Using Causal Discovery to Design Agent-based Models

Janssen, S.A.M.; Sharpanskykh, Alexei; Mohammadi Ziabari, S.S.

DOI
10.1007/978-3-030-94548-0_2

Publication date
2022

Document Version
Final published version

Published in
Multi-Agent-Based Simulation

Citation (APA)
Janssen, S. A. M., Sharpanskykh, A., & Mohammadi Ziabari, S. S. (2022). Using Causal Discovery to Design Agent-based Models. In K. H. Van Dam, & N. Verstaevel (Eds.), *Multi-Agent-Based Simulation* (Vol. XXII, pp. 15-28). (Lecture Notes in Computer Science; Vol. 13128). https://doi.org/10.1007/978-3-030-94548-0_2

Using Causal Discovery to Design Agent-Based Models

Stef Janssen, Alexei Sharpanskykh, and S. Sahand Mohammadi Ziabari(✉)

Delft University of Technology, Delft, The Netherlands
{s.a.m.janssen,O.A.Sharpanskykh,s.s.mohammadiziabari}@tudelft.nl

Abstract. Designing agent-based models is a difficult task. Some guidelines exist to aid modelers in designing their models, but they generally do not include specific details on how the behavior of agents can be defined. This paper therefore proposes the AbCDe methodology, which uses causal discovery algorithms to specify agent behavior. The methodology combines important expert insights with causal graphs generated by causal discovery algorithms based on real-world data. This causal graph represents the causal structure among agent-related variables, which is then translated to behavioral properties in the agent-based model. To demonstrate the AbCDe methodology, it is applied to a case study in the airport security domain. In this case study, we explore a new concept of operations, using a service lane, to improve the efficiency of the security checkpoint. Results show that the models generated with the AbCDe methodology have a closer resemblance with the validation data than a model defined by experts alone.

Keywords: Causal discovery · Agent-based modelling · Airport security

1 Introduction

Agent-based models have shown to be useful in a variety of areas, ranging from urban planning to ecology and security [1]. Some guidelines exist to aid (new) modelers in their model development, and they share some similarities [2, 3]. Most of these guidelines are quite high-level, and do not go beyond a description of which elements have to be defined. A notable exception is the 'overview, design concepts, and details'(ODD) protocol, which has been used widely in literature [4, 5]. It provides a detailed set of steps, along with guidelines, to design agent-based models and individual-based models. The ODD protocol has evolved over time. Firstly, it was observed that there was a lack of decision-making features in the ODD protocol; the protocol did not support the definition of human behavior that contains decisions, adaptation, and learning. This leads to a revised version of ODD, the ODD+D (Decision) protocol [6]. However, the lack of properly incorporating data in the empirical models was an important issue with ODD+D. Therefore, ODD+2D (ODD + Decision + Data) [7] was introduced to solve this lacking. While this method added structure to incorporating data in agent-based models, the creativity of the modeler remains a central component. With more and more data becoming available over the last decades, methods to interpret and understand this data became better as well. Numerous methods to find these patterns and relationships exist, of which regression, neural networks, and clustering are three examples [8]. A particularly promising method to find relationships between variables is that of causal inference [9]. These causal graphs show the causal relationships between variables and identify structure in the dataset. In this work, we propose the Agent-based Causal discovery Design methodology (AbCDe), a novel methodology that aids the development of agent-based models using causal discovery.

This paper is structured as follows. In Sect. 2, related work in the areas of designing agent-based models and causal discovery are reviewed. Then, the AbCDe methodology as proposed in this work is outlined in Sect. 3. The case study is outlined in Sect. 4. Finally, the work is concluded in Sect. 5.

2 Related Work

Designing agent-based models is a complex task. An agent-based model has three main components: agents, environment, and interactions. Klugl and Bazzan [3] state that agent-based models are an appropriate choice when a system meets a set of six different conditions. These conditions for instance include the existence of local interactions and heterogeneity in states and behavioral rules. However, the work lacks a proper method to describe how agents can be defined. It is recognized by the community that a uniform framework or methodology for designing agent-based models is lacking [2]. To design a conceptual model or model a software program, there are two popular standards, the Unified Modeling Language (UML) and the 'overview, design concepts, and details' (ODD). UML is used to represent different classes in object-oriented programming and the connection between them [29].

The ODD protocol has been used widely in literature [4, 5]. While the ODD protocol contains detailed steps to design agent-based models, no insights on how to design the behavioral properties of agents are provided. With the right dataset, data-driven methods may be useful in specifying the behavioral properties of agents. These methods find relationships between variables in a dataset, which could determine relationships between actions of agents and the outcomes in the environment. The successor of the ODD protocol, the ODD+2D protocol, consists of two major steps: 1) data preprocessing, and 2) linking data [7]. The link between data components and agent-based modeling elements in ODD+2D consists of three main sections: 1) overview, 2) design concepts, and 3) details.

While the ODD+2D adds structure to incorporating data in agent-based models, it remains up to the modeler how to use this data. This topic was explored in detail in [10]. In that work, behavioral properties of agents are learned from data by applying machine learning techniques, such as support vector machines. While these more traditional machine learning techniques are effective tools to learn behavioral properties, they do not reveal the structure of relationships between variables related to agents. A particularly promising method to reveal this structure is that of causal discovery.

Traditionally, mathematical analysis is used to find relationships between variables. However, a relationship between two variables does not necessarily mean that one variable causes the other. In the field of causality, causal relationships between variables are found by using causal graphs [9, 11]. Causal graphs can be created using two main methods. In the first method, experts use their knowledge to manually create a graph, while in the second method causal graphs are generated using data. An example of manually creating a causal graph using expert knowledge is discussed in [12]. The second method, which is also applied in this work, uses causal discovery algorithms. These algorithms can broadly be categorized into two categories: score-based methods [13] and constraint-based methods [14]. Score-based methods assign a score to a set of potential graphs and then select the graph with the highest score. Constraint-based methods define constraints on causal graphs based on statistical independence of variables. In the constraint-based category, the PC algorithm [16] is very popular, while in the score-based category, the GES algorithm [22] is frequently used in literature.

The most extensive work in the interaction of agent-based modelling and causality is by Casini and Manzo [15], who provide a technical report on the differences and similarities of agent-based modeling and causality. However, their work only focuses on analyzing agent-based models, and not on designing them. Then, Kvassay et al. [16] provide a method, based on causal partitioning, to analyze causal relationships relating to emergence in agent-based models. This work also focuses on analyzing agent-based model behavior and does not cover designing them either. Guerini and Moneta [17] cover the topic of agent-based model validation. They estimate time-series of economic models using structural vector autoregressive (SVAR) models. Using causal discovery algorithms, they generate two SVAR models: one based on results generated by the designed agent-based model, and the other based on actual data. When the two SVAR models are similar enough, the agent-based model is considered validated. This work only covers agent-based model validation and does not cover designing agent-based models. Finally, Janssen et al. [18] developed a methodology that uses causal discovery algorithms to analyze emergent behavior in agent-based models. They show that causal graphs can he experts to interpret and identify emergent behavior in agent-based models. To overcome the lack of work using causal discovery algorithms to design agent-based models, we introduce the AbCDe methodology below.

3 Methodology

This section outlines the novel Agent-based Causal discovery Design methodology, called AbCDe, which is used to design agent-based models with the aid of causal discovery algorithms. The methodology includes all aspects needed to design agent-based models, and uses causal discovery algorithms in one of its steps.

The methodology exploits the ever-growing availability of data to design agent-based models. Using data on the behavior of agents, a causal graph is generated using causal discovery algorithms. This graph is then, with the aid of experts, translated into behavioral properties of agents. These properties ultimately determine the dynamics of the model, leading to insights into the phenomenon that is modeled. Causal discovery algorithms provide a more structured method to develop agent-based models than relying on experts alone. However, experts are still needed in many aspects of the methodology to ensure that the model is of high quality. This combination of causal discovery algorithms and experts can lead to better models than models created by experts alone.

The AbCDe methodology contains five steps, which are graphically outlined in Fig. 1. As with many modelling studies, in the first step of AbCDe, the purpose of the model is specified, research questions are formulated and hypothesis are formulated. Then, in the second step of the methodology the scope of the model is determined. This specifies what elements will be included in the model and what will not be included. Once the scope is clarified, a conceptual model is formalized. The conceptual model forms the basis for the remainder of the methodology. In this model, agents are identified first. An agent, in this work, is defined as an entity that perceives its environment through sensors and acts upon that environment through effectors [19]. In this step, we specifically focus on the identification of the agents to be modeled, along with their characteristics and the behavior that they can exhibit. Only the higher-level behavior that the agent exhibits is specified (i.e., what the agent can do); the full specification of the behavior (i.e., how the agent does it) will be done in step 4.

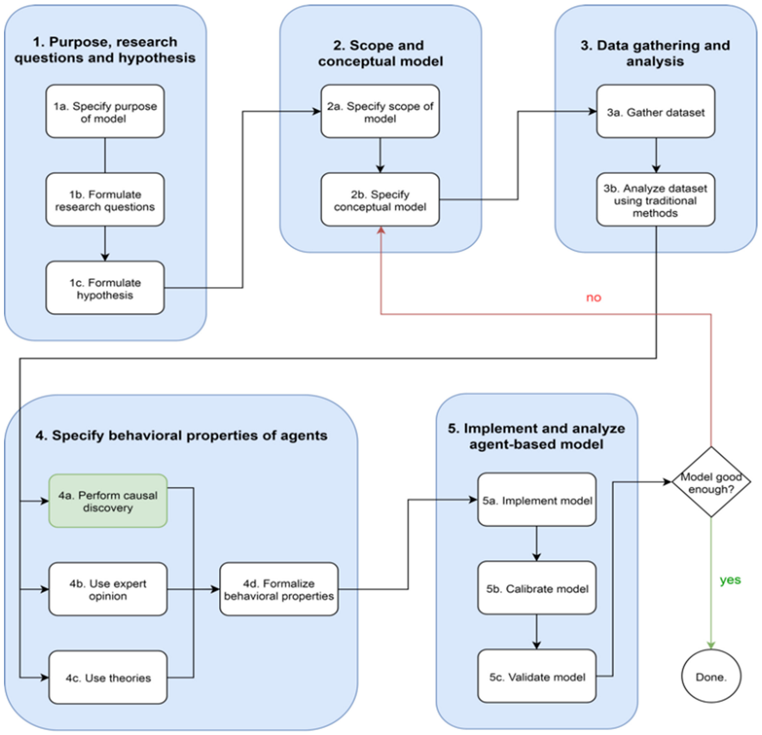

1. Purpose, research questions and hypothesis
1a. Specify purpose of model
1b. Formulate research questions
1c. Formulate hypothesis
2. Scope and conceptual model
2a. Specify scope of model
2b. Specify conceptual model
3. Data gathering and analysis
3a. Gather dataset
3b. Analyze dataset using traditional methods
no
4. Specify behavioral properties of agents
4a. Perform causal discovery
4b. Use expert opinion
4c. Use theories
4d. Formalize behavioral properties
5. Implement and analyze agent-based model
5a. Implement model
5b. Calibrate model
5c. Validate model
Model good enough?
yes
Done.

Fig. 1. The AbCDe methodology as used in this work. (Color figure online)

After identifying the agents and specifying the environment, data is collected about the behavior and characteristics of the agents in the model in step 3. This data is obtained by observing agents, their actions, and the consequences of these actions in the real world. This will later be used to specify behavioral properties of agents. Depending on what is modeled, different types of data can be collected. The collected data that will be used to generate behavioral properties is always on the agent-level (and not population-level), and should therefore contain as much detail about the characteristics of the agent (as defined in the conceptual model of the previous step), its behavior and the results of this behavior. Data is therefore in the form of characteristics of agents, actions performed by agents, effects of agent actions on the environment and effects of agent actions on other agents. The collected data is then analyzed following standard data analysis techniques, such as clustering, regression, and statistical tests. This provides early insights into the behavior of agents and will be useful for the next step of the methodology.

In step 4, behavioral properties are formalized based on two sources: causal discovery and expert input. This step includes the unique contribution of this paper in which causal discovery is used to define behavioral properties (green box of Fig. 1). Both causal discovery and expert input are discussed in detail below, combined with a discussion on how to translate them into behavioral properties.

Causal discovery algorithms (see also Sect. 2) are used to infer a causal structure from the gathered data of step 3. Before applying the algorithm, the data has to be preprocessed. This preprocessing is done to ensure that only individual agent behavior is found, and not collective emergent effects. These emergent effects should be part of the model, but not explicitly coded into the behavior of agents. It should emerge from the behavior and interaction of agents in the model (see also the work of Janssen et al. [18]).

Furthermore, the dataset has to be organized such that a single causal graph for a single agent is produced. Data of other agents can be included in the dataset for the agent under consideration, so that observable behavior, such as communication and alterations of the environment, can be found by the causal discovery algorithms as well.

After preprocessing, a causal discovery algorithm is applied to the dataset, leading to a causal graph representing the behavior of an agent in the model. The generated graphs relate characteristics of agents to exhibition of their behavior by means of including an arrow between them. Results of the behavior of other agents or properties of environmental objects are included in the graph following the same standard.

After generating the causal graphs, an expert provides input for two purposes. While causal discovery is useful, applying it still requires some level of expert knowledge [23]. The expert checks the graph that was generated for inconsistencies with their knowledge and the original data analysis that was performed in the previous step. These inconsistencies are then fixed in the graph. Second, the expert provides additional insights based on theories from literature or their experience. These insights can be used to compensate for missing data in the dataset, and provide another means to specify behavioral properties in the next step. After obtaining both the causal graph and the input of experts, the behavioral properties are specified. These properties can be obtained from the graph (enhanced by the expert) by selecting a variable to be used as a behavioral property and using its parents as building blocks to specify the behavior.

Finally, in step 5, the defined model is implemented, calibrated, and validated. When the model sufficiently resembles validation data, the AbCDe methodology is finished. When this is not the case, the methodology returns to step 2.

4 Case Study

We apply the AbCDe methodology to a case study in the field of airport security. In airport terminals, the security checkpoint is the most important bottleneck for passengers (leading to unwanted waiting time) and an important source of costs for airport management. As airport passenger numbers are projected to increase in the future, security checkpoints have to be operated efficiently. In this case study, we explore a new concept of operations, using a service lane, to improve the efficiency of the security checkpoint. Service lanes process passengers that are expected to be slow, and the other open lanes (defined as normal lanes) process the remaining passengers. A standard lane is a lane in which no experiment took place, and all passengers are processed. This concept of operations is projected to improve the overall throughput of the system, as faster passengers do not have to wait for slower passengers in front of them. Slow passengers also receive extra help from experienced security officers, potentially increasing the throughput as well.

We design an agent-based model following the AbCDe methodology to determine the effects of implementing a service lane on the throughput of the security checkpoint, as compared to a standard setup.

The purpose of the model is to determine the effects of implementing a service lane on the throughput of the security checkpoint, as compared to a standard setup. The scope of this experiment is to find passengers behavioral traits in the collect and drop section at the security checkpoint in the airport, while disregarding cognitive behavior of the passengers. Now that the scope of the model is clarified, we specify the conceptual model (step 2 of AbCDe). This conceptual model is specified in more detail in a technical report [28], but the most important elements are provided below. We identify the environmental objects that are modeled first. These are outlined below.

- **Luggage.** Luggage is owned by a passenger and has a specific threat level. This is a real value between 0 and 1.
- **Box.** Object in which luggage is dropped. Luggage can be dropped into multiple boxes.
- **Walk-through metal detector (WTMD).** Randomly specifies passengers that require an explosive trace detection (ETD) test or patdown.
- **Flight.** Abstract concept that has an associated flight time. Passengers are associated with exactly one flight.
- **Queue separator.** Physical objects that are used to form queue areas for passengers.

Now that the environment of the model is specified, we specify the agents of the model.

- **Passenger.** Agent that is associated with a flight, moves through the security checkpoint.
- **X-ray operator.** Uses the X-ray sensor to determine if luggage needs an extra check, and communicates this with the luggage check operator.
- **Luggage check operator.** Checks luggage when requested by the X-ray operator.
- **Patdown operator.** Performs patdowns and ETD checks.

We collected data of passengers moving through the security checkpoint at Rotterdam The Hague Airport [26] (step 3 of AbCDe). Data for a total of 2277 passengers, flying to 16 different destinations was gathered. Three types of lanes were considered: standard, normal and service lanes. Data for standard lanes were gathered between 23 February 2018 and 17 April 2018, while data for normal and service lanes were collected on the experimental days: 17 December 2018 and 18 December 2018. A service lane was used to process passengers that are expected to be slow, while the normal lanes processed the other passengers. As mentioned in earlier, the scope of this paper is based on the generation of behavioral properties on the drop and collect behavior of passengers. We use the data of the standard lanes to generate the behavioral properties of the agent, while we use data of the service lane experiment to validate the models.

To generate the graphs, we combine the score-based GES [22] algorithm and the constraint-based PC algorithm [16], following the work of Janssen et al. [18] (see also Sect. 2). We use the following variables from the dataset to generate the graph for the characteristics model: *drop*, *collect* (the time a passenger takes to drop/collect luggage on/from the belt), *boxes* (the number of boxes the passenger uses at the security checkpoint), *type* (the type of passenger, see Table 2 and 3), and *group size* (the size of the group the passenger travels with, see Table 2). These variables are a combination of the characteristics of the agent, and the two behavioral properties that we are interested in (*drop* and *collect*) and are used to generate the causal graph that we will refer to as the characteristics model.

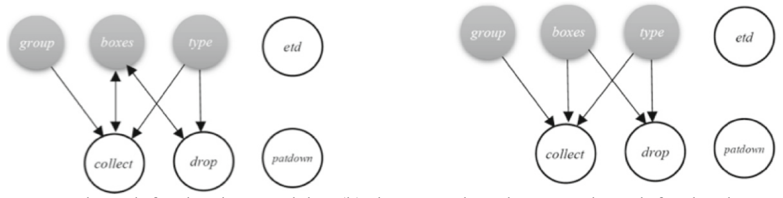
The same variables are used for a causal model that we define as the extended model. However, the following variables are additionally used in the extended model: $drop_p$ (the drop time of the previous passenger in line), $wait\ I_p$ (the time the previous passenger waited between dropping luggage and going through the WTMD), $boxes_p$, $type_p$, and $group\ size_p$ (the boxes, type, and group size of the previous passenger respectively). It is important to note that these consist of the observable behavior and characteristics (i.e. observable by the passenger) of the passenger in front of the passenger for which the behavioral properties are defined.

Figure 2a and Fig. 3a show the graphs that were generated by the causal discovery algorithm for the characteristics model and the extended model respectively. Based on expert insights, these graphs are translated to their final versions, as shown in Fig. 2b and Fig. 3b. This procedure of using expert insight to update the graphs is aided by the work of Shrier and Platt [12].

The graph generated for the characteristics model shows that both *ETD* (the time the passenger receives an Explosive Trace Detection) and *patdown* (the time the passenger receives a patdown) are not connected to any other variable in the graph. That means that these are independent variables that will be generated in the model independently as well. Both *boxes* and *type* show a causal relationship with both drop and collect. This implies that these characteristics combined are of influence on the speed in which passengers drop and collect luggage. The generated graph additionally shows that boxes is caused by both drop and collect. Based on expert advice, we assume this link to be unidirectional in the direction of drop and collect. Finally, the size of the group influences *collect*, but not *drop*. In a security checkpoint, passengers traveling in groups often wait for each other to finish collecting their luggage. In this way, they can continue their journey to the gate together. This is not the case for dropping luggage, as passengers can only pass through the WTMD individually.

The extended model is based on a generated graph that contains five more variables and is therefore more complex. This shows that some variables related to the previous passengers are closely related to the same variables of the passenger under consideration. To allow for a fair comparison between the two models, we assume that both the group size and the type of agent are independently generated. The arrows between variables show the causalities between them.

To this end, we remove the links $group_p \leftrightarrow group$, $type_p \leftrightarrow etd$, $typep \leftrightarrow type$ and $dropp \rightarrow group$. Another important factor that we use to correct the graph, is the assumption that the passenger under consideration cannot influence the characteristics or behavior of the passenger that is next in line.



(a) The generated graph for the char. model. (b)The expert-based corrected graph for the char. model.

Fig. 2. The generated graph for the characteristics model, along with the expert-based corrections. Gray variables are characteristics of passengers, while white variables are observable behaviors.

The links $drop \rightarrow dropp$, $boxes \rightarrow boxesp$ and $type \rightarrow boxesp$ are therefore removed. Finally, we reverse the direction of the arrow $drop \rightarrow boxes$ to correct the direction of causality.

Now that the graphs are complete, we transform them into agent behavior (step 4d of the AbCDe methodology). This is done by fitting distributions of a variable using its parent variables in the causal graph. For the characteristics model (Fig. 2b), we generate conditional random distributions for the time the passenger takes to drop luggage (based on *boxes* and *type*), and collect luggage (additionally based on the *group size*). To fit these distributions, we use data of all passengers in the calibration set that possess the right characteristics. Equations 1–2 below show the *drop* and *collect* distributions for a *business* passenger traveling alone with one box worth of luggage.

$$drop = GeneralizedExtremeValueDistribution(43.95, 19.81, -0.07) \tag{1}$$

$$collect = NormalDistribution(36.12, 20.93) \tag{2}$$

where the Normal distribution is parameterized by its mean (first param.) and standard deviation (second param.), and the Generalized Extreme Value distribution is parameterized by its location (1st param.), scale (2nd param.) and shape (3rd param.).

A similar procedure as above is followed for the extended model. However, the parent variables that specify the drop and collect distribution are continuous, as compared to discrete and categorical variables in the characteristics model. We therefore use a method to fit a generalized linear model [25], based on maximum likelihood estimation (MLE), for the drop and collect distributions. We use the Poisson distribution as a basis for both the *drop* and *collect* variables, and a linear combination of their respective parent variables to specify the parameter λ of the Poisson distribution. Equations 3–6 show the distributions for *drop* and *collect*.

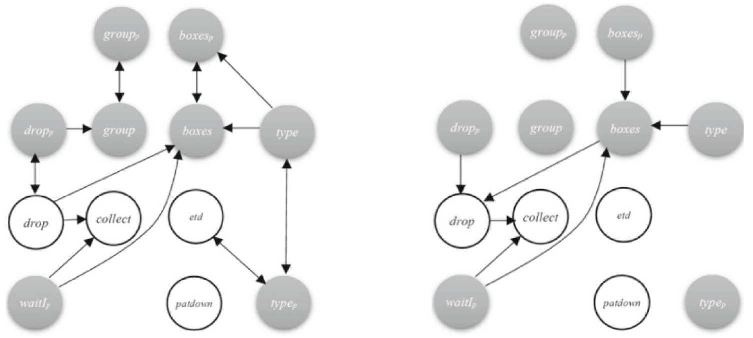
$$\lambda_1 = 3.30 + 0.24 \times boxes + 0.009 \times drop_p - 0.001 \times \left(boxes \times drop_p\right) \tag{3}$$

$$drop_p = PoissonDistribution(exp\, \lambda_1) \tag{4}$$

$$\lambda_2 = 3.86 + 0.006 \times drop - 0.002 \times waitI_p - 2.18e^{-5} \times (drop \times waitI_p) \tag{5}$$

$$collect = PoissonDistribution(exp\, \lambda_2) \tag{6}$$

The *boxes* parameter is based on the passenger *type*, the number of boxes that the previous passenger used (*boxesp*) and the time between dropping luggage and going through the WTMD of the previous passenger (*waitIp*). When collecting data, we observed that passengers will take longer to drop their luggage if they cannot continue to the WTMD yet. For instance, they realize they have their belts still on and use an extra box to put that in, or take off their shoes and put that in a new box. This may explain the relationship between the number of boxes and these parameters.



(a) The generated graph for the extended model. (b) The corrected graph for the extended model.

Fig. 3. The generated graph for the extended model, along with the expert-based corrections.

We follow a generalized linear modeling approach to specify the boxes distribution in the extended model as well. However, as *type* is a categorical variable, we specify a distribution for each passenger type individually. Equations 7–8 show the distribution for the business passenger; other passenger types are defined similarly.

$$\lambda_3 = 0.9 - 0.03 \times boxes_p - 0.01 \times waitI_p + 0.003 \times \left(boxes_p \times waitI_p\right) \tag{7}$$

$$drop = PoissonDistribution(exp\,\lambda_3) \tag{8}$$

We have implemented these two models in the AATOM simulator, an agent-based airport terminal operations simulator [25], as well as a model based on expert-input alone. For calibration, we focus our analysis on a setup with a single standard lane open. For validation, we focus the analysis on a single service lane and a single normal lane open. We calibrated the model with the data that was collected for the nine standard lanes. All important parameters, their descriptions, and their calibrated values can be found in Tables 1, 2 and 3. We ran a total of N = 1,000 simulations for all three models, and extracted the following four output values for each simulation run: *wait I time*, *wait II time* (time between WTMD passage and collecting luggage of passengers), *throughput* (number of passengers processed per hour) and *occupation* (mean number of passengers in the security checkpoint). We perform linear normalization for each of these output values, using the following functions.

$$\sigma = sd(X) \tag{9}$$

$$x_{min} = mean(X) - 2\sigma \tag{10}$$

$$x_{max} = mean(X) + 2\sigma \tag{11}$$

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{12}$$

Where X represents the vector of all output values of a specific type (i.e. all simulated *wait I* times), and $x \in$ X. We perform the same procedure for these output parameters in the real data. We calculate the Euclidean distance between each of the simulated and the real data and find that the calibration data closely resembles simulated data.

For validation, we change the distribution of arriving passenger types, have two lanes open instead of one, and specify one lane as a service lane (see Table 3). The proportion of passengers per type that are sent to the service lane is also specified in Table 3. We normalize the data following the same approach as the calibration and calculate the distance again. The resulting distances to the validation data for the calibrated models are shown in Table 4.

Table 1. The calibrated parameters of the model (simplified table).

Parameter	Description	Calibrated value
Passenger		
desiredSpeed	The desired speed (in m/s) that the passenger moves through the checkpoint	Calibrated based on the data that was collected for the nine standard lanes
groupSize	The size of the group the passenger travels with	Based on groupSizeDistribution (Table 2)
Operator		
WTMDCheckDistribution	The distribution of patdown times	GeneralizedExtremeValueMathDistribution (19.19, 9.35, −0.01);
Flight		
arrivalDistribution	The distribution in which passengers arrive	20% (first half hour), 60% (second), 20% (third), 0% (last) based on the expert knowledge at the airport
Passenger distribution		
passengerTypeDistribution	The distribution of passenger types in the population	Table 3
groupSizeDistribution	The distribution of group sizes in the population	Table 2
serviceLaneDistribution	The proportion of passengers per type that will be directed to the service lane	Table 3

Results show that the extended model has the lowest distance to validation data. It is followed by the expert model and finally the characteristics model. These results indicate that building a model with our methodology can improve the accuracy of the models over models developed by experts alone. While more work is needed to show the (dis)advantages of the methodology, these initial results are promising.

Table 2. The distribution of group sizes for the different passenger types.

	Group size 1	Group size 2	Group size 3
Business	0.75	0.16	0.09
Senior	0.12	0.67	0.21
Young	0.02	0.15	0.83
Family	0.33	0.52	0.27
PRM	0.16	0.58	0.26
Regular	0.34	0.50	0.16

Table 3. The proportion of passengers.

	Calibration	Validation	Service lane
Business	0.15	0.17	0.21
Senior	0.17	0.23	0.60
Young	0.15	0.13	0.41
Family	0.11	0.07	0.76
PRM	0.012	0.004	1.00
Regular	0.41	0.37	0.51

Table 4. The calibrated models along with their distances to the validation data.

Model	desiredSpeed	Distance
Expert	1.4	3.6071
Characteristics	1.4	3.6486
Extended	1.5	3.4862

5 Discussion and Conclusion

Causal discovery algorithms translate data into a directed causal graph that reveals the causal structure among variables. In this paper, we investigated how these algorithms can be incorporated in the design process of agent-based models. We proposed an agent-based model-design methodology, called AbCDe, that uses causal discovery algorithms and the growing availability of data to specify behavioral properties. This methodology combines traditional expert-based model design techniques with causal graphs to design better models. We applied the methodology to a case study in the airport domain. The models that were generated with the AbCDe methodology show closer resemblance to validation data than an existing expert-based model. Future work can also focus on developing dedicated causal discovery algorithms for agent-based model development, instead of adapting existing algorithms for that purpose.

An important issue that occurred during the generation of causal graphs is that different algorithms and parameters produce quite diverse causal graphs. By integrating the PC algorithm with the GES algorithm, this problem is partially addressed, but certainly not solved. Further developments in the field of causality will address this problem.

A major advantage of our methodology is that it provides modelers with a toolbox to design agent-based models. In cases where agent-specific data can be gathered, such as our example of airport security checkpoint, our methodology can impact the final quality of the developed model.

References

1. Janssen, S., Sharpanskykh, A., Curran, R.: Agent-based modelling and analysis of security and efficiency in airport terminals. Transp. Res. Part C Emerg. Technol. **100**, 142–160 (2019)
2. Klugl, F., Oechslein, C., Puppe, F., Dornhaus, A., et al.: Multi-agent modelling in comparison to standard modelling. Simul. News Europe **40**, 3–9 (2004)
3. Klugl, F., Bazzan, A.L.: Agent-based modeling and simulation. AI Mag. **33**(3), 29 (2012)
4. Grimm, V., et al.: A standard protocol for describing individual based and agent-based models. Ecol. Model. **198**(1–2), 115–126 (2006)
5. Grimm, V., Berger, U., DeAngelis, D.L., Polhill, J.G., Giske, J., Railsback, S.F.: The odd protocol: a review and first update. Ecol. Model. **221**(23), 2760–2768 (2010)
6. Muller, B., et al.: Describing human decisions in agent-base models- odd + D, an extension of the odd protocol. Environ. Model. Softw. **48**, 37–48 (2013)
7. Laatabi, A., Marilleau, N., Nguyen-Huu, T., Hbid, H., Babram, M.A.: Odd+2D: an odd based protocol for mapping data to empirical ABMs. J. Artif. Soc. Soc. Simul. **21**(2), 9 (2018)
8. Witten, I.H., Frank, E., Hall, M.A., Pal, C.J.: Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann, San Francisco (2016)
9. Pearl, J.: Causality. Cambridge University Press, New York (2009)
10. Kavak, H., Padilla, J.J., Lynch, C.J., Diallo, S.Y.: Big data, agents, and machine learning: towards a data-driven agent-based modeling approach. In: Proceedings of the Annual Simulation Symposium, p. 12. Society for Computer Simulation International (2018)
11. Peters, J., Janzing, D., Scholkopf, B.: Elements of Causal Inference: Foundations and Learning Algorithms. MIT press, Cambridge (2017)
12. Shrier, I., Platt, R.W.: Reducing bias through directed acyclic graphs. BMC Med. Res. Methodol. **8**(1), 70 (2008)
13. Magliacane, S., Claassen, T., Mooij, J.M.: Ancestral causal inference. In: Advances in Neural Information Processing Systems, pp. 4466–4474 (2016)
14. Colombo, D., Maathuis, M.H., Kalisch, M., Richardson, T.S.: Learning high-dimensional directed acyclic graphs with latent and selection variables. Ann. Stat. **40**, 294–321 (2012)
15. Casini, L., Manzo, G.: Agent-based models and causality: a methodological appraisal, Linkoping University, Department of Management and Engineering, The Institute for Analytical Sociology, The IAS Working Paper Series 2016:7
16. Kvassay, M., Krammer, P., Hluchý, L., Schneider, B.: Causal analysis of an agent-based model of human behaviour. In: Complexity 2017, pp. 1–18 (2017). https://doi.org/10.1155/2017/8381954
17. Guerini, M., Moneta, A.: A method for agent-based models validation. J. Econ. Dyn. Control **82**, 125–141 (2017)
18. Janssen, S., Sharpanskykh, A., Curran, R., Langendoen, K.: Using causal discovery to analyze emergence in agent-based models. Simul. Model. Pract. Theory **96**, 101940 (2019)
19. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach. Pearson Education Limited, New Delhi (2016)
20. Hauser, A., Buhlmann, P.: Characterization and greedy learning of interventional Markov equivalence classes of directed acyclic graphs. J. Mach. Learn. Res. **13**, 2409–2464 (2012)
21. Spirtes, P., Zhang, K.: Causal discovery and inference: concepts and recent methodological advances. Appl. Inform. **3**, 3 (2016)
22. Tisue, S., Wilensky, U.: NetLogo: a simple environment for modeling complexity. In: International Conference on Complex Systems, Boston, MA, vol. 21, pp. 16–21 (2004)
23. Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K., Balan, G.: MASON: a multiagent simulation environment. Simulation **81**(7), 517–527 (2005)

24. North, M.J., et al.: Complex adaptive systems modeling with repast symphony. Complex Adapt. Syst. Model. **1**, 1–26 (2013)
25. Janssen, S., Sharpanskykh, A., Curran, R., Langendoen, K.: AATOM: an agent-based airport terminal operations model simulator. In: Proceedings of the 51st Computer Simulation Conference, SummerSim 2019, Berlin, Germany, 22–14 July 2019
26. Janssen, S., van der Sommen, R., Dilweg, A., Sharpanskykh, A.: Data-driven analysis of airport security checkpoint operations. Aerospace **7**(6), 69 (2020)
27. Daniel, W.W.: Kolmogorov-smirnov one-sample test. Appl. Nonparametr. Stat. **2** (1990)
28. Nelder, D.J.A., Wedderburn, R.W.: Generalized linear models. J. Royal Stat. Soc. Ser. A (General) **135**(3), 370–384 (1972)
29. Bersini, H.: UML for ABM. J. Artif. Soc. Soc. Simul. **15**(1), 9 (2012). http://jasss.soc.surrey.ac.uk/15/1/9.html