Data underlying the paper: Using agent-based simulation for emergent behavior detection in cyber-physical systems

Rob Bemthuis[1,*,†], **Martijn Mes**[2,‡], **Maria-Eugenia Iacob**[2,‡], **and Paul Havinga**[1,‡]

[1]Pervasive Systems, University of Twente, Enschede, The Netherlands
[2]Industrial Engineering and Business Information Systems, University of Twente, Enschede, The Netherlands
[*]Corresponding author: r.h.bemthuis@utwente.nl
[†]Role: Creator
[‡]Role: Contributor

ABSTRACT

The dataset contains a collection of experiment results and event logs generated. The experiments comprise a logistics case study involving the transport of products that are subject to quality depreciation. The products are transported on smart pallets, which enables us to keep track of various measures (e.g., location, quality depreciation level, etc.). We considered the problem as a special case of the dynamic pickup and delivery problem in which objects need to be transported between an origin and a destination. The case study is implemented in a discrete-event simulation model. This dataset contains the experiment input (i.e., input parameters) and results (i.e., simulation output files and filtered output files considering the removal of a warm-up period). As output, we also provide event logs, which could, for example, be used for model verification or process mining purposes.

Dateset submitted: 8th of June 2021; Data paper submitted: 8th of June 2021.

ERRATUM

Not applicable.

PAPER ABSTRACT

Traditional modeling approaches, based on predefined business logic, offer little support for today's complex environments. In this paper, we propose a conceptual agent-based simulation framework to help not only discover complex business processes but also to analyze and learn from emergent behavior arising in cyber-physical systems. Techniques originating from agent-based modeling as well as from the process mining discipline are used to reinforce agent-based decision-making. Whereas agent-technology is used to orchestrate the integration and relationship between the environment and business logic activities, process mining capabilities are mainly used to discover and analyze emergent behavior. Using a functional decomposition approach, we specified three agent types: cyber-physical controller agent, business rule management agent, and emergent behavior detection agent. We use agent-based simulation of a logistics cold chain case study to demonstrate the feasibility of our approach.

CONCISE CASE DESCRIPTION

The logistics case study involves the movement of cargo within a production facility (Bemthuis et al., 2020). The plant layout (see Figure 1) involves three regions. Products arrive in the first region and need to be transported to a climate-controlled environment (the second region). In the second region, the products are undergoing processing steps. Once a product is completed in the second region, it needs to be transported to the climate-controlled trucks, which takes place in the third region.

The case study is mainly about the movements and decisions of the fleet of cargo transporters. The fleet comprises a heterogeneous fleet of cargo transporters with fixed capacities (Bemthuis et al., 2020). More precisely, three types of vehicles are considered: unmanned aerial vehicles (UAVs), human-driven forklifts (HDFs), and automated guided vehicles (AGVs). Each vehicle has its own properties (e.g., speed, capacity, etc.), as further described in the paper of Bemthuis et al. (2020). Vehicles only operate in the first and third regions.

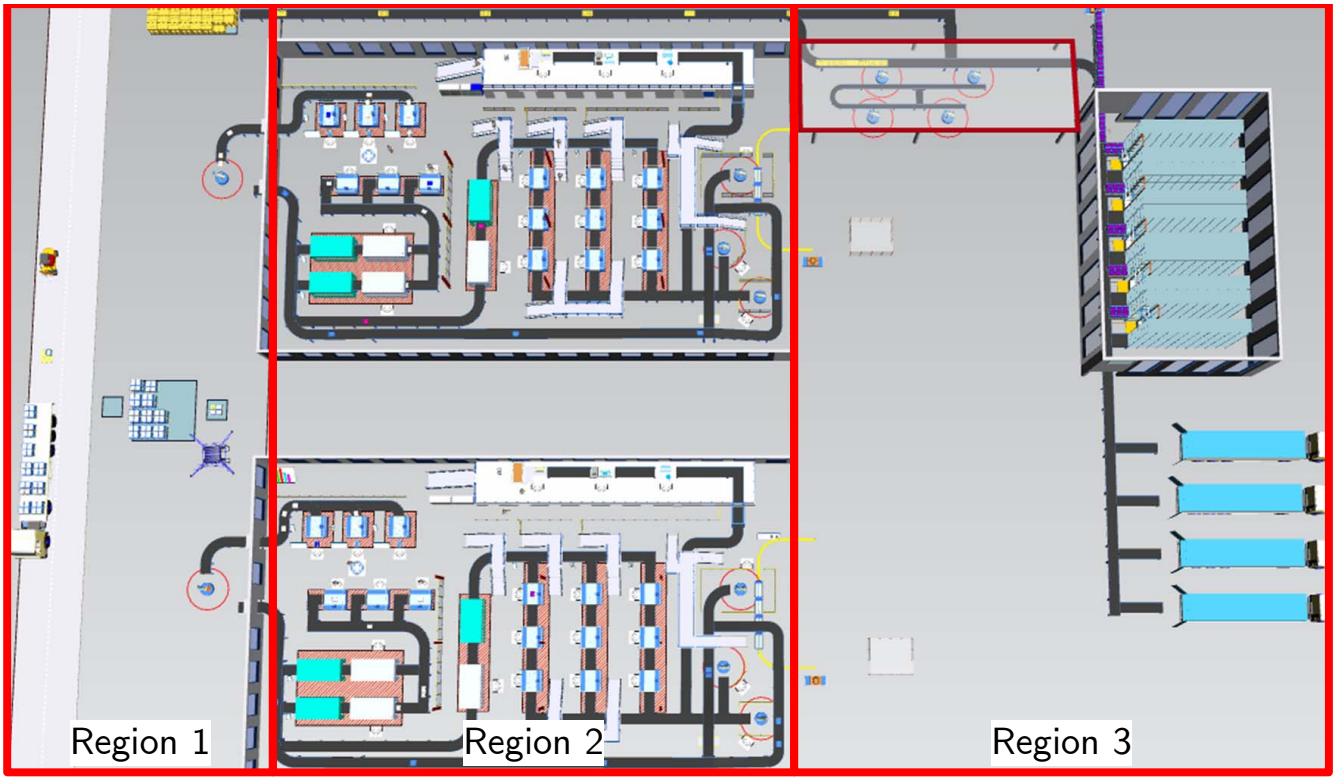

Figure 1. High-level overview of the logistics case study (Bemthuis et al., 2020).

The problem concerns finding suitable configurations for minimizing quality decay of the products against minimizing operating costs. Furthermore, the management of the plant is interested in unveiling insights into event data collected during the lifetime of a product (Bemthuis et al., 2020). To this end, each product is transported via a smart pallet, which is able to obtain information about the state of the cargo. A conceptual blueprint of the simulation study including, e.g., modeling assumptions and simplifications is given in the research paper of Bemthuis et al. (2020).

FILE STRUCTURE

The dataset files are captured in five folders (one input and four output). As output, we distinguish between, on the one hand, raw simulation output and event logs, and, on the other hand, with and without a warm-up period filtering. The total uncompressed file size is roughly 15GB (compressed it is roughly 2GB). The overall data structure, which we describe in more detail below, is as follows:

- Input
  - *expSettings.txt*: gives experiment numbers and the corresponding settings. In total, there are 27 experiments conducted (in the dataset structure, we denote an experiment with X) and 20 runs (denoted with Y) per experiment.
  - *businessRules{X}.txt*: describes the input business rules per experiment (this info is also given in the *expSettings.txt* file).
  - *productSettings{X}.txt*: describes the product characteristics.
  - *vehicleSettings{X}.txt*: describes the vehicle characteristics.
- Output
  - *Exp{X}Run{Y}.txt*: gives raw/unprocessed simulation output per product instantiated.
- OutputWarmupFilter

  - *DisposedProducts{X}Run{Y}.txt*: lists disposed products (due to violating the quality depreciation business rule).
  - *Exp{X}Run{Y}.txt*: gives simulation output per product instantiated, after removing a warm-up period (the first two hours) and the last hour (for statistical data collection purposes).

- LogFiles
  - *Exp{X}Run{Y}.txt*: contains event records, following a convention of a typical event log format (i.e., timestamp, activity, case, resource, etc.).

- LogFilesProductWarmupFilter
  - *Exp{X}Run{Y}.txt*: contains event records after removing a warm-up period and the last hour.

- *experimentResults.xlsx*: summarizes per experiment the quality decays, cycle times, and number of products disposed per hour.

Input - expSettings.txt

Table 1 shows the inputs per experiment, as also addressed in the *expSettings.txt* file.

Table 1. Experiment settings

Experiment number	Vehicles*	Dispatching rules		Decay threshold value
		Vehicle-initiated	Product-initiated	
1	3UAVs;1HDF;1AGV	Random	Random	0.60
2	3UAVs;1HDF;1AGV	Random	Lowest utilization	0.60
3	3UAVs;1HDF;1AGV	Random	Shortest travel distance	0.60
4	3UAVs;1HDF;1AGV	Lowest quality decay	Lowest utilization	0.60
5	3UAVs;1HDF;1AGV	Lowest quality decay	Shortest travel distance	0.60
6	3UAVs;1HDF;1AGV	Highest quality decay	Lowest utilization	0.60
7	3UAVs;1HDF;1AGV	Highest quality decay	Shortest travel distance	0.60
8	3UAVs;1HDF;1AGV	Lowest quality decay	Random	0.60
9	3UAVs;1HDF;1AGV	Highest quality decay	Random	0.60
10	3UAVs;2HDFs;2AGVs	Random	Random	0.60
11	3UAVs;2HDFs;2AGVs	Random	Lowest utilization	0.60
12	3UAVs;2HDFs;2AGVs	Random	Shortest travel distance	0.60
13	3UAVs;2HDFs;2AGVs	Lowest quality decay	Lowest utilization	0.60
14	3UAVs;2HDFs;2AGVs	Lowest quality decay	Shortest travel distance	0.60
15	3UAVs;2HDFs;2AGVs	Highest quality decay	Lowest utilization	0.60
16	3UAVs;2HDFs;2AGVs	Highest quality decay	Shortest travel distance	0.60
17	3UAVs;2HDFs;2AGVs	Lowest quality decay	Random	0.60
18	3UAVs;2HDFs;2AGVs	Highest quality decay	Random	0.60
19	2HDFs;2AGVs	Random	Random	0.60
20	2HDFs;2AGVs	Random	Lowest utilization	0.60
21	2HDFs;2AGVs	Random	Shortest travel distance	0.60
22	2HDFs;2AGVs	Lowest quality decay	Lowest utilization	0.60
23	2HDFs;2AGVs	Lowest quality decay	Shortest travel distance	0.60
24	2HDFs;2AGVs	Highest quality decay	Lowest utilization	0.60
25	2HDFs;2AGVs	Highest quality decay	Shortest travel distance	0.60
26	2HDFs;2AGVs	Lowest quality decay	Random	0.60
27	2HDFs;2AGVs	Highest quality decay	Random	0.60

*:per region, for example experiment 19 concerns 2 HDFs and 2 AGVs in region 1 and 2 HDFs and 2 AGVs in region 3.

Input - businessRules{X}.txt

This file describes in words which vehicle- and product-initiated dispatching rules are used, as well as which threshold value is used (0.60 for all cases). The threshold value indicates when a product will be thrown away automatically, given that the product is still waiting to be assigned to a vehicle in region one.

Input - productSettings{X}.txt

This file describes product characteristics (see also Table 1 in Bemthuis et al. (2020)).

Input - vehicleSettings{X}.txt

This file describes vehicle characteristics (see also Table 1 in Bemthuis et al. (2020)).

Output - Exp{X}Run{Y}.txt

Each row concerns a unique instantiated product and the columns are as follows:

- productIDString = unique identifier of the product;
- productType = product type;
- productNr = product identifier of the product type;
- creationTime = time of creation of a product (i.e., when a product enters the buffer station at region one for the first time);
- productInitCallRegion1 = time when a product calls for transport in region one (i.e., firing the product-initiated dispatching rule) in region one;
- assignedToVehicleRegion1 = time when a product is assigned to a transporter (i.e., after the product-initiated or vehicle-initiated dispatching rule is triggered);
- pickedUpRegion1 = time when a product is picked up at a waiting station in region one;
- waitingTimeRegion1 = waiting time from the call for transport to being picked up in region one;
- droppedOffTimeRegion2 = dropped off time in region two;
- travelTimeRegion1 = total travel time in region one;
- qualityDecayUntillPickedUpRegion1 = quality decay level until a product is picked up in region one;
- qualityDecayUntillDroppedOffRegion2 = quality decay level until a product is dropped off in region two;
- startTimeProcessingRegion2 = start time of processing a product in region two;
- endTimeProcessingRegion2 = end time of processing a product in region two;
- processingTimeRegion2 = processing time of a product in region two;
- productInitCallRegion3 = time when a product calls for transport in region three;
- assignedToVehicleRegion3 = time when a product is assigned to a transporter in region three;
- pickedUpTimeRegion3 = time when a product is picked up at a waiting station in region three;
- waitingTimeRegion3 = waiting time from the call for transport to being picked up in region three;
- droppedOffTimeRegion3 = dropped off time in region three;
- travelTimeRegion3 = total travel time in region three;
- qualityDecayUntillPickedUpRegion3 = quality decay level until a product is picked up in region three;
- qualityDecayUntillDroppedOffRegion3 = quality decay level until a product is dropped off in region three;

- totalWaitingTimeRegions = waiting time in region one plus waiting time in region three;
- totalTravelTimeRegions = travel time in region one plus travel time in region three;
- totalQualityDecayRegions = quality decay level until a product is dropped off in region three;
- qualityBelowMarginTime = time indicating when a product's quality decay will reach the threshold (we used this to initiate an event when the quality decay threshold is reached);
- productDisposed = boolean indicating if a product is disposed or not (due to reaching the quality decay threshold);
- totalTimeInSystem = cycle time of a product;
- totalTimeInSystemInSeconds = a replicate of *totalTimeInSystem*;
- vehicleTypePickedUpRegion1 = vehicle that picked up the product at region one (may contain invalid data points);
- vehiclePickedUpRegion1 = vehicle that picked up the product at region one;
- vehicleTypePickedUpRegion3 = vehicle that picked up the product at region three (may contain invalid data points);
- vehiclePickedUpRegion3 = vehicle that picked up the product at region three.

Notice that it may happen that a data output value of a particular column (e.g., "vehicleTypePickedUpRegion1") is invalid (e.g., "ERROR DATA TYPE!"). This has to do with how the statistics were collected during the simulation run. The column next to it (e.g., "vehiclePickedUpRegion1") contains the same value. So, we recommend using these values only.

OutputWarmupFilter - DisposedProducts{X}Run{Y}.txt

These files list all products that are thrown away because of reaching the quality decay threshold before being assigned to a transporter in region one.

OutputWarmupFilter - Exp{X}Run{Y}.txt

These files consider the "Output - Exp{X}Run{Y}.txt" file but with the removal of a warm-up period and the last hour. That is, we deleted all rows that are (1) created before the warm-up period and (2) created one hour before the end of the run. In addition, we removed products from the list that were thrown away because of the low-quality level.

LogFiles - Exp{X}Run{Y}.txt

The discrete-event simulation model stored the occurrence of certain events in event record files. Each event record at least contains a timestamp, activity, and characteristics regarding the vehicle and/or product. The following details are kept track of (insofar relevant):

- uniqueID = unique identifier for the event record;
- productID = identifier of a product;
- productIDStr = identifier of a product (similar to *productID*, but does not contain void values);
- productType = product type;
- productNr = product identifier of the product type;
- event = description of something that happened;
- timeStamp = moment of occurrence;
- vehicleType = vehicle type;
- vehicle = vehicle identifier of the vehicle type;
- currentDecayLevel = quality decay level of a product;

- processingStation = involved process station.

Notice that it may happen that a data output value of a particular column (e.g., "productID") is invalid (e.g., "(?)"). This has to do with how the statistics were collected during the simulation run. The column next to it (e.g., "productIDStr") contains the same value. So, we recommend using these values only. Furthermore, as we store different types of events, e.g., some are vehicle-specific (involving no product) and some are product-specific (involving no vehicle), not all data records contain the same information.

LogFilesProductWarmupFilter - Exp{X}Run{Y}.txt

Similar to *OutputWarmupFilter - Exp{X}Run{Y}.txt*, these log files consider the removal of a warm-up period and the last hour. Furthermore, we only considered the event records that contained a product case. For the column description, we refer to the description given above.

EXAMPLE

This part describes an example of using an event log file to construct a process model. To this end, we consider the first iteration as described in Section 6.2.2. of Bemthuis et al. (2020). This step involves experiment number 10 (see Table 1) run 1 only. For demonstration purposes, we used the software Disco. The example is as follows:

1. Open *Exp10Run1.txt* (within the *LogFilesProductWarmupFilter* directory) in the Disco tool.

2. Check whether the "productIDStr" column is selected as "Case", the "event" column as "Activity", and the "timeStamp" column as "Timestamp". You may leave the other columns as default.

3. Press the "Start import" button.

4. Figure 2 shows a mined model (Disco uses a fuzzy miner). You could tune the model (e.g., change the level of detail) or have a look at other performance metrics and statistics. Figure 3 shows the model used in the paper of Bemthuis et al. (2020).

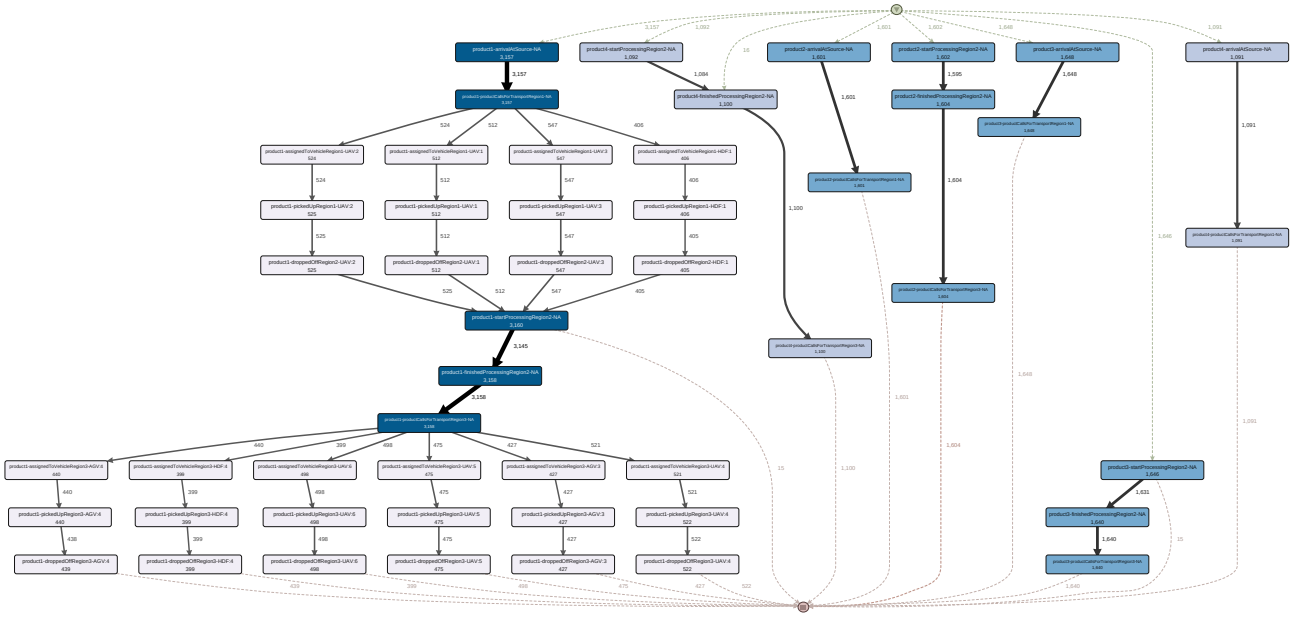

Figure 2. Discovered process mining model of experiment 10 run 1, using the Disco software. The numbers show the absolute frequency of events.

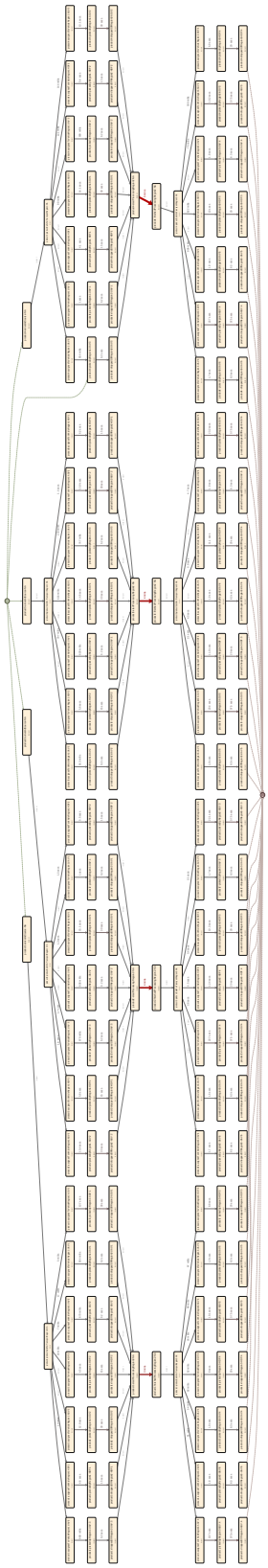

Figure 3. Process mined model used in Bemthuis et al. (2020) for their first iteration. The values indicate the mean duration.

PAPER REFERENCE

Bemthuis, R., Mes, M., Iacob, M.-E., & Havinga, P. (2020). "Using Agent-Based Simulation for Emergent Behavior Detection in Cyber-Physical Systems". In *2020 Winter Simulation Conference (WSC)*, pp. 230-241. DOI: 10.1109/WSC48552.2020.9383956.

DATASET REFERENCE

Bemthuis, R., Mes, M., Iacob, M.-E. & Havinga, P. (2020). "Data underlying the paper: Using agent-based simulation for emergent behavior detection in cyber-physical systems". *4TU.Centre for Research Data*. DOI: 10.4121/14743263.

ACKNOWLEDGEMENTS

This project is funded by the Netherlands Organization for Scientific Research (NWO) (under grant 628.009.015). The authors thank all partners of the DataRel project.