UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR INFORMATIONSSYSTEME

From the Institute of Information Systems
of the University of Lübeck
Director: Prof. Dr. rer. nat. habil. Ralf Möller

Navigate Through Troubled Waters: Adapt to Environmental Changes in Dynamic Probabilistic Relational Models

Dissertation
for Fulfillment of
Requirements
for the Doctoral Degree
of the University of Lübeck

from the Department of Computer Sciences and Technical Engineering

Submitted by

Nils Finke
from Mölln

Lübeck 2023

First referee: Prof. Dr. rer. nat. Ralf Möller
Second referee: Prof. Dr. rer. nat. Esfandiar Mohammadi

Date of oral examination: 12. October 2023

Approved for printing.
Lübeck, 10. November 2023

Abstract

In an increasingly interconnected world, a growing need arises for expressive formalisms to describe uncertainty and complex relational structures that are evident in our world. Above all, this means that the influence of more and more objects, as well as their interconnections, must be taken into account in an appropriate model. If a model includes only a few objects, it is still considerably small. But once more objects are included, i.e., with increasing domain size, the complexity of the model increases. As the model complexity increases, so does the runtime complexity when reasoning based on the model takes place. However, with an increasing size of the domain, the likelihood of keeping repetitive information in the model also increases. If repetitive information is available in data, it can be exploited to address the challenge of increasing complexity. In statistical relational artificial intelligence (StaRAI), dynamic probabilistic relational models (DPRMs) along with lifted inference approaches have successfully emerged to take advantage of this matter. DPRMs describe dependencies between objects, their attributes and their relations in a sparse manner by taking advantage of repetitive information. In lifted inference, repetitive information, also referred to as symmetries, is exploited by using one object from a group of symmetrical objects as a representative for the whole group. Symmetrical objects and structures are omnipresent in nature, but in almost any real-life setting they change over time, so models that represent the respective setting must be adaptable to environmental changes. Thus, this work focuses on adapting DPRMs to environmental changes, specifically, changes in the model's domain size, probability distributions and symmetry structures, resulting from the requirements of any real-world setting.

We categorise the contributions of this work into three parts. In the first part, we extend the semantics of DPRMs for dynamically changing sets of domain objects to be substituted for variables used in the model such that sparse domain models can be specified and efficient query answering algorithms can be provided. In the second part, we introduce symmetry construction, an approach to approximate temporal symmetries by a symbolisation scheme to cluster objects that tend to behave the same over time. Symmetry structures can break over time if the behaviour of one object changes in comparison to its symmetry group or if certain information is not available at runtime and only becomes known retrospectively. In such cases, however, the behaviour can be derived from the behaviour of other objects in its symmetry group to retain a lifted representation. In the third part, we introduce two change point detection approaches to identify points in time with a significant change in symmetry structures and distributions that requires re-learning of model parameters. In the context of an application from dry-bulk-shipping, we show that the extensions to DPRMs enable adaptions to environmental changes and increase the robustness, accuracy and ability to perform efficient inference.

Kurzfassung

In einer zunehmend vernetzten Welt besteht ein wachsender Bedarf an aussagekräftigen Formalismen zur Beschreibung von Unsicherheit und komplexen, relationalen Strukturen, die in unserer Welt zu beobachten sind. Dies bedeutet vor allem, dass der Einfluss von immer mehr Objekten sowie deren Beziehungen zueinander in einem geeigneten Modell berücksichtigt werden müssen. Wenn ein Modell nur wenige Objekte umfasst, ist es noch recht klein. Sobald jedoch mehr Objekte einbezogen werden, d.h. mit zunehmender Domänengröße, steigt die Komplexität des Modells. Mit zunehmender Komplexität des Modells steigt auch die Laufzeitkomplexität bei der Durchführung von Schlussfolgerungen auf der Grundlage des Modells. Mit zunehmender Größe der Domäne steigt jedoch auch die Wahrscheinlichkeit, dass repetitive Informationen im Modell enthalten sind. Wenn repetitive Informationen in den Daten vorhanden sind, können diese genutzt werden, um die Herausforderung der zunehmenden Komplexität zu bewältigen. Im Feld der *Statistical Relational Artificial Intelligence* (StaRAI) haben sich dynamische probabilistische relationale Modelle (DPRMs) zusammen mit *angehobenen* Inferenzansätzen, welche genau diesen Umstand auszunutzen, als erfolgreich herausbilden können. DPRMs beschreiben Abhängigkeiten zwischen Objekten, ihren Attributen und ihren Beziehungen auf eine kompakte Art und Weise, indem repetitive Informationen ausgenutzt werden. In *angehobener* Inferenz werden repetitive Informationen, die auch als Symmetrien bezeichnet werden, ausgenutzt, dadurch dass ein Objekt aus einer Gruppe von symmetrischen Objekten als Repräsentant für die gesamte Gruppe verwendet wird. Symmetrische Objekte und Strukturen sind in der Natur allgegenwärtig, allerdings verändern sich diese in fast jeder realen Umgebung im Laufe der Zeit, sodass Modelle, die die jeweilige Umgebung repräsentieren, an die Veränderungen der Umwelt kontinuierlich angepasst werden müssen. Daher konzentriert sich diese Arbeit auf die Anpassung von DPRMs an Umweltveränderungen, insbesondere an Veränderungen der Domänengröße des Modells, der Wahrscheinlichkeitsverteilungen und der Symmetriestrukturen, die sich aus den Anforderungen einer realen Umgebung ergeben. Wir gliedern die Beiträge dieser Arbeit in drei Teile.

Im ersten Teil erweitern wir die Semantik von DPRMs für sich dynamisch ändernde Mengen von Domänenobjekten, die anstelle von Variablen im Modell verwendet werden, sodass kompakte Domänenmodelle spezifiziert und effiziente Algorithmen zur Beantwortung von Anfragen bereitgestellt werden können. Im zweiten Teil stellen wir eine Symmetriekonstruktion, ein Ansatz zur Identifikation von temporalen Symmetrien, vor. Mittels eines Symbolisierungsschema werden Objektgruppen identifiziert, die sich im Laufe der Zeit gleich verhalten. Symmetriestrukturen können sich im Laufe der Zeit auflösen, z.B. wenn sich das Verhalten eines Objekts im Vergleich zu seiner Symmetriegruppe ändert oder wenn bestimmte Informationen zur Laufzeit nicht verfügbar sind und erst im Nachhinein bekannt

werden. In diesem Fall kann das Verhalten jedoch aus dem Verhalten anderer Objekte, die sich ähnlich verhalten, abgeleitet werden, um eine *angehobene* Darstellung beizubehalten. Im dritten Teil stellen wir zwei Ansätze zur Erkennung von signifikanten Änderungen in der Modellstruktur vor, die ein Neulernen der Modellparameter erfordern. Im Kontext einer Anwendung aus dem Bereich der Bulk-Schifffahrt zeigen wir, dass die Erweiterungen von DPRMs Anpassungen an Umweltveränderungen ermöglichen und somit die Robustheit, Genauigkeit und die Fähigkeit zur effizienten Inferenz erhöhen.

Acknowledgements

The title of my dissertation could not be more appropriate – *Navigate Through Troubled Waters [...]*. This sentence not only refers to the specific use case that motivated this work but also vividly describes the journey I took in writing this dissertation. First of all, I have to give a little disclaimer before presenting these acknowledgements, as it might sound a little *cheesy* after all. Yet, it fills me with pride to finally see the result after the last years of hard work. This moment represents a rare opportunity for me to unashamedly present the results of my work and the gratitude I have for those who have supported me along the way.

Above all, I would like to thank my supervisor, Ralf Möller, who offered me the opportunity to work on this topic as part of a dissertation after completing my master's thesis. I look back on many years that have represented a lot of work not only for myself but certainly for Ralf as well. I am absolutely grateful for the opportunity, which I do not take for granted. I would also like to express my appreciation to Tanya and Marcel, who worked in the same research area at the institute, and who have accompanied and supported me on this journey from the very beginning. Their invaluable help in guiding me and determining the focus and topic of this thesis has been instrumental in its success. Further, I am particularly grateful to my employer, Oldendorff Carriers, for providing me with the unique opportunity to work on my PhD while continuing to develop my career. A special mention goes to Christoph and Soenke, who have always supported and encouraged me throughout my professional career. In addition, I would like to express my deep gratitude to Daniel and Barne for their understanding in giving me the opportunity to concentrate on my research while they devoted themselves to our adventure alongside our current commitments. Lastly, I would like to express my deepest appreciation to Marisa for her endless patience and willingness to engage in topics outside of her area of expertise and to assist me so greatly in my research. Spending countless nights and weekends with her working together on our respective PhDs made the experience much more enjoyable.

Thank you all for your incredible support and fellowship. Your contributions have been invaluable to the completion of this thesis. Against all odds, writing my dissertation ultimately led to personal and academic growth. I learned the importance of resilience and adaptability, lessons that will be useful to me beyond the scope of academia, which would not have been possible without you all.

Nils
Lübeck, October 2023

Contents

List of Algorithms

List of Abbreviations

AIS	automatic identification system
BMF	boolean matrix factorisation
BP	belief propagation
CNN	Convolutional Neural Network
RBN	relational Bayesian network
DBSCAN	density-based spatial clustering of applications with noise
DMA	Danish Maritime Authority
DMDTW	dependent multivariate dynamic time warping
DPRM	dynamic probabilistic relational model
DTW	dynamic time warping
FO jtree	first-order junction tree
FO-POMDP	first-order partially observable Markov decision processes
HMM	hidden Markov model
HM-MDP	hidden-mode Markov decision process
IMO	International Maritime Organisation
KLD	Kullback Leibler divergence
LBP	lifted belief propagation
LDJT	lifted dynamic junction tree
logvar	logical variable
LVE	lifted variable elimination
MDP	Markov decision process
MEU	maximum expected utility
meuLDJT	maximum expected utility lifted dynamic junction tree
MLN	Markov logic network

MOP	multivariate ordinal pattern
MOP_4SA	multivariate ordinal pattern symbolisation for symmetry approximation
MOP_4SCD	multivariate ordinal pattern symbolisation for symmetry change detection
parfactor	parametric factor
PDDecM	parameterised probabilistic dynamic decision model
POMDP	partially observable Markov decision process
PRV	parameterised random variable
randvar	random variable
SA_4PG	symmetry approximation for preventing groundings
StaRAI	statistical relational artificial intelligence
VE	variable elimination

List of Symbols

$\mathbf{R}$	set of random variables
$\mathbf{L}$	set of logical variables
Φ	set of factor names
$\mathbf{D}$	set of entities
$\mathcal{D}(L)$	domain of a logvar
$C, (\mathcal{X}, C_{\mathbf{X}})$	constraint restricting logical variables
$A(L_1, \ldots, L_n)$	parameterised random variable (PRV)
ϕ	potential function
$g, \phi(\mathcal{A})_{\mid C}$	parfactor
$gr(P)$	grounding of a (set of) parameterised random variable
$lv(P)$	logical variables of expression P
$dm(l)$	Domain $\mathcal{D}(L)$ of some object $l \in \mathcal{D}(L)$
$\mathcal{R}(A)$	range of a PRV A
G	model
$\mathcal{G}$	template model
G_t	local model
α	alpha message
$\mathbf{E}$	evidence, set of events
Q	query term
Ω	state space
ω	state
$\mathcal{A}$	σ-algebra
T	length of a time series
$\mathcal{X}$	multivariate time series
τ	delay between successive time points
d	order of ordinal pattern
w_{ij}	similarity count
$\mathcal{W}$	similarity graph matrix
S	symmetry cluster
$en(S)$	objects in a symmetry cluster

$\mathbf{S}$	set of symmetry clusters
P	parfactor partition
L	Laplacian matrix
D_{KL}	Kullback-Leibler divergence
$d(\mathcal{W}^t, \mathcal{W}^{t+1}, \mathcal{S})$	similarity change measure
$\mathbf{V}$	set of violation maps
$v_{\|g}$	local violation map
H	violation threshold
a	alteration factor
A	sequence of alteration factors
ϵ	mean squared error
ℓ	logistic function count
δ	external threshold

1 Introduction

Today, in almost all disciplines, be it science, technology, business, sociology, politics or other fields, vast amounts of data are collected to make more profound decisions by adopting a data-driven perspective. Since the world is characterised by uncertainty as well as complex relational structures that carry temporal information, so does the recorded data, yielding large dynamic probabilistic relational models (DPRMs)[1] being the centre of many applications. Using probabilistic models in applications involves answering sets of queries, e.g., regarding the probability of events, probability distributions, or actions leading to a maximum expected utility (MEU). Moreover, reasoning on probabilistic models is often performed under time-critical conditions, such that computational tractability is essential.

In this respect, DPRMs, together with lifted inference approaches, provide an efficient formalism to address this challenge. More specifically, DPRMs describe dependencies between entities, their attributes and their relations by sets of random variables (randvars) in a sparse manner. To encode uncertainty, DPRMs encode probability distributions by exploiting independencies between randvars using factor graphs. Factor graphs are combined with relational logic, using logical variables (logvars) as parameters for randvars to compactly represent sets of randvars, that are considered indistinguishable (without further evidence), introducing a technique called *lifting* (Poole, 2003; Gehrke *et al.*, 2018). A lifted representation of a probabilistic model allows for a sparse representation to restrain state complexity and support decreasing runtime complexity in inference.

To illustrate the potential of lifting, we consider a probabilistic model for navigational route planning, an application which is meanwhile a ubiquitous task in our everyday life, e.g., when using car navigation, in arranging pickups in ride-sharing, or in logistics for delivery – just to name a few examples. In general, routing plans are affected by several aspects, such as demand, traffic or congestion. When modelling congestion, for example, it is necessary to take into account rush hours that are related to people's movement patterns, e.g., determined by working hours or commute. Setting up a probabilistic model requires assigning a set of randvars, such as working and commuting hours, for each person. The number of people is directly related to the number of randvars in the model. If such a model includes only a few people, the model might still be considerably small in terms of

[1] pronounced *deeper* models

the number of randvars, but once it includes more people, i.e., with increasing domain size of people, the state complexity of the model also increases. Moreover, with an increase in complexity due to an increase in the domain size, the likelihood of keeping repetitive information within the model also increases. For example, in the route planning application, repetitive information is given when the same observations are made for different people with respect to the features of the probabilistic model. Intuitive examples are siblings who travel to school together or people who live in the same neighbourhood and have the same employer. In StaRAI, identical observations with respect to the features of the model made two or more times for different entities, here people, are referred to as *symmetries*[2] In contrast, unequal observations, are referred to as *asymmetries*. Lifting exactly exploits the existence of symmetries in a model: People who are symmetrical with respect to the features of the model can be treated as one person to obtain a sparse representation. Further, by exploiting symmetries, reasoning in lifted representations has no longer a complexity exponential in the number of entities represented by the model but is restricted to the number of entities with asymmetries only (Niepert and Van den Broeck, 2014; Van den Broeck and Niepert, 2015). More specifically, symmetries across entities of a model's domain, i.e., entities over randvars of the same type, are exploited by means of performing calculations in inference only once for groups of similarly behaving entities, instead of performing the same calculations over and over again for all entities individually.

Generally, lifting is based on the assumption that symmetric structures and reoccurring patterns, i.e., redundancies, are omnipresent in nature and therefore are also recorded in data. However, in any real-life setting, DPRMs have to deal with environmental changes, i.e., parameters or properties of the model are most likely to change over time and affect previously exploited symmetry structures. This leads to asymmetries in the model, relieving entities from a symmetrical consideration, called *grounding*. In particular, groundings occur with changes (a) in the domain size, (b) in symmetry structures, or (c) in probability distributions encoded in the model. These changes affect not only the accuracy of the model but also the efficiency of any lifted inference algorithm. To preserve runtime benefits in lifted inference, it is necessary to limit the effect of asymmetries on the liftability of the model. Handling asymmetries is one of the major challenges in lifted inference and is crucial for its effectiveness (Kersting, 2012; Van den Broeck and Darwiche, 2013). Therefore, the problem that this work focuses on is

the adaptation to *environmental changes*,
specifically, changes in the domain size, symmetry structures or distributions,
in *dynamic probabilistic relational models*.

More specifically, in this work, we refine and enhance DPRMs for real-world use. First, we extend the theory of probabilistic dynamic relational modelling with dynamically changing

[2]More formally, in context of DPRMs, the term *symmetries* is associated with sub-isomorph graph structures, which will be introduced and explored in the course of this work.

sets of domain objects over time. Second, we introduce a novel approach to approximate temporal symmetries to prevent groundings. Third, we present two approaches to detect environmental changes in terms of changes in symmetry structures and probability distributions.

1.1 Related Work

In recent years, there has been intensive research in the field of lifted inference, aiming to use the symmetries present in the data to speed up query answering. DPRMs, together with lifted inference approaches, have been shown to be particularly suitable for use in complex real-life settings, due to their ability to describe (a) *probabilistic, relational structures*, (b) *temporal behaviour*, and (c) *large domains in a compact manner*. Further, they (d) *provide efficient marginal query answering*, and (e) *enable sequential decision support*. Note that we derive those requirements later in this work when we set up an example application as a reference for other real-world scenarios. Furthermore, DPRMs allow decisions or calculations of inference algorithms to be interpreted, i.e., to provide explainable decision support. This is essential whenever human expertise still needs to be considered in the decision-making process.

In the following, we first distinguish DPRMs from other formalisms using the criteria (a)-(e) to highlight its fit for real-world use. Subsequently, we present related work connected to the contributions of this work, i.e., refinements and extensions of DPRMs for real-world use.

In general, DPRMs benefit from several other popular models that have emerged in the field of temporal probabilistic relational models, in particular, parameterised factor graphs (Poole, 2003), Markov logic networks (MLNs) (Richardson and Domingos, 2006), and first-order partially observable Markov decision processess (FO-POMDPs) (Wang and Schmolze, 2005; Sanner and Kersting, 2010). A (non-FO) partially observable Markov decision process (POMDP) generalises a Markov decision process by maintaining a probability distribution over the set of all possible states. Solving a POMDP involves calculating a policy that specifies an expected optimal action to be performed for each state an agent believes it is in. The optimal action maximises the expected reward of the agent over a given horizon. As such, POMDPs enable sequential decision support (Item e) in probabilistic scenarios (Item a), with temporal behaviour implicitly reflected in POMDPs (Item b). Nevertheless, algorithms to solve POMDPs are executed offline, meaning that best actions for all possible states are computed prior to execution in the form of a policy and therefore do not allow for marginal queries (Item d). Additionally, large domain sizes are not addressed (Item c). The first-order variant of POMDPs, i.e., FO-POMDPs, builds on relational logic and allows problems to be represented in a more compact manner (Wang and Schmolze, 2005; Sanner and Kersting, 2010), Item c. Propositional temporal models like dynamic Bayesian nets, dynamic factor graphs, Markov Random Fields cater for temporal behaviour by hav-

ing a model template copied for each time steps with randvars of a time step t connected to randvars of the previous time step $t-1$. They represent the state space in a factored form, with Bayesian nets being a directed formalism and the other two undirected ones. Such formalisms represent probabilistic, relational scenarios including temporal behaviour (Item a and b). Variable elimination (VE) (Zhang and Poole, 1994) is an algorithm for exact inference, enabling query answering (Item d) in them. Still, the propositional modelling formalisms do not cater for sequential decision making (Item e) and do not handle large domain sizes very well (Item c). Probability and utility theory make up decision theory, in which probabilistic models are extended with action and utility nodes. The MEU principle states that a rational agent should choose an action in the current state that maximises its expected utility based on the probability of reaching an outcome state. A utility function is used to rate the desirability of states. An agent selects those actions over time, triggering transitions from a current state to the next, leading to the maximum expected utility. These models still do not cater for large domains (Item c) and in general, inference based on the model is intractable (Koller and Friedman, 2009). The fact that inference is intractable becomes especially apparent for large domains, i.e., in models with many randvars, since the answer to a query can no longer be solved in a reasonable amount of time or with the available resources. Lifted inference approaches reduce computational effort by performing inference using representatives for sets of indistinguishable randvars, which allows for tractable inference w.r.t. domain size (Niepert and Van den Broeck, 2014). Parameterised probabilistic dynamic decision models (PDDecMs), an extension of DPRMs, provide a formalism based on factor graphs to compactly represent a probabilistic scenario with many entities in relation to each other and incorporate actions and utilities to support decision making (Gehrke *et al.*, 2019). In contrast to PDDecMs, MLNs (Richardson and Domingos, 2006) are another type of a probabilistic relational modelling language that uses weighted rules that specify a probabilistic relationship of varying strength or importance, defined by weight, between different variables in the model, fulfilling Items a, c, and d. There exists a dynamic variant (Kersting *et al.*, 2009) and a decision-theoretic variant (Nath and Domingos, 2009), but no combination of both, and therefore no inference algorithm handling both aspects in a joint effort. Additionally, we do not want to impose any structure on the relationship between entities using rules, while factor-based formalisms allow for keeping the relationship more vague. For PDDecMs, the maximum expected utility lifted dynamic junction tree (meuLDJT) algorithm (Gehrke *et al.*, 2019) efficiently solves lifted MEU problems while also answering marginal queries (Item a to e).

Just as each of the previous models solves a specific challenge, an DPRM combines many of these challenges and thus fulfils the characteristics mentioned at the beginning. Nevertheless, DPRMs together with lifted inference approaches still have research gaps in the application in complex real-world environments. The dynamics of the real world itself require the encoding for such dynamics in different properties of the model such as domain

sizes, symmetry structures and probability distributions over time.

To further improve DPRMs for real-world use, we extend the formalism with

a) dynamically changing sets of domain objects over time.

Based on the extended formalism, we

b) introduce an approach to prevent groundings a priori by approximating temporal symmetries that allow for intermediate deviations, and

c) present two approaches to detect changes in symmetries and probability distributions requiring to adjust the model setup.

As follows, we review related work in each respective area.

Dynamically Changing Sets of Domain Objects.

The sizes of sets of domain objects determine the overall joint distribution behind a DPRM. While for different points in time, information about objects may differ, the set of objects under consideration is the same for all time points as per the current formalism of DPRMs. Research about the effect of changing sets of domain objects on overall probability distributions of static relational models has provided partial solutions. One major solution is the concept of *projectivity*, which Shalizi and Rinaldo (2013) introduce in statistical theory. In the context of domain sizes, models are projective, if the model's full joint distribution keeps being valid with a change in the domain, i.e., if an agent is able to accurately make predictions about new entities that have not been present in the training data. In particular, Jaeger and Schulte (2018) show that relational Bayesian networks (RBNs) and MLNs, among others, remain projective when the instantiated (grounded) models inherit independent sub-networks. For non-projective models, Jain *et al.* (2010) propose adaptive MLN describing dynamic model parameters, such as domain sizes, by a set of functions. The authors focus on how changes in the domain size affect probabilities assigned to relations of entities and propose how to adjust distributions accordingly. In addition, Weitkämper (2021) presents scaling of MLN weight parameters with changing domain sizes for MLNs. Poole *et al.* (2014) perform a more detailed analysis on scaling distributions, which was later refined by Mittal *et al.* (2019). Indeed, the notion of non-stationarity has been discussed quite often in the literature. However, non-stationarity as found in the literature refers to changes in distributions (and corresponding factors in a model). To account for varying domain sizes in probabilistic temporal relational models, we extend DPRMs to include changing domain objects as part of the formalism. Adding new entities to an existing domain may result in an expansion of the probability space. For this reason, we introduce strategies that take into account the safe addition to the domain.

Approximate Symmetries to Retain a Lifted Representation.

For static (non-temporal) models, Singla *et al.* (2014) propose to approximate model symmetries as part of the construction process of the lifted network based on historical evidence as training data. They perform lifted belief propagation (LBP) (Singla and Domingos, 2008), which constructs a lifted network, and apply belief propagation (BP) to it. The lifted network is constructed by simulating message passing and identifying nodes sending the same message. To approximate the lifted network, message passing is stopped at an earlier iteration to obtain an approximate instance. Ahmadi *et al.* (2013) also approximate symmetries with LBP, but propose piece-wise learning (Sutton and Mccallum, 2009) of the lifted network. The entire model is divided into smaller sub-models, whose probability distributions are learned from historical evidence independently and then combined again into the overall model. In this way, historical evidence only affects the factors in each part, yielding a "more liftable" model. In addition to approaches that use LBP, Venugopal and Gogate (2014) propose evidence-based clustering to determine similar groundings in an MLN. They measure the similarity between groundings and replace all similar groundings with their cluster centre to obtain a domain-reduced approximation. As the model becomes smaller, inference in the approximated lifted MLN is also much faster. Van den Broeck and Darwiche (2013) propose so-called over-symmetric evidence approximation by performing low-rank boolean matrix factorisation (BMF) (Miettinen *et al.*, 2006) on MLNs. Evidence is represented in the form of a matrix, which is decomposed by low-rank BMF into the product of two or more lower-dimensional matrices. Simply put, finding a low-rank BMF corresponds to removing noise and repetitive information from the data, here the evidence matrix, resulting in a more compact representation. The idea is to use the approximated evidence instead of the raw evidence for inference, which allows for more efficient inference as it maintains "more symmetries" in the model and allows for a "more liftable" solution. As with any existing approach to symmetry approximation, inference is performed on the "symmetrised" model, ignoring the introduction of potentially spurious marginals in the model. Van den Broeck and Niepert (2015) propose a general framework that provides improved probability estimates for an approximate model. Here, a new proposal distribution is computed using the Metropolis-Hastings algorithm (Metropolis *et al.*, 1953; Hastings, 1970) on the symmetrised model to improve the distribution of the approximate model. The approach can be combined with any existing approaches to approximate model symmetries.

Most of the existing research is based on static models and requires getting evidence in advance. However, the problem of asymmetric evidence is particularly noticeable in temporal models, and even more so in an *online setting*, since evidence becomes available only at runtime and therefore performing symmetry approximation as part of the lifted network construction process is not feasible (Nath and Domingos, 2010). That implies that it is necessary to construct a lifted temporal model once and to encode evidence as it comes in over time. However, continuous relearning, i.e., reconstructing the lifted temporal model

before performing query answering, is too costly. For temporal models, Gehrke *et al.* (2020) propose creating a new lifted representation by merging groundings introduced over time. They perform clustering to group sub-models and perform statistical significance checks to test if groups can be merged. However, so far, no one has focused on preventing groundings before they even occur. To this end, we propose to learn entity behaviour in time and cluster entities that behave approximately similarly over time into clusters with similar behaviour, which we refer to in the remainder as *symmetry clusters*. We use symmetry clusters to accept or reject incoming evidence to prevent the model from grounding while accepting a small loss in predictive accuracy due to the approximation. Clustering entity behaviour requires approaches for finding symmetries in entity behaviour, i.e., clustering entities which tend to behave the same according to observations made for them.

Environmental Changes in Symmetry and Distributions.

Changes in symmetries and changes to probability distributions are related to each other. In the model construction process of a DPRM, domain objects being symmetrical to each other and their corresponding probability distributions are determined based on historical data. Thus, the identification of symmetrical structures within the model, as well as the probability distributions encoded in the model, are derived from the patterns and relationships observed in historical data, i.e., data composed of events (related to variables of the DPRM) that have been observed in the past. Sets of symmetrical domain objects are then encoded within the model by assigning probability distributions for each set of objects that best describes the likelihood of possible events to occur (w.r.t. variables of the DPRM). Over time, different domain objects may continue to behave the same in contrast to those which behaved the same at the point of setting up the model. Then a change in the symmetry structures has occurred, e.g., due to a change in the environment. In addition, the probability distribution describing the likelihood of events has most likely also changed.

Detecting changes in symmetry structures or drifts towards new probability distributions is necessary to maintain the accuracy of the model. For drifts of probability distributions, Robinson and Hartemink (2009) have introduced non-stationary dynamic Bayesian networks which allow an underlying conditional dependency structure to change over time. They explicitly encode drifts within the model by defining sets of edges to be changed. That is, edges are added or deleted to adjust conditional dependencies over time, thus defining different manifestations of the model in advance. Choi *et al.* (1999) define hidden-mode Markov decision processs (HM-MDPs), in which changes are limited to a fixed and known number of modes. Each mode represents a stationary environment, formalised as an Markov decision process (MDP). The modes are part of a hidden Markov model (HMM), encoding when the process switches from one mode to another. Hadoux *et al.* (2014) also treat non-stationary environments within Markov models by a set of different contexts, i.e., modes encoded as MDPs. This involves detecting changes in the transition and reward func-

tions of an MDP and using hypothesis tests to test whether a drift has occurred. Note that observations are assumed to be independent and identically distributed (iid). If the hypothesis test reveals that the observations no longer match the MDP, a new MDP is learned. Nevertheless, no one has yet examined concept drifts in factorised (relational) models without knowing the new process in advance. Further, no one has yet looked at symmetry change detection to understand symmetries ahead of query answering.

1.2 Research Objectives and Scientific Contributions

In Section 1.1, we have presented work on the field of relational probabilistic models, lifted inference, and related areas. We have provided an overview of efforts being made to address issues that constrain DPRMs and lifted inference approaches for real-world use. Compared to research in recent years, the focus of this work is on formalisms and mechanisms for adapting to environmental changes. We summarise the contributions as follows:

1) **Investigate Matureness in an Example Application:** We examine requirements and characteristics for explainable decision support using an example application from dry-bulk shipping as a representative for real-world problems. Throughout all contributions, we continue to use the dry-bulk shipping example to address the open challenges identified.

    a) **Construct Reference Application:** We introduce our example application from dry-bulk shipping, and based on that example derive requirements and characteristics of applications in practice.

    b) **Carry out Empirical Evaluation:** On DPRMs, the lifted dynamic junction tree (LDJT) algorithm provides efficient lifted inference. We compare lifted inference with ground inference by applying the UUMLN, short for University of Ulm Markov Logic Networks, the algorithm implemented by Geier and Biundo (2011), which works on a lifted model, but performs its calculations at a ground level. We show that lifted calculations have an advantage over ground calculations.

    c) **Define Open Challenges:** Given the model we set up for the dry-bulk shipping example, we identify three major challenges that limit DPRMs and lifted inference for real-world use. The first challenge, namely the impact of non-stationary processes, which we later refine into (a) dynamic domain sizes, (b) dynamic symmetry structures, and (c) dynamic probability distributions, is the challenge that this work focuses on and serves as motivation for all subsequent contributions.

2) **Deal with Dynamically Changing Sets of Domain Objects:** We conceptualise allowing for changing domain sizes in relational temporal models as a specific form of non-stationarity. For this, we provide semantics for dynamically changing sets of domain

objects for DPRMs taking into account handling of evidence encoded in the model.

a) **Define the Open Domain Challenge:** We describe why adding entities to an existing domain leads on the one hand to an expansion of the probability space and a shift in marginal distributions and on the other hand to the challenge of adopting the information already encoded in the model.

b) **Describe Semantics:** We extend DPRMs to enable dynamically changing domains and introduce domain change operators for this purpose.

c) **Introduce Model Management:** We provide a formalism for changing domain objects and discuss how historical knowledge can be applied and transferred to new domain objects. We propose strategies that take into account support for improving query answering accuracy by transferring knowledge already encoded in the model to new domain objects.

3) **Approximate Temporal Symmetries for Retaining a Lifted Representation:** DPRMs exploit symmetries that are at risk to break in any temporal setting, i.e., even if entities tend to behave the same in the long run, temporary deviations are most likely to occur over time. We approximate the behaviour of entities over longer time periods, ignoring temporary deviations to yield sets of entities with symmetrical behaviour. By understanding entity behaviour a priori, the same can be used to prevent groundings by dismissing negligible evidence that would lead to a grounding.

a) **Symmetry Approximation and Clustering:** We introduce multivariate ordinal pattern symbolisation for symmetry approximation (MOP_4SA), an approach to determine temporal symmetries across entities of the model domains. For this purpose, we use a multivariate ordinal pattern symbolisation approach and build a similarity graph for clustering to identify sets of entities with symmetrical behaviour regarding different contexts of the model (symmetry clusters).

b) **Preventing Groundings:** We introduce symmetry approximation for preventing groundings (SA_4PG), an approach that uses symmetry clusters as part of query answering to avoid unnecessary groundings that would result from events that contradict other events observed for domain objects that actually behave in the same way. By dismissing or inferring events, we reduce the number of unnecessary groundings to retain a lifted representation a priori.

c) **Empirical Evaluation:** Based on the example from dry-bulk shipping, we perform an extensive evaluation of MOP_4SA together with SA_4PG and show that both help to successfully prevent groundings a priori. At the same time, inference can be sped up while only introducing a considerably small error.

4) **Detect Environmental Changes in Symmetries and Distributions:** Changes in symmetry structures and probability distributions, in the literature the latter is also referred to as concept drift, define two forms of non-stationarity in DPRMs. To efficiently detect environmental changes in the model, we make use of the factorisation and compact encoding of relations in DPRMs.

   a) **Concept Drift Detection:** We introduce an efficient concept drift detection algorithm exploiting the factorisation and compact encoding of relations in DPRMs to identify points at which a drift has settled, in order to relearn the model once needed. Since concept drift only manifests itself over time steps, we determine the points in time at which a new probability distribution has settled.

   b) **Symmetry Change Detection:** We introduce multivariate ordinal pattern symbolisation for symmetry change detection (MOP_4SCD) to detect changes in symmetry structures based on a models similarity graph, an intermediate step of MOP_4SA, to re-learn symmetries once needed.

1.3 Structure

After this introduction, **Chapter 2** starts with a real-world example from dry-bulk shipping in which we investigate the matureness of DPRMs. The example serves as a reference for other applications in practice. We demonstrate the potential of lifting, including a more formal introduction to exploiting computational symmetries, which is related to identifying sub-isomorph graph structures, explain why DPRMs together with lifted approaches are particularly well-suited for real-world problems and discuss open challenges and limits (Contribution 1a-c).

The content of this first section is published in

> Nils Finke, Marcel Gehrke, Tanya Braun, Tristan Potten, and Ralf Möller. Investigating Matureness of Probabilistic Graphical Models for Dry-Bulk Shipping. In Manfred Jaeger and Thomas Dyhre Nielsen (Eds.), *Proceedings of the 10th International Conference on Probabilistic Graphical Models*, volume 138 of *Proceedings of Machine Learning Research*, pages 197–208. PMLR, 2020.

The example is followed by **Chapter 3** on preliminaries, which give a formal description of DPRMs. After this first example and preliminaries, the main part of this work begins, in which we present three challenges limiting DPRMs for real-world use. We address each challenge as a single part of this work.

In the **first part**, we extend the semantics of DPRMs for dynamically changing sets of domain objects to be substituted for variables used in the model, such that sparse domain models can be specified and efficient query-answering algorithms can be provided for practically

relevant application domains. In **Chapter 4**, we present the characteristics and challenges associated with domain size changes, in particular the possible expansion of the probability space (Contribution 2a). In **Chapter 5**, we extend DPRMs to support dynamically changing domains and introduce strategies to safely handle new entities considering the extension of the probability space (Contribution 2b-c). **Chapter 6** concludes the first part with an interim conclusion and provides motivation for the second part of this work to better understand the temporal behaviour of entities. The contents of the first part were mainly published in the following conference paper.

Nils Finke, Tanya Braun, Marcel Gehrke, and Ralf Möller. Dynamic Domain Sizes in Temporal Probabilistic Relational Models. *The International FLAIRS Conference Proceedings*, 34, 2021.

In the **second part**, we address the problem of symmetries that suffer under the dynamics of the real world, so that any lifted representation of a temporal model is counteracted.

In **Chapter 7**, we first discuss why it is not sufficient to assume, as in current research, that symmetries are present in the data, and why one should therefore "construct" symmetries during the model construction process. Finding temporally exact symmetric behaviour of entities is computationally complex, but not always necessary in real-world applications. Further, in **Chapter 7**, we introduce MOP_4SA, an approach to approximate symmetries in (multivariate) time series based on a symbolisation scheme that encodes the overarching trend in up and down movements (Contribution 3a). In **Chapter 8**, we present SA_4PG, an approach that uses the symmetry structures identified by MOP_4SA to prevent groundings in lifted models a priori (Contribution 3b) and conclude the second part of this thesis with an empirical evaluation of MOP_4SA and SA_4PG (Contribution 3c) using the dry-bulk shipping example that we introduce in Chapter 2. We conclude the part with an interim conclusion in **Chapter 9**. The second part is based on the following four publications, the first introducing MOP_4SAs and SA_4PG for the univariate case, the second MOP_4SAs for the multivariate case, the third reviewing the expressiveness of multivariate ordinal patterns for any classification task, and fourth providing an extended summary of MOP_4SAs including an in-depth experimental analysis.

Nils Finke and Marisa Mohr. A Priori Approximation of Symmetries in Probabilistic Dynamic Relational Models. In Stefan Edelkamp, Ralf Möller, and Elmar Rueckert (Eds.), *KI 2021: Advances in Artificial Intelligence*, pages 309–323. Springer, 2021.

Nils Finke, Ralf Möller, and Marisa Mohr. Multivariate Ordinal Patterns for Symmetry Approximation in Dynamic Probabilistic Relational Models. In Guodong Long, Xinghuo Yu, and Sen Wang (Eds.), *AI 2021: Advances in Artificial Intelligence*, pages 543–555. Springer, 2022.

Nils Finke, Marisa Mohr, Alexander Lontke, Marwin Zünfle, Samuel Kounev, and Ralf Möller. Recommendations for Data-Driven Degradation Estimation with Case Stud-

ies from Manufacturing and Dry-Bulk Shipping. In Samira Cherfi, Anna Perini, and Selmin Nurcan (Eds.), *Research Challenges in Information Science - 15th International Conference, RCIS 2021*, volume 415 of *Lecture Notes in Business Information Processing*, pages 189–204. Springer, 2021.

Nils Finke and Ralf Möller. On the Approximation of Symmetries and Related Structural Changes in Dynamic Probabilistic Relational Models. *Advances in Science, Technology and Engineering Systems Journal*, 7(2), 2022.

In the **third part**, we introduce two mechanisms for detecting changes reflected in observations/evidence. Specifically, the mechanisms identify points in time when the DPRM no longer accurately describes reality. We first delineate different types of environmental changes in DPRMs in **Chapter 10**, followed by two chapters in which we examine one specific form of structural change at a time. In **Chapter 11**, we introduce an efficient detection principle for concept drifts using the factorisation and relations of a DPRM, i.e., a drift in the underlying probability distribution of the model (Contribution 4a). In **Chapter 12**, we present MOP_4SCD for symmetry structure change detection (Contribution 4b). In Contribution 3b, we use symmetry structures to determine unknown entity behaviour, i.e., we derive evidence based on observations from other entities, to prevent groundings. MOP_4SCD complements MOP_4SA and SA_4PG since both rely on the validity of symmetry structures in the model. We end the part in **Chapter 13** with an interim conclusion. This third part was mainly published in the following two conference papers. While in the first paper we introduce the detection of concept drifts, in the second paper/journal article, which I have already cited, we introduce the detection of changes in the symmetry structure, in addition to the topics relevant to the second part of this work.

Nils Finke, Tanya Braun, Marcel Gehrke, and Ralf Möller. Concept Drift Detection in Dynamic Probabilistic Relational Models. *The International FLAIRS Conference Proceedings*, 34, 2021.

Nils Finke and Ralf Möller. On the Approximation of Symmetries and Related Structural Changes in Dynamic Probabilistic Relational Models. *Advances in Science, Technology and Engineering Systems Journal*, 7(2), 2022.

All parts end with an interim conclusion, while **Chapter 14** concludes with an overall summary of contributions and ends with broad future work for DPRMs and their abilities for real-world use.

2 Investigating Matureness of DPRMs for Real-World Use

Before giving a formal introduction to DPRMs, we take a step back and highlight the effectiveness of DPRMs and lifting in real-world applications using a motivational example from dry-bulk shipping. This chapter provides contributions 1a–c of this work. While giving an introduction to our example application, which serves as a representative for other real-world challenges, in **Section 2.1**, we outline characteristics and requirements of real-world applications that need to be covered by a suitable model. In **Section 2.2** we highlight the potential of lifting by elaborating more on exploiting symmetries and comparing ground versus lifted inference approaches. Finally, in **Section 2.3** we determine three major challenges yet limiting the use of DPRMs for real-world applications, namely non-stationary processes, scalability in lifting and its action modelling. A summary of our reference application from dry-bulk shipping, based on which we derive characteristics and requirements for real-world use, is first presented in the following conference paper.

Nils Finke, Marcel Gehrke, Tanya Braun, Tristan Potten, and Ralf Möller. Investigating Matureness of Probabilistic Graphical Models for Dry-Bulk Shipping. In Manfred Jaeger and Thomas Dyhre Nielsen (Eds.), *Proceedings of the 10th International Conference on Probabilistic Graphical Models*, volume 138 of *Proceedings of Machine Learning Research*, pages 197–208. PMLR, 2020

2.1 An Example From Dry-Bulk Shipping

As part of the global supply chain, dry-bulk shipping is one of the most important forms of transportation (Başer and Açık, 2019). With the growth of the world economy, the need for seaborne transportation of dry-bulk commodities has increased over the years. Especially the last years (2020-2022), which were marked by the coronavirus pandemic, represented the importance of proper supply chain management. The global supply chain was heavily affected by the lockdowns required all over the world, causing disruptions and significant delays in deliveries. We motivate and discuss two different interdependent sub-challenges in dry-bulk shipping, namely (a) positioning of vessels according to supply and demand, and (b) congestion avoidance to prevent delays in delivery. Both have always been impor-

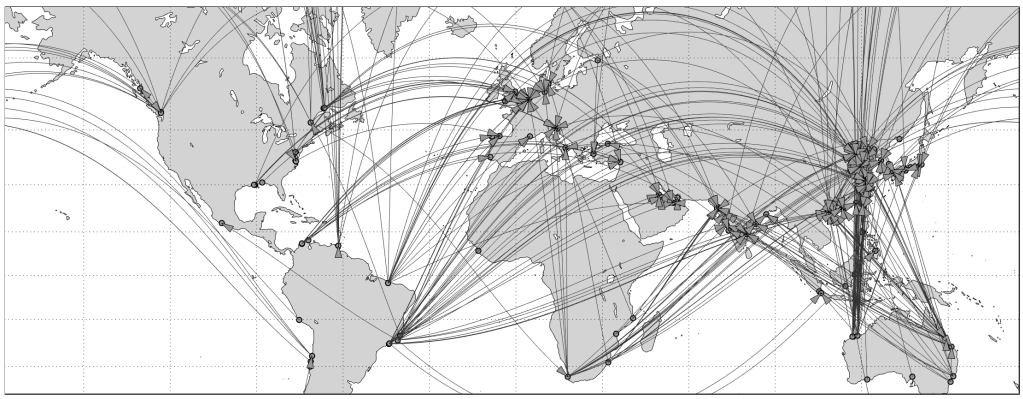


Figure 2.1.: Example of the transportation of iron ore, coal and grain of an anonymous shipping company for the year 2020.

tant topics in research, e.g., for vessel positioning see the works of Michail and Melas (2021) and Peng *et al.* (2016), and for congestion avoidance see the works of Wang *et al.* (2021), Jiang *et al.* (2017), and Notteboom (2006).

Dry-Bulk shipping is about the transportation of dry-bulk commodities, i.e., non-liquid, unpacked goods that have undergone little to no processing, such as iron ore, coal, or grain. Typically, these commodities are transported from their mining or production regions to different places around the world for further processing. In contrast to liner shipping, where individual containers can be rented on a fixed route with fixed schedules, it is common in bulk shipping to transport bulk goods on individual routes on behalf of just one customer. Companies operate vessels and cargo suppliers negotiate contracts individually. The dry-bulk shipping industry is closely related to the commodity industry. Commodity trade flows emerge based on exports and imports as natural resources (commodities) are unevenly distributed over the earth. Vessels perform voyages representing these trade flows by transporting cargo from exporting nations to importing nations. Dry-bulk shipping, therefore, has a strong geospatial component: Vessels travel between ports, and ports are located in countries, which are located on continents. However, trade flows are not evaluated based on the landmass. Instead, they are analysed from an ocean-centric perspective by splitting the world into different *regions/zones*, e.g., Western and Eastern Australia or North, Central, and South Pacific. To give another more specific example: The ports of Vancouver and Seven Islands are both in Canada, but one is located on the west coast (Pacific) and the other on the east coast (Atlantic). For illustration purposes, Figure 2.1 depicts shipping operations to transport iron ore, coal and grain of an anonymous shipping company for the year 2020. The figure clearly depitcs that trade flows are imbalanced, as China, for example, has the highest demand for unprocessed goods.

For the transportation of cargoes, a fee per ton, the so-called freight rate, is charged.

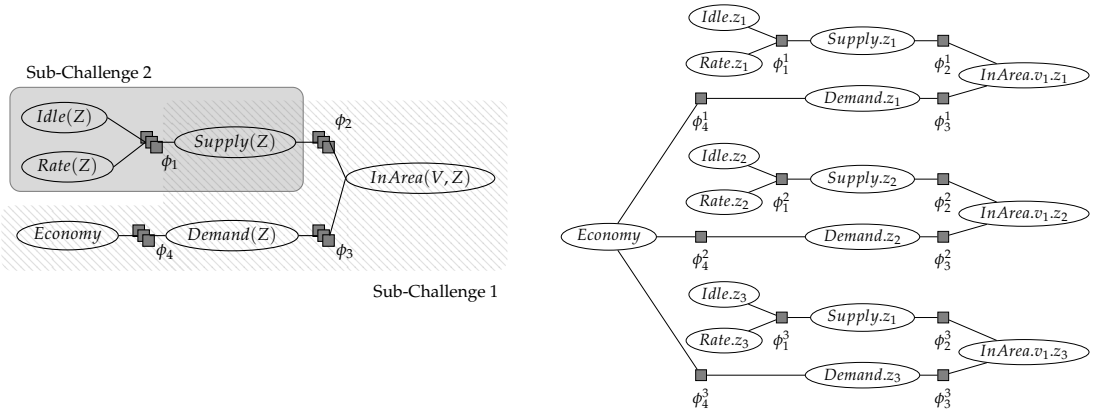
Freight rates are negotiated between shipping companies and cargo suppliers. Thus, freight rates are dynamically priced and influenced by conditions in the regions. Therefore, vessel owners try to position their vessels in zones with high supply to load cargo to benefit from good freight rates being paid. Supply is driven by demand, i.e., vessels end up in zones with high demand to discharge cargoes. Demand itself varies in response to fluctuations in the global economy. In phases of expansion, i.e., during growth, demand for goods is generally higher than in phases of contraction, i.e., during a recession.

Another challenge for vessel owners is to recognise when export regions become supersaturated due to too many vessels being on-site, requiring them to adjust the positioning of their vessels accordingly. Therefore, operators need to keep track of the export and import behaviour of countries on the one hand and of vessels from other vessel owners being operated on the other hand. Additionally, operators face the challenge of overlooking a market with several operators, e.g., with already over ten thousand dry-bulk vessels as of the year 2019 (United Nations Conference on Trade and Development, 2019).

The dry-bulk shipping market is a volatile environment characterised by uncertainty as decisions of participators on the market, such as cargo suppliers and owners of vessels, are not transparent to the market and are driven by each participator's own belief on how the market is evolving. From the perspective of a company operating vessels on the market, it is of great interest to know how other operators position their fleet, how trade flows will change over time, and how to position its vessels to meet demand and cover costs given all these uncertainties. In summary, the reference example of dry-bulk shipping has the following characteristics and requirements: (a) Encode probabilistic, relational structures, i.e., dependencies between objects that are subject to uncertainties, (b) afflicted with temporal behaviour, (c) for a large domain in a compact manner. Further, (d) provide efficient marginal query answering, and (e) sequential decision support, (f) under time constraints. Figure 2.2a represents a simplified graphical representation of the dry-bulk shipping market with two sub-challenges, which we describe as follows:

(1) The first challenge is to position vessels V in zones Z of high supply to load cargo and in zones Z of high demand to discharge cargo. The nodes and edges framed by the dashed rectangle describe the interplay of vessel positions in zones $InArea(V, Z)$ according to supply $Supply(Z)$, and demand $Demand(Z)$ within zones, wherein $Demand(Z)$ varies depending on the global economic situation $Economy$.

(2) The second challenge is to take idle times into account as high supply leads to more vessels being on site which causes waiting times due to congestion. Nodes and edges framed by the grey rectangle describe the interplay between supply $Supply(Z)$ and freight rates $Rate(Z)$ as drivers for idle times $Idle(Z)$ within zones. Note that congestion can also occur in zones where cargo is discharged, which is not considered here for simplicity.



(a) *Lifted* graphical representation to describe supply and demand on the dry-bulk shipping market.

(b) Corresponding *grounded* graphical representation of Fig. 2.2a for a selection of zones and one vessel, here zones z_1, z_2, z_3 and vessel v_1.

Figure 2.2.: Simplified representation of challenges in the dry-bulk shipping market.

2.2 A Deeper Understanding of Symmetries and the Potential of Lifting

The potential of lifting depends on the availability of symmetries in the data, which we illustrate using the previous example. We first give an example and its corresponding grounded representation, then explain the identification of symmetries by identifying sub-isomorphic graph structures in the grounded graphical representation, and finally demonstrate the transformation of the grounded representation into a lifted representation by exploiting identified symmetries.

Grounded Representation

In Fig. 2.2b each node represents a randvar encoding uncertainty. All dependent randvars are connected through a factor node ϕ which, informally speaking, holds a joint probability distribution of all connected randvars. More specifically, the model encodes probability distributions through a factor node ϕ, here,

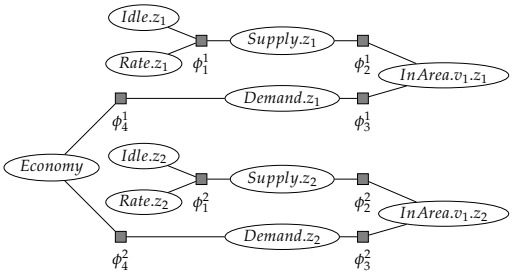
$$\begin{aligned}
&\phi_1^1(Idle.z_1, Rate.z_1, Supply.z_1), \phi_2^1(Supply.z_1, InArea.v_1.z_1),\\
&\quad \phi_3^1(InArea.v_1.z_1, Demand.z_1), \phi_4^1(Demand.z_1, Economy),\\
&\phi_1^2(Idle.z_2, Rate.z_2, Supply.z_2), \phi_2^2(Supply.z_2, InArea.v_1.z_2),\\
&\quad \phi_3^2(InArea.v_1.z_2, Demand.z_2), \phi_4^2(Demand.z_2, Economy),\\
&\phi_1^3(Idle.z_3, Rate.z_3, Supply.z_3), \phi_2^3(Supply.z_3, InArea.v_1.z_3),\\
&\quad \phi_3^3(InArea.v_1.z_3, Demand.z_3), \phi_4^3(Demand.z_3, Economy).
\end{aligned} \tag{2.2.1}$$

As the number of domain objects increases, here zones z_i and vessels v_j, the dependencies and relations in the graph depicted in Fig. 2.2b also increase. Likewise, as the number of domain objects in the model increases, so does the likelihood of *repetitive information* being present in the model. Repetitive information in our example application for route planning can exist in terms of zones that are similar w.r.t. to the features of the model. More specifically, repetitive information can be zones offering the same commodities, i.e., zones with similar mineral resources, potentially leading to the same distributions encoded in factor nodes for varying zones z_i. In general, challenges arise from the presence of repetitive information. First, repetitive information in a model takes up unnecessary storage space and, second, when reasoning based on the model the same calculations due to the same data may be done again and again. Therefore, it is important to identify and handle repetitive information to improve efficiency both in terms of representing complex models and performing inference.

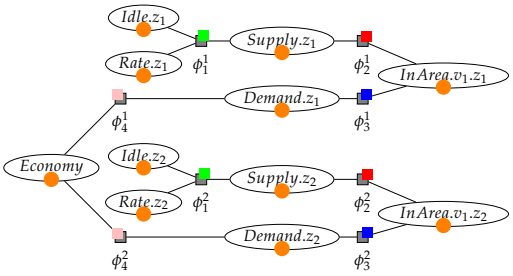
From a Grounded Representation to a Lifted Representation

Repetitive information can be understood as symmetries in the data, which can be exploited in terms of lifting, e.g., to speed up inference. While Fig. 2.2a depicts the general structure of the model, Fig. 2.2a represents a copy of this structure for individual domain objects, here zones z_1, z_2 and z_3 and a vessel v_1. To come up with its lifted representation, i.e., getting from Fig. 2.2b to Fig. 2.2a requires the identification of patterns or symmetries in the ground version of the graph. In this matter, we use the term identifying symmetries as an equivalent for finding *sub-isomorph graph structures* and *identical probability distributions across factor nodes* of a sub-isomorph graph structure. A sub-isomorphic graph structure refers to a smaller graphical model that has the same structure, and possibly identical probability distributions, as part of a larger graphical model. To identify sub-isomorphic graph structures, we use the colouring algorithm refined by Singla and Domingos (2008); Kersting *et al.* (2009); Ahmadi *et al.* (2013).

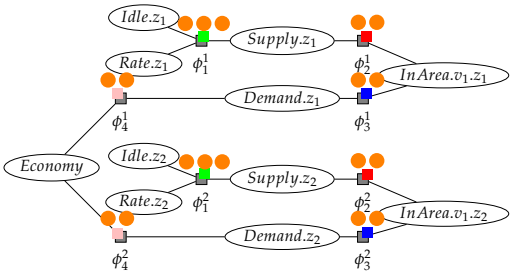
Before we go into the process of determining symmetries and compression of the graph structure using the colouring algorithm, we give the following example for better understanding. Figure 2.3a depicts a grounded version of a probabilistic model for zones z_1 and z_2 and one vessel v_1 as introduced in the previous section. To answer queries based on a graphical model like the one represented in Fig. 2.3a, one can use belief propagation, an inference algorithm used in probabilistic graphical models to compute marginal probabilities. In the belief propagation process, messages are iteratively passed between nodes in the graph, updating beliefs about the marginal probabilities between neighbouring nodes. To answer a query such as $P(Demand.z1 = high|Economy = Expansion)$, which seeks to determine the probability of high demand in zone z_1 given the current economic situation of expansion, messages are exchanged between all relevant nodes in the graph. Using the sum-product algorithm for belief propagation, all nodes except the node $Demand.z_1$ are



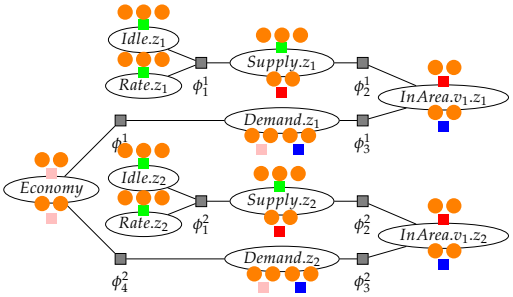
(a) Input of a model for compression (exploiting symmetries and sub-isomorph graph structures).



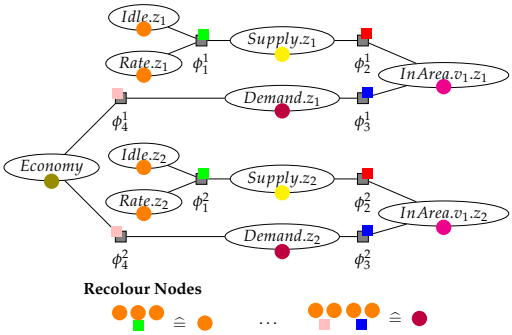
(b) Initialisation Phase



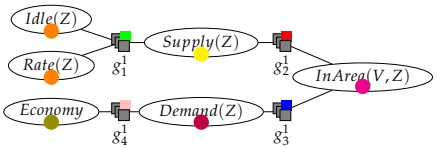
(c) Iterative Colour Refinement: Factors collecting colours from the neighbouring nodes.



(d) Iterative Colour Refinement: Nodes collecting colours from the neighbouring factors.



(e) Iterative Colour Refinement: Recolour nodes with a unique colour depending on the colour signature.



(f) Compressed version of the graph in Fig. 2.3a.

Figure 2.3.: Exploit computational symmetries: The colour passing algorithm for compressing a ground representation to come up with its corresponding lifted representation.

marginalised, resulting in the calculation of marginal probabilities for the query variable. However, assuming that the conditional probability distributions between variables, e.g., *Demand*.z_1 and *Demand*.z_2 given *Economy*, are identical, then the same messages are in this example sent between nodes *Economy* and *Demand*.z_1 and nodes *Economy* and *Demand*.z_2. This is also true for all other nodes that form a sub-isomorphic graph structure.

The concept of determining a compressed graph and passing messages in the compressed graph is useful, provided messages in the grounded graph are the same. To determine the sub-isomorphic graph structures in Fig. 2.3a and therefore determine a compressed version of the graph, we outline the colour-passing algorithm as follows.

1. **Initialisation Phase**: In the initialisation phase, all factors ϕ between nodes are coloured with the same colour if they are considered to be identical, i.e., encode the same distribution such as ϕ_1^1 and ϕ_1^2, and ϕ_2^1 and ϕ_2^2, and ϕ_3^1 and ϕ_3^2, and ϕ_4^1 and ϕ_4^2 as depicted in Fig. 2.3b. All nodes representing a randvar are assigned the same colour.

2. **Iterative Colour Refinement**: In the process of colour refinement, factors and nodes iteratively collect colours from their neighbours as represented in Fig. 2.3c and Fig. 2.3d. First, factors between nodes collect the colours of their neighbouring nodes. Subsequently, the nodes collect the collected colours from associated factors, which we refer to as a colour signature, of their neighbouring factors. Then, each unique colour signature of a node is replaced by one unique single colour and factors ϕ get the same colours as in the initialisation phase, as depicted in Fig. 2.3e. This process is repeated until no new colour is generated and, therefore, the algorithm converged to a stable assignment of colours.

3. **Sub-Graph and Symmetry Identification**: Once no new colour signatures are generated, the nodes and factors with the same colour are merged into a common node, resulting in a compressed graph as represented in Fig. 2.3f.

The main idea behind this approach is that nodes and factors with similar neighbourhood structures are more likely to belong to isomorphic sub-graphs. The colour passing algorithm refines colours based on local connectivity, implying that the new colour for a node depends only on the colours of its immediate neighbours. At each iteration, the colour of a node or factor incorporates information from the colours of its neighbours and therefore, as the algorithm progresses, the colours begin to capture the connectivity patterns of larger neighbourhoods. Exact symmetries in factors are exploited as they are getting the same colour if factors are considered equivalent. Next, we elaborate on the compressed version of the original graph, which is considered as a lifted representation.

Lifted Representation

Using the colour-passing algorithm, we are able to determine a compressed, also called lifted, version of the grounded probabilistic graphical model as depicted in Fig. 2.3f. More specifically, each randvar can be parameterised with multiple logvars, e.g., the randvar name $Supply$ is parameterised with a logvar Z yielding a parameterised randvar $Supply(Z)$. Thus, a parameterised randvar represents a set of objects, e.g., the randvar $Supply$ refers to several objects $z_1, z_2, \ldots$ from the set of all objects Z to come up with a sparse representa-

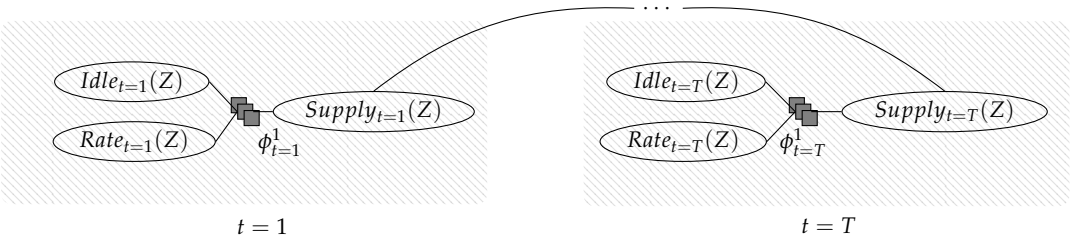


Figure 2.4.: Temporal probabilistic relational model for sub-challenge 2.

tion. This is possible if there are symmetries in the data, i.e., if the values of supply level $Supply(Z)$, idle times $Idle(Z)$ and freight rates $Rate(Z)$ are equal in zones z_1 and z_2 and therefore have the same distribution encoded in factors ϕ^1_1 and ϕ^2_1. Note that we present all detailed definitions in the next chapter.

Conversely, lifting exploits the existence of repetitive information that are revealed by the colouring algorithm: Regions which are symmetrical with respect to the features used in the model can be treated by one representative for a group of symmetrical objects to obtain a sparse representation of the model. This is the case, for example, if in all zones Z the level of ($Supply(Z)$) is the same, resulting in the same freight rates ($Rate(Z)$) paid for transportation, while also leading to similar interest for vessel operators and in conclusion to the same idle times ($Idle(Z)$). Then, all zones can be represented through a single (parameterised) randvar as a representative for all other zones to come up with a lifted representation as depicted in Fig. 2.2a. Both Fig. 2.2a and Fig. 2.2b represent only a static model, i.e., a model for a single time step. Once a temporal dimension is included, the complexity of the model increases further. To account for arbitrary temporal structures, DPRMs, as well as other temporal probabilistic models, adapt a copy pattern by duplicating randvars of one-time step and copying them for all subsequent time steps to unroll a temporal model. Figure 2.4 presents an example of how to encode and unroll a temporal model, which becomes necessary when reasoning about further time steps, i.e., about $t \in 1, \ldots, T$. As a result, the complexity of the model increases further with each time step for which the model is unrolled.

Lifted Reasoning

By exploiting symmetries, the complexity of reasoning (also called inference) in lifted representations is no longer exponential in the number of objects represented by the model, here regions, but is polynomial w.r.t. domain sizes only for a specific class of queries (Niepert and Van den Broeck, 2014; Van den Broeck and Niepert, 2015). Instead of performing the same calculations over and over again for all objects individually, in lifted inference, calculations are performed once for one representative of a group of similarly structured objects,

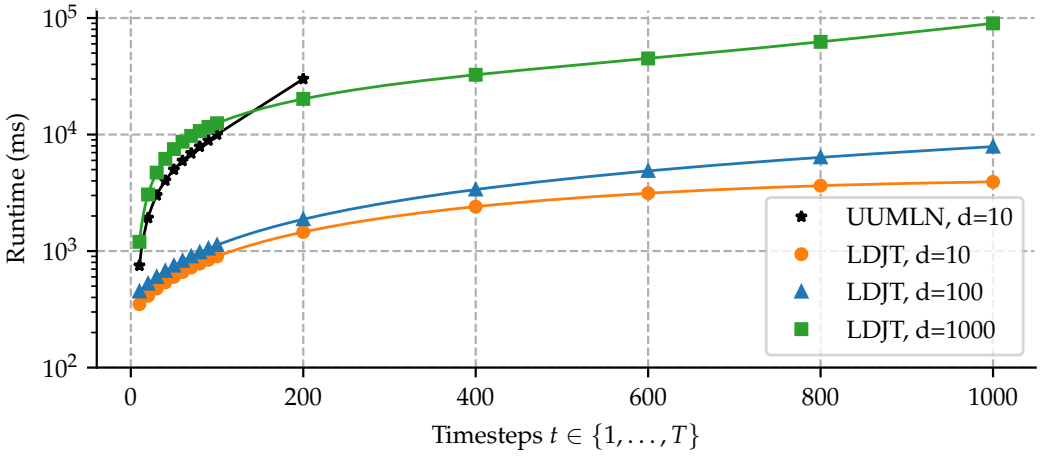


Figure 2.5.: Comparison of ground query answering versus exploiting symmetries. Note that the y-axis is logarithmically scaled.

which results in runtime advantages.

Figure 2.5 depicts a comparison of runtimes between ground inference, i.e., query answering without exploiting symmetries, and lifted inference, i.e., query answering exploiting symmetries. For comparison, we consider the following implementations for lifted reasoning:

- Alchemy[1] for lifted static models,
- Forclift[2] for lifted static models,
- UUMLN[3] for lifted temporal models, but with calculations all performed on a ground level,
- LDJT[4] for lifted temporal models.

Note that the details about the implementations and model setup are not of relevance here. We use Fig. 2.5 only to illustrate runtime differences between different methods.

For each implementation, we perform three different experiments: To illustrate the problem of answering queries in large models, we ask queries based on the model with varying domain sizes. More specifically, in each experiment, we consider a model with a different domain size of $d \in \{10, 100, 1000\}$. We ask the same query to obtain the marginal probabilities for the randvar $Rate_t(z_1)$ for a time step t for a zone z_1. Additionally, we answer the same query for different future time steps t, specifically $t \in \{1, \ldots, T\}$ with $T \in \{10, 20, \ldots, 100, 200, \ldots, 1000\}$. As the first three implementations use (dynamic) MLNs

[1] `http://alchemy.cs.washington.edu`
[2] `https://github.com/UCLA-StarAI/Forclift`
[3] `https://www.uni-ulm.de/en/in/ki/inst/alumni/thomas-geier/`
[4] `https://www.ifis.uni-luebeck.de/index.php?id=483`

as inputs, we transform the DPRM into an equivalent MLN for use in Alchemy, Forclift and UUMLN as described by Van den Broeck (2013).

Figure 2.5 depicts the runtimes in milliseconds for all three domain sizes (10: circles, 100: triangles, 1000: squares) over time T averaged over five runs, i.e., performing the experiment as described above five times with a cutoff time of 3 hours and 16GB of working memory. Since LDJT performs its calculations in a lifted fashion, i.e., exploits symmetries, we expect it to perform better than UUMLN as all computations in UUMLN are done on a ground level. UUMLN fully grounds a lifted model by instantiating parameterised randvars (with its represented objects). Further, we expect that both UUMLN and LDJT perform better than Alchemy and Forclift since Alchemy and Forclift cannot exploit temporal structures as efficiently as UUMLN and LDJT do. Alchemy and Forclift unroll the temporal model for t time steps to get a static model on which a query for a specific time step t can be answered.

Implementations of Alchemy and Forclift run into memory errors even for a model with the smallest domain size of $d = 10$. This confirms our expectation that algorithms for static models do not handle temporal behaviour well. UUMLN can perform the experiment at least on a model with a domain size of $d = 10$ until $T = 200$, then triggering the cutoff time, which shows that lifted calculations have an advantage over ground calculations. Our experiments demonstrate that the LDJT runtimes exhibit a linear dependence on time. The runtimes of LDJT for different domain sizes indicate a polynomial increase since the difference between the lines indicates an almost linear factor. In conclusion, lifted temporal models outperform propositional models. Nevertheless, although DPRMs are a good fit for any complex real-life setting, DPRMs still face some open challenges, which we elaborate on in the following.

2.3 The Challenge of Non-Stationarity in Lifted Models

Even if DPRMs and lifting are successfully applied, they are subject to several assumptions that do not apply in the real world. The efficiency of a lifted inference approach is directly related to the degree symmetries are present in the data that constitute the model. Therefore, handling asymmetries is one of the major challenges in lifted inference and crucial for its effectiveness (Kersting, 2012; Van den Broeck and Darwiche, 2013). Asymmetries arise especially under the dynamics of the real world, e.g., with environmental changes. In the conference paper (Finke *et al.*, 2020b), we identified three main challenges in terms of query answering, namely (a) non-stationary processes, (b) constraints on the liftability of the model, and (c) action modelling. In this work, we do not elaborate further on the second and third challenges. Rather, we focus on different forms of (non-)stationarity that pose an open challenge and constrain DPRMs for real-world use. For probabilistic models, stationarity is commonly understood as that the underlying stochastic process, i.e., the probability distribution, of a temporal model for a time slice is not affected when shifting in

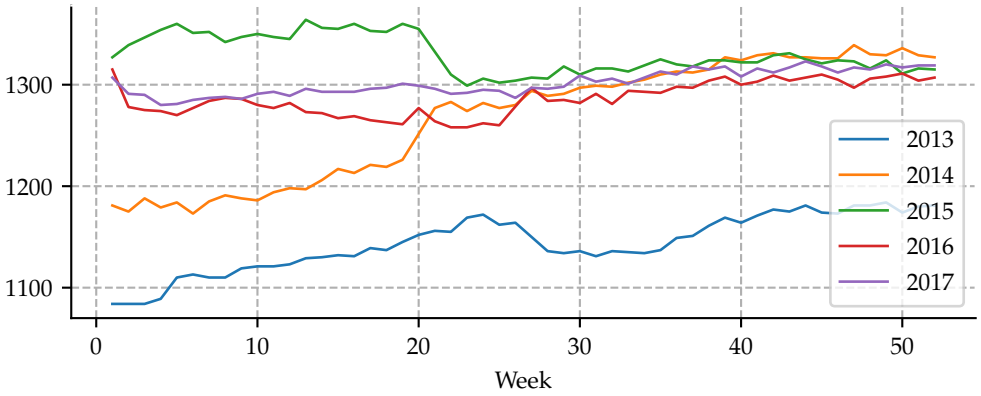


Figure 2.6.: Non-stationary Domain Sizes: Changing vessel counts over time.

time. More specifically, the joint probability distribution does not change over time. Consequently, parameters such as mean, variance or autocorrelation structure also do not change over time. Vice versa, non-stationarity characterises a stochastic process whose joint probability distribution or corresponding parameters are affected by a temporal shift. We do not limit the general understanding of stationarity to joint **probability distributions** but extend it to other properties of a DPRM, such as **domain sizes** and **symmetry structures**. All three aspects can change over time, which limits DPRMs for real-world use. In the following, we address each of the aspects in detail.

Domain Sizes

The sizes of sets of domain objects determine the overall joint distribution behind a DPRM unrolled over time. While the information about objects may be different at different points in time, the set of objects under consideration is assumed to be the same for all points in time in classical DPRMs. Figure 2.6 depicts the number of so called *Capesize* vessels available on the market over time. In particular, the figure indicates that in shipping, vessels continuously join or leave the market. The lines indicate continuous up-and-down changes in the number of vessels due to a variety of different reasons, such as new vessels being built, dumped, going into maintenance, or being laid up in case freight rates do not cover running costs anymore. The challenge of changing domain sizes is w.r.t. handling evidence encoded within the model, i.e., keeping and handling new knowledge that is encoded in the model in the form of new observations that have been collected after the initial model setup. Specifically, the primary challenge upon including a new domain object in an existing domain is to find other domain objects already encoded in the model that have symmetries to the new domain object w.r.t. to the variables of the model. This is important as in the DPRM

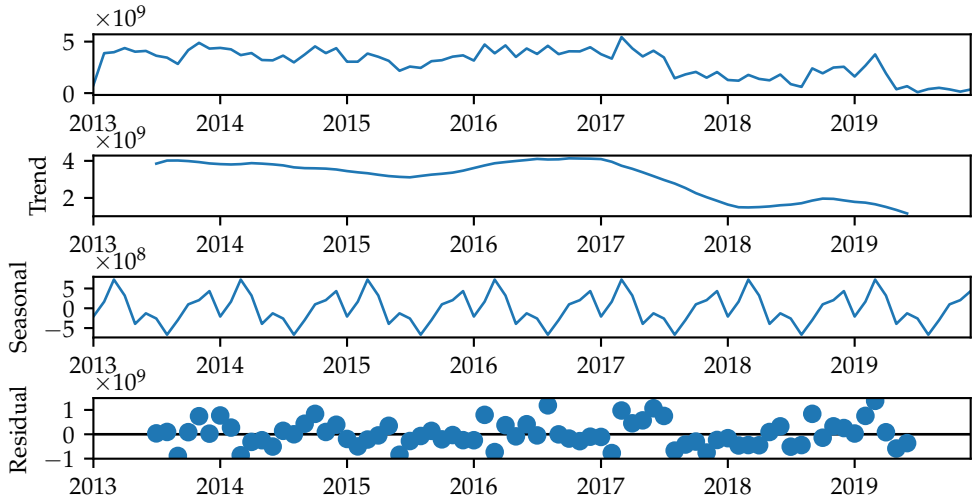


Figure 2.7.: Non-stationary Probability Distributions: Changing demand over time.

domain objects with exact symmetries to each other are represented by a single domain object within its group of domain objects with exact symmetries. Domain objects are encoded in such groups using common factors to describe the likelihood of an event occurring in terms of the model's variables as depicted in Fig. 2.2. Extending DPRMs with dynamic domains while exploiting symmetries is beneficial and important for two reasons. On the one hand, we aim to maintain a lifted representation as much as possible to retain the benefits of a compact representation including being able to perform inference efficiently. On the other hand, we achieve that knowledge already encoded in the model is transferred to the new domain object in order to increase the accuracy in inference.

Probability Distributions

The phenomenon of probability distributions changing over time is also referred to as *concept drifts* in the literature (Widmer and Kubat, 1996). If the underlying initial probability distribution changes over time and the events occurring do not match the expected belief of an agent when answering queries, the accuracy of the model suffers and the results are biased. Thus, the detection and handling of concept drifts are important to ensure the accuracy of an underlying model. To detect potential concept drifts, in classical time series analysis probability distributions are derived from historical data and examined for changes. More specifically, in classical time series analysis, the decomposition of time series is a common approach to uncover reoccurring patterns in temporal data (Hyndman and Athanasopoulos, 2018). In the process, the time series is decomposed in trend, seasonality, and residual

components (see also Fig. 2.7). The first two are also summarised as systematic components, both of which violate stationarity. The latter is called non-systematic and includes what remains in the time series after systematic components are removed. In mathematical or statistical modelling, variables are often added for systematic components to encode all known influencing factors within the model.

In the following, we consider an example from the field of dry-bulk shipping, specifically about shipping coal. Figure 2.7 (top) represents a time series for the demand for coal in Germany between 2013 and 2019. The three plots below represent the decomposition of the time series into its components. The trend component in the second graph represents a general downward trend within the data, which can be explained by the decision of the German government to slowly reduce the use of fossil energy sources (Renn and Marshall, 2016). The seasonality component in the third graph represents a seasonal dependency that can be explained by the demand for heating in winter, as coal is still important for the production of electricity in Germany. The residual component in the fourth graph represents the remainder which can neither be specified by a trend nor by seasonality. When performing inference given new evidence, it is possible that due to unknown effects of non-systematic components (including one-time events altering the distribution), the probability distribution may no longer accurately describe the likelihood of observations (events) occurring. Therefore, it is critical to observe and adapt to any shifts in the residual component in particular or in the probability distribution of the model in general to ensure stationarity.

DPRMs are able to compactly represent complex relational structures that carry temporal information by exploiting symmetries, i.e., to come up with a lifted representation of the same. Moreover, lifted reasoning is made possible, making inference more efficient. This work is motivated by an application example in which exactly such a large complex model is represented. Due to the size of the model, continuous relearning to come up with a new model incorporating new information to ensure that the model still encodes a full joint distribution that accurately describes reality is not practical. In this work, we, therefore, extend LDJT, a lifted inference algorithm on DPRMs, to an online algorithm. However, in order to ensure accuracy in inference, only recreating the model when necessary is essential, i.e., when a concept drift has occurred and settled. Nevertheless, detecting concept drifts in complex models can be challenging simply due to their sizes, however, the concept drift detection process can be accelerated by exploiting the factorisation and lifted representation exploited in the DPRM.

Symmetry Structures

For temporal models, the determination of symmetries is more complicated than for a static model. For example, the determination of symmetries at a certain time step, i.e., in a local model, can be achieved by clustering objects by their observations. Objects for which the

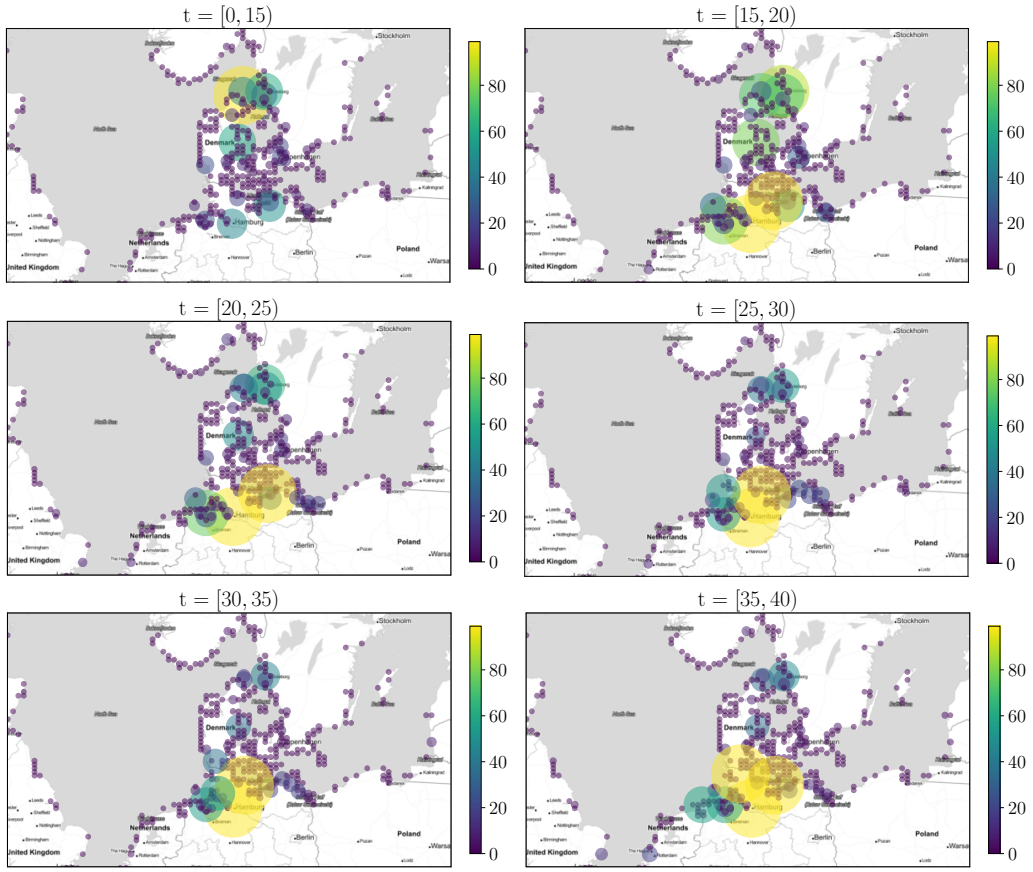


Figure 2.8.: Non-stationary Symmetry Structures: Changing Supply.

same observations are made are considered symmetrical at that point in time. However, when unrolling the model, i.e., moving forward in time, it is possible that some objects no longer *share* the same behaviour as they did before. Thus, the problem of identifying symmetries in a local state, i.e., in a local model, extends to identifying symmetries over time, i.e., identifying symmetrical behaviour. Finding symmetries in the context of DPRMs refers to finding symmetries in observations for sets of objects as realisations of a common randvar and this over time. As observations collected over time (i.e. realisations of random variables) result in time series, the problem extends to identify symmetries between time series associated with objects. Depending on the resolution and the length of the time series, the direct search for similarity is computationally intensive due to the high dimensionality of the time series itself, and additionally complex due to many objects represented in a lifted model. While the detection of exact symmetric behaviour, i.e., two or more objects having exactly the same behaviour, is not realistic, often approximately symmetric behaviour with a bounded error is desired, i.e., the behaviour of several objects is approximately the same over time.

Figure 2.8 gives an idea of how cargo supply in the Baltic Sea region develops over time t. Here, each point illustrates the normalised cargo supply amount in tons that was shipped to a specific point on the coast and from there is distributed further in the country. The size of each point, more precisely the diameter of each point, indicates the amount of cargo delivered. The figure indicates that at the beginning of the year (for $0 < t < 20$, with t referring to data of a calendar week) the supply, i.e., the need for resources, is higher in the northern regions, while for the rest of the year (for $20 < t < 40$) the supply slowly decreases there and increases in the southern regions. It is important to note that the supply for $20 < t < 40$ in the respective regions is more or less constant over a longer period of time before it changes. As represented by this example, a mechanism is needed to identify such points in time when symmetry structures change in order to take it into account when exploiting symmetries in lifted inference.

In conclusion, there is a single important idea that links all three motivational examples in this section: non-stationarity in data. All forms of non-stationarity, i.e., changes in domain sizes, probability distributions, or symmetry structures, affect the potential of applying DPRMs for real-world use. Throughout this work, we work on each of those aspects. Before going into details, we first give a formal introduction to DPRMs and then elaborate on each of the challenges motivated here.

3 Preliminaries

In this chapter, we present definitions for dynamic probabilistic relational models (DPRMs), and we review lifted variable elimination (LVE) together with lifted dynamic junction tree (LDJT) for fast bulk query answering. Furthermore, we elaborate on how LDJT is extended to an online algorithm by making small adjustments to the original version. In the remainder of this work, we rely on the online version of LDJT. In the introduction and motivation of this work, we have already discussed the concept of symmetry at one point in time and similar behaviour, i.e., temporal symmetries, over time. In this context, we, furthermore, establish a connection between DPRMs and time series analysis and present previous work in this field. To this end, we present ordinal pattern symbolisation to encode the behaviour of a time series.

3.1 Dynamic Probabilistic Relational Models (DPRMs)

In the following, we recapitulate DPRMs as introduced by Gehrke *et al.* (2018) in the context of the example from dry-bulk shipping according to Chapter 2. To summarise once again, we set up a DPRM for two sub-challenges in the shipping market, namely the interaction of (a) supply and demand and, accordingly, positioning of vessels, and (b) the effect of supply on freight rates as a driver for idle times, i.e., waiting time, due to congestion. We iteratively build a corresponding DPRM, thereby formalising the syntactic constructs of the DPRM. We conclude by providing a parameterized model G^{ex} as a running example for the remainder of this work. The following definitions and notations have their roots in the works by Poole (2003) and Taghipour *et al.* (2013a), but we provide the definitions given by Braun and Möller (2016) as well as Gehrke *et al.* (2018).

Example 3.1.1 (Sub-Challenges in Shipping)**.** *Freight rates, a fee per ton paid to transport cargo, are drivers for idle times in zones. These rates vary from zone to zone and play a crucial role in determining the operational planning of vessels. High freight rates in a particular zone may result in increased idle time, as operators seek to maximise profits by directing vessels to those zones. Even though freight rates are higher in certain zones, not every vessel operator will be able to fix business in those zones since zones might be over-crowded or have higher waiting times increasing costs for lay-time. The interaction between waiting times and freight rates in a zone is described with one*

randvar each. Freight rates are driven by the supply of commodities in zones represented by another randvar. The supply itself depends on the demand for commodities in other countries, resulting in global commodity streams, which are reflected by vessels transporting these commodities. Demand on the other hand varies in response to fluctuations in the global economy. In phases of expansion, i.e., during growth, demand for goods is generally higher than in phases of contraction, i.e., during a recession, which we encode by another randvar for the economic situation. Since idle conditions, freight rates and supply can be similar in multiple zones, we are able to develop a much smaller model by combining individual randvars for different zones. This is accomplished by parameterising randvars with one logvar to represent a set of zones.

A parameterised random variable (PRV) is defined as follows.

Definition 3.1.1 (PRV)**.** Let **R** be a set of randvar names (or randvars for short), **L** a set of logvar names (or logvars for short), and **D** a set of entities (universe). Each logvar L has a finite domain $\mathcal{D}(L) \subseteq \mathbf{D}$. The elements of $\mathcal{D}(L)$ are called objects or entities. The term $dm(l)$ refers to the domain $\mathcal{D}(L)$ of some object $l \in \mathcal{D}(L)$. The number of objects within a domain is denoted by the cardinality $|\mathcal{D}(L)|$. A *constraint* is a tuple $(\mathcal{X}, C_{\mathcal{X}})$ of a sequence of logvars $\mathcal{X} = (X_1, \ldots, X_n)$ and a set $C_{\mathcal{X}} \subseteq \times_{i=1}^{n} \mathcal{D}(X_i)$. A *PRV* $A(L_1, \ldots, L_n), n \geq 0$ is a construct of a randvar name $A \in \mathbf{R}$ combined with logvars $L_1, \ldots, L_n \in \mathbf{L}$. If $n = 0$, the PRV is parameterless and constitutes a propositional randvar. Then, the term $\mathcal{R}(P)$ denotes the (range) values of a PRV P. Further, the term $lv(Y)$ refers to the logvars and $rv(Y)$ to the randvars in some element Y, a PRV, a parfactor, or sets thereof. The term $gr(Y_{|C})$ denotes the set of instances of Y with all logvars in Y grounded w.r.t. constraint C. An instance is an instantiation (grounding) of Y, substituting the logvars in Y with a set of objects from the given constraint C.

The idea behind a PRV is to represent entities with similar behaviour through a single parameterized randvar, in order to obtain a sparser representation by replacing many individual randvars with a single parameterized randvar.

Example 3.1.2 (PRVs)**.** *To describe the interaction between idle times, freight rates and supply in zones across the globe, we use randvar names Idle, Supply and Rate parameterised with a logvar Z representing zones, building PRVs $Idle(Z)$, $Supply(Z)$ and $Rate(Z)$. To model the interaction between supply and demand according to vessel positioning, we additionally introduce a randvar name Demand parameterised with a logvar Z and a randvar name InArea parameterised with logvars V and Z, building PRVs $Demand(Z)$ and $InArea(V, Z)$. To encode the effect of the global economic situation on demand for goods, we add a randvar name Economy, which is not parameterised by a logvar, yielding an un-parameterised PRV Economy. The domain of Z is $\{z_0, z_1, \ldots, z_n\}$ with each z_i representing a different zone, and the domain of V is $\{v_0, v_1, \ldots, v_n\}$ each v_i representing a vessel on the market. Whilst the PRV $InArea(V, Z)$ has boolean range values and the PRV Economy has*

range values $\{Expansion, Contraction\}$*, all other PRVs have range values* $\{high, medium, low\}$[1]*. A constraint* $C = ((Z), \{(z_1), (z_2)\})$ *for* Z *allows to restrict* Z *to a subset of its domain in a specific PRV, for instance here to* z_1 *and* z_2*. Using this constraint, the expression* $gr(Idle(Z)_{|C})$ *evaluates to* $\{Idle(z_1), Idle(z_2)\}$*.*

To represent independent relations, PRVs are linked by a so-called parametric factor (parfactor) to compactly encode the full joint distribution of the DPRM.

Definition 3.1.2 (Parfactor, Parameterised Model)**.** Let Φ be a set of factor names, $\mathbf{X} \subseteq \mathbf{L}$ a set of logvars, $\mathcal{A} = (A^1, \ldots, A^n)$ a sequence of PRVs built from $\mathbf{R}$ and $\mathbf{X}$, and $(\mathcal{X}, C_{\mathcal{X}})$ a constraint on $\mathbf{X}$. A parfactor g is given by $\phi(\mathcal{A})_{|(\mathcal{X}, C_{\mathcal{X}})}$ with $\phi : \times_{i=1}^{n} \mathcal{R}(A^i) \mapsto \mathbb{R}^+$ a function with name $\phi \in \Phi$. The result of the function ϕ is called a parfactor table. A parameterised model *PRM* G is a set of parfactors $\{g^i\}_{i=1}^{n}$, representing the full joint distribution $P_G = \frac{1}{Z} \prod_{f \in gr(G)} f$, where Z is a normalising constant with $Z = \sum_{v \in \mathcal{R}(rv(gr(G)))} \prod_{\phi(\mathcal{A}) \in gr(G)} \phi(\pi_{\mathcal{A}}(v))$ and $\pi_{\mathcal{A}}(v)$ a projection of the current set of range values v onto $\mathcal{A}$.

Finally, the sub-challenges described in Example 3.1.1 can now be formalised as a PRM.

Example 3.1.3 (Parameterised Model)**.** *All dependent PRVs are combined by a parfactor, specifically*

$$
\begin{aligned}
g^1 &= \phi^1\big(Idle(Z), Rate(Z), Supply(Z)\big),\\
g^2 &= \phi^2\big(Supply(Z), InArea(V, Z)\big),\\
g^3 &= \phi^3\big(InArea(V, Z), Demand(Z)\big), \quad \textit{and}\\
g^4 &= \phi^4\big(Economy, Demand(Z)\big),
\end{aligned}
$$

each encoding their joint probability distribution. We omit the concrete mappings of range values to potentials for ϕ^1*,* ϕ^2*,* ϕ^3*, and* ϕ^4*.*

The parfactors as named above form a local but static model, i.e., temporal behaviour is not yet encoded within the model. To encode temporal behaviour, so called DPRMs follow the idea of having an initial model and a temporal copy pattern to describe model changes over time. DPRMs model a stationary process, i.e., changes from one time step to the next follow the same distribution.

Definition 3.1.3 (Dynamic Probabilistic Relational Model (DPRM))**.** A DPRM G is a pair of PRMs $(G_0, G_{\rightarrow})$ where G_0 is a PRM representing the first time step and $G_{\rightarrow}$ is a two-slice temporal parameterised model representing $\mathbf{A}_{t-1}$ and $\mathbf{A}_t$ where $\mathbf{A}_{\pi}$ is a set of PRVs from time slice π. An inter-slice parfactor $\phi(\mathcal{A})_{|C}$ has arguments $\mathcal{A}$ under constraint C that include PRVs from both $\mathbf{A}_{t-1}$ and $\mathbf{A}_t$, encoding transitioning from time step $t-1$ to t. A

[1] For sake of simplicity we only consider three range values here.

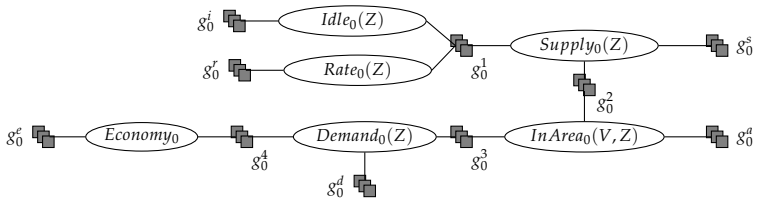


Figure 3.1.: First time step G_0^{ex} for model G^{ex}.

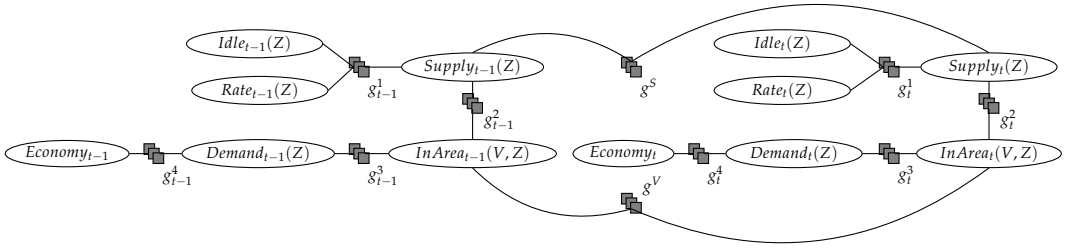


Figure 3.2.: Two-slice parameterised probabilistic model $G_{\rightarrow}^{ex}$ for model G^{ex}.

DPRM $(G_0, G_{\rightarrow})$ represents the full joint distribution $P_{(G_0,G_{\rightarrow}),T}$ by unrolling the DPRM for T time steps, forming a PRM as defined above.

Example 3.1.4 (DPRM)**.** *Both Fig. 3.1 and Fig. 3.2 depict the final example DPRM G^{ex}. Variable nodes (ellipses) correspond to PRVs, factor nodes (boxes) to parfactors. Edges between factor and variable nodes denote relations between PRVs, encoded in parfactors. Figure 3.1 illustrates G_0^{ex}, which represents the initial time step of our model G^{ex}, and features time-indexed parfactors g_0^1, g_0^2, g_0^3, and g_0^4 for time step $t = 0$. Additionally, each PRV in G_0^{ex} is assigned an additional parfactor as a prior. Figure 3.2 illustrates $G_{\rightarrow}^{ex}$, which depicts how G^{ex} evolves over time, i.e., how to transition from a time step $t-1$ to a time step t. $G_{\rightarrow}^{ex}$ consists of two equal submodels, one submodel for time step $t-1$ and one submodel for time step t. The two submodels $G_{\rightarrow}^{ex}$ are separated by inter-slice parfactors g^S and g^V that separate the past from the present. Parfactors in a slice reference time-indexed PRVs, namely $Idle_t(Z)$, $Rate_t(Z)$, $Supply_t(Z)$, $InArea_t(V,Z)$, $Demand_t(Z)$ and $Economy_t$.*

A DPRM as represented in Fig. 3.1 and Fig. 3.2 is assumed to be *fully lifted* in the initial model state. That is, all objects of zones and vessels are represented by a respective logvar Z or V. Over time, with making observations, i.e., with new knowledge for specific domain objects like specific vessels from the vessel domain $\mathcal{D}(V)$ or specific zones from the zone domain $\mathcal{D}(Z)$, the fully lifted model can ground, which we elaborate on as follows.

3.2 Query Answering and Evidence Handling

Given a DPRM, one can ask queries about (marginal) probability distributions or the probability of an event given evidence, i.e., evidence is encoded into the model as part of asking queries. A query is formally defined as.

Definition 3.2.1 (Queries)**.** Given a DPRM $(G_0, G_{\rightarrow})$, a ground PRV Q_t, and a set of events $\mathbf{E}_{0:t} = \{\{E_{s,i} = e_{s,i}\}_{i=1}^{n}\}_{s=0}^{t}$ for time steps 0 to t with $e_{s,i} \in \mathcal{R}(E_{s,i})$ fixed range values, the term $P(Q_\pi \mid \mathbf{E}_{0:t})$, $\pi \in \{0, \ldots, T\}$, $t \leq T$, denotes a query w.r.t. $P_{(G_0,G_{\rightarrow}),T}$, which is called prediction for $\pi > t$, filtering for $\pi = t$, and hindsight for $\pi < t$.

Example 3.2.1 (Queries)**.** *In the context of the shipping application, an example query for time step $t = 1$, such as $P(Supply_1(z_1) \mid Supply_0(z_1) = high)$, asks for the probability distribution of supply at time step $t = 1$ in a certain zone z_1, given that in the initial time step $t = 0$ the supply was high, denoted with an event $Supply_0(z_1) = high$.*

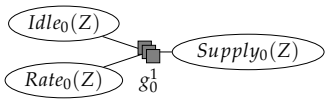
To *encode* the updated knowledge within the model, i.e., the events provided as part of a query, parfactor potentials are adjusted to allow for making more accurate predictions based on new information provided.

Definition 3.2.2 (Evidence)**.** A parfactor $g^e = \phi^e(E(\mathbf{X}))_{|C^e}$ specifies *evidence* for a set of *events* $\{E(\mathbf{x}^i) = o\}_{i=1}^{n}$ of a PRV $E(\mathbf{X})$. The function ϕ^e maps the value o to 1 and the remaining range values of $E(\mathbf{X})$ to 0. Constraint C^e encodes the observed groundings $\mathbf{x}^i$ of $E(\mathbf{X})$, i.e., $C^e = (\mathbf{X}, \{\mathbf{x}^i\}_{i=1}^{n})$.

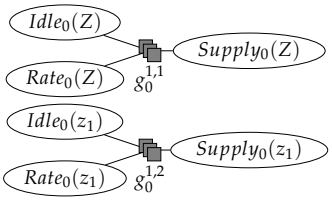
Queries can include a series of events, which in turn can also include symmetries. Symmetries in this case refer to the same event for multiple groundings, i.e., whenever events occur w.r.t. instances of the same PRV. The concept is to split the parfactors into two groups, one for instances with evidence and the other for instances without evidence. This way, the parfactor that deals with the instances with evidence can handle the given evidence while the other parfactor remains unchanged after updating its constraint. This method of separating instances is referred to as *shattering*.

Definition 3.2.3 (Shattered Model)**.** Under evidence, a model $G_t = \{g_t^i\}_{i=1}^{n}$ at time step t, is split w.r.t. its parfactors such that its structure remains $G_t = \{g_t^{i,1}, \ldots, g_t^{i,k_i}\}_{i=1}^{n}$ with $k_i \in \mathbb{N}^+$. Every parfactor g_t^i can have up to $k_i \in \mathbb{N}^+$ splits $g_t^{i,j} = \phi_t^{i,j}(\mathcal{A}^i)_{|C^{i,j}}$, where $1 \leq j \leq k_i$ and $\mathcal{A}^i$ is a sequence of the same PRVs but with different constraint $C^{i,j}$ and varying functions $\phi_t^{i,j}$ due to evidence. All k_i splits of the parfactor g_t^i are called a parfactor group.

Example 3.2.2 (Evidence)**.** *Assume we observe one event $Supply_0(z_1) = high$, i.e., high supply in a zone z_1 in the initial time step $t = 0$. For a better illustration, see also Fig. 3.3a and Fig. 3.3b, on*



(a) Parfactor graph of G^{ex}, here of G_0^{ex}, for factor and variables of sub-challenge two before the encoding of any events.



(b) Parfactor graph of G^{ex}, here of G_0^{ex}, for factor and variables of sub-challenge two shattered by the event $Supply_0(z_i) = high$.

Figure 3.3.: Shattering of the model due to evidence. Note that we omit parfactors for priors in G_0^{ex} for clarity.

which we elaborate as follows. Furthermore, note that the given parfactor potentials in this example are randomly chosen and have no further meaning. The event is encoded in a model afflicted without evidence, as represented in Fig. 3.3a, by an evidence parfactor that results by splitting g_0^1 in G_0^{ex} into two parts, namely $g_0^{1,1}$ and $g_0^{1,2}$ replacing the original parfactor g_0^1 as represented in Fig. 3.3b accordingly: The parfactor $g_0^{1,1}$ encodes domain objects without evidence, and the parfactor $g_0^{1,2}$ encodes domain objects w.r.t. to the provided event:

- *The evidence parfactor $g_0^{1,2} = \phi_0^{1,2}(Idle_0(Z), Rate_0(Z), Supply_0(Z))_{|C^{1,2}}$ encodes the event $Supply_0(z_1) = high$ as follows:*
  - *The factor $\phi_0^{1,2}$ has mappings $\phi_0^{1,2}(Idle_0(Z), Rate_0(Z), Supply_0(Z)) = 1$ where the range value $Supply_0(Z) = high$, and $\phi_0^{1,2}(Idle_0(Z), Rate_0(Z), Supply_0(Z)) = 0$ for all other range values, i.e., where $Supply_0(Z) \neq high$.*
  - *The constraint $C^{1,2} = ((Z), \{(z_1)\})$ restricts the domain of Z to z_1 only.*
- *The original parfactor $g_0^{1,1} = \phi_0^{1,1}(Idle_0(Z), Rate_0(Z), Supply_0(Z))_{|C^{1,1}}$ encodes all remaining domain objects as follows:*
  - *The factor $\phi_0^{1,1}$ remains unchanged and keeps original mappings.*
  - *The constraint $C^{1,1} = ((Z), \{(z_2), \ldots, (z_n)\})$ restricts the domain of Z to all remaining domain objects in $\mathcal{D}(Z) \backslash \{z_1\}$.*

An example of the non-normalised parfactor potential function of $g_0^{1,1}$ and $g_0^{1,2}$ can look as follows. The adjusted potentials are highlighted below in bold.

$Idle_0(Z)$	$Rate_0(Z)$	$Supply_0(Z)$	$g_0^{1,1}$
high	*high*	*high*	5
low	*low*	*high*	3
high	*medium*	*high*	7
...	...	...	...
high	*high*	*mid*	2
high	*high*	*low*	5

$Idle_0(Z)$	$Rate_0(Z)$	$Supply_0(Z)$	$g_0^{1,2}$
high	*high*	*high*	5
low	*low*	*high*	3
high	*medium*	*high*	7
...	...	...	...
high	*high*	*mid*	**0**
high	*high*	*low*	**0**

Assume we observe high supply for more than one zone, e.g., for 50 zones, i.e., we are provided with

events $\{Supply_0(z_i) = high\}_{i=1}^{50}$. *Then, the constraint* $C^{1,2} = ((Z), \{(z_1), \ldots, (z_{50})\})$ *restricts the domain of* Z *to* $z_1, \ldots, z_{50}$ *in* $\phi_0^{1,2}(Idle_0(Z), Rate_0(Z), Supply_0(Z))_{|C^{1,2}}$.

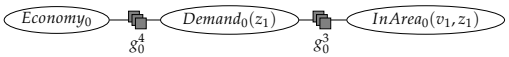
As demonstrated in Example 3.2.2, evidence leads to groundings in the lifted model, i.e., having multiple copies of an original parfactor with each copy being limited by the use of a constraint to a different group of entities. At this point, we'll introduce the concept of a *template model*, an auxiliary construct we'll use throughout this work. A template model is a copy of the freshly instantiated DPRM, but with empty constraints on parfactors, such the model can no longer be shattered but still encodes relations and potential functions. The template model will become useful in the course of the work for comparisons between a shattered model and the corresponding model that describes the initial state of the model.

Definition 3.2.4 (Template Model)**.** A *template model* $\mathcal{G}$ is a replica of a freshly instantiated DPRM G where every parfactor $\tilde{g} = \tilde{\phi}(\mathcal{A})_{|C}$ in $\mathcal{G}$ has an empty constraint $C = (\mathcal{X}, C_{\mathcal{X}})$ with $C_{\mathcal{X}} = \bot$. As G is shattered w.r.t. its parfactor over time, its structure at time step t remains $G_t = \{g_t^{i,1}, \ldots, g_t^{i,k_i}\}_{i=1}^{n}$ with $k_i \in \mathbb{N}^+$. Each parfactor g_t^i can have up to $k \in \mathbb{N}^+$ splits $g_t^{i,j} = \phi_t^{i,j}(\mathcal{A}^i)_{|C^{i,j}}$, where $1 \leq j \leq k_i$ and $\mathcal{A}^i$ is a sequence of the same PRVs but with different constraint $C^{i,j}$ and varying functions $\phi_t^{i,j}$ due to evidence. For every evidence parfactor g_t^{i,k_i} in G_t, exists one corresponding single *template parfactor* $\tilde{g} \in \mathcal{G}$ with the same sequence of PRVs $\mathcal{A}^i$.

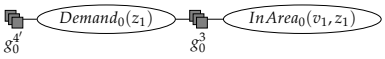
A model G_t that is shattered at a time step t will also cause groundings in the model G_{t+1} when the model is rolled out for another time step. This is as groundings are carried over to all subsequent time steps through message passing when performing query answering, i.e., in inference. As follows, we give a high-level overview of lifted query answering in DPRMs and elaborate in more detail about the transfer of groundings to subsequent time steps and the challenges involved.

3.3 Lifted Variable Elimination and the Lifted Dynamic Junction Tree Algorithm

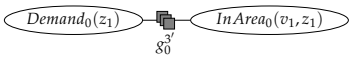
For DPRMs, LDJT (Gehrke *et al.*, 2018) provides an efficient way to answer a set of queries as defined in Definition 3.2.1. We refrain from repeating all details about LDJT and leave out formal definitions here as they are not relevant for this work and focus on giving an intuition how LDJT as an query answering algorithm works. For further details see the original work by Braun and Möller (2016) for static models and for temporal models by Gehrke *et al.* (2018). The important part for this work is that LDJT uses LVE (Poole, 2003; Taghipour *et al.*, 2013b) as a subroutine during its calculations. LVE exploits symmetries that lead to duplicate calculations, while LDJT utilises previously computed intermediate inference results to efficiently answer multiple queries, making LDJT an efficient query answering algorithm.



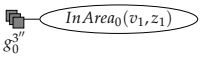
(a) Parfactor graph for a submodel of G^{ex}, here of G_0^{ex}, with just one domain object, here v_1 and z_1.



(b) Elimination of $Economy_0$ by summing out range values of $Economy_0$ from g_0^4 yielding $g_0^{4'}$.



(c) Joining parfactors $g_0^{4'}$ and g_0^3 yielding $g_0^{3'}$.



(d) Elimination of $Demand(z_1)_0$ by summing out range values of $Demand(z_1)_0$ from $g_0^{3'}$ yielding the parfactor $g_0^{3''}$.

Figure 3.5.: Example for query answering using VE on a ground model with only one domain object each, i.e., z_1 for zones $\mathcal{D}(Z)$ and v_1 for vessels $\mathcal{D}(V)$ only.

We first give an overview of LDJT on DPRMs followed by extending LDJT to an online algorithm, which we require as a basis for contributions made in this work.

Lifted Dynamic Junction Tree Algorithm

LDJT efficiently answers queries given a DPRM and sets of events as evidence. For efficient query answering, LDJT constructs a first-order junction tree (FO jtree) as a helper structure before answering queries. We will not go into further details about how a FO jtree based on a DPRM is constructed. The important part of converting a DPRM into an FO jtree is to allow multiple queries to be answered efficiently, on which we elaborate as follows.

For query answering, LDJT uses LVE as a subroutine, eliminating PRVs from the model that are not part of a query itself. A PRV is eliminated by summing out range values from the table of a parfactor given by the potential function and all possible combinations of its PRV range values as input pairs, followed by joining the resulting *condensed* table with neighbouring parfactor tables. In the following, we first illustrate how eliminating variables works using an example of a grounded model with only one domain object. Then, the complexity of the example is extended by adding more domain objects to show how LVE exploits symmetries and keeps complexity low during inference.

Example 3.3.1 (Variable Elimination)**.** *For simplicity, we illustrate the elimination process for a static model, here for time step $t = 0$ with ground PRVs $Economy_0$, $Demand_0(z_1)$, $InArea_0(v_1, z_1)$ and parfactors g_0^4, and g_0^3 as depicted in Fig. 3.5a, i.e., a submodel of the full model depicted in Fig. 3.1. We ask the query $P(InArea_0(v_1, z_1))$ to determine the marginal distribution of whether vessel v_1 is in zone z_1 or not. Query answering on the ground model works as follows:*

1. *All non-query variables are eliminated. First, the PRV $Economy_0$ is eliminated from the table of the parfactor g_0^4, which is given by the potential function and all possible combinations of*

its connected PRV range values, i.e., here $\mathcal{R}(Economy_0)$ *and* $\mathcal{R}(Demand_0(z_1))$*, as input pairs as represented in Fig. 3.6a on the left. To eliminate* $Economy_0$ *from the parfactor* g_0^4*, all potentials where* $Demand_0(z_1) = high$*,* $Demand_0(z_1) = medium$ *and* $Demand_0(z_1) = low$ *are summed up yielding a new parfactor* $g_0^{4'}$ *as represented in Fig. 3.6a on the right.*

2. *Subsequently, the resulting parfactor* $g_0^{4'}$ *is joined with parfactor* g_0^3 *yielding a parfactor* $g_0^{3'}$ *as depicted in Fig. 3.6b. Joining works by multiplying potentials of matching range values in both parfactor tables.*

3. *As a final step, the last non-query PRV* $Demand_0(z_1)$ *is summed out from the parfactor* $g_0^{3'}$ *(in the same manner as illustrated in Fig. 3.6a), yielding a parfactor* $g_0^{3''}$ *that only includes potentials for PRV* $InArea_0(v_1, z_1)$*. The parfactor potentials in* $g_0^{3''}$ *are count-normalised with potentials as counts to obtain the marginal distribution.*

The factorised representation of the joint distribution in graphical models allows efficient computation and inference. We can perform computations on each factor independently and then combine them to obtain the full joint distribution (after count normalisation) as shown. The full joint distribution is obtained by multiplying the factors with overlapping PRVs. For the factorised model depicted in Fig. 3.5a, the full joint distribution is formally defined as

$$P(Economy_0, Demand_0(z_1), InArea_0(v_1, z_1)) = \frac{1}{Z}\, \phi_0^3(Demand_0(z_1), InArea_0(v_1, z_1)) \cdot \phi_0^4(Economy_0, Demand_0(z_1)) \tag{3.3.1}$$

$Economy_0$	$Demand_0(z_1)$	g_0^4
Expansion	high	5
Contraction	high	3
Expansion	medium	4
Contraction	medium	3
Expansion	low	1
Contraction	low	6

$Demand_0(z_1)$	$g_0^{4'}$
high	5+3 = 8
medium	4+3 = 7
low	1+6 = 7

(a) Summing out $Economy_0$ in g_0^4 yielding a *smaller* parfactor $g_0^{4'}$.

$Demand_0(z_1)$	$g_0^{4'}$
high	8
medium	7
low	7

·

$Demand_0(z_1)$	$InArea_0(v_1, z_1)$	g_0^3
high	true	3
high	false	4
medium	true	4
medium	false	6
low	true	5
low	false	8

=

$Demand_0(z_1)$	$InArea_0(v_1, z_1)$	$g_0^{3'}$
high	true	$8 \cdot 3 = 24$
high	false	$8 \cdot 4 = 32$
medium	true	$7 \cdot 4 = 28$
medium	false	$7 \cdot 6 = 42$
low	true	$7 \cdot 5 = 35$
low	false	$7 \cdot 8 = 56$

(b) Multiplication/Joining of $g_0^{4'}$ and g_0^3 yielding parfactor $g_0^{3'}$.

Figure 3.6.: Illustration of operations *summing out* and *joining* in lifted variable elimination. Note that the potentials in this example are randomly chosen.

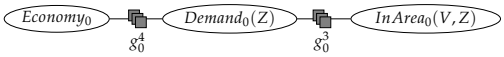
where Z is the normalisation count with

$$Z = \sum_{a \in \mathcal{R}(InArea_0(v_1,z_1))} \sum_{d \in \mathcal{R}(Demand_0(z_1))} \phi_0^3(Demand_0(z_1) = d, InArea_0(v_1,z_1) = a) \sum_{e \in \mathcal{R}(Economy_0)} \phi_0^4(Economy_0 = e, Demand_0(z_1) = d) \tag{3.3.2}$$

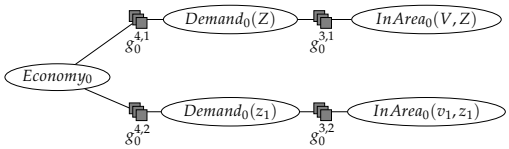
and $\cdot$ as a join over shared randvars (here $Demand_0(z_1)$). The sum $\sum$ is over the range values of a randvar. Note that the computation for Z is not necessary, but a simple so-called α normalisation of the resulting parfactor table, here the one of $g_0^{3''}$, is sufficient.

If the domain of the submodel is extended, i.e., $|\mathcal{D}(Z)| > 1$ and $|\mathcal{D}(V)| > 1$, the complexity in VE increases as more grounded PRVs have to be eliminated. Let's look at the same example query again, but extend the domain of the model to illustrate how LVE works.

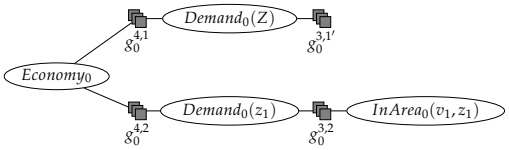
Example 3.3.2 (Lifted Variable Elimination)**.** *We illustrate lifted variable elimination steps for answering the marginal query $P(InArea_0(v_1,z_1))$ as in the preceding example, but the underlying model features more than just one domain object, i.e., $|\mathcal{D}(V)| > 1$ and $|\mathcal{D}(Z)| > 1$ with $M =$*



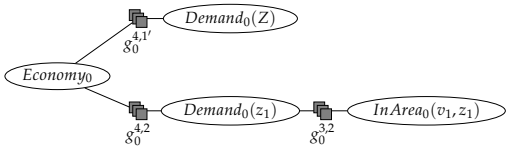
(a) Parfactor graph for a submodel of G^{ex}, here of G_0^{ex}, with multiple domain objects, here $\mathcal{D}(V) > 1$ and $\mathcal{D}(Z) > 1$.



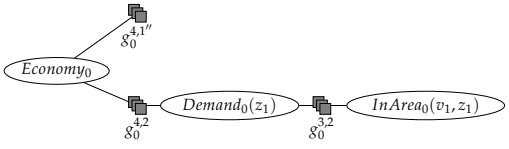
(b) Shattering G^{ex} by domain objects v_1 and z_1.



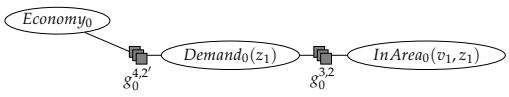
(c) Eliminate $InArea_0(V,Z)$ by summing out its range values from $g_0^{3,1}$ yielding $g_0^{3,1'}$.



(d) Joining parfactors $g_0^{3,1'}$ and $g_0^{4,1}$ yielding $g_0^{4,1'}$.



(e) Eliminate $Demand_0(Z)$ by summing out its range values from $g_0^{4,1'}$ yielding $g_0^{4,1''}$.



(f) Joining parfactors $g_0^{4,1''}$ and $g_0^{4,2}$ yielding $g_0^{4,2'}$. All further steps are similar to elimination as depicted in Fig. 3.5.

Figure 3.7.: Example for query answering using LVE on a model with multiple domain objects, i.e., $\mathcal{D}(Z) > 1$ for zones and $\mathcal{D}(V) > 1$ for vessels.

$|\mathcal{D}(V)|$ *and* $N = |\mathcal{D}(Z)|$*. Then, the full joint distribution is defined as follows:*

$$\begin{aligned} &P(Economy_0, Demand_0(z_1), \ldots, Demand_0(z_N), InArea_0(v_1, z_1), \ldots, InArea_0(v_M, z_N)) \\ &\quad = \frac{1}{Z} \prod_{j=1}^{M} \prod_{i=1}^{N} \phi_0^3(Demand_0(z_i), InArea_0(v_j, z_i)) \prod_{i=1}^{N} \phi_0^4(Economy_0, Demand_0(z_i)) \end{aligned} \tag{3.3.3}$$

with Z as

$$\begin{aligned} Z = \prod_{j=1}^{M} \prod_{i=1}^{N} & \sum_{d_i \in \mathcal{R}(Demand_0(z_i))} \sum_{a_{ji} \in \mathcal{R}(InArea_0(v_1, z_1))} \phi_0^3(Demand_0(z_i) = d_i, InArea_0(v_j, z_i) = a_{ji}) \\ & \prod_{i=1}^{N} \sum_{e \in \mathcal{R}(Economy_0)} \sum_{d_i \in \mathcal{R}(Demand_0(z_i))} \phi_0^4(Economy_0 = e, Demand_0(z_i) = d_i) \end{aligned} \tag{3.3.4}$$

Compared to Eq. (3.3.1)*, the full joint distribution is extended by factors for the additional domain objects (here the product* $\prod$ *over vessels* v_j *and zones* z_i*). Therefore, the complexity in VE increases as more grounded PRVs have to be eliminated. To answer the marginal query* $P(InArea_0(v_1, z_1))$*, VE has to sum out each grounded PRV not occurring in the query:*

$$\begin{aligned} & P(InArea_0(v_1, z_1)) \\ = \frac{1}{Z} & \sum_{d_2 \in \mathcal{R}(Demand_0(z_2))} \sum_{a_{22} \in \mathcal{R}(InArea_0(v_2, z_2))} \phi_0^3(Demand_0(z_2) = d_2, InArea_0(v_2, z_2) = a_{22}) \\ & \qquad \cdot \ldots \cdot \\ & \sum_{d_N \in \mathcal{R}(Demand_0(z_N))} \sum_{a_{MN} \in \mathcal{R}(InArea_0(v_M, z_N))} \phi_0^3(Demand_0(z_N) = d_N, InArea_0(v_M, z_N) = a_{MN}) \\ & \cdot \sum_{e \in \mathcal{R}(Economy_0)} \sum_{d_2 \in \mathcal{R}(Demand_0(z_2))} \phi_0^4(Economy_0 = e, Demand_0(z_2) = d_2) \\ & \qquad \cdot \ldots \cdot \\ & \sum_{e \in \mathcal{R}(Economy_0)} \sum_{d_N \in \mathcal{R}(Demand_0(z_N))} \phi_0^4(Economy_0 = e, Demand_0(z_N) = d_N) \end{aligned} \tag{3.3.5}$$

VE sums out the grounded PRVs that are not $InArea_0(v_1, z_1)$*, followed by multiplying the resulting factors and normalising the result. In Equation* (3.3.5) *all sums* $\sum_{e \in \mathcal{R}(Economy_0)} \sum_{d_i \in \mathcal{R}(Demand_0(z_i))}$ *and* $\sum_{d_i \in \mathcal{R}(Demand_0(z_i))} \sum_{a_{ji} \in \mathcal{R}(InArea_0(v_j, z_i))}$ *are equal since* ϕ_0^3 *and* ϕ_0^4 *are identical without evidence encoded within the model. This is where LVE, i.e. lifting, helps reducing complexity in query answering. Instead of computing the sum over and over again, LVE computes the sum only once for one representative of the model and exponentiates the result to the number of sums to reduce the number*

of overall operations. Therefore, the query above can be simplified as follows

$$
\begin{aligned}
&P(InArea_0(v_1, z_1)) \\
&= \frac{1}{Z}\Big(\sum_{e \in \mathcal{R}(Economy_0)} \sum_{d \in \mathcal{R}(Demand_0(z_2))} \phi_0^4(Economy_0 = e, Demand_0(Z) = d)\Big)^{N-1} \\
&\Big(\sum_{d \in \mathcal{R}(Demand_0(z_2))} \sum_{a \in \mathcal{R}(InArea_0(v_2, z_2))} \phi_0^3(InArea_0(V, Z) = a, Demand_0(Z) = d)\Big)^{(M-1)(N-1)}
\end{aligned}
\tag{3.3.6}
$$

Figure 3.7 depicts the process of LVE. First, the lifted model as depicted in Fig. 3.7a is shattered w.r.t. to the domain objects z_1 and v_1 as part of the query $P(InArea_0(v_1, z_1))$. Shattering on domain objects leads to ground PRVs involving z_1 and v_1. This step is necessary to be able to eliminate all other non-query variables. Shattering splits the original parfactor g_0^3 creating two copies $g_0^{3,1}$ and $g_0^{3,2}$ and parfactor g_0^4. The new parfactors replace the original parfactor accordingly as depicted in Fig. 3.7b. Further, each parfactors constraint is altered, i.e., the parfactors constraints for $g_0^{4,2}$ and $g_0^{3,2}$ limit its parfactor to z_1 and v_1 accordingly, whilst the parfactors constraints for $g_0^{4,1}$ and $g_0^{3,1}$ limit the parfactors to all other domain objects except z_1 and v_1.

After shattering, the model is prepared for LVE which works similarly as illustrated in Example 3.3.1. The difference is with the elimination of PRVs $InArea_0(V, Z)$, $Demand_0(Z)$ and its parfactors $g_0^{4,1}$ and $g_0^{3,1}$. As both $InArea_0(V, Z)$ and $Demand_0(Z)$ are representatives for the same randvar and all ϕs of $g_0^{4,1}$ and $g_0^{3,1}$ are identical, LVE computes the sum once and exponentiates the result to the number of sums as represented in Eq. (3.3.6)*. Once $g_0^{4,1}$ and $g_0^{3,1}$ and its connected PRVs are eliminated, only the ground model for domain objects z_1 and v_1 remains. All further steps are the same as in the previous example.*

Since the number of operations, i.e., sums and joins, are reduced depending on the degree of symmetries in the model, lifting speeds up inference. We have shown that LVE is able to efficiently answer marginal queries. However, LVE has one big disadvantage. LVE is able to answer one query, but once asking another query LVE has to redo all calculations again. LVE computes many intermediate results, e.g., distributions after summing out PRVs, as a side-product before being able to return the answer of a query. Those intermediate results can be saved for queries which need to be answered at a later stage.

The LDJT algorithm does that by performing two runs of LVE to initialise an FO jtree as data structure to hold those precomputed results to be then able to very efficiently answer a set of queries, i.e., LDJT enables for repeated query answering. An FO jtree is an acyclic graph, whose nodes are clusters of PRVs called parclusters. The parclusters are constructed by clustering PRVs of the DPRM into cliques and then creating a tree connecting the cliques yielding the FO jtree. Therefore, parclusters in the FO jtree form a submodel whose PRVs are conditionally independent of the PRVs in the remaining model given some PRVs that act as so-called *separators* between the submodels of the full model. Neighbouring parclusters in

the FO jtree share the same set of PRVs as separators. This has the advantage that answering a query for a PRV being part of the submodel simplifies, as all remaining PRVs except the factors of the separator PRVs are not needed to compute the answer to the query. Note that without the separators given, the influence of the remaining PRVs in the model is missing. Therefore, to answer a query for a PRV Q for a submodel G_i of G, one can first query for each separator PRVs of $G \setminus G_i$, and G_i and the answer to separator PRVs are sufficient to answer the query Q. When given different queries, the answers to the separator PRVs can be reused to enable for efficient repeated inference as initially motivated. In an FO jtree, the result of a query for a separator is called a message. LVE is used to eliminate all PRVs but the separator from the parclusters receiving messages from other neighbours, a scheme that is called message passing. For temporal models such as the DPRM, the interslice PRVs, which separate the past from the present and the present from the future, can be used in a FO jtree by constructing time-independent and time-dependent parclusters.

The parcluster holding the interface PRVs and its parfactors is called an *interface cluster*. The interface cluster allows for separating the past from the present and the present from the future by calculating a so-called α_t message that collects all information up to now (t) and passing that information on to the next time step ($t+1$). So-called α_t messages, which are passed from one time step to the next, encode the parfactors evolving over time, i.e., denote the full joint distribution of the current time step. Thus, the forward message provides the information necessary to answer queries of the next time step. LDJT performs the following steps:

(i) Construct two FO jtrees J_0 and J_t with in- and out-clusters from $(G_0, G_{\rightarrow})$.

(ii) For time step $t = 0$, use J_0, enter E_0, pass messages, answer each query term $Q_\pi \in Q_0$, and preserve the state at time 0 in message α_0.

(iii) For $t \geq 0$, instantiate J_t for the time step t, add α_{t-1} to the in-cluster, enter E_t in J_t, pass messages, answer each query term $Q_\pi \in Q_t$, and preserve the state in message α_t.

Whenever time moves on, i.e., t increases, the current FO jtree (J_t) is replaced by a new FO jtree for $t+1$. The FO jtree is freshly instantiated (J_0 if $t = 0$, $J_{\rightarrow}$ otherwise) and receives the state in message α_t, which has been added to the interface cluster and now holds all information about the past t time steps. Here, in principle, instantiating a fresh FO jtree J_t implies setting up a new FO jtree from the corresponding model structure G_t for the current time step t without any evidence entered. Given a current FO jtree J_t, LDJT distributes messages in J_t and then answers the queries that apply to this time step. Algorithm 1 represents an outline of LDJT. LDJT works based on a set of queries $\mathbf{Q}_{0:T}$ queries and evidence $\mathbf{E}_{0:T}$ to be provided in advance, i.e., before the execution of LDJT. In the following, we extend LDJT to online query answering for $T \rightarrow \infty$, which we rely on in due course of this work.

Algorithm 1 Lifted dynamic junction tree algorithm

Input: DPRM $(G_0, G_{\rightarrow})$, Evidence $\mathbf{E}_{0:T}$, Queries $\mathbf{Q}_{0:T}$
Construct FO jtrees $J_0, J_{\rightarrow}$
for $t = 0, 1, ..., T$ **do**
    Instantiate FO jtree J_t from $J_0, J_{\rightarrow}$
    Add α_{t-1} message and $E_t \in \mathbf{E}_{0...t...T}$ in J_t
    Pass messages in J_t
    Answer queries $Q_t \in \mathbf{Q}_{0...t...T}$ in J_t
    Calculate α_t message

An Online Version of the Lifted Dynamic Junction Tree Algorithm

LDJT can easily be extended for online query answering, i.e., can work based on evidence and queries that are provided at runtime. In Algorithm 1, evidence $\mathbf{E}_{0:T}$ and queries $\mathbf{Q}_{0:T}$ are provided in advance, i.e., LDJT expects to get evidence and queries before its execution. For online query answering LDJT gets two streams as inputs. One stream $\mathcal{E}$ for evidence that is formally defined as:

Definition 3.3.1 (Evidence Stream)**.** Given PRVs $X_{t,1}, X_{t,2}, \ldots, X_{t,n}$ of a slice G_t derived from a DPRM $(G_0, G_{\rightarrow})$ with range values $\mathcal{R}(X_{t,1}), \mathcal{R}(X_{t,2}), \ldots, \mathcal{R}(X_{t,n})$, and $\mathcal{X}_t = \{x_{t,1}, x_{t,2}, \ldots, x_{t,n}\}$ a set of all possible assignments of range values to PRVs denoting all possible probability states, i.e., $\forall r_{t,i} \in \mathcal{R}(X_{t,i}) : X_{t,i} = r_{t,i}$ with each $x_{t,i} \in \mathcal{X}_t$ is one specific assignment $X_{t,i} = r_{t,i}$. Then, an event stream $\mathcal{E}$ is defined as a function $\mathcal{E} : \mathbf{T} \rightarrow \mathcal{X}$, that maps each time step $t \in \mathbf{T}$ with $\mathbf{T} = \{0, \ldots, T\}$ to a set of assignments $x_i \in \mathcal{X}$. The stream of events $\mathcal{E}$ can then be represented as the collection of all events across all time steps, i.e., $\mathbf{E}_{0:T} = \{\{E_{s,i} = e_{s,i}\}_{i=1}^{n}\}_{s=0}^{T}$ where T is the final time step of the stream.

and one stream $\mathcal{Q}$ for queries that is formally defined as:

Definition 3.3.2 (Query Stream)**.** Given a DPRM $(G_0, G_{\rightarrow})$ and PRVs $\mathbf{X} = X_1, X_2, \ldots, X_n$ of a slice G_t from a DPRM $(G_0, G_{\rightarrow})$, let $P(Q_t)$ denote a query w.r.t. $P_{(G_0, G_{\rightarrow}),t}$ at time step $t \in \{0, \ldots, T\}$. Then, a query stream $\mathcal{Q}$ is defined as a function $\mathcal{Q} : \mathbf{T} \rightarrow \mathcal{P}$, that maps each time step $t \in \mathbf{T}$ with $\mathbf{T} = \{0, \ldots, T\}$ to a set of probability queries $Q_t(X_i) \in \mathcal{P}$ over PRVs $X_i \in \mathbf{X}$ at time step t. The stream of events $\mathcal{Q}$ can then be represented as the collection of all queries across all time steps, i.e., $\mathbf{Q}_{0:T} = \{\{Q_s(X_i)\}_{i=1}^{n}\}_{s=0}^{T}$ where T is the final time step of the stream.

Algorithm 2 outlines an online version of LDJT with minior changes compared to the original version in Algorithm 1. Instead of providing sets of evidence and query terms in advance, evidence and queries are provided on the fly during the execution of LDJT. We use the online version of LDJT in this work as a query answering algorithm. By converting LDJT

Algorithm 2 Online lifted dynamic junction tree algorithm

Input: DPRM $(G_0, G_{\to})$, Evidence stream $\mathcal{E}$, Query stream $\mathcal{Q}$
Construct FO jtrees $J_0, J_{\to}$
for $t = 0, 1, ...$ **do**
  Instantiate FO jtree J_t from $J_0, J_{\to}$
  Add α_{t-1} message
  Add evidence E_t from evidence stream $\mathcal{E}$
  Pass messages in J_t
  Answer queries Q_t from query stream $\mathcal{Q}$ on J_t
  Calculate α_t message

to an online algorithm, overhead of the whole algorithm can be reduced. This is in particular the creation of the FO jtrees J_0 and $J_{\to}$. However, switching to an online algorithm also leads to challenges, particularly those manifested by non-stationarity.

In the course of the work, we analyse the *behaviour* of domain objects in the context of other domain objects using approaches from the field of time series analysis. To this end, we will first combine DPRM and time series analysis, and present some preliminary work in this area.

3.4 From DPRMs to Time Series Analysis

In a DPRM, (real-valued) random variables observed over time are considered as a time series describing the behaviour of a specific domain object. While DPRMs describe the overall behaviour of a system at a macro level, time series analysis provides methods for analysing the micro-level behaviour of individual components, such as domain objects, within the DPRM. Combining the two levels allows us to gain a more comprehensive understanding of how complex systems can behave.

Example 3.4.1 (Micro- and Macro-Level Analysis)**.** *The PRV $InArea_t(V, Z)$, which is part of our example DPRM G^{ex}, encodes the position of vessels $v_i \in V$ in zones $z_j \in Z$ on the globe (macro level). With observations for $InArea_t(V, Z)$ a time series results (micro level). One example time series for this specific PRV with length $t = 3$ is*

$$\big(InArea_1(v_1, z_1) = true, InArea_2(v_1, z_2) = true, InArea_3(v_1, z_1) = true\big),$$

i.e., the position of the vessel v_1 for each time step t. The time series results from observations for v_1 being at $t = 1$ in zone z_1, at $t = 2$ in zone z_2 and at $t = 3$ back in zone z_1. The time series describes the specific behaviour for one domain object v_1 that is part of a more complex system described by the DPRM. Considering further observations for other domain objects and other randvars of the DPRM,

the general system behaviour, in particular also mutual dependencies in it, is described.

Next, we more formally define the connection between DPRMs and time series. We first define a stochastic process, followed by formally defining notation for a univariate and multivariate time series.

Definition 3.4.1 (Stochastic Process)**.** Let Ω be a set describing all possible states of a dynamical system, also called state space. Events are taken from a σ-algebra $\mathcal{A}$ on Ω. Then $(\Omega, \mathcal{A})$ is a measurable space. A sequence of random variables, all defined on the same probability space $(\Omega, \mathcal{A}, \mu)$ with μ as the probability measure, is called a stochastic process. For real-valued random variables, a stochastic process is a function

$$X : \Omega \times \mathbb{N} \to \mathbb{R}, \tag{3.4.1}$$

where $X(\omega, t) := X_t(w)$ with $\omega \in \Omega$ depending on both, randomness (or coincidence) and time t.

Over time, the individual variables $X_t(\omega)$ of this stochastic process are observed, so-called realisations. The sequence of realisations is called time series.

Definition 3.4.2 (Time Series)**.** Given a stochastic process $X : \Omega \times \mathbb{N} \to \mathbb{R}$ and fixing of some $\omega \in \Omega$, a time series is given by

$$(X_1(\omega), X_2(\omega), X_3(\omega), \dots) = (x_t)_{t \in \mathbb{N}}.$$

In the case, $x_t \in \mathbb{R}$, the time series is called univariate, while in the case $x_t \in \mathbb{R}^m$ it is called multivariate. For stochastic processes, we use the capitalisation $(X(t))_{t \in \mathbb{N}}$, while for observations, i.e., paths or time series, we use the small notation $(x(t))_{t \in \mathbb{N}}$.

In summary, evidence in a DPRM encoding stochastic processes $(X(t))_{t \in \mathbb{N}}$ forms a canonical time series $(x_t)_{t \in \mathbb{N}}$. In contrast to canonical time series, which embed original data in a high-dimensional space (here time t) while preserving its dynamics and structure, ordinate time series focuses only on the order of values over time, taking into account time lags or delays. In this work, we rely on ordinal pattern symbolisation to make accruing time series data for domain objects comparable. Next, we introduce ordinal pattern symbolisation for univariate and multivariate time series data.

3.5 Multivariate Ordinal Pattern (MOP) Symbolisation

To encode the behaviour of a time series, we use ordinal pattern symbolisation based on works from Bandt and Pompe (2002), generalised by Mohr *et al.* (2020b) for multivariate time series. For this, let $X_t \in \mathbb{R}^{m \times T}$ be a (multivariate) time series and $X_t \in \mathbb{R}^{m \times T \times n}$ be the

Figure 3.8.: All $d!$ possible univariate ordinal patterns of order $d = 3$.

reference database of $n \in \mathbb{N}$ (multivariate) time series. In the case of $m = 1$, the time series is univariate. For a better understanding, we start with univariate ordinal patterns that encode the up and downs in a time series by determining the total order between two or more neighbouring values of the original time series. The encoding gives a good abstraction, an approximation, of the overall behaviour or generating process. Univariate ordinal patterns are formally defined as follows.

Definition 3.5.1 (Univariate Ordinal Pattern)**.** A vector $(x_1, ..., x_d) \in \mathbb{R}^d$ has *ordinal pattern* $(r_1, ..., r_d) \in \mathbb{N}^d$ of order $d \in \mathbb{N}$ if $x_{r_1} \geq ... \geq x_{r_d}$ and $r_{l-1} > r_l$ in the case $x_{r_{l-1}} = x_{r_l}$.

Figure 3.8 depicts all possible ordinal patterns of order $d = 3$ of a vector $(x_1, x_2, x_3) \in \mathbb{R}^3$.

For a *multivariate* time series

$$((x_t^i)_{i=1}^m)_{t=1}^T, \tag{3.5.1}$$

each variable x^i for $i \in 1, ..., m$ depends not only on its past values but also has some dependency on other variables. To establish a total order between two time points $(x_t^i)_{i=1}^m$ and $(x_{t+1}^i)_{i=1}^m$ with m variables is only possible if $x_t^i > x_{t+1}^i$ or $x_t^i < x_{t+1}^i$ for all $i \in 1, ..., m$. Therefore, there is no trivial generalisation to the multivariate case. An intuitive idea, based on some theoretical discussion in (Antoniouk *et al.*, 2014; Keller, 2012) and introduced in (Mohr *et al.*, 2020b), is to store univariate ordinal patterns of all variables at a time point t all together into a single symbol, which is defined as follows.

Definition 3.5.2 (Multivariate Ordinal Pattern)**.** A matrix $(x_1, ..., x_d) \in \mathbb{R}^{m \times d}$ has *multivariate ordinal pattern* (MOP) of order $d \in \mathbb{N}$

$$\begin{pmatrix} r_{11} & \cdots & r_{1d} \\ \vdots & \ddots & \vdots \\ r_{m1} & \cdots & r_{md} \end{pmatrix} \in \mathbb{N}^{m \times d} \tag{3.5.2}$$

if $x_{r_{i1}} \geq ... \geq x_{r_{id}}$ for all $i = 1, ..., m$ and $r_{il-1} > r_{il}$ in the case $x_{r_{il-1}} = x_{r_{il}}$.

For $m = 1$ the multivariate case matches with the univariate case in Definition 3.5.1. Figure 3.9 depicts all $(d!)^m$ possible MOPs of order $d = 3$ and number of variables $m = 2$. The number of possible MOPs $d!$ increases exponentially with the number of variables m, i.e.,

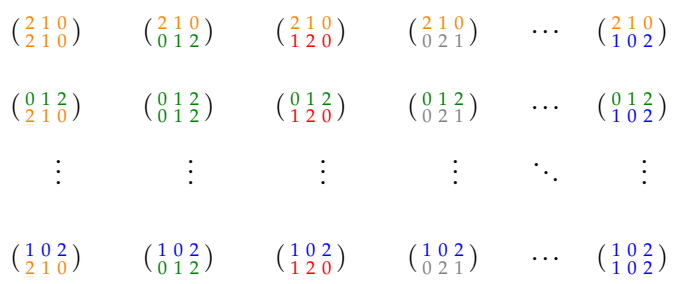


Figure 3.9.: All $(d!)^m$ possible multivariate ordinal patterns of order $d = 3$ with $m = 2$ variables.

$(d!)^m$. Therefore, if d and m are too large, depending on the application, each pattern occurs only rarely or not at all. Thus, with a very large number of symbols, the relative frequency of each symbol occurring is very small and as a result, the distribution of all symbols is almost uniform (Mohr *et al.*, 2020b). Nevertheless, for a small order d and sufficiently large T the use of MOPs can lead to higher accuracy in learning tasks, e.g., classification (Mohr *et al.*, 2020b) because they incorporate interdependence of the spatial variables in the multivariate time series as argued by the authors.

To symbolise a multivariate time series $X_t \in \mathbb{R}^{m \times T}$ each pattern is identified with exactly one of the *ordinal pattern symbols* $o \in \{1, 2, ..., d!\}$, before each point $t \in \{d, ..., T\}$ is assigned its ordinal pattern symbol of order $d \ll T$. The order d is chosen much smaller than the length T of the time series to look at small windows in a time series and their distributions of up and down movements. To assess long-term trends, delayed behaviour is of interest, revealing various details of the structure of the time series. The time delay $\tau \in \mathbb{N}_{>0}$ is the delay between successive points in the symbol sequences.

Based on this understanding, in the course of this work we introduce an approach to identify temporal symmetries at the micro level, i.e., in time series resulting from observing random variables over time, in order to further use these structures at the macro level of the DPRM.

3.6 Measuring Divergence between Probability Distributions

All the adjustments and additions to the online variant of LDJT that we propose in order to deal with the various challenges arising from non-stationarity make onlineLDJT an approximate rather than an exact inference algorithm. To measure the impact of the proposed additions on exact inference, we use the Kullback Leibler divergence (KLD) divergence (Kullback and Leibler, 1951). The KLD is additive as it satisfies the chain rule of probability and therefore allows the calculation of the KLD according to the factorisation of a model (here the DPRM), making it suitable for use throughout this work.

We measure the distance between the true belief state ϕ, and an approximation ψ of the true belief state, i.e., the divergence between a distribution and its approximation.

Definition 3.6.1 (Kullback-Leibler Divergence)**.** For discrete models, the KL divergence of two probability distributions ϕ, ψ over the same probability space Ω is given by

$$D_{KL}(\phi, \psi) = \sum_{\omega \in \Omega} \phi(\omega) \log \left(\frac{\phi(\omega)}{\psi(\omega)} \right).$$

If $\frac{\phi(\omega)}{\psi(\omega)}$ becomes 0, resulting in an undefined $log(0)$, we assign $D_{KL}(\phi, \psi) = 1$, i.e., indicating maximum dissimilarity.

The KL divergence is a non-negative measure $D_{KL}(\phi, \psi) \geq 0$ with $D_{KL}(\phi, \psi) = 0$ indicating that both distributions are equal.

Part I

Dynamic Domain Sizes

4 Exploring Dynamic Domains in DPRMs

The sizes of sets of domain objects determine the overall joint distribution behind a DPRM unrolled over time. While information about objects may differ for different points in time, the set of objects under consideration is the same for all time points in *standard* DPRMs, i.e., DPRMs as originally defined by Gehrke *et al.* (2018). Providing semantics for dynamically changing sets of domain objects is important for modelling real-world applications, as it is difficult to know in advance which objects an intelligent system needs to reason about. This becomes especially important in the area of online algorithms, such as the online variant of LDJT on DPRMs, on which we rely in this work. In this part, we extend the semantics of DPRMs for dynamically changing sets of domain objects to be substituted for variables used in the model, such that sparse domain models can be specified and efficient query answering algorithms can be provided for practically relevant application domains.

Prior to formally introducing the extension of the existing formalism by different strategies for domain expansion in the next chapter of this part, we discuss the characteristics and challenges of domain size changes in **Section 4.1**, followed by giving a decision framework along which decisions upon adding domain objects to an existing domain are made in **Section 4.2**. Further, we provide auxiliary functions for syntactically integrating domain extensions into DPRMs in **Section 4.3**.

The results were mainly published in the following conference paper.

Nils Finke, Tanya Braun, Marcel Gehrke, and Ralf Möller. Dynamic Domain Sizes in Temporal Probabilistic Relational Models. *The International FLAIRS Conference Proceedings*, 34, 2021.

4.1 Characteristics and Challenges with Dynamic Domain Sizes

A static and fixed domain size limits the scope of DPRMs for real-world tasks. Domain sizes often change at runtime, making it necessary to *re-construct* a model as the models probability distirbutions might be affected. However, the model construction process, which we won't elaborate on here, can be quite time-consuming, especially for larger models with multiple domain objects and relationships. Considering the example application from dry-bulk shipping, domain size change can be characterised as follows. In shipping, vessels are

continuously joining or leaving the market due to a variety of different reasons. For example, vessels are newly built, dumped, going into maintenance, or laid up if freight rates do not cover running costs anymore. We assume that these domain changes are directly observable and are not affected by uncertainty. In general, we differentiate between the following cases: An object (here, a vessel) is (a) entirely removed from the model (*Removal*), (b) completely new to the model (*Addition*), or (c) is temporally removed and at a later stage added to the model again (*Restoration*). Note that restoration might not be feasible for all use cases. One good example of restoration is if a vessel is laid up at a particular location and later joins the market again. Such a vessel will be available again in the same location. Thus, events of a previous time step, like the position of a vessel, still apply to the current time step. In the following two chapters of this work, we focus on giving the formalism and strategies on how to add a new domain object, since removing and restoring them is a sub-case of addition.

Domain size changes refer to enlarging or reducing domains of the DPRM, which has an effect on the overall joint distribution behind a DPRM. The effect on the full joint distribution manifests itself in two ways:

(a) The **state space expands** as the number of possible outcomes in the state space increases, since for each new domain object different observations w.r.t. the randvars in the model might occur that must be accounted for in the full joint distribution.

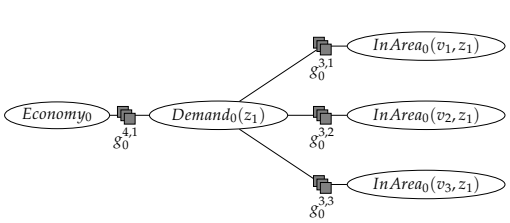
(b) The **marginal distributions shift**, i.e., the probabilities of occurrence of the range values of the individual randvars may shift, provided the distributions are not *projective*.

In this section, we explain these two effects in more detail. First, we discuss the expansion of the state space and consider a fully grounded model to illustrate this. Then, we demonstrate how lifting can help address this issue but also highlight the challenges it implies, ultimately leading us to examine the problem in terms of the projectivity of probability distributions.

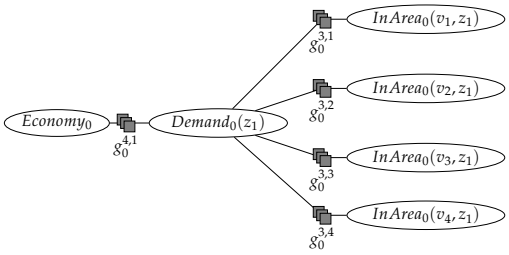
State Space Expansion

We consider the following example from the shipping domain and explain the effects of adding a new domain object to the state space of an exisiting probabilistic model.

Example 4.1.1 (State Space Expansion in a Propositional Graphical Model)**.** *We consider a submodel of the full model G^{ex} presented in Fig. 3.1, here for time step $t = 0$ with PRVs $Economy_0$, $Demand_0(Z)$, $InArea_0(V, Z)$ and parfactors g_0^4, and g_0^3 for one zone $\mathcal{D}(Z) = \{z_1\}$ and three vessels $\mathcal{D}(V) = \{v_1, v_2, v_3\}$ as presented in Fig. 4.1a. The state space Ω equals all possible combi-*



(a) Representation of a DPRM with PRVs $Economy_0$, $Demand_0(Z)$, $InArea_0(V, Z)$ with $\mathcal{D}(Z) = \{z_1\}$ and $\mathcal{D}(V) = \{v_1, v_2, v_3\}$ in the propositional case.



(b) Expanding the domain $\mathcal{D}(V)$ to $\mathcal{D}(V) = \{v_1, v_2, v_3, v_4\}$ by adding a new domain object, here a vessel v_4, i.e., adding a new randvar for $InArea_0(v_4, z_1)$.

Figure 4.1.: Representation of an example model to explain the effects of domain expansion having a propositional graphical model.

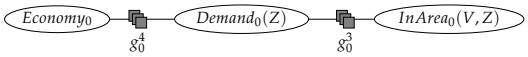
nations of range values of the randvars in the model, i.e., here

$$\Omega = \mathcal{R}(Economy_0) \times \mathcal{R}(Demand_0(z_1)) \times \mathcal{R}(InArea_0(v_1, z_1)) \times \mathcal{R}(InArea_0(v_2, z_1)) \times \mathcal{R}(InArea_0(v_3, z_1)). \tag{4.1.1}$$

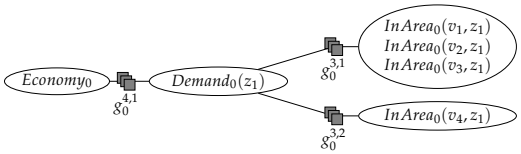
As the randvar $Demand_0(Z)$ can take three range values $\{low, medium, high\}$ and both boolean randvars $Economy_0$ and $InArea_0(V, Z)$ can take two range values, overall the state space Ω has a size of 48*. Adding a new domain object, e.g., a vessel v_4 to $\mathcal{D}(V)$ results in a more complex model in the propositional case as depicted in Fig. 4.1b. As the number of domain objects in the model increases, the joint probability distribution becomes more complex as more randvars are added to the model. More specifically, the full joint distribution expands by a randvar $Demand(v_4, z_1)$. Accordingly, the size of the state space Ω expands to a size of* 96*. This appears if the previous equation Eq.* (4.1.1) *is extended by another factor $\mathcal{R}(InArea_0(v_4, z_1))$.*

By expanding the state space in a probabilistic model, inference possibly becomes more computationally complex. Lifting helps solving this problem if symmetries can be exploited but also brings up other challenges. If symmetries are exploited, domain objects are arranged in groups of domain objects with one domain object as a representative for the whole group of domain objects. The challenge of a domain extension is to select the set of domain objects in which the new domain object can be included, which we will explain in more detail using the following examples.

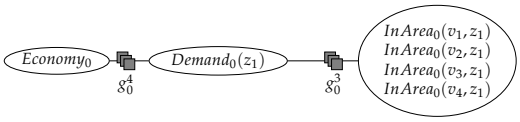
Example 4.1.2 (State Space Expansion in a Lifted Graphical Model)**.** *We consider the same example as depicted in Fig. 4.1a. In Fig. 4.1a, each domain object of $\mathcal{D}(V)$ and $\mathcal{D}(Z)$ is represented by a single parameterised randvar for $Demand_0(Z)$ and $InArea_0(V, Z)$. Moreover, each randvar is connected by a factor $g_0^{3,1}$, $g_0^{3,2}$, and $g_0^{3,3}$, respectively, where each factor can possibly entail its own adjusted potential function denoting potentials (or probabilities after count normalisation) for the*



(a) Lifted Representation of the model depicted in Figure 4.1a.



(b) Option 1: Domain expansion leading to state expansion due to non-projectivity.



(c) Option 2: Counteracting domain expansion due to projectivity.

Figure 4.2.: Illustration of the effects of a domain expansion for DPRMs. **Note that in this and the following illustrations, we enumerate the randvars directly with the individual domain object combinations in the nodes for the PRVs and omit the constraint notation here for ease of reading.**

occurrence of an event. Provided the potential function is the same for all factors $g_0^{3,1}$, $g_0^{3,2}$ and $g_0^{3,3}$ (symmetrical), the model can be transformed into a lifted version as depicted in Fig. 4.2a. Here all three factors are combined in a common parfactor g_0^3, i.e.,

$$g_0^3 = \phi_0^3(Demand_0(Z), InArea_0(V,Z))_{|\,C^3} \quad with \quad C^3 = \big((V,Z), \{(v_1,z_1),(v_2,z_1),(v_3,z_1)\}\big)$$

limited by a constraint C^3 to domain objects $\mathcal{D}(Z) = \{z_1\}$ and $\mathcal{D}(V) = \{v_1, v_2, v_3\}$. Now, analogous to the previous Example 4.1.1, the vessel domain $\mathcal{D}(V)$ is extended at runtime by another domain object v_4. Then, there are two options to include the new domain object in the lifted model. Either, the constraint C_3 is adjusted to include the domain object tuple (v_4, z_1) keeping the model in a fully lifted state as depicted in Fig. 4.2c, i.e.,

$$g_0^3 = \phi_0^3(Demand_0(Z), InArea_0(V,Z))_{|\,C^3} \quad with \quad C^3 = \big((V,Z), \{(v_1,z_1),(v_2,z_1),(v_3,z_1),(v_4,z_1)\}\big) \tag{4.1.2}$$

or alternatively, the model is shattered by splitting the parfactor g_0^3 into two parts $g_0^{3,1}$ and $g_0^{3,2}$, i.e.,

$$\begin{aligned} g_0^{3,1} &= \phi_0^{3,1}(Demand_0(Z), InArea_0(V,Z))_{|\,C^{3,1}} \quad with \quad C^{3,1} = \big((V,Z), \{(v_1,z_1),(v_2,z_1),(v_3,z_1)\}\big) \\ g_0^{3,2} &= \phi_0^{3,2}(Demand_0(Z), InArea_0(V,Z))_{|\,C^{3,2}} \quad with \quad C^{3,2} = \big((V,Z), \{(v_4,z_1)\}\big) \end{aligned} \tag{4.1.3}$$

Of course, the state space of the lifted model expands in this case as well, just like in the propositional/grounded case, but by exploiting the symmetries in the data, a more compact presentation can be preserved.

As presented in this example, in the lifted model one faces the challenge of deciding

whether to split the lifted model (Option 1) or to keep the lifted representation (Option 2). However, which option comes into consideration or can be considered at all depends very much on whether, for a new domain object (such as v_4 here), the potential function encoded in the respective parfactors of the model also represent the likelihood of events to occur for the new domain object well. This leads us to assess the similarities in terms of potential functions between domain objects, which in turn motivates considering the second effect of the shifts in the marginal distributions due to an expansion of one of the model domains.

Shift in Marginal Distributions

When incorporating a new domain object into the model, it's important to choose that option to avoid the risk of unwanted shifts in the marginal distributions. Let us consider the following example.

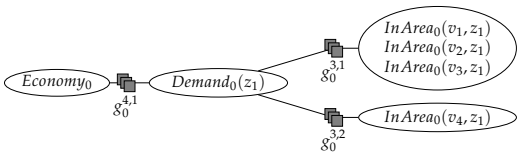
Example 4.1.3 (The Need for Assessing Similarity Between Domain Objects to Avoid Marginal Distributions to Shift)**.** *Let's consider the two tables below denoting potential functions, with the left table referring to a potential function for the group of domain object tuples (z_1, v_1), (z_1, v_2) and (z_1, v_3) (as encoded in parfactor g_0^3 prior to adding v_4) and the right table to a potential function for (z_1, v_4) describing the likelihood of events for randvars $Demand_0(Z)$ and $InArea_0(V, Z)$. Note that the potentials here are randomly chosen and are just used for illustration purposes.*

$Demand_0(z_1)$	$InArea_0(v_1, z_1)$ $InArea_0(v_2, z_1)$ $InArea_0(v_3, z_1)$	ϕ_0^3
high	*true*	10
medium	*true*	2
low	*true*	2
high	*false*	2
medium	*false*	2
low	*false*	2

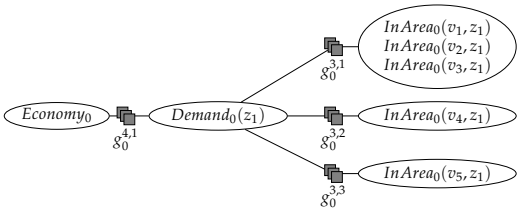
$Demand_0(z_1)$	$InArea_0(v_4, z_1)$	ϕ_0^3
high	*true*	1
medium	*true*	1
low	*true*	1
high	*false*	5
medium	*false*	1
low	*false*	1

At this point, we will not go into detail on how these potential functions can be determined. A trivial approach here is by counting occurrences of range value combination (counting the different states the model has been in in the past) for the specific domain objects.

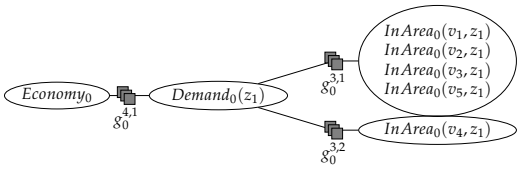
The two potential functions depicted above differ in terms of the potentials for the occurrence of the range values of the variables $Demand_0(Z)$ and $InArea_0(V, Z)$. More specifically, the potential for $Demand_0(z_1) = high$ to be high in zone z_1 is greater whenever the vessel v_4 is not in that zone, i.e., $InArea_0(v_4, z_1) = false$, which is opposite to the potential function for the other domain objects. Given the potential function for the new domain object (such as the one as shown in the table above on the right), it is possible to assess which option must be chosen. So in this case either shattering the model and introducing a new parfactor $g_0^{3,2}$ for the entity tuple (v_4, z_1), see Eq. (4.1.3)*, or adjusting the constraint of the existing parfactor $g_0^{3,1}$, see Eq.* (4.1.2)*. However, including the new domain object by simply changing the constraint C_3, i.e., Eq.* (4.1.2)*, would affect the accuracy of*



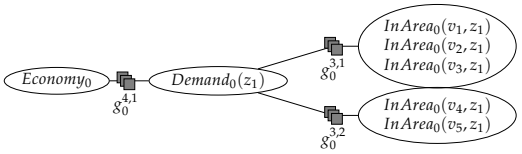
(a) Example model as depicted in Figure 4.2a.



(b) Option 1: New split parfactor $g_0^{3,3}$.



(c) Option 2: Adjust constraint of split parfactor $g_0^{3,1}$.



(d) Option 3: Adjust constraint of split parfactor $g_0^{3,2}$.

Figure 4.4.: Options for adding a new domain object in a more complex example. **Note that in this and the following illustrations, we enumerate the randvars directly with the individual domain object combinations in the nodes for the PRVs and omit the constraint notation here for ease of reading.**

predictions in inference. More specifically, changing the condition C_3 to include the new domain object would also require shifting the potentials of ϕ_0^3 to accommodate for the higher likelihood for $InArea_0(v_4, z_1) = false$ and $Demand_0(z_1) = high$. However, if this had been done accordingly, it would have affected the accuracy of reasoning about the already existing domain objects as the potential function in g_0^3 becomes more generalised.

In general, it is beneficial to not further shatter the model in order to maintain a lifted representation of the model, but this is not always feasible. Determining the most precise option for integrating a new domain object into one of the domains of an existing model is a complex task. This complexity even further increases, as we illustrate in the following example.

Example 4.1.4 (Options Depend on the Degree a Model Is Shattered)**.** *In the previous example, we assumed a fully lifted model. After adding another domain object v_4, we have reached a model state as depicted in Fig. 4.4a. Now, another domain object v_5 has to be included in the vessel domain of the model. The sub-figures Fig. 4.4(b)-(d) depict the different options for adding a domain object v_5:*

- *Another evidence parfactor $g_0^{3,3}$ is created limited by a constraint to the new domain object v_5, or more specifically to the tuple (v_5, z_1) as depicted in Fig. 4.4b.*
- *The constraint of the existing split parfactor $g_0^{3,1}$ is adjusted to include the new domain object v_5 as depicted in Fig. 4.4c.*

- *The constraint of the existing split parfactor $g_0^{3,2}$ is adjusted to include the new domain object v_5 as depicted in Fig. 4.4d.*

The example demonstrates that the further a model is split, i.e., grounded, the more options are available to either include a new domain object in an existing split parfactor by adjusting its constraint or to introduce further split parfactors.

The example shows that the number of available options (see Fig. 4.4) depend on the degree a model is shattered and on the validity for the different instances of the parfactors present potential functions for a new domain object. This is true with one strict exception, namely if the full joint distribution of the model is projective, a concept we introduce in the following for completeness.

The Concept of Projectivity

In the previous example, we assumed that we knew the potential function for the new domain object. However, this may not be the case in practice and is not needed under the assumption of *projectivity*. The concept of projectivity refers to the property that a distribution, e.g., a distribution that results after count normalisation of a parfactor's potential function, is unaffected by a change in the domain. This property is important as it guarantees that the new domain object can be inserted into an existing domain without having to recalculate the entire joint distribution as there is no effect on the marginals. We define projectivity based on the definition by Jaeger and Schulte (2018) for DPRMs as follows.

Definition 4.1.1 (Projectivity)**.** Given a parfactor $g = \phi(\mathcal{A})_{|(\mathcal{X},C_\mathcal{X})}$ with $\mathcal{A} = (A^1, \ldots, A^n)$ a sequence of PRVs, and $(\mathcal{X}, C_\mathcal{X})$ a constraint that limits ϕ to a set of entities $C_\mathcal{X}$ made up from the respective domains $\mathcal{D}(X)$ of each logical variable X in the sequence $\mathcal{X} \in lv(\mathcal{A})$, then ϕ is said to be projective, if the following condition is satisfied:

$$\frac{1}{Z}\phi(\mathcal{A})_{|(\mathcal{X},C_\mathcal{X})} = \frac{1}{Z}\phi(\mathcal{A})_{|(\mathcal{X},C'_\mathcal{X})}$$

where $C'_\mathcal{X} \subset C_\mathcal{X}$ is a subset of entities and Z the normalisation count.

In other words, the count-normalised distribution of $\phi(\mathcal{A})_{|(\mathcal{X},C_\mathcal{X})}$ is independent of which subset of variables in $C_\mathcal{X}$ is used to compute it. This property ensures that the distribution remains unchanged even if we condition on or marginalise over different subsets of variables, i.e., $C'_\mathcal{X} \subset C_\mathcal{X}$ If this property is given, the domain can be expanded without having to recompute the joint distribution entirely. However, as Jaeger and Schulte (2018) shows, the criteria for projectivity are very restrictive and therefore difficult to achieve. This becomes especially apparent in real-world applications, i.e., in scenarios with complex domains with many domain objects and relationships.

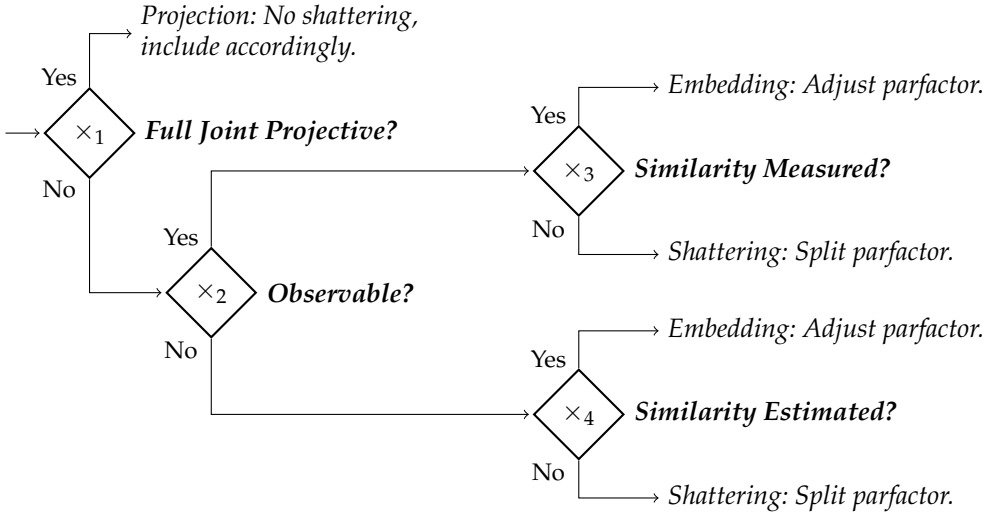


Figure 4.5.: Decision path on how to add a new domain object to an existing domain.

In the previous examples, we assumed that the past of a new object to be added was observable, i.e., historical data for this specific domain object is given, to come up with a potential function (or multiple) for the domain object under consideration. This allows the relevant potential functions to be compared to discover whether there is already a potential function in the model for other domain objects that is identical, or at least nearly identical, to the potential function for the new domain object. However, this condition may not always be met, which requires the use of an estimator to assess the similarity between the new domain object and the existing ones encoded in the model, which is another challenge in enabling for dynamic domain support. In this section, we have given an outline of the options and challenges when adding a new domain object. In the next section, we informally summarise these options again in a framework that outlines the decision path when adding a new domain object to an existing domain.

4.2 An Informal Decision Framework

As outlined in the previous section, including a new domain object into an existing domain of a potentially already shattered model is not a trivial task. Figure 4.5 depicts a decision tree outlining the decision-making process. Nodes in the decision tree marked with $\times$ indicate points of a decision to be made. The following three decisions must be made when adding a domain object:

- Decision $\times_1$: Is the full joint distribution projective? If this property is given, it is beneficial in two ways. The full joint distribution remains correct as the factorised distributions are independent of the set of objects used for inference. That is, (a) adding

a domain object is a straightforward task, and, (b) the runtime in inference remains consistently high as the model is not shattered at all.

- Decision $\times_2$: If the full joint distribution is not projective, then depending on whether the new domain object is (was) observable, potential functions for all the parfactors affected by a domain change with respect to a new domain object can be derived and subsequently used for similarity assessment.
- Decision $\times_3$: If the new domain object is observable, we can determine its corresponding parfactor potential functions ϕ, which can be used to measure similarity to potential functions already encoded in the model for domain objects other than the new domain object. If the existing potential functions accurately represent the behaviour of the new domain object, there are two ways to proceed: (a) the existing constraints of parfactors are adjusted to embed the new domain object, or, (b) if the potential functions do not adequately capture the behaviour of the new object, the model must be further modified, resulting in shattering.
- Decision $\times_4$: If the domain object is not observable, we cannot determine its corresponding parfactors' potential functions ϕ. Thus, we cannot directly measure similarity as in case $\times_3$, but instead, we can only estimate similarity to other domain objects. However, similar to $\times_3$, the same two options (a) and (b) can be considered.

In the following section, we look at the detailed syntactical procedures for implementing a domain extension in DPRMs. So far, we have only demonstrated domain expansion using a simple example with a single parfactor. However, in more complex models involving multiple parfactors, all parfactors connecting PRVs concerning the same domain need to be adapted to accommodate the domain expansion. To facilitate this, we introduce auxiliary constructs in the following section before providing strategies for the different cases in the decision framework in the next chapter.

4.3 Domain Change Handling in DPRMs

In general, domain size changes can be encoded within DPRMs without changing the model structure itself. In DPRMs, objects of the model's domain are encoded through PRVs that represent sets of those domain objects with a single representative randvar. In comparison, in the propositional case, randvars are duplicated in the model for each object they represent, resulting in a much larger model. Thus, encoding changing domain sizes within DPRMs relates to just changing the set of objects represented through the logvars of a model. More specifically, adjusting parfactor constraints that limit the potential function encoded in a parfactor to a subset of the domain objects represented by the PRVs associated with the respective parfactors. In the preceding two sections, we explored the process of extending an existing domain on a submodel, in which the modification of only one parfactor, specifi-

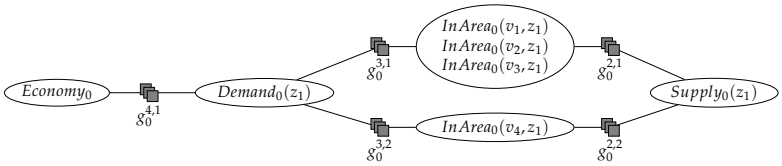


Figure 4.6.: Extended example model of the shattered model in Figure 4.2a. The model contains another PRV $Supply_t(Z)$ and parfactor g_0^2. **Note that in this and the following illustrations, we enumerate the randvars directly with the individual domain object combinations in the nodes for the PRVs and omit the constraint notation here for ease of reading.**

cally its constraint, was sufficient. However, since DPRMs factorise the full joint probability distribution through sets of parfactors that represent a partial distribution of the full joint distribution, all those parfactors that are affected by a domain change must be identified and adjusted, a process which we formalise next.

Identify Affected Parfactors

In summary, a DPRM with a local model $G_t = \{g_t^i\}_{i=1}^n$ at time step t is split due to evidence w.r.t. its parfactors such that its structure remains:

$$G_t = \{g_t^{i,1}, \ldots, g_t^{i,k}\}_{i=1}^n, \quad k \in \mathbb{N}^+ \tag{4.3.1}$$

Every parfactor g_t^i can have up to $k \in \mathbb{N}^+$ splits

$$g_t^{i,j} = \phi_t^{i,j}(\mathcal{A}^i)_{|C^{i,j}}, \quad 1 \leq j \leq k \tag{4.3.2}$$

with $\mathcal{A}^i$ as a sequence of the same PRVs but with different constraint $C^{i,j}$, varying functions $\phi_t^{i,j}$ due to evidence, and k depending on the degree a parfactor is shattered due to evidence. Now, to include a new domain object in a model as represented in Eq. (4.3.1), all *parfactors affected* by a domain change must first be identified. We consider a parfactor affected if the parfactor combines PRVs $A(L_1, \ldots, L_n) \in \mathcal{A}$ that are parameterised with logvars $(L_1, \ldots, L_n)$ such that a new object l is of the type of at least one of the logvars, i.e.,

$$\exists L \in (L_1, \ldots, L_n) : dm(l) = \mathcal{D}(L) \tag{4.3.3}$$

and must be included in its domain $\mathcal{D}(L)$. Note that we assume we know to which domain $\mathcal{D}(L)$ a new domain object must be added. However, a shattered model as per Eq. (4.3.2) has potentially k splits of the same parfactor, i.e., the check as given in Eq. (4.3.3) yields the *same* parfactor multiple times limited by the specific parfactor constraints to different domain objects, a situation which we explain in more detail in the next example.

Example 4.3.1 (Domain Change Affected (Split) Parfactors)**.** *We consider a shattered model as illustrated in Fig. 4.6. Original parfactors g_0^3 and g_0^2 are split, e.g., due to events, leading to (split) parfactor $g_0^{3,1}$, $g_0^{3,2}$ and $g_0^{2,1}$, $g_0^{2,2}$. As the local model G_0 is shattered the local model G_0 has the form*

$$G_0 = \{g_0^{2,1}, g_0^{2,2}, g_0^{3,1}, g_0^{3,2}, g_0^{4,1}\}. \tag{4.3.4}$$

Now, we assume that we must include a new domain object v_5 in the vessel domain $\mathcal{D}(V)$ in the model that is in the state as per Eq. (4.3.4)*. In this example, we include the new domain object by adjusting existing parfactor constraints and not by further shattering the model. The (split) parfactors $g_0^{2,1}$, $g_0^{2,2}$, $g_0^{3,1}$, and $g_0^{3,2}$ are affected by a domain change of $\mathcal{D}(V)$ since all are connected to the PRV $InArea_0(V, Z)$, which is the only PRV, which binds logvar V and therefore reflects the domain of all vessels $\mathcal{D}(V)$ into which v_5 has to be included. The remaining parfactor $g_0^{4,1}$ binds PRVs $Economy_0$ and $Demand_0(Z)$ and therefore is not associated with the vessel domain $\mathcal{D}(V)$. Therefore, the set of affected parfactors as per the local model G_0 is*

$$G_0{}' = \{g_0^{2,1}, g_0^{2,2}, g_0^{3,1}, g_0^{3,2}\}. \tag{4.3.5}$$

The parfactors $g_0^{3,1}$ and $g_0^{3,2}$ are both limited by constraints to tuples of domain objects $\{(v_1, z_1), (v_2, z_1), (v_3, z_1)\}$ and $g_0^{3,1}$ and $g_0^{3,2}$ accordingly to (v_4, z_1) only. However, not all parfactors in $G_0{}'$ need to be adjusted to include the new domain object v_5 in $\mathcal{D}(V)$, but only a subset of $G_0{}'$. If we were to adjust all constraints of all parfactors and include v_5, the potential function $\phi_0^{3,1}$ in $g_0^{3,1}$ would conflict with the potential function $\phi_0^{3,2}$ in $g_0^{3,2}$ because, for example, $g_0^{3,1}$ has a potential function with events for $InArea_0(V, Z) = true$ and $g_0^{3,2}$ has a potential function with events for $InArea_0(V, Z) = false$, i.e., are contradictory.

In order to adjust only the relevant parfactors in the model, and to avoid the conflict demonstrated in the previous example, we divide the set of affected parfactors into subsets, making it easier to decide which parfactors to adjust.

Partitioning Affected Parfactors

The partitioned set of affected parfactors can then be used as an auxiliary construct to avoid this problem. The idea behind partitioning is to obtain subsets of parfactors so that for each subset itself, a parfactor is selected whose constraint is adjusted to encode the new domain object.

Example 4.3.2 (Parfactor Partitioning)**.** *We continue with the Example 4.3.1. To avoid the previously mentioned conflict, the affected parfactors as given in Eq.* (4.3.5) *are divided into subsets of parfactors such that the form is*

$$G_0{}'' = \{\{g_0^{2,1}, g_0^{2,2}\}, \{g_0^{3,1}, g_0^{3,2}\}\}. \tag{4.3.6}$$

Now, to include a new domain object v_5, one parfactor must be selected from each of the subsets $\{g_0^{2,1}, g_0^{2,2}\}$ and $\{g_0^{3,1}, g_0^{3,2}\}$. For each selected parfactor, the parfactors constraint is adjusted to include the new domain object v_5 accordingly. This simplifies the process, as it ensures that if, for example,

- *the constraint $C^{3,1}$ of $g_0^{3,1}$ is adjusted such that $C^{3,1} = \big((V,Z), \{(v_1,z_1),(v_3,z_1),(v_3,z_1)\}\big)$ is now $C^{3,1} = \big((V,Z), \{(v_1,z_1),(v_3,z_1),(v_3,z_1),(v_4,z_1)\}\big)$,*
- *the remaining constraint $C^{3,2}$ of $g_0^{3,2}$ is not adjusted in this case.*

To be precise, this step is required as the original parfactor g_0^3 was shattered into $g_0^{3,1}$ and $g_0^{3,2}$, to represent an adapted potential function for a specific subset of domain-object pairs in $\mathcal{D}(V) \times \mathcal{D}(Z)$. This guarantees the accurate representation of the model state/behaviour for each corresponding domain-object tuple, and therefore the new domain object v_5 must be added by either changing $C^{3,1}$ or $C^{3,2}$.

The approach used to identify parfactors that are affected by changes in the domain and subsequently partition those affected parfactors is referred to as *parfactor partitioning* as follows. More formally, given a set of parfactors $G_t = \{g_t^{i,1}, \ldots, g_t^{i,k}\}_{i=1}^n$ as defined in Eq. (4.3.1), parfactor partitioning is rearranging G_t into a set of sets of parfactors

$$\begin{aligned} G_t{}' &= \{\{g_t^{1,1}, \ldots, g_t^{1,k}\}, \ldots, \{g_t^{i,1}, \ldots, g_t^{i,k}\}\} \\ &=: \{G_t^1, \ldots, G_t^i\} \end{aligned} \tag{4.3.7}$$

such that all parfactors $g \in G_t^i$ with $g = \phi(\mathcal{A})_{|C}$ have the same sequence of PRVs $\mathcal{A}$.

Both the identification of affected factors and the partitioning are combined into one single procedure as depicted in Algorithm 3. The following procedure takes as inputs a local model G_t and a domain object x to be included in an existing domain, and returns as a result a partitioned set of parfactors affected by such a domain expansion. The partitioning is then being used in each respective strategy that we propose to properly include a new domain object into an existing domain. Once a parfactor has been determined to be adapted, its constraints have to be recalculated. For this purpose, we specify a domain change operator as follows.

Domain Change Operator

The inclusion of a new domain object into an evidence parfactor requires the re-creation of the constraint of that specific parfactor. We highlight constraint recreation in the following example.

Example 4.3.3 (Adding Domain Objects: Parfactor Constraint Recreation)**.** *We consider having n vessels in the vessel domain $\mathcal{D}(V) = \{v_1, v_2, \ldots, v_n\}$ and m zones in the zone domain $\mathcal{D}(Z) = \{z_1, z_2, \ldots, z_m\}$. If a parfactor g_0^2 is split*

Algorithm 3 Return the partitioned set of parfactors affected by domain expansion

procedure PARTITIONEDAFFECTEDPARFACTORS(Local Model G_t, Domain Object l):
  $G_t' \leftarrow \emptyset$
  for $g_t^{i,j} \in G_t$ **do**
    $\mathcal{A} \leftarrow$ Get sequence of PRVs $\mathcal{A} = (A^1, \ldots, A^n)$ in $g_t^{i,j}$
    for $A(L_1, \ldots, L_n) \in \mathcal{A}$ **do**
      if domain object l is in one of $\mathcal{D}(L_1), \ldots, \mathcal{D}(L_n)$ ▷ see Equation (4.3.3)
      then $G_t' \leftarrow G_t' \cup \{g_t^{i,j}\}$
  $G_t'' \leftarrow$ Partition parfactors in G_t' ▷ see Equation (4.3.7)
  return G_t''

- *as a result of, e.g., a filter query* $P(Supply_0(z_1) \mid InArea_0(v_1, z_1), InArea_0(v_2, z_1))$,
- *then an evidence parfactor* $g_0^{2,2} = \phi_0^{2,2}(Supply_0(Z), InArea_0(V, Z))_{|C^{2,2}}$ *is created as a copy of the original parfactor* g_0^2, *and,*
- *is limited by a constraint* $C^{2,2} = (\mathcal{X}, C_\mathcal{X})$ *with* $\mathcal{X} = (V, Z)$ *logvars and* $C_\mathcal{X}$ *a set of* 2*-tuples, i.e., the subset* $C_\mathcal{X}$ *is the result of the cross product of all objects of sets* $V = \{v_1, v_2\}$ *and* $Z = \{z_1\}$ *yielding* $C_\mathcal{X} = \{(v_1, z_1), (v_2, z_1)\}$.

In case a new domain object z_2 *has to be included in* $\mathcal{D}(Z)$ *and in parfactor* $g_1^{2,2}$, *the parfactor constraint* $C^{2,2}$ *is adjusted such that* $C_\mathcal{X}$ *also includes* z_2. *Therefore,* $Z = \{z_1\} \cup \{z_2\}$ *is adjusted to include* z_2, *followed by recreating* $C_\mathcal{X}$ *yielding* $C_\mathcal{X} = \{(v_1, z_1), (v_1, z_2), (v_2, z_1), (v_2, z_2)\}$.

To recreate the constraints to include a new domain object in an existing parfactor, specifically re-creating $C_\mathcal{X}$ in a parfactors constraint, we introduce an operator *Inclusion*.

Definition 4.3.1 (INCLUSION)**.** A parfactor $g_e = \phi_e(\mathcal{A})_{|C_e}$ encodes evidence for PRVs $A \in \mathcal{A}$ by means of a specific potential function denoted by ϕ_e. Constraint $C_e = (\mathcal{X}, C_\mathcal{X})$ limits the logvars in ϕ_e to a subset $C_\mathcal{X} \subseteq \times_{i=1}^n \mathcal{D}(X_i)$ of domain objects $D(X_i)$ with logvars $X_i \in \mathcal{X}$. To include any new object x, the constraint C_e is adjusted such that the set $C_\mathcal{X} \subseteq \times_{i=1}^n \mathcal{D}(X_i)$ in terms of n-tuples $(x_i, \ldots, x_j) \in (\mathcal{D}(X_1), \ldots, \mathcal{D}(X_n))$ is extended by n-tuples concerning the new object x, i.e,

$$C_\mathcal{X} = \times_{i=1}^n [dm(x) = \mathcal{D}(X_i)] \tag{4.3.8}$$

where $[y] = gr(X_{i|_{C^e}}) \cup \{x\}$ if y is true and, $[y] = gr(X_{i|_{C^e}})$ otherwise.

Based on this preliminary work, we now give different strategies for each case, i.e., embedding and shattering, in our decision framework in the next chapter.

5 Extending DPRMs with Dynamic Domain Strategies

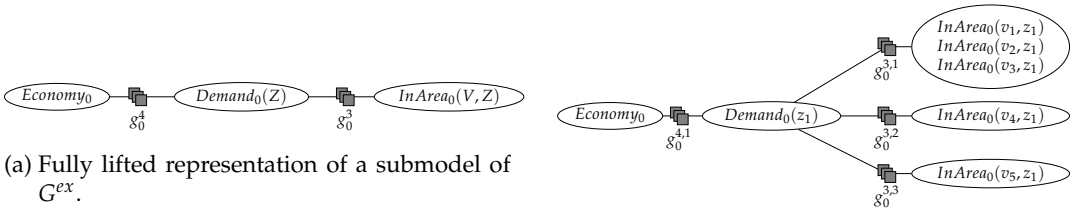
In this chapter, we extend semantics and model management in DPRMs to dynamic domains, focusing on providing solutions for adding new domain objects. We provide strategies for applying historical knowledge to newly introduced domain objects, thereby improving accuracy in answering queries related to these new entities. As there is no general solution, we present different strategies for the identified cases according to the decision framework presented in the previous chapter, and give practical considerations on how to choose the optimal strategy to maintain a lifted solution. The contents of this chapter are mainly published in the following conference paper.

> Nils Finke, Tanya Braun, Marcel Gehrke, and Ralf Möller. Dynamic Domain Sizes in Temporal Probabilistic Relational Models. *The International FLAIRS Conference Proceedings*, 34, 2021.

In **Section 5.1**, we provide an overview and motivation for the strategies that guide the incorporation of domain objects into an existing model. The focus of this section is on determining the similarities between a new domain object and those already encoded in the model. After making this distinction, we present three strategies for identifying potential functions suitable for a new domain object in **Section 5.2**. The proposed strategies aim to identify (existing) potential functions (and as such parfactors) that describe well the likelihood for the occurrence of events associated with the new domain object. Both sections represent contribution 2b of this work. We conclude this chapter in **Section 5.3** with an online lifted dynamic junction tree algorithm, including all strategies, as well as an empirical evaluation in **Section 5.4**.

5.1 Assessing Domain Object Similarity for Guiding Embedding and Shattering Decisions

In Section 4.2 we presented an informal decision framework that outlines the key considerations when adding a new domain object to an existing model domain, i.e., to a model domain of an instantiated DPRM that may already be shattered due to evidence. At the end



(a) Fully lifted representation of a submodel of G^{ex}.

(b) Shattered version of the submodel depicted in Fig. 5.1a.

Figure 5.1.: Illustration of a fully lifted model (left) and, analogously, a shattered variant of the same model (right), which we use throughout this chapter to explain strategies for domain expansion. **Note that in this and the following illustrations, we enumerate the randvars directly with the individual domain object combinations in the nodes for the PRVs and omit the constraint notation here for ease of reading.**

of each decision path within the presented decision framework, different strategies for the integration of a new object become necessary. With regard to the strategies presented, we pursue the following two main objectives.

1. **Preserving benefits of lifted inference**: Limiting additional groundings is key to maintaining a compact representation and keeping computational advantages in lifted inference. This typically involves incorporating new domain objects by reusing existing evidence parfactors, thereby incorporating them into pre-existing groups of domain objects associated with a common evidence parfactor within the model (by adjusting the parfactor constraint that limits a group of domain objects to one parfactor and a respective potential function).

2. **Ensure correct reasoning in relation to new domain objects**: It is critical that the potential function applied to the new domain object accurately represents the actual behaviour of that new object, i.e., the likelihood of events to occur for the new domain object (w.r.t. the variables associated with the potential function), thereby preventing losses in accuracy in inference when handling queries about the new domain object.

As a result, the decision on how to incorporate new domain objects involves a trade-off between accuracy and runtime in inference. In the previous chapter, we outlined the different decision paths, while in this chapter we give the specific strategies, at the end of each decision path, i.e., handling the different cases to incorporate new domain objects. We discuss the decisions and strategies along the path for $\times_2$, $\times_3$ and $\times_4$, and do not consider the trivial case $\times_1$ (projectivity) here. For the sake of simplicity, we will explain domain expansion strategies on the following simple submodel, which we use as a running example throughout this chapter.

Example 5.1.1 (A Running Example: Template Model and Shattered Model)**.** *Figure 5.1 depicts a submodel of G^{ex} for time step $t = 0$ with domain object $\mathcal{D}(V) = \{v_1, v_2, v_3, v_4, v_5\}$ and $\mathcal{D}(Z) = \{z_1\}$. While Fig. 5.1a depicts the submodel in a fully lifted state, Fig. 5.1b depicts the submodel for G_0 in a state after shattering due to evidence with the original parfactor g_0^3 replaced by three instances $g_0^{3,1}$, $g_0^{3,2}$ and $g_0^{3,3}$ with each parfactor limited by constraints to a different subset of the domain objects in $\mathcal{D}(V)$ and $\mathcal{D}(Z)$. Given G_0 in the shattered state, we elaborate on adding a domain object v_6 to the vessel domain $\mathcal{D}(V)$. To include v_6 in $\mathcal{D}(V)$, we identify the partitioned set of affected parfactors using the* PARTITIONEDAFFECTEDPARFACTORS *algorithm (see Algorithm 3), yielding*

$$G_0'' = \{\{g_0^{3,1}, g_0^{3,2}, g_0^{3,3}\}\}. \tag{5.1.1}$$

The algorithm returns all split (or evidence) parfactors that are affected by a domain change, with each subset in G_0'' indicating which parts of the model need to be adjusted. Independent of whether v_6 is observable or not, the two options presented below are applicable (see $\times_3$ and $\times_4$ in Fig. 4.5):

*(a) Either v_6 is **embedded** in G_0 such that one of the existing split parfactors in G_0'' is reused by means of adjusting the parfactors constraint, or,*

*(b) G_0 is further **shattered**, and therefore another split parfactor $g_0^{3,4}$ for the domain tuple (v_6, z_1) is added.*

Next, we give an overview of estimating and measuring similarities to existing domain objects and provide an overview of strategies for selecting partitions or performing new shatterings in this context. W.r.t. Example 5.1.1 this is assessing similarity between the new domain object v_6 and those domain objects behind respective parfactors $g_0^{3,1}$, $g_0^{3,2}$, and $g_0^{3,3}$.

Measuring Similarity

We first consider strategies that follow the decision path $\times_2 = \mathit{Yes}$, which implies that the domain object to be added is observable. Being observable means that we have knowledge of all the historical values that the associated randvars of the domain object might have taken, as well as the probabilities of each possible event occurring. This allows us to derive potential functions for that domain object corresponding to the parfactors in the model. Consequently, we can identify any existing evidence parfactors in the model by comparing the similarity of the new domain object's potential functions to those of domain objects already encoded in the model. Let's consider the following example to illustrate this.

Example 5.1.2 (Measuring Similarity)**.** *We continue with our running example in Example 5.1.1. The following evidence parfactors $G_0'' = \{\{g_0^{3,1}, g_0^{3,2}, g_0^{3,3}\}\}$ are affected by a change in $\mathcal{D}(V)$, here triggered by including a new vessel v_6. Each subset in G_0'' refers to a specific parfactor in the template model. More precisely, each subset includes all copies of the same parfactor, such as here*

*$\{g_0^{3,1}, g_0^{3,2}, g_0^{3,3}\}$ that relate to the original parfactor g_0^3 in the template model. Since the history of v_6 in terms of historical events for affected randvars (here $InArea_0(V,Z)$ and $Demand_0(Z)$) **is assumed to be known**, we can determine potential functions for v_6 for the respective parfactors involved, i.e., here only for g_0^3. Let*

$$\phi_0^{3'}(Demand(Z), InArea(V,Z))_{|\ C^{3'}} \quad with \quad C^{3'} = \big((V,Z), \{(v_6, z_1)\}\big) \tag{5.1.2}$$

be the potential function for the respective parfactor g_0^3. Note that $C^{3'}$ includes all object tuples for $\{v_6\} \times \mathcal{D}(Z)$, i.e., vessel v_6 fixed and all zones in $\mathcal{D}(Z)$. Note that, $\mathcal{D}(Z) = \{z_1\}$ is only one zone, and thus $C^{3'}$ only includes the single object tuple (v_6, z_1). Now $\phi_0^{3'}$ can be used by comparing $\phi_0^{3'}$ with the corresponding ϕs from $G_0{''}$, i.e. $\phi_0^{3,1}, \phi_0^{3,2}, \phi_0^{3,3}$, with a distance measure such as the KLD D_{KL}, to measure the similarity to the other domain objects, e.g., $D_{KL}(\phi_0^{3,1}, \phi_0^{3'})$. Then,

- *if the KLD D_{KL} is below a given threshold δ for one of the existing potential functions $\phi_0^{3,1}, \phi_0^{3,2}$, or, $\phi_0^{3,3}$ in of parfactors in $G_0{''}$, then the option of **embedding**, i.e., adjusting one of the existing constraints on the potential functions to include the new domain object accordingly, becomes applicable,*
- *provided that the KLD D_{KL} for none of the existing potential functions $\phi_0^{3,1}, \phi_0^{3,2}$, or, $\phi_0^{3,3}$ is below the threshold δ, then the option of **shattering**, i.e., introducing a new parfactor $g_0^{3,4}$ for v_6 with ϕ according to Eq. (5.1.2), becomes applicable.*

This analysis applies KLD to a single local factor within our relational model, primarily due to computational constraints and the efficiency of this approach in certain contexts. This focused application provides useful, though partial, insights into the compatibility of new domain objects with the existing model. However, it's important to recognise the limitations of not evaluating the full distribution, and the results should be seen as indicative rather than comprehensive. Further analysis, possibly covering the entire distribution, is recommended for a more complete understanding.

Before proceeding with this example, we give the alternative decision path in comparison.

Estimating Similarity

We discuss the strategies that follow the decision path $\times_2 = No$. This implies that the new domain object to be incorporated into an existing model domain is unobservable, which means that we can't derive potential functions based on the historical information about this domain object. Therefore, similarity to the parfactors for pre-existing domain objects can only be estimated. In these cases, similarity estimation is achieved through a heuristic. Despite the less precise identification of domain objects that potentially behave similarly, the same domain expansion strategies are applicable. Let's consider the following example.

Example 5.1.3 (Estimating Similarity)**.** *We continue with our running example in Example 5.1.1. The following evidence parfactors $G_0{''} = \{\{g_0^{3,1}, g_0^{3,2}, g_0^{3,3}\}\}$ are affected by a change in $\mathcal{D}(V)$, here*

*by including a new vessel v_6. Again, each subset in $G_0{}''$ relates to one parfactor in the template model, as here $\{g_0^{3,1}, g_0^{3,2}, g_0^{3,3}\}$ to g_0^3 . Since the history of v_6 in terms of historical events for affected randvars (here $InArea_0(V,Z)$ and $Demand_0(Z)$) **is not known**, we cannot determine potential functions for v_6 for the respective parfactors involved, i.e., here for g_0^3.*

- *For **embedding** v_6 into an existing parfactor, i.e., $g_0^{3,1}$, $g_0^{3,2}$, or $g_0^{3,3}$, we estimate the likelihood of the similarity between v_6 and the existing domain objects behind $g_0^{3,1}$, $g_0^{3,2}$, and, $g_0^{3,3}$. This could be done using the number of groundings $|gr(g_0^{3,1})|$, $|gr(g_0^{3,2})|$, and $|gr(g_0^{3,2})|$ as a heuristic. For example for*

$$g_0^{3,1} = \phi_0^{3,1}(Demand_0(Z), InArea_0(V,Z))_{|C^{3,1}} \tag{5.1.3}$$

  with $C^{3,1} = (\mathcal{X}, \{(v_1,z_1),(v_2,z_1),(v_3,z_1)\})$ this is $|gr(g_0^{3,1})| = 3$.

- *Alternatively, for example, if no sufficiently significant results can be determined via the heuristic, **shattering** and adding another split parfactor with a more generalised potential function for v_6 may be beneficial.In our example, $|gr(g_0^{3,2})|$ is 3 for $g_0^{3,1}$ and 1 for both $|gr(g_0^{3,2})|$ and $|gr(g_0^{3,3})|$. Embedding v_6 in $g_0^{3,1}$ may be a good choice considering a total $|\mathcal{D}(V)| = 5$. This may indicate that $g_0^{3,1}$ encodes a good generalised potential function describing object behaviour well.*

Identical strategies apply to both decision paths (measuring and estimating similarity). However, it's important to note that direct measurement of similarity to existing domain objects is not possible in the estimating-similarity-decision-path. Therefore, it is essential to rely on effective heuristic methods for similarity estimation. In this work, we adopt a simplified approach to estimating similarity with new domain objects, as demonstrated in the previous example.

Given that many manifestations in an evidence-rich model are described in terms of different potential functions detailing the different likelihoods for the occurrence of (dependent) events for different domain objects, it may be useful to choose just one of these potential functions, or alternatively to merge several of these potential functions. Exactly at this point, decisions have to be made about the trade-off between accuracy and liftability, for which we provide three different strategies in the next section.

5.2 Embedding and Shattering Strategies

We will first give an informal introduction to the strategies for both embedding and shattering. Subsequently, we examine each strategy individually, giving the corresponding formal definitions and presenting examples for both cases – similarity measurement and similarity estimation – one by one.

An Overview of the Embedding and Shattering Strategies

The strategies differ between embedding a new domain object into existing parfactors by adjusting its constraints and shattering the model and thus adding another parfactor.

In case the decision is made to **embed** a new domain object by **adjusting constraints of an existing parfactor**, only one strategy, which we summarise as follows, applies.

- The strategy is to adopt the so-called **Most Relevant Potential Function (MRPF)**, which is the potential function associated with an existing evidence parfactor that most closely aligns with the actual behaviour of the new domain object by accurately describing the likelihood of events occurring for that new domain object. The discovery of other domain objects with similarities to the new domain object, as represented by the potential functions in the evidence parfactors, is either measured via KLD or estimated using a heuristic over the number of groundings, as presented in Section 5.1. Finding and adopting the MRPF makes sense for two reasons.
  - As with new events the model dissolves from having one generalised potential function applicable to all domain objects into a shattered model having more specialised potential functions for subsets of the domain objects, i.e., the MRPF in this context reflects more accurate information by means of its potentials to encode the likelihood of future events as in the initial model state.
  - Furthermore, as no additional groundings are introduced we achieve our overall goal of not further shattering the model to keep the benefits of a lifted representation and in lifted reasoning.

  Nevertheless, on the other hand,
  - encoding the new domain objects with a more specialised potential function can lead to overfitting, where the model captures the noise in the data rather than the underlying patterns.

In case the decision is made to further **shatter** the model **by adding a new parfactor** for the new domain object, we propose two strategies.

- The first strategy is to adopt the so-called **Default Potential Function (DPF)**, i.e., the generalised potential function applied to all domain objects initially after the model construction process, i.e., in a fully lifted state. More precisely, this is the corresponding potential function as stored in the template model. Since the shattered model might no longer encode the DPF due to events for all already in the model encoded domain objects, the DPF is reintroduced by creating a new parfactor. We consider the DPF to be a generalised potential function since it describes the likelihood of the broad range of events to occur for all domain objects. This approach makes sense as
  - DPRMs model a stationary process, i.e., the full joint distribution of the model at a fixed time step is assumed to be still valid for all subsequent time steps as

the model is unrolled. Under this assumption, the DPF is a good generalised potential function that describes the behaviour of a new domain object well.

On the other hand, this assumption remains only valid

- if the encoded distributions in the model continue to accurately represent the behaviour of domain objects over time, which requires the implementation of concept drift detection in any temporal model.
- Furthermore, due to shattering benefits of lifted reasoning is counteracted.

- The second strategy is to adopt a so-called **Weighted Default Potential Function (WDPF)**, which involves weighting the potentials in the DPF by potentials of other potential functions that closely align with the actual behaviour of the new domain object by accurately describing the likelihood of events occurring for that new domain object. The similarity is again either measured via the KLD or estimated depending on the available information, i.e., observability, for the new domain object. This approach
  - combines both positive effects of the DPF and the MRPF and relaxes the assumption that new objects behave similarly to existing objects, and in addition it allows for behaviour (likelihood of events occurring) defined by the DPF, as well as all behaviour defined by the MRPF for many other existing domain objects in the model.

  However, on the other hand,
  - the model is further shattered, and therefore benefits of lifted reasoning are counteracted.

Figure 5.2 illustrates the applicable strategies for each path along the decision framework. From these three types of potential functions, namely the MRPF, the DPF, and the WDPF result analogously, the respective strategies, which we define formally as follows.

Across all strategies, given a local model G_t and some new domain object l, parfactor partitions with parfactors affected by that domain change are determined as per Algorithm 3, yielding parfactor partitions

$$\begin{aligned} G_t'' &= \{\{g_t^{1,1}, \ldots, g_t^{1,k}\}, \ldots, \{g_t^{i,1}, \ldots, g_t^{i,k}\}\}, \\ G_t'' &=: \{G_t^1, \ldots, G_t^i\} \end{aligned} \tag{5.2.1}$$

as a basis for the subsequent strategies to work.

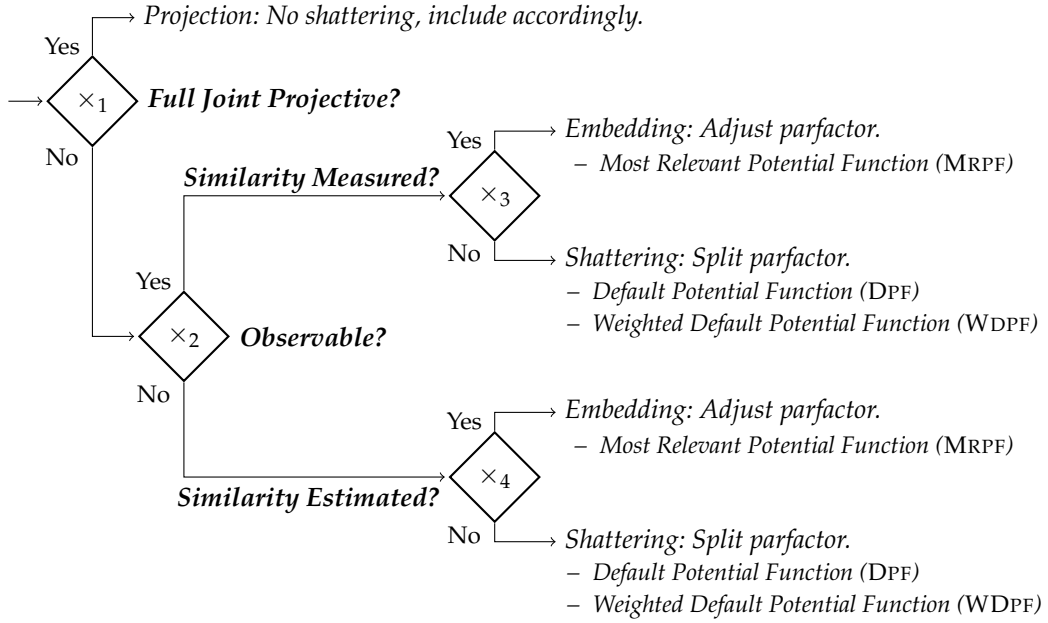


Figure 5.2.: Illustration of the decision path on how to add a new domain object to an existing domain including respective domain expansion strategies.

The MOST RELEVANT POTENTIAL FUNCTION (MRPF) Strategy

In the MRPF strategy, parfactors g of a local model G_t are determined for some new domain object l into which, by adjusting the constraints of g, the object l is included, provided that g satisfies the requirements of similarity. For each partition $G_t^i \in G_t''$ in Eq. (5.2.1), the MRPF corresponds to one of the potential function ϕ of one of the parfactors $g \in G_t^i$. Given some similarity function Θ and a similarity threshold $\delta \in \mathbb{N}^+$, that g is identified such that g fulfils

$$\min_{g \in G_t^i}(\Theta) \leq \delta. \tag{5.2.2}$$

Depending on whether the new domain object under consideration is (was) fully observable (Decision $\times_2$), i.e., whether historical information (observations) w.r.t. the variables of the model are available, the similarity to other domain objects already existing in the respective model domain is **estimated** or **measured**. As such the similarity function Θ, differs depending on the above case.

- For $\times_2 = \textit{Yes}$, we take KLD for Θ as the similarity measure. The DKL $D_{KL}(\phi, \phi')$ is calculated between the potential function ϕ of some parfactor g and a potential function $\phi'(\mathcal{A}')$ determined for some new domain object l (for instance, derived from given

historical event data). I.e., for a new domain object a potential function is determined that has the same definition as $g = \phi(\mathcal{A})$ with $\mathcal{A}' = \mathcal{A}$, but not necessarily $\phi' = \phi$.

- For $\times_2 = No$, Θ is equivalent to $(|gr(g)|)^{-1}$, i.e., the reciprocal number of groundings $|gr(g)|$ for a parfactor g as a heuristic to asses similarity.

Both similarity functions return a value close to 0 denoting the highest similarity. As a final step, for each parfactor partition $G_t^i \in G_t{}'$, the constraint of the parfactor $g \in G_t^i$ that encodes the MRPF according to Eq. (5.2.2) is modified. This is including the new domain object l in g by using the INCLUSION operator on g, which modifies the constraint of g accordingly. We give the following two examples.

Example 5.2.1 (Most Relevant Potential Function: Measure Similarity)**.** *We continue with Example 5.1.2. To include v_6 in a local model G_0, first, the vessel domain is adjusted, i.e., $\mathcal{D}(V) = \mathcal{D}(V) \cup \{v_6\}$. Given a potential function $\phi_0^{3\prime}$ for the new domain object v_6 (determined based on given historical information from the outside), and by a domain change affected parfactors $G_0{}'' = \{\{g_0^{3,1}, g_0^{3,2}, g_0^{3,3}\}\}$, we measure similarity to potential functions $\phi_0^{3,1}, \phi_0^{3,2}, \phi_0^{3,3}$ by using the KLD accordingly. The potential function $\phi_0^{3,1}, \phi_0^{3,2}$, or, $\phi_0^{3,3}$ that has the smallest KLD D_{KL} and is below the threshold δ provided the from the outside and therefore has the highest similarity to $\phi_0^{3\prime}$, is considered the* MRPF*. Without specifying a particular potential function for the respective parfactors under consideration in this example, we assume at this point that $\phi_0^{3,1}$ has the smallest KLD D_{KL} to $\phi_0^{3\prime}$. Further, D_{KL} is below a provided similarity threshold δ. By running the* INCLUSION *operator on $g_0^{3,1}$ with v_6, the constraint of $g_0^{3,1}$ is adjusted to include v_6. Since $G_0{}''$ only contains one partition, just one parfactor needs to be adjusted to handle the domain change accordingly.*

For similarity estimation, we give the following example.

Example 5.2.2 (Most Relevant Potential Function: Estimate Similarity)**.** *We continue with Example 5.1.3. First, to include v_6 in the model, the vessel domain is adjusted, i.e., $\mathcal{D}(V) = \mathcal{D}(V) \cup \{v_6\}$. No historical knowledge is available for the new domain object v_6, so no potential functions can be derived for v_6 that can be used for similarity assessment. Therefore,* MRPF *corresponds to that parfactor g in the by a domain change affected parfactor partitions $G_0{}'' = \{\{g_0^{3,1}, g_0^{3,2}, g_0^{3,3}\}\}$, which is here only one, i.e., $G^1 = \{g_0^{3,1}, g_0^{3,2}, g_0^{3,3}\}$, such that $\min_{g \in G^1}\left((|gr(g)|)^{-1}\right)$ is below the provided threshold δ. This is here $g_0^{3,1}$ with $\min_{g^i \in G^i}\left((|gr(g^i)_C|)^{-1}\right) = 3^{-1}$. By running the* INCLUSION *operator on $g_0^{3,1}$ with v_6, the constraint on parfactor $g_0^{3,1}$ is adjusted to include v_6.*

If no MRPF is identified, it holds that there may not be a parfactor with a sufficiently high similarity available. In such cases, shattering the model by applying one of the following strategies remains an option.

The DEFAULT POTENTIAL FUNCTION (DPF) Strategy

As part of the application of the DPF strategy, a new parfactor for some new domain object l

is added to the model. This is adding a new parfactor $g_t^{i,k+1}$ for each partition $G_t^1 \in G_t{}'$ as per Eq. (5.2.1), such that after the application of the DPF strategy $G_t{}'$ is as follows

$$\begin{aligned} G_t{}' &= \{\{g_t^{1,1}, \ldots, g_t^{1,k}, g_t^{1,k+1}\}, \ldots, \{g_t^{i,1}, \ldots, g_t^{i,k}, g_t^{i,k+1}\}\}, \\ G_t{}' &=: \{G_t^1, \ldots, G_t^i\}. \end{aligned} \tag{5.2.3}$$

Each new parfactor $g_t^{i,k+1}$ that is added to G_t is created basis

- the potential function as per the corresponding template parfactor $\tilde{g}_t^i = \tilde{\phi}_t^i(\mathcal{A})$ of the template model $\mathcal{G}_t$, such that
- the new parfactor

$$g_t^{i,k+1} = \tilde{\phi}_0^{i,k+1}(\mathcal{A})_{|\ C^{i,k+1}} \tag{5.2.4}$$

  is assigned the potential function $\tilde{\phi}$ as given by the template parfactor $\tilde{g}_t^i$, and
- the constraint $C^{i,k+1}$ is created by running the INCLUSION operator on $g_t^{i,k+1}$ with l.

We give the following example.

Example 5.2.3 (Default Potential Function)**.** *In both Example 5.1.2 and Example 5.1.3, the* DPF *is similarly defined as no historical knowledge about the new domain object v_6 is needed. Again, first, v_6 is included in the vessel domain, i.e., $\mathcal{D}(V) = \mathcal{D}(V) \cup \{v_6\}$. Further, a corresponding* DPF *is determined basis the a corresponding parfactor of the template model for each partition of the $G_0{}''$, i.e., here for the partition $\{g_0^{3,1}, g_0^{3,2}, g_0^{3,3}\}$ the* DPF *is $\phi_0^{3,4}$ which is applied to v_6 by adding a new split parfactor*

$$g_0^{3,4} = \phi_0^{3,4}(Demand(Z), InArea(V,Z))_{\big|_{C^{3,4}}} \quad with \quad C^{3,4} = \big((V,Z), \{(v_6, z_1)\}\big). \tag{5.2.5}$$

The object tuple (v_6, z_1) is generated using the INCLUSION *operator.*

Next, we consider the WDPF strategy, by which similarly to the DPF strategy a new custom parfactor for some new domain object is created.

The WEIGHTED DEFAULT POTENTIAL FUNCTION (WDPF) Strategy

The WDPF strategy, is a combination of both the DPF strategy and the MRPF strategy. However, instead of adjusting one existing parfactor as in the MRPF strategy, the MRPF is used and combined with the DPF by weighing. For each partition $G_t^i \in G_t{}'$

- one parfactor $\hat{g} \in G_t^i$ with the parfactor's potential function $\hat{\phi}$ denoting the MRPF, and
- one template parfactor $\tilde{g} \in \mathcal{G}_t$ with the parfactors potential function $\tilde{\phi}$ denoting the DPF is identified.

Then the WDPF results from $\tilde{\phi} + \hat{\phi}$ summing up potentials of the matching range values of both MRPF and DPF, followed by adding a new parfactor for the WDPF to be encoded in the model as already described in Eq. (5.2.3) and Eq. (5.2.4).

Note that when summing up potential functions, those potential functions with significantly higher potential values may become overly dominant in the resulting WDPF. This could lead to an imbalance where potential functions, e.g. the DPF, with lower potential values have little or negligible influence on the result, causing an imbalance in the overall behaviour of the model. In such cases it is necessary to normalise or downscale the dominant potential function to ensure a more balanced model, which we do not consider here.

Example 5.2.4 (Weighted Default Potential Function)**.** *We continue with Example 5.1.2. Similar to the other strategies, to include v_6 in the model, first the vessel domain is adjusted, i.e., $\mathcal{D}(V) = \mathcal{D}(V) \cup \{v_6\}$. Since G_0'' has only a single partition $\{g_0^{3,1}, g_0^{3,2}, g_0^{3,3}\}$, just one parfactor*

$$g_0^{3,4} = \phi_0^{3,4}(Demand(Z), InArea(V,Z))_{\big| C^{3,4}} \quad with \quad C^{3,4} = \big((V,Z), \{(v_6, z_1)\}\big) \tag{5.2.6}$$

to encode a potential function for v_6 needs to be introduced. The object tuple (v_6, z_1) of the constraint $C^{3,4}$ is generated using the INCLUSION *operator. To generate the* WDPF*, we pre-assign $\phi_0^{3,4}$ in $g_0^{3,4}$ with the* DPF *derived from the template model. Then, given a potential function ϕ_0^3 for v_6, we identify the corresponding* MRPF *in $\{g_0^{3,1}, g_0^{3,2}, g_0^{3,3}\}$, which we assume to be $g_0^{3,1}$ in this example. Given the parfactor for the* MRPF*, i.e.,*

$$g_0^{3,1} = \phi_0^{3,1}(Demand(Z), InArea(V,Z))_{\big| C^{3,1}} \quad with \quad C^{3,1} = \big((V,Z), \{(v_1, z_1), (v_2, z_1), (v_3, z_1)\}\big) \tag{5.2.7}$$

we use potentials of $\phi_0^{3,1}$ and $\phi_0^{3,4}$ as new potentials for the WDPF*. Weighting is done by considering the groundings of parfactor $g_0^{3,1}$. More specifically, as $g_0^{3,1}$ is limited to domain object tuples (v_1, z_1), (v_2, z_1), and (v_3, z_1), and $g_0^{3,4}$ to the domain object tuple (v_6, z_1), $\phi_0^{3,4}$ for the* WDPF *results*

$$\begin{aligned} \phi_0^{3,4} = {} & \phi_0^{3,4}((Demand(z_1), InArea(v_6, z_1)) + \phi_0^{3,1}((Demand(z_1), InArea(v_1, z_1)) \\ & + \phi_0^{3,1}((Demand(z_1), InArea(v_2, z_1)) + \phi_0^{3,1}((Demand(z_1), InArea(v_3, z_1)) \end{aligned} \tag{5.2.8}$$

Let the tables below depict parfactor tables. The table on the left depicts $g_0^{3,4}$ before adjusting potentials, on the right depicts $g_0^{3,1}$, and in the middle depicts $g_0^{3,4}$ after adjusting potentials.

The resulting WDPF *as represented in Example 5.2.4 is then set in $g_0^{3,4}$.*

Since the WDPF is calculated using the MRPF, the determination of the WDPF also differs depending on whether the MRPF can be determined by similarity measurement or by similarity estimation. We will omit a further example of how to determine the WRDF in the case where the dependent MRDF can be determined by similarity estimation. The previ-

$Demand_0(Z)$	$InArea_0(V,Z)$	$g_0^{3,4}$
high	*true*	10
medium	*true*	2
low	*true*	2
high	*false*	2
medium	*false*	2
low	*false*	2

(a) Split Parfactor $g_0^{3,4}$ before weighting.

$Demand_0(Z)$	$InArea_0(V,Z)$	$g_0^{3,1}$
high	*true*	1
medium	*true*	1
low	*true*	1
high	*false*	5
medium	*false*	1
low	*false*	1

(b) Split Parfactor $g_0^{3,1}$ with $\phi_0^{3,1}$ as MRPF.

$Demand_0(Z)$	$InArea_0(V,Z)$	$g_0^{3,4}$
high	*true*	13
medium	*true*	5
low	*true*	5
high	*false*	17
medium	*false*	5
low	*false*	5

(c) Split Parfactor $g_0^{3,4}$ after weighting with $\phi_0^{3,4}$ as WDPF.

ous example applies here by analogy, with the difference that the MRDF is determined as presented in Example 5.2.2.

Which of the three strategies presented should be applied depends on the specific use case. In general, it is advisable to minimise shattering and adapt existing parfactors to incorporate new domain objects into an existing model. However, this approach is only effective if the new domain object has a significantly strong similarity to the other domain objects of that parfactor, whose potential function is restricted to a particular subset of domain objects. If this condition is not met, the new domain object is likely to break up over time as more events occur, due to the inability of the potential function to accurately represent its behaviour. In the next section, we combine the presented strategies and adapt online LDJT with dynamic domain support.

5.3 Online Lifted Dynamic Junction Tree Algorithm with Dynamic Domains

We extend the online version of the LDJT algorithm to enable new domain objects to be incorporated at runtime in an existing model domain based on the MRPF, DPF, and WDPF strategies proposed before. Algorithm 4 outlines the overall algorithm. In addition to the input parameters of the online version of LDJT, the algorithm receives a domain object stream $\mathcal{O}$, which we define as follows.

Definition 5.3.1 (Object Stream)**.** Given a universe $\mathbf{D}$ of objects and a DPRM $(G_0, G_{\rightarrow})$ consisting of, among others, $\mathbf{L}$ a set of logvar names, and $\mathbf{D}$ a set of entities with each logvar L having an active domain $\mathcal{D}(L) \subseteq \mathbf{D}$, then an object stream $\mathcal{O}$ is defined as a function

$\mathcal{O} : T \to \mathbf{D}_{\subseteq}$, that maps each time step $t \in 0, \ldots, T$ to a set of domain objects $\mathbf{D}_t \subseteq \mathbf{D}$ with

$$\forall d \in \mathbf{D}_t : d \notin \bigcup_{L \in \mathbf{L}} \mathcal{D}(L) \tag{5.3.1}$$

for domain objects d and $\mathbf{D} \subseteq \mathbf{D}$.

The domain object stream includes new domain objects, i.e., domain objects that do not yet exist in any of the model's domains but need to be included in one of them. Note that the object stream $\mathcal{O}$ involves objects that are provided from the outside, i.e., we consider domain object changes as directly observable and not subject to uncertainty. While moving forward in time $t = 0, 1, \ldots$ a local FO jtree J_t and a local model G_t is created and the outgoing message of the previous time step $\alpha_t - 1$ is set within J_t. Before setting the evidence E_t from the evidence stream $\mathcal{E}$ and thus continuing the algorithm as introduced in Algorithm 2, dynamic domain handling is performed.

Based on this extension of the LDJT algorithm, we perform an empirical evaluation to assess the performance of the three presented strategies in the context of our example application from shipping.

5.4 Empirical Evaluation

Whilst in the previous sections we gave examples of the addition of a new vessel, in this evaluation we look at adding a new zone due to the availability of real-world data. We first describe the setup of the DPRM for the evaluation including the evaluation approach, followed by summarising the results. We evaluate the effectiveness of each proposed strategy in successfully integrating a new domain object into the model while transferring existing knowledge encoded in the model to the new domain object. Our goal is to achieve good prediction accuracy by effectively exploiting the existing knowledge for the new domain object. Conversely, poor prediction results are due to the inability of the presented approaches to effectively incorporate the new domain object into the model. Specifically, we examine the impact of three factors on the accuracy of each strategy, including

(a) the amount of additional information required to determine similarity with other domain objects,

(a) the importance of recent knowledge compared to

(a) less recent knowledge about the new domain object, i.e., historical events that facilitate the induction of potential functions for the domain object used in the similarity measurement.

For the sake of simplicity, for this case, we only look at the submodel around the parfactor

Algorithm 4 Online LDJT algorithm with dynamic domain support

Input: DPRM $(G_0, G_{\to})$, Evidence stream $\mathcal{E}$, Query stream $\mathcal{Q}$, Object stream $\mathcal{O}$
Construct FO jtrees $J_0, J_{\to}$ from $(G_0, G_{\to})$
for $t = 0, 1, ...$ **do**
  Instantiate FO jtree J_t from $J_0, J_{\to}$ and local model G_t from $(G_0, G_{\to})$
  Add α_{t-1} message
  while $\mathcal{O}$ contains objects d at t to add **do**
    Include domain object d into its respective domain $\mathcal{D}$
    $G_t'' \leftarrow$ PARTITIONEDAFFECTEDPARFACTORS(G_t, d) ▷ See Algorithm 3
    for each partition $G_t^i = \{g_t^{i,1}, \ldots, g_t^{i,k}\}$ in G_t'' **do**
      ▷ Depending on the choice of the respective strategy
      if Embedding **then**
        $g_t^{i,k} \leftarrow$ Identify parfactor with MRPF in G_t^i
        $g_t^{i,k} \leftarrow$ Perform INCLUSION for d in $g_t^{i,k}$ ▷ See Definition 4.3.1
      else if Shattering **then**
        $g_t^{i,k+1} \leftarrow$ Create split parfactor with DPF or WDPF and perform INCLUSION for d
        $G_t \leftarrow G_t \cup \{g_t^{i,k+1}\}$
  Add evidence E_t from evidence stream $\mathcal{E}$
  Pass messages in J_t
  Answer queries Q_t from query stream $\mathcal{Q}$ on J_t
  Calculate α_t message

g_t^1 of the overall DPRM. To recap again, the submodel

$$g_t^1 = \phi_t^1(Idle_t(Z), Rate_t(Z), Supply_t(Z)) \quad \text{and} \quad g^S = \phi^S(Supply_t(Z), Supply_{t+1}(Z)) \tag{5.4.1}$$

describes the interaction between idle times, freight rates and supply in zones across the globe over time. To set up the DPRM, we use historical vessel movements from 2020 based on automatic identification system (AIS) data[1] provided by the Danish Maritime Authority for the Baltic Sea. The aim of providing AIS data is to improve the safety and guidance of vessel traffic by exchanging navigational and other vessel data. It was introduced as a mandatory standard by the International Maritime Organisation (IMO) in 2000. Meanwhile, AIS data is used in many different applications in research, such as trade flow estimation, emission accounting, and vessel performance monitoring (Yang *et al.*, 2019). Preprocessing for retrieving variables *Supply* and *Idle* for 367 defined *Zones* can be found on GitHub[2].

[1] https://dma.dk/safety-at-sea/navigational-information/ais-data
[2] https://github.com/FinkeNils/Processed-AIS-Data-Baltic-Sea-2020-v2

All steps to derive the necessary data for the model based on the AIS dataset are described in the Appendix A.1. Using the data provided, which consists of idle times and supply in zones during all calendar weeks in 2020, the evaluation process is defined as follows:

1. We randomly choose one zone z out of the 367 zones in the AIS dataset and remove its related data from the dataset.

2. Then, all data from the dataset is used to determine a potential function for idle and supply events. Potentials are derived by counting the number of occurrences of events, i.e., all combinations of range values for $Idle_t(Z)$ and $Supply_t(Z)$ such as $\phi_t^1(Idle_t(Z) = high, Supply_t(Z) = high)$ or $\phi_t^1(Idle_t(Z) = high, Supply_t(Z) = medium)$ for all $t \in \{1, \ldots, 51\}$ and any zone in $\mathcal{D}(Z) \setminus \{z\}$. Likewise, we determine the potential function for the interslice parfactor g^S.

3. The resulting potential function ϕ_t^1 and ϕ^S are then set in the DPRM $(G_0, G_{\rightarrow})$ for g_t^1 and g^S, i.e.,

$$\begin{aligned} g_t^1 &= \phi_t^1(Idle_t(Z), Supply_t(Z))_{\big|\left((Z), C_Z\right)} \\ g^S &= \phi^S(Supply_t(Z), Supply_{t+1}(Z))_{\big|\left((Z), C_Z\right)} \end{aligned} \tag{5.4.2}$$

   with $C_Z = \{z_1, \ldots, z_n\} \setminus \{z\}$.

4. Then, we split the dataset into two parts at a randomly chosen time step $t \in \{1, \ldots, 51\}$: The first part of the dataset consists of all data until calendar week t, and the second part of all remaining data.

5. Given the initialised DPRM $(G_0, G_{\rightarrow})$ and LDJT-online, we roll the out the DPRM and push events available in the first part of the dataset on the evidence stream $\mathcal{E}$ such that we are left with a shattered model G_t and the respective time step t (end of the data in the first part of the dataset).

Subsequently, the local model G_t is used as a basis for testing the presented strategies MRPF, DPF and WDPF, i.e., adding the domain object z, that we initially removed from the data, into the domain $\mathcal{D}(Z)$ and the shattered model G_t. To test each strategy, we answer a prediction query for $P(Supply_{t+1}(z)|Idle_{t+1}(z) = i)$ and build up a confusion matrix to store the actual, i.e., correct, supply value s_{true} (as per the dataset) and the predicted supply s^* value at $t+1$ for the previously excluded zone z given the specific idle condition i, i.e, $Idle_{t+1}(z) = i$, with i provided as per the dataset. Here, the predicted supply s^* value corresponds to the value with the highest probability resulting from computing $P(Supply_{t+1}(z)|\ Idle_{t+1}(z) = i)$. More specifically, the predicted supply value s^* for

$P(Supply_{t+1}(z)|\ Idle_{t+1}(z) = i)$ is defined as

$$s^* = \underset{s \in \mathcal{R}(Supply_{t+1}(z))}{\operatorname{argmax}} P(Supply_{t+1}(z) = s \big| Idle_{t+1}(z) = i) \tag{5.4.3}$$

given the idle condition i. Then, the accuracy is calculated as the ratio of correct predictions, i.e., the number of experiments where s^* matches s_{true}, to the total number of experiments performed.

In this matter, we perform 108 experiments in total and compare all three strategies. In each experiment, we test prediction accuracy for a different zone z that got removed from the dataset basis a shattered model G_t with varying time steps t. Since strategies MRPF and WDPF require further external knowledge to determine objects with similar behaviour, we provide events for varying intervals $[t-i, t]$ given in the dataset for z to determine potential functions for z used for similarity measurement. In these experiments, the end of the interval $[t-i, t]$ corresponds to a randomly chosen time t, which also serves as the starting point for the evaluation based on the shattered model G_t. With varying experiments, the interval is expanded by increasing i. Consequently, as the interval grows, more knowledge about z becomes available, enabling a more accurate assessment of the similarity with other domain objects. Table 5.1 represents accuracy scores per strategy. The highest accuracy score per experiment is highlighted in bold. The first column in Table 5.1 represents the time interval within which we provide historical evidence for z. In addition, strategies two and three require a similarity threshold δ provided from the outside. We test similarity thresholds of $\delta \in \{0.05, 0.1, 0.15\}$, which can be read off in the columns of Table 5.1. Further, columns MRPF, DPF and WDPF correspond to results for the respective strategies.

Note that the DPF strategy is shown in the table below under the combined column header for δ and the combined row header indicating the historical time interval $[t-i, t]$ from which events were aggregated to derive potential functions for the new domain object for use in the MRPF and WDPF strategies. This is for clarity and comparison only as the DPF strategy does not require a similarity threshold or additional evidence to work.

The results in Table 5.1 reveal that with smaller δ, the accuracy of all approaches is slightly higher. This makes sense as in order to identify both the WDPF and the MRPF as part of the similarity assessment for a new domain object, all domain objects already encoded in the model must be benchmarked via the respective evidence parfactor potential functions. The lower the threshold δ is, the higher the similarity between the new domain object and the existing domain objects must be, and in turn the higher the accuracy in inference should be. Further, the results reveal that accuracy is in most cases higher using the WDPF strategy, however, accuracy results are only slightly better compared to results from MRPF. Higher accuracies with the WDPF strategy can be explained by the fact that the WDPF results by combining the DPF and the MRPF into a new potential function. As such the WDPF combines both the generalised potential functions of the template model and the specific poten-

tial functions that describe the relevant behaviour for other domain objects at the point in time under consideration. Nevertheless, the losses in accuracy might be acceptable considering that under application of the MRPF the model is not further shattered, i.e., keeping runtime benefits of lifted inference.

Looking at the number of historical data provided for a new object to be added (a), there is no clear difference between having lots of knowledge over the new domain object to assess similarity compared to having only little knowledge available. However, it becomes clear that observations in the recent past (c) are most relevant for finding such domain objects that behave similarly to the new object. In general, the best accuracies of all experiments are in the range of 0.35 to 0.43, which are good results considering the number of prediction classes, here 5. Results in the range of 0.2 would be random results assuming a uniform distribution. Besides accuracy measures, further metrics such as precision, recall and F_1-score can be found in Appendix A.2.

	$\delta = 0.05$			$\delta = 0.1$			$\delta = 0.15$		
$[t-i, t]$	DPF	WDPF	MRPF	DPF	WDPF	MRPF	DPF	WDPF	MRPF
[5, 8]	**0.370**	0.326	0.240	**0.400**	0.273	0.284	0.310	0.280	**0.311**
[5, 9]	**0.280**	0.210	0.245	0.250	0.340	**0.369**	0.340	0.310	**0.406**
[5, 10]	0.280	**0.416**	0.340	0.290	**0.325**	0.320	0.260	0.240	**0.273**
[5, 11]	0.270	**0.354**	0.280	**0.280**	0.244	0.200	**0.300**	0.292	0.290
[5, 12]	0.260	**0.290**	0.270	0.300	0.210	**0.301**	0.310	**0.379**	0.350
[5, 13]	0.280	**0.313**	0.280	**0.340**	0.265	0.190	**0.310**	0.246	0.210
[5, 14]	0.250	0.210	**0.253**	0.250	**0.280**	0.298	0.200	0.270	**0.305**
[6, 8]	0.280	**0.356**	0.350	0.350	0.350	**0.420**	0.360	0.350	**0.447**
[6, 9]	0.270	**0.347**	0.310	0.330	**0.319**	0.230	0.320	0.260	**0.329**
[6, 10]	**0.250**	0.140	0.140	0.220	0.260	**0.265**	0.240	0.390	**0.460**
[6, 11]	0.250	**0.367**	0.270	0.300	**0.392**	0.310	**0.310**	0.270	0.252
[6, 12]	**0.330**	0.235	0.160	0.300	0.340	**0.410**	0.290	0.310	**0.345**
[6, 13]	0.240	**0.318**	0.240	0.270	**0.308**	0.250	0.240	**0.484**	0.390
[6, 14]	0.290	**0.354**	0.260	**0.350**	0.293	0.270	0.200	**0.262**	0.230
[7, 8]	**0.340**	0.290	0.310	0.310	**0.324**	0.250	**0.390**	0.383	0.350
[7, 9]	**0.380**	0.260	0.342	0.320	**0.306**	0.280	**0.340**	0.280	0.302
[7, 10]	0.280	0.310	**0.334**	0.200	**0.322**	0.250	**0.380**	0.305	0.220
[7, 11]	0.280	**0.345**	0.290	0.340	0.200	**0.249**	0.210	**0.334**	0.250
[7, 12]	**0.300**	0.310	0.230	**0.340**	0.251	0.250	0.250	**0.314**	0.290
[7, 13]	0.290	0.280	**0.345**	0.260	0.280	**0.361**	0.290	**0.376**	0.290
[7, 14]	0.180	**0.416**	0.330	0.310	**0.351**	0.310	0.290	**0.334**	0.250
[8, 9]	**0.360**	0.310	0.339	0.290	**0.238**	0.210	0.380	**0.400**	0.340
[8, 10]	**0.350**	0.270	0.349	0.280	0.330	**0.397**	**0.440**	0.335	0.260
[8, 11]	0.270	**0.308**	0.250	0.280	**0.285**	0.280	0.270	**0.407**	0.320
[8, 12]	**0.380**	0.310	0.349	0.310	0.260	**0.359**	**0.420**	0.150	0.195
[8, 13]	0.230	0.270	**0.327**	**0.390**	0.355	0.340	0.290	**0.357**	0.300
[8, 14]	0.260	**0.332**	0.300	0.240	**0.275**	0.260	**0.310**	0.280	0.291
[9, 10]	0.270	0.360	**0.398**	0.360	**0.381**	0.300	**0.370**	0.271	0.260
[9, 11]	**0.430**	0.376	0.300	**0.400**	0.351	0.300	**0.380**	0.261	0.260
[9, 12]	0.260	**0.400**	0.340	**0.380**	0.336	0.240	**0.370**	0.230	0.312
[9, 13]	0.240	0.210	**0.262**	0.190	0.340	**0.418**	**0.280**	0.265	0.240
[9, 14]	0.230	**0.323**	0.300	**0.340**	0.264	0.240	**0.330**	0.301	0.270
[10, 11]	0.300	**0.422**	0.390	0.390	**0.450**	0.360	0.320	**0.437**	0.350
[10, 12]	**0.350**	0.280	0.323	0.360	0.360	**0.456**	0.300	**0.342**	0.250
[10, 13]	0.300	**0.332**	0.240	**0.360**	0.316	0.220	0.260	0.340	**0.420**
[10, 14]	**0.310**	0.284	0.200	0.330	**0.353**	0.310	0.230	**0.345**	0.330

Table 5.1.: Accuracy scores for different strategies in LDJT with Dynamic Domain Support

6 Part I: Interim Conclusion

In this part, we have extended DPRMs and LDJT with dynamic domain sizes. In general, we distinguish between the three cases *Addition*, *Removal* and *Restoration*, where the biggest challenge lies in the first case, i.e. the addition of domain objects, which we have focused on in this part. In the context of adding new domain objects, the primary challenge is to find the parfactors within the shattered model that encode potential functions that accurately describe the likelihood of the occurrence of events that are associated with the new domain object and the respective randvars linked by the potential function, i.e., those that closely match the behaviour of the newly introduced domain object. In doing so, we pursue two goals. On the one hand, we avoid splits in the model so that a lifted representation is maintained as much as possible. On the other hand, we achieve that knowledge already encoded in the model is transferred to the new entity in order to increase the accuracy of the model.

Since there is no universal strategy for handling the introduction of new domain objects, we introduce three different approaches, each focusing on a specific type of potential function:

- the MOST RELEVANT POTENTIAL FUNCTION (MRPF),
- the DEFAULT POTENTIAL FUNCTION (DPF), and,
- the WEIGHTED DEFAULT POTENTIAL FUNCTION (WDPF).

These potential functions are identified when a new domain object is added, with the aim to identify a potential function accurately encoding the potential, i.e., the likelihood, of events associated with the new domain object and effectively capturing its behaviour in the model. While the MRPF aims to identify an existing parfactor in which the new domain object can be incorporated by adjusting its constraints, thus avoiding further shattering. With the DPF and WDPF approaches a new split parfactor is introduced counteracting the overall goal. The MRPF approach is only effective if the new domain object has a significantly strong similarity to the other domain objects of that parfactor, whose potential function is restricted to a particular subset of domain objects. If this condition is not met, the new domain object is prone to be shattered out over time as more events occur. This shattering results in the creation of a new split parfactor whose potential function is constrained to the new object. This is due to the potential function not accurately representing the behaviour of the new

domain object, making the DPF and WDPF approaches more appropriate in such cases.

The strategies differ in the kind of (not) existing knowledge for the new domain object to be added. If additional knowledge is available about the new domain object, this information can be used to measure its similarity to other domain objects already present in the model. The comparison is performed by examining potential functions of parfactors that are restricted to groups of domain objects that exhibit similar behaviour. These groups represent sets of domain objects in which the new domain object can potentially be included, ensuring a valid representation of their common characteristics. If no additional knowledge is available about the new domain object, still the same strategies are applicable, however identifying other domain objects by similarity measurement is not possible. In such cases, the similarity has to be estimated by different characteristics. In cases where similarity needs to be estimated using different characteristics, in this work we use a straightforward approach to determine the MRPF by analysing the number of domain objects a potential function is constrained to. The underlying assumption is that if a potential function remains a suitable representative for a large set of domain objects, then the probability of it being applicable to a new domain object is also higher.

Finally, we have presented LDJT with dynamic domain support combining the proposed strategies. In an empirical evaluation of our reference example from dry shipping, we have compared the accuracy of predictions for unknown domain objects using the presented strategies. We show that the WDPF strategy achieves the best prediction results. This demonstrates that when adding a new domain object, it is useful to use both historical model knowledge and current knowledge about a new domain object in deriving appropriate potential functions. Nevertheless, the WDPF strategy achieves only slightly better results than the MRPF strategy. The use of the MRPF strategy is consistent with the primary objective of avoiding further model shattering. Therefore, this strategy is preferable as the lifted representation retains all its runtime advantages given a small loss of accuracy is tolerable.

A crucial factor for the effectiveness of the proposed strategies is the identification of similar domain objects. In the presented strategies, we determine the similarity between domain objects based on *their* potential functions. However, this approach has a limitation as potential functions aggregate historical information into a single time slice. That is, we can determine symmetry in the current time step, but not symmetry with respect to behaviour resulting from events across multiple time steps. To overcome this drawback, additional methods for capturing and evaluating temporal patterns and relationships between domain objects can be integrated to potentially increase the strategies' effectiveness in detecting similarities over time. In the next part of this work, we present such a temporal approach to approximate symmetries. However, we do not link the approach proposed in the next part to improve the strategies presented in this part of the thesis and leave this step for future work. Another topic that is left for future work is belief revision. In cases where the integration of a new domain object using one of the proposed strategies turns out to

be incorrect, belief revision can play a crucial role in refining the model, i.e., to adapt and improve its representation of domain objects and their relationships as more data becomes available. This would allow for better handling of incorrect initial assumptions and lead to improved overall performance in predicting and representing the relationships between domain objects over time.

Part II

Constructing Symmetries

In the previous part of this work, we extended DPRMs and LDJT to accommodate dynamic domains. A major challenge is to integrate new domain objects into the model and to transfer the knowledge implicitly encoded in the model's parfactors, more specifically in the parfactor's potential functions, to the new domain object. To achieve this, it is essential to identify other domain objects that exhibit symmetries to the new domain object. We proposed an algorithm that assesses similarity via the potential functions at the time of adding the new domain object to the model. This approach has a notable drawback: potential functions accumulate events collected over time. In other words, a potential function at the current time implicitly encodes historical information, but not the sequence of events that led to the potential function at this time step. The sequence of historical events, or more precisely, the (real-valued) observation of a randvar given as part of the event, for individual domain objects can be interpreted as the behaviour of those domain objects. When identifying domain objects with symmetries to each other, domain object behaviour should be considered and not just the aggregated events that lead to a particular potential function at a particular time step. In a DPRM, as detailed in the preliminaries in Section 3.4, (real-valued) random variables observed over time are treated as time series representing the behaviour of a specific domain object. More specifically, DPRMs describe the overall behaviour of a system at a macro level, while time series analysis provides methods for examining the micro-level behaviour of individual components, such as domain objects, within the DPRM. Understanding symmetries in behaviour between domain objects is not only relevant for extending DPRMs and LDJT to account for dynamic domains, but can also be used to avoid unnecessary groundings and to improve predictive accuracy in inference. This is the focus of this part of the work.

The part of this work is structured as follows. In Chapter 7, we propose an approach that runs as part of the initial model construction process and is used to determine groups of objects with similar behaviour, yielding *symmetry clusters* as a result. In Chapter 8, we propose an approach that runs during performing inference, exploiting symmetry clusters to avoid unnecessary groundings and improve prediction accuracy.

7 Multivariate Ordinal Patterns for Temporal Symmetry Approximation

In this chapter, we introduce an approach that determines groups of objects with similar behaviour, so-called *symmetry clusters*. In **Section 7.1**, we first discuss the benefits of constructing symmetry clusters in more detail. This motivates the introduction of our contributions 3a–c, and, constitutes the remainder of this part. In **Section 7.2** we then give an overview of how to represent some important aspects of a time series using ordinal pattern symbolisation to obtain a simplified representation while retaining essential characteristics. The chosen representation of the time series then simplifies the subsequent task of identifying clusters of symmetric time series. In **Section 7.3**, we finally introduce *multivariate ordinal patterns for temporal symmetry approximation*, MOP_4SA for short, an approach for determining symmetry clusters.

The content of this chapter is mainly published in the following three conference papers. The first paper introduces temporal symmetry approximation for the univariate case, i.e., determining object symmetries on a PRV level. The second paper extends symmetry approximation to the multivariate case, i.e., determining object symmetries on a parfactor level according to the factorisation of the model. The third paper demonstrates the expressiveness of ordinal patterns in the context of subsequent learning tasks such as clustering.

Nils Finke and Marisa Mohr. A Priori Approximation of Symmetries in Probabilistic Dynamic Relational Models. In Stefan Edelkamp, Ralf Möller, and Elmar Rueckert (Eds.), *KI 2021: Advances in Artificial Intelligence*, pages 309–323. Springer, 2021.

Nils Finke, Ralf Möller, and Marisa Mohr. Multivariate Ordinal Patterns for Symmetry Approximation in Dynamic Probabilistic Relational Models. In Guodong Long, Xinghuo Yu, and Sen Wang (Eds.), *AI 2021: Advances in Artificial Intelligence*, pages 543–555. Springer, 2022.

Nils Finke, Marisa Mohr, Alexander Lontke, Marwin Zünfle, Samuel Kounev, and Ralf Möller. Recommendations for Data-Driven Degradation Estimation with Case Studies from Manufacturing and Dry-Bulk Shipping. In Samira Cherfi, Anna Perini, and Selmin Nurcan (Eds.), *Research Challenges in Information Science - 15th International Conference, RCIS 2021*, volume 415 of *Lecture Notes in Business Information Processing*, pages

189–204. Springer, 2021.

Note that the ratio of authorship of Nils Finke and Marisa Mohr is equal for the second and third papers. The results of the papers are summarised in the following journal article, with additional applications from dry-bulk shipping.

Nils Finke and Ralf Möller. On the Approximation of Symmetries and Related Structural Changes in Dynamic Probabilistic Relational Models. *Advances in Science, Technology and Engineering Systems Journal*, 7(2), 2022.

7.1 On the Construction of Symmetries

DPRMs incorporate temporal aspects, allowing to effectively encode and represent information as it evolves over time, making DPRMs useful in any *online scenario*, i.e., in scenarios where new knowledge is encoded on the fly to enable temporal query answering. In existing research it is assumed that lifted graphical models already embed symmetries, i.e., simply speaking, a model is set up so that all objects are assigned the same generalised distribution describing the likelihood of events to occur w.r.t. to the model's variables. New knowledge is then incorporated into the model with new events, i.e., observations, for each object. Observations are encoded within the model as evidence parfactors, i.e., a new split parfactor is created for the observed domain object with an adjusted potential function that accounts for the event, releasing the object from a previously symmetrical representation. If the same observation is made for multiple objects, those objects are split off together and continue to be treated as a group. Over time, models dissolve into groups of symmetrically behaving objects, i.e., symmetries are implicitly exploited. Note that in the worst case, the models are split in such a way that all objects are treated individually, i.e., no symmetries are available in the model so that lifted inference can no longer be applied and all its advantages disappear.

Still, no one has yet focused on constructing symmetries in advance instead of deriving symmetries implicitly. Constructing symmetries in advance has benefits in applications and becomes necessary due to the characteristics of real-world applications:

(i) Certain information about model objects may not be available at a particular point in time but only become known downstream. In such cases, it is beneficial to infer information according to the open-world assumption from the behaviour of other objects which tend to behave similarly, i.e., applying an intrinsic default in the form of an inferred event that is similarly applied to the unobserved domain object. On the one hand, wrong information can be introduced in the model, but on the other, it is likely that objects continue to behave the same, i.e., according to the behaviour of their symmetrical objects.

(ii) Symmetrical objects can behave the same in the long term but may deviate for shorter

periods of time. Even minor deviations lead to groundings in the model, which, if prevented, introduce a small error in the model, which is negligible in the long term.

In both cases, a model grounds, which, if prevented, helps keeping reasoning in polynomial time w.r.t. the maximum domain size. Therefore, we construct groups of objects with similar behaviour, which we denote as *symmetry clusters*. The idea is to apply a symbolisation scheme to approximate temporal symmetries, i.e., objects that tend to behave the same over time. Using the symmetry clusters, it is possible to selectively prevent groundings, which helps retaining a lifted representation. At the same time, using symmetry clusters also helps increasing the prediction accuracy in inference by applying an intrinsic default for objects for which no or only a small amount of observations are available.

Since DPRMs integrate temporal aspects allowing for the encoding of events as they appear over time, it is about understanding symmetrical behaviour instead of, in the static case, only deriving symmetries based on a specific state of the model. Behaviour results from events w.r.t. to randvars of the model observed for a specific domain object over time. Depending on the number of interdependent randvars in the model, events for a domain object over time either generate a univariate or multivariate time series, as described in Section 3.4. In the following section, we review methods from the field of time series analysis to determine symmetries in time series.

7.2 Symmetry Approximation in Time Series

In time series analysis, the notion of similarity, known as symmetry in DPRMs, has often been discussed in the literature (Agrawal *et al.*, 1993; Keogh *et al.*, 2001; Kramer, 2020). In general, approaches for finding similarities in a set of time series are either (a) value-based, or (b) symbol-based. *Value-based* approaches compare the observed values of time series. By comparing the value of each point x_t, $t = 1, \ldots, T$ in a time series X with the values of each other point $y_{t'}$, $t' = 1, \ldots, T'$ in another time series Y (warping), with value-based approaches shifts and frequencies can be included. Accordingly, two popular algorithms evolved, namely dynamic time warping (DTW) (Kruskal and Liberman, 1983) and matrix profiling (Yeh *et al.*, 2016), which measure similarity between two time series by aligning them in a non-linear way, allowing the time axis to be stretched or compressed. As DPRMs can encode interdependencies between multiple variables, respective multivariate procedures should be used to assess similarities. The first dependent multivariate dynamic time warping (DMDTW) approach is reported by Petitjean *et al.* (2012), in which the authors treat a multivariate time series with all its m interdependencies as a whole. The flexibility of warping in value-based approaches leads to a high computational effort and is therefore unusable for large amounts of data. Although there are several extensions to improve runtime (Salvador and Chan, 2004) by limiting the warping path or reducing the number of data points, e.g., FastDTW (Salvador and Chan, 2004) or PrunedDTW (Silva and Batista,

2016), the use of dimensionality reduction is inevitable in the context of DPRMs.

For dimensionality reduction, *symbol-based* approaches encode the time series observations as sequences of symbolic abstractions that match the shape or structure of the time series. More specifically, the continuous or discrete values of the time series are converted into a series of symbols that capture the essential patterns or trends within the data. These symbols can represent different features of the time series, such as the overall trend (increasing, decreasing or constant), local peaks and valleys, or specific patterns such as periodic cycles. Since PRVs in DPRMs encode discrete range values, depending on the degree of discretisation, symbol-based approaches are preferred as they allow for discretisation directly. As far as research is concerned, there are two general ways of symbolisation. On the one hand, *classical symbolisation* partitions the data range according to specified mapping rules to encode a numerical time series into a sequence of discrete symbols. A corresponding and well-known algorithm is Symbolic Aggregate ApproXimation (SAX) introduced by Chiu *et al.* (2003). On the other hand, as introduced by Bandt and Pompe (2002) *ordinal pattern symbolisation* encodes the total order between two or more neighbours ($x < y$ or $x > y$) into so-called ordinal symbols ($(0, 1)$ or $(1, 0)$). Mohr *et al.* (2020b) extend univariate ordinal patterns to the multivariate case, taking into account not only the dependencies of neighbouring values over time but also the dependencies between variables in a time series.

Specifically, here, an ordinal approach has notable advantages in applications:

(i) The method is conceptionally simple,

(ii) the ordinal approach supports robust and fast implementations (Keller *et al.*, 2017; Piek *et al.*, 2019), and

(iii) compared to classical symbolisation approaches such as SAX, it allows for an easier estimation of a good symbolisation scheme (Keller *et al.*, 2015; Stolz and Keller, 2017).

The concept of entropies is intuitive, simple, and not particularly complex, especially compared to other well-known feature representations such as statistical features or principal components. In Finke *et al.* (2021c), we provided recommendations for combining several methods, e.g., from frequency analysis and feature extraction, to predict the remaining useful life of a system (in this case, bearings and vessels). Overall, in the experiments we have demonstrated that entropies prove to be a suitable representation for classical regression models. Moreover, we show that entropy features are almost as effective as other, more advanced features such as internal representations computed by Convolutional Neural Networks (CNNs) from the field of deep learning. Especially when working with high-dimensional data, as in our use case, a simple and easy-to-compute solution is key to the effectiveness of the approach. More specifically, depending on the number of dependent PRVs and the number of domain objects in a model, complex multivariate time series data emerge. Therefore, in the following chapter, we introduce ordinal pattern symbolisation and

explain our approach to classify similar entity behaviour to construct symmetries.

7.3 Examining Objects with Similar Symbolisation Scheme (MOP$_4$SA)

First, we incorporate the symbolisation approach as presented in the preliminaries in Section 3.5, with adjustments to account for the data range, followed by introducing a clustering method based on the symbolisation scheme to effectively identify symmetry clusters, altogether making up MOP$_4$SA. We anticipate that MOP$_4$SA, by considering its underlying ordinal approach, will efficiently be able to analyse the high-dimensional time series data, i.e., sequences of events related to random variables and specific domain objects, to then be able to cluster symmetric time series. Our approach aims to efficiently identify symmetric time series and consequently symmetric behaviour between domain objects. The steps of MOP$_4$SA can be summarised as follows:

1. Performing ordinal pattern symbolisation with data range dependence, followed by
2. creating a similarity graph using the determined ordinal patterns, and,
3. basis the similarity graph do clustering to determine symmetry clusters.

Each paragraph of this section highlights one of the steps of MOP$_4$SA, which we compile into one algorithm at the end of this section.

Ordinal Pattern Symbolisation with Data Range Dependence

To encode the behaviour of a time series, we use ordinal pattern symbolisation based on the work of Bandt and Pompe (2002), extended by Mohr *et al.* (2020b), who themselves extended univariate ordinal patterns to the multivariate case. To determine symmetries, we use historical events for domain objects with respect to the randvars of the model. The sequences of events can then be treated as multivariate time series, which we explain using the following example.

Example 7.3.1 (DPRM and Time-Series)**.** *Let us consider a DPRM with PRVs related to subchallenge 1 as introduced in Chapter 2, i.e., with PRVs* $Idle_t(Z)$*,* $Rate_t(Z)$*, and* $Supply_t(Z)$ *connected by parfactor* g_t^1 *at a time step* t *and the PRV* $Supply_t(Z)$ *connected via an interslice parfactor* g^S *to the PRV* $Supply_{t+1}(Z)$ *at a subsequent time step* $t+1$*. Each PRV is parameterised with*

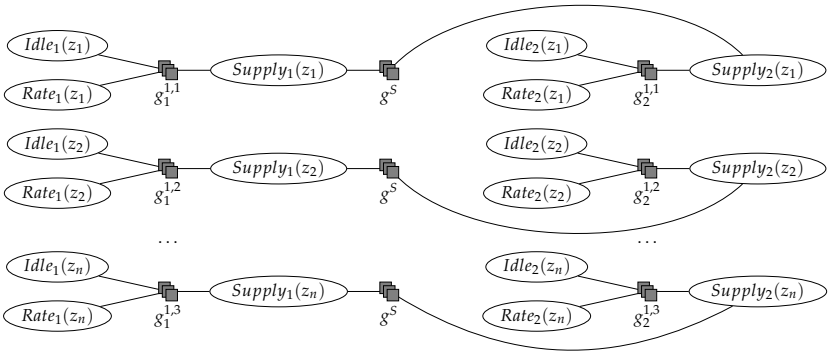


Figure 7.1.: Grounded submodel of G^{ex} for PRVs related to sub-challenge one.

logvar Z with a domain $\mathcal{D}(Z)$ of size $n = |\mathcal{D}(Z)|$. With events, e.g.,

$$\begin{aligned}\mathbf{E}_{1:T} = \{&Idle_1(z_1) = high, Rate_1(z_1) = medium, Supply_1(z_1) = high,\\ &Idle_1(z_2) = high, Rate_1(z_2) = high, Supply_1(z_2) = high,\\ &\dots\\ &Idle_1(z_n) = high, Rate_1(z_n) = low, Supply_1(z_n) = medium,\\ &Idle_2(z_1) = medium, Rate_2(z_1) = high, Supply_2(z_1) = high,\\ &Idle_2(z_2) = medium, Rate_2(z_2) = high, Supply_2(z_2) = high,\\ &\dots\\ &Idle_2(z_n) = low, Rate_2(z_n) = high, Supply_2(z_n) = medium, \dots\}\end{aligned} \tag{7.3.1}$$

for each domain object, i.e, zones $z_1, z_2, \dots, z_n$, for each of the PRVs for time steps $t = 1, \dots, T$, the fully lifted model is shattered, i.e., grounded. Figure 7.1 depicts the corresponding ground model after events for $\mathbf{E}_{1:2}$, i.e., for time steps $t = 1$ and $t = 2$, are encoded as evidence parfactors. The sequence of observations (events) over time for individual domain objects with respect to the randvars in the model, where the randvars are in turn linked in a common parfactor provided that they are dependent on each other, can then be considered as a multivariate time series. More precisely, if the events in $\mathbf{E}_{1:T}$ are arranged in a sequence over time t for each domain object and a given parfactor, then the following structure emerges as indicated in the below table. Each row represents events over time t related to PRVs linked in the parfactor g_t^1, with each row considered a multivariate time series with its values for variables as columns. E.g., for zone z_1 the sequence of events for randvars $Idle_t(z_1)$, $Rate_t(z_1)$, and $Supply_t(z_1)$ over time are considered a multivariate time series.

Note that in this example we have called the discretised values for the multivariate time series, whereas in the following we will refer to the nominal values of the multivariate time series for the determination of the symmetry clusters.

$\mathcal{D}(Z)$	g_t^1	$t=1$	$t=2$	...	$t=T$
z_1	$Idle_t(z_1)$	*high*	*medium*	...	
	$Rate_t(z_1)$	*medium*	*low*	...	
	$Supply_t(z_1)$	*high*	*high*	...	
z_2	$Idle_t(z_2)$	*high*	*medium*	...	
	$Rate_t(z_2)$	*high*	*high*	...	
	$Supply_t(z_2)$	*high*	*high*	...	
		$\vdots$			
z_n	$Idle_t(z_n)$	*high*	*low*	...	
	$Rate_t(z_n)$	*low*	*high*	...	
	$Supply_t(z_n)$	*medium*	*medium*	...	

A DPRM exploits (conditional) in-dependencies between randvars by encoding interdependencies in parfactors. As such, parfactors describe interdependent data through its linked PRVs, e.g., as in the previous example the correlation between supply $Supply_t(Z)$, idle times $Idle_t(Z)$ and freight rates $Rate_t(Z)$ within a common zone Z encoded by the parfactor g_t^1. For each object $z_i \in \mathcal{D}(Z)$ from the PRVs $P = \{Supply_t(Z), Idle_t(Z)$ and $Rate_t(Z)\}$ in g_t^1 observations are made over time, i.e., a time series

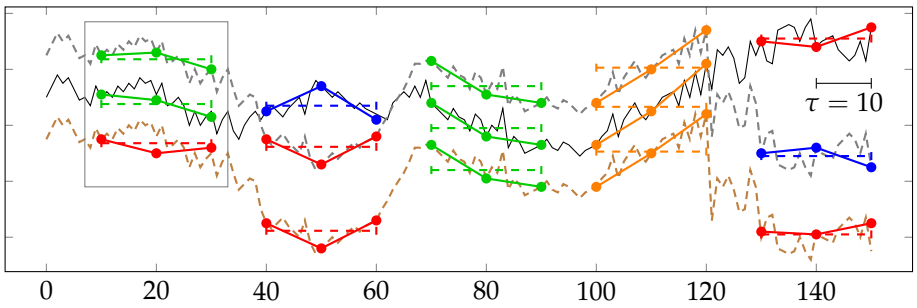
$$((x_t^i)_{i=1}^m)_{t=1}^T \tag{7.3.2}$$

with events $x_t^i \in \mathcal{R}(P^i)$ with $P^i \in P$ and $m = 3$ as the number of PRVs linked in g_t^1 is generated. In this work, we assume that the time series is multivariate and has variables that are dependent on each other, i.e., $m > 1$. Note that in Finke and Mohr (2021) we consider the case $m = 1$. Having $|\mathcal{D}(Z)|$ objects in Z, we consider $n = |\mathcal{D}(Z)|$ groundings, i.e., n multivariate time series

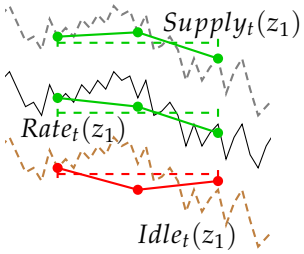
$$\mathcal{X} = (((x_t^i)_{i=1}^m)_{t=1}^T)_{j=1}^n \quad \in \mathbb{R}^{m \times T \times n}, \tag{7.3.3}$$

e.g., for $m = 3$ with observations $(x_t^1, x_t^2, x_t^3) = (Supply_t(z_j), Idle_t(z_j), Rate_t(z_j))$ for every $z_j \in \mathcal{D}(Z)$ at time $t \in \{1, \ldots, T\}$. As such, a multivariate time series is defined for several PRVs linked in a single parfactor, while a univariate time series is defined for a single PRV. One example for the data structure $\mathcal{X}$, is represented in the form of a table, or more specifically matrix, in the preceding example. Note that a PRV can be parameterised with more than one logvar, but for the sake of simplicity, we introduce our approach using PRVs with only one logvar throughout this work. Symmetry detection for m-logvar PRVs works similarly to one-logvar PRVs, with the difference that in symmetry detection, object tuples, i.e., m-tuples, are used. As an example, for any 2-logvar PRV $P_t(X, Y)$, an object tuple is a 2-tuple (x_1, y_1) with $x_1 \in \mathcal{D}(X)$ and $y_1 \in \mathcal{D}(Y)$, such that the size of $\mathcal{X}$ increases as $n = |\mathcal{D}(X) \times \mathcal{D}(Y)|$ increases.

The identification of symmetrical object behaviour is done on sets of (multivariate) time



(a) Ordinal pattern determination of order $d = 3$ and delay $\tau = 10$ of a multivariate time series with three variables, i.e., $m = 3$. The solid coloured lines represent the ordinals, while the dashed and coloured lines represent the mean data range of a specific ordinal. The same ordinals have the same colour. Ordinals for each variable at the same time step make up the multivariate ordinal pattern as illustrated by the gray box for the first pattern.



$$\bar{x}_{10}^{3,10} = \frac{1}{3}\Big(\frac{1}{3}\left(Supply_{10}(z_1) + Supply_{20}(z_1) + Supply_{30}(z_1)\right) + \frac{1}{3}\left(Rate_{10}(z_1) + Rate_{20}(z_1) + Rate_{30}(z_1)\right) + \frac{1}{3}\left(Idle_{10}(z_1) + Idle_{20}(z_1) + Idle_{30}(z_1)\right)\Big)$$

(b) Example calculation for the arithmetic mean for one multivariate ordinal pattern at $t = 10$ with $d = 3$ and $\tau = 10$. Here for the first pattern as in Fig. 7.2a.

Figure 7.2.: Examples for multivariate ordinal patterns on a multivariate time series with three variables for one domain object and accordingly the calculation of the arithmetic mean of one pattern.

series, i.e., across different (multivariate) time series that are observed for every object individually. Figure 7.2a represents a multivariate time series with three variables such as in our running example with PRVs $Idle_t(Z)$, $Rate_t(Z)$, and $Supply_t(Z)$ for one domain object, e.g., here $z \in \mathcal{D}(Z)$. Each line in the figure corresponds to observations for one variable. Note that for the sake of better illustration, the real numerical values of individual variables are used in this graph instead of the discrete values, i.e., the range values of the PRVs. However, to create the DPRM, the continuous numerical values would be converted to discrete values using a discretization approach of choice. Ignore the coloured patterns for now. Then, given a (multivariate) time series $\mathcal{X}$, for each time step $t = \tau(d-1)+1, \ldots, T$ of the multivariate time series

$$((x_t^i)_{i=1}^m)_{t=1}^T) \in \mathcal{X}, \tag{7.3.4}$$

MOP is determined as introduced by Mohr *et al.* (2020b) and described in the preliminaries in Section 3.5. Here x_t^i corresponds to the value of a variable i at time t. With reference to the previous example, this would be either $Idle_t(z)$, $Rate_t(z)$, or $Supply_t(z)$ for a domain object $z \in \mathcal{D}(Z)$. MOP is determined at given order d and delay τ, yielding a symbolised time series. Each ordinal at a time step t is determined by taking d real values skipping intermediate values corresponding to the next τ time steps of each time series i in the multivariate time series with m variables. In Fig. 7.2a some examples of ordinal patterns for order $d = 3$ and $\tau = 10$ are illustrated as coloured solid lines with circle markers corresponding to the d values taken to determine the ordinal. The greyed box around the first ordinal in each time series of the overall multivariate time series illustrates the corresponding MOP.

Ordinal patterns are well suited to capture the general dynamics of time series, mainly due to their ability to operate independently of the specific range of numerical values. This makes them applicable regardless of whether the values of the variables range from high to low. However, in certain applications, the specific data range (the exact actual value of a variable) may still be relevant after symbolisation, although this information is typically lost in the process of ordinal pattern symbolisation. One example is the green-coloured ordinal pattern for time step $10 \geq t < 30$ highlighted in the grey box in Fig. 7.2a. Even though the ordinals are the same for the variable supply and rate (green ordinals) due to the same up-and-down movements in data, the underyling data tuples differ considering their specific value on the y-axis. For the time series represented by the dashed black line, the real values, i.e., the intercept on the y-axis, for the ordinal in green are larger than for the time series represented by the solid black line. In other words, transforming a sequence

$$x = (x_t^i)_{a \leq t \leq b} \tag{7.3.5}$$

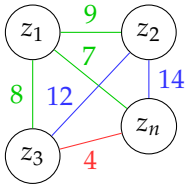
as $y = x + c$, where $c \in \mathbb{R}$ is a constant, should change y's similarity to other sequences, although the shape is the same. To measure the similarity between time series and consequently determine domain objects with similar behaviour, the real-valued level of the time series, i.e. the data range, is relevant. Therefore, in addition to each ordinal, we determine the arithmetic mean

$$\overline{x}_t^{d,\tau} = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{d} \sum_{k=1}^{d} x_{i,t-(k-1)\tau} \tag{7.3.6}$$

of the multivariate time series' values corresponding to the ordinal pattern, where $x_{i,t-(k-1)\tau}$ is min-max normalised, as an additional characteristic or feature of behaviour. More specifically, the arithmetic mean $\overline{x}_t^{d,\tau}$ is determined for each ordinal by averaging the values of the ordinals, i.e., $\frac{1}{d} \sum_{k=1}^{d} x_{i,t-(k-1)\tau}$, and further taking the average over all m arithmetic means that are determined for each ordinal of each time series (variable) in the overall multivariate time series. One example for the calculation of $\overline{x}_t^{d,\tau}$ is given in Fig. 7.2b. We use the

	z_1	z_2	z_3	...	z_n
z_1	0	9	8	...	7
z_2	9	0	12	...	14
z_3	8	12	0	...	4
...	...	...	...	...	...
z_n	7	14	3	...	0

(a) Similarity matrix of object pairs.



(b) Corresponding Similarity Graph to similarity matrix in Fig. 7.3a.

Figure 7.3.: (a) Similarity matrix and (b) similarity graph. The similarity matrix stores similarity counts between all object pairs. The higher the count, the more similar the two domain objects are. The similarity graph visually depicts nodes that are closer to each other given the similarity count as an edge weight.

arithmetic mean as a simple feature besides ordinals themselves for similarity clustering. For similarity clustering, (a) the ordinals contribute to identifying similar behaviour (i.e., upward and downward movements), and, (b) the arithmetic mean gives an understanding of the similarity of time series in terms of the range value of each variable. There are other features than the arithmetic mean that may be suitable for measuring similarity in terms of the data range. However, in this work, we focus on the approach itself and leave improvements, e.g. by choosing other features, as future work. After determining MOP and the corresponding arithmetic mean, we yield a new data representation

$$\mathcal{X}' = \langle o, \overline{x} \rangle^{(T-(\tau(d-1))\times|\mathcal{D}(Z)|} \tag{7.3.7}$$

where $\langle o, \cdot \rangle_{tj} \in \mathcal{X}'$ represents the MOP and $\langle \cdot, \overline{x} \rangle_{tj} \in \mathcal{X}'$ represents the corresponding mean $\overline{x}_t^{d,\tau}$ for one domain object at time step t. We refer to $\mathcal{X}'$ as the symbolisation scheme of a multivariate time series. The order d and delay τ are passed in from the outside and might depend on, e.g., the frequency of the data, to capture the long-term behaviour of each object.

After encoding the behaviour of objects by ordinal pattern symbolisation, similar objects are identified using clustering. For this purpose, and based on the derived symbolisation representation in Eq. (7.3.7), a similarity graph indicating the similarities based on a distance measure between object pairs is created.

Creating a Similarity Graph

Object similarity is determined per parfactor, i.e., per multivariate time series, separately. To measure similarity between domain objects, we use the symbolic representation $\mathcal{X}'$, which uses tuples of multivariate ordinal numbers and mean values to describe the behaviour of an object. The similarity of two objects z_i and z_j is given by counts w_{ij} of equal behaviours,

i.e.,

$$w_{ij} = \sum_{t \leq T} \Big[\langle o, \cdot \rangle_{it} = \langle o, \cdot \rangle_{jt} \ \wedge \ |\langle \cdot, \overline{x} \rangle_{it} - \langle \cdot, \overline{x} \rangle_{jt}| < \delta \Big], \tag{7.3.8}$$

where $[x] = 1$ if x and, 0 otherwise with δ provided as a threshold from the outside. As an auxiliary structure to store similarity counts, we use a square matrix

$$\mathcal{W} \in \mathbb{N}^{|\mathcal{D}(Z)| \times |\mathcal{D}(Z)|}, \tag{7.3.9}$$

where each $w_{ij} \in \mathcal{W}$ describes the similarity between objects z_i and z_j by simple counts of equal behaviour over time $t \in T$. Simply put, one counts the time steps t at which both multivariate time series of z_i and z_j have the same MOP and the absolute difference of the mean values of the corresponding MOPs is less than $\delta > 0$. Finally, as represented in Figure 7.3b, the counts w_{ij} correspond to the weights of the edges in the similarity graph $\mathcal{W}$, where zero indicates no similarity between two objects, while the larger the count, the more similar two objects are. The weights associated with the edges of the similarity graph represent the similarity between two nodes, or more precisely, how closely related two objects are.

Example 7.3.2 (Similarity Counting)**.** *Given two symbolised multidimensional time series for two different domain objects z_1 and z_2, as partially illustrated in Fig. 7.2 for one domain object, let's say z_1, the similarity is measured by comparing MOPs in each time step. This is for time step $t = 1$ comparing $\langle o, \overline{x} \rangle_{11}$ for z_1 with $\langle o, \overline{x} \rangle_{12}$ for z_2. If ordinals o in $\langle o, \overline{x} \rangle_{11}$ and $\langle o, \overline{x} \rangle_{12}$ are the same and $\overline{x}_{11} - \overline{x}_{12} < \delta$ is below a provided threshold δ, then in the similarity matrix the count at position z_1, z_2 is increased by one. Figure 7.3a depicts an example representation of a similarity matrix.*

Approximating symmetries based on the similarity graph leaves us with a classical clustering problem, where the objective is to group objects exhibiting sufficient similarities while leaving distinct objects independent.

Clustering Objects with Similar Symbolisation Scheme

Theoretically, any clustering algorithm can be applied on top of the similarity graph. Each weight in the similarity matrix, or each edge weight between an object pair, denotes the similarity between two objects, i.e., the higher the count, the more similar the two objects are to each other. As the number of clusters, i.e., groups of domain objects with similar behaviour, is unknown, we perform density-based clustering, specifically density-based spatial clustering of applications with noise (DBSCAN) for use in multivariate ordinal patter for symmetry approximation (MOP$_4$SA). DBSCAN clusters data points, here the nodes, based on their neighbourhood and density. It identifies clusters by discovering areas with a high concentration of data points while separating areas with lower point densities as noise. Simply put, looking at the graph representation of the similarity matrix as depicted in Fig. 7.3b, it be-

Algorithm 5 Multivariate ordinal pattern for symmetry approximation (MOP$_4$SA)

procedure MOP$_4$SA(DPRM $(G_0, G_{\rightarrow})$, Evidence $E_{0:T}$, Order d, Delay τ, Threshold δ)**:**
  $\mathcal{S} \leftarrow \emptyset$
  for $g_t^i \in (G_0, G_{\rightarrow})$ **do**
    $X^{|\mathcal{A}| \times T} \leftarrow$ get all evidence from $\mathbf{E}_{0:T}$ concerning PRVs $\mathcal{A}$ in $g_t^i = \phi_t^i(\mathcal{A})_{|C^i}$
    $\mathcal{X}^{|\mathcal{A}| \times (T - \tau(d-1))} \leftarrow$ symbolic representation matrix initialised with zeros
    for every dimension $i = 1, \ldots, |\mathcal{A}|$ of $\mathcal{X}$ **do**
      $\mathcal{X}_{i.} \leftarrow$ create a time series of tuples with $\langle ordinal, mean \rangle$
    $\mathcal{W}^{n \times n} \leftarrow$ similarity-matrix initialised with zeros of size $n = |gr(g_t^i)|$
    for every time step t of $\mathcal{X}_{.t}$ **do**
      $w_{ij} \leftarrow$ do similarity counting given δ
    $\mathcal{S}_{|g_t^i} \leftarrow$ Perform clustering based on the similarity-matrix $\mathcal{W}^{n \times n}$
    $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{S}_{|g_t^i}$
  return $\mathcal{S}$

comes apparent that the nodes z_1, z_2, and z_n are in a neighbourhood due to the higher edge weights, i.e., greater similarity, while z_3 is located further off. Therefore, domain objects z_1, z_2, and z_n might end up in one symmetry cluster and z_3 in another symmetry cluster. After clustering is done, we are left with multiple clusters of objects for each parfactor, which we denote as symmetry clusters. More specifically, for each parfactor that represents an independent context of the overall model, multiple clusters of objects with similar behaviour are determined. We formalise symmetry clusters as follows.,

Definition 7.3.1 (Symmetry Cluster)**.** For a parfactor $g_t^i \in G_t$ with G_t being the PRM at time step t and $g_t^i = \phi_t^i(\mathcal{A})_{|C^i}$ a parfactor with a sequence of PRVs $\mathcal{A} = (A^1, \ldots, A^n)$, a symmetry cluster S^i is a set of individual objects $l \in \mathcal{D}(L)$ with l in one of the domains $\mathcal{D}(L)$ of one of the logvars $L \in \mathbf{L}$ with $\mathbf{L} = \bigcup_{i=1}^{n} lv(A^i)$. Let the term $en(S)$ refer to the set of objects/entities in any symmetry cluster S. Each parfactor $g_t^i \in G_t$ can comprise up to m symmetry clusters $\mathcal{S}_{|g_t^i} = \{S^i\}_{i=0}^{m}$, such that $en(S^i) \cap en(S^j) = \emptyset$ for $i \neq j$ and $i, j \in \{1, \ldots, m\}$.

Algorithm 5 presents the corresponding pseudocode for the symbolisation and clustering steps presented in this chapter. The algorithm expects to get a fresh DPRM $(G_0, G_{\rightarrow})$, historical evidence $E_{0:T}$, and hyper-parameters order d, delay τ and threshold δ as inputs and gives symmetry clusters as an output used for further tasks such as preventing groundings. For every parfactor g_t^i in the fresh model $(G_0, G_{\rightarrow})$, i.e., a model which is not yet split due to any evidence, symmetry clusters $S_{|g_t^i}$ are determined based on historical evidence $E_{0:T}$. First, ordinal pattern symbolisation is performed by extracting observations for all PRVs in g_t^i from $E_{0:T}$ yielding a multivariate time series $X^{|\mathcal{A}| \times T}$, followed by applying the symbolisation scheme to yield $\mathcal{X}^{|\mathcal{A}| \times (T - \tau(d-1))}$. Second, similarity counting is performed to set up the similarity matrix $\mathcal{W}^{n \times n}$, which is used for clustering to obtain similarity clusters $S_{|g_t^i}$.

The similarity matrix per parfactor has a size of $n \times n$ with $n = |gr(g_t^i)|$, i.e., the number of domain objects the parfactor $g_t^i = \phi(\mathcal{A})_{|C^i}$ is limited to by means of its constraint C^i. Symmetry clustering is done individually for each parfactor with $\mathcal{S}$ denoting the set of all object symmetry clusters for all parfactors.

In the following chapter, we demonstrate the use of symmetry clusters to prevent unnecessary groundings in a lifted model. We introduce another algorithm, SA_4PG (Symmetry Approximation for Preventing Groundings), which operates on the output of MOP_4SA. Finally, by integrating MOP_4SA and SA_4PG in the online version of LDJT, we provide empirical proof of the effectiveness of these algorithms in achieving their objectives.

8 Preventing Groundings a Priori

In any lifted model, evidence leads to groundings. With no further countermeasures, groundings are carried over in message passing when moving forward in time leading to a fully grounded model in the worst case. That is, in the worst case, no more symmetries are present in the model, and thus, the advantages of lifting dissolve. For temporal models, Gehrke *et al.* (2020) propose to create a new lifted representation by merging groundings introduced over time. In comparison to that approach, no one has focused on preventing groundings caused by evidence before they even occur, which we propose in this chapter providing contribution 3b. In the previous chapter, we introduced MOP_4SA as an algorithm to construct groups of objects with similar behaviour denoted as *symmetry clusters*. Using the symmetry clusters, it is possible to selectively prevent groundings, i.e., to accept, reject or align incoming events, i.e., yielding intrinsic defaults, to prevent the model from grounding, which helps to retain a lifted representation. Deriving intrinsic defaults not only helps to retain a lifted representation but can also lead to higher accuracy in predictions, especially for those domain objects for which few events are observed. This chapter is structured as follows. In **Section 8.1** we first elaborate on the concept of yielding intrinsic defaults, followed in **Section 8.2** by introducing *symmetry approximation for preventing groundings*, SA_4PG for short, an approach that uses symmetry clusters to counteract any unnecessary groundings. Finally, in **Section 8.3**, we perform an empirical evaluation based on our reference application from shipping comparing MOP_4SA in conjunction with SA_4PG when answering online queries using LDJT, and show that groundings can be efficiently prevented while accuracy in inference only suffers slightly.

The content of this chapter is published in the following conference paper, while the following journal article provides further specifications as well as a detailed evaluation.

Nils Finke and Marisa Mohr. A Priori Approximation of Symmetries in Probabilistic Dynamic Relational Models. In Stefan Edelkamp, Ralf Möller, and Elmar Rueckert (Eds.), *KI 2021: Advances in Artificial Intelligence*, pages 309–323. Springer, 2021.

Nils Finke and Ralf Möller. On the Approximation of Symmetries and Related Structural Changes in Dynamic Probabilistic Relational Models. *Advances in Science, Technology and Engineering Systems Journal*, 7(2), 2022.

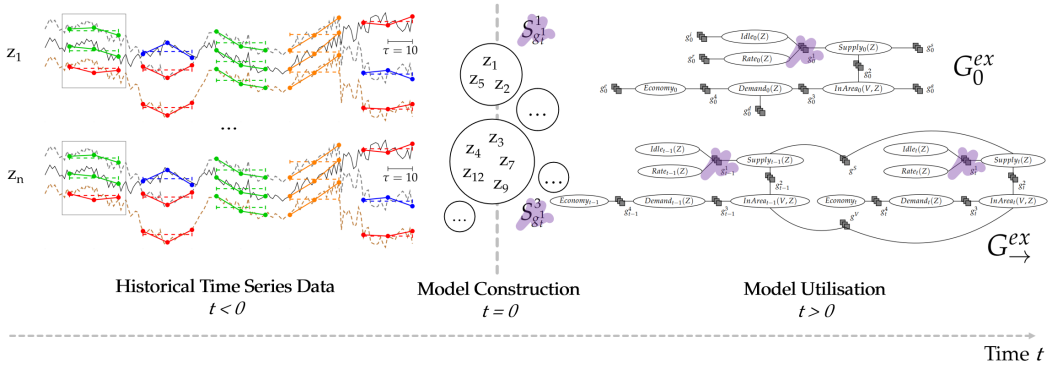
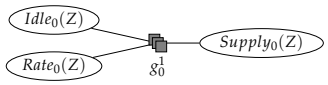


Figure 8.1.: Simplified representation of the individual process steps from model construction based on historical data to answering queries based on the derived model. While MOP$_4$SA is running during model construction, SA$_4$PG is running as part of LDJT before encoding evidence, i.e., during the model utilisation phase.

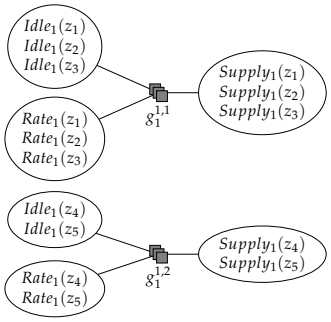
8.1 Yielding Intrinsic Defaults

Before we come to inferring intrinsic defaults to prevent groundings, we first lay out the overall process, i.e., the step from the construction of symmetry clusters to their actual use to prevent groundings. Figure 8.1 gives a high-level overview aligned along the temporal axis. Here, the point in time for $t = 0$ marks the time of model construction. During model construction, (a) the DPRM itself is set up, which includes determining the potential functions of each parfactor, and, (b) on the other hand, the determination of symmetry clusters per parfactor using MOP$_4$SA. For both steps, historical data, i.e., sequences of historical events, for the individual domain objects is used. We assume that the past is (was) completely observable and that, therefore, complete information about the historical behaviour of individual domain objects is available. With respect to our example, see Fig. 8.1, historical data ($t < 0$) is available in the form of multivariate time series for all domain objects, as depicted in the figure for $z_1, \ldots, z_n$. Given the multivariate time series, symmetry clusters using MOP$_4$SA are determined yielding clusters of domain objects for each parfactor in the DPRM, such as clusters $\mathbf{S}^1_{g^1_t}$ and $\mathbf{S}^3_{g^1_t}$ that both refer to the parfactor g^1_t in the instantiated DPRM as highlighted in purple.

Once the DPRM has been set up, the model is updated over time ($t > 0$) by means of adjusting parfactor potential functions as new knowledge, i.e., events become known (see online LDJT). In the preliminaries in Section 3.2, we have elaborated on encoding events as evidence parfactors replacing the original parfactor by two new parfactors. One parfactor is created to encode the object affected by an event and the other parfactor for all remaining domain objects. Our approach starts precisely in this ongoing process of incorporating new observations over time, in order to prevent inappropriate groundings. In general, we aim at



(a) Fully lifted submodel of G^{ex} for sub-challenge one at time step $t = 0$.



(b) A submodel of G^{ex} for sub-challenge one at time step $t = 1$ shattered according to symmetry clusters as described in Example 8.1.1.

Figure 8.2.: The problem of events leading to groundings and the use of symmetry clusters to prevent same. The sub-figures depict local models at time steps $t = 0$ (left) and $t = 1$ (right). For reasons of simplicity, we omit the representation of interslice parfactors, i.e., transitions between the time steps.

preventing groundings that occur due to two different types of collections of observations with respect to symmetric domain objects, i.e., those that are still represented by a common parfactor in a local model, i.e., at a particular time step. This is either

1. **partial evidence**, i.e., only for a subset of the domain objects represented by a common parfactor in a local model, evidence is available, or
2. **conflicting evidence**, i.e., for a few domain objects in a set of domain objects represented by a common parfactor, divergent events are observed compared to the majority of domain objects.

For better illustration, we first give an example of a model that is shattered according to the domain objects in symmetry clusters, which we use to explain how symmetry clusters can be used to prevent further groundings.

Example 8.1.1 (Towards Deriving Intrinsic Defaults)**.** *Let us consider PRVs related to sub-challenge 1 as introduced in Chapter 2, i.e., PRVs $Idle_t(Z)$, $Rate_t(Z)$, and $Supply_t(Z)$ connected by parfactor g_t^1. For simplicity in this example, we assume $\mathcal{D}(Z) = \{z_1, z_2, z_3, z_4, z_5\}$ with objects in $\mathcal{D}(Z)$ divided into two symmetry clusters $\mathbf{S}^1_{g_t^1} = \{z_1, z_2, z_3\}$ and $\mathbf{S}^2_{g_t^1} = \{z_4, z_5\}$ for parfactor g_t^1. In the initial model state, i.e., at $t = 0$ the submodel is still in a fully lifted state as illustrated in Fig. 8.2a with the logvar Z as a representative for all domain objects. However, assuming the validity of the symmetry clusters, it is expected that the model will be split as represented in Fig. 8.2b in time step $t = 1$. Since symmetry clusters are groups of objects whose behaviour is symmetric with respect to each other, as reflected by the fact that typically the same events are observed, it is expected that this will continue to be the case if the clusters remain valid. This is the case, for example, when the same*

observations are made for groups of the domain objects respectively, i.e., for the group of objects z_1, z_2 and z_3 and the group of objects z_4 and z_5, with respect to the variables referenced by the parfactor g_t^1.

Now, to explain the concept of deriving intrinsic defaults, consider the model G_1^{ex} as represented in Fig. 8.2b, i.e., at time $t = 1$, as a starting point for further queries for $t > 1$ given some further evidence that becomes known over time. We will start by describing the first case, i.e., partial evidence leading to groundings, which can be prevented by exploiting symmetry clusters.

Breaking Symmetries due to Partial Evidence

Suppose that events are given for only a subset of the domain objects, which are all in the same symmetry cluster and further represented by a common parfactor as elaborated on in the previous example. Then the model will be further shattered, i.e., additional groundings will occur that split the domain object in a symmetry cluster according to the evidence given. Those objects for which no events are available will continue to be encoded in the parfactor in which they are currently encoded, but for those domain objects for which events are available, new evidence parfactors will be added to the model. Now, symmetry clusters can be exploited to prevent groundings. Knowing that objects in a symmetry cluster share similar behaviour, the events observed only for a subset of the domain objects can implicitly be applied to all domain objects in the same symmetry cluster, as it's likely that those domain objects continue to show similar behaviour. By doing so, no new evidence parfactors are added to the model and instead, the potential function encoded in the existing parfactor that applies to the group of domain objects in a symmetry cluster is adjusted according to the events. We continue with the previous example and add to the example accordingly.

Example 8.1.2 (Partial Evidence)**.** *We observe events $Idle_1(z_1) = low$, $Rate_1(z_1) = high$, and $Supply_1(z_1) = medium$ for the domain object z_1 only. Given the model in a state G_1 as illustrated in Fig. 8.2b, these events (without preventing groundings) would be encoded in the model such that a new evidence parfactor $g_1^{1,3}$ is added limited by constraints to just the domain object z_1, leading to in total 3 groundings for the parfactor g_t^1 at the current time step. The domain objects z_1, z_2, and z_3, would only continue to be treated together in one group, i.e., by one common parfactor if for all domain objects the same or no events were observed. Now, using symmetry clusters, i.e. taking advantage of the knowledge that it is likely that the domain objects actually exhibit the same behaviour just observed for a subset of objects, an algorithm can apply these observations not only to the domain objects actually observed but to the entire group of domain objects, i.e., to all domain objects that are in the same symmetry cluster as those for which the events were observed. This is in this example, applying the events $Idle_1(z_1) = low$, $Rate_1(z_1) = high$, and $Supply_1(z_1) = medium$ observed for z_1, to all domain objects in $\mathbf{S}^1_{g_t^1}$, i.e., also to z_2 and z_3. By doing so, no new evidence parfactor*

is added to encode the new information, but the existing potential function of the parfactor $g_1^{1,1}$ is adjusted to cater for the events, i.e., no new groundings occur.

Next, we elaborate on the second case on how to prevent groundings, which is aligning conflicting evidence.

Breaking Symmetries due to Conflicting Evidence

The second case where groundings occur is due to different observations for domain objects that are still represented by a common parfactor in the current local model, i.e., at a certain point in time. However, if these domain objects are in the same symmetry clusters, i.e., if the domain objects are expected to behave similarly in the long run, then single observations that conflict with the majority of observations for objects in a symmetry cluster can be classified as outliers and discarded, so that the *dominating* observations are applied to all domain objects of a symmetry cluster, and thus the behaviour is aligned again between all objects and no new evidence parfactors are created. We give the following example.

Example 8.1.3 (Conflicting Evidence)**.** *Given the events $Rate_1(z_1) = low$, $Rate_1(z_2) = high$ and $Rate_1(z_3) = high$ for objects z_1, z_2, and z_3, which are altogether in a symmetry cluster $\mathbf{S}^1_{g_1^1}$ and represented via one evidence parfactor $g_1^{1,1}$ in G_1, i.e., prior encoding those new events, the domain objects would now no longer show symmetric behaviour as the events differ. To prevent groundings, we consider $Rate_1(z_1) = low$ as an outlier as the majority of events are $Rate_1 = high$ for domain objects in the same symmetry cluster. Therefore, the event $Rate_1(z_1) = low$ is rejected, i.e., no new evidence parfactor is created to encode this event, but the event $Rate_1 = high$ is considered as the dominating event and is applied to all domain objects accordingly. By doing so, no new evidence parfactor is added to encode the new information, but the existing potential function of the parfactor $g_1^{1,1}$ is adjusted to cater for the events, i.e., no new groundings occur.*

In the first scenario, preventing groundings can potentially enhance prediction accuracy. However, in this scenario, the accuracy of model predictions will certainly suffer, which nevertheless is expected to be negligible as domain objects will align again in the long run if the validity of the symmetry clusters is given. Next, we combine those two cases into the algorithm SA_4PG.

8.2 Applying an Intrinsic Default to Align Object Behaviour (SA_4PG)

In this section, we introduce *symmetry approximation for preventing groundings*, SA_4PG for short, an algorithm which derives intrinsic defaults from symmetry clusters as elaborated in the previous section. While symmetry clusters are constructed using MOP_4SA as part of the model construction process at $t = 0$, SA_4PG is running as part of online LDJT to prevent the model from unnecessary groundings aligning or rejecting events. SA_4PG is executed before

evidence $\mathbf{E}_t$ at a particular time step t is encoded in the local model G_t in the classical flow of LDJT, as described in Algorithm 2. Events in $\mathbf{E}_t$ are (rejected) and aligned for domain objects according to the given symmetry cluster. Symmetry clusters are determined on a parfactor level, and as such (rejecting) and aligning events is done accordingly for each parfactor separately. To not force domain objects to stick to their initial symmetry clusters and in consequence keep those objects in one common evidence parfactor, objects are relieved from their symmetry clusters until an object has reached a violation threshold H, which is a count on how often events were aligned or rejected for a specific domain object. To keep track of the number of violations, i.e., how often events were aligned or rejected, we introduce a violation map as a helper data structure to store the number of violations.

Definition 8.2.1 (Violation Map)**.** For a parfactor $g_t^i \in G_t$ with G_t being the local model at time step t and $g_t^i = \phi_t^i(\mathcal{A})_{|C^i}$ comprising a sequence of PRVs $\mathcal{A} = (A^1, \ldots, A^n)$, a violation map $v_{|g_t^i} : \bigcup_{i=1}^n gr(A^i) \to 0$ is initialised with zero values for all objects in all PRVs $\mathcal{A}$ in g_t^i. In case a PRV A^i is parameterised with more than one logvar, i.e., $m = |lv(A^i)|$ with $m > 1$, v maps m-tuples as object pairs to the violation count. A model includes up to n parfactors in G_t. The set of violation maps is denoted by $\mathbf{V} = \{v_{|g_t^i}\}_{i=0}^n$. Let $viol(P)$ refer to the violation count of some object m-tuple P in $\mathbf{V}$.

Let's take the following example, which follows up on the example in the previous section.

Example 8.2.1 (Violation Map)**.** *Given a model in a state as depicted in Fig. 8.2b, symmetry clusters* $\mathbf{S}^1_{g_t^1} = \{z_1, z_2, z_3\}$ *and* $\mathbf{S}^2_{g_t^1} = \{z_4, z_5\}$*, and a violation map* $v_{|g_t^1}$ *for parfactor* g_t^1 *then*

$\mathcal{D}(Z)$	H
z_1	0
z_2	0
z_3	0
z_4	0
z_5	0

- *aligning events within the symmetry cluster as described in Example 8.1.2 for domain objects in* $\mathbf{S}^1_{g_t^1}$ *leads to an increase of H for the unobserved domain objects such as here* z_2 *and* z_3*, or,*
- *rejecting an event for a domain object* z_2 *as described in Example 8.1.3, leads to an increase of H for the domain objects for which an event was rejected such as here* z_2*.*

To make rejecting and aligning events easier, we introduce another helper structure used during preventing groundings, which we refer to as *evidence clusters*. Creating evidence clusters is done by partitioning the set of events $\mathbf{E}_t$ given at a time step t according to a specific parfactor and its symmetry clusters. More specifically, a set of events

$$\mathbf{E}_t = \{E_t(\mathbf{x}^i) = o\}_{i=1}^m \tag{8.2.1}$$

of some PRV $E_t(\mathbf{X})$ given at a specific time step t (where the events are to be encoded as evidence parfactors $g^e = \phi^e(E_t(\mathbf{X}))_{|C^e}$ in the local model G_t), is rearranged in a sense such that $\mathbf{E}_t$ includes multiple collections of observations, i.e.,

$$\mathbf{E}_t{}' = \left\{ \{E_t(\mathbf{x}^i) = o\}_{i=0}^n, \ldots, \{E_t(\mathbf{x}^i) = o\}_{i=0}^k \right\}, \tag{8.2.2}$$

with each element E_t originally being directly in $\mathbf{E}_t$ and $n + k = m$. Further, each subset $\{E_t(\mathbf{x}^i) = o\}_{i=0}^n$ relates to one symmetry cluster in the set of symmetry clusters $\mathbf{S}_{|g_t}$ of a specific parfactor g_t, i.e.,

$$\forall \{E_t(\mathbf{x}^i) = o\}_{i=0}^n \in \mathbf{E}_t{}' : \exists\, \mathbf{S}^l \in \mathbf{S}_{|g_t} \quad \text{with} \quad \mathbf{x}^i \subseteq en(\mathbf{S}^l) \tag{8.2.3}$$

We give the following example.

Example 8.2.2 (Rearranging Evidence in Evidence Clusters)**.** *Given a set of events, e.g.,*

$$\begin{aligned} \mathbf{E}_1 = \{&Idle_1(z_1) = high, Idle_1(z_2) = high, Idle_1(z_3) = low, \\ &Rate_1(z_4) = high, InArea_1(z_1) = true\} \end{aligned} \tag{8.2.4}$$

and symmetry clusters $\mathbf{S}^1_{g_t^1} = \{z_1, z_2, z_3\}$ *and* $\mathbf{S}^2_{g_t^1} = \{z_4, z_5\}$ *for a parfactor* g_t^1*,* $\mathbf{E}_1$ *is partitioned into*

$$\mathbf{E}_1{}' = \Big\{ \{Idle_1(z_1) = high, Idle_1(z_2) = high, Idle_1(z_3) = low\}, \{Rate_1(z_4) = high\} \Big\} \tag{8.2.5}$$

such that each subset relates to a symmetry cluster currently under consideration of a specific parfactor. More specifically, the first subset in $\mathbf{E}_1{}'$ *relates to* $\mathbf{S}^1_{g_t^1}$ *and the second subset to* $\mathbf{S}^2_{g_t^1}$*. The event* $InArea_1(z_1) = true$ *is not considered in* $\mathbf{E}_1{}'$ *as the parfactor under consideration, here* g_t^1*, is not connected with the PRV* $InArea_t(V, Z)$ *in* G_t*, i.e., the event* $InArea_1(z_1) = true$ *needs to be encoded as another evidence parfactor.*

Therefore, encoding events is done by iterating over each parfactor in G_t, partitioning $\mathbf{E}_t$ according to symmetry clusters of the specific parfactor and subsequently rejecting and aligning events in each evidence cluster. Evidence clusters are a counterpart to symmetry clusters. In the previous section, we have introduced the two cases, i.e., the alignment of events within an evidence cluster, if necessary also under rejection of single events. This is being done by identifying the *dominating observation* of each evidence cluster and applying that observation to all objects of a symmetry cluster accordingly if the threshold H is not yet met. The dominating observation is the observation that can be observed the most within the evidence cluster, i.e., selecting the dominating observation based on a simple majority score as demonstrated in the following example.

Algorithm 6 Preventing groundings a priori (SA$_4$PG)

Input: Local model G_t, Evidence $\mathbf{E}_t$, Order d, Delay τ, Symmetry Clusters $\mathbf{S}$
for *each parfactor* $g_t^i \in G_t$ **do**
  $v_{|g_t^i} \leftarrow$ init violation map ▷ see Definition 8.2.1
  $\mathbf{S}_{|g_t^i} \leftarrow$ get symmetry clusters for g_t^i from $\mathbf{S}$
  $\mathbf{E}_t{}' \leftarrow$ Partition $\mathbf{E}_t$ according to $\mathbf{S}_{|g_t^i}$ ▷ see Eqs. (8.2.2) and (8.2.3)
  for each evidence cluster $\{E_t(\mathbf{x}^i) = o\}_{i=0}^n \in \boldsymbol{E}_t'$ **do**
    $o' \leftarrow$ get dominating observation
    for $\mathbf{x}^i$ **do**
      Apply dominating $E_t(\mathbf{x}^i) = o'$ observation

Example 8.2.3 (Dominating Observation in Evidence Clusters)**.** *Given evidence clusters for* g_1^1

$$\mathbf{E}_1{}' = \Big\{ \big\{ Idle_1(z_1) = high, Idle_1(z_2) = high, Idle_1(z_3) = low \big\}, \big\{ Rate_1(z_4) = high \big\} \Big\} \tag{8.2.6}$$

with each evidence cluster relating to an evidence parfactor in G_1*, here* $\{ Idle_1(z_1) = high, Idle_1(z_2) = high, Idle_1(z_3) = low \}$ *relating to* $g_1^{1,1}$ *with its dominating observation being* $Idle_1(Z) = high$ *and* $\{ Rate_1(z_4) = high \}$ *relating to* $g_1^{1,2}$ *with its dominating observation being* $Rate_1(Z) = high$*, preventing groundings works applying the dominating observation to all domain object in each symmetry cluster.*

Algorithm 6 depicts the outline of SA$_4$PG given symmetry clusters as an outcome of MOP$_4$SA to prevent groundings by aligning object behaviour, i.e., limiting outliers and deriving defaults for unobserved domain objects. In the next section, we evaluate MOP$_4$SA and SA$_4$PG as part of a case study from our example shipping domain to validate its effectiveness in preventing groundings while only introducing a considerably small error in prediction accuracy.

8.3 Empirical Evaluation: MOP$_4$SA and SA$_4$PG in Applications

This section concludes with an empirical evaluation of MOP$_4$SA and SA$_4$PG on the dry-bulk shipping example introduced in Chapter 2 and thus provides contribution 3c, which is proving that MOP$_4$SA together with SA$_4$PG helps to avoid groundings effectively and therefore speeding up inference while only introducing a considerably small error in the prediction accuracy. Evaluation is done on AIS data[1] provided by the Danish Maritime Authority (DMA) for the Baltic Sea, on the basis of which we set up a DPRM for the submodel

[1] https://dma.dk/safety-at-sea/navigational-information/ais-data

around the parfactor g_t^1. To recap again, the submodel

$$g_t^1 \;=\; \phi_t^1(Idle_t(Z), Rate_t(Z), Supply_t(Z)) \quad \text{and} \quad g^S \;=\; \phi^S(Supply_t(Z), Supply_{t+1}(Z)) \tag{8.3.1}$$

describes the interaction between idle times, freight rates and supply in zones across the globe over time. All steps to derive the necessary data for the model setup based on the AIS dataset are described in the Appendix A.1 and resulting data for variables $Supply_t(Z)$ and $Idle_t(Z)$ for 367 defined $Zones$ over time t can be found on GitHub[2]. The potentials of the potential function ϕ_t^1 are derived from the AIS dataset for $t = 1, \ldots, 5$ by counting occurrences of the dependent events, i.e., the different states of the DPRM. Further, the same data for $t = 1, \ldots, 5$ is used to determine symmetry clusters using MOP$_4$SA. To validate the hypothesis formulated, we proceed as follows. Using online LDJT, we perform inference by answering the prediction query $P(Supply_t(Z), Idle_t(Z)|\mathbf{E}_t)$ for each time step $t > 5$ providing events $\mathbf{E}_t$ from the dataset and obtain a marginal distribution for each entity $z \in \mathcal{D}(Z)$. However, for each time step, we answer the same query two times:

1. Once without the intermediate step of preventing groundings, i.e., without running SA$_4$PG prior query answering, and,

2. once with running SA$_4$PG prior query answering.

The results of the prediction queries without SA$_4$PG are considered as the ground truth and are used to compare against the results of the prediction queries with running SA$_4$PG. To evaluate the accuracy, we compare the number of correct predictions generated in the second iteration, i.e., when SA$_4$PG is incorporated as an intermediate step for answering the query, to the total number of predictions made. In this context, the ground truth serves as the benchmark for assessing the correct predictions. In addition, we evaluate runtime in seconds s to answer the prediction query at all time steps t. The negative impact of SA$_4$PG on the accuracy in inference depends on the quality of the symmetry clusters as an outcome of MOP$_4$SA. More specifically, if symmetry clusters include objects that do not exhibit symmetry with one another, the error introduced by our approach to reduce groundings will also lead to a larger error in prediction accuracy. As MOP$_4$SA is affected by (a) the efficiency of the clustering algorithm used, (b) the similarity measure itself, (c) and its hyperparameters such as order d, delay τ and δ for the arithmetic mean as defined in Eq. (7.3.6), the experiment is repeated with varying parameter combinations. We ran 54 experiments in total with different parameter combinations $d \in \{2, 3, 4\}$, $\tau \in \{1, 2, 3\}$, $\delta \in \{0.05, 0.1, 0.15\}$. Figure 8.3 depicts a comparison between the accuracy for both cases and different parameter combination. The corresponding data for the plots can be found in the Appendix in Table A.2. Each subplot corresponds to a different order d and delays τ, while the box-plot itself de-

[2]https://github.com/FinkeNils/Processed-AIS-Data-Baltic-Sea-2020-v2

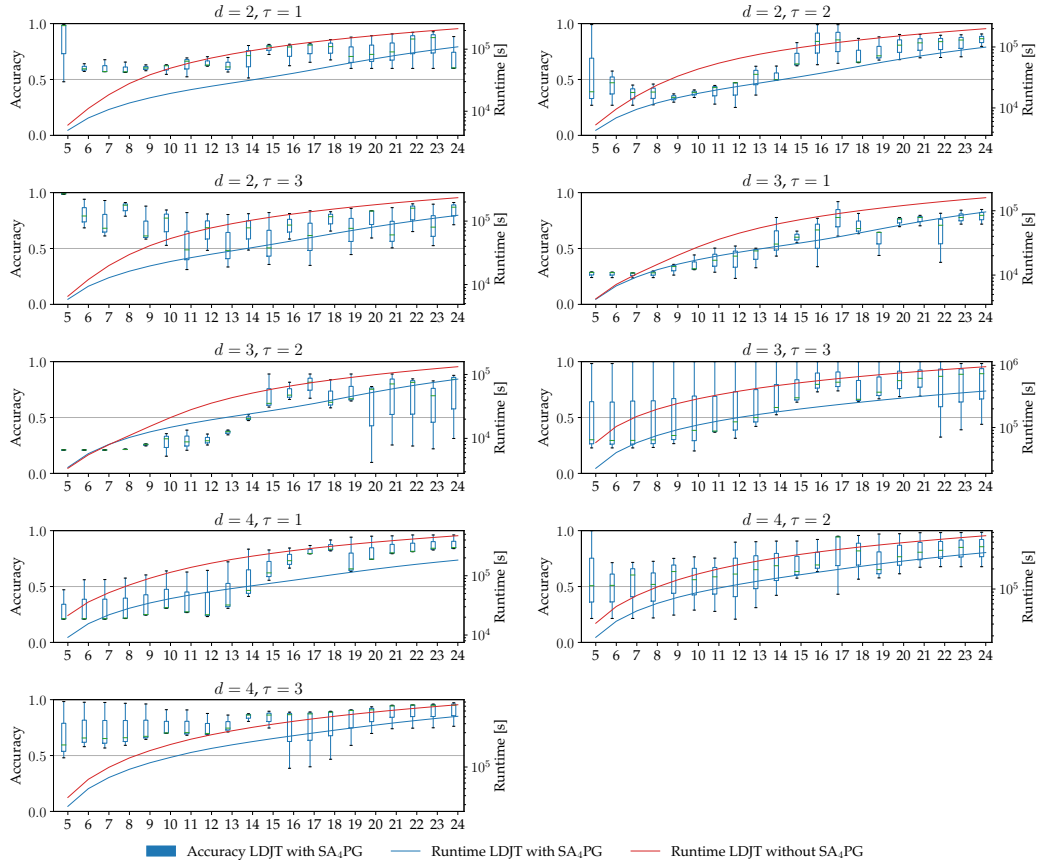


Figure 8.3.: Accuracy and runtime in inference under SA$_4$PG for different parameter combinations with symmetry clusters determined by DBSCAN. The x-axis corresponds to different time steps t.

picts the variation of the accuracy over different deltas $\delta = \{0.05, 0.1, 0.15\}$ over time t. The box plots represent the prediction accuracy as a function of the parameter δ for the interval $\delta = [0.05, 0.15]$. The solid blue and red lines correspond to the runtime for answering a query for the specific time step.

Generally, over time, new events continue to shatter the underlying model, resulting in a longer runtime. This is consistent across all experiments regardless of the parameter combination and expected behaviour. This is where preventing groundings becomes important, keeping a lifted model as long as possible and increasing runtime in inference. Compared to exact inference (red line), the runtime is significantly lower for inference including SA$_4$PG (blue line) across all experiments. This advantage must be considered in terms of influence on accuracy. More precisely, Fig. 8.3 reveals that the runtime savings (spread between the red and blue lines) for predictions with increasing t, i.e., for queries that require rolling out the local model G_t from a time when the model is initially set up ($t = 4$). On average over

all experiments, the spread between runtimes of LDJT with and without SA$_4$PG starts to increase up to a time of $10 < t < 18$ and then only slightly decreases. The spread varies here in the maximum between a reduction of the runtime from 25% to 50% (experiment with $d = 2$ and $\tau = 3$), or in the minimum between a reduction of the runtime from 25% to 35% (experiment with $d = 4$ and $\tau = 3$). The fact that the spread decreases only negligible little even after the maximum spread is reached, is interesting, as preventing groundings is not done over the entire runtime, but only until an object in a symmetry cluster reaches the object's internal threshold H (the threshold for how often events are allowed to be rejected or inferred). That is, preventing groundings, even if performed for only a short period of time (here until $H = 5$), will maintain the advantage of improved runtime benefits for a long time.

Next, we consider SA$_4$PG in the context of its impact on accuracy in predictions. For this, we look at the results of the experiment for $d = 2$ and $\tau = 3$, as we achieve good results with this parameter combination. In the first time steps, i.e., for $5 < t < 12$, the accuracy slowly decreases until the accuracy reaches its lowest point and then slowly increases again. In all experiments, the violation threshold H is set to 5, i.e., it is expected that the majority of groundings are prevented in the initial time steps after learning the clusters. If, in the process of preventing groundings, domain objects were artificially held for too long in a group of objects that do not actually exhibit temporal symmetries with respect to each other, this would negatively impact accuracy. In the experiment featuring $d = 2$ and $\tau = 3$, the accuracy decreases only slightly, indicating that the symmetry clusters effectively capture domain objects with temporal symmetries. The accuracy then starts to increase again at $t > 12$, which can be explained by the fact that after reaching the violation threshold H, objects are no longer held together, so that over time an error that was introduced early in the model, slowly decreases again and thus the accuracy with respect to the ground truth increases again. A similar result can be observed for the parameter combination with $d = 4$ and $\tau = 3$. This implies that $H = 5$ is reached relatively quickly at the beginning of the runtime. Here we can assume that, on average, H increases per time step for some domain objects, due to the fact that the accuracy is decreasing initially, but then starts to increase again after some time steps, specifically at $t = 5 + H$ when the preventing groundings gradually starts to phase out. This further demonstrates that MOP$_4$SA is not able to capture long-term symmetries, i.e., recurring patterns or periodicity over a long period of time. The determination of the similarity between two domain objects is further dependent on the data range of the ordinals of a respective domain object determined by the parameter δ. As δ increases, the importance of ordinals within the same data range decreases, resulting in being quantified as similar. Consequently, in MOP$_4$SA, the actual trajectory of the time series, including its fluctuations, becomes more relevant in determining similarity. It is noticeable, that especially for $d = 4$ and $\tau = 3$ the deviation in the box plot is very small, suggesting that for this parameter combination the dependence of a low δ is unimportant for quanti-

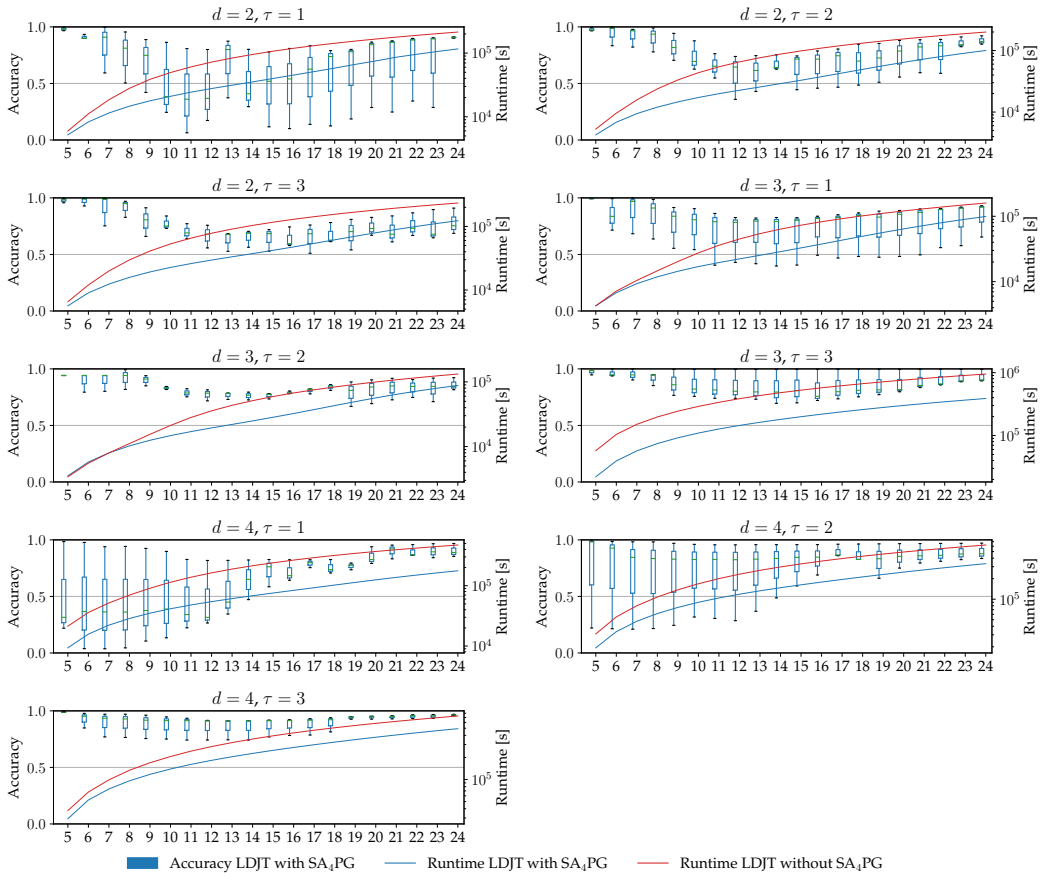


Figure 8.4.: Accuracy and runtime in inference under SA$_4$PG for different parameter combinations with symmetry clusters determined by spectral clustering.

fying similar ordinals. Generally, with increasing order d the number of neighbouring data points is increasing, i.e., the classification encompasses more long-term patterns. Further, with increasing delay τ, long-term behaviour is extended even further, while also allowing for temporary deviations. The plots depict that for higher orders d and delays τ, which correspond to a longer time period represented by each ordinal, the prediction accuracy tends to decrease or show greater fluctuations, as indicated by the variations in the box plots.

Since the effectiveness of SA$_4$PG is strongly dependent on the determination of the symmetry clusters, we give a small excursion and perform the following experiment again, but this time with the difference that in MOP$_4$SA spectral clustering is used as the clustering approach.

Excursion: Spectral Clustering as a Clustering Approach in MOP$_4$SA

While DBSCAN is advantageous for clustering, since the number of clusters does not have to be specified, spectral clustering is known to be beneficial for use with high dimensional data as dimensionality reduction is performed prior to clustering. Depending on the domain sizes, MOP$_4$SA has to deal with high dimensional data as in our case in which we perform clustering basis a similarity matrix of dimension 367×367. Figure 8.4 represents the results of the experiments including Spectral clustering in MOP$_4$SA. For comparison, we have set the number of clusters to be determined by spectral clustering to the number of clusters determined by DBSCAN for better comparison. Comparing the results represented in Fig. 8.4 with the results in the previous section, it is evident that spectral clustering is better able to identify symmetry clusters. While DBSCAN in MOP$_4$SA gives the best results for two parameter combinations, spectral clustering in MOP$_4$SA gives good results for further experiments with other parameter combinations. More specifically, plots for experiments with parameter-combinations of $d = 3$ and $\tau = 1$, $d = 3$ and $\tau = 2$, and $d = 3$ and $\tau = 3$ indicate high accuracy.

Generally, when reasoning under time constraints, preventing groundings is a reasonable approach as it prevents groundings in the long term and therefore speeds up inference. However, symmetry (structures) can change over time. That is, to further prevent the model from grounding it is beneficial to relearn symmetry structures at some time. In the next part of this work, we introduce techniques to detect such changes in either probability distributions or symmetry structures.

9 Part II: Interim Conclusion

Evidence over time leads to groundings in a DPRM, which reduces runtime benefits in lifted inference. In this part of this work, we have introduced two algorithms MOP_4SA and SA_4PG, where the former determines temporal symmetries over domain objects of a DPRM, and the latter uses the identified temporal symmetries between domain objects to prevent groundings a priori in reasoning. More specifically, MOP_4SA detects similarities across objects of the model's domains using a multivariate ordinal pattern symbolisation approach. Based on a similarity graph, clustering identifies sets of objects with symmetrical behaviour, so-called symmetry clusters, with respect to the context of the model, i.e., PRVs linked via a common parfactor. These symmetry clusters are then used in SA_4PG, which itself runs before encoding evidence in the process of answering queries using LDJT, to prevent unnecessary groundings due to evidence. More specifically, SA_4PG selectively prevents groundings before they even occur by accepting, rejecting, and consequently aligning incoming events within symmetry clusters, i.e., inferring intrinsic defaults to keep domain objects of the same symmetry cluster in groups and avoid splitting.

We show that MOP_4SA is not able to capture long-term symmetries, i.e., recurrent patterns or periodicity over a long time period, e.g., due to seasonality. Rather, MOP_4SA aims to determine symmetry structures that remain valid for shorter time periods. This has the advantage that MOP_4SA only requires a small amount of *training data* to achieve a good approximation of the symmetry structures. Obviously, MOP_4SA can also be extended to capture long-term symmetry structures, e.g., by increasing the order d. However, this also increases the complexity of the whole approach, i.e., the model complexity, the number of training data required, as well as the run-time complexity, so we restrict ourselves to short-term symmetries in this work. Further, we show that the a priori introduction of symmetry clusters to prevent groundings, i.e., using SA_4PG, speeds up inference while introducing only a very small error in terms of accuracy when answering a query. By preventing groundings a priori, we complement existing approaches which focus on retaining the lifted representation after a model has already been split. Moreover, our approach works well with any other approach that undoes splits, for example, together with the approach of Gehrke *et al.* (2020), which merges groups of objects when they align again over time. Combining both kinds of approaches brings together the best of both worlds: (a) With *determination of temporal model symmetries* we can use the entire set of historical training data as part of the model

construction process to not only derive distributions for the model itself but also to determine symmetry clusters which itself can be used to prevent groundings a priori, (b) and with *temporal approximate merging* we can merge non-preventable groundings, i.e., evidence parfactors, even after their occurrence, i.e., a posterior. The introduction of MOP_4SA and SA_4PG, in combination with known approaches, open up great potential for improvements in StaRAI to be explored in future work. We briefly discuss two aspects for improving the approaches presented below.

Since MOP_4SA is designed to uncover short-term symmetry clusters with relatively little training data, the frequency with which MOP_4SA must be (re-)run to determine new symmetry clusters is also higher. The overhead that MOP_4SA incurs in query answering is (a) depending on the number of executions of MOP_4SA itself, and, (b) depending on the complexity of the approaches for symbolisation and clustering. The use of the presented symbolisation scheme to identify symmetries is a performant approach (Piek *et al.*, 2019), however, the cost of the selected clustering approach as a final step in MOP_4SA is noticeable. Nevertheless, the investigation of more powerful clustering approaches, e.g., exploiting some kind of incremental changes to the clustering after the initial learning step, is reserved for future work. However, the overhead of MOP_4SA can be reduced by determining the point in time at which it is useful to determine new symmetry clusters, thereby reducing the number of total executions of MOP_4SA. Since the clustering approach is based on the degree of similarity represented by the similarity score as edge weights between individual domain objects, i.e., here on the similarity graph, we can use the shifts in edge weights across the substructures (i.e., the similarity clusters) in the graph to derive when a redefinition of the symmetry clusters is justified. This introduces a second type of change point, in addition, to change points due to drifts in the probability distributions over time, two aspects we examine for DPRMs in the next part. Furthermore, even though inferring and aligning events within a symmetry cluster to prevent groundings has advantages, it also happens that events are applied that should not have been applied for all domain objects in the symmetry cluster. One case, for example, is whenever events for a domain object are not available at runtime and therefore events are inferred basis the behaviour of domain objects within the same symmetry cluster but then events become known downstream revealing that the events inferred initially were actually wrong. In such cases, one would like to roll back the events that were transmitted incorrectly and undo the adjustments to the potential functions caused by the events, i.e., perform *Belief Revision*, which is open for future work.

Part III

Detecting Environmental Changes

10 Towards Environmental Change Point Detection in DPRMs

Recurring patterns and symmetries can be observed everywhere in nature and are therefore also reflected in the observed data. In the context of DPRMs, recurring patterns and symmetry refers to the domain objects within the model exhibiting symmetry with respect to the variables of the model, sharing the same potentials in potential functions that describe the likelihood of events occurring. In the first part of this work, in which we extended DPRMs with dynamic domains, we made use of the assumption that symmetrical domain objects exist when extending the model by adding a new domain object. Given this expectation, information (in the form of adjusted potential functions due to new observations) for groups of domain objects that already exist in the model and exhibit symmetries to the new domain object can be transferred to the new domain object. As a result, the prediction accuracy for the new domain object is improved while maintaining a lifted representation. In the second part of this work, we presented an approach for the detection of temporal symmetry (structures), i.e., sets of domain objects with similar behaviour arranged in so-called symmetry clusters. By exploiting symmetry clusters, we presented an approach to imply intrinsic defaults, which are events for domain objects, that are e.g., unobserved, reducing groundings. Both the potential functions for parfactors to describe the potential for individual events to occur and the symmetry clusters are obtained as part of the initial model construction process. For better understanding, we revisit Fig. 10.1, which we presented partly in the previous part to illustrate the process of collecting historical (time series) data to model construction and usage. Given the DPRM and the symmetry clusters established at $t = 0$, it is likely that the potential functions encoded in the parfactors and the symmetry structures will change over time ($t > 0$) due to any external effects in the, e.g., real-world environment. Especially in any online scenario, i.e., during model utilisation with new information, i.e., realisations of randvars yielding new time series data, become known, it is essential to have mechanisms in place that detect these changes, thereby ensuring that predictions are made using a model that continues to accurately represent reality. An illustration in Fig. 10.1 of evolving symmetry clusters is presented for the time step t', where different domain objects from those at the beginning of the model have converged in their behaviour, thus forming different symmetry groups. In addition, an example of potential function shifts is shown for

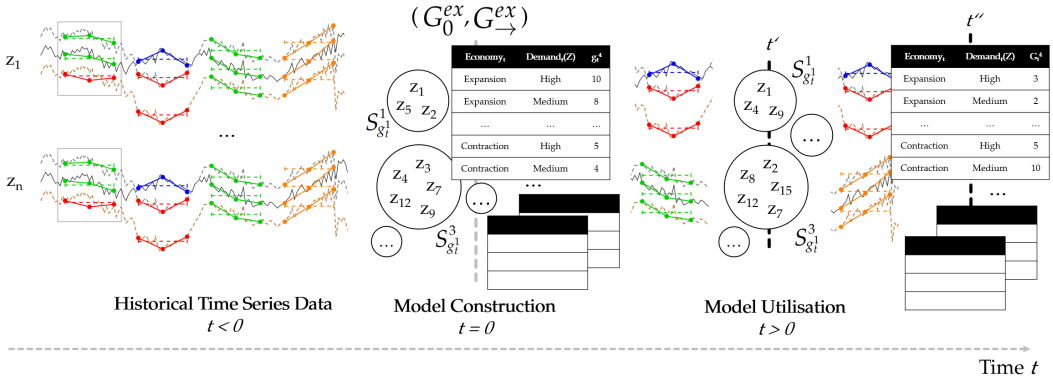


Figure 10.1.: Extended overview of the process steps: Learning symmetry clusters using MOP$_4$SA, preventing groundings using SA$_4$PG and change point detection for detecting environmental changes.

the time step t'', where new potential describing the likelihood for events to occur would emerge if new potential functions were to be determined.

In the last part of this work, we introduce two change point detection strategies to cover non-stationarity in terms of (i) changing probability distributions, also known as concept drifts, and (ii) changing symmetry structures. More specifically on item (i), as we introduced DPRMs with dynamic domains, the size of the sets of domain objects determines the total joint distribution encoded in a DPRMs. Consequently, addressing changing probability distributions to account for non-stationarity, particularly in terms of dynamic domains, is of significant importance.

Changing Probability Distributions

With changing object behaviour, changing domain sizes or other unpredictable circumstances, the probability distributions encoded in a DPRM are affected. To this end, we introduce a concept drift detection approach that uses evidence and the compact encoding of DPRMs in terms of factorisation and relations. In particular, a piece-wise analysis of the potential functions, which yield a probability distribution when count-normalised, is performed on the parfactors encoded within the model. The detection of concept deviations is thus not performed on the full joint distribution of the model in order to reduce the complexity of the approach.

Changing Symmetries

To understand object behaviour and construct symmetry structures in DPRMs a priori, we have introduced MOP$_4$SA in the previous part of this work. Since object behaviour and thus

also symmetry structures in real-world applications are often not static in time, symmetry structures have to be re-learned at some point. One could run MOP$_4$SA a second time to get a new structure for each time step, but due to the size of the model in terms of the domain objects represented in it, this causes a lot of overhead and thus requires a more efficient method for this task. To this end, we introduce *multivariate ordinal patterns for symmetry change detection*, MOP$_4$SCD for short, to keep track of how long symmetry clusters remain valid and when re-learning of a model is required. To be efficient, MOP$_4$SCD is based on an intermediate step of MOP$_4$SA, in particular the similarity graph.

As depicted in Fig. 10.1, we present change point detection approaches that run after the time of model construction, i.e., for $t > 0$. This is the detection of new symmetry clusters at a time t' or also the detection of new potential functions at a time t'' prior to answering queries at G_t to ensure an underlying model that represents reality well. Chapter 11 follows with an approach to detect concept drifts, before we present an approach to detect changes in the symmetry structure in Chapter 12.

11 Concept Drift Detection in Factorised Models

DPRMs factorise a full joint distribution into multiple factors by exploiting (in-)dependencies among randvars in a model. More specifically, in DPRMs its parfactors encode potential functions which, when combined with all other parfactors in a DPRM, give a joint potential function. This joint potential function, after count normalisation, determines the full joint distribution. Further, DPRMs are based on the first-order Markov assumption, i.e., randvars from time slice t depend only on randvars from the previous time slice $t-1$. More specifically, DPRMs model a stationary process, i.e., the overall joint distribution of a DPRM is not dependent on time. However, the assumption of a stationary underlying process frequently conflicts with the real situations encountered in practice. At a parfactor level, this corresponds to the fact that potential functions at one time step t are no longer good representatives at another future time step $t' > t$ (see Fig. 10.1), i.e., the potentials that implicitly describe the probabilities for the occurrence of individual events are no longer suitable predictors at time t', a concept that is known as *concept drifts* in probability theory. This leads to challenges as predictions in query answering become less accurate over time. However, in order to still ensure the accuracy of predictions, it is essential to detect and handle concept drifts in terms of changing potential functions as highlighted before.

As non-stationarity highly limits DPRMs for real-world use, we propose a universal approach to detect concept drifts for different kinds of relational models. In this chapter, we present a three-step process for handling concept drifts in DPRMs. First, we present an approach to derive and measure the deviation from an original dynamic factorised model. Second, we introduce an approach to detect points in time when a concept drift has settled. Third, we extend our proposed online variant of LDJT with concept drift detection The contents of this chapter are mainly published in the following conference paper:

> Nils Finke, Tanya Braun, Marcel Gehrke, and Ralf Möller. Concept Drift Detection in Dynamic Probabilistic Relational Models. *The International FLAIRS Conference Proceedings*, 34, 2021.

In **Section 11.1**, we first present an approach to concept drift detection which exploits the factorisation of the model, followed by an introduction of the concept drift detection approach. Both subsections provide contribution 4a of this work. We conclude this chapter in **Section 11.2** with a runtime analysis and an empirical evaluation, verifying whether the

presented approach is able to properly detect concept drifts.

11.1 Concept Drift Detection

In general, it is assumed that events associated with the randvars of the model, i.e., specific observations for a randvars for a specific domain object, occur according to the potentials specified by the potential functions in the parfactors. More specifically, after count-normalisation, the potential functions of a parfactor specify the likelihood of the occurrence of events, i.e., also the frequency the observations are expected to occur. We use this relationship to detect concept drift by checking over time whether events occur according to the probability given by the potential function of a parfactor. As events are encoded in evidence parfactors as progressing in time, the likelihood of these events occurring can be determined by examining the corresponding evidence parfactors. Before introducing concept drift detection more formally, we outline concept drift detection. Concept drift detection is performed in two steps: (i) Determine an ongoing *alteration* factor over time t between the initial model at G_0 prior to setting any evidence and the current model at G_t afflicted with evidence, and then (ii) detect a *concept drift* in the sequence of alteration factors. We give the following example for one parfactor.

Example 11.1.1 (Alteration Factor)**.** *Let's consider the parfactor g_t^4 connecting PRVs $Economy_t$ and $Demand_t(Z)$ with $\mathcal{D}(Z) = \{z_1, z_2, z_3\}$ and g_0^4 at time step $t = 0$ at which no evidence is yet encoded as per the following table*

$Economy_0$	$Demand_0(Z)$	g_0^4
Expansion	*high*	10
Expansion	*medium*	2
Expansion	*low*	2
Contraction	*high*	2
Contraction	*medium*	2
Contraction	*low*	2

and evidence $\mathbf{E}_0 = \{Economy_0 = Expansion, Demand_0(z_1) = high, Demand_0(z_2) = medium, Demand_0(z_3) = low\}$ resulting in the following evidence parfactors replacing g_0^4 with $g_0^{4,1}$, $g_0^{4,2}$, and $g_0^{4,3}$ accordingly, i.e.,

$Economy_0$	$Demand_0(z_1)$	$g_0^{4,1}$
Expansion	*high*	10
Expansion	*medium*	0
Expansion	*low*	0
Contraction	*high*	0
Contraction	*medium*	0
Contraction	*low*	0

$Economy_0$	$Demand_0(z_2)$	$g_0^{4,2}$
Expansion	*high*	0
Expansion	*medium*	2
Expansion	*low*	0
Contraction	*high*	0
Contraction	*medium*	0
Contraction	*low*	0

$Economy_0$	$Demand_0(z_3)$	$g_0^{4,3}$
Expansion	*high*	0
Expansion	*medium*	0
Expansion	*low*	2
Contraction	*high*	0
Contraction	*medium*	0
Contraction	*low*	0

Then, events in $\mathbf{E}_0$ are expected to occur according to the likelihood specified by g_0^4, i.e., events $Demand_0(Z) = high$ have a higher likelihood occurring in case $Econamy_0 = Expansion$ than

other events occurring. To determine an alteration factor between the initial model state and the current model state, we aggregate evidence parfactors back into one by summing up potentials of matching range values. In this example, this leads to an aggregated evidence parfactor $g_0^{4\prime}$ as follows

$Economy_0$	$Demand_0(Z)$	$g_0^{4\prime}$
Expansion	*high*	10
Expansion	*medium*	2
Expansion	*low*	2
Contraction	*high*	0
Contraction	*medium*	0
Contraction	*low*	0

After count-normalisation of both $g_0^{4\prime}$ and g_0^4 the resulting probability distributions can be used for determining a simple KLD between both. The KLD is then considered as the alteration factor for the parfactor g_0^4 at the time step $t = 0$. In this example, KLD is

$$\begin{aligned} D_{KL} &= (0.714 \cdot \log\left(\frac{0.714}{0.5}\right)) + (0.142 \cdot \log\left(\frac{0.142}{0.1}\right)) + \dots \\ &\approx 0.352. \end{aligned} \tag{11.1.1}$$

The values used here as an example for the calculation of the KLD are the normalised potential values of $\phi_0^{4\prime}$ and ϕ_0^4. A KLD, i.e., our alteration factor, close to 0 in our context for detecting concept drifts implies that the events occur according to the likelihood of the initial model probability distribution. Given a model with multiple parfactors, this alteration factor is similarly calculated for each parfactor and summed up to yield an alteration factor for the entire model. Over time t, we yield a sequence of alteration factors used for concept drift detecting.

Of course, the above example pertains to a model with a relatively small domain, resulting in a limited set of possible events. Consequently, the alteration factor may not be significant enough in such cases. However, on a larger scale, events are expected to occur according to the distribution specified in the initial model state. If this is not the case, the distribution(s) no (longer) describe reality (by means of events w.r.t. the randvars of the model occurring) good enough and a concept drift might have occurred. Algorithm 7 outlines the calculation of an alteration factor for a local model G_t at time step t. Note that in the previous example, we have summed up the potentials of all evidence parfactors before calculating the alteration factor. This step is not part of the algorithm mentioned in Algorithm 7, but alteration factors are calculated as a distance between every evidence parfactor and the original parfactor as per the template model individually and then combined into a common alteration factor.

We assume that a concept drift manifests itself only across multiple time steps, i.e., a random incident has occurred, changing the environment to the point that the model no longer accurately describes the environment. Such an incident may occur over several time steps, slowly building up to a point where the environment has changed significantly. A

Algorithm 7 Compute alteration-factor among parfactors of a local model G_t.

procedure GETALTERATION(Template Model G_0, Local Model G_t)**:**
  $a \leftarrow 0$
  for each parfactor $g_t^i \in G_t$ **do**
    Find corresponding parfactor $g_t^{i\prime}$ in G_0
    Normalise distributions of $g_t^i, g_t^{i\prime}$
    $a = a + D_{KL}(g_t^i, g_t^{i\prime})$
  return a

prominent example is the Corona pandemic outbreak. At the beginning of the pandemic, the market environment was little affected, but as the number of infections increased, uncertainty grew and settled again until people, the market etc. learned to better cope with the new circumstances. For this reason, specifically, as a temporal model includes the past and a changing event can still persist, evidence will not immediately reveal that the original model no longer accurately describes the environment. Further, altering events may only be temporary and therefore may be ignored in concept drift detection. Therefore, we monitor an increasing alteration factor until it plateaus. More specifically, we aim at identifying a point in time at which a concept drift has settled. Consider Fig. 11.2 (a) for an example based on a sequence A of alteration-factors a towards a drift (blue dots). An alteration-factor $a = 0$ indicates that both models are equal. Ignore all other lines and subplots for now. We aim at finding a new partial sequence of alteration factors of in-average the same distance to the initial distribution, denoted as a *plateau*, such as the sequence starting at $t > 70$ (blue line). During $0 < t < 70$, the drift is still ongoing. We next elaborate on how to identify such plateaus in the sequence of alteration factors.

Detecting Points in Time a Concept Drift has Settled

Using a sequence A of alteration factors a, we detect a concept drift by performing curve fitting to find plateaus in the sequence. We assume that the change in alteration factors follows the pattern of a logistic function. The logistic function starts with a phase of slow growth, followed by a phase of rapid growth, and finally, a phase of slowing growth, approaching a limit known as *carrying capacity*. Once the carrying capacity is reached, a plateau in the alteration factors is found, which we define as the point in time once a concept drift has settled. In order to find sequences of alteration factors that follow the pattern of a logistic function using the curve fitting approach, however, there must be a benchmark that provides information about whether a fitted curve also represents this pattern *sufficiently well*. To determine whether the coefficients of the logistic function were determined sufficiently well, we make use of the mean-squared error which is minimised in curve fitting to determine the coefficients of any function. More specifically, in curve fitting, coefficients of any

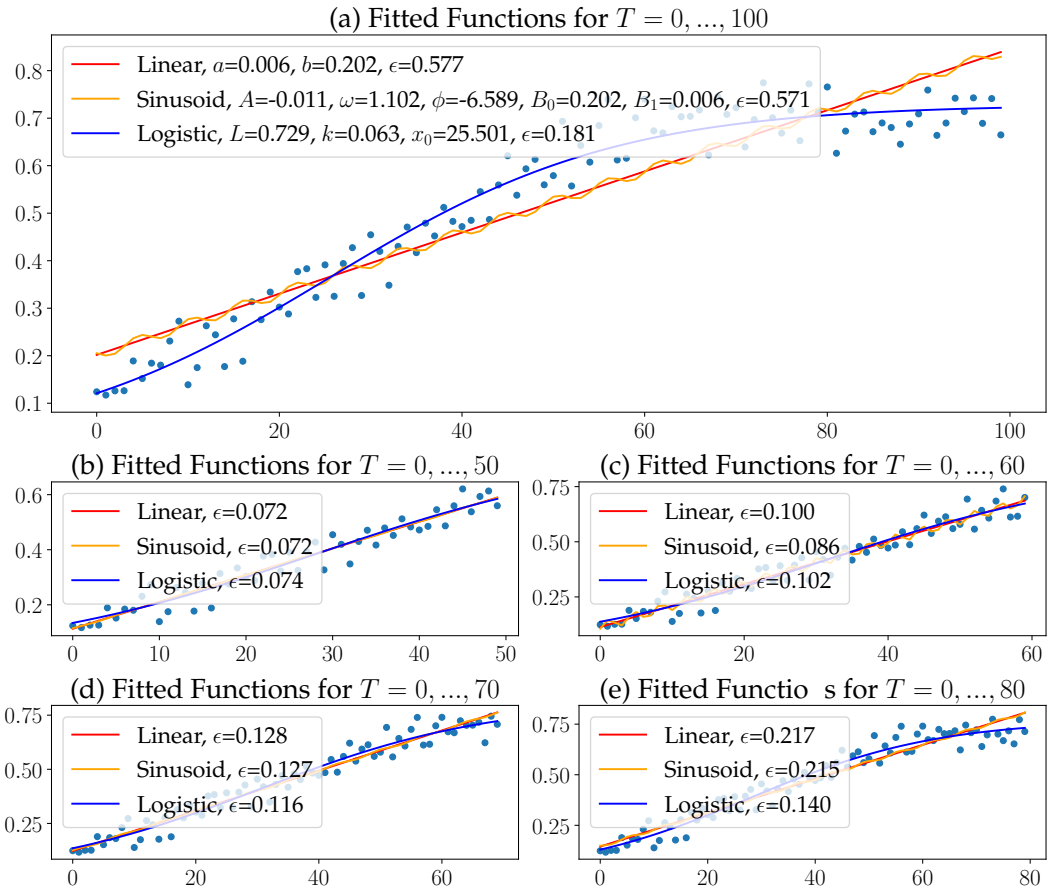


Figure 11.2.: Curve fitting on a sequence of alteration factors A (blue dots) with linear (red line), sinusoid (yellow line) and logistic (blue line) function.

generic function $F(x)$ are determined such that a mean squared error

$$\epsilon = \sum_{i=1}^{S} (F(x_i) - S_i)^2 \tag{11.1.2}$$

is minimised. To determine whether the error ϵ is small enough to conclude that a logistic function, and thus concept drift, has been found, we need to compare it with other functions used as benchmarks. More specifically, we compare against a set of functions, namely, a linear, a sinusoid, and a logistic function. Note, that except for the common definitions of the named functions, we describe the sinusoid function as a combination of a sinusoid function and a linear function to encode upward or downward behaviour. Therefore the sinusoid

Algorithm 8 Concept drift detection through curve fitting

```
static Queue F ← () of length ℓ
procedure DETECTDRIFT(Sequence of alterations A, Sequence Length ℓ):
    E ← ()
    for each f in {linear, sinusoid, logistic} do
        Perform curve fitting for f on A
        Get ϵ error between f and A
        E ← E ∘ ϵ
    Get f according to min(E)
    if F full then
        Dequeue first element of F
    Enqueue f in F
    if ∀f ∈ F[0, ℓ − 1] : f = logistic then
        return true
    return false
```

function is defined by

$$f(x) = A \cdot sin(\omega \cdot x + \phi) + (b_0 + b_1 \cdot x) \tag{11.1.3}$$

with A as the mean level, ω the frequency, ϕ the phase, b_0 as the y-intercept and b_1 the overall slope. Performing curve fitting with the other two non-logistic functions serves the following purpose:

- Fitting to a linear function is intended to reveal whether the concept drift is still currently evolving and has not yet settled. That is, ϵ as the result of a curve fitting for a linear and a logistic function will be smaller for a linear function than for the logistic function as long as the alteration factors continue to increase over time.
- Fitting a sinusoid function is used to detect fluctuations in the alteration factors, i.e., random upward and downward movements, potentially also increasing over time.

Generally, if ϵ for the other fitted functions is smaller than the ϵ for the fitted logistic function, a concept drift has not yet been detected. Algorithm 8 outlines drift detection given a sequence of alteration-factors A and based on a limit ℓ, that is, a threshold provided a priori. Once for ℓ consecutive time steps still a logistic function is fitted, a concept drift has been identified. We give the following example.

Example 11.1.2 (Curve Fitting for Concept Drift Detection)**.** *Figure 11.2 (a) depicts the fitted functions on A. Within the algorithm, linear and sinusoid functions are used as exclusion criteria. Based on the least square error ϵ, we select the function describing the data the best. Figure 11.2 (b-e) represents how ϵ evolves with getting more data (here $T = \{50, 60, ..., 80\}$). With $T > 70$ and $\epsilon = 0.116$, the carrying capacity of the logistic function is reached, and a concept drift can be identified. Still, with $T < 70$ a sinusoid function fits better to the data. Thus, the sinusoid and the linear function work serving as exclusion criteria for the point in time at which no concept drift has*

settled yet. A linear function with slope $a > 0$ fits better to data points in case no plateau at the end of the sequence A can be identified. Still, the KL divergences increase over time. It can therefore be assumed that the system is still in a drift that has not settled down. On the other hand, a sinusoid function fits better to data points, in case the KL divergence periodically increases and decreases. A general upward trend with sinusoidal abnormalities in Fig. 11.2 (a) is apparent. As long as $T < 70$, a linear and a sinusoid function describes the data points better than a logistic function. Now, once with $T > 70$ for ℓ consecutive time steps the fitting logistic function had the smallest ϵ compared to the linear and sinusoid function, a concept drift is identified.

Next, we incorporate concept drift detection into online query answering with LDJT.

Online Query Answering under Concept Drift

As follows, we present an online version of LDJT that includes concept drift detection. As our approach to drift detection is not limited to DPRMs and LDJT only, before turning to LDJT as a specific algorithm, we look at the general case. The approach for any type of temporal probabilistic model where query answering is performed with new evidence as part of the queries (see Definition 3.2.1) is defined as follows:

(1) While moving forward in time, collect and encode current evidence.

(2) Perform drift detection using Algorithm 7 and Algorithm 8.

(3) Using the detection result: (a) If a drift is detected: Determine new distributions for the model given the new evidence. (b) Otherwise: Continue.

(4) Answer current queries and go back to (1).

Turning to LDJT, a main advantage becomes apparent: The point where LDJT condenses all past information is during the calculation of the α message. Thus, instead of considering the complete model, i.e., the joint distribution of a model G_t, we only need to consider the parfactors in α_t and compare these to the parfactors that occur in α_0, the α message that would be calculated when transitioning from $t = 0$ to $t = 1$ without any evidence. The α_t only contain evidence for PRVs that are connected through interslice parfactors to subsequent time steps. Algorithm 9 represents an outline of an online version of LDJT with concept drift detection, which is based on the online version of LDJT as outlined in Section 3.3 and Algorithm 2. Before instantiating the FO jtree for the current time step, LDJT$^{\text{drift}}$ performs concept drift detection using Algorithm 7 and Algorithm 8. If a drift is detected, a learning algorithm is called to learn new parameters, updates the FO jtrees, and resets the current time t to 0. Since drift detection works with a new model, the queue in Algorithm 8 is reset as well.

Algorithm 9 Online LDJT algorithm with concept drift detection

Input: $(G_0, G_{\rightarrow})$ DPRM, $\mathcal{E}$, $\mathcal{Q}$ streams
Construct FO jtrees $J_0, J_{\rightarrow}$
Sequence of alterations $A \leftarrow ()$
for $t = 0, 1, ...$ **do**
  Get evidence E_t from evidence stream $\mathcal{E}$
  $S \leftarrow S \cdot$ GETALTERATION(α_0, α_{t-1}) ▷ see Algorithm 7
  if DETECTDRIFT(A) ▷ see Algorithm 8
  then
    Learn new parameters for $G_0, G_{\rightarrow}$
    Update $J_0, J_{\rightarrow}$
    $t \leftarrow 0$
    Reset $\mathcal{F}$ in detectDrift(A)
  Instantiate FO jtree J_t from $J_0, J_{\rightarrow}$
  Add α_{t-1} (if $t \neq 0$) and E_t
  Pass messages in J_t
  Answer queries Q_t from query stream $\mathcal{Q}$ on J_t
  Calculate α_t

11.2 Runtime Analysis and Empirical Evaluation

Due to lack of available labeled data for concept drifts, this evaluation is twofold. We first make a theoretical runtime analysis followed by an empirical evaluation on artificial data.

Runtime Analysis

Comparing two models based on their full joint probability distribution has a runtime complexity that is exponential in the number of entities represented by the model, i.e., $O(r^n)$ with $r = \max_{R \in rv(G')} |\mathcal{R}(R)|$, G' referring to the current model, and $n = |gr(rv(G'))|$. In a relational model, we can use a lifted representation of the full joint probability distribution, i.e., compute $\frac{1}{Z} \prod_{g \in G'} g$. Using a lifted representation turns the concept drift detection problem into a tractable one in the sense that the domain sizes do not appear in the exponent n, implying that $n = |rv(G')| \ll |gr(rv(G'))|$ if the domain sizes dominate compared to the number of PRVs. Switching to comparing two models on the local level has the effect that the complexity goes down to $O(|G'| \cdot r^l)$ with $r = \max_{R \in rv(G')} |\mathcal{R}(R)|$ as before and $l = max_{g=\phi(\mathcal{A})_{|C} \in G'} |\mathcal{A}|$. The expression r^l denotes an upper bound on the largest size among the parfactors in G'. In LDJT$^{\text{drift}}$, G' refers to the current α message α_t, which in turn implies a reduction, namely, $O(|\alpha_t| \cdot r^l)$ with $r = \max_{R \in rv(\alpha_t)} |\mathcal{R}(R)|$ and $l = max_{g=\phi(\mathcal{A})_{|C} \in \alpha_t} |\mathcal{A}|$ with G' replaced by α_t, where $|\alpha_t| \ll |G'|$. The components r and l are either the same or smaller switching from G' to α_t as $|rv(\alpha_t)| \ll |rv(G')|$, meaning that PRVs with large range sizes or parfactors with many arguments might not be a part of α_t, again having a positive effect on r or l.

For drift detection, the runtime depends on the curve fitting used (O_{curve}; times 3, which is constant) and ℓ, which leads to a complexity of $O(\ell) + O_{curve}$, being independent of the model characteristics. Overall, the additional runtime is in $O(|G'| \cdot r^l) + O(\ell) + O_{curve}$ with $G' = \alpha_t$ if using LDJT, which is usually negligible compared to the runtime complexity of online query answering in DPRMs.

Empirical Evaluation

Due to the lack of available labelled data for concept drifts, this evaluation is partly done on artificial data. We consider historical vessel movements based on AIS data similar to an evaluation in the two preceding parts of this work. Also see Appendix A.1 for preprocessing steps. Based on processed AIS data for the year 2020 for the Baltic Sea, we set up a DPRM with PRVs for sub-challenge 1 as introduced in Chapter 2. For the evaluation, we derive potential functions for parfactors g_t^2 and g_t^3 by counting observations, i.e., observations for supply and demand in the different zones and according to that vessel positions (counting different model states). The potential functions are determined based on data for the month of January to March. The remaining data is left for testing the concept drift detection approach. To be able to validate the accuracy of the concept drift detection approach, labels on data, denoting whether a drift is present or not, are necessary. Therefore, we make use of real-world data as described before, but additionally (randomly) generate concept drifts by manipulating available evidence data. We manipulate available evidence data in a sense, that whenever we introduce a concept drift, we make sure that the drift remains over the whole evidence sequence. Noise, i.e., data without any concept drifts, serves to check whether the algorithm is able to distinguish between data with and without concept drifts. With this approach, we ensure on the one hand that the algorithm is fed with realistic data patterns, but on the other hand, we control the occurrence of concept drifts and therefore are able to verify if a concept drift was detected by our approach at the correct point in time.

For evaluation, we consider accuracy results as well as the runtime performance of the concept drift detection approach. We unroll the DPRM for $T = 12$ time steps, i.e., for three further months and apply evidence, i.e. events that are in the test dataset. We test concept drift detection with a detection limit $\ell = 9$ on the data for 392 zones separately on distributions denoted by the parfactors g_t^2 and g_t^3, thus perform 784 experiments in total. More specifically, we test concept drift detection for each zone on each parfactor, i.e., one experiment for g_t^2 and one g_t^3 per zone. We generate concept drifts within events as described before. We generate *slow* and *abrupt* concept drifts. A slow drift is denoted by an increasing alteration factor a until a (the drift) settles, while an abrupt drift immediately settles from the current time step t to the next $t + 1$. Section 11.2 represents results for the scenarios described above. For the cases of a successful concept drift detection, we look at the accuracy *acc* (number of successful detections) and the average statistical error $\varnothing\epsilon$ for the underlying functions used in curve fitting. Finally, we put these results in context with the cases where

	Drift detected			Drift not detected		
Data set	*acc*	$\varnothing\epsilon_{logistic}$	$\varnothing\epsilon_{other}$	$\varnothing\epsilon_{logistic}$	$\varnothing\epsilon_{other}$	$\varnothing$runtime/t
Slow Drift	0.843	0.875	3.833	0.299	0.282	8.62 ms
Abrupt Drift	0.921	0.768	4.035	0.065	0.062	6.77 ms
Noise	0.138	0.031	0.032	0.064	0.061	7.81 ms
Combined	0.903	0.745	3.542	0.100	0.095	8.35 ms

Table 11.1.: Accuracy results for concept drift detection algorithm (with detection limit $\ell = 9$) on sequences with length $T = 12$ including randomly generated concept drifts (in evidence) and on noise (as validation and exclusion criteria).

the algorithm failed to detect a drift.

From the result it can be seen that the algorithm detects abrupt drifts (accuracy of 0.921) better than slow drifts (accuracy of 0.843). In case of successful detections, we see for both abrupt and slow drifts, that the error measure ϵ (average mean squared error in curve fitting) is much smaller than for the sinusoid and linear exclusion functions ($\varnothing\epsilon_{logistic} < 0.9$ and $3.8 < \varnothing\epsilon_{other} < 4.1$), whilst for unsuccessful detections, ϵ does not vary much (for slow drifts the difference in the error is 0.017, and for abrupt drifts 0.003). Thus, for false detections, our approach only needs smaller improvements to be more exact. Besides testing with slow and abrupt drifts, we also test the approach on data without any drifts (noise only). In 13.8% a concept drift was found in sequences without any drift (false classification). When considering error ϵ, the average error for a false classification is only 0.001 smaller than for correct detection (difference between $\varnothing\epsilon_{logistic}$ and $\varnothing\epsilon_{other}$). Thus, in some cases, it is hard for the algorithm to differentiate. Lastly, we have performed a test combining all of the above scenarios with an overall accuracy of $\sim 90\%$.

Besides accuracy, we have measured the average runtimes of the concept drift detection itself. For every time step t and with evidence, we keep on building the alteration sequence A and perform drift detection involving curve fitting for the set of defined functions, thus, three curve fitting runs per time step t. The average runtime per time step t amounts to ~ 8.3 milliseconds. Run times vary depending on the data set as curve fitting sometimes determines optimal parameters quicker than in other runs. In general, the concept drift detection algorithm does not affect runtime too heavily. Thus, the empirical evaluation supports that our approach works well for the detection of slow and abrupt drifts while preserving runtime performance in inference.

In the next chapter, we deal with changing symmetry structures, i.e., objects that develop different similarities towards other domain objects over time.

12 Symmetry Change Detection in Relational Models

In the second part of this work, we introduced MOP_4SA, an algorithm for determining symmetries across domain objects of a DPRM. The output of the algorithm are so-called symmetry clusters, which represent groups of domain objects with symmetrical behaviour. We use these symmetry clusters to derive, for example, intrinsic defaults for unobserved domain objects. This helps avoid groundings due to missing or conflicting events for domain objects within a cluster, an approach called SA_4PG. We have shown that preventing groundings applied a priori and implemented as part of LDJT, helps preserve the runtime benefits in lifted inference. However, the effectiveness of SA_4PG is linked to the validity of the symmetry clusters themselves. The validity of the symmetry clusters can change over time when the behaviour of domain objects change and different objects converge in their behaviour and create new, different symmetry clusters.

For this reason, we introduce *multivariate ordinal patterns for symmetry change detection*, MOP_4SCD for short, an algorithm for detecting changes in symmetry clusters. The algorithm is based on the similarity graph, an intermediate step of MOP_4SA. By using MOP_4SCD, points in time are determined at which new symmetry clusters must be determined. The content of this chapter is mainly published in the following journal paper:

> Nils Finke and Ralf Möller. On the Approximation of Symmetries and Related Structural Changes in Dynamic Probabilistic Relational Models. *Advances in Science, Technology and Engineering Systems Journal*, 7(2), 2022.

In **Section 12.1** we formally introduce MOP_4SCD and thus provide contribution 4b. We then evaluate the effectiveness of MOP_4SCD in **Section 12.2** by extending the evaluation of MOP_4SA and SA_4PG from Section 8.3.

12.1 Multivariate Ordinal Patterns for Symmetry Change Detection (MOP_4SCD)

In temporal models, symmetries and thus derived symmetry clusters can change over time, as already illustrated in the motivation for this work in Section 2.3, especially in Fig. 2.8.

For reference, the graph depicts that the amount of supply in the Baltic Sea Region zones for the year 2020 changes over time. In the early months, supply is higher in northern areas and shifts to southern regions as the year progresses. Accordingly, once symmetry clusters have been learned, they may only remain valid for a certain period of time. In particular, the validity of the individual symmetry clusters varies. Some symmetry clusters are valid for a longer period of time, others for a shorter period.

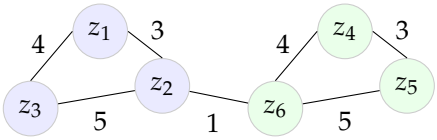
The idea of MOP$_4$SCD is to identify the points in time when relearning symmetry clusters is reasonable. For this purpose, we use the **similarity graph** determined as an intermediate step in MOP$_4$SA and check whether the similarity graph has changed (i) in an *unbalanced manner* and (ii) in a *significant way* . Specifically, we continue to run MOP$_4$SA within LDJT during the process of query answering over time. Thus, the similarity graph is updated at each time step, reflecting the changes in the similarity score between domain objects as new information becomes known, i.e., new observations are made for randvars of the model. However, instead of continuously relearning symmetry clusters, we suppress this learning process in MOP$_4$SA after its initial execution and allow it to resume only when we observe a *significant and unbalanced change* in the similarity graph indicated by MOP$_4$SCD. In the following, we introduce MOP$_4$SCD by discussing when the similarity graph changes in an unbalanced manner and when this change is significant.

(i) When does the similarity graph change in an *unbalanced manner?*

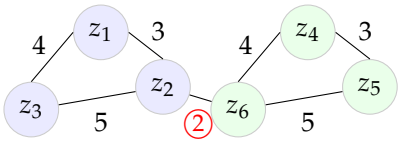
It can be assumed that a symmetry cluster is no longer valid if objects within the cluster no longer have the same similarity to its cluster objects as in the previous time steps. Therefore, MOP$_4$SCD is based on the similarity graph for two consecutive time steps, because its similarity counts would no longer scale proportionally over time when the objects no longer behave similarly. To illustrate this, Fig. 12.1a represents a similarity graph based on which two clusters were identified. Nodes $S^1 = \{z_1, z_2, z_3\}$ (coloured in blue) refer to cluster S^1 and the nodes $S^2 = \{z_4, z_5, z_6\}$ (coloured in green) refer to cluster S^2. Both clusters are connected by nodes z_2 and z_6 since similarity was measured for both at a time step before cluster learning. Relearning clusters becomes necessary if the cluster structure itself changes. This happens either if

(a) similarities between objects of different clusters change, e.g., if the similarity between z_2 and z_6 increases (see example in Fig. 12.1b) and might require merging the clusters or even splitting them into more than two clusters (*unbalanced intercluster change*), or

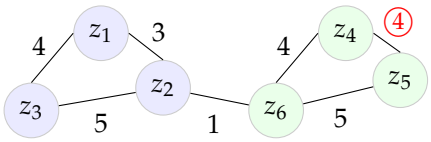
(b) similarities within a cluster change disproportionately, e.g., if similarities for S^2 change only for a subset of the objects such as for z_4 and z_5 but not proportionally for all objects such as z_4 and z_6 and z_6 and z_5 (see example in Fig. 12.1c) requiring to split the cluster even further (*unbalanced intracluster change*).



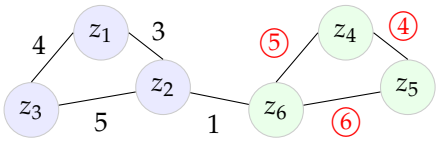
(a) Symmetry graph with two symmetry clusters, one coloured in blue, and one coloured in green, at a time step t.



(b) Example for an **unbalanced intercluster change** in the symmetry graph between domain objects z_2 and z_6 at a consecutive time step $t = t+1$ (compared to the symmetry graph depicted in Fig. 12.1a).



(c) Example for an **unbalanced intracluster change** in the symmetry graph between domain objects z_4 and z_5 at a consecutive time step $t = t+1$ (compared to the symmetry graph depicted in Fig. 12.1a).



(d) Example for **balanced change** in the symmetry graph as for all domain objects z_4, z_5, and z_6 the similarity score increased by 1 at a consecutive time step $t = t+1$ (compared to the symmetry graph depicted in Fig. 12.1a).

Figure 12.1.: Overview of potential unbalanced changes in a similarity graph. No edge is drawn between nodes in case the similarity score between the domain object pairs is zero. Edge weights highlighted in red represent unbalanced and balanced changes.

(ii) When is the change of a similarity graph *significant*?

To measure the change in the similarity graphs between time points, a distance measure is used. If the distance measure is above a set threshold, the change can be considered to be significant and thus to use these points as change points that trigger the process of cluster relearning. In the following, we define both intercluster and intracluster change measures and combine both into a distance measure that indicates the overall change between consecutive time steps.

To verify if domain objects from two different disjoint symmetry clusters do not start to behave similarly in terms of the similarity graph, the weights between all object pairs from different clusters should not increase over time. Let $\mathcal{W}^t$ and $\mathcal{W}^{t+1}$ be the similarity graphs from the current to the next time step, and $\mathcal{S}$ the symmetry clusters at time t.

Definition 12.1.1 (Intercluster change measure)**.** The intercluster change measure is

defined as

$$d_{inter}(\mathcal{W}^t, \mathcal{W}^{t+1}, \mathcal{S}) = \sum_{\substack{S^i \in \mathcal{S} \\ S^j = en(S^i) \cap en(\mathcal{S})}} \frac{\sum_{i \in en(S^i)} \left[w_{ij}^{t+1} = w_{ij}^t + 1 \right]}{\substack{j \in en(S^j) \\ |en(S^i)| \cdot |en(S^j)|}}, \tag{12.1.1}$$

where $[x] = 1$ if x and, 0 otherwise for $en(S^i) \cap en(S^j) = \emptyset$.

Simply speaking, d_{inter} counts the number of increases in weights across different clusters S^i and S^j (as depicted in Fig. 12.1b). The resulting count is normalised by dividing through the number of comparisons between object pairs of the clusters $en(S^i)$ and $en(S^j)$. This results in a measure between 0 and 1, where a value close to 1 denotes maximum dissimilarity between similarity graphs $\mathcal{W}^t$ and $\mathcal{W}^{t+1}$.

Similarly, to validate if domain objects within a cluster continue behaving the same, in terms of the similarity graph, the weights between all domain object pairs within the same symmetry cluster must also increase proportionally in an evenly distributed manner (see example in Fig. 12.1d). If the weights do not increase, the objects are most likely no longer behave similarly.

Definition 12.1.2 (Intraclusteral change measure)**.** The intracluster change measure is defined as

$$d_{intra}(\mathcal{W}^t, \mathcal{W}^{t+1}, \mathcal{S}) = \sum_{S^i \in \mathcal{S}} \frac{\sum_{i,j \in en(S^i), i<j} \left[w_{ij}^{t+1} - w_{ij}^t = 0 \right]}{|en(S^i)| \cdot |en(S^i)|}, \tag{12.1.2}$$

where $[x] = 1$ if x and, 0 otherwise.

Simply put, d_{intra} counts the occurrence of no weight increases within pairs of objects of a similar cluster S^i before they are normalised (as depicted in Fig. 12.1d). Again, this gives a measure between 0 and 1, with a value close to 1 denoting maximum dissimilarity between similarity graphs $\mathcal{W}^t$ and $\mathcal{W}^{t+1}$.

The idea of measuring inter- and intracluster changes is essentially based on the assumption that, due to unbalanced changes, entities may suddenly become candidates for inclusion in a unified symmetry cluster, or conversely, existing clusters may need to be subdivided into seperate clusters. Finally, since both inter- and intracluster changes of the similarity graph should trigger a relearning of the symmetry clusters, both d_{inter} and d_{intra} are combined into one overall measure.

Algorithm 10 Symmetry change detection (MOP$_4$SCD)

procedure DETECTSYMMETRYCHANGE(Similarity Matrix $\mathcal{W}^t$, Similarity Matrix $\mathcal{W}^{t+1}$, Symmetry Clusters $\mathcal{S}$, Similarity threshold b)**:**
  $s \leftarrow d(\mathcal{W}^t, \mathcal{W}^{t+1}, \mathcal{S})$ ▷ see Eq. (12.1.3)
  if $s > b$ **then**
    return true
  return false

Definition 12.1.3 (Change measure)**.** The change measure is defined as

$$d(\mathcal{W}^t, \mathcal{W}^{t+1}, \mathcal{S}) = \frac{d_{inter} + d_{intra}}{|\mathcal{S}|}. \tag{12.1.3}$$

Now the change measurements can be used in combination with a threshold $b \in \mathbb{N}_{>0}$ to determine if the change in the similarity plot is significant.

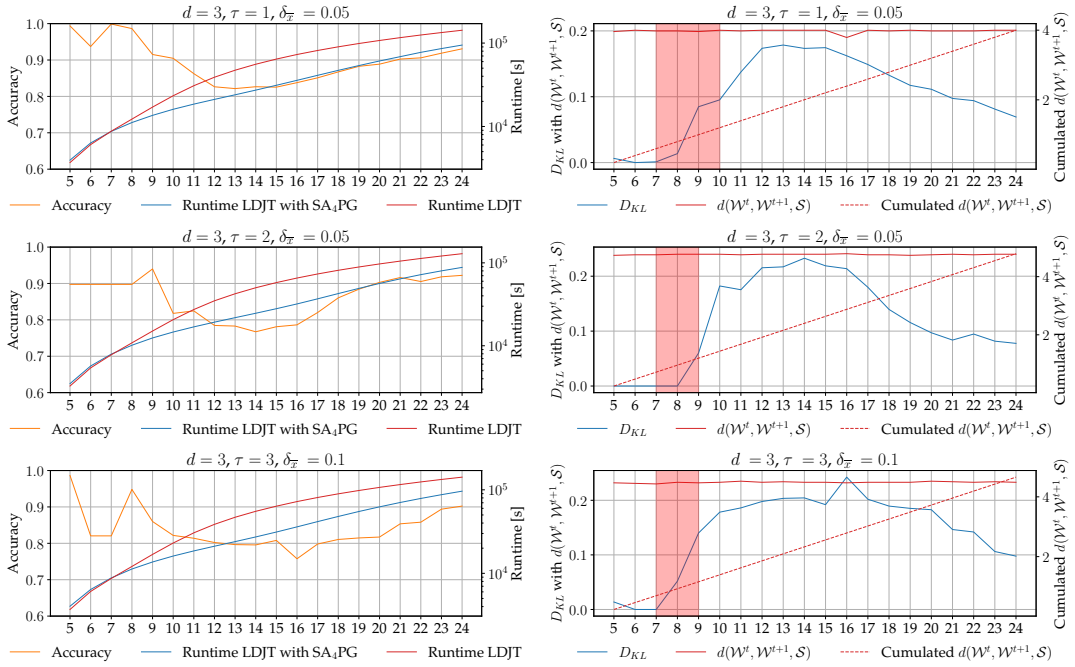
- If $d(\mathcal{W}^t, \mathcal{W}^{t+1}, \mathcal{S}) = 0$, the change in the similarity graph is balanced.
- If $d(\mathcal{W}^t, \mathcal{W}^{t+1}, \mathcal{S}) > 0$, there exist unbalanced changes in the similarity graph.
- If $d(\mathcal{W}^t, \mathcal{W}^{t+1}, \mathcal{S}) > b$, it may be worthwhile to (re)perform clustering and (re)build symmetry clusters.

Algorithm 10 outlines the overall procedure of change detection in and between symmetry clusters, which we denote as MOP$_4$SCD as follows. For change detection as part of LDJT and SA$_4$PG, MOP$_4$SCD is then integrated into SA$_4$PG (Algorithm 6) as an initial step to validate symmetry clusters. If Algorithm 10 returns the value *true*, the symmetry clusters have to be relearned, i.e., MOP$_4$SA (Algorithm 5) is re-run.

12.2 MOP$_4$SCD in Application

In this section, we evaluate MOP$_4$SCD based on clusters determined by MOP$_4$SA under the same conditions as in the experiments performed in Section 8.3. Since in the previous experiment spectral clustering achieved better results than DBSCAN for identifying symmetry clusters, we use only the clusters identified by spectral clustering in this evaluation. In total, we perform 27 experiments for the same parameter combinations $d \in \{2, 3, 4\}$, $\tau \in \{1, 2, 3\}$ and $\delta \in \{0.05, 0.1, 0.15\}$ as in Section 8.3. The symmetry clusters are determined based on a similarity graph resulting from applying MOP$_4$SA on time series data[1] as a result of observations for randvars of sub-challenge two[2] for time steps $t = 1, \ldots, 4$. We then compute the change measure $d(\mathcal{W}^t, \mathcal{W}^{t+1}, \mathcal{S})$ for each subsequent time step $t = 5, \ldots, 24$ based on new updated versions of the similarity graph resulting from the execution of MOP$_4$SA, i.e., con-

[1] https://dma.dk/safety-at-sea/navigational-information/ais-data
[2] https://github.com/FinkeNils/Processed-AIS-Data-Baltic-Sea-2020-v2



(a) Results of MOP$_4$SA and SA$_4$PG.

(b) Results of MOP$_4$SCD.

Figure 12.2.: Results of MOP$_4$SCD in conjunction with results for MOP$_4$SA for three selected parameter combinations which gave good results in the experiments in Part II.

tinuing the symbolisation and updating the similarity scores as the model is rolled out over time $t > 4$ with new observations provided as per the mentioned data set. In this section, we present the results of the evaluation of MOP$_4$SCD with symmetry clusters determined by MOP$_4$SA given those parameter combinations as input to MOP$_4$SA that lead to good results in the experiments performed in the previous part. For an overview of results for all experiments with all parameter combinations, see the extended results in Table A.3 in the Appendix. For reference, symmetry clusters obtained in the experiments presented in Section 8.3 with parameter combinations $d = 3$, $\tau = 1$, $\delta = 0.05$ (referred to as **Experiment (a)** as follows), $d = 3$, $\tau = 2$, $\delta = 0.05$ (referred to as **Experiment (b)** as follows) and $d = 3$, $\tau = 3$, $\delta = 0.1$ (referred to as **Experiment (c)** as follows) gave good results. More specifically, the use of these symmetry clusters was successfully demonstrated in the use in SA$_4$PG, i.e., avoiding groundings, without any strong negative impact on the accuracy in inference. For better comparison, results for these experiments are depicted again in Fig. 12.2a.

Whilst Fig. 12.2a depicts accuracy scores and runtime of running LDJT with preventing groundings, Fig. 12.2b accordingly depicts the results of performing MOP$_4$SCD for each respective experiment (a)-(c) to be able to compare accuracy with the change measure given by MOP$_4$SCD. Each subplot in Fig. 12.2b corresponds experiments (a)-(c) with

- the blue line corresponding to the KLD D_{KL} between the full joint distribution of the local model G_t at a time step t and the full joint distribution at G_0,
- the solid red line to the distance measure $d(\mathcal{W}^t, \mathcal{W}^{t+1}, \mathcal{S})$ for two consecutive time steps t and $t+1$,
- the dashed red line for the cumulative distance measure, i.e., from $t = 0$ until the current time step t, and,
- the red area highlighting the interval when the cumulative distance measure becomes greater than 0.5 until it has reached a value of 1.0.

The results of the experiments for SA$_4$PG show that the accuracy (orange line) decreases continuously until time $t = 13$ for experiment (a) and until time $t = 10$ for experiments (b) and (c). This is as avoiding groundings (SA$_4$PG) can cause an error in the predictions (compared to predictions without running SA$_4$PG). Since preventing groundings is only done for an object in a symmetry cluster until the object's internal threshold H (threshold on how often events are allowed to be rejected or inferred) is reached, the accuracy first decreases due to potential errors in the prediction until it increases over time again. This holds as over time the effect of any wrong events that lead to an adjustment in the (evidence) parfactor potential functions lose relevance upon new information becomes known after the wrong events were applied. Until reaching low points in accuracy, the accuracy decreases less rapidly in the first time steps, see for example the interval $5 < t < 9$ for experiment (b), and then more strongly in the following ones until reaching the low point. While the accuracy (orange line) in Fig. 12.2a falls, conversely the KLD (blue line) increases as depicted in Fig. 12.2b. The red lines represent $d(\mathcal{W}^t, \mathcal{W}^{t+1}, \mathcal{S})$ are relatively consistent for the experiment (a) at a level of 0.2, and for the experiment (b) and (c) at a level of 0.25. That is, with each time step unbalanced changes of 20% for experiment (a) and unbalanced changes of 25% for experiments (b) and (c) are measured. Correspondingly, the cumulative distance reaches a value of 0.5 at time step $t = 9$ until it reaches a value of 1 at $t = 14$ for the experiment (a), while for the two other experiments (b) and (c), the cumulative distance reaches a value of 0.5 at $t = 7$ and a value of 1 already at $t = 9$.

It is precisely this cumulative distance that can be used as an indicator to determine a point in time when a new determination of symmetry clusters makes sense. The cumulative distance serves as a valuable indicator of when a recalculation of the symmetry clusters may be beneficial. For example, in experiment (a) this point is at time $t = 8$ where the KLD D_{KL} starts to escalate more rapidly. This increase suggests that the entire joint distribution may be negatively affected if grounding prevention is continued. In particular, if the intrinsic default, i.e., aligning events between domain objects in a symmetry cluster, is continued, this would excessively affect the accuracy. This problem arises as the domain objects in the symmetry clusters no longer behave similarly, obviating the need for applying intrinsic defaults within these clusters. Relearning clusters at a threshold of 0.5 is beneficial

to prevent the KLD D_{KL} from further increasing. Thus, symmetry clusters should be determined again at $t = 9$ for the experiment (a), i.e., symmetry clusters are valid for about 4 time steps after initially constructing them, which corresponds to a whole month in our example application and therefore is a good result to considerably speed up inference while only introducing a small error in inference. Moreover, this again supports the result from the evaluation in Part II, in which we showed that MOP$_4$SA is better suited to determine symmetry structures that remain valid for shorter time periods.

As in the three experiments presented here, and equally for experiments with further parameter combinations in the Appendix, we have shown that MOP$_4$SCD is successfully able to determine time points beyond which symmetry clusters lose their validity (because from the points in time on-wards the accuracy decreases strongly) and require symmetry clusters to be determined again. Therefore, MOP$_4$SCD complements MOP$_4$SA and SA$_4$PG as a rich toolset for preventing groundings giving valuable insights about the validity of the symmetry clusters.

13 Part III: Interim Conclusion

The algorithms introduced in Part I and Part II are based on an online variant of LDJT (see Section 3.3), i.e., the model is set up once and then continuously adjusted at run time. In the last part of this work, we presented two approaches that identify (i) concept drifts and (ii) changes in symmetry structures that become visible in the model with new observations being made after the model was initially set up. At certain points in time, these changes are significant, necessitating an adjustment of the model – ideally at run time as well.

Concept drift detection is necessary for temporal models as the potential functions encoded in the model no longer match reality well enough, i.e., potential functions no longer describe the likelihood of events to occur accurately. This part contributes with an efficient change point detection algorithm to trigger a relearning of the model and thus adapt the model to the new reality. More specifically, the factorisation of the model is exploited by performing piece-wise concept drift detection per parfactor and combining outputs for each factor into a common measure. Finally, LDJT with concept drift detection was introduced, an algorithm that involves the detection of concept drifts while answering queries under evidence. In an empirical evaluation, we showed that our approach to concept drift detection is well suited for detecting points in time at which a concept drift has settled while preserving run time performance during inference.

Just as initially defined or learned potential functions encoded in a model can change over time due to changes in reality and lead to concept drift, so too can initially defined symmetry structures change. In the previous part of this work we have introduced MOP_4SA to determine symmetry clusters which denote symmetry structures in terms of objects that have temporal similarities. Based on these symmetry clusters, we have introduced SA_4PG, an algorithm for a priori prevention of groundings of a model. Groundings are prevented by applying an intrinsic default, i.e., unobserved realisations of an object are derived from the realisations of a similar object, or contradictory behaviour of an object towards other objects that actually behave similarly is smoothed out by the majority behaviour of other objects. By doing so, the advantages of lifting can be preserved. However, the effectiveness of MOP_4SA and SA_4PG generally depends on the validity of the symmetry clusters, i.e., that the symmetry structures are still valid at execution time. Since symmetry structures can change over time, we introduce MOP_4SCD, an algorithm that measures the validity

of symmetry clusters. More specifically, MOP$_4$SCD detects changes in symmetry structures based on the similarity graph, an intermediate output of MOP$_4$SA, and provides a distance measure indicating the degree of unbalanced changes within the similarity graph, thus indicating that new symmetry clusters should be determined upon reaching a certain threshold. Thus, points in time are identified when rerunning is MOP$_4$SA beneficial.

All in all, MOP$_4$SCD complements MOP$_4$SA, SA$_4$PG as a rich toolset to prevent groundings a priori. In addition to MOP$_4$SA, MOP$_4$SCD supports more accurate inference by identifying points in time when relearning symmetry clusters is reasonable for SA$_4$PG. In an experiment, we show the effectiveness of MOP$_4$SCD as a reliable indicator to relearn the clusters. Furthermore, with MOP$_4$SA, SA$_4$PG and MOP$_4$SCD and their general aim of preventing groundings a priori, we complement existing approaches, which focus on retaining lifted representation after a model has already been shattered. The interaction of existing approaches with those presented here needs to be investigated in future work.

14 Conclusion

Due to the increasing interconnectedness that can be observed in our world, the need for expressive formalisms to describe complex and relational structures also increases. Recent innovations in the field of artificial intelligence expand the possibilities for modelling complex relationships in our world. Nevertheless, theoretical model assumptions clash with practical real-world requirements. In this final chapter, we summarise the contributions of this work, all of which aim to extend DPRMs to allow for adaptation to environmental changes during model use. Further, we give some directions for future work.

14.1 Summary of Contributions

The effectiveness of DPRMs and lifted inference is inextricably linked to the continuous availability of symmetric structures and recurrent patterns captured in data. However, in any real-life setting, symmetric structures can change over time, which must also be taken into account when preserving the effectiveness of DPRMs and lifted inference. Specifically, parameters or properties of the model are most likely to change over time and affect previously exploited symmetry structures. This leads to asymmetries in the model that relieve objects from a symmetrical consideration, which is also referred to as *grounding*. In this work, we have extended the traditional understanding of non-stationarity (change in probability distributions over time) to changing domain sizes and changing symmetry structures with the aim of extending DPRMs to allow adaptations to environmental changes. Overall, we summarise the contributions made in this work as follows.

(I) **Extending DPRMs to Dynamic Domain Sizes**

In Part I, we have extended the semantics of DPRMs for dynamically changing sets of domain objects to be substituted for variables used in the model, such that sparse domain models can be specified and efficient query answering algorithms can be provided for practically relevant application domains. In order to obtain a sparse and compact representation, in DPRMs domain objects which are symmetrical with respect to the randvars in the model are treated by one representative for a group of symmetrical objects. Compared to a propositional model, adding or removing objects in a DPRMs does not require changing the number of randvars itself and thus the struc-

ture of the model, but only adjusting the set (the domain) of objects represented in the model. Nevertheless, there exist challenges in supporting dynamic domains in terms of handling evidence to ensure that new domain objects are accurately encoded in the model. Dynamic domain support also involves removing domain objects from an existing model, however, we have only focused on adding domain objects in this work, as this is a non-trivial case in DPRMs. Since there is no universal strategy for adding domain objects, three strategies called the MOST RELEVANT POTENTIAL FUNCTION (MRPF), DEFAULT POTENTIAL FUNCTION (DPF) and WEIGHTED DEFAULT POTENTIAL FUNCTION (WDPF) were introduced to deal with adding objects to the domain of the model. All strategies aim to identify potential functions that are already present in the model's parfactors and encode potentials, i.e., the likelihood of events occurring in relation to the new domain object, and effectively capture its behaviour in the model, as they already do for other domain objects already present in the model. Ultimately, we provided an extension to LDJT by dynamic domain support incorporating the strategies. Finally, an empirical evaluation was performed using an example from dry shipping, comparing the different strategies in order to substantiate the characteristics of the individual strategies. Overall, it has been shown that the exploitation of symmetries is useful for this use case, both to maintain a lifted representation (and its advantages) but also to increase the accuracy when making predictions about new domain objects.

(II) **Constructing Temporal Symmetries to Retain Lifted Representations**

Existing research in the field of StaRAI yet focuses on the exploitation of symmetries that are uncovered a posteriori over time, i.e., when the same observations are realised across multiple objects. However, due to the characteristics of real-world applications, constructing symmetries a priori is beneficial. In Part II, we introduce MOP$_4$SA, an approach for the approximation and construction of temporal symmetries a priori using a symbolisation scheme, similarity counting and clustering. Symmetries are constructed by means of identifying sets of domain objects with temporal symmetries, i.e., the same behaviour over time w.r.t. to the different parfactors and their linked PRVs of the model. Symmetry structures that have been determined a priori can be exploited to prevent a model from grounding. For this reason, we have introduced SA$_4$PG, an algorithm that allows preventing groundings a priori. By applying SA$_4$PG, information about the behaviour of objects that may not be available at runtime can be inferred from the behaviour of other objects that behave similarly. More specifically, SA$_4$PG selectively prevents groundings before they even occur by accepting, rejecting, and consequently aligning incoming events for groups of domain objects with temporal symmetries to each other, i.e., inferring intrinsic defaults to keep domain objects in groups and avoid splitting. In an experiment, we show that the use of MOP$_4$SA and SA$_4$PG speeds up inference while only causing a negligible error in the inference

and that by inferring observations for domain objects that are actually unobserved, the accuracy can even be increased. Overall, the lifted model becomes more robust to temporary deviations in behaviour of individual domain objects, which actually behave similarly in the long run to a larger group of domain objects.

(III) **Detecting Environmental Changes in DPRMs**

Exploiting symmetry structures a priori and using them to prevent groundings in lifted inference can lead to a noticeable error in the predictions based on the underlying model if symmetry structures are no longer valid. Thus, a significant change in the symmetry structure on which a DPRM is based requires the model to be relearned. In Part III, two approaches were introduced for detecting any changes to the environment, leading to either change in probability distributions or symmetry structures. First, an algorithm was presented for detecting concept drifts in DPRMs, i.e., for detecting points in time when the probability distribution encoded in the model no longer represents reality well enough. Second, MOP_4SCD was introduced, an algorithm for detecting points in time of changes in symmetry structures and evaluating the degree of the change. MOP_4SCD complements MOP_4SA and SA_4PG as a powerful tool to prevent groundings a priori. In different experiments, it was shown that both algorithms are able to detect time points of change (in terms of probability distributions or symmetry structures), and when handled, subsequent adjustments increase the overall accuracy of the temporal model over time.

All in all, DPRMs and lifted inference approaches are powerful approaches to model uncertainty and complex, relational structures in real-world applications and to perform efficient inference. In the context of this work, we have provided several extensions to DPRMs that enable adaptions in the context of non-stationarity, more specifically changes in domain sizes, probability distributions and symmetry structures. Nevertheless, the adaptability of DPRMs with regard to all three aspects remains an exciting area of research with several further possibilities.

14.2 Future Work

Based on this work, there are various opportunities to explore the real-world application of DPRMs and beyond. We highlight three topics.

Exploiting Temporal Symmetries in Domain Expansion

In Part I, we extended DPRMs to handle domain extension, i.e., adding new domain objects to an existing domain as they become known. A crucial factor for the effectiveness of the proposed strategies is the identification of similar domain objects, which we did by comparing the similarity between domain objects based on potential functions encoded in

parfactors of the model. However, this approach has a limitation as the potential functions aggregate historical information in a single time slice. That is, we can determine the symmetry in the current time step, but not the symmetry with respect to the behaviour resulting from events over several time steps. These are such temporal symmetries whose identification we introduced in Part II – but for the purpose of preventing groundings to keep advantages of a lifted representation and lifted reasoning. However, using the approach from Part II to identify similar domain objects during domain expansion can also further improve the strategies proposed in Part I, which is still open for future work.

Incremental Symmetry Clustering Approaches

The approach proposed in Part II to determine groups of domain objects with temporal symmetries is executed initially once, i.e., as part of the model construction process. Depending on whether the groups continue to represent the respective objects with symmetries to each other well, the approach is then executed again over time. More specifically, to determine symmetry clusters we have proposed ordinal pattern symbolisation for time series. Counting identical symbolisation patterns between domain objects yields a similarity measure for every two domain objects. If the similarity measures of all domain objects are arranged in a graph, the result is a similarity graph that is used to cluster similar domain objects. While symbolisation and similarity counting are performant approaches, for clustering, we fall back on well-proven clustering approaches, which are unfortunately also costly. If new observations are made, the symbolisation can be continued iteratively so that the similarity graph is updated over time. However, the costly clustering procedure would have to be rerun for each new observation that is made. To reduce costs, in Part III we have introduced a change point detection approach that identifies unbalanced changes in the similarity graph and triggers the clustering procedure only when it is significantly needed. To further reduce costs, incremental clustering methods or other heuristic or methods that allow clusters to be incrementally combined or separated could help. This remains to be investigated in future work.

Belief Revision

In both Part I and Part II we have presented approaches that result in only approximate inference being possible. This results in particular from the fact that in the approaches proposed in Part I new knowledge encoded in the model is transferred (inferred) to another (new) domain object. Similarly, in Part II, temporal symmetries of multiple domain objects that behave approximately similarly (but not always exactly the same) are exploited to prevent groundings. Even though inferring and aligning events within a symmetry cluster to prevent groundings has advantages, it also happens that events are applied that should not have been applied at all. One case, for example, is whenever events for a domain object are

not available at runtime and therefore events are inferred basis the behaviour of domain objects within the same symmetry cluster but then events become known afterwards and it turns out that the originally inferred events were wrong. In such cases, one would like to roll back the events that were transmitted incorrectly and undo the adjustments to the potential functions caused by the events, i.e., perform *Belief Revision*. Belief revision can play a crucial role in refining the model, i.e., adapting and improving its representation of domain objects and their relationships as more data becomes available. This would allow better handling of incorrect initial assumptions and lead to improved overall performance in predicting and representing the relationships between domain objects over time. Research in the context of belief revision is open for future work.

Part IV

Appendix

A Datasets

A.1 Automatic Identification System (AIS) Data

All data for experiments in this work, namely data for variables $Supply_t(Z)$, $Demand_t(Z)$ and $Idle_t(Z)$ as introduced in Chapter 2, are derived from AIS data for the Baltic Sea provided by the DMA. The processed data can be found on GitHub[1], raw data provided by the DMA can be found here[2].

The supply and idle data is derived from AIS signals provided by the DMA. Since the AIS signals, besides some specifications about the vessel itself, only provide information about the current geo-position of same, further preprocessing steps are necessary to calculate the supply and demand in the different zones of the Baltic Sea. The following parameters, given in AIS signals, are used for the calculations: timestamp, latitude, longitude, imo, mmsi, draught, destination. Fig. A.1 shows the 392 zones, in which we geographically have split up the Baltic Sea. Since we are interested in the supply flow between landmass, we only consider polygons on the coastal line. To get the polygon definition, please load the polygones.pkl in Python using pickle.

To calculate supply and idle in zones based on AIS, following steps have been performed:

(1) Get all AIS signals related to one vessel. The imo or mmsi is a unique identifier for a vessel and can be used to consider every vessel one by one.

(2) The time wise ordered AIS data (use timestamp), more precisely, the geo-position based on latitude and longitude illustrate the route a vessel is taking.

(3) To derive supply zones with its idle times, the zone in which the vessel is loading (supply zone) has to be determined. To do so, the vessels draft and destination give an indication. The point in time, i.e., the AIS signal, in which the destination changed compared to the previous signal, denotes the start of a new voyage. Further, if the vessels draft has increased, the vessel was loading cargo, thus, the vessel is in a supply zone and travels towards a new zone to discharge.

[1] https://github.com/FinkeNils/Processed-AIS-Data-Baltic-Sea-2020-v2
[2] https://www.dma.dk/SikkerhedTilSoes/Sejladsinformation/AIS/

Figure A.1.: Zone Definition for the Baltic Sea with each zone illustrated as a green rectangle.

(4) Using a vessel database, e.g. Vesselfinder[3], we determine the vessels maximum capacity in metric tons, its so called deadweight (dwt).

(5) Since the dwt denotes the capacity of a vessel in metric tons, we take the dwt of the vessel as the amount of cargo, the vessel was loading. Note, it may not reflect one hundred percent of reality.

(6) Performing such steps for each vessel, yields voyages between supply zones including the amount of cargo transported.

(7) Aggregating the data on a weekly and zone basis, gives data as provided in the supply.csv file.

(8) To determine the idle time in a supply zone, one simply calculates the time from entering a zone until loading. Averaging idle times over all vessels for the same zone at the same time step gives the idle time reflected in the idle.csv file.

[3]https://www.vesselfinder.com/de

A.2 Precision, Recall and F_1-Score Results for LDJT with Dynamic Domains Experiment

Precision p and recall r are defined by

$$\text{prevision } p = \frac{TP}{TP+FP}, \tag{A.2.1}$$

$$\text{recall } r = \frac{TP}{TP+FN}, \tag{A.2.2}$$

with TP as true positives, TN as true negatives, FP as false positives, and FN denotes false negatives.

	Predicted Positive	**Predicted Negative**
True Instances	True Positives (TP)	False Negatives (FN)
False Instances	False Positives (FP)	True Negatives (TN)

For a multiclass problem, i.e., with more than two supply range values as in our example, the term *positive* refers to a specific class, and the term *negative* refers to all other classes combined. For example, considering $\mathcal{R}(Supply_t(Z)) = \{high, medium, low\}$ and the actual positive (P) supply value is $Supply_t(Z) = high$ and the not positives/negatives (N) are $Supply_t(Z) = medium$ and $Supply_t(Z) = low$, then in this scenario the following applies:

- True Positive (TP): Predicted *high* and actual value is *high*.
- False Positive (FP): Predicted *high* but the actual value is *medium* or *low*.
- True Negative (TN): Predicted *medium* or *low* and the actual value is *medium* or *low*.
- False Negative (FN): Predicted *medium* or *low* when the actual value is *high*.

Then, the accuracy is defined by

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad , \tag{A.2.3}$$

and the F_1-score is defined as a measure that combines precision and recall, i.e., the harmonic mean of precision and recall, by

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{A.2.4}$$

In the experiments, we use the weighted F_1-score. That means in our multi-class classification tasks, F_1-score for each class label is averaged to yield an overall F_1-score. Table A.1 shows accuracy, precision, recall and F_1-score of the experiments performed in Section 5.4.

		$\delta = 0.05$			$\delta = 0.1$			$\delta = 0.15$		
$[t-i,t]$	Measure	DPF	WDPF	MRPF	DPF	WDPF	MRPF	DPF	WDPF	MRPF
[5, 8]	accuracy	0,370	0,285	0,240	0,400	0,244	0,304	0,310	0,280	0,375
	precision	0,210	0,125	0,120	0,240	0,203	0,181	0,190	0,170	0,197
	recall	0,370	0,307	0,240	0,400	0,293	0,287	0,310	0,280	0,350
	F_1-score	0,268	0,178	0,160	0,300	0,240	0,222	0,236	0,212	0,252
[5, 9]	accuracy	0,280	0,210	0,285	0,250	0,340	0,367	0,340	0,310	0,378
	precision	0,140	0,110	0,125	0,180	0,230	0,296	0,220	0,200	0,294
	recall	0,280	0,210	0,303	0,250	0,340	0,430	0,340	0,310	0,388
	F_1-score	0,187	0,144	0,177	0,209	0,274	0,351	0,267	0,243	0,334
[5, 10]	accuracy	0,280	0,346	0,340	0,290	0,393	0,320	0,260	0,240	0,264
	precision	0,200	0,234	0,220	0,180	0,172	0,170	0,120	0,110	0,143
	recall	0,280	0,370	0,340	0,290	0,366	0,320	0,260	0,240	0,254
	F_1-score	0,233	0,287	0,267	0,222	0,234	0,222	0,164	0,151	0,183
[5, 11]	accuracy	0,270	0,307	0,280	0,280	0,246	0,200	0,300	0,381	0,290
	precision	0,190	0,212	0,200	0,170	0,196	0,130	0,220	0,306	0,210
	recall	0,270	0,361	0,280	0,280	0,217	0,200	0,300	0,373	0,290
	F_1-score	0,223	0,267	0,233	0,212	0,206	0,158	0,254	0,336	0,244
[5, 12]	accuracy	0,260	0,278	0,270	0,300	0,210	0,305	0,310	0,410	0,350
	precision	0,130	0,176	0,130	0,170	0,120	0,186	0,220	0,282	0,230
	recall	0,260	0,279	0,270	0,300	0,210	0,267	0,310	0,424	0,350
	F_1-score	0,173	0,216	0,176	0,217	0,153	0,219	0,257	0,339	0,278
[5, 13]	accuracy	0,280	0,286	0,280	0,340	0,252	0,190	0,310	0,211	0,210
	precision	0,160	0,248	0,150	0,190	0,106	0,100	0,220	0,245	0,170
	recall	0,280	0,280	0,280	0,340	0,254	0,190	0,310	0,255	0,210
	F_1-score	0,204	0,263	0,195	0,244	0,150	0,131	0,257	0,250	0,188
[5, 14]	accuracy	0,250	0,210	0,248	0,250	0,280	0,370	0,200	0,270	0,329
	precision	0,120	0,090	0,159	0,210	0,220	0,243	0,090	0,130	0,176
	recall	0,250	0,210	0,256	0,250	0,280	0,315	0,200	0,270	0,360
	F_1-score	0,162	0,126	0,196	0,228	0,246	0,274	0,124	0,176	0,237
[6, 8]	accuracy	0,280	0,435	0,350	0,350	0,350	0,426	0,360	0,350	0,380
	precision	0,220	0,270	0,250	0,240	0,230	0,264	0,250	0,240	0,287
	recall	0,280	0,431	0,350	0,350	0,350	0,384	0,360	0,350	0,380
	F_1-score	0,246	0,332	0,292	0,285	0,278	0,313	0,295	0,285	0,327
[6, 9]	accuracy	0,270	0,326	0,310	0,330	0,244	0,230	0,320	0,260	0,334
	precision	0,140	0,167	0,160	0,200	0,184	0,120	0,190	0,150	0,249
	recall	0,270	0,379	0,310	0,330	0,247	0,230	0,320	0,260	0,276

	F_1-score	0,184	0,232	0,211	0,249	0,211	0,158	0,238	0,190	0,262
[6, 10]	accuracy	0,250	0,140	0,168	0,220	0,260	0,310	0,240	0,390	0,413
	precision	0,110	0,050	0,101	0,120	0,120	0,188	0,160	0,250	0,319
	recall	0,250	0,140	0,205	0,220	0,260	0,278	0,240	0,390	0,436
	F_1-score	0,153	0,074	0,135	0,155	0,164	0,224	0,192	0,305	0,369
[6, 11]	accuracy	0,250	0,362	0,270	0,300	0,329	0,310	0,310	0,244	0,260
	precision	0,120	0,189	0,130	0,150	0,160	0,150	0,200	0,244	0,203
	recall	0,250	0,357	0,270	0,300	0,320	0,310	0,310	0,313	0,319
	F_1-score	0,162	0,247	0,176	0,200	0,213	0,202	0,243	0,275	0,248
[6, 12]	accuracy	0,330	0,188	0,160	0,300	0,340	0,424	0,290	0,310	0,337
	precision	0,200	0,118	0,110	0,240	0,260	0,274	0,150	0,150	0,203
	recall	0,330	0,190	0,160	0,300	0,340	0,395	0,290	0,310	0,342
	F_1-score	0,249	0,146	0,130	0,267	0,295	0,323	0,198	0,202	0,255
[6, 13]	accuracy	0,240	0,328	0,240	0,270	0,329	0,250	0,240	0,462	0,390
	precision	0,240	0,251	0,240	0,190	0,172	0,170	0,190	0,302	0,270
	recall	0,240	0,338	0,240	0,270	0,267	0,250	0,240	0,417	0,390
	F_1-score	0,240	0,288	0,240	0,223	0,210	0,202	0,212	0,350	0,319
[6, 14]	accuracy	0,290	0,300	0,260	0,350	0,351	0,270	0,200	0,260	0,230
	precision	0,170	0,213	0,130	0,220	0,155	0,140	0,080	0,091	0,080
	recall	0,290	0,325	0,260	0,350	0,338	0,270	0,200	0,277	0,230
	F_1-score	0,214	0,257	0,173	0,270	0,213	0,184	0,114	0,137	0,119
[7, 8]	accuracy	0,340	0,290	0,372	0,310	0,292	0,250	0,390	0,353	0,350
	precision	0,250	0,210	0,281	0,170	0,204	0,130	0,260	0,274	0,230
	recall	0,340	0,328	0,290	0,310	0,265	0,250	0,390	0,412	0,350
	F_1-score	0,288	0,256	0,285	0,220	0,231	0,171	0,312	0,329	0,278
[7, 9]	accuracy	0,380	0,260	0,332	0,320	0,371	0,280	0,340	0,280	0,326
	precision	0,220	0,150	0,209	0,250	0,259	0,220	0,200	0,150	0,200
	recall	0,380	0,260	0,295	0,320	0,294	0,280	0,340	0,280	0,321
	F_1-score	0,279	0,190	0,245	0,281	0,275	0,246	0,252	0,195	0,247
[7, 10]	accuracy	0,280	0,310	0,322	0,200	0,306	0,250	0,380	0,244	0,220
	precision	0,160	0,219	0,189	0,100	0,129	0,120	0,230	0,224	0,140
	recall	0,280	0,310	0,310	0,200	0,250	0,250	0,380	0,247	0,220
	F_1-score	0,204	0,256	0,235	0,133	0,170	0,162	0,287	0,235	0,171
[7, 11]	accuracy	0,280	0,340	0,290	0,340	0,200	0,247	0,210	0,322	0,250
	precision	0,180	0,188	0,180	0,170	0,090	0,130	0,110	0,153	0,130
	recall	0,280	0,290	0,290	0,340	0,200	0,277	0,210	0,288	0,250

	F_1-score	0,219	0,228	0,222	0,227	0,124	0,177	0,144	0,200	0,171
[7, 12]	accuracy	0,300	0,320	0,230	0,340	0,289	0,250	0,250	0,368	0,290
	precision	0,140	0,106	0,090	0,250	0,280	0,190	0,100	0,160	0,120
	recall	0,300	0,301	0,230	0,340	0,272	0,250	0,250	0,290	0,290
	F_1-score	0,191	0,157	0,129	0,288	0,276	0,216	0,143	0,207	0,170
[7, 13]	accuracy	0,290	0,280	0,287	0,260	0,280	0,352	0,290	0,366	0,290
	precision	0,260	0,260	0,298	0,170	0,180	0,237	0,190	0,229	0,200
	recall	0,290	0,280	0,349	0,260	0,280	0,332	0,290	0,356	0,290
	F_1-score	0,274	0,270	0,322	0,206	0,219	0,277	0,230	0,278	0,237
[7, 14]	accuracy	0,180	0,416	0,330	0,310	0,383	0,310	0,290	0,262	0,250
	precision	0,120	0,258	0,200	0,220	0,235	0,210	0,150	0,165	0,120
	recall	0,180	0,356	0,330	0,310	0,403	0,310	0,290	0,251	0,250
	F_1-score	0,144	0,299	0,249	0,257	0,297	0,250	0,198	0,199	0,162
[8, 9]	accuracy	0,360	0,310	0,395	0,290	0,263	0,210	0,380	0,417	0,340
	precision	0,260	0,220	0,250	0,170	0,147	0,120	0,260	0,277	0,240
	recall	0,360	0,310	0,311	0,290	0,218	0,210	0,380	0,410	0,340
	F_1-score	0,302	0,257	0,277	0,214	0,175	0,153	0,309	0,331	0,281
[8, 10]	accuracy	0,350	0,270	0,295	0,280	0,330	0,415	0,440	0,355	0,260
	precision	0,220	0,170	0,211	0,190	0,210	0,245	0,330	0,242	0,210
	recall	0,350	0,270	0,343	0,280	0,330	0,417	0,440	0,289	0,260
	F_1-score	0,270	0,209	0,261	0,226	0,257	0,309	0,377	0,263	0,232
[8, 11]	accuracy	0,270	0,302	0,250	0,280	0,295	0,280	0,270	0,326	0,320
	precision	0,160	0,212	0,130	0,190	0,224	0,180	0,160	0,222	0,190
	recall	0,270	0,291	0,250	0,280	0,332	0,280	0,270	0,362	0,320
	F_1-score	0,201	0,245	0,171	0,226	0,268	0,219	0,201	0,275	0,238
[8, 12]	accuracy	0,380	0,310	0,375	0,310	0,260	0,288	0,420	0,150	0,234
	precision	0,270	0,220	0,263	0,210	0,170	0,243	0,260	0,060	0,129
	recall	0,380	0,310	0,329	0,310	0,260	0,350	0,420	0,150	0,191
	F_1-score	0,316	0,257	0,292	0,250	0,206	0,287	0,321	0,086	0,154
[8, 13]	accuracy	0,230	0,270	0,309	0,390	0,367	0,340	0,290	0,388	0,300
	precision	0,170	0,279	0,180	0,270	0,250	0,306	0,140	0,196	0,140
	recall	0,230	0,270	0,342	0,390	0,423	0,340	0,290	0,300	0,351
	F_1-score	0,196	0,275	0,236	0,319	0,314	0,322	0,189	0,237	0,200
[8, 14]	accuracy	0,260	0,321	0,300	0,240	0,261	0,260	0,310	0,280	0,350
	precision	0,150	0,175	0,170	0,140	0,142	0,140	0,150	0,120	0,124
	recall	0,260	0,380	0,300	0,240	0,351	0,260	0,310	0,280	0,289

	F_1-score	0,190	0,240	0,217	0,177	0,203	0,182	0,202	0,168	0,173
[9, 10]	accuracy	0,270	0,360	0,404	0,360	0,396	0,300	0,370	0,341	0,260
	precision	0,200	0,318	0,250	0,260	0,240	0,262	0,210	0,156	0,140
	recall	0,270	0,360	0,418	0,360	0,369	0,300	0,370	0,261	0,260
	F_1-score	0,230	0,338	0,313	0,302	0,291	0,280	0,268	0,195	0,182
[9, 11]	accuracy	0,430	0,399	0,300	0,400	0,304	0,300	0,380	0,309	0,260
	precision	0,300	0,286	0,200	0,300	0,301	0,240	0,260	0,257	0,190
	recall	0,430	0,322	0,300	0,400	0,392	0,300	0,380	0,321	0,260
	F_1-score	0,353	0,303	0,240	0,343	0,340	0,267	0,309	0,285	0,220
[9, 12]	accuracy	0,260	0,355	0,340	0,380	0,323	0,240	0,370	0,230	0,256
	precision	0,200	0,267	0,250	0,240	0,175	0,160	0,210	0,100	0,175
	recall	0,260	0,352	0,340	0,380	0,329	0,240	0,370	0,230	0,246
	F_1-score	0,226	0,304	0,288	0,294	0,228	0,192	0,268	0,139	0,205
[9, 13]	accuracy	0,240	0,210	0,212	0,190	0,340	0,348	0,280	0,289	0,240
	precision	0,160	0,140	0,190	0,110	0,211	0,180	0,160	0,143	0,140
	recall	0,240	0,210	0,305	0,190	0,340	0,440	0,280	0,269	0,240
	F_1-score	0,192	0,168	0,234	0,139	0,260	0,255	0,204	0,187	0,177
[9, 14]	accuracy	0,230	0,325	0,300	0,340	0,318	0,240	0,330	0,310	0,270
	precision	0,150	0,175	0,170	0,190	0,216	0,130	0,200	0,228	0,170
	recall	0,230	0,366	0,300	0,340	0,259	0,240	0,330	0,360	0,270
	F_1-score	0,182	0,237	0,217	0,244	0,236	0,169	0,249	0,279	0,209
[10, 11]	accuracy	0,300	0,420	0,390	0,390	0,380	0,360	0,320	0,421	0,350
	precision	0,220	0,323	0,270	0,280	0,352	0,270	0,190	0,274	0,200
	recall	0,300	0,477	0,390	0,390	0,455	0,360	0,320	0,408	0,350
	F_1-score	0,254	0,385	0,319	0,326	0,397	0,309	0,238	0,328	0,255
[10, 12]	accuracy	0,350	0,280	0,309	0,360	0,360	0,402	0,300	0,251	0,250
	precision	0,220	0,170	0,219	0,220	0,210	0,239	0,200	0,264	0,180
	recall	0,350	0,304	0,280	0,360	0,453	0,360	0,300	0,252	0,250
	F_1-score	0,270	0,218	0,246	0,273	0,287	0,288	0,240	0,258	0,209
[10, 13]	accuracy	0,300	0,264	0,240	0,360	0,231	0,220	0,260	0,340	0,427
	precision	0,230	0,243	0,190	0,220	0,192	0,120	0,150	0,233	0,180
	recall	0,300	0,240	0,271	0,360	0,220	0,248	0,260	0,340	0,417
	F_1-score	0,260	0,242	0,224	0,273	0,205	0,162	0,190	0,276	0,251
[10, 14]	accuracy	0,310	0,282	0,200	0,330	0,383	0,310	0,230	0,412	0,330
	precision	0,170	0,094	0,090	0,220	0,272	0,220	0,130	0,193	0,180
	recall	0,310	0,262	0,200	0,330	0,351	0,310	0,230	0,415	0,330

F_1-score	0,220	0,138	0,124	0,264	0,307	0,257	0,166	0,264	0,233

Table A.1.: Accuracy, precision, recall and F_1-score for experiments in LDJT with Dynamic Domains

A.3 Extended Results for MOP_4SA, SA_4PG and MOP_4SCD

In part two of this work we have introduced MOP_4SA and SA_4PG for the construction of symmetries based on historical evidence yielding symmetry clusters. SA_4PG works based on those symmetry clusters by preventing groundings through applying an intrinsic default, i.e., inferring observations for objects which are either unobserved or behave contrary to its cluster objects. The effectiveness of MOP_4SA and SA_4PG rely very much on the validity of the clusters, i.e., that the symmetry structures are still valid. Since symmetry structures can change over time, we introduce MOP_4SCD in part three of this work to measure the validity of the symmetry clusters.

Table A.2 shows extended results for MOP_4SA and SA_4PG from experiments performed and explained in Section 8.3, while Table A.3 shows extended results for MOP_4SCD from experiments performed and explained in Chapter 12.

d	τ	δ	DBSCAN Runtime [s]	DBSCAN *Acc*	Spectral Runtime [s]	Spectral *Acc*	Exact Runtime [s]	DBSCAN Runtime [s]	DBSCAN *Acc*	Spectral Runtime [s]	Spectral *Acc*	Exact Runtime [s]
			t=[5,10)					**t=[10,15)**				
2	1	0.05	19.6	0.66	19.1	0.68	38.7	37.8	0.71	37.5	0.38	95.2
		0.1	15.3	0.72	19.8	0.96	29.8	29.9	0.60	39.6	0.81	78.1
		0.15	14.5	0.56	14.5	0.87	25.1	28.4	0.60	28.8	0.14	69.6
	2	0.05	14.1	0.42	13.9	0.83	22.8	27.6	0.46	27.7	0.61	65.7
		0.1	17.6	0.54	17.5	0.94	33.2	34.9	0.51	34.4	0.56	87.5
		0.15	12.5	0.28	12.4	0.99	15.8	24.4	0.35	24.8	0.78	50.9
	3	0.05	15.7	0.76	15.7	0.97	29.1	31.3	0.56	30.8	0.75	78.7
		0.1	22.0	0.76	21.8	0.91	41.1	42.5	0.49	42.4	0.69	103.9
		0.15	21.2	0.93	20.2	0.87	37.7	41.0	0.82	39.0	0.61	96.5
3	1	0.05	13.6	0.24	13.6	0.98	17.1	26.5	0.32	27.3	0.85	55.8
		0.1	15.5	0.30	14.5	0.72	20.0	29.5	0.43	28.8	0.44	62.0
		0.15	15.2	0.29	15.1	0.94	20.3	29.3	0.38	30.2	0.79	63.7
	2	0.05	12.6	0.22	12.4	0.99	15.1	24.4	0.33	24.7	0.80	50.1
		0.1	12.5	0.22	12.4	0.85	15.1	24.4	0.30	24.9	0.80	50.0
		0.15	12.8	0.22	12.7	0.98	15.4	24.9	0.40	25.2	0.75	51.3
	3	0.05	249.7	1.00	247.8	1.00	244.0	488.7	1.00	485.8	1.00	480.0
		0.1	13.6	0.24	13.7	0.96	16.8	26.4	0.43	27.3	0.81	55.3
		0.15	14.8	0.31	14.5	0.89	19.8	28.6	0.40	28.8	0.73	61.8
4	1	0.05	79.0	0.55	79.9	0.95	90.9	154.3	0.69	155.1	0.84	214.2
		0.1	12.5	0.22	12.6	0.09	15.0	24.3	0.32	24.5	0.29	49.4
		0.15	12.8	0.22	12.8	0.36	15.4	24.9	0.31	25.0	0.43	51.1
	2	0.05	127.3	0.78	127.9	0.98	137.4	251.4	0.84	249.1	0.96	297.7
		0.1	86.1	0.55	86.9	0.88	97.4	168.1	0.62	169.1	0.83	228.5
		0.15	12.5	0.22	12.4	0.22	14.9	24.2	0.30	24.2	0.35	48.9
	3	0.05	79.7	0.57	80.6	0.97	89.1	155.4	0.72	156.3	0.92	212.1
		0.1	105.5	0.65	107.6	0.94	114.9	205.7	0.74	208.1	0.91	261.3
		0.15	158.5	0.98	164.7	0.83	169.2	315.9	0.88	316.1	0.74	356.4
			t=[15,20)					**t=[20,25)**				
2	1	0.05	71.7	0.83	73.7	0.64	154.5	123.9	0.85	133.1	0.88	214.0
		0.1	58.9	0.68	69.4	0.80	127.9	104.8	0.82	115.3	0.88	178.4
		0.15	55.2	0.63	56.3	0.13	116.6	98.9	0.14	100.4	0.42	164.2
	2	0.05	55.1	0.85	55.3	0.72	111.2	97.6	0.90	98.5	0.83	157.0
		0.1	67.9	0.65	64.9	0.47	143.1	117.1	0.71	114.6	0.69	199.6
		0.15	46.7	0.80	49.4	0.81	90.1	83.8	0.84	87.3	0.90	129.9
	3	0.05	63.8	0.53	58.6	0.66	130.0	111.3	0.72	104.7	0.68	182.2
		0.1	78.1	0.61	80.5	0.76	168.6	131.4	0.65	139.4	0.87	233.9
		0.15	75.5	0.83	71.3	0.56	160.2	127.5	0.88	125.5	0.69	221.3
3	1	0.05	49.6	0.74	53.8	0.85	98.8	90.3	0.81	94.7	0.91	142.5
		0.1	54.6	0.65	57.1	0.47	108.3	98.0	0.71	100.4	0.55	155.5
		0.15	55.0	0.25	58.8	0.82	111.1	99.4	0.09	103.0	0.89	159.5
	2	0.05	46.8	0.83	49.2	0.83	89.2	84.6	0.81	88.1	0.91	128.9
		0.1	46.8	0.64	49.4	0.78	89.0	84.5	0.23	86.9	0.74	128.8
		0.15	46.3	0.01	49.8	0.78	91.5	84.8	0.60	89.4	0.84	132.3
	3	0.05	728.7	1.00	726.9	1.00	714.6	968.8	1.00	968.8	1.00	950.4
		0.1	51.0	0.69	55.0	0.80	98.2	91.9	0.51	96.3	0.86	141.8
		0.15	54.2	0.73	56.6	0.74	107.6	98.9	0.87	99.4	0.86	154.4
4	1	0.05	257.8	0.88	245.3	0.82	343.5	382.2	0.96	361.1	0.87	472.5
		0.1	46.6	0.73	45.3	0.69	88.0	83.8	0.81	82.0	0.96	127.2
		0.15	46.3	0.69	49.6	0.74	93.4	84.7	0.81	87.4	0.84	134.2
	2	0.05	401.4	0.94	382.2	0.96	460.5	562.2	0.98	533.9	0.97	624.7
		0.1	275.1	0.30	264.5	0.84	362.6	406.3	0.66	381.1	0.87	497.2
		0.15	45.0	0.75	45.8	0.73	87.1	81.6	0.82	82.3	0.80	126.1
	3	0.05	253.7	0.52	248.7	0.93	340.1	382.7	0.74	361.2	0.96	468.2
		0.1	325.9	0.90	321.7	0.92	413.8	472.8	0.95	460.3	0.94	564.5
		0.15	480.2	0.88	478.9	0.82	545.1	663.8	0.95	656.8	0.95	733.7

Table A.2.: Extended results for the empirical evaluation of MOP_4SA and SA_4PG.

d	τ	δ	$d(\mathcal{W}^t, \mathcal{W}^{t+1}, \mathcal{S})$ **5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**
2	1	0.05	.067	.067	.068	.065	.066	.068	.067	.066	.063	.065	.069	.068	.062	.061	.064	.069	.064	.067	.068
		0.1	.102	.112	.116	.107	.114	.115	.109	.114	.111	.105	.101	.11	.111	.116	.11	.109	.114	.107	.113
		0.15	.135	.132	.142	.125	.136	.142	.142	.138	.14	.135	.137	.133	.139	.132	.129	.135	.131	.134	.143
	2	0.05	.105	.113	.114	.116	.113	.115	.113	.114	.115	.115	.116	.113	.117	.114	.115	.112	.113	.113	.116
		0.1	.068	.068	.069	.073	.068	.074	.061	.082	.071	.071	.077	.08	.073	.065	.075	.072	.082	.074	.081
		0.15	.177	.18	.18	.185	.18	.184	.178	.185	.18	.182	.184	.181	.181	.181	.178	.184	.185	.184	.188
	3	0.05	.079	.082	.081	.082	.086	.092	.087	.08	.09	.083	.079	.089	.08	.085	.084	.089	.086	.082	.088
		0.1	.037	.038	.037	.039	.039	.039	.037	.035	.038	.039	.034	.039	.038	.037	.036	.038	.038	.038	.038
		0.15	.044	.043	.043	.046	.046	.044	.043	.045	.044	.045	.044	.046	.048	.046	.044	.044	.046	.046	.05
3	1	0.05	.199	.201	.2	.2	.199	.201	.2	.201	.201	.201	.201	.19	.201	.2	.201	.2	.2	.2	.201
		0.1	.214	.219	.219	.218	.219	.218	.217	.216	.216	.218	.218	.217	.217	.217	.219	.217	.218	.217	.219
		0.15	.213	.218	.216	.217	.218	.219	.218	.219	.217	.218	.218	.218	.218	.217	.218	.219	.218	.218	.217
	2	0.05	.238	.239	.239	.24	.24	.24	.239	.24	.24	.24	.24	.241	.239	.239	.238	.239	.24	.239	.24
		0.1	.237	.239	.238	.239	.239	.239	.239	.24	.239	.24	.239	.239	.239	.238	.239	.239	.24	.239	.24
		0.15	.236	.237	.238	.238	.239	.238	.239	.238	.238	.239	.238	.238	.237	.237	.238	.238	.238	.238	.238
	3	0.05	.001	.001	.0	.001	.001	.001	.001	.001	.001	.001	.001	.001	.001	.001	.001	.001	.001	.001	.001
		0.1	.232	.231	.23	.233	.232	.233	.235	.233	.234	.233	.233	.232	.233	.233	.233	.235	.234	.233	.234
		0.15	.223	.226	.223	.228	.227	.227	.226	.227	.23	.227	.227	.227	.229	.228	.228	.229	.226	.227	.228
4	1	0.05	.056	.055	.056	.055	.056	.056	.056	.055	.055	.056	.056	.056	.056	.056	.056	.056	.056	.056	.056
		0.1	.176	.178	.178	.178	.178	.178	.178	.176	.178	.178	.178	.178	.178	.178	.178	.178	.178	.178	.178
		0.15	.211	.21	.213	.213	.213	.213	.213	.213	.212	.212	.213	.213	.213	.213	.213	.213	.213	.213	.213
	2	0.05	.033	.034	.035	.034	.035	.034	.035	.035	.035	.035	.035	.035	.035	.035	.035	.035	.035	.035	.035
		0.1	.075	.076	.076	.076	.076	.076	.076	.075	.075	.076	.076	.076	.075	.076	.076	.075	.076	.075	.076
		0.15	.218	.216	.218	.218	.218	.218	.218	.218	.218	.218	.218	.218	.218	.218	.218	.218	.218	.218	.218
	3	0.05	.064	.064	.064	.065	.065	.064	.065	.065	.065	.065	.065	.065	.065	.065	.065	.064	.065	.065	.065
		0.1	.049	.05	.049	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.049	.05	.05	.05
		0.15	.016	.015	.015	.016	.016	.016	.016	.016	.016	.016	.016	.016	.016	.016	.016	.016	.016	.016	.016

d	τ	δ	$d(\mathcal{W}^t, \mathcal{W}^{t+1}, \mathcal{S})$ **24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**
2	1	0.05	.066	.067	.064	.064	.064	.068	.067	.064	.063	.065	.066	.067	.062	.068	.066	.067	.066	.067	.067
		0.1	.116	.11	.103	.103	.115	.115	.108	.11	.107	.115	.109	.107	.114	.117	.107	.116	.112	.115	.114
		0.15	.135	.135	.139	.137	.139	.136	.132	.143	.133	.143	.132	.134	.129	.137	.137	.142	.13	.141	.14
	2	0.05	.111	.115	.113	.114	.111	.11	.11	.112	.108	.112	.11	.111	.112	.112	.115	.114	.11	.113	.109
		0.1	.078	.079	.073	.077	.077	.067	.074	.08	.068	.072	.065	.076	.072	.074	.076	.073	.068	.073	.067
		0.15	.184	.185	.182	.182	.182	.182	.178	.183	.172	.184	.175	.179	.183	.177	.184	.186	.176	.187	.18
	3	0.05	.075	.087	.08	.086	.09	.085	.091	.086	.089	.083	.091	.076	.086	.088	.082	.091	.087	.088	.092
		0.1	.039	.039	.036	.036	.039	.037	.039	.038	.037	.036	.036	.037	.036	.038	.035	.038	.037	.035	.039
		0.15	.046	.044	.045	.045	.048	.048	.048	.046	.045	.043	.047	.042	.045	.049	.048	.048	.049	.044	.048
3	1	0.05	.201	.2	.2	.2	.201	.198	.199	.2	.198	.201	.191	.2	.19	.2	.199	.2	.2	.2	.198
		0.1	.218	.218	.217	.217	.217	.218	.217	.217	.216	.217	.218	.217	.218	.217	.216	.217	.215	.219	.217
		0.15	.218	.218	.218	.218	.219	.217	.217	.218	.217	.216	.216	.217	.219	.217	.218	.216	.218	.219	.216
	2	0.05	.24	.239	.24	.239	.239	.239	.238	.239	.237	.239	.239	.237	.24	.238	.238	.24	.239	.24	.239
		0.1	.239	.239	.239	.239	.239	.239	.238	.239	.238	.237	.238	.238	.24	.237	.238	.239	.239	.239	.239
		0.15	.238	.237	.238	.237	.237	.238	.236	.238	.236	.237	.237	.237	.238	.236	.237	.238	.236	.238	.236
	3	0.05	.001	.001	.001	.001	.001	.001	.001	.001	.001	.001	.001	.001	.001	.001	.001	.001	.001	.001	.001
		0.1	.233	.234	.232	.235	.233	.232	.233	.231	.233	.233	.233	.231	.232	.232	.234	.235	.233	.232	.235
		0.15	.229	.229	.228	.228	.228	.227	.228	.227	.226	.228	.226	.227	.226	.225	.228	.228	.228	.228	.226
4	1	0.05	.056	.055	.056	.056	.055	.055	.055	.055	.055	.055	.054	.055	.056	.056	.056	.056	.056	.056	.056
		0.1	.178	.178	.178	.178	.178	.178	.178	.178	.178	.178	.178	.178	.178	.178	.178	.178	.178	.178	.178
		0.15	.213	.213	.213	.213	.213	.213	.213	.213	.213	.213	.213	.213	.213	.213	.213	.213	.213	.213	.213
	2	0.05	.035	.035	.035	.034	.035	.034	.035	.035	.035	.034	.035	.034	.035	.034	.035	.035	.035	.035	.035
		0.1	.075	.076	.076	.076	.076	.076	.076	.075	.076	.074	.076	.075	.076	.076	.076	.076	.076	.076	.075
		0.15	.218	.218	.218	.218	.218	.216	.218	.218	.218	.218	.218	.218	.216	.218	.218	.218	.218	.218	.218
	3	0.05	.065	.065	.063	.065	.065	.065	.065	.065	.065	.065	.065	.065	.064	.064	.065	.065	.064	.065	.065
		0.1	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05	.05
		0.15	.016	.016	.016	.016	.016	.016	.016	.016	.016	.016	.016	.016	.016	.016	.016	.016	.016	.016	.016

Table A.3.: Distances as a result of running MOP$_4$SCD between consecutive time steps.

Bibliography

Rakesh Agrawal, Christos Faloutsos, and Arun Swami. Efficient Similarity Search in Sequence Databases. In David B. Lomet (Ed.), *Foundations of Data Organization and Algorithms*, pages 69–84. Springer, 1993.

Babak Ahmadi, Kristian Kersting, Martin Mladenov, and Sriraam Natarajan. Exploiting Symmetries for Scaling Loopy Belief Propagation and Relational Training. *Machine Learning*, 92:91–132, 2013.

Alexandra Antoniouk, Karsten Keller, and Sergiy Maksymenko. Kolmogorov-Sinai Entropy via Separation Properties of Order-Generated σ-Algebras. *Discrete & Continuous Dynamical Systems*, 34(5):1793–1809, 2014.

Christoph Bandt and Bernd Pompe. Permutation Entropy: A Natural Complexity Measure for Time Series. *Physical Review Letters*, 88(17):174102, 2002.

Sadik Özlen Başer and Abdullah Açık. The Effects of Global Economic Growth on Dry Bulk Shipping Markets and Freight Rates. *Journal of International Trade and Economic Researches*, 3(1):1 – 17, 2019.

Tanya Braun and Ralf Möller. Lifted Junction Tree Algorithm. In *Proceedings of KI 2016: Advances in Artificial Intelligence*, pages 30–42. Springer, 2016.

Bill Chiu, Eamonn Keogh, and Stefano Lonardi. Probabilistic Discovery of Time Series Motifs. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 493–498. ACM, 2003.

Samuel Choi, Dit-Yan Yeung, and Nevin Zhang. Hidden-Mode Markov Decision Processes. In *Proceedings of the International Joint Conference on Artificial Intelligence 1999*, 1999.

Nils Finke and Marisa Mohr. A Priori Approximation of Symmetries in Probabilistic Dynamic Relational Models. In Stefan Edelkamp, Ralf Möller, and Elmar Rueckert (Eds.), *KI 2021: Advances in Artificial Intelligence*, pages 309–323. Springer, 2021.

Nils Finke and Ralf Möller. On the Approximation of Symmetries and Related Structural Changes in Dynamic Probabilistic Relational Models. *Advances in Science, Technology and Engineering Systems Journal*, 7(2), 2022.

Nils Finke, Marcel Gehrke, Tanya Braun, Tristan Potten, and Ralf Möller. Investigating Ma-

tureness of Probabilistic Graphical Models for Dry-Bulk Shipping. In Manfred Jaeger and Thomas Dyhre Nielsen (Eds.), *Proceedings of the 10th International Conference on Probabilistic Graphical Models*, volume 138 of *Proceedings of Machine Learning Research*, pages 197–208. PMLR, 2020.

Nils Finke, Marcel Gehrke, Tanya Braun, Tristan Potten, and Ralf Möller. Investigating Matureness of Probabilistic Graphical Models for Dry-Bulk Shipping. In *Proceedings of the 10th International Conference on Probabilistic Graphical Models*, pages 197–208, 2020.

Nils Finke, Tanya Braun, Marcel Gehrke, and Ralf Möller. Concept Drift Detection in Dynamic Probabilistic Relational Models. *The International FLAIRS Conference Proceedings*, 34, 2021.

Nils Finke, Tanya Braun, Marcel Gehrke, and Ralf Möller. Dynamic Domain Sizes in Temporal Probabilistic Relational Models. *The International FLAIRS Conference Proceedings*, 34, 2021.

Nils Finke, Marisa Mohr, Alexander Lontke, Marwin Zünfle, Samuel Kounev, and Ralf Möller. Recommendations for Data-Driven Degradation Estimation with Case Studies from Manufacturing and Dry-Bulk Shipping. In Samira Cherfi, Anna Perini, and Selmin Nurcan (Eds.), *Research Challenges in Information Science - 15th International Conference, RCIS 2021*, volume 415 of *Lecture Notes in Business Information Processing*, pages 189–204. Springer, 2021.

Nils Finke, Ralf Möller, and Marisa Mohr. Multivariate Ordinal Patterns for Symmetry Approximation in Dynamic Probabilistic Relational Models. In Guodong Long, Xinghuo Yu, and Sen Wang (Eds.), *AI 2021: Advances in Artificial Intelligence*, pages 543–555. Springer, 2022.

Marcel Gehrke, Tanya Braun, and Ralf Möller. Lifted Dynamic Junction Tree Algorithm. In *Proceedings of the International Conference on Conceptual Structures*, pages 55–69. Springer, 2018.

Marcel Gehrke, Tanya Braun, and Ralf Möller. Lifted Temporal Maximum Expected Utility. In *Proceedings of the 32nd Canadian Conference on Artificial Intelligence*, pages 380–386. Springer, 2019.

Marcel Gehrke, Ralf Möller, and Tanya Braun. Taming Reasoning in Temporal Probabilistic Relational Models. In *Proceedings of the 24th European Conference on Artificial Intelligence (ECAI 2020)*, 2020.

Thomas Geier and Susanne Biundo. Approximate Online Inference for Dynamic Markov Logic Networks. In *2011 IEEE 23rd International Conference on Tools with Artificial Intelligence*, pages 764–768, 2011.

Emmanuel Hadoux, Aurélie Beynier, and Paul Weng. Sequential Decision-Making under Non-stationary Environments via Sequential Change-point Detection. In *Learning over*

Multiple Contexts (LMCE), 2014.

Wilfred Keith Hastings. Monte Carlo Sampling Methods Using Markov Chains and Their Applications. *Biometrika*, 57(1):97–109, 1970.

Rob J. Hyndman and George Athanasopoulos. *Forecasting: Principles and Practice*. OTexts, 2018.

Manfred Jaeger and Oliver Schulte. Inference, Learning, and Population Size: Projectivity for SRL Models. In *Eighth International Workshop on Statistical Relational AI*. Statistical Relational AI (StarAI), 2018.

Dominik Jain, Andreas Barthels, and Michael Beetz. Adaptive Markov Logic Networks: Learning Statistical Relational Models with Dynamic Parameters. In *Proceedings of the 2010 Conference on ECAI 2010: 19th European Conference on Artificial Intelligence*, page 937–942. IOS Press, 2010.

Changmin Jiang, Yulai Wan, and Anming Zhang. Internalization of Port Congestion: Strategic Effect behind Shipping Line Delays and Implications for Terminal Charges and Investment. *Maritime Policy & Management*, 44(1):112–130, 2017.

Karsten Keller, Sergiy Maksymenko, and Inga Stolz. Entropy Determination Based on the Ordinal Structure of a Dynamical System. *Discrete and Continuous Dynamical Systems - Series B*, 20(10):3507–3524, 2015.

Karsten Keller, Teresa Mangold, Inga Stolz, and Jenna Werner. Permutation Entropy: New Ideas and Challenges. *Entropy*, 19(3), 2017.

Karsten Keller. Permutations and the Kolmogorov-Sinai Entropy. *Discrete & Continuous Dynamical Systems*, 32(3):891–900, 2012.

Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra. Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases. In *Knowledge and Information Systems*, pages 263–286, 2001.

Kristian Kersting, Babak Ahmadi, and Sriraam Natarajan. Counting Belief Propagation. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, pages 277–284. AUAI Press, 2009.

Kristian Kersting. Lifted Probabilistic Inference. In *Proceedings of the 20th European Conference on Artificial Intelligence*, ECAI'12, pages 33–38. IOS Press, 2012.

Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques – Adaptive Computation and Machine Learning*. MIT Press, 2009.

Stefan Kramer. A Brief History of Learning Symbolic Higher-Level Representations from Data (And a Curious Look Forward). In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 4868–4876, 2020.

Joseph B. Kruskal and Mark Liberman. The Symmetric Time Warping Problem: From Con-

tinuous to Discrete. In *Time Warps, String Edits and Macromolecules: The Theory and Practice of Sequence Comparison*. Addison-Wesley Publishing Co., 1983.

S. Kullback and R. A. Leibler. On Information and Sufficiency. *Ann. Math. Statist.*, 22(1), 03 1951.

Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

Nektarios A. Michail and Konstantinos D. Melas. Sentiment-Augmented Supply and Demand Equations for the Dry Bulk Shipping Market. *Economies*, 9(4):171, 2021.

Pauli Miettinen, Taneli Mielikäinen, Aristides Gionis, Gautam Das, and Heikki Mannila. The Discrete Basis Problem. In Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou (Eds.), *Knowledge Discovery in Databases: PKDD 2006*. Springer, 2006.

Happy Mittal, Ayush Bhardwaj, Vibhav Gogate, and Parag Singla. Domain-Size Aware Markov Logic Networks. In Kamalika Chaudhuri and Masashi Sugiyama (Eds.), *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 3216–3224. PMLR, 2019.

Marisa Mohr, Nils Finke, and Ralf Möller. On the Behaviour of Permutation Entropy on Fractional Brownian Motion in a Multivariate Setting. In *Proceedings of the Asia-Pacific Signal and Information Processing Association Annual Summit and Conference 2020 (APSIPA-ASC)*, pages 189–196, 2020.

Marisa Mohr, Florian Wilhelm, Mattis Hartwig, Ralf Möller, and Karsten Keller. New Approaches in Ordinal Pattern Representations for Multivariate Time Series. In *Proceedings of the 33rd International Florida Artificial Intelligence Research Society Conference*, 2020.

Aniruddh Nath and Pedro Domingos. A Language for Relational Decision Theory. *Graphical Models*, 2009.

Aniruddh Nath and Pedro Domingos. Efficient Lifting for Online Probabilistic Inference. In Maria Fox and David Poole (Eds.), *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, (AAAI 2010)*. AAAI Press, 2010.

Mathias Niepert and Guy Van den Broeck. Tractability through Exchangeability: A New Perspective on Efficient Probabilistic Inference. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI 2014)*, pages 2467–2475. AAAI Press, 2014.

Theo Notteboom. The Time Factor in Liner Shipping Services. *Maritime Economics and Logistics*, 8:19–39, 2006.

Zixuan Peng, Wenxuan Shan, Feng Guan, and Bin Yu. Stable Vessel-Cargo Matching in Dry Bulk Shipping Market with Price Game Mechanism. *Transportation Research Part E: Logistics and Transportation Review*, 95:76–94, 2016.

François Petitjean, Jordi Inglada, and Pierre Gancarski. Satellite Image Time Series Analysis Under Time Warping. *IEEE Transactions on Geoscience and Remote Sensing*, 50(8), 2012.

Albert B. Piek, Inga Stolz, and Karsten Keller. Algorithmics, Possibilities and Limits of Ordinal Pattern Based Entropies. *Entropy*, 21(6), 2019.

David Poole, David Buchman, Seyed Mehran Kazemi, Kristian Kersting, and Sriraam Natarajan. Population Size Extrapolation in Relational Probabilistic Modelling. In Umberto Straccia and Andrea Calì (Eds.), *Scalable Uncertainty Management*, pages 292–305. Springer, 2014.

David Poole. First-order Probabilistic Inference. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pages 985–991, 2003.

Ortwin Renn and Jonathan Paul Marshall. Coal, nuclear and renewable energy policies in Germany: From the 1950s to the "Energiewende". *Energy Policy*, 99:224–232, 2016.

Matthew Richardson and Pedro Domingos. Markov Logic Networks. *Machine Learning*, 62:107–136, 2006.

Joshua W. Robinson and Alexander J. Hartemink. Non-Stationary Dynamic Bayesian Networks. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou (Eds.), *Advances in Neural Information Processing Systems 21*. Curran Associates, Inc., 2009.

Stan Salvador and Philip Chan. FastDTW: Toward Accurate Dynamic Time Warping in Linear Time and Space. In *KDD Workshop on Mining Temporal and Sequential Data*, pages 70–80, 2004.

Scott Sanner and Kristian Kersting. Symbolic Dynamic Programming for First-order POMDPs. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, pages 1140–1146. AAAI Press, 2010.

Cosma Rohilla Shalizi and Alessandro Rinaldo. Consistency under Sampling of Exponential Random Graph Models. *The Annals of Statistics*, 41(2):508–535, 2013.

Diego F. Silva and Gustavo E. A. P. A. Batista. Speeding Up All-Pairwise Dynamic Time Warping Matrix Calculation. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, pages 837–845. Society for Industrial and Applied Mathematics, 2016.

Parag Singla and Pedro Domingos. Lifted First-Order Belief Propagation. In *Proceedings of the 23rd National Conference on Artificial Intelligence*, page 1094–1099. AAAI Press, 2008.

Parag Singla, Aniruddh Nath, and Pedro Domingos. Approximate Lifting Techniques for Belief Propagation. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, page 2497–2504. AAAI Press, 2014.

Inga Stolz and Karsten Keller. A General Symbolic Approach to Kolmogorov-Sinai Entropy. *Entropy*, 19(12), 2017.

Charles Sutton and Andrew Mccallum. Piecewise Training for Structured Prediction. *Ma-*

chine Learning, 77:165–194, 12 2009.

Nima Taghipour, Daan Fierens, Jesse Davis, and Hendrik Blockeel. Lifted Variable Elimination: Decoupling the Operators from the Constraint Language. *Journal of Artificial Intelligence Research*, 47(1):393–439, 2013.

Nima Taghipour, Daan Fierens, Jesse Davis, and Hendrik Blockeel. Lifted Variable Elimination: Decoupling the Operators from the Constraint Language. *Journal of Artificial Intelligence Research*, 47(1):393–439, 2013.

United Nations Conference on Trade and Development. *Review of Maritime Transport 2019*. United Nations, 2019.

Guy Van den Broeck and Adnan Darwiche. On the Complexity and Approximation of Binary Evidence in Lifted Inference. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

Guy Van den Broeck and Mathias Niepert. Lifted Probabilistic Inference for Asymmetric Graphical Models. In Blai Bonet and Sven Koenig (Eds.), *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 3599–3605. AAAI Press, 2015.

Guy Van den Broeck. *Lifted Inference and Learning in Statistical Relational Models*. PhD thesis, KU Leuven, 2013.

Deepak Venugopal and Vibhav Gogate. Evidence-Based Clustering for Scalable Inference in Markov Logic. In Toon Calders, Floriana Esposito, Eyke Hüllermeier, and Rosa Meo (Eds.), *Machine Learning and Knowledge Discovery in Databases*, pages 258–273. Springer, 2014.

Chenggang Wang and J. Schmolze. Planning with POMDPs using a Compact, Logic-based Representation. In *Proceedings of the 17th IEEE International Conference on Tools with Artificial Intelligence*, pages 8–530, 2005.

Zhijie Wang, Xianhua Wu, Kai Lisa Lo, and Jackson Jinhong Mi. Assessing the Management Efficiency of Shipping Company from a Congestion Perspective: A Case Study of Hapag-Lloyd. *Ocean & Coastal Management*, 209:105617, 2021.

Felix Weitkämper. Scaling the Weight Parameters in Markov Logic Networks and Relational Logistic Regression Models. *CoRR*, abs/2103.15140, 2021.

Gerhard Widmer and Miroslav Kubat. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23(1):69–101, 1996.

Dong Yang, Lingxiao Wu, Shuaian Wang, Haiying Jia, and Kevin X. Li. How Big Data Enriches Maritime Research – A Critical Review of Automatic Identification System (AIS) Data Applications. *Transport Reviews*, 39(6):755–773, 2019.

Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding,

Hoang Anh Dau, Diego Furtado Silva, Abdullah Mueen, and Eamonn Keogh. Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View That Includes Motifs, Discords and Shapelets. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 1317–1322, 2016.

Nevin L. Zhang and David Poole. A Simple Approach to Bayesian Network Computations. In *Proceedings of the 10th Canadian Conference on Artificial Intelligence*, pages 171–178. Springer, 1994.

List of Publications

Conference Papers

Nils Finke, Marcel Gehrke, Tanya Braun, Tristan Potten, and Ralf Möller. Investigating Matureness of Probabilistic Graphical Models for Dry-Bulk Shipping. In Manfred Jaeger and Thomas Dyhre Nielsen (Eds.), *Proceedings of the 10th International Conference on Probabilistic Graphical Models*, volume 138 of *Proceedings of Machine Learning Research*, pages 197–208. PMLR, 2020.

Nils Finke, Tanya Braun, Marcel Gehrke, and Ralf Möller. Dynamic Domain Sizes in Temporal Probabilistic Relational Models. *The International FLAIRS Conference Proceedings*, 34, 2021.

Nils Finke and Marisa Mohr. A Priori Approximation of Symmetries in Probabilistic Dynamic Relational Models. In Stefan Edelkamp, Ralf Möller, and Elmar Rueckert (Eds.), *KI 2021: Advances in Artificial Intelligence*, pages 309–323. Springer, 2021.

Nils Finke, Ralf Möller, and Marisa Mohr. Multivariate Ordinal Patterns for Symmetry Approximation in Dynamic Probabilistic Relational Models. In Guodong Long, Xinghuo Yu, and Sen Wang (Eds.), *AI 2021: Advances in Artificial Intelligence*, pages 543–555. Springer, 2022.

Nils Finke, Marisa Mohr, Alexander Lontke, Marwin Zünfle, Samuel Kounev, and Ralf Möller. Recommendations for Data-Driven Degradation Estimation with Case Studies from Manufacturing and Dry-Bulk Shipping. In Samira Cherfi, Anna Perini, and Selmin Nurcan (Eds.), *Research Challenges in Information Science - 15th International Conference, RCIS 2021*, volume 415 of *Lecture Notes in Business Information Processing*, pages 189–204. Springer, 2021.

Nils Finke, Tanya Braun, Marcel Gehrke, and Ralf Möller. Concept Drift Detection in Dynamic Probabilistic Relational Models. *The International FLAIRS Conference Proceedings*, 34, 2021.

Marisa Mohr, Nils Finke, and Ralf Möller. On the Behaviour of Permutation Entropy on Fractional Brownian Motion in a Multivariate Setting. In *Proceedings of the Asia-Pacific Signal and Information Processing Association Annual Summit and Conference 2020 (APSIPA-ASC)*, pages 189–196, 2020

Journal Articles

Nils Finke and Ralf Möller. On the Approximation of Symmetries and Related Structural Changes in Dynamic Probabilistic Relational Models. *Advances in Science, Technology and Engineering Systems Journal*, 7(2), 2022.