

The intermediate result of object allocation is the set of target objects an event gets allocated to, $\alpha(e) \subseteq \Omega$. The framework then computes a target emission distribution, em^O . The allocation measures only make use of $\alpha(e)$ to provide an isolated evaluation of the allocation rule itself, disregarding the input emissions.

Two extreme cases in object allocation are events that get allocated to all target objects, and those being allocated to exactly one target object. The allocation measures are built around the frequency of these two cases.

Definition 4.11 (Uniform and Unique Allocation): For an OCEL L , $\emptyset \neq \Omega \subseteq O$, an allocation rule $\alpha: E \rightarrow \mathcal{P}(\Omega) \setminus \{\emptyset\}$ and $e \in E$, we write that

- e is allocated *uniformly* using α if $\alpha(e) = \Omega$,
- e is allocated *uniquely* using α if $|\alpha(e)| = 1$.

Based on this,

$$\begin{aligned} f_{\text{uniform}}(\alpha) &:= \frac{1}{|E|} \cdot |\{e \in E \mid \alpha(e) = \Omega\}|, \\ f_{\text{unique}}(\alpha) &:= \frac{1}{|E|} \cdot |\{e \in E \mid |\alpha(e)| = 1\}|. \end{aligned}$$

Note that $f_{\text{uniform}}(\alpha) + f_{\text{unique}}(\alpha) \leq 1$. Equality is not reached in general, as events may be allocated to more than one, but not all target objects. In the following, different statements for the previously defined allocation rules are made using these measures.

Corollary 4.12:

- (a) ALLTARGETS only allocates uniformly, i.e., $f_{\text{uniform}}(\alpha_{\text{AT}}) = 1$.
- (b) PARTICIPATINGTARGETS allocates an event $e \in E$ uniformly if $\text{obj}_L(e) \cap \Omega = \emptyset$.
- (c) If CLOSESTTARGETS allocates an event uniformly, so does PARTICIPATINGTARGETS. As a consequence, $f_{\text{uniform}}(\alpha_{\text{PT}}) \geq f_{\text{uniform}}(\alpha_{\text{CT}}^{OG})$ for every object interaction graph OG .
- (d) If PARTICIPATINGTARGETS does not allocate uniformly, CLOSESTTARGETS allocates to the same target objects.

Proof. (a) and (b) follow directly from Definitions 4.6 and 4.7.

- (c) Let L be an OCEL, $\emptyset \neq \Omega \subseteq O$ a set of target objects, and OG an undirected graph with $V(OG) = O$. Now let $e \in E$ s.t. CLOSESTTARGETS allocates e uniformly, i.e., $\alpha_{\text{CT}}^{OG}(e) = \Omega$. Then by Definition 4.8 there is some $d \in \mathbb{N}_0$ s.t. $d = \min_{o' \in \text{obj}_L(e)} \text{dist}_{OG}(o, o')$ for all $o \in \Omega$.

If $d = 0$, then $\Omega \subseteq \text{obj}_L(e)$. Otherwise, $\Omega \cap \text{obj}_L(e) = \emptyset$. In both cases, Definition 4.7 gives $\alpha_{\text{PT}}(e) = \Omega$, i.e., PARTICIPATINGTARGETS allocates e uniformly. \square

- (d) Let L be an OCEL, $\emptyset \neq \Omega \subseteq O$ a set of target objects, and OG an undirected graph with $V(OG) = O$. Now let $e \in E$ s.t. PARTICIPATINGTARGETS does not allocate e uniformly, i.e., $\alpha_{\text{PT}}(e) \subsetneq \Omega$. Then $\alpha_{\text{PT}}(e) = \text{obj}_L(e) \cap \Omega \neq \emptyset$ from Definition 4.7. As $\text{dist}_{OG}(o, o) = 0$ for all $o \in \text{obj}_L(e) \cap \Omega$ and $\text{dist}_{OG}(o, o') \geq 1$ for all $o \in \Omega \setminus \text{obj}_L(e)$, and $o' \in \text{obj}_L(e)$, Definition 4.8 yields

$$\alpha_{\text{CT}}^{OG}(e) = \text{obj}_L(e) \cap \Omega = \alpha_{\text{PT}}(e). \quad \square$$

Uniform allocation signals that allocation is not able to narrow down the set of target objects for a given event. In contrast to that, unique allocation is the most precise result of allocating an event. Chapter 6 elaborates on this interpretation further.

Chapter 5

Evaluation

In this chapter, the object allocation framework from Section 4.2 is applied to four OCELS available in the OCEL 2.0 format. The evaluation of the allocation rules addresses RG4b by analyzing their allocation results and runtime performance. Furthermore, it assesses influences of event log characteristics on these outcomes. After explaining the evaluation setup (Section 5.1), the input data are summarized (Section 5.2). The results are given in Section 5.3.

5.1 Setup

The evaluation consists of two parts. The first part examines characteristics of the resulting emission distributions. In the second part, the runtime of the CLOSESTTARGETS rule is analyzed. The experiments are conducted on four OCELS which are introduced after defining the setup. The assignments of the datasets' object types to HU or resource are given and justified in the dedicated result sections. Each object type is tested as target object type, with the exception of those types that only have one object, as allocation results are trivial in this case. For an object type $ot \in OT(L)$ with at least two objects, this leads to $\Omega := \{o \in O \mid objtype(o) = ot\}$. The experiment is repeated for each of these target object type settings.

On these input data, different allocation rules are applied: ALLTARGETS (α_{AT}), PARTICIPATINGTARGETS (α_{PT}) and CLOSESTTARGETS (α_{CT}). The CLOSESTTARGETS rule is applied repeatedly, testing all four object interaction graphs defined in Section 4.2.4. Depending on the graph, the rule is abbreviated as follows:

- CT: Full object interaction graph ($OG^\Omega(L)$),
- CTx: Full object interaction graph without object type self-loops ($OG^{\Omega, \times}(L)$),
- CTHU: HU interaction graph ($OG_{HU}^\Omega(L)$),
- CTHUx: HU interaction graph without object type self-loops ($OG_{HU}^{\Omega, \times}(L)$).

The graphs without object type self-loops are created in order to reduce the edge count and thus speed up distance computation. We claim that the allocation results are the same regardless of using an object interaction graph with or without object type self-loops. This

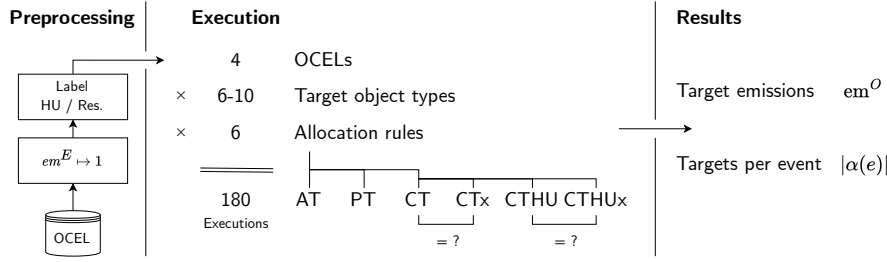


Figure 5.1: Schematic overview of the evaluation procedure. In total, 180 different combinations of input data and parameters are tested. Next to the ALLTARGETS and PARTICIPATINGTARGETS rules, CLOSESTTARGETS is tested with the four different object interaction graphs. The target emissions em^O and number of targets per event $|\alpha(e)|$ are saved as results for further analysis.

claim is to be verified in the experiment for the given OCELS and sets of target object types.

Figure 5.1 shows an overview of this procedure, highlighting the number of different input datasets and execution parameters. The experiment is repeated for every OCEL and for every object type as target object type. The result of each execution is the target object emission distribution and the number of targets per event, additionally containing the distance at which they are found in the selected object interaction graph when using the CLOSESTTARGETS rule.

5.1.1 Target Emission Distribution

The goal of this experiment is to evaluate the allocation rules and the resulting target object emissions and to relate these results to characteristics of the input data. Because of this, the event emissions are set to 1 kg CO₂e per event, with no E2O emissions, leading to $em_{rem}^E(e) = 1 \forall e \in E$. This prevents potential influence of event emission variance on the resulting object emissions.

The allocation results are further processed, computing measures used for comparison. The list of measures is shown in Table 5.1. The variation coefficient (cv) measures the relative variance of target emissions. It is chosen over standard deviation because the emissions per target deviate by magnitudes depending on both the overall emissions (due to different numbers of events), and the number of target objects. Thus, comparability between results for different target object types is guaranteed. The number of target objects an event is allocated to indicates among how many objects the emissions are distributed. Two extreme cases are events that get allocated uniquely (to one target) or uniformly (to all targets), as introduced in Section 4.2.5. These measures are included as well as the median number of targets per event ($|\alpha(e)|_{50}$). For the CLOSESTTARGETS rule, additional measures based on the selected object interaction graph OG are computed. The distances of shortest paths to target objects used for allocation provide information on how target objects are located in the graph. The maximum of these distances (d_{max}^E) is included. Degrees in the object interaction graph differ between the full and HU interaction graph as resources have higher degrees than HUs. This difference is assessed by giving the median and maximum degrees of all objects (\deg_{50}, \deg_{max}) and target objects only ($\deg_{50}^\Omega, \deg_{max}^\Omega$). Lastly, the number of edges ($|E(OG)|$) and connected components ($|C(OG)|$) are given to compare size and connectivity of the different graphs.

Table 5.1: Statistics computed for all allocation results. For allocation with the CLOSESTTARGETS rule, additional statistics based on the object interaction graph OG are computed.

Statistic	Symbol	Rules
Variation coefficient of target object emissions	cv	all
Number of target objects per event (median)	$ \alpha(e) _{50}$	all
Fraction of events uniformly allocated	$f_{\text{uniform}}(\alpha)$	all
Fraction of events uniquely allocated	$f_{\text{unique}}(\alpha)$	all
Distances of shortest paths used for event allocation (max.)	dist_{\max}^E	CT...
Degrees in object interaction graph (med. / max.)	$\text{deg}_{50}, \text{deg}_{\max}$	CT...
Degrees of targets in object interaction graph (med. / max.)	$\text{deg}_{50}^{\Omega}, \text{deg}_{\max}^{\Omega}$	CT...
Number of edges / components in object interaction graph	$ E(OG) , C(OG) $	CT...

5.1.2 Runtime Analysis

The CLOSESTTARGETS allocation rule involves computing distances on an object interaction graph. As opposed to the other allocation rules, the computation time is potentially high for large datasets. In the following, a runtime experiment is conducted, repeatedly executing allocation rules on all input data and configurations from the first experiment. Each configuration is tested three times, capturing the minimum runtime.

As the graph is unweighted and only shortest distances to any target object are requested, distances are computed by means of breadth-first search (BFS). For processing and saving the underlying graph structure, the `networkx` Python library is utilized. A custom BFS algorithm is implemented that allows for early stopping once a target object has been found, except for processing all objects at the same distance. Distances are first computed between pairs of objects. After that, closest targets are looked up for the objects participating in each event, resulting in a set of closest targets per event.

The experiments are conducted on a system running Ubuntu 24.04 with an Intel i7-14700K processor (16 cores: 8x 5.3 GHz, 8x 4.2 GHz) and 32 GB DDR5-6000 MHz CL36 RAM.

5.2 Input Data

To evaluate the allocation technique, different OCEls are used as input data. Given that OCPM is a novel concept, the availability of real-world datasets is limited. Alongside the OCEL 2.0 standard [5], three simulated event logs have been published, based on order management [33], container logistics [34], and Purchase-to-Pay (P2P) [35] processes. Each of these OCEls contains about 10k objects and between 14k and 36k events. The only real-world dataset published with the OCEL 2.0 standard is not considered, as it refers to the history of a git repository which is not considered relevant for emission analysis [36].

Another OCEL used for evaluation has been developed to exemplify the integration of sustainability data, including carbon equivalents of events and objects, into OCEls. This dataset is based on a hinge manufacturing process and is the largest dataset used for evaluation, containing 24k objects and 39k events [37].

Details about the underlying processes and object relations of the individual OCEls are given in dedicated sections of Section 5.3, followed by results specific to that dataset.

Table 5.2: Results of the allocation evaluation. For each target object type setting (OT_Ω), the rules ALLTARGETS, PARTICIPATINGTARGETS, and CLOSESTTARGETS with full (CT) and HU graph (CTHU) are applied. The table includes the variation coefficient of target emissions (cv), the median number of targets per event ($|\alpha(e)|_{50}$), the percentages of events allocated uniquely ($f_{\text{unique}}(\alpha)$) and uniformly ($f_{\text{uniform}}(\alpha)$), and the maximal graph distance used to allocate events (dist_{\max}^E).

L	OT_Ω	$ \Omega $	cv				$ \alpha(e) _{50}$				$f_{\text{unique}}(\alpha)$ [%]				$f_{\text{uniform}}(\alpha)$ [%]				dist_{\max}^E	
			AT	PT	CT	CTHU	AT	PT	CT	CTHU	AT	PT	CT	CTHU	AT	PT	CT	CTHU	CT	CTHU
Ord. Mgmt.	item (HU)	7659	.00	.39	.39	.39	all	1	1	1	0	54	54	54	100	0	0	0	0	0
	ord (HU)	2000	.00	.06	.30	.38	all	all	345	1	0	31	31	84	100	69	0	0	1	1
	pckg (HU)	1128	.00	.04	.20	.37	all	all	526	1	0	18	18	77	100	82	0	0	1	1
	cust (Res.)	15	.00	.02	.02	.09	all	all	all	1	0	19	19	100	100	81	81	0	1	1
	empl (Res.)	18	.00	.60	.60	.71	all	1	1	1	0	73	73	73	100	22	22	0	1	1
	prod (Res.)	20	.00	.06	.06	.06	all	1	1	1	0	54	54	54	100	0	0	0	0	0
Ctr. Log.	Ctr (HU)	1996	.00	.09	.17	.17	all	1	1	1	0	63	93	93	100	35	0	0	2	2
	CO (HU)	600	.00	.00	.38	.38	all	all	6	6	0	3	9	9	100	97	0	0	3	3
	HU (HU)	10552	.00	.00	.14	.17	all	1	1	1	0	60	61	61	100	40	0	0	4	3
	TD (HU)	594	.00	.01	.37	.37	all	all	6	6	0	5	9	9	100	95	0	0	2	2
	Vcl (HU)	127	.00	.03	.39	.46	all	all	1	1	0	7	80	96	100	92	6	2	5	3
	FL (Res.)	3	.00	.00	.01	.01	all	all	2	2	0	22	50	50	100	78	7	7	5	4
	Trk (Res.)	6	.00	.01	.01	.01	all	all	1	1	0	35	93	93	100	65	4	4	3	3
P2P	goods rcpt. (HU)	1941	.00	.12	.19	.19	all	1	1	1	0	57	70	70	100	38	0	0	2	2
	invoice rcpt. (HU)	927	.00	.14	.50	.50	all	all	1	1	0	34	100	100	100	66	0	0	2	2
	material (HU)	1807	.00	.05	.26	.26	all	all	3	3	0	3	27	27	100	90	19	19	2	2
	payment (HU)	927	.00	.04	.50	.50	all	all	1	1	0	8	100	100	100	92	0	0	2	2
	purch. ord (HU)	1598	.00	.10	.19	.19	all	1	1	1	0	54	72	72	100	43	0	0	1	1
	purch. req. (HU)	927	.00	.05	.50	.50	all	all	1	1	0	16	100	100	100	84	0	0	3	3
	quotation (HU)	927	.00	.12	.50	.50	all	all	1	1	0	28	100	100	100	72	0	0	2	2
Hinge	FemalePart (HU)	2915	.00	.00	.00	.03	all	all	all	1	0	23	38	91	100	77	61	0	2	6
	FormedPart (HU)	5911	.00	.02	.05	.05	all	all	2	1	0	46	46	92	100	54	31	0	2	2
	Hinge (HU)	2907	.00	.00	.00	.02	all	all	1	1	0	8	68	98	100	92	31	0	3	7
	HingePack (HU)	290	.00	.00	.00	.02	all	all	all	1	0	1	1	99	100	99	99	0	3	8
	MalePart (HU)	2914	.00	.00	.00	.03	all	all	all	1	0	23	38	91	100	77	61	0	2	6
	SteelCoil (HU)	4	.00	.00	.00	.01	all	all	all	1	0	15	15	99	100	85	85	0	3	4
	SteelPin (HU)	2907	.00	.00	.00	.02	all	all	1	1	0	8	68	98	100	92	32	0	3	7
	SteelSheet (HU)	5911	.00	.00	.05	.05	all	all	1	1	0	46	61	92	100	54	16	0	3	3
	Machine (Res.)	7	.00	.40	.46	.46	all	1	1	1	0	85	85	85	100	15	0	0	2	1
	Workstation (Res.)	3	.00	.80	.80	.80	all	1	1	1	0	100	100	100	100	0	0	0	1	0

5.3 Results

After executing object allocation for all combinations of input data, the first finding is that removing object type self-loops preserves allocation results on all input data tested. This is verified based on the set of closest target objects and the respective distance from each object. Moreover, it is found that the sets of connected components of each object interaction graph get preserved.

This confirms the claim formulated before and at the same time allows for summarizing the results by omitting the CTx and CTHUx configurations, just keeping CT and CTHU.

Target Emission Distribution

The measures computed for all allocation results are given in Table 5.2. It shows the four datasets with each object type selected as target object types with their respective counts, together with the statistics selected for evaluation.

As the ALLTARGETS rule (AT) allocates uniformly across all target objects, it always yields $cv = 0$, 100 % uniformly allocated events, and therefore $|\alpha_{AT}(e)|_{50} = |\Omega|$ and 0 % uniquely allocated events.

The PARTICIPATINGTARGETS rule (PT) allocates those events uniformly that do not have any participating target objects. As this is the case for many events in each dataset, the percentages of uniformly allocated events are high. Few exceptions include the *items* and *products* types in *order management*, as some objects of these types participate in every event, leading to 0 % uniformly allocated events.

Variation in target emissions is highest in the CLOSESTTARGETS rule (CT / CTHU). The number of targets per event is much lower, with the median reaching 1 for most object types, especially in CTHU. Accordingly, the number of events allocated uniquely is high and the number of events allocated uniformly is low, with CTHU allocating no events uniformly for 26 of 30 target object types.

The maximum distance used for allocation in CLOSESTTARGETS shows varied distributions across the four OCELS. In *order management*, the maximum distance is 1, with *Hinge* with CTHU yielding by far the highest distances (up to 8 when allocating to *HingePacks*).

Sections 5.3.1 to 5.3.4 address the individual OCELS, both introducing the dataset and the selected HU-resource partition and summarizing the allocation results specific to that dataset. Thereby RQ6 is addressed by relating characteristics of the OCELS to the results obtained by applying different allocation rules.

Runtime Analysis

For assessing the runtime results, both the graphs with and without object type self-loops are considered again. Figure 5.2 shows the runtime distribution in relation to number of edges and maximum degree in the object interaction graph. Runtime of the CLOSESTTARGETS allocation reaches a maximal value of 4m10s when using the full graph to allocate to *HingePacks* in the *Hinge* OCEL. Overall, the 16 highest runtimes are measured when allocating to the HUs of the *Hinge* OCEL using the full graph. The lowest 28 runtimes are those from the *P2P* OCEL, bounded by 0.5s. Order management has the largest graphs

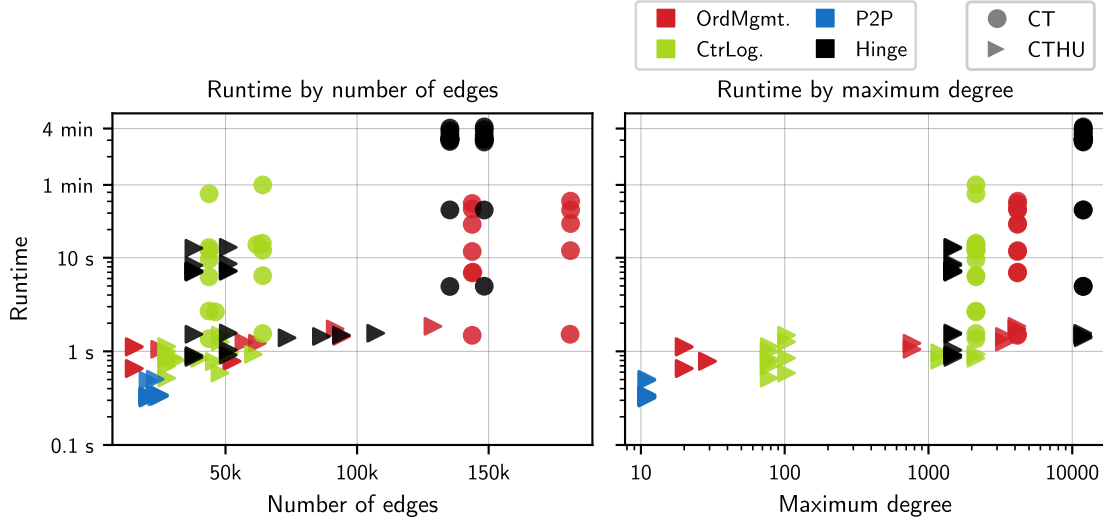


Figure 5.2: Runtime of CLOSESTTARGETS in relation to the number of edges in the object interaction graph (left) and the maximum degree (right). Different colors indicate the OCELS used for evaluation, different marker shapes the two graph versions.

with up to 180k edges in the full object interaction graph. This leads to runtimes up to one minute. Edge counts in container logistics are centered around 50k, with similar runtimes.

Degrees are highest in the hinge OCEL (up to 11k). Among these graphs is one HU interaction graph. The object type causing this outlier is addressed in Section 5.3.4. Every graph containing resources has degrees exceeding 1000.

Figure 5.3 shows the effects of choosing the HU interaction graph over the full graph, as well as removing object type self-loops. The choice of the object interaction graph has significant influence on both the number of edges and allocation runtime. In most runs, execution time drops by over 65 %. The few outliers, including two in which runtime increases, all have an absolute runtime below 2 s. Therefore, this does not change the overall performance. Many runs reach runtime savings close to 100 %. On average, 95.4 % of the runtime is saved (77.8 s in absolute figures).

The P2P OCEL is not shown because all of its object types are HUs, making the HU interaction graph equal to the full object interaction graph. In the other datasets, the edge number of both graphs mostly differ by values between 20 %–85 %.

Both values are also reduced when removing object type self-loops (on both graph types). The differences are not as large as between the two graph types, but equally significant. Runtime gets reduced by at most 32.9 % (median 2.8 %), edge numbers by at most 70.8 % (median 17.4 %). If runtime increases, it does by at most 2.4 %.

Edge number reduction is especially high in HU graphs, as most object type self-loops are edges between HUs. This is due to HUs of the same type appearing more often in the same event than do resources of the same type.

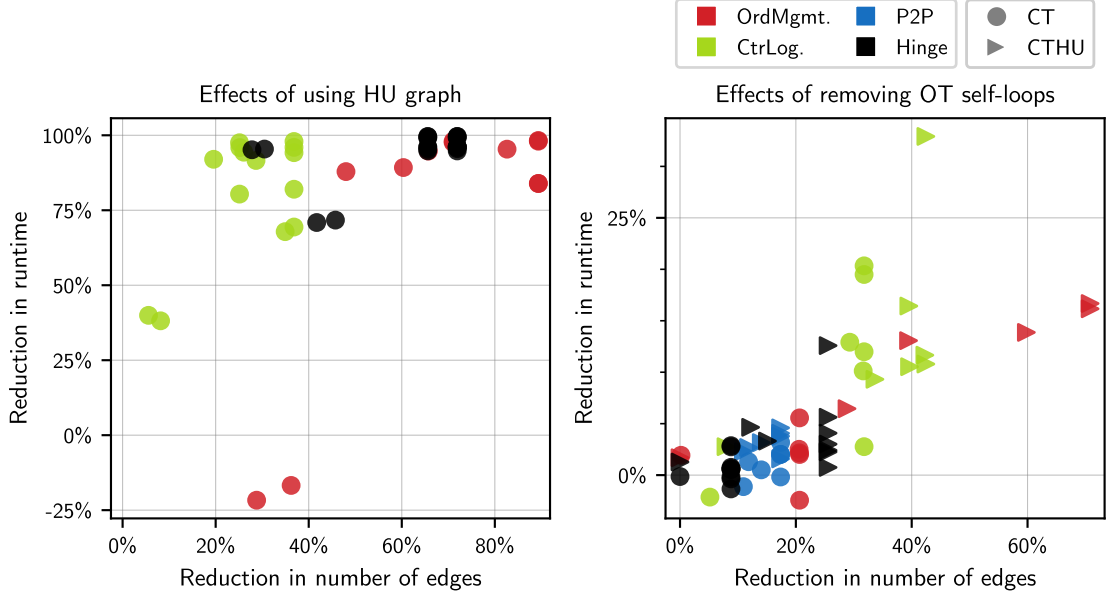


Figure 5.3: Relative reduction in number of edges and runtime when using the HU graph over the full graph (left) and removing OT self-loops (right). OCEls are distinguished by colors, the P2P OCEL is omitted on the left as it only contains HUs.

Table 5.3: Object types of *order management* with their assigned class (HU or Resource), number of events per object (median [min-max]), percentage of events related to at least one object of the type, and number of related objects (degree in $OG^\Omega(L)$)

Object type	Count	Events per object	Ev. covered	Degree	Class
items	7659	7 [7–16]	100 %	25 [9–51]	HU
orders	2000	3 [3–7]	31 %	8 [4–27]	HU
packages	1128	3 [3–10]	18 %	15 [4–37]	HU
products	20	2716 [2439–3009]	100 %	3857 [3530–4174]	Res.
employees	18	470.5 [267–1906]	78 %	2230.5 [1262–3399]	Res.
customers	15	262 [206–304]	19 %	646 [531–792]	Res.

5.3.1 Order Management

The *order management* OCEL [33] contains simulated data from a process that handles incoming orders from customers, from registering them to packing and shipping the packages and receiving payments. A *package* contains one or more *items*, which are the tangible instances of *products*.

Products, as well as *employees* and *customers* are objects that recur within the process and do not enter nor leave the system. Table 5.3 shows their high number of events and objects they interact with, making them resources. *Items* belong to a unique *order* and are packed in a *package*. All of these three object types are case candidates in classical PM, with a maximum trace length of 16 events and 51 interacting objects, and are therefore labeled HUs.

For the *order management* log, Table 5.2 shows that CTHU allocates the majority of

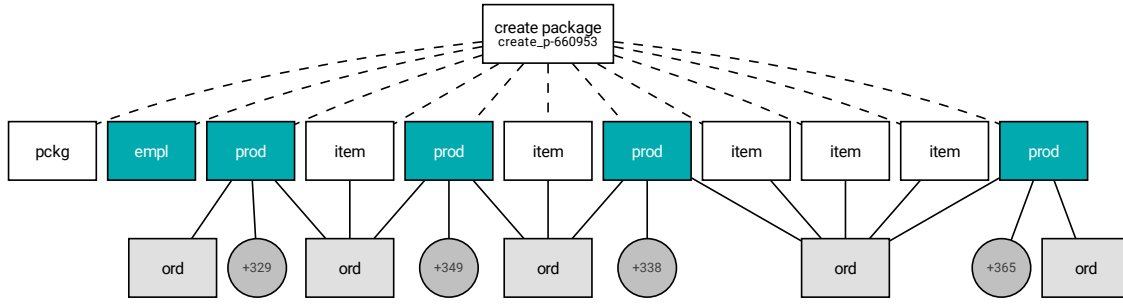


Figure 5.4: Paths in the full object interaction graph leading from objects participating in a *create package* event to *order* target objects (gray). Resources (turquoise) add many additional targets at distance 1.

events uniquely for each target object type setting. *Customers* can uniquely be linked to each event. In *employees*, emission variation is highest, exceeding 0.6 for all non-trivial rules. Only one other object type across all OCEs yields higher variation. This can be explained by different *employee* roles (sales, shipment, and warehousing). Based on this attribute, an *employee* takes part in a different number of events. A warehousing *employee* takes part in at least 1766 events, the rest in at most 513 events. When dividing allocated emissions by the number of events per object, variation is only 0.05.

In all target types, events are allocated over distances of at most 1. This is explained by the fact that at least one *item* participates in every event, and each item is linked to some object of every other type.

The *order management* OCE has the largest full object interaction graphs. Also, edge reduction is highest with values up to 89.4 % when using the HU graph over the full graph. The few resource objects are mostly linked to many HUs (e.g., each product has over 3500 items), while each HU is related to at most 20 objects per type in the HU graph.

Figure 5.4 depicts how CLOSESTTARGETS allocates emissions to *order* target objects. Using the HU interaction graph, only three orders are found at distance 1 via *item* objects. Using the full object interaction graph, also resources like *products* are considered. Each product linked to the event adds more than 300 additional orders to the set of target objects at distance 1.

5.3.2 Container Logistics

The *container logistics* OCE [34] describes a process of a logistics company that ships goods using containers. The goods are picked up at a production site and transferred to a terminal from where they are further transported by different company.

Customer orders and *transport documents* are artifacts documenting the process from registering an order, ordering *containers*, transporting and storing *handling units* in the terminal, to booking and finally loading *vehicles*. The process involves the use of *forklifts* and *trucks*. These are resources with long lifecycles and many related objects (see Table 5.4). The aforementioned object types *customer order*, *transport document*, *container*

Table 5.4: Object types of *container logistics* with their assigned class (HU or Resource), number of events per object (median [min-max]), percentage of events related to at least one object of the type, and number of related objects (degree in $OG^\Omega(L)$)

Object type	Count	Events per object	Ev. covered	Degree	Class
Handl. Unit	10 552	2 [1–2]	60 %	2 [1–2]	HU
Container	1996	14 [1–14]	65 %	34 [2–65]	HU
Cust. Order	600	2 [1–2]	3 %	1	HU
Transp. Doc.	594	4 [2–8]	5 %	30 [1–103]	HU
Vehicle	127	21 [4–53]	8 %	26 [5–59]	HU
Truck	6	2084.5 [2038–2139]	35 %	2085.5 [2039–2140]	Res.
Forklift	3	2565 [2546–2610]	22 %	1193 [1180–1199]	Res.

and *handling unit*¹ are case-like types with shorter lifecycles and therefore considered HUs.

The *vehicle* object type shows the ambiguity of the HU-resource distinction. *Vehicles* are trains operated by a third-party company used for further transport. Judging by the quantities from Table 5.4, it significantly falls inbetween those object types labeled as HUs and resources, both regarding the object count and median number of events per object. Labeling them as resources is an intuitive choice based on the fact that vehicles usually have a long lifetime and are rather *used* by a process than being *processed*. However, the degrees of *vehicles* in $OG^\Omega(L)$ are rather small and similar to those of other HUs. Also, a *vehicle’s* lifespan in the process is limited as it has a fixed departure and the number of *containers* it can load is bounded. Based on these arguments, *vehicles* are labeled HUs.

5.3.3 Procure-to-Pay (P2P)

The Procure-to-Pay (P2P) OCEL [35] is the smallest of the four datasets (14671 events, 9054 objects). It is motivated by real-world procurement processes that use SAP software. It contains *purchase requisitions* getting created and approved, *quotations* being created and sent to a vendor, up to *invoices* and *payments* being processed. All objects in the OCEL are considered HUs: All have short lifecycles (maximum 11 events per object, see Table 5.5), and low degrees in the object interaction graph (maximum 11).

In addition to not containing resources, the P2P OCEL exhibits a very regular structure. The numbers of four of the seven object types are equal (927), also being the number of components in the object interaction graph, with each component containing exactly one object of the types in question. Components are small (at most 16 objects). This makes BFS fast, with the runtime bounded by 0.5s. Events are always allocated uniquely by CLOSESTTARGETS to those target types unique to components. For the other types, no uniform allocation is observed, except for *material* (19 % of events). *materials* are only present in 21 % of the connected components, all other object types in 100 %.

¹*Handling unit* here refers to the name of an object type in the *container logistics* OCEL [34]. At the same time, this object type is labeled as handling unit (HU), the concept introduced in Section 4.2.3.

Table 5.5: Object types of *P2P*, all labeled as HU, together with the number of events per object (median [min-max]), percentage of events related to at least one object of the type, and number of related objects (degree in $OG^\Omega(L)$)

Object type	Count	Events per object	Ev. covered	Degree	Class
goods rcpt.	1941	5 [4–9]	62 %	6 [3–9]	HU
material	1807	2	10 %	5 [4–7]	HU
purch. order	1598	6 [4–9]	57 %	8 [4–11]	HU
invoice rcpt.	927	5 [3–11]	34 %	5 [3–9]	HU
payment	927	1 [1–5]	8 %	5 [3–9]	HU
purch. req.	927	3 [1–3]	16 %	3 [1–5]	HU
quotation	927	3 [3–9]	28 %	4 [2–9]	HU

Table 5.6: Object types of *Hinge* with their assigned class (HU or Resource), number of events per object (median [min-max]), percentage of events related to at least one object of the type, and number of related objects (degree in $OG^\Omega(L)$)

Object type	Count	Events per object	Ev. covered	Degree	Class
FormedPart	5911	3 [2–3]	46 %	7 [4–7]	HU
SteelSheet	5911	3	46 %	6	HU
FemalePart	2915	3 [2–3]	23 %	9 [4–9]	HU
MalePart	2914	3 [2–3]	23 %	9 [4–9]	HU
Hinge	2907	2 [1–2]	8 %	16 [5–16]	HU
SteelPin	2907	1	8 %	5	HU
HingePack	290	1	1 %	12	HU
SteelCoil	4	1478 [1477–1478]	15 %	1480 [1479–1480]	HU
Machine	7	5911 [290–5911]	85 %	5916 [3191–11 823]	Res.
Workstation	3	11 658 [3197–23 644]	100 %	11 831 [11 661–11 921]	Res.
Facility	1	29	0 %	4	Res.
Worker	1	5858	15 %	5831	Res.

5.3.4 Hinge Manufacturing

The *Hinge* [37] OCEL is the largest dataset, with 38528 events and 23771 objects. It simulates a hinge manufacturing process, where *hinges* are produced from the raw material, a *steel coil*, that is cut into *steel sheets* and then further processed into *female* and *male parts*. After assembling, the hinges are packed into *hinge packs*, which is the process output. All aforementioned object types have short traces (Table 5.6), making them HUs. The *steel coil* type is not a common HU example judging by quantities. *Steel coils* are split into over 1400 *steel sheets* each, leading to a high number of events and a high degree. Still, the relation between these two types is considered meaningful as it represents input and output of a production step, and each *steel sheet* is only related to exactly one *steel coil*. Therefore, *steel coil* is labeled an HU, too. Other object types like *Machine*, *WorkStation* and *Worker* are prototypical examples of resources, with many events and high object interaction graph degrees. As there is only one object of the types *Facility* and *Worker*, these objects are not tested as target objects in the experiment – allocation would be trivial.

Figure 5.5 shows a summarized version of the HU interaction graph of the hinge OCEL.

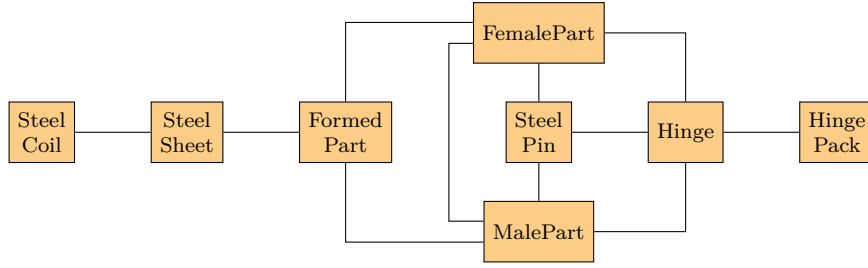


Figure 5.5: HU interaction graph of the *Hinge* OCEL, summarized by object types. Two object types are connected when any object pair of the respective types has an edge in $OG_{\text{HU}}^{\Omega}(L)$. The visualization shows that objects of the types *SteelCoil* and *HingePack* have at least a distance of 5 in $OG_{\text{HU}}^{\Omega}(L)$.

Objects are grouped by type, connecting two types if any object pair of those types has an edge in $OG_{\text{HU}}^{\Omega}(L)$. This graph exhibits the structure of interactions in the log, showing the production process: The input (a steel coil) gets transformed to steel sheets, then female and male parts, and finally hinges that are packed to hinge packs. This also explains the high path distances shown in Table 5.2: The highest value (8) is reached in allocation to hinge packs, aligning with the fact that they represent one end of this process. Combined with the fact that the hinge OCEL is the largest among the four tested, the high distances also explain the high runtimes visible in Figure 5.2.

Chapter 6

Discussion

The evaluation of the object allocation framework gives insights into various quantities extracted from target emission distribution and the number of target objects the events are allocated to. This amount is summarized by counting the number of events allocated to exactly one target (uniquely) or all targets (uniformly).

Unique and Uniform Allocation

Uniform allocation leads to distributing an event’s emissions evenly among all target objects. We claim that in most cases, this is not desirable: Object allocation is intended to identify the target objects closest related to each event, and uniform allocation means that the allocation rule is not capable of narrowing down the set of related targets. Assuming an HU type is chosen as target object type, these object types typically exhibit a high count and short object lifetimes. This leads to many non-overlapping target lifespans within the event log. This assumption is contradictory to all targets getting assigned the emissions of a single event.

This picture changes when regarding emissions that are caused by the use of resources. This scenario is of high relevance as process resources like machines and vehicles are considered main emission drivers [3]. Whenever a resource emission is not closely linked to a target HU or to no HU at all, the long resource lifetime justifies allocating this emission to many or even all target objects. This is for example the case in regularly scheduled events like a vehicle that needs to undergo inspection (see example from Section 1.1).

The experiment, however, considers constant emissions per event, and it is found that in the four OCEs, each event is linked to at least one HU. This nullifies the aforementioned scenario for the scope of this evaluation, in which uniform allocation is, therefore, considered undesirable. Moreover, the ALLTARGETS rule allocates every event uniformly ($f_{\text{uniform}}(\alpha_{\text{AT}}) = 1$, Corollary 4.12 (a)), ignoring all information contained in the OCE. Thus, its use is discouraged, and the rule is considered a minimal example demonstrating trivial allocation.

Comparing the other two allocation rules, PARTICIPATINGTARGETS and CLOSESTTARGETS, Corollary 4.12 (c) states that CLOSESTTARGETS always allocates fewer or equally many events uniformly. This also becomes visible in Table 5.2. Additionally, any event not allocated uniformly by PARTICIPATINGTARGETS is allocated to the same set of targets by

CLOSESTTARGETS, no matter what object interaction graph is used (Corollary 4.12 (d)). All in all, this means that if the allocation of PARTICIPATINGTARGETS is desirable, so is the one of CLOSESTTARGETS, no matter what object interaction graph is chosen. Note that this only implies that CLOSESTTARGETS is at least as powerful as PARTICIPATINGTARGETS. It does not prove that results from CLOSESTTARGETS are always desirable. Still, we recommend always choosing CLOSESTTARGETS over PARTICIPATINGTARGETS.

As an opposite to uniform allocation, unique allocation signals that a single connection between the event in question and a target object is found. This is desirable for multiple reasons: First, it shows that the allocation rule is capable of narrowing down the target set to a maximum extent. Second, the emissions are assigned as a whole and no distribution is needed. Considering approaches different from evenly splitting the emissions is a possible extension of the current framework. For example, attributes of the target objects could be used as a weighting factor. In a transport of multiple packages, the individual emission shares are usually determined by the packages' weights and individual transport distances [38]. This is currently achieved by specifying an E2O emission rule using those attributes. As opposed to object allocation, this requires a direct E2O relation. Moreover, it has to be performed during emission estimation and cannot be used to apportion existing event emission values.

Still, unique allocation is not always the desired result. In the aforementioned package example, multiple packages are the correct allocation result. Therefore, $f_{\text{unique}}(\alpha) = 1$ does *not* necessarily indicate an optimal result.

HU Interaction Graph vs. Full Object Interaction Graph

With CLOSESTTARGETS identified as the recommended rule among the three presented, the implications of using different object interaction graphs are now assessed.

First, the experiment shows that the HU interaction graph has significantly fewer edges as compared to the full version. Among the OCEs featuring resources (all but P2P), 97 278 edges are saved (median), corresponding to 65.6 % of all edges. Values observed range from 5.6 %–89.4 %. Figure 5.3 shows edge number reduction in relation to runtime savings. There is a clear correlation as BFS runtime depends on graph size. What also changes upon limiting the graph to HUs is the degree of objects. This is due to resources usually interacting with significantly more objects than do HUs. The maximum degree in HU interaction graphs is on average 4847.5 lower than in the corresponding full graph. The degrees determine to how many vertices BFS traverses, scaling exponentially by path distance. In total, runtime in HU graphs is lower by up to 99.6 % (3m56s), 93.1 % (11s) on median.

The allocation results also show why the HU graph is to be preferred: Uniform allocation drops further, occurring in only 4 of 30 target object types (in 14 object types using the full graph, 27 using PARTICIPATINGTARGETS). Apart from these measures, interactions between HUs are considered more meaningful than those involving resources. Allocating emissions using a path containing a resource thus makes use of weaker object relations that do not make objects part of the same process execution. This becomes especially clear in production processes, like the one portrayed in the Hinge OCE: When a *SteelSheet* interacts with a *FormedPart*, this means the *FormedPart* is produced from the *SteelSheet*. Therefore, upstream emissions stemming from producing the *SteelSheet* are rightfully al-

located to a *FormedPart*. This is not the case in interactions with resources like *Machines*: A machine producing both a *SteelSheet* and a *FormedPart* does not make them connected in any way.

The distance-1 paths in the order management log (Figure 5.4) support this argument. Here, *products* are resources corresponding to *items* as their tangible instances. It is not logical to allocate emissions to other orders which contain items of the same product. In total, the use of the HU interaction graph over the full object interaction graph is recommended for three reasons: First, it captures more logical connections between objects. Second, results are more precise, allocating fewer events uniformly. Third, runtime drops significantly.

Removing Object Type Self-Loops

Another option to reduce graph size has been proposed by removing edges between objects of the same type, called OT self-loops. The experiment shows that on the data tested, allocation results are fully preserved. Edge numbers hereby drop by up to 70.8% (median 17.4%), runtime by up to 11.7s (32.9%), 0.08s (2.8%) on median. While this influence is much lower than the one introduced by the HU graph, it does further optimize performance.

Though results are not affected on the data tested, we are not able to prove this in general. One can easily construct counterexamples, for example by making an object interaction graph consist only of OT self-loops. However, this implies that each event’s participating objects are of the same type. In the four OCELS tested, each event is related to objects of at least two types. Assuming this property holds and a shortest path now contains one OT loop from shared participation in event e , there is always a path longer by just 1 using an object of a different type that also participates in e . This illustrates how the influence of the adjustment is bounded, but still does not suffice to prove full result preservation.

Emission Estimation

As opposed to the object allocation framework, rule-based emission estimation is not included in the experiment. Accuracy of emission rules is influenced by many uncertainties, including quality of the input data from OCEL attributes, and the choice of the emission factor. Here, the OCEAn application relies on the user’s domain and sustainability knowledge. The concept of emission rules itself is designed to be domain-agnostic. Its only limitation is to exclusively use emission factors linear to the input data, an assumption supported by all major emission factor databases [16], with the word *factor* already hinting towards linearity.

Emission rules can be applied on event level and on E2O level. The only difference is that the E2O rules allow counting objects and using attribute values of multiple objects. Towards object allocation, E2O emission rules bind emissions to an object, ensuring unique allocation in case the object is chosen as target. When exporting emission data to the OCEL 2.0 format, E2O emissions are aggregated on event level. This is due to OCEL 2.0 not supporting E2O attributes. When again importing such data, the above linkage of E2O emissions to a potential target object is lost.

The framework does not support rule-based emission computation on object level. For adding upstream emissions along a supply chain, emissions have to be assigned to a certain activity, e.g., corresponding to ordering a certain product. This practice only works when there is a one-to-one correspondence between the objects and the chosen activity, i.e., the activity appears exactly once within the object’s lifecycle. In order to drop this prerequisite and directly attach the emission to the object, an artificial activity can be added. This is however currently not supported by the OCEAn application.

Limitations and Extensions

Data-driven emission computation has the inherent limitation of being restricted to the scope of the data available. In PM, this is usually a certain business process. In order to perform corporate CA, this is a major issue as businesses need to report their overall emissions [2]. Therefore, employing OCEAn only partially performs corporate CA.

Possible extensions of emission rules include the discovery of interval durations. This allows for employing time-based emission factors, for example for machines with constant energy consumptions, relating to TDABC principles that have already been combined with process mining [20, 21].

In object allocation, certain scenarios require approaches not solvable by the current framework. This includes events only linked to resources, for example the inspection of vehicles mentioned before. If some vehicle requires more carbon-intensive inspections than others, it may be desirable to allocate these emissions only to targets interacting with the vehicle – to this end, the full object interaction graph is required. However, other emissions are more logically allocated via the HU interaction graph. A modified allocation framework that executes multiple allocation rules after each other is capable of satisfying both conditions, gradually allocating events based on different conditions.

In order to compute shortest distances between objects, an undirected object interaction graph is used. However, directed graphs are also applicable, with a different outcome: Modified emission rules then could only allow passing the emissions in one direction. The hinge production process serves as an application example for this: As stated before, many object interactions here correspond to production steps. This is clearly a directed relation, as one object is produced from the other. The TOTeM model [30] is capable of extracting this information from the event log.

Choosing directed graphs allows for an entirely different notion of object carbon footprints. In the production example, it is logical to assign each (intermediate) product a footprint, accumulating emissions along the process (i.e., with increasing event timestamps). This contradicts the allocation invariant (Definition 4.2), raising the need for a different allocation framework. However, this idea is not applicable in all processes and object types. For example, an *order* should be assigned all downstream emissions from package delivery. These examples indicate that a universal solution to the object allocation problem does not exist.

Chapter 7

OCEAn: Object-Centric Emission Analysis Tool

This chapter provides a description of the OCEAn application. This includes implementation details and information on the libraries used (Section 7.1) and a typical user journey (Section 7.2).

7.1 Architecture

The OCEAn application is a web-based program consisting of two parts: A python backend for importing and exporting OCEL files and executing all computations for emission estimation and object allocation, and a TypeScript [39] frontend based on React¹.

The backend provides a wrapper class for pm4py [40] OCEs offering various statistics and transformations, such as the discovery of object interactions for building the object interaction graph.

The frontend accesses backend functions by means of an API developed with FastAPI². This library offers data validation based on pydantic [41] and thus ensures type safety in all parameters passed through the API. Additional sanity checks are integrated as custom pydantic validators and assertions.

The application is designed to handle large datasets. All computations involving OCEL data are implemented using vectorized operations with the `pandas` [42, 43] package. As the OCEL import is potentially a long-running, the imported OCEL is always kept in server memory during a session. This makes the API stateful, allowing for further performance optimizations. Caching is used for repeated calls to the same function. To enable caching for functions with list or set parameters, the `cachetools`³ library is used, facilitating customization of function argument hashing as opposed to python's built-in caching functionality.

¹<https://react.dev/>

²<https://fastapi.tiangolo.com/>

³<https://github.com/tkem/cachetools/>

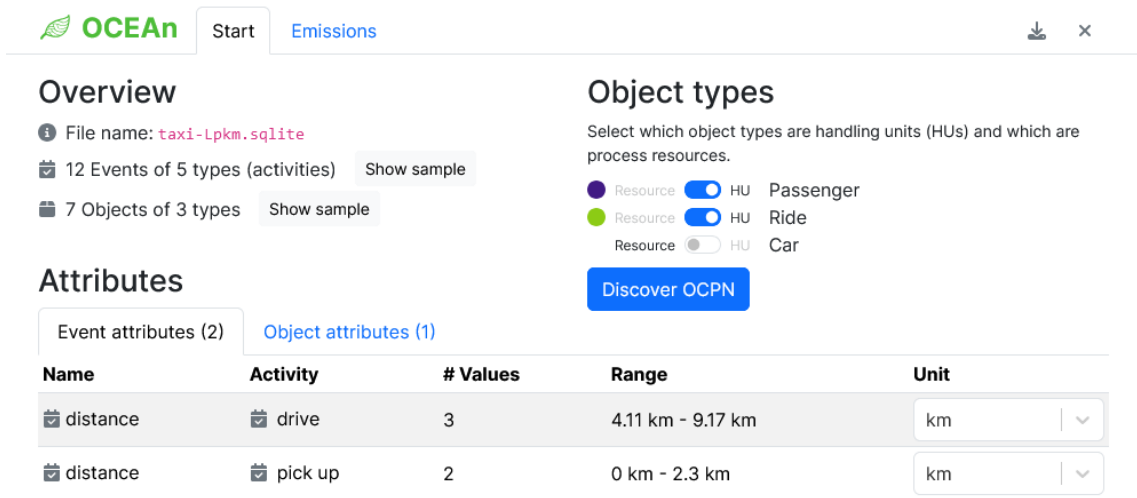


Figure 7.1: Dashboard of the OCEAn app. On the top left, basic information about the OCEL is shown. On the bottom, the available attributes are listed and assigned a unit. On the top right, the HU/resource partition is defined.

Allocation graphs are handled using `networkx` [44] data structures. However, as the distance computation for the CLOSESTTARGETS rule requires finding not only one, but all closest target objects, a custom BFS implementation is used, as stated in Figure 5.1.

The OCEAn application is publicly accessible on GitHub ⁴.

7.2 User Journey

When opening the application, the user is first offered a file upload for event logs in the OCEL 2.0 sqlite format. Alternatively, some example logs are provided for quick access, including the logistics example OCEL from Section 3.2.1 and four OCEs later used for evaluation (Chapter 5).

After loading an OCEL into the application, the main dashboard (Figure 7.1) is shown. The page consists of multiple sections: The *Overview* section contains basic information about the dataset, including the number of events and objects and its filename. Additional metadata like a source URL can be included here.

In the *Attributes* section, the event and object attributes contained in the OCEL are listed. Basic statistics such as number of distinct values of categorical attributes and minimum and maximum of numerical attributes are offered. For numerical attributes, a unit selection is available. As OCEs do not natively contain information about the unit of quantities represented as attributes, this has to be added manually by the user. This enables automatic unit conversion when applying emission rules. To this end, the `pint` python library is employed. When defining emission rules, a sanity check is performed to ensure the multiplication result is always a weight (e.g., kg CO₂e), prohibiting erroneous selection of attributes or units.

⁴<https://github.com/rwth-pads/ocean>

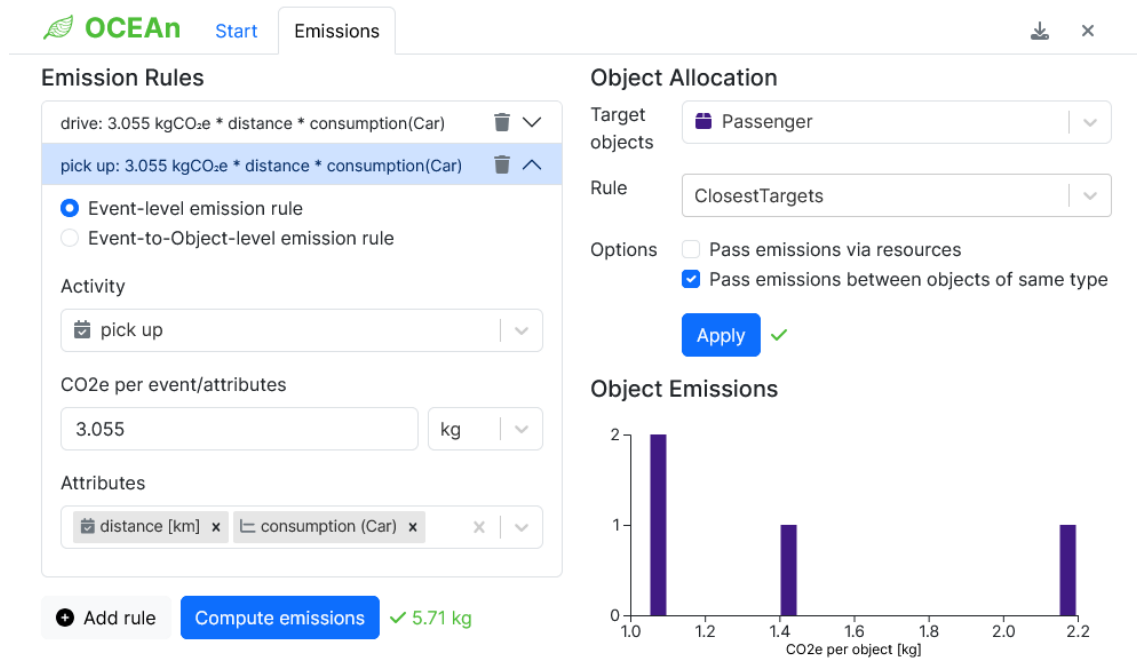


Figure 7.2: Emissions page of the OCEAn app. On the left, emission rules are defined specifying an activity and an emission factor linked to attributes. On the right, object allocation parameters are controlled. Below, the resulting target object emissions are shown in a histogram.

The *Object types* section offers controls to partition the object types into sets of HUs and resources. This is necessary in order to configure the CLOSESTTARGETS rule for object allocation. Object types are always displayed with an icon that is colored in different hues for HUs and gray for resources. HU colors are automatically generated every time the set of HUs changes. Colors are generated with hues distributed equidistantly and randomized brightness and saturation values.

In the header, the *Emissions* tab can be accessed. Also, a download button allows exporting the OCEL with newly integrated data. In case emission data has already been computed on both event and object level, the user is prompted on what level to include these data. This is because the overall emission sum has to be conserved. Apart from event emissions, the export includes all user inputs made in the app, like attribute units, emission rules and the HU/resource partition. These data are integrated into the OCEL 2.0 *sqlite* file by adding an additional table. When importing an OCEL, the settings are loaded in order to increase usability and session portability.

The *Emissions* tab (Figure 7.2) includes an interface for adding and editing emission rules. Both event and E2O emission rules are supported. An activity (and object type and qualifier in the case of E2O rules) is selected and an emission factor is introduced. The user can select one or multiple attributes linked to the respective activity or its unique object types. After emission computation is triggered, the overall emissions are shown to the user.

As soon as emission data are available, the object allocation interface is shown on the right side. Here, all parameters like the allocation rule and additional configuration of the CLOSESTTARGETS rule are set. Target objects can be chosen, with the selection limited to

object types for usability reasons. In practice, this means that $\Omega = \{o \in O \mid objtype(o) = ot\}$ for the selected object type $ot \in OT(L)$. After running allocation, the resulting object emission distribution is shown in a histogram.

Chapter 8

Conclusion

In this thesis, we explored the potential of OCPM in CA, motivated by the growing need for organizations to report their GHG emissions. As a novel approach, OCPM offers a more comprehensive analysis of processes involving multiple types of objects, addressing key limitations of classical PM.

Our contributions are both theoretical and practical. On a theoretical level, emissions are integrated into an OCEL in two steps: First, we propose a framework employing emission rules capable of estimating event and E2O emissions based on hand-selected emission factors. Second, allocation to events and objects is applied in order to create various emission perspectives and gain sustainability-related insights. Allocation to objects hereby determines carbon footprints along an object’s lifecycle, aligning with LCA principles. To this end, three different allocation rules are proposed.

As a practical contribution, we developed the OCEAn web application, which serves as a frontend tool enabling users to apply these methods in practice. OCEAn allows for the annotation of event logs with emission data, facilitates the execution of different allocation rules, and the export of enriched OCEL files for further use.

After evaluating the emission rules w.r.t. runtime and different characteristics of the resulting emission distribution, a clear recommendation is deduced: Only the graph-based approach is capable of correctly allocating emissions to objects, at the cost of high runtime when interactions of process resources are considered. With the distinction of resources and HUs, runtime is lowered by 93.1% (11s, median). In addition, more logical object relationships are captured, and allocation is more precise. Pruning of the object interaction graph by removing edges between objects of the same type is suggested, reducing runtime further, while results are fully preserved on the test datasets.

For future directions, emission estimation can be extended by additional techniques such as the discovery of event intervals for the use in time-based emission rules. Apart from that, sustainability and domain knowledge is needed for defining emission rules. Future research should address this limitation, for example by automatic emission factor recommendation.

Evaluation shows the importance of the distinction of resources and HUs. This partition, currently defined manually, exhibits clear patterns in terms of the lifecycle lengths and

degrees in the object interaction graph. However, we have identified outliers. Therefore, automating this setting is desirable for deployment in industry, but needs to be thoroughly addressed.

The object allocation framework can be equipped with further rules using directed graphs, allowing for footprint annotation of intermediate products in manufacturing. Other enhancements could include the sequential execution of multiple allocation rules, making the framework more robust to different OCEL characteristics. Furthermore, the option of using object attributes for weighted emission distribution has been discussed.

In summary, this work lays the groundwork for applying OCPM to emission assessments and carbon accounting. It provides both theoretical tools and a practical implementation that allows enriching event data with emission values. Next to direct utilization for CA and LCA, providing emission-enriched datasets supports further research in the application of PM for sustainability.

Bibliography

- [1] Wil M. P. van der Aalst. Object-Centric Process Mining: Dealing with Divergence and Convergence in Event Data. In *SEFM*, pages 3–25, Oslo, Norway, 2019. doi:10.1007/978-3-030-30446-1_1.
- [2] Lars Brehm, Jessica Slamka, and Andreas Nickmann. Process Mining for Carbon Accounting: An Analysis of Requirements and Potentials. In *Digitalization Across Organizational Levels: New Frontiers for Information Systems Research*, Progress in IS, pages 209–244. Springer International Publishing, Cham, 2022. doi:10.1007/978-3-031-06543-9_9.
- [3] Nina Graves, István Koren, and Wil M. P. van der Aalst. ReThink Your Processes! A Review of Process Mining for Sustainability. In *ICT4S*, pages 164–175, Rennes, France, 2023. doi:10.1109/ICT4S58814.2023.00025.
- [4] Christian Ortmeier, Nadja Henningsen, Adrian Langer, Alexander Reiswich, Alexander Karl, and Christoph Herrmann. Framework for the integration of Process Mining into Life Cycle Assessment. *Procedia CIRP*, 98:163–168, 2021. doi:10.1016/j.procir.2021.01.024.
- [5] Alessandro Berti, István Koren, Jan Niklas Adams, Benedikt Knopp, Majid Raffei, Nina Graves, Gyunam Park, Yisong Zhang, Leah Tacke genannt Unterberg, Viki Peeva, Marco Pegoraro, and Wil M. P. van der Aalst. OCEL (Object-Centric Event Logs) 2.0 Specification. 2023.
- [6] Dirk Fahland. Process Mining over Multiple Behavioral Dimensions with Event Knowledge Graphs. In *Process Mining Handbook*, volume 448 of *Lecture Notes in Business Information Processing*, pages 274–319. Springer International Publishing, Cham, 2022. doi:10.1007/978-3-031-08848-3_9.
- [7] *The Greenhouse Gas Protocol: A Corporate Accounting and Reporting Standard*. World Resources Institute and World Business Council for Sustainable Development, o.O., 2004.
- [8] *Global Logistics Emissions Council Framework For Logistics Emissions Accounting and Reporting, V3.0*. Smart Freight Center, 2023.
- [9] Stefan Schaltegger and Maria Csutora. Carbon accounting for sustainability and management. Status quo and challenges. *Journal of Cleaner Production*, 36:1–16, 2012. doi:10.1016/j.jclepro.2012.06.024.

-
- [10] Steven J. Davis and Ken Caldeira. Consumption-based accounting of CO₂ emissions. *Proceedings of the National Academy of Sciences of the United States of America*, 107(12):5687–5692, 2010. doi:10.1073/pnas.0906974107.
- [11] Konstantin Stadler, Richard Wood, Tatyana Bulavskaya, Carl-Johan Södersten, Moana Simas, Sarah Schmidt, Arkaitz Usubiaga, José Acosta-Fernández, Jeroen Kuenen, Martin Bruckner, Stefan Giljum, Stephan Lutter, Stefano Merciai, Jan-nick H. Schmidt, Michaela C. Theurl, Christoph Plutzar, Thomas Kastner, Nina Eisenmenger, Karl-Heinz Erb, Arjan de Koning, and Arnold Tukker. EX-IOBASE 3: Developing a Time Series of Detailed Environmentally Extended Multi-Regional Input-Output Tables. *Journal of Industrial Ecology*, 22(3):502–515, 2018. doi:10.1111/jiec.12715.
- [12] Marcel P. Timmer, Erik Dietzenbacher, Bart Los, Robert Stehrer, and Gaaitzen J. de Vries. An Illustrated User Guide to the World Input–Output Database: the Case of Global Automotive Production. *Review of International Economics*, 23(3):575–605, 2015. doi:10.1111/roie.12178.
- [13] ecoinvent. The ecoinvent database, 2024. URL: <https://ecoinvent.org/>. Accessed: Sep 11th, 2024.
- [14] Department for Energy Security & Net Zero. Greenhouse Gas Reporting: Conversion Factors 2024, 2024. URL: <https://www.gov.uk/government/publications/greenhouse-gas-reporting-conversion-factors-2024>. Accessed: Sep 11th, 2024.
- [15] Jan Recker, Michael Rosemann, and Ehsan Roohi Gohar. Measuring the Carbon Footprint of Business Processes. In *BPM Workshops*, pages 511–520, Hoboken, NJ, USA, 2010. doi:10.1007/978-3-642-20511-8_47.
- [16] climatiq - API for Carbon Footprint Calculations, 2024. URL: <https://www.climatiq.io>.
- [17] Andreas Fritsch, Lara Bauer, Fiona Dittrich, Dominic Graf, Karsten Schischke, and Christoph Teusch. Scope 3 Data in Electronics Supply Chains: How to Address Common Challenges and Pitfalls. 2023.
- [18] Wen-Hsien Tsai, Yu-Shan Shen, Pei-Ling Lee, Hui-Chiao Chen, Lopin Kuo, and Chi-Chou Huang. Integrating information about the cost of carbon through activity-based costing. *Journal of Cleaner Production*, 36:102–111, 2012. doi:10.1016/j.jclepro.2012.02.024.
- [19] Robert S. Kaplan and Steven R. Anderson. Time-driven activity-based costing. *Harvard Business Review*, 82(11):131–138+150, 2004.
- [20] Michal Halaška and Roman Šperka. TDABC and Estimation of Time Drivers Using Process Mining. In *Agents and Multi-Agent Systems: Technologies and Applications 2021*, volume 241 of *Smart Innovation, Systems and Technologies*, pages 489–499. Springer Singapore, Singapore, 2021. doi:10.1007/978-981-16-2994-5_41.
- [21] Michael Riesener, Christian Dolle, Alexander Menges, and Günther Schuh. Time-driven Activity-based Costing Using Process Data Mining. In *TEMSCON-EUR*, pages 1–6, Dubrovnik, Croatia, 2021. doi:10.1109/TEMSCON-EUR52034.2021.9488654.

- [22] Moe Thandar Wynn, Wei Zhe Low, Arthur H. M. ter Hofstede, and Wiebe Nauta. A framework for cost-aware process management: cost reporting and cost prediction. *Journal of Universal Computer Science*, 20(3):406–430, 2014.
- [23] Celonis SE. Celonis’ New Material Emissions App Uses the Power of Process Intelligence to Simplify Scope 3 Reporting, 2023. URL: <https://www.celonis.com/press/celonis-new-material-emissions-app-uses-the-power-of-process-intelligence-to-simplify-scope-3-reporting/>. Accessed: Sep 11th, 2024.
- [24] Wil M. P. van der Aalst. Twin Transitions Powered By Event Data: Using Object-Centric Process Mining To Make Processes Digital and Sustainable. In *PN4TT*, Caparica, Portugal, 2023.
- [25] Andreas Berzl. Building a product to reduce emissions with every process step, 2022. URL: <https://www.celonis.com/blog/building-a-product-to-reduce-emissions-with-every-process-step/>. Accessed: Sep 11th, 2024.
- [26] Julia Horsthofer-Rauch, Sarah Ranjana Guesken, Jonas Weich, Alexander Rauch, Maik Bittner, Julia Schulz, and Michael F. Zaeh. Sustainability-integrated value stream mapping with process mining. *Production & Manufacturing Research*, 12(1), 2024. doi:10.1080/21693277.2024.2334294.
- [27] Wil M. P. van der Aalst and Alessandro Berti. Discovering Object-Centric Petri Nets. Preprint of a paper to be published in *Fundamenta Informaticae*, 2020.
- [28] Anahita Farhang Ghahfarokhi, Gyunam Park, Alessandro Berti, and Wil M. P. van der Aalst. OCEL Standard. 2020.
- [29] Jan Niklas Adams, Daniel Schuster, Seth Schmitz, Günther Schuh, and Wil M. P. van der Aalst. Defining Cases and Variants for Object-Centric Event Data. In *ICPM*, pages 128–135, Bolzano, Italy, 2022. doi:10.1109/ICPM57379.2022.9980730.
- [30] Lukas Liss, Jan Niklas Adams, and Wil M. P. van der Aalst. TOTeM: Temporal Object Type Model for Object-Centric Process Mining. In *BPM Forum*, pages 107–123, Krakow, Poland, 2024. doi:10.1007/978-3-031-70418-5_7.
- [31] Wil M. P. van der Aalst. *Process mining: Data science in action*. Springer, Berlin and Heidelberg and New York and Dordrecht and London, 2016. doi:10.1007/978-3-662-49851-4.
- [32] Behrang Shirizadeh, Aurélien Ailleret, Clément Cartry, Sébastien Douguet, Torben Gehring, Sezin Maden, Bjoern Mais, Lennart Mross, Julian Theis, Clément Cabot, Manuel Villavicencio, and Johannes Trüby. Climate neutrality in European heavy-duty road transport: How to decarbonise trucks and buses in less than 30 years? *Energy Conversion and Management*, 309:118438, 2024. doi:10.1016/j.enconman.2024.118438.
- [33] Benedikt Knopp and Wil M. P. van der Aalst. Order Management Object-centric Event Log in OCEL 2.0 Standard, 2023. doi:10.5281/zenodo.8428112.
- [34] Benedikt Knopp, Nina Graves, and Wil M. P. van der Aalst. Container Logistics Object-centric Event Log, 2023. doi:10.5281/zenodo.8428084.

-
- [35] Gyunam Park and Leah Tacke genannt Unterberg. Procure-To-Payment (P2P) Object-centric Event Log in OCEL 2.0 Standard, 2023. doi:10.5281/zenodo.8412920.
 - [36] Marco Pegoraro and Wil M. P. van der Aalst. Angular GitHub Commits Object-centric Event Log, 2023. doi:10.5281/zenodo.8430332.
 - [37] Marco Heinisch and Nina Graves. sOCEL 2.0: A Sustainability-Enriched OCEL of a Hinge Production Process, 2024. doi:10.5281/zenodo.13638680.
 - [38] Åse Jevinger and Jan A. Persson. Consignment-level allocations of carbon emissions in road freight transport. *Transportation Research Part D: Transport and Environment*, 48:298–315, 2016. doi:10.1016/j.trd.2016.08.001.
 - [39] Gavin Bierman, Martín Abadi, and Mads Torgersen. Understanding TypeScript. In *ECOOP*, pages 257–281, Uppsala, Sweden, 2014. doi:10.1007/978-3-662-44202-9_11.
 - [40] Alessandro Berti, Sebastiaan J. van Zelst, and Daniel Schuster. PM4Py: A process mining library for Python. *Software Impacts*, 17:100556, 2023. doi:10.1016/j.simpa.2023.100556.
 - [41] Samuel Colvin, Eric Jolibois, Hasan Ramezani, Adrian Garcia Badaracco, Terrence Dorsey, David Montague, Serge Matveenko, Marcelo Trylesinski, Sydney Runkle, David Hewitt, and Alex Hall. Pydantic, 2024. URL: <https://docs.pydantic.dev/latest/>.
 - [42] The pandas development team. pandas-dev/pandas: Pandas, 2024. URL: <https://doi.org/10.5281/zenodo.3509134>.
 - [43] Wes McKinney. Data Structures for Statistical Computing in Python. In *SciPy*, pages 56–61, Austin, TX, USA, 2010. doi:10.25080/Majora-92bf1922-00a.
 - [44] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring Network Structure, Dynamics, and Function using NetworkX. In *SciPy*, pages 11–15, Pasadena, CA, USA, 2008. doi:10.25080/TCWV9851.

Acknowledgments

After rather randomly taking the IDS course four years ago, my master's studies have been steered in a totally new direction. This is why I would like to thank the entire PADS team for their great teaching in the past years and catching my interest in process mining and data science.

Shout-out to my supervisor, Nina Graves, for her incredible support, 27 constructive yet fun meetings, a lot of helpful guidance, and for cracking some good and some bad jokes.

Special thanks to Eddy, Pascal, Michelle and Till for proofreading, and to Jannes for offering his computer for my evaluation.

Finally, I want to express my gratitude to my family, friends and my boyfriend for their unwavering encouragement and support.