The Knowledge Engineering Review, Vol. 25:1, 69–107. © Cambridge University Press, 2010
doi:10.1017/S0269888909990348

Learning Qualitative Differential Equation models: a survey of algorithms and applications

WEI PANG[1,2] and GEORGE M. COGHILL[2]

[1]*Computational Intelligence Group, College of Computer Science and Technology, Jilin University, Changchun, P.R. China;*
[2]*Department of Computing Science, School of Natural & Computing Sciences, University of Aberdeen, Aberdeen, UK;*
e-mail: pang.wei@abdn.ac.uk, pangwei@jlu.edu.cn, g.coghill@abdn.ac.uk

Abstract

Over the last two decades, qualitative reasoning (QR) has become an important domain in Artificial Intelligence. QDE (Qualitative Differential Equation) model learning (QML), as a branch of QR, has also received an increasing amount of attention; many systems have been proposed to solve various significant problems in this field. QML has been applied to a wide range of fields, including physics, biology and medical science. In this paper, we first identify the scope of this review by distinguishing QML from other QML systems, and then review all the noteworthy QML systems within this scope. The applications of QML in several application domains are also introduced briefly. Finally, the future directions of QML are explored from different perspectives.

1 Introduction

1.1 Qualitative simulation

Qualitative simulation (Kuipers, 1986, 1994) is one of the fundamental approaches in qualitative reasoning (QR; Forbus, 1997). It involves predicting the possible qualitative behaviours of complex dynamic systems, which are described as Qualitative Differential Equation (QDE) models. A single QDE can be an abstraction of several Ordinary Differential Equations (ODEs), in the sense that an ODE can be obtained by appropriately parameterizing and quantizing this QDE, and making the function relations of this QDE of an appropriate form (Shen & Leitch, 1993; Coghill & Chantler, 1994). A QDE model is the conjunction of several *qualitative constraints*. These qualitative constraints link the *qualitative variables* in the model and express the relations among these variables. Each qualitative variable is associated with a *quantity space* from which it takes its values. In QSIM (Qualitative SIMulation) (Kuipers, 1994), a quantity space is a finite set of symbols, which has the following form:

$$l_1 < l_2 < l_3 < \ldots < l_n,$$

where each symbol above stands for a qualitatively important value, and is termed as a *landmark value*. The quantity space can also have other forms; for instance, in FuSim (Shen & Leitch, 1993), a quantity space is composed of fuzzy numbers, which are in the form of fuzzy four tuples.

Some types of the qualitative constraints are derived from arithmetic relations, such as qualitative addition, subtraction and multiplication; others represent incomplete knowledge about the function relations under study, for example, in QSIM, there are M^+ (monotonically increasing function) and M^- (monotonically decreasing function) constraints, which state that one variable will monotonically increase with the increase (decrease) of another variable. Table 1 shows some qualitative constraints used in QSIM and their corresponding mathematical equations. In the right

https://doi.org/10.1017/S0269888909990348 Published online by Cambridge University Press

Table 1 Qualitative constraints in QSIM and their corresponding mathematical equations

Qualitative Constraints	Mathematical Equations
ADD(X,Y,Z)	$Z(t) = X(t) + Y(t)$
MULT(X,Y,Z)	$Z(t) = Y(t) \times X(t)$
DERIV(X,Y)	$dX(t)/dt = Y(t)$
MINUS(X,Y)	$Y(t) = -X(t)$
M^+(X,Y)	$Y(t) = f(X(t)), f' > 0$
M^-(X,Y)	$Y(t) = -f(X(t)), f' > 0$

QSIM: Qualitative SIMulation.

column of the table, variables $X(t)$, $Y(t)$ and $Z(t)$ are functions of time t, f is a function that is continuously differentiable over its domain, and f' is the first derivative of f.

Starting from a QDE model, sometimes together with an initial condition of some qualitative variables, a qualitative simulation engine will apply the QR algorithm to obtain a global picture of the dynamics of the system being modelled at a qualitative level. For example, in the case of QSIM, the output is a *behaviour tree*, which is composed of several qualitative states and their transitions. Each *qualitative state* is a complete assignment of values to all the qualitative variables, and a *qualitative behaviour* in QSIM is a path from the root to a leaf in the behaviour tree.

Qualitative simulation is a practical and essential approach because when we lack quantitative knowledge, it is impossible to model a dynamic system by a precise ODE and perform the simulation quantitatively, although it may still be possible to use a QDE. During the past two decades, several qualitative simulation engines have been proposed, such as QSIM (Kuipers, 1986), FuSim (Shen & Leitch, 1993), and Morven (Coghill, 1996).

1.2 QDE model learning (QML)

Qualitative simulation can be performed when QDE models of a dynamic system are available. The QDE models can be obtained either directly from textbooks, for example, textbooks of physical science and biological science, or from the suggestion of the domain experts, such as biological scientists, physical scientists, and electrical engineers. However, for some complex dynamic systems, when little theoretical knowledge is available, even an appropriate QDE model may not be obtained easily. This makes another subfield of QR, QML, necessary. QML involves extracting QDE models of dynamic systems from the available observed data, which can be either quantitative or qualitative. The output of QML can be a single or several QDE models (in the case of hybrid dynamic systems (Alur *et al*., 2000), as described in section 3.3.6). QML is the inverse of qualitative simulation, and it can be considered as a subfield of *system identification* (Ljung, 1999), for which the term *qualitative system identification* is sometimes used (Say & Kuru, 1996). It can also be treated as a special case of *machine learning* (Mitchell, 1997), because some machine learning algorithms, such as *inductive logic programming* (ILP; Bergadano & Gunetti, 1996) can be employed to induce the QDE models. From a model composition point of view, QML can be seen as the process of building 'model fragments' from scratch, or an individual approach for compositional modelling (Keppens & Shen, 2001).

1.3 Challenges of QML

QML is a vital part of QR and can be effectively applied to study many real-world dynamic systems. To implement the distinctive characteristics and powerful learning ability of QML, the designers are confronted with the following four challenges:

1. *Learning from sparse data*: It is known that experimental data acquired from some dynamic systems are often sparse. For example, in some biological experiments, it is often the case that only a few temporal data points can be measured, because of technical limitations and the nature of biological systems.

https://doi.org/10.1017/S0269888909990348 Published online by Cambridge University Press

2. *Learning from imprecise data*: The presence of imprecise data makes it more difficult for QML to distinguish the target model from other candidate models. This imprecise data arise because of measurement errors, or because the dynamic system itself is noisy or imprecise.
3. *Learning from positive data alone*: There are actually no negative data available for learning qualitative models, because it is only possible to observe the behaviours that the dynamic system provides (positive data) and not possible to know the behaviours that the system cannot achieve. This requires corresponding learning algorithms, such as the work of Muggleton (1996), to deal only with the positive data. Although some QML systems, such as GOLEM (Muggleton & Feng, 1990), did make use of 'negative data', these data are obtained based on the assumption that all the possible behaviours of a system are known, and consequently all the other behaviours can be seen as 'negative data'.
4. *Learning under the existence of hidden variables*: Another important factor that influences the learning is the existence of hidden variables. Hidden variables are those variables that are not observed in the experiments. To identify the target model successfully, we must postulate possible hidden variables in the learning process. This makes the learning task more challenging.

1.4 The scope and organization of this review

In some literature, the term *Learning Qualitative Models* is used to cover a set of algorithms, which includes not only learning QDE models, but also learning models that have other qualitative representations, for example, qualitative trees (Bratko & Šuc, 2003). In addition, some model learning systems use qualitative approaches to facilitate the learning task. These systems are quite similar to QML to some extent, but the final output of these systems are either ordinary and partial differential equations, for example, PRET (Bradley & Stolle, 1996; Bradley *et al*., 2001) and the LAGRANGE family (Džeroski & Todorovski, 1993, 1994; Todorovski & Džeroski, 1997; Todorovski *et al*., 2000; Todorovski, 2003), semi-quantitative differential equation (SQDE) models, for example, SQUID (Semi-QUantitative system IDentification) (Kay *et al*., 2000), the work of Vatcheva *et al*. (2005) and Khoury *et al*. (2007), or fuzzy rule-based systems, for example, the work of Bellazzi *et al*. (1998).

In this paper, we restrict our scope to learning QDE models. The remainder of this paper is organized as follows: In section 2, a generalized structure for a QML system is proposed; in sections 3 and 4, the noteworthy QML systems are examined; in section 5, all the reviewed QML systems are summarized; in section 6, the applications of QML are briefly introduced; possible future directions for QML research are suggested in section 7; and finally, some conclusions are presented in section 8.

2 The generalized structure of QML systems

The generalized structure of a QML system is shown in Figure 1. The rectangles stand for the basic execution components, and the rounded rectangles stand for the data and information. Generally speaking, a QML system is composed of the following four abstract execution components:

1. *Data pre-processing*: This component will take the original raw data as input, and produce the processed data. In most cases, this module is a quantitative to qualitative data transformation (Q2Q) algorithm. In some learning systems, this module may be a specialized pre-processing algorithm, such as the method used in QMN (Qualitative Models from Numerical traces) (Džeroski & Todorovski, 1994).
2. *Model space generation*: The model space is the set that contains all possible models. The *model space generation* will take the processed data as input, which can be either qualitative or quantitative, and provide a way of listing all models in the model space. Considering that the size of the model space is often very big, in most cases the *model space generation* will only provide a *rule* for generating all possible models, rather than actually list all possible models exhaustively. In this situation, the model space is implicit. For instance, in GENMODEL (Coiera, 1989a), after the execution of *model space generation*, all qualitative constraints that

https://doi.org/10.1017/S0269888909990348 Published online by Cambridge University Press

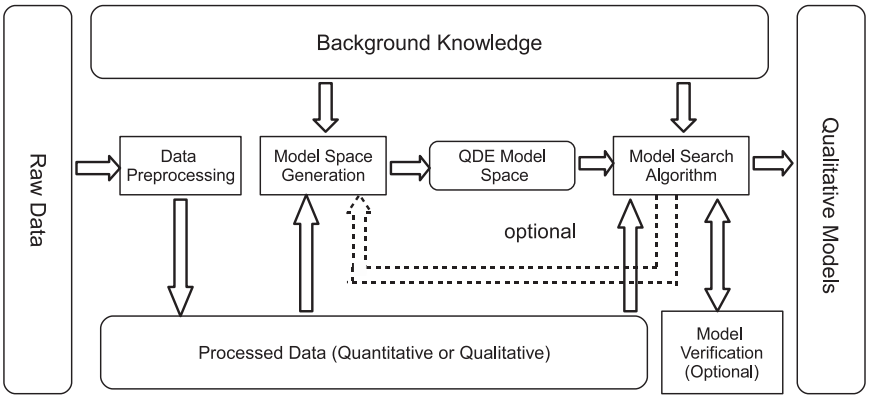


Figure 1 Basic components of the Qualitative Differential Equation model learning systems

are consistent with the given data are generated, and the model space is indicated as all possible combinations of these qualitative constraints.

3. *Model search algorithm*: This component will perform the search in the generated QDE model search space by means of some learning strategy. The output of the model search algorithm will be all the consistent QDE models, which can reproduce the observed data and agree with the *background knowledge* (in this section).
4. *Model verification*: This component is called by the model search algorithm to verify the candidate models and can be a qualitative simulation engine that is embedded into the learning system, for example QSIM, or other error-based verification algorithms.

QML systems also need *background knowledge*, which can reduce the search space, and make the learning easier. The background knowledge can be problem-independent. For instance, the QSIM theory (Kuipers, 1994), dimensional information (Bhaskar & Nigam, 1990), and well-posed model assumptions (Coghill *et al.*, 2008) can be encoded into the background knowledge. The background knowledge can also be problem-dependent. When learning a QDE model in a specific domain, the domain knowledge can also be part of the background knowledge. For example, in a typical biochemical reaction system, a plausible reaction only makes and breaks a few bonds, and it also conserves matter and atom type (Valdés-Pérez, 1994). The domain knowledge will further narrow down the search space and reduce the computational complexity. The background knowledge can be used by either the *Model Space Generation* or the *Model Search Algorithm*.

The output of the QML systems is all the QDE models that do not violate the background knowledge, and that can cover all the given data. When only incomplete data are provided, the output may be composed of many candidate models, and the target model will be among them.

Finally, it should be pointed out that some QML systems, such as QSI (Qualitative System Identification) (Say, 1992; Say & Kuru, 1996), may execute the *Model Space Generation* and *Model Search Algorithm* components in an iterative way, as the dashed arrows indicate in Figure 1.

3 Symbolic QML systems

All the noteworthy QML systems, which have different implementations of basic components of QML and different input data, are examined and described in detail in this and the next sections with reference to the generalized structure proposed in section 2. We argue that the difference in the search strategies is the main reason for the existence of various QML systems. Furthermore, all these search strategies fall into the following two categories: symbolic and evolutionary approaches.

Symbolic approaches tend to explore the search space in a systematic and deterministic manner. The simplest symbolic approach is the generate-and-test algorithm, which straightforwardly generates all possible candidate models and tests them one by one. In the symbolic approaches,

https://doi.org/10.1017/S0269888909990348 Published online by Cambridge University Press

heuristic and pruning techniques can be used to reduce the computational effort, for instance, the Bayesian scoring function (as the heuristic function) and the branch-and-bound algorithm (pruning) used in ILP-QSI (Coghill *et al*., 2008).

Evolutionary approaches employ evolutionary algorithms, such as genetic algorithms (Goldberg, 1989), to perform the search tasks. Because of the nature of these algorithms, evolutionary approaches explore the search space in a non-deterministic manner.

In this section, all the QML systems that employ symbolic approaches as search strategies are introduced in detail. The QML systems that use evolutionary approaches will be described in the next section.

3.1 A general-purpose system: GOLEM

QML systems can also be coarsely divided into two types: general-purpose systems, such as GOLEM (Muggleton & Feng, 1990), and special-purpose systems, such as GENMODEL (Coiera, 1989a). A general-purpose system can fulfil not only the QML tasks, for example, GOLEM but also used for protein secondary structure prediction (Muggleton *et al*., 1992), whereas a special-purpose system is specially designed to fulfil QML tasks only.

Bratko *et al*. (1991) argued that a general ILP (Muggleton & Raedt, 1994; Bergadano & Gunetti, 1996) system, such as GOLEM[1], can achieve the goal of learning QDE models. An ILP system can fulfil the following learning task: given background knowledge B, examples E find a hypothesis H, which satisfies:

$$B \wedge H \vdash E \tag{1}$$

3.1.1 The learning strategy of GOLEM

For learning models with GOLEM, QSIM is first formulated in first order logic form, that is, several Horn Clauses, and then the qualitative constraints in QSIM are converted into ground facts, which are the forms that GOLEM can accept. The learning task becomes to construct a model in terms of Horn Clauses that can cover the positive examples obtained from given system behaviours and exclude the hand-generated negative examples:

$$QSIM\ Theory \wedge QDE\ Model \vdash Examples\ Of\ Behaviours \tag{2}$$

GOLEM employs a bottom-up search strategy by iteratively generalizing a most specific clause. For each iteration, first a certain number of clauses are constructed by *relative least general generalization* (Plotkin, 1971), among which the clause that can cover the maximum number of the positive examples and exclude all the negative examples will be chosen to be processed for the next iteration (see details in Muggleton & Feng (1990)). Hence, GOLEM is not a complete algorithm in the sense that it uses this greedy hill-climbing heuristic search, and cannot guarantee to find the target model if one exists.

3.1.2 Evaluation of GOLEM for learning QDE models

The validity of GOLEM was tested by learning one of the most commonly used benchmark problems in the QR community: the U-Tube system, which is described in Appendix A[2]. It was reported that on learning the U-Tube system, given a behaviour composed of four states, GOLEM can successfully construct two QDE models in terms of Horn Clauses, both of which are equivalent to the original U-Tube model. Furthermore, GOLEM is capable of finding hidden variables by borrowing the name of available variables (atoms in the clauses).

[1] The source code of GOLEM can be downloaded from `http://www.doc.ic.ac.uk/~shm/Software/golem/src.tar`

[2] The experimental data used in GOLEM for learning U-Tube can be accessed from `ftp://ftp.cs.york.ac.uk/pub/ML_GROUP/Datasets/utube/golem/utube.tar.Z`

https://doi.org/10.1017/S0269888909990348 Published online by Cambridge University Press

However, the learning result is based not only on four positive examples, but also six additional hand-generated 'negative examples'. As mentioned before, the use of negative data is not reasonable in practice, because it is not possible to obtain them unless all the possible behaviours of the system are exactly known. In addition, GOLEM needs a large number of ground facts as background knowledge to support its learning, which inevitably limits its scalability. For learning the U-Tube, there are still 5408 ground facts that have to be generated even after making the following simplifications: (i) the corresponding values[3] are assumed empty; (ii) the consistency of infinite values is ignored; and (iii) multiplication functions are taken out of the constraint repertoire. For dynamic systems that involve long lists of landmark values, the number of ground facts will increase dramatically.

GOLEM demonstrates that ILP is suitable for QML tasks. The idea of employing ILP in QML is adopted by many other later special-purpose QML systems, which explicitly or implicitly employ ILP technology.

3.2 GENMODEL

GENMODEL (Coiera, 1989a, 1989b) is probably the first special-purpose QML system, and can be viewed as an ILP-based system, which is able to learn from positive examples only. A later version of GENMODEL (Hau & Coiera, 1993) improves the original version by adding additional dimensional analysis (Bhaskar & Nigam, 1990), fault tolerance, and a Q2Q component. In this section, unless otherwise specified, all the descriptions of GENMODEL are based on this improved version.

3.2.1 Data pre-processing

In contrast to GOLEM, which assumes that the qualitative states are already available, the updated version of GENMODEL possesses the ability to deal with raw numerical data by integrating a Q2Q component. The Q2Q component is made up of two subcomponents: the Front-end Processing module and the Segmenter. The signals pass through the Front-end Processing module, which is composed of several filters, then they are transformed to smooth signals with better mathematical properties. The derivative signals are also obtained by a differentiator. The Segmenter will further divide the smooth signals and the derivative signals, resulting in a complete set of qualitative states.

3.2.2 Model space generation

The model space generation component of GENMODEL is intuitive: the input is a set of qualitative variables (system functions), their dimension information (units), and landmark values. The generation module will exhaustively generate all the possible combinations of the given qualitative constraints and variables, then all the dimensionally inconsistent constraints will be taken out of the constraint set. Under the closed world assumption, the generated model space is complete, which means that it includes all possible models.

3.2.3 Learning strategy

The learning strategy of GENMODEL is straightforward. For each qualitative state obtained from the Q2Q component, all the constraints that are inconsistent with this state will be deleted. After checking all the states, the constraint set is further examined by the redundancy check, which removes the redundant constraints. The redundant constraints are those describing the same relation; for example, $M^+(A, B)$ and $M^-(B, A)$ are redundant, and one of them should be removed. Finally, the rest of the constraints construct a most specific model that can cover all the qualitative states. To deal with noise, a fault tolerance mechanism is added; each constraint is

[3] In QSIM, corresponding values of a constraint are defined as tuples of landmark values that variables in this constraint can take at the same time.

https://doi.org/10.1017/S0269888909990348 Published online by Cambridge University Press

assigned a counter, recording the number of qualitative states that this constraint is inconsistent with. Only a constraint whose counter exceeds a predefined number can be deleted.

3.2.4 Experimental results

On learning the U-Tube, the early version of GENMODEL (Coiera, 1989a) produced 14 constraints from two behaviours that include six states. Other four physical systems described by Kuipers & Kassirer (1984) and Kuipers (1986), including the bathtub and the spring systems, were also tested by the same version of GENMODEL. Experimental results showed that this version of GENMODEL was able to learn these models, but GENMODEL tended to generate overconstrained models when the given data were insufficient. Furthermore, the updated version of GENMODEL (Hau & Coiera, 1993) was successfully applied to a real-world application, which will be detailed in section 6.2.

3.2.5 Evaluation

One of the advantages of GENMODEL is that it does not need negative examples, and the dimensional information can serve as a form of directed negative example generation. It also has the following features: (i) It has the ability to handle real-valued experimental data; (ii) it can deal with noisy data to some extent because GENMODEL introduces the fault tolerance mechanism described in section 3.2.3; (iii) unlike GOLEM, it does not need ground facts; and (iv) it is a complete algorithm, which means that it can guarantee to find the target model if one exists.

The limitations of GENMODEL are as follows: (i) it cannot introduce hidden variables. If not all the system variables are identified, GENMODEL will terminate at a 'shallow model'. (ii) As it tends to find the most specific models, the models obtained are usually overconstrained, that is, it contains more constraints than necessary to fully characterize the dynamic system being modelled.

GENMODEL's primitive idea of generate-and-test is also adopted by later QML systems. These learning systems also added other techniques based on other considerations. For instance, the constraint *determination* stage of QSI (Say & Kuru, 1996) is basically the same as GENMODEL; MISQ (Richards *et al*., 1992) also uses the same method as GENMODEL to generate its initial constraint set.

3.3 MISQ and MISQ-RT

The preliminary version of MISQ (Kraan *et al*., 1991) is similar to GENMODEL in many respects. The later version of MISQ (Richards *et al*., 1992) was re-implemented within a general-purpose relational learning programme: Forte (Richards & Mooney, 1995). This version possesses additional features and can work in some complicated situations that cannot be handled by GENMODEL. There are three phases in MISQ, which correspond to the three components in the generalized structure in Figure 1 separately.

3.3.1 Data pre-processing

This component is called the *Conversion of qualitative data* phase in MISQ. This module can take as input the data from a high-resolution sensor, or quantitative behaviours directly generated by hand, which are composed of those time points where some variables reach extreme values or zero. If the inputs are sensor data, they will be converted to quantitative behaviours by extracting the significant data points. The quantitative behaviours will further be transformed to qualitative behaviours by constructing the landmark values and estimating the derivatives. This phase will be bypassed if the input data are qualitative behaviours.

3.3.2 Model space generation

The Model Space Generation (Constraint *generation* phase in MISQ) is basically the same as GENMODEL, that is, it first generate all the possible combinations of qualitative constraint type and variables, then use dimensional information to filter out the inconsistent constraints.

https://doi.org/10.1017/S0269888909990348 Published online by Cambridge University Press

3.3.3 Learning strategy

MISQ will generate one and only one most specific model if the input qualitative behaviours to this module satisfy the following two conditions: (i) These qualitative behaviours are consistent. A set of behaviours is consistent if these behaviours can be achieved by the same real dynamic system. Note that MISQ does not require complete behaviours. (A set of behaviours is complete if it includes and only includes all possible behaviours that a real dynamic system can achieve.) Hence, this is a more relaxed condition than that in GENMODEL, which requires that the input qualitative behaviours must be complete in order to obtain the right model. (ii) The descriptions of these behaviours are complete, in the sense that there are no hidden variables in the system, and there are also no missing values and dimensions for all the qualitative variables. Considering that condition (i) only requires the consistency of the input qualitative behaviours, the input can be a subset of the complete behaviours, in this case MISQ may produce an overconstrained model.

MISQ can also learn models when the above two conditions are not satisfied, that is, (a) there are hidden variables and (b) there are missing qualitative values or dimensions. In the case of (a), MISQ will use the *relational path-finding* algorithm (Richards & Mooney, 1992) to find the hidden variables. In the case of (b), an inconsistent constraint set may be generated. All the subsets of this constraint set construct a QDE model search space. MISQ will search in this space by employing some heuristics, such as 'the subset must contain at least one derivative constraint' (the system being modelled must be dynamic) and 'form a connected graph' (no disjoint models). These heuristics are also adopted in a later system, ILP-QSI (Coghill *et al.*, 2004, 2008), in which the well-posed models are formally and completely defined to include more properties that a QDE model should have.

3.3.4 Experimental results

On learning the U-tube, MISQ inferred the right model from the only one behaviour, which was the same used in GOLEM as positive examples. Experiments on learning the cascaded tanks (described in Appendix C) were also performed when given data were at different resolutions (qualitative, quantitative, and high-resolution data), or had hidden variables. Results from these experiments showed that MISQ could produce the righ-cascaded tanks model if the dimensional information was complete, and if there were missing variables, MISQ could also infer the right model by finding the hidden variables through the relation path-finding approach.

3.3.5 Evaluation

MISQ goes further than GENMODEL by possessing the ability to process incomplete information and deal with hidden variables. It can work with both quantitative and qualitative data. One of the characteristics of MISQ is that it allows partially specified variables in a qualitative state, which is useful when measuring such variables is not very easy or some points of measurements for them are missing. All of the above make MISQ an advanced learning system that holds an important position in the field of QML.

The limitations of MISQ are as follows: (i) Like GENMODEL, even when complete behaviours are provided, MISQ may produce overconstrained models because it tends to generate the most specific models. (ii) When dealing with incomplete information (a), the heuristics used to reduce the model search space are not completely and formally defined; so additional heuristics can be added, such as causal ordering (Iwasaki & Simon, 1986), which is also used in a later learning system ILP-QSI (Coghill *et al.*, 2004, 2008). This can help it overcome the generation of overconstrained models; (b) no further systematic experiments were performed to support its learning ability, although some examples such as learning the cascaded tanks under a few types of input were demonstrated. (iii) It is not apparent from the descriptions or experimental results of MISQ whether or not it can handle noisy data. The Correctness and Uniqueness Theorem presented only applies for complete noise-free data.

3.3.6 MISQ-RT (MISQ for region transitions)

MISQ-RT (Ramachandran *et al.*, 1994) can be seen as an extended version of MISQ, and it can deal with *hybrid dynamic systems* (Alur *et al.*, 2000). Briefly, a hybrid dynamic system is

https://doi.org/10.1017/S0269888909990348 Published online by Cambridge University Press

composed of several continuous dynamic systems and the switching conditions among them. In the context of qualitative modelling, each of these dynamic systems can be modelled by an individual QDE. Each QDE holds under specific conditions, which are termed *operating conditions* of the QDE, and expressed by the values of the variables involved. The operating conditions specify a 'region' within which this QDE holds, and this region is called the *operating region* (Kuipers, 1994). If a switching condition is satisfied, a *region transition* will occur, which means that the hybrid dynamic system will switch from the operating region of one QDE to that of another.

MISQ-RT fulfils the task of learning hybrid dynamic systems by executing the following four consecutive steps: (i) Detecting region transition points by heuristic methods, such as detecting the discontinuous change of the magnitude and the sign of the derivative. Then, the behaviours are divided by these transition points into segments. Each segment is an operating region, or a subset of an operating region. (ii) Learning QDE models within each segment by MISQ. (iii) Identifying the operating conditions for each region. The operating condition for an operating region is initially set as the conjunction of the range of all the qualitative values for each variable within this region. (iv) Unifying the regions to get a compact description of the model. In the first three steps, for each behaviour segment, a corresponding QDE is obtained. However, some of the segments may have the same QDE model. So it is necessary to unify these segments. Two heuristics are used: (a) The *identical constraints* heuristic will try to unify two regions with the same QDE model, and the operating condition for the newly unified region is updated according to those for the original regions; (b) the *identical operating conditions* heuristic will try to unify two regions with the same operating conditions. The QDE model in the new region is the intersection of the models in the original regions.

As far as we are aware, MISQ-RT is the only QML system that can deal with hybrid dynamic systems.

3.4 QSI

In this section, we will introduce another important QML system, QSI (Say, 1992; Say & Kuru, 1996), which is probably the most complicated QML system. It involves several subtle subfunctions, and constructs a special iterative search space extension and verification process.

3.4.1 Data pre-processing

There is no explicit description of a Q2Q component in QSI. However, QSI introduces a not fully implemented qualitative noise filter. This filter tries to smooth the qualitative behaviours by replacing those qualitative states with unnecessarily noisy landmarks with a more general qualitative state. A parameter is specified to indicate the sensitivity of the filter, that is, the length of the qualitative state sequence to be smoothed. If within this given length, a sequence is the so-called *tooth* one, that is, for one or more than one step a variable increases (or decreases) first, then becomes steady, and finally decreases (or increases), the filter will replace this sequence with a single qualitative state.

3.4.2 Initial search space generation

The initial search space generation is implemented by a subcomponent named constraint *determination*, which is also reused by the later search process. Similar to GENMODEL, *constraint determination* tries to combine all the input variables and constraint repertoire to generate all the possible qualitative constraints. In contrast to GENMODEL, first, as QSI can start from a subset of all the system variables and postulate the missing ones, this subcomponent can take even a small fraction of all the system variables as input. Second, as QSI assumes that the input states are complete and noise free, this subcomponent will remove those generated constraints, which are inconsistent with any of the input states. Third, the newly generated qualitative constraints, which are the consequences of the existing constraints, are also ignored.

https://doi.org/10.1017/S0269888909990348 Published online by Cambridge University Press

3.4.3 Learning strategy

As mentioned above, QSI employs a unique iterative search approach over the model space: it will first try to construct the model using only the given variables. If the given variables are only part of all the variables in the system, the constructed intermediate model will be more general than the target model, which means that it will output more behaviours than expected. This model is termed a 'shallow' model in QSI. Then the algorithm will try to find a more specialized model based on this 'shallow' model, which can cover a few other behaviours, which are not included in the input qualitative states, as possible. In order to achieve this, the following three steps are executed iteratively until a satisfactory model is found:

Step 1: The current intermediate model undergoes the *model depth test*. The model depth test employs a slightly modified QSIM simulator to verify the current intermediate model. If this intermediate model is a 'shallow' model, the QSIM simulator will output more behaviours than those input to QSI. In this case, a *model extension* process (*Step 3*) is necessary.

Step 2: If the current model passes the model depth test, a *dimensional consistency* operation will be performed: Additional M^+ constraints are added to the model to eliminate the dimensional inconsistency. In each of these M^+ constraints, one argument is a variable in a dimensionally inconsistent constraint, and another argument is a newly introduced variable (the so-called 'buffer parameter'), which has the same quantity space structure as the other variable in this M^+ constraint but has a different dimension. Lastly, QSI will return the final resulting model.

Step 3: If the current model fails to pass the model depth test, the *model extension* will be performed:

- *Step 3.1*: New hidden variables are postulated. This step can be run in several *postulation modes* to control the computational expense. In the worst case, *full postulation mode*, which postulates new variables by generating the defining constraints[4] from all combinations of the constraint repertoire (except the monotonically increasing and decreasing functions) and available variables, will be employed if no additional knowledge about the model is known. For example, on learning the U-tube, two new variables may be postulated: P_X and P_Y, and their defining constraints are as follows: $DERIV$(*amount* 1, P_X), ADD(*amount* 1, *amount* 2, P_Y).
- *Step 3.2*: The newly postulated variables will be assigned behaviours according to the following two heuristics: the newly assigned behaviour should have the minimum number of qualitative direction changes, and the postulated variables should be constant whenever possible. Thus, a wider and bigger qualitative state set, described by all the variables (including the newly postulated ones), is obtained.
- *Step 3.3*: Extended *constraint determination* will be performed on this qualitative state set. This will return a new model for the next iteration. Again because of computational expense considerations, different search modes are provided; the *full search mode* will try all the combinations, while *half search mode* will only try the combinations involved in at least one old variable.

3.4.4 Learning the U-tube as an example

On learning the U-tube, suppose the given data contained the information about only two variables, the amount of liquid in tanks 1 and 2, *amounts* 1 and 2, a model consist of only one constraint M^-(*amounts* 1 and 2) was first constructed. This model failed to pass the *model depth test* (*Step 1*) as it was too 'shallow'. Then the *model extension* (*Step 3*) was executed. If in *Step 3* the half search mode was applied and only the derivatives of existing variables were postulated, the QDE model after the execution of *Step 3* is shown in Table 2, where P_1 and P_2 are newly postulated variables.

In the second iteration, the model shown in Table 2 passed the *model depth test* in *Step 1*, and the *dimensional consistency* (*Step 2*) was performed subsequently, resulting in a final output model, which is dimensionally consistent.

[4] In QSI, The *defining constraint* of a variable, say variable A, is defined as the constraint of which variable A appears on the left-hand side of the corresponding mathematical equation.

https://doi.org/10.1017/S0269888909990348 Published online by Cambridge University Press

Table 2 The U-Tube QDE after the first iteration of QSI

Qualitative Constraints
M^-(*amounts* 1 and 2)
DERIV (*amount* 1, P_1)
DERIV (*amount* 2, P_2)
ADD (*amount* 1, P_1, *amount* 2)
ADD (*amount* 2, P_2, *amount* 1)

QDE, Qualitative Differential Equation; QSI, Qualitative System Identification.

3.4.5 Evaluation

Possessing most of the desired features of a QML system, QSI can be deemed a state-of-the-art QML system. The unique search algorithm makes QSI a distinctive QML system, and also gives it the ability to deal with hidden variables effectively.

The limitations of QSI are as follows: (i) Unlike MISQ, QSI is unable to handle incomplete input behaviours. QSI requires that the input must be correct and complete, otherwise it may output a wrong model. (ii) It cannot deal with quantitative data. The input must be in qualitative states. (iii) The noisy data processing is not fully implemented. Although reasonable, the qualitative noise filter needs further development. (iv) QSI cannot easily incorporate additional constraints upon the model to guide its search, as ILP-QSI does.

3.5 ILP-QSI

One of the design goals of ILP-QSI (Coghill *et al.*, 2004, 2008) is to retain the abilities of MISQ and QSI, focusing specifically on non-classical system identification for metabolic systems. ILP-QSI adopted the framework of ILP, but unlike GOLEM, the ILP system employed is essentially a modification of C-Progol, which is one of the best-known ILP instantiations. The general principles underlying the implementation of C-Progol are detailed in Muggleton (1995). In particular, ILP-QSI uses the same cost function as C-Progol to evaluate the candidate models, which is computed using a Bayesian posterior probability estimate that does not require any negative examples of system behaviour (McCreath, 1999). This means that ILP-QSI can learn only from the positive examples. Unlike C-Progol, which returns a single model, in ILP-QSI, a set of all models with the lowest cost is returned. In the search process, only the acceptable models (namely the well-posed models, which will be described later) are evaluated, and this can reduce the evaluation number of the computationally expensive Bayesian estimation. All these modifications are achieved by using the software environment provided by ALEPH (Srinivasan, 1999), which is a general-purpose tool that allows the development of specialized ILP systems by using customized procedures for search and evaluation.

3.5.1 Data pre-processing: Q2Q

The core algorithm of ILP-QSI can learn from only qualitative data, but quantitative to qualitative conversion is also provided. The central difference approach (Shoup, 1979) is used to estimate the first and second derivatives of a quantitative variable. The following simple formulae are used:

$$\frac{dx_i}{dt} = \frac{(x_i - x_{i-1}) + (x_{i+1} - x_i)}{2} \tag{3}$$

$$\frac{d^2x_i}{dt^2} = \frac{(x_i - x_{i-1}) - (x_{i+1} - x_i)}{2} \tag{4}$$

In the above formulae $i = 2,3,\ldots,N-1$, and N is the number of input time points. Then a Blackman filter (Blackman & Tukey, 1958) is used to smooth the estimated first and second derivatives. Finally, a quantitative variable x is converted to a qualitative variable $q = \langle qmag, qdir \rangle$, where *qmag* is generated from x and *qdir* is generated from dx/dt. The qualitative derivative

https://doi.org/10.1017/S0269888909990348 Published online by Cambridge University Press

of q and q', is obtained in a similar way, whose *qmag* and *qdir* are generated from dx/dt and d^2x/dt^2, respectively.

3.5.2 *Learning strategy*

As mentioned above, the search algorithm of ILP-QSI is implemented within the ALEPH system, and is a variant of the branch-and-bound algorithm (Papadimitriou & Steiglitz, 1982). The search strategy is detailed as follows:

- In ILP-QSI, a QDE model is represented as a definite clause.
- Constraints upon the model, for example, the well-posed model constraints (described in the next section), are assumed to be encoded in the background knowledge.
- The algorithm starts from an empty set, while maintaining a prioritized queue *Active*, which is initially empty and later will store the explored models, which are sorted by the cost function.
- The cost of function of a model is calculated from

$$f_{Bayes}(C, B, E) = -P(C|B, E) \tag{5}$$

where $P(C|B,E)$ is the Bayesian posterior probability estimate of clause C, given the background knowledge B and positive examples E. The model with the maximal posterior probability is approximated by maximizing the following function (McCreath, 1999):

$$Q(C) = \log D_H(C) + p \times \log \frac{1}{g(C)} \tag{6}$$

where D_H is a prior probability, g is the generality of a model, p is the number of positive examples. These functions and parameters can be calculated in the manner of C-Progol (see Muggleton (1996) for details).

- Successor Function: The unexplored nodes are expanded by a successor function, which can efficiently enumerate the well-posed models. The successor function is defined to return a set of all the well-posed models when the current node is empty and return empty when the current node is not empty. All the returned nodes are added to the branch set.
- Branching: (i) A node k from the queue *active* is popped out and evaluated by the above-defined cost of function. (ii) The cost of the best node so far, *best*, is updated if necessary. (iii) Node k is expanded by the above-mentioned successor function. All the expanded nodes are added to the branch set.
- Bounding: Prune the branch set according to the cost *best* and obtain a bounded set, then add the elements in the bounded set to the queue *active*.
- The branch-and-bound operation executes iteratively until the queue *active* is empty or the number of explored nodes exceeds a pre-specified number.

3.5.3 *Well-posed models*

The concept of the well-posed model was proposed by analyzing the properties of a class of dynamic systems in depth, and it reveals the internal mechanism of these systems; on the other hand, well-posed model constraints can reduce the search space; the branch-and-bound algorithm will only expand and evaluate those nodes, which satisfy the well-posed model constraints. A well-posed model must satisfy the following syntactic constraints:

- *Size*: The size of the model must be specified; this is obtained by counting the number of qualitative constraints in the model.
- *Complete*: The model must include all the observed variables.
- *Determinate*: The model must contain as many qualitative relations as qualitative variables.
- *Language*: The number of instances of any qualitative relation in the model must be below some pre-specified limit. Detailed discussion can be found in Camacho (2000).

The well-posed model must also satisfy the following semantic constraints:

- *Sufficient*: The model must adequately explain the observed data.
- *Redundant*: There are no redundant relations in the model.

https://doi.org/10.1017/S0269888909990348 Published online by Cambridge University Press

- *Contradictory*: There are no contradictory relations in the model.
- *Dimensional*: All the constraints in the model must be dimensionally consistent. The details of dimensional analysis can be seen in Bhaskar & Nigam (1990).

In addition, the following constraints are preferred but not mandatory:

- *Single*: The model must not contain disjoint models, that is, the target model must describe the qualitative behaviours of a dynamic system within the same operating region.
- *Connected*: All the hidden variables should appear in at least two qualitative constraints in the model.
- *Causal*: The model must be causally ordered (Iwasaki & Simon, 1986; Wiegand, 1991).

3.5.4 Experimental design and results

Using the above-mentioned search strategy, together with the well-posed model assumption, a series of experiments are tested.

Learning without noise: Noise-free data were provided to the learning system and the experiments were tested by the following three steps: (i) Obtain the complete envisionment (Kuipers, 1994) of the system with specified values of exogenous variables[5]. (ii) Power Set Experiments: all the non-empty subsets of the qualitative states in the complete envisionment were treated as training data. If the number of qualitative states was N, the number of experiments was $2^N - 1$. For small N, ILP-QSI exhaustively tested on these training data sets. For large N, a fixed number of experiments was randomly sampled from the power set of all the qualitative states. (iii) The learning reliability was measured by the precision of learning from these training data sets. The average precision under different sizes of training data was obtained and visualized.

Learning in the presence of noise: Non-empty subsets of the states in the complete envisionment were replaced by randomly generated noisy qualitative states. All the possible replacements, with the number of $2^N - 1$, were tested and again the average precision over different sizes of noisy data was visualized.

Kernel subsets and solution space analysis: With noise-free data, a *kernel subset* in ILP-QSI is a subset of the complete qualitative states, and is composed of minimal qualitative states required for successful learning (i.e. the target model can be successfully identified in an experiment). The meaning of this is twofold: first, any superset of a kernel subset will also lead to successful learning; second, any subset of a kernel subset will fail to identify the target model. For example, on learning the U-Tube, the kernel subsets are eight pairs of qualitative states. This means that in all the successful experiments, the training data must be a superset of at least one of these eight pairs.

A further investigation by analyzing the solution space of the U-Tube and coupled tanks (described in Appendix B) reveals that the qualitative states in the kernel subset include at least one critical point and these states cover different branches in the envisionment graph (Coghill *et al*., 2008).

3.5.5 Evaluation

ILP-QSI is an ILP-based QML system, which retains all the characteristics of the earlier QML systems, such as MISQ and QSI. In addition, ILP-QSI has its own new features: (i) the concept of well-posed models is explicitly and completely defined; (ii) it can learn in the presence of noise; (iii) It can avoid finding overconstrained models; and (iv) it can be used to learn complex biological systems. The metabolic pathway of glycolysis was chosen to test the learning ability of ILP-QSI on large systems, and this will be described in section 6.4. All of the above make ILP-QSI a state-of-the-art QML system. In addition to the above-mentioned new features, a novel contribution of ILP-QSI to the QML research is the kernel subset experiments that were performed.

[5] Exogenous variables are those variables determined from outside the model.

https://doi.org/10.1017/S0269888909990348 Published online by Cambridge University Press

There are, however, a number of ways in which ILP-QSI could be improved: (i) Like many other QML systems, it has not been tested to deal with the dynamic systems with multiple operating regions. (ii) Another problem is the computational limitation of ILP-QSI for learning complex systems. In identifying the pathway of glycolysis, several months of compute time have to be used. This will also be discussed in section 6.4.

3.6 Learning directly from numerical data I: QMN

In the previously introduced learning systems, the final training data are qualitative. Even when the input data are quantitative, a Q2Q component will convert the quantitative data to qualitative states. Rather than doing this, there are two QML systems that can directly learn models from quantitative data, without a Q2Q transformation. In this section, we will introduce one of them, that is, QMN (Džeroski & Todorovski, 1994). In the next section, the other system, LYQUID (Gerçeker & Say, 2006), will be described.

3.6.1 Data pre-processing

As QMN directly learns the QDE models from numerical data, the data pre-processing is quite simple; it mainly involves estimating the time derivatives of the system variables. Given the time series points x_i ($i = 1,2,\ldots,N$), the derivatives are obtained by the following formula from (Bohte, 1991):

$$\dot{x}_i = \frac{1}{12 \times h}(x_{i-2} - 8 \times x_{i-1} + 8 \times x_{i+1} - x_{i+2}) \tag{7}$$

where h is the time interval between two time points. Given the highest order o, QMN will generate all the o derivatives for a variable using the above formula. Then the time series points and their derivatives are directly used by QMN. It should be pointed out that there is no noise processing in the pre-processing.

3.6.2 Learning strategy

The core algorithm of QMN uses the generate-and-test strategy, and is based on GENMODEL, except that it can introduce new variables in a simple manner. (i) First, all the variables and their derivatives are added to an initial set V. (ii) *Introduce new variable*: Given the model depth d, new variables will be iteratively introduced by combining the variables and their derivatives in V. Four basic arithmetic operations: addition, subtraction, multiplication, and division are used to combine any two elements in V. The newly introduced variables are added into V for the next iteration. (iii) *Generate-and-test*: Given two tolerance parameters δ and ε, any possible combination of QSIM constraints and variables will be generated and tested. All the generated constraints which are dimensionally consistent are transformed to their equivalent constraints in the form *zero(T)*, which holds when the term T is considered as a constant zero given the tolerances δ and ε. For example, *add(X,Y,Z)* is converted to *zero(Z−X−Y)*, and if *zero(Z−X−Y)* holds, the original constraint *add(X,Y,Z)* also holds. Then the test of the validity of a constraint is converted to test whether *zero(T)* holds or not. This is achieved by testing the validity of $P(|T| > \delta) < \varepsilon$, where P is the probability function, which can be approximated by calculating the percentage of the measurements in which $|T| > \delta$ holds over the total number of measurements.

3.6.3 Evaluation

Instead of learning from qualitative data, QMN can directly handle numerical data, and this is achieved by introducing two tolerance parameters and the corresponding test method. The highest order of the system o and model depth d determine the complexity of the search space. Careful selection of these parameters is very important for learning.

Like GENMODEL, QMN tends to generate most specific constraints; this often leads to an overconstrained model. Experiments on the U-Tube and cascaded tanks system (described in Appendix C) showed that not only redundant constraints, but also some unexpected constraints were generated. It was reported that for the U-Tube with $d = 2$, there were 362 constraints generated. At this point, a further improvement for QMN is required to reduce the number of

https://doi.org/10.1017/S0269888909990348 Published online by Cambridge University Press

constraints generated. Another limitation of QMN is that it needs a large amount of high-resolution quantitative data. For learning the U-Tube, 1000 qualitative data points had to be provided to QMN. This restricts its learning ability when only sparse data are available.

3.7 Learning directly from numerical data II: LYQUID

In this section, another learning system which can learn QSIM-like models directly from numerical data, LYQUID (Gerçeker & Say, 2006), will be discussed. Rather than working on time series data points in QMN, LYQUID uses orthogonal polynomials, which are functions of time, to approximate the numerical input data. This approach is taken from Ralston & Rabinowitz (2001). Then the model discovery process is purely performed on these polynomials, and the input numerical data points are discarded.

3.7.1 Data pre-processing: interval partition and polynomial approximation

The data pre-processing is a significant part of LYQUID. For each of the system variables, given the observed data samples, together with the time points of measurements, the *interval partitioning* algorithm will try iteratively to partition the time intervals into subintervals, and approximate the variable within these intervals by individual orthogonal polynomials. All the intervals on which the square error of the current polynomial is greater than a specified parameter *Tolerance* will be dealt with by the *interval partitioning* algorithm.

When the *interval-partitioning* algorithm tries to split a time interval, first it temporarily splits this interval into two subintervals from the midpoint, then uses two new polynomials to fit on these two subintervals separately. If the improvement of the precision upon the original interval is greater than the pre-specified *parameter improvement*, or the improvement of precision on either of the two subintervals is greater than the pre-specified parameter *split improvement*, this split is accepted; otherwise, the split is rejected and the original interval will be kept. The precision of the polynomial in an interval is calculated by the square error of this polynomial for approximating the variable within the interval over the length of the interval. The newly split intervals will also be recorded for further processing by the partitioning algorithm. The partitioning algorithm will terminate when no further intervals can be split.

3.7.2 Learning strategy

Like QMN and GENMODEL, LYQUID uses the generate-and-test strategy; all the possible combinations of qualitative constraints and variables are tested. Similar to QMN, testing whether a qualitative constraint holds or not is equivalent to testing whether two polynomials are equal, or strictly speaking, their difference is within a tolerance interval. LYQUID uses the following formula to measure the difference between two polynomials p and q on a time interval (a and b):

$$e = \frac{\int_b^a [p(t) - q(t)]^2 dt}{b - a} \tag{8}$$

If e is less than a specified parameter *iTolerance*, polynomials p and q are deemed to be equal. *iTolerance* is the specified tolerance square error per unit time. As in QMN, the parameter o specifies the highest order of a model. The derivatives of a variable are generated by differentiating the corresponding polynomial that represents this variable. As LYQUID evaluates the validity of a qualitative constraint on many time intervals, and this qualitative constraint may not hold on all the time intervals, the ratio $R = s/k$ is calculated by LYQUID, where s is the number of intervals on which the constraint holds, and k is the total number of intervals. Additional parameters have to be set to determine whether the constraint holds or not, and if the ratio R falls into the range determined by these parameters, the qualitative constraint can be considered to hold on all time intervals.

The steps of the LYQUID core algorithm are as follows: (i) For all the observed variables, add the corresponding polynomials and the dimensional information into the set of variables V. (ii) Hidden variable generation: generate the polynomials for the derivatives, sums, and products of the variables in V. Add these newly generated polynomials into V. (iii) Check all the possible

https://doi.org/10.1017/S0269888909990348 Published online by Cambridge University Press

combinations of QSIM constraints and the variables in V, adding the consistent constraints into the constraint set C. (A *consistent constraint* is defined as the constraint that agrees with the data. More specifically, a consistent constraint in LYQUID is the constraint in which the difference between its corresponding polynomials on both sides is within the tolerance interval.) The algorithm will finally return the constraint set C.

3.7.3 Experiments

LYQUID was tested on the U-Tube, cascaded (see Appendix C) and coupled tanks (see Appendix B), the same physical systems used in ILP-QSI. The algorithm was tested on both clean and noisy data samples. In addition, the experiments on the data, which are obtained from irregular sampling frequencies, and the experiments on the missing data were tested.

3.7.4 Evaluation

LYQUID, as a QML system, which can learn models directly from numerical data, uses polynomials as a tool to fulfil the learning task. Because of the nature of polynomial approximation, LYQUID has the following features: (i) it is suitable for dealing with noisy data; (ii) the variables can be sampled at different frequencies; and (iii) It can also work on incomplete data, or even arbitrary sampled data.

A major limitation of LYQUID is that many parameters have to be specified properly by the users. These parameters are even more than those in QMN. Another limitation is that, like GENMODEL and QMN, LYQUID often generates overconstrained models with unnecessary constraints.

3.8 Learning simplified models: Abe's algorithm

In this section, we introduce a simplified QML system developed by Abe (1993). In Abe's system, the following assumptions are made:

- There is only one landmark value, *zero*. Then the qualitative values based on this quantity space are defined, such as: [0] stands for landmark value *zero*; [−] stands for the interval $(-\infty,0)$; [0+] stands for $[0,+\infty]$; and [+−] stands for $(-\infty,0)$ $(0,+\infty)$.
- There are no hidden variables, and each variable in the target model is a differentiable function of time defined in Euclidean space.
- Time is quantized, and the input data are in the form of a time series of qualitative values. For example, for a system with two variables, x_i and x_j, the possible input sequence is shown in Table 3, where t_i $(i = 1,2,\ldots)$ is a time interval or a time point.
- There are no missing or noisy data for any variables in the input sequence.

3.8.1 Data pre-processing

On the basis of the above assumptions, a set of rules has been established to achieve the following two goals: (i) distinguishing the time intervals and moments from the input sequence; and (ii) estimating the qualitative states. The latter is carried out by estimating the possible qualitative derivatives according to the input time series of qualitative values. Two state estimation rules have been set up, depending on whether the current quantized time is a time interval or a time point. Table 4 shows part of the State Estimation Rule 1, when the current quantized time is a time point. Table 5 is part of the State Estimation Rule 2, when the current quantized time is a time interval.

Table 3 Possible input sequence in Abe's system

Time	t_0	t_1	t_2	t_3	t_4	……
$x_i(t)$	[−]	[0]	[+]	[+]	[+]	……
$x_j(t)$	[+]	[+]	[+]	[0]	[−]	……

https://doi.org/10.1017/S0269888909990348 Published online by Cambridge University Press

Table 4 Part of the State Estimation Rule 1 in Abe's System

$[Q(t_{prev})]$	$[Q(t_{now})]$	$[Q(t_{next})]$	$\partial[Q(t_{now})]$
[+]	[0]	[+]	[0]
[+]	[0]	[0]	[0]
[+]	[0]	[−]	[−0]
[0]	[0]	[+]	[0]
…	…	…	…
…	…	…	…

Table 5 Part of the State Estimation Rule 2 in Abe's System

$[Q(t_{prev})]$	$[Q(t_{now})]$	$[Q(t_{next})]$	$\partial[Q(t_{now-s})]$	$\partial[Q(t_{now-m})]$	$\partial[Q(t_{now-e})]$
[+]	[+]	[+]	[+0−]	[+0−]	[+0−]
[+]	[+]	[0]	[+0−]	[−0]	[−]
[0]	[+]	[+]	[+]	[0+]	[+0−]
[0]	[+]	[0]	[+]	[0]	[−]
…	…	…	…	…	…
…	…	…	…	…	…

In Table 4, $[Q(t_{now})]$ stands for the qualitative value at the current time point; $[Q(t_{prev})]$ and $[Q(t_{next})]$ stand for the qualitative values on the previous and succeeding time intervals separately. $\partial[Q(t_{now})]$ is the estimated derivative. In Table 5, $[Q(t_{now})]$ is the qualitative value within the current time interval, so $[Q(t_{prev})]$ and $[Q(t_{next})]$ are both qualitative values on time points. $\partial[Q(t_{now-s})]$ is the estimated derivative in the interval immediately after the time point t_{prev}; $\partial[Q(t_{now-e})]$ is the value in the interval immediately before the time point t_{next}; $\partial[Q(t_{now-m})]$ is the value in the time interval between t_{now-s} and t_{now-e} and Q(t) is assumed to be strictly monotonic in t_{now-m}.

By applying the State Estimation Rules 1 and 2 on the input sequence data, all the possible qualitative states can be obtained.

3.8.2 Model space generation

All the possible qualitative differential equations, which are different from those used in QSIM, are represented by the following expressions:

$$[0] \subset \sum_{m=0}^{order} [C_{k1,\ldots,kn,l1,\ldots.ln}] \prod_{i=1}^{n} [Q_i]^{k_i} \prod_{j=1}^{n} \partial Q_j^{l_j}$$
$$m = k_1 + \ldots + k_n + l_1 + \ldots + l_n$$

where, $\subset$ means 'subset'. For example, $[Q] \subset [0+]$ means $[Q] = [0]$ or $[Q] = [+]$. $\sum$ and $\prod$ means qualitative addition and multiplication, respectively. *order* is the highest order of the polynomial, and $[C_{k1,\ldots,kn,l1,\ldots.,ln}]$ are the coefficients. All possible qualitative differential equations indicated by equation (9) are tested for validity by all the estimated qualitative states, which are obtained by the two state estimation rules described in section 3.8.1. Finally, the validated constraints that are consistent with the estimated qualitative states constitute the search space.

3.8.3 Learning strategy

The learning task is to find the so-called Simultaneous Differential Equations, which are the conjunction of the qualitative differential equations in the above search space, to construct a model. Abe's system uses the concept of *Entropy Reduction* (Cellier, 1991) as a heuristic function

https://doi.org/10.1017/S0269888909990348 Published online by Cambridge University Press

to select the appropriate Simultaneous Differential Equations. A bigger value for *entropy reduction* means a stronger forecasting power. *Entropy reduction* is defined as:

$$H_r = 1.0 - \frac{logk}{2nlog3} \tag{10}$$

where n is the number of variables and k is the number of qualitative states that satisfy the model. When n is fixed, a smaller k will result in a larger entropy reduction H_r. Then the models that can generate fewer states will be chosen as the solution models.

3.8.4 Experimental results

A randomly generated input sequence of qualitative values with two variables is selected as the test example. It is reported that six models are found. All these models are transformed to QSIM models and simulated by QSIM. They can reproduce the same data as the input.

3.8.5 Evaluation

Abe's QML system is based on several assumptions and particular conditions, and these assumptions and conditions can reduce the complexity of the learning system and simplify the learning task. On the other hand, because the algorithm restricts itself to the above conditions, it will encounter some difficulties when applied to some real applications that do not obey the above conditions. For example, when the given data are sparse, incomplete, or there are hidden variables, the algorithm will not be able to learn the right model.

3.9 An incremental learning approach: QML-IL

In this section, we discuss an incremental approach to QML (Srinivasan & King, 2008). Again for the sake of description, we term this system as QML-IL, which stands for 'QML system using an Incremental Learning approach'. QML-IL is also an ILP-based system and can be considered as an extension of ILP-QSI, which was introduced in section 3.5.

3.9.1 An incremental decomposition strategy

QML-IL uses a simple incremental decomposition strategy to decrease the computational complexity of ILP-QSI. The learning task is divided into several stages and the decomposition is achieved by domain-specific knowledge. Each stage is provided with the following parameters: (i) the set of variables involved in this stage; (ii) the number of constraints in this stage.

The background knowledge B and examples E are transformed into a sequence $\langle B_m,E_m \rangle$, where $m = 1,2,\ldots,n$, and n is the total number of stages. Each of the elements in this sequence, $\langle B_i,E_i \rangle$, is provided to the corresponding stage i. Where E_i contains the observations for the variables involved from stage 0 to stage i, and B_i is the background knowledge that involves the corresponding variables in E_i.

In each stage, say, stage i, given $\langle B_i,E_i \rangle$, and H_{i-1}, which is the set of partial models that are identified in stage $(i-1)$, the learner L will output the set of partial models H_i. This process can be denoted as the following formula:

$$H_i = L(B_i, E_i, H_{i-1}, \rho, f), \tag{11}$$

where $i = 0,1,\ldots,n$, and n is the total number of stages; f is the cost function as described in equation (5). H_0 is an empty set, and H_n is the set of final models. ρ is the successor function defined as follows: Given each element k in H_{i-1}, $\rho(k,B_i,E_i)$ will return all models that obey the well-posed model constraints and the number of qualitative constraints in the current stage.

The implementation within the learner L is exactly the same as ILP-QSI, which uses the branch-and-bound search strategy. In addition, before the set of partial models, H_{i-1}, is submitted to L in stage i, there is an optional operation that performs a limited generalization on each model in H_{i-1}. This is implemented by disconnecting some or all components in each model, but respecting the original constraints provided in this model. In ILP, this is called removing variable co-references.

https://doi.org/10.1017/S0269888909990348 Published online by Cambridge University Press

After the learner L returns a set of models H'_i in stage i, there is another optional operation: randomly selecting a certain number of models from H'_i, and these randomly selected models will compose H_i, which is used for the input of the next stage.

For example, for learning the coupled tanks system described in Appendix B, the learning task can be divided into two stages by assuming that tanks 1 and 2 are relatively separated (domain-specific knowledge): The first stage concerns only the variables related to tank 2, and tries to identify the models that are consistent with the observations; the second stage concerns all the variables and tries to learn the final model based on the models found in stage 1.

3.9.2 Automatic decomposition

In some real-world problems, the decomposition of a dynamic system may not be made appropriately, and thus the automatic decomposition algorithm is required. The decomposition task is actually to determine the partition sequence (S_0, S_1, S_2,….,S_n) on the set of all the given system variables, namely S, where S_i ($i = 1, 2,\ldots,n$) is a subset of S, and $S = S_0 \bigcup S_1 \bigcup S_2 \ldots \bigcup S_n$.

A randomized local search algorithm, which is similar to GSAT (Selman *et al.*, 1992), is used to perform the automatic decomposition. The steps are briefly described as follows:

Suppose the first $(i-1)$ subsets of variables from S for the first $(i-1)$ stages have been selected: $S_0, S_1, \ldots, S_{i-1}$; $S_0 = \phi$; $VarsLeft = S$, when $i = 0$; and $VarUsed = S_0 \bigcup S_1 \ldots \bigcup S_{i-1}$. Then for the selection of S_i:

1. First, randomly select a subset V from the unselected variables $VarLeft = S\text{-}VarUsed$ for current decomposition.
2. Determine the cost of this selection c by the incremental learner L. c is the cost of the model which has the least cost in H'_i, and $H'_i = L(H_{i-1}, \mathrm{B}, \mathrm{E}, VarUsed \bigcup V)$. Updating the current best cost, *bestcos,t* by c and current selection, *VarsSelected,* by V, if c is lower than *bestcost*.
3. $BestLocal = V$.
4. Construct all the possible 'local moves' for *BestLocal*. This is implemented by removing one variable from V and adding another variable, which is not included in V.
5. Determine the cost of all these moves and selecting V', which has the least cost c'.
6. Update *bestcost* and *Varselected* with c' and V', if c' is lower than *bestcost*.
7. $BestLocal = V'$.
8. 4–7 are executed M times.
9. 1–8 are executed R times.
10. $S_i = VarsSelected$.

In the above steps, M and R are very important parameters. M is the 'depth' of the local moves and R is the number of restarts of the randomized procedure. An initial investigation of the influence of these two parameters is also provided: both the coupled tanks system and other three artificial systems are tested by the automatic decomposition algorithm. The experimental result indicates that (i) generally, the algorithm will perform better if $R \geq M$; (ii) when the number of variables in the system or the number of stages is increased, R and M should also be increased.

3.9.3 Experiments and evaluation

Besides the coupled tanks system, the models of different biological systems were also tested. They are listed as follows: (i) predator–prey model; (ii) human lung model; and (iii) glycolysis pathway (the same model is used in ILP-QSI).

QML-IL employs a new approach to QML. It demonstrates that using the incremental decomposition strategy, the learning efficiency can be improved dramatically. This makes QML-IL suitable for learning models of complex dynamic systems, such as biological systems.

A major limitation of QML-IL is that the set of variables and the number of constraints must be provided to each stage by the decomposition solution, and this is difficult in some cases. Although some preliminary work has been carried out on automatic decomposition, further research and experiments are needed to investigate the adaptability and scalability of the decomposition

https://doi.org/10.1017/S0269888909990348 Published online by Cambridge University Press

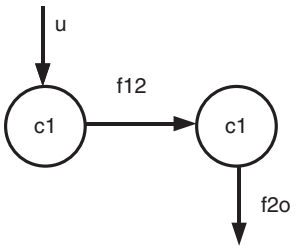


Figure 2 A cascaded two-compartment model

QSIM model	JMorven Model
	Differential Plane 0
M^+ (c1, f12)	*func* (dt 0 f12, dt 0 c1)
M^+ (c2, f2o)	*func* (dt 0 f2o, dt 0 c2)
ADD (q1, f12, u)	*sub* (dt 1 c1, dt 0 u, dt 0 f12)
ADD (q2, f2o, f12)	*sub* (dt 1 c2, dt 0 f12, dt 0 f2o)
DERIV (c1, q1)	*Differential Plane 1*
DERIV (c2, q2)	*func* (dt 1 f12, dt 1 c1)
	func (dt 1 f2o, dt 1 c2)

Figure 3 The QSIM and JMorven descriptions of the cascaded two compartment model

algorithm. In addition, the incremental learner uses the same search strategy as ILP-QSI, so it has the same limitations. This suggests an interesting line of research.

3.10 Advanced experiments on learning QDE models by QML-BKFC

QML-BKFC (Pang & Coghill, 2007b) is part of a QML framework, which is named as QML-Morven in this paper for the sake of description. QML-Morven uses the Morven formalism (Coghill, 1996; Bruce, 2007) to represent and verify QDE models, rather than QSIM. QML-BKFC uses the Backtracking with Forward Checking algorithm (BKFC) as its search strategy. Another part of QML-Morven, which is termed as QML-CSA (Pang & Coghill, 2007a), uses the clonal selection algorithm (CSA) as the search strategy. QML-CSA will be described in section 4.2.

3.10.1 Model representation

Morven and JMorven: Morven (Coghill, 1996) is a state-of-the-art QR engine, which possesses all the benefits of QSIM and introduces many useful features. The introduction of differential planes (Wiegand, 1991) and vector envisionment (Morgan, 1988; Coghill, 1992) makes it convenient to reason about more than two derivatives. By introducing Fuzzy Set theory (Zadeh, 1965), each qualitative variable in Morven is associated with a fuzzy quantity space, which makes it possible for Morven to perform fuzzy vector envisionment (Coghill, 1996) and deal with fuzzy qualitative data. JMorven (Bruce & Coghill, 2005; Bruce, 2007) is a Java implementation and extension of the original framework of Morven.

Models represented in JMorven: The QSIM and JMorven Model (up to two diffrential planes) for a two-compartment model (described in Figure 2) is shown in Figure 3. In the figure, c1 and c2 denote the concentrations in compartments 1 and 2, respectively; f12 stands for the flow from compartments 1–2; *u* and f2o are the input and output flows, respectively. In a JMorven model, Plane 0 contains the constraints that construct the actual dynamic system. The constraints in Plane 1 are obtained by differentiating the corresponding constraints in Plane 0. The quantity spaces of the qualitative variables are composed of trapezoidal fuzzy numbers, which are represented by four tuples (see Shen & Leitch (1993) or Coghill (1996) for details of this presentation).

https://doi.org/10.1017/S0269888909990348 Published online by Cambridge University Press

3.10.2 Pre-processing phase

The learning process can be divided into two phases: the pre-processing and the backtracking search phases. The pre-processing phase is introduced first.

The pre-processing phase of QML-BKFC includes four steps: (i) *Constraint generation*: This is similar to GENMODEL, except that given the maximum number of hidden variables, it also generates the qualitative constraints for the hidden variables. (ii) *Constraint filtering*: All the generated constraints in step (i) are checked for consistency with the input qualitative data. (iii) *Precalculation*: In this phase, the dependency and conflict relations of the remaining constraints are calculated and recorded. (iv) *Constraint set partition*: The filtered constraints are divided into several subsets, each of these subsets contains all the defining constraints[6] for the same system variable[7]. Then the learning task becomes selecting only one constraint from each of the above subsets to construct a candidate model, then check the validity of this model. The correctness of this selection is guaranteed by a theorem (Pang & Coghill, 2007b), which states that under the assumptions being made in ILP-QSI (Coghill *et al*., 2004, 2008), a well-posed model as defined in ILP-QSI contains one and only one defining constraint for each system variable. By this theorem, the search space can be reduced from exponential to polynomial size.

3.10.3 Learning strategy

After the constraint *set partition* in the pre-processing phase, as described in section 3.10.2, all the remaining constraints construct the refined search space *FCS* (Filtered Constraint Set), and these constraints are divided into several subsets, each of which contains all the defining constraints for the same variable. Let DS be a set that takes each of these subsets as an element, denoted by $DS = \{S_1, S_2, \ldots, S_N\}$, where N is the number of variables (including hidden variables), and for any two elements in DS, $|S_i| \leq |S_j|$, if $i \leq j$ $(i,j \in 1 \ldots N)$.

The learning task also becomes that of selecting only one constraint from a different S_i in DS to construct a candidate model, then checking the validity of this model. The backtracking algorithm will incrementally add constraints from different subsets in DS. The subset that has the minimum number of *legal constraints* will be explored first. The *legal constraints* are defined as the qualitative constraints in *FCS*, which do not conflict with all the constraints in the current partial model. Once a new qualitative constraint C_i from a subset, say S_i in DS, is added into the current partial model PM, $(PM + C_i)$ will be checked for validity by the well-posed model constraints; if $(PM + C_i)$ is a *full* model and passes the well-posed model check, it will be further simulated by JMorven (Bruce, 2007) to test the coverage. If $(PM + C_i)$ fails to pass the check, C_i will be excluded in order to avoid searching for hopeless nodes (the models which include $PM + C_i$). The algorithm then backtracks to the previous node (PM), and selects the next legal constraint in the same subset S_i to continue the search. If no further legal constraints can be selected in S_i, the algorithm will further backtrack to the previous node, $(PM - C_j)$, where C_j is the most recent constraint that was added into PM.

An auxiliary forward checking method is also performed, that is, when a new constraint is added, all the constraints in *FCS* that conflict with this new constraint will be ignored in the later search process, as they are *illegal constraints* for the current partial model. The algorithm will not terminate until the search space has been explored exhaustively.

3.10.4 Advanced experiments

On the basis of the proposed algorithm, some advanced experiments, which extend the experiments in ILP-QSI, were performed to investigate the learning reliability and kernel subsets (as defined in ILP-QSI) under different experimental conditions. These conditions are categorized as

[6] The concept of defining constraint is taken from QSI. In QML-Morven, a defining constraint for a variable is the constraint in which this variable or its derivatives appear in the leftmost position.

[7] The system variables are all the endogenous variables in a model.

https://doi.org/10.1017/S0269888909990348 Published online by Cambridge University Press

(i) the specification of the state variables: [8]fully, partially, or not specifying the state variables; (ii) different number of hidden variables: from zero to the number that leads to unsuccessful learning; (iii) different types of hidden variables, making state or non-state variables become hidden.

Two kinds of two-compartment model were selected for the test: The first model is a closed system with no input and output, but with bidirectional flows; the second model is the cascaded compartmental system as shown in Figure 2. QML-BKFC performed the experiments on these two models under the above-mentioned conditions. The learning reliability and kernel set were obtained for each condition in each model. Some meaningful conclusions are obtained according to the experimental result: (i) The state variables are very important for learning. If some of them become hidden variables, the learning task becomes difficult. (ii) Partially or not specifying the state variables will result in a large search space and may lead to unsuccessful experiments. (iii) The non-state variables have relatively less influence on learning, compared with the state variables. (iv) Too many hidden variables can lead to failure to learn.

3.10.5 Evaluation

QML-BKFC performs a systematic search in the model search space, and is a complete algorithm. It is suitable for small- and medium-size problems. Using this complete algorithm, some reliable experimental results can be obtained and this helps us better understand how different factors can influence the QML. Furthermore, as QML-BKFC uses the Morven framework, it has the potential to deal with fuzzy data.

The major limitation of QML-BKFC is its scalability, compared with QML-CSA to be described in section 4.2. The systematic search will become time consuming or even impractical for large-size problems.

4 Evolutionary QML systems

Learning QDE models is essentially a search task, which involves searching in a QDE model space. This task can be fulfilled by different approaches, not only ILP and other symbolic ones, but also evolutionary algorithms, such as genetic algorithms (Goldberg, 1989) and genetic programming (Koza, 1992). In this section, we introduce two QML systems that employ evolutionary approaches to search in the model space: QME (Qualitative Model Evolution) and QML-CSA (Qualitative Model Learning using a CSA).

4.1 QME

In this section, a genetic algorithm (GA) for learning QDE models, QME (Varšek, 1991), is discussed. In QME, as there is no description of the data pre-processing, the input data are assumed to be qualitative. So, we will begin with the introduction of its learning strategy.

4.1.1 Learning strategy

QME employs a modified GA to search in the model space. The output of QME is a set of QSIM-like models. In order to reduce the complexity, the QDE models under study are simplified as follows: (i) An additional 'RANGE' constraint is added to the QSIM constraint repertoire. This constraint is to indicate the legal range of the qualitative variables. (ii) Like GOLEM, the corresponding values are ignored. (iii) In the U-Tube model, the variable that denotes the difference between two levels, together with its related monotonically increasing constraint are also ignored. The simplified U-Tube model is shown in Figure 4. In addition, the following conditions are also assumed: (i) no hidden variables; and (ii) all the landmark values are known beforehand.

[8] In a causally ordered model, the state variables are those variables whose magnitudes only appear on the right-hand side of the qualitative constraints, and whose first derivatives must appear at least once on the left-hand side of some qualitative constraints.

https://doi.org/10.1017/S0269888909990348 Published online by Cambridge University Press

RANGE(La, 0...∞/dec...inc)
RANGE(Lb, 0...∞/dec...inc)
ADD(Fab, Lb, La,[])
MINUS(Fab, Fba,[])
DERIV(La,Fba)
DERIV(Lb,Fab)

Figure 4 The simplified U-Tube model

Encoding strategy: QME uses *binary trees* to encode the individuals (chromosomes): The leaf nodes denote the qualitative constraints and the branching points represent the hierarchical structure. This encoding method is quite similar to the approach taken in genetic programming (Koza, 1992).

Crossover and mutation: The crossover operation is performed on two parents: randomly choosing the crossover points for both parents and exchanging the subtree below the crossover point. The mutation is implemented by randomly choosing a mutation node and replacing the subtree below this node by a randomly generated one.

Fitness calculation: The fitness value for each chromosome is calculated by the following three steps: (i) *Raw fitness calculation*: Like GOLEM, QME uses both positive examples and hand-generated negative examples. The raw fitness function is the sum of the following two ratios: the ratio of the covered positive examples to all the positive examples, and the ratio of the excluded negative examples to all the given negative examples. If the above two ratios are both equal to 1, an additional bonus score, which is a decreasing function of the model size, will be added to the raw fitness. (ii) *Fitness sharing*: In order to prevent the premature convergence of the population, raw fitness is shared by similar chromosomes. The phenotype similarity function is used for calculating the shared fitness, that is, the similarity function for any two models is determined by the number of examples (positive and negative) that both models can cover. (iii) *Linear scaling*: After the fitness sharing, all the fitness values undergo a linear transformation, such that after scaling, the average of these fitness values remains unchanged and the most fit individual is assigned the value of a certain multiple (namely the scaling factor) of the average fitness.

On the basis of the above-mentioned operators, the GA will iteratively explore and exploit the search space in a beam search manner, until the target models are found or the maximum number of generations is exceeded.

4.1.2 Experiments

It is reported that to learn the U-tube, 17 positive and 78 negative examples were used, and QME found five equivalent solutions. Other physical systems, such as the spring system and resistor-capacitor (RC) circuit system, were also tested. In these experiments, QME always found the model that covered all the positive data and excluded all the given negative data.

4.1.3 Evaluation

QME is an evolutionary approach that utilizes a GA to fulfil the learning task. The experimental results show that GA, as a general-purpose searching and learning method, has potential utility in the field of QML. On the basis of the *building block hypothesis* (Goldberg, 1989), some meaningful sets of qualitative constraints (building blocks) will emerge if appropriate genetic operations are performed, which makes the search efficient. As the chromosomes vary in size, QME does not need to specify the size of the target model in advance, and in the search process, the algorithm will find the solution model with the appropriate size. The inherent parallelism of GA makes the search more efficient.

Although it benefits from the advantages of GA, QME also suffers from the limitations of GA: (i) It is not a complete algorithm, this means that QME cannot always guarantee to find the target model. (ii) Premature convergence of the population may happen if the related parameters, especially the crossover and mutation probability, are not chosen appropriately. In addition to the

https://doi.org/10.1017/S0269888909990348 Published online by Cambridge University Press

influences of GA, there are also other limitations: Like GOLEM, negative examples have to be used; the model has to be simplified by, for example, ignoring the corresponding values.

4.2 QML-CSA

In this section, we will introduce another evolutionary approach, the CSA (de Castro & Von Zuben, 2000), for learning QDE models—QML-CSA (Pang & Coghill, 2007a). As mentioned in section 3.10, QML-CSA is another part of the QML-Morven framework. The Model Presentation and Pre-processing component are the same as QML-BKFC, as described in sections 3.10.1 and 3.10.2, respectively. The description of QML-CSA will skip these two components and start from the learning strategy.

4.2.1 Evolutionary search

A CSA (de Castro & Von Zuben, 2000, 2002; de Castro & Timmis, 2002) is used for searching in the model space. CSA is a population-based evolutionary algorithm inspired by the clonal selection theory (Burnet, 1959) of the immune system.

When negative examples are not available and the positive data are complete, the landscape of the model space may have numerous local optima, each of them satisfying the well-posed model constraints, but not all of them generating exactly the same data as the given ones (this means some of the models are underconstrained and some are overconstrained). In addition, when only incomplete data are provided, there may be more than one candidate solution that can cover the incomplete input data. CSA is particularly suitable for dealing with the above situations, because in the above cases, the search space is highly multi-modal and CSA has proven to be efficient for searching in the multi-modal problem space.

The basic idea is that each antibody is encoded to represent a potential model, and in the process of the evolutionary search the antibodies are expected to find the target models.

Encoding strategy: Rather than the binary string encoding used in many conventional evolutionary algorithms, QML-CSA uses an integer encoding strategy. In QML-CSA, each antibody is composed of several slots, and each slot corresponds to a subset, which contains all the defining constraints for the same variable. The value of each slot, which is an integer, indicates a qualitative constraint selected from the corresponding subset. The correctness of this encoding strategy is also guaranteed by the same theorem as was used in QML-BKFC (see section 3.10). In addition, to avoid the antibody getting trapped in a local optimum for an intolerable length of time, a survival time is associated with each antibody. If an antibody exceeds its survival time, it will be replaced by a randomly generated antibody. Figure 5 shows an example of antibody encoding for the two-compartment model in Figure 2. In this antibody, S_0, S_1, and S_i contain all the defining constraints for f12, f2o, and the hidden variable *Hid0*, respectively.

Each antibody in the antibody repertoire (population) will be cloned to create several copies (*clonal expansion*). All the cloned antibodies, forming a temporary population, will be submitted to the 'hyper-mutation' scheme. The hyper-mutation is correspondingly modified due to the modification of the antibody encoding strategy. The modified hyper-mutation operator will replace the value of each slot in the antibody with a randomly generated integer, and the range of this randomly generated integer is from 1 to N (N is the number of constraints in the corresponding subset). This means that each constraint in the model will be replaced by a randomly selected constraint within the same defining subset. In the search space, each mutated antibody can be considered as a neighbourhood of the original antibody.

The *affinity evaluation* will consider the following aspects (as in ILP-QSI): *model conflict*, *dimensional consistency*, *model language*, *model connection*, *model completeness*, *model singularity*, and *causal ordering and coverage*. These criteria establish a scoring system for evaluating the affinities of the antibodies, such that the model satisfying more criteria get a higher score.

The *re-selection* process is based on the above scoring system. The antibodies, which have higher scores and have not exceeded their survival time, will be selected to enter the next generation's population (antibody repertoire).

https://doi.org/10.1017/S0269888909990348 Published online by Cambridge University Press

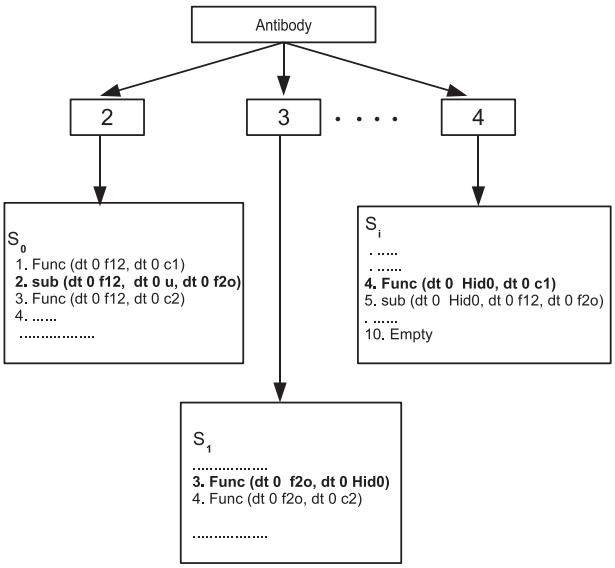


Figure 5 Antibody encoding

The modules described above, clonal expansion, hyper-mutation, affinity evaluation, and re-selection, are the basic components of CSA. CSA will iteratively execute these components until it finds the target model or exceeds the maximum number of generations.

4.2.2 Experimental design and results

Two kinds of experiments were designed to test the reliability and scalability of QML-CSA.

- *Reliability tests*: For simple models with small sets of training data, the same two kinds of two-compartment models as those used in QML-BKFC were selected. The learning reliability of the algorithm was tested by the power set experiments, as in ILP-QSI. Thus, the kernel subsets and learning curve, defined by Coghill *et al.* (2008; see section 3.5.4), were obtained.
- *Scalability tests*: For more complex models, a three-compartment and a four-compartment model were selected. They are the extensions of the model in Figure 2. The scalability of QML-CSA was tested by providing complete data but with different hidden variables.

Both of the above experiments were performed by QML-CSA and QML-BKFC[9]. QML-BKFC served as a verification algorithm to test the correctness of QML-CSA for small-size problems, and were also used to compare the efficiency of QML-CSA. The experimental results showed the following two conclusions: (i) For the same power set experiments on small-size models, both QML-BKFC and QML-CSA obtained the same results as in ILP-QSI, that is, the same learning reliability and kernel subsets. (ii) For more complex models, QML-CSA was much more efficient than QML-BKFC, especially when the number of hidden variables was increased. For example, for learning the three-compartment model with three hidden variables, it took QML-BKFC more than 7 days to find the target model, and for QML-CSA, it was approximately 11 hours, which is a 15-fold improvement[10].

4.2.3 Evaluation

QML-CSA is another evolutionary approach to learn QDE models. Unlike QME, it does not need negative examples, as the dimensional information and well-posed model constraints serve the

[9] All the experiments were performed on a computer cluster with eight compute nodes, each of which has two Opteron 850 (2.4 GHz) CPUs and 4 GB RAM.

[10] The result is based on the average value of 10 trials.

https://doi.org/10.1017/S0269888909990348 Published online by Cambridge University Press

function of negative examples to some degree. QML-CSA can perform the same kernel set experiments as ILP-QSI and obtain consistent results that demonstrates its reliability. More importantly, compared with QML-BKFC, a major advantage of QML-CSA is its scalability, which enables it to learn large-scale models more efficiently. In addition, like QML-BKFC, it also has the potential ability to deal with fuzzy data because of the adoption of the Morven framework.

The limitations of QML-CSA are as follows: (i) Like ILP-QSI, as it is based on the well-posed model constraints, it cannot learn a model which does not satisfy all the properties of well-posed models; for example, it does not deal with hybrid dynamic systems. (ii) Like QME, QML-CSA is not a complete algorithm; it does not guarantee to find the target model if one exists.

5 A summary of the QML systems

In this section, we summarize the QML systems introduced above, and provide more explanations about these systems for the readers.

Table 6 shows the characteristics of all the systems. It should be pointed out that for some systems, the absence of some functions or modules does not mean that they have worse performance, as they may focus on solving some particular problems in this field. These characteristics are listed to provide a global view of the QML systems, and show which problems have been solved in this field.

A few explanations about these systems are given as follows: (i) In theory, all QML systems can take more than one behaviours as input, although for the following four systems, GOLEM, Abe's system, QMN, and LYQUID, there are no explicit experiments to demonstrate this. The more behaviours input to the QML system, the higher chance we have to obtain the right model. (ii) The following QML systems are state-based systems, which means that they can take states as input across different behaviours, rather than individual behaviours: ILP-QSI, QML-IL, QML-BKFC, and QML-CSA. (iii) Most of the QML systems are not available on the Web except GOLEM. However, the following systems, QSI, ILP-QSI, and QML-Morven, can be obtained from their authors.

6 Applications of QML

In this section, the application domains of QML will be briefly introduced. Generally speaking, QML can, in principle, be used to help us better understand, control, or diagnose any dynamic systems of interest, especially systems for which the data are imprecise or sparse. In particular, the following application domains are selected and introduced in detail.

6.1 Benchmark physical systems

As early research on QR mainly focused on the study of the *de facto benchmark* physical systems, such as the U-Tube (see Appendix A), the coupled tanks (see Appendix B), the cascaded tanks (see Appendix C), and the spring-mass system (see Coghill *et al*. (2008)), QML correspondingly takes these systems as learning objects.

These systems are representative and can be considered as abstractions of the models in many other disciplines, and they can be highly nonlinear. For example, the coupled tanks systems can be treated as a high-level abstraction of metabolic systems (Coghill *et al*., 2004); the spring-mass system is an abstraction of many oscillatory systems, which exist not only in mechanical systems, but also biological, electrical, chemical, and even social systems.

6.2 Clinical applications

Besides the *benchmark systems*, there are also some other dynamic systems that have been learnt by QML systems. Hau and Coiera (1993) applied an updated version of GENMODEL to

https://doi.org/10.1017/S0269888909990348 Published online by Cambridge University Press

https://doi.org/10.1017/S0269888909990348 Published online by Cambridge University Press

Table 6 A Summary of the characteristics of different QDE model learning systems

Algorithm	Pre-processing	Positive Data Only	Hidden Variable	Noisy Data	Formalism	Basic Search Strategy	Kernel Subsets	HVA[a]	MOR[b]
GOLEM	×	×	✓	×	QSIM	ILP	×	×	×
GENMODEL	Q2Q	✓	×	✓[c]	QSIM	RLGG	×	×	×
MISQ	Q2Q	✓	✓	×	QSIM	Generate-and-test	×	×	×
MISQ-RT	Q2Q	✓	✓	×	QSIM	Separate-and-Reunite	×	×	✓
QSI	Q2Q[d]	✓	✓	✓[d,e]	QSIM	Adjustable Postulation	×	×	×
ILP-QSI	Q2Q	✓	✓	✓[e]	QSIM	ILP and Branch-and-Bound	✓	×	×
QMN	NP[f]	✓	✓	✓[c]	QSIM	Generate-and-test	×	×	×
LYQUID	NP[f]	✓	✓	✓[c]	QSIM	Generate-and-test	×	×	×
QME	×	×	×	×	QSIM	Genetic algorithm	×	×	×
QML-CSA	×	✓	✓	×	Morven	Clonal selection algorithm	✓	×	×
Abe's system	Q2Q	✓	×	×	SDE[g]	Entropy reduction heuristic search	×	×	×
QML-IL	×	✓	✓	×	QSIM	ILP and incremental decomposition	×	×	×
QML-BKFC	×	✓	✓	×	Morven	Backtracking with forward checking	✓	✓	×

QDE, Qualitative Differential Equation; ILP, inductive logic programming; QSI, Qualitative System Identification; QMN, Qualitative Models from Numerical traces; QME, Qualitative Model Evolution; IL, Incremental Learning; BKFC, Backtracking with Forward Checking; RLGG, Relative Least General Generalization; QSM, Qualitative SIMulation.

[a]Hidden variable analysis: analysis of the influence of different number and types of hidden variables.

[b]Multiple operating regions.

[c]Quantitative level noise processing.

[d]Not fully implemented.

[e]Qualitative level noise processing.

[f]Numerical processing.

[g]Simultaneous Differential Equations.

learn a qualitative cardiovascular (CVS) model. In that CVS model, the following eight variables are involved:

1. HR: heart rate.
2. $ABPM$: mean arterial blood pressure.
3. $CVPM$: mean central venous pressure.
4. SV: stroke volume.
5. CO: cardiac output.
6. VC: ventricular Contractility.
7. ΔT: skin-to-core temperature gradient.
8. RPP: rate pressure product.

And the following qualitative constraints exist:

1. $CO = HR \times SV$
2. $RPP = HR \times ABPS$
3. $M^+(CVPM, CO)$: the Frank–Starling law of the heart.
4. $M^+(SV, VC)$: autoregulation of the heart.
5. $M^+(HR,CO)$ and $M^+(HR,VC)$: under normal heart rate.
6. $M^-(HR,VC)$ and $M^-(HR, SV)$: when the heart rate exceeds the critical level.
7. $M^-(CO, \Delta T)$, $M^-(CO, HR)$ and $M^-(CO, VC)$: when hypovolemia happens.

The quantitative data segments for the above eight variables were obtained from six patients. Each of these segments was 1000-second long, and the data points were sampled in every second. Then these quantitative data were converted to qualitative states by the Q2Q module of GENMODEL, as described in section 3.2.1. The qualitative data were submitted to GENMODEL to get the qualitative constraints. The experiments showed that GENMODEL successfully reproduced the right qualitative constraints. Some spurious constraints were also generated because of the lack of data.

6.3 Applications in ecology

6.3.1 Plant water-balance system

A plant water-balance system was proposed for study by Ramachandran *et al.* (1994). This model describes how the plant balances the amount of water in its body when the amount of water in its external environment changes. Four QDE models were used to model this system. In order to learn this model, two qualitative behaviours, one of which contains 16 qualitative states, were provided to MISQ-RT.

MISQ-RT initially generated eight QDE models. After the unification module, six QDE models were obtained. A further analysis showed that these six models were equivalent to the four QDE models that were generated by the domain experts.

6.3.2 Predator–prey model

Learning the predator–prey model, which is also called the Lotka–Volterra model (Lotka, 1925), was studied by QML-IL (Srinivasan & King, 2008). The ODEs of the simplified model are as follows:

$$\frac{dN}{dt} = g(N) - c(P) \tag{12}$$

$$\frac{dP}{dt} = c(P) - d(P) \tag{13}$$

In the above, N and P are the population of prey and predators, respectively; $g(N)$ is the growth rate of the prey, which is a linear function with the constant coefficient; $c(P)$ is the rate of predation upon the prey, and $d(P)$ is the death rate of the predators. If $c(P)$ and $d(P)$ are assumed to be a monotonically increasing function of P, the above ODE model can be converted to a QDE model. Then the qualitative data provided to QML-IL are generated by simulating this QDE model.

https://doi.org/10.1017/S0269888909990348 Published online by Cambridge University Press

QML-IL used a one-stage decomposition (which meant not decomposing the model at all) to learn the model. After running QML-IL, five models were returned, one of which was the target model.

6.4 Applications in cell biology

6.4.1 Learning the glycolysis pathway

The QDE model of the glycolysis pathway was learnt by ILP-QSI (King *et al.*, 2005) and QML-IL (Srinivasan & King, 2008). The glycolysis pathway is one of the most important pathways in biology, which involves 15 metabolites, they are pyruvate (Pyr), glucose (Glc), phosphoenolpyruvate (PEP), fructose 6-phosphate (F6P), glucose 6-phosphate (G6P), dihydroxyacetone phosphate (DHAP), 3-phosphoglycerate (3PG), 1,3-bisphos phoglycerate (1,3BP), fructose 1,6-biphosphate (F16BP), 2-phosphoglycerate (2PG), glyceraldehyde 3-phosphate (G3P), ADP, ATP, NAD, and NADH. As H^+, H_2O, and orthophosphate are assumed to be ubiquitous in the cell, they are not included in the pathway. Figure 6 shows all the reactions involved in the pathway. The first reaction is taken as an example to illustrate the meaning of these reactions; it means that Glc can be converted to G6P, this reaction is catalysed by the enzyme hexokinase. In this reaction, the ATP will be consumed and ADP will be produced.

Modeling metabolites and enzymes: To model this pathway qualitatively, the enzymes and the metabolites are modelled as components. An enzyme is assumed to have at most two substrates and at most two products. If there are two substrates in a reaction, they are considered to form a substrate complex before the reaction, and the amount of the substrate complex is proportional to the amount of the two substrates multiplied together. It is similar with the products. Each enzyme is considered as a component: the 'flow' through this enzyme equals the amount of the substrate complex that enters into it minus the amount of the product complex that leaves from it. Each metabolite is also considered as a component correspondingly. Therefore, the ODE and QDE models for the metabolites and the enzymes are as follows:

Metabolites:

$$\frac{dM}{dt} = \sum_{i=1}^{n} Flow_i \tag{14}$$

DERIV(Metabolites, Mdt),
SUM($Flow_1$, …, $Flow_n$, Mdt).

Enzymes:

$$Flow_i = f(\prod_{s=1}^{S} Metablolite_s) - g(\prod_{p=1}^{P} Metabolite_P) \tag{15}$$

PROD($Metabolite_1$, …, $Metabolite_s$, S),
PROD($Metabolite_1$, …, $Metabolite_p$, P),
M^+(S, Ds),
M^+(P, Dp),
SUB(Ds, Dp, Flow),
MINUS(Flow, $Flow_{minus}$),

1. (Hexokinase): Glc + ATP $\Longleftrightarrow$ G6P + ADP
2. (Phosphoglucose isomerase): G6P $\Longleftrightarrow$ F6P
3. (Phosphofructokinase): F6P + ATP $\Longleftrightarrow$ F16BP + ADP
4. (Aldolase): F16BP $\Longleftrightarrow$ DHAP + G3P
5. (Triose phosphate isomerase): DHAP $\Longleftrightarrow$ G3P
6. (Glyceraldehyde 3-phosphate dehydrogenase): G3P + NAD $\Longleftrightarrow$ 13BP + NADH
7. (Phosphoglycerate kinase): 13BP + ADP $\Longleftrightarrow$ 3PG + ATP
8. (Phosphoglycerate mutase): 3PG $\Longleftrightarrow$ 2PG.
9. (Enolase): 2PG $\Longleftrightarrow$ PEP
10. (Pyruvate kinase): PEP + ADP $\Longleftrightarrow$ Pyr + ATP

Figure 6 The reactions in the glycolysis pathway (from King *et al.* (2005))

https://doi.org/10.1017/S0269888909990348 Published online by Cambridge University Press

where S represents the input substrates and P the products. PROD and SUM are the extensions of MULT and ADD in QSIM, respectively. Based on the above qualitative constraints, more general constraints can be generated to represent the metabolite and enzyme components, as shown below:

$$\text{ENZYME}((S1, S2), (P1, P2), \textit{enzymeFlow})$$
$$\text{METABOLITE}(\textit{metaboliteConc}, \textit{metaboliteFlow}, ([\textit{enzyme}_1\textit{Flow}, \text{sign}], \ldots [\textit{enzyme}_n\textit{Flow}, \text{sign}])),$$

where in the *ENZYME* components, S1 and S2, denote the substrates, P1 and P2 are the products, *enzymeFlow* is the return value of the component, indicating the *flow* through this enzyme, which has been described before. While in the *METABOLITE* component, *metaboliteConc* and *metaboliteFlow* are the concentration and flow of this metabolite, respectively, and the third argument denotes all the enzyme *flows* that can influence this metabolite. The value of the sign can be '+' or '−', which means increase and decrease of this metabolite, respectively. For instance, [$enzyme_1$-*Flow*,+] means the flow through $enzyme_1$ will increase this metabolite.

Modeling the glycolysis pathway: These two kinds of general constraints (components) are finally used to model the glycolysis pathway, as shown in Figure 7. Two constraints are taken as examples to illustrate the meaning of this model. The first constraint, ENZYME((Glcc, ATPc),(G6Pc, ADPc),Enz1f), means that the flow through *Enz1* (hexokinase), *Enz1f*, controls the transformation of the concentrations of *Glc*(Glcc) and *ATP*(ATPc) into *G6Pc* and *ADPc*. The 17th constraint, METABOLITE(PEPc,PEPf,(Enz9f, +),(Enz10f, −)), means that the concentration and flow of PEP, *PEPc* and *PEPf*, are controlled by the flow through Enz9 (Enolase), *Enz9f*, and the flow through Enz10 (Pyruvate kinase), *Enz10f*. Enz9 will produce PEP (according to the sign '+') and Enz10 will remove PEP (according to the sign '−').

Learning process: First, all the possible ways of combining all the metabolites are generated. Then the domain knowledge of chemical and biochemical reactions is used to reduce the search space. Finally, 18 reactions, which are both chemically and biochemically plausible are selected to construct the initial search space. The qualitative data for learning the pathway are obtained by simulating the target model.

After several months of compute time on a cluster computer, 35 possible models (out of more than 27 000 models) of the glycolysis pathway, among which the target model is included, were returned by ILP-QSI. The long compute time is acknowledged, as it is an approach that would not be done in practice; the problem would generally be tackled in stages. It is a proof-of-concept, and shows that ILP-QSI can narrow down the set of the candidate models to a small scope even when some domain-specific knowledge is omitted.

```
ENZYME((Glcc, ATPc),(G6Pc,ADPc),Enz1f),
ENZYME((G6Pc),(F6Pc),Enz2f),
ENZYME((F6Pc,ATPc),(F16BPc,ADPc),Enz3f),
ENZYME((F16BPc),(G3Pc,DHAPc),Enz4f),
ENZYME((DHAPc),(G3Pc),Enz5f),
ENZYME((G3Pc,NADc),(13BPc,NADHc),Enz6f),
ENZYME((13BPc,ADPc),(3PGc,ATPc),Enz7f),
ENZYME((3PGc),(2PGc),Enz8f),
ENZYME((2PGc),(PEPc),Enz9f),
ENZYME((PEPc,ADPc),(Pyrc,ATPc),Enz10f),
METABOLITE(ATPc,ATPf, (Enz10f +), (Enz7f, +),(Enz1f, -),(Enz3f, -)),
METABOLITE(ADPc,ADPf,(Enz1f, +),(Enz3f, +),(Enz10f, -)(Enz7f, -)),
METABOLITE(NADc,NADf,(Enz6f, -)),
METABOLITE(NADHc,NADHf,(Enz6f,+)),
METABOLITE(Pyrc,Pyrf,(Enz10f, +)),
METABOLITE(Glcc,Glcf,(Enz1f,-)),
METABOLITE(PEPc,PEPf,(Enz9f, +),(Enz10f, -)),
METABOLITE(F6Pc,F6Pf,(Enz2f, +),(Enz3f, -)),
METABOLITE(G6Pc,G6Pf,(Enz1f, +),(Enz2f, -)),
METABOLITE(DHAPc,DHAPf,(Enz4f, +),(Enz5f, -)),
METABOLITE(3PGc,3PGf,(Enz7f, +),(Enz8f, -)),
METABOLITE(13BPc,13BPf,(Enz6f, +),(Enz7f, -)),
METABOLITE(F16BPc,F16BPf,(Enz3f, +), (Enz4f, -)),
METABOLITE(2PGc,2PGf,(Enz8f, +),(Enz9f, -)),
METABOLITE(G3Pc,G3Pf,(Enz5f, +),(Enz4f, +),(Enz6f, -)).
```

Figure 7 The qualitative representation of the glycolysis pathway (from (King *et al.* (2005))

https://doi.org/10.1017/S0269888909990348 Published online by Cambridge University Press

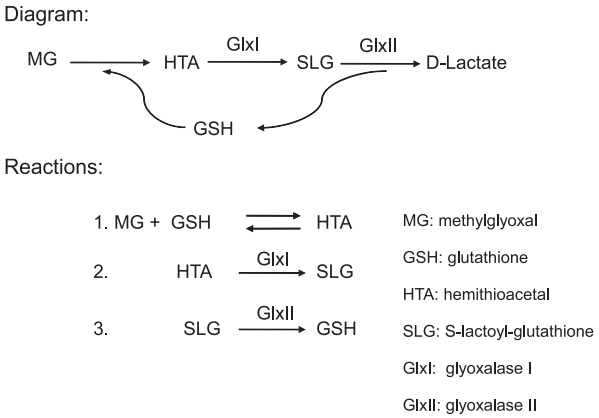


Figure 8 The detoxification pathway of methylglyoxal (based on Pang & Coghill (2009))

For QML-IL, the glycolysis pathway was reconstructed in three stages, namely *priming*, *splitting*,and *phosphorylation*. The *priming* stage includes the first three reactions shown in Figure 6; the splitting stage includes the fourth and fifth reactions; and the rest of the reactions belong to the phosphorylation stage. By this decomposition, QML-IL learnt the model stage by stage. It was reported that the computation times for different stages were 6 seconds in stage 1135 seconds in stage 2 and 5296 seconds in stage 3. After the three-stage learning, only two models were returned, one of which is the target model.

6.4.2 Learning the detoxification pathway of methylglyoxal

QML-Morven was used to learn the qualitative model of the detoxification pathway of methylglyoxal (Pang & Coghill, 2009). Methylglyoxal is a naturally occurring toxic electrophile, which is harmful to cells. Its detoxification pathway was initially studied by Ferguson *et al.* (1998), and current understanding of this pathway is shown in Figure 8. In this pathway, the first reaction follows the mass action law, and the second and third are enzymatic reactions, which are taken to obey Michaelis–Menten kinetics (Michaelis & Menten, 1913). Glyoxalase I and II are the enzymes that catalyse the second and third reactions, respectively.

Research on this pathway is still ongoing, and current understanding is based on incomplete knowledge and limited experimental results. Consequently, the quantitative analysis (de Almeida *et al.*, 2008) based on current understanding cannot precisely describe the dynamics of the system. This makes it helpful to reconstruct this pathway at a qualitative level using QML-Morven.

The converting algorithm: On the basis of the standard assumption that all non-enzymatic reactions obey the law of mass action and all enzymatic reactions follow Michaelis–Menten kinetics, a converting algorithm was proposed to convert a possible pathway comprised of several biochemical reactions to QDE models.

Learning strategy: Given known species and types of reactions, all possible reactions are generated. A possible pathway is a subset of all these generated reactions, and all possible pathways constitute the search space. A search algorithm will list the candidate pathways and convert these pathways to QDE models by the converting algorithm. Then the converted QDE models are simulated by JMorven, and the obtained simulation results are compared with the given qualitative data.

Learning results: Given the complete set of 33 qualitative states obtained from simulating the target model, QML-Morven significantly narrowed down the number of possible pathways from 18 772 to 13. The corresponding QDE models for these 13 pathways could generate exactly the same set of states, which was the same as the given complete set.

To deal with the potential complexity of this pathway, a CSA was used to search the target pathway. This CSA is different from the one used in QML-CSA, because the search space for this

https://doi.org/10.1017/S0269888909990348 Published online by Cambridge University Press

problem is composed of pathways, rather than qualitative models in QML-CSA. To perform search in this kind of search space, a scoring system was set up to evaluate a pathway, and this scoring system served as a heuristic function to guide the search. Three criteria were used in this scoring system: *completeness* (the pathway includes all species involved), *consistency* (no conflicting relations in the pathway), and *coverage* (the corresponding qualitative model of the pathway covers all the given data).

The CSA was tested not only on the original methylglyoxal detoxification pathway, but also on two more complicated pathways created artificially based on the original one. Experimental results showed that CSA achieved better performance than the exhaustive and backtracking search, which indicated a promising way of improving the scalability of QML-Morven, when learning such complicated systems (Pang & Coghill, 2009).

6.5 Summary and discussion

In this section, several applications of QML were introduced, and they were the major applications found in the literature. Although it has strong application potentials, QML is a relatively small research field, and this is why there are not many applications found in this area. Theoretically speaking, QML can be applied to learning any dynamic system under the following two circumstances: (i) The training data are qualitative. For example, the available data for the system are sparse and imprecise, and it is better to represent them qualitatively. (ii) Not only the quantitative model, but also the QDE model of the system is difficult to obtain, which is caused by having insufficient knowledge about the system.

We again point out that QML is a general way and powerful tool to understand scientific theories, perform the engineering applications and even study the social systems. However, for different problems and applications, customized and specialized QML systems are needed. There is no QML system that can deal with all the problems. Appropriately choosing or designing a QML system for a specific problem is a very important issue, and two factors must be considered: (i) The features of the given data, which includes the completeness, accuracy, and resolution of the data. For example, for some biological systems with sparse data, ILP-QSI will be more suitable than QMN. On the other hand, if the high-resolution data are obtained, QMN can be applied. (ii) The specific nature of the search space. For instance, for large-scale search space, an evolutionary QML system will perform better than a symbolic QML system.

Finally, we acknowledge that there is another challenge when applying QML to the real-world applications. In real-world applications, it is often the case that not all qualitative states can be obtained, which makes QML tasks more difficult. In this sense, many of the above-described examples (except GENMODEL for learning the CVS model) are at the proof-of-concept stage, because in these examples complete qualitative data are required. Dealing with incomplete data is a very important issue that needs to be solved in order to make QML practicable. There are two feasible solutions to address this issue: (i) Developing better data pre-processing components. An appropriate data pre-processing component will make the best use of the raw quantitative data obtained from real-world applications. (ii) Better utilization of the domain knowledge. Learning from incomplete data often leads to a large number of candidate models. To narrow down a set of candidate models, the domain knowledge related to the real-world applications must be fully utilized to filter out as many false-positive models as possible. Although in the past two decades the volume of research carried out to study the above two solutions is relatively small, the progress made for the number of researchers engaged is very encouraging. In particular, the investigation of *kernel subsets* (described in section 3.5.4) gives us very promising results.

7 Future trends of QML

In previous sections, different QML systems have been examined, and parts of their applications have also been briefly introduced. As an active research field, QML has been hardly studied and

https://doi.org/10.1017/S0269888909990348 Published online by Cambridge University Press

broadly applied to different domains. However, there still remain many interesting topics and unsolved problems in QML research. In this section, we will explore the future of QML from the aspects of both its development and applications.

7.1 Integration with semi-quantitative and quantitative system identification

The real-world dynamic systems under study can be highly complex. If the modelling and learning approaches are restricted to purely qualitative ones, one may suffer from obtaining incomplete results, because qualitative approaches alone sometimes cannot reveal the dynamics of a system at more concrete levels, namely the semi-quantitative and quantitative levels. Consequently, hybrid approaches are required, which can model and learn the system at different levels of abstraction: quantitative, semi-quantitative, and qualitative levels. This also raises many questions and suggests several research directions, such as: (i) How can these approaches at different levels be integrated to better fulfil the learning task? (ii) How do we choose different hybrid approaches according to different learning tasks? (iii) How do we present the learning result? It seems that a general framework, which can address the above problems, is needed.

7.2 Dealing with more complicated QDE models

The currently available QML systems can successfully handle small- and medium-size problems. However, the size of the real-world dynamic systems can be very large, such as biological networks. When dealing with these problems, new search strategies, which can efficiently search the generated model space, are required. There have been already several attempts in this respect: the evolutionary (QME and QML-CSA) and the decomposition approaches (QML-IL). The initial investigation of these two strategies shows their potential abilities to deal with complex dynamic systems, although much work needs to be carried out in the future.

7.3 Integration with Fuzzy Set and Rough Set theory

Recent research in QR tends to incorporate other theories that can handle uncertain knowledge, such as the theory of Fuzzy Set (Zadeh, 1965) and Rough Set (Pawlak, 1991). For example, FuSim (Shen & Leitch, 1993) and Morven (Coghill, 1996) incorporate Fuzzy Set theory into QR, and in Rebolledo (2006), Rough Set theory is utilized. Correspondingly, QML research should also explore these directions. In fact, as mentioned before, the QML-Morven framework has the potential ability to deal with fuzzy data, because the Morven Framework is a fuzzy qualitative reasoner.

7.4 Utilising parallel computing

The development of computer hardware and distributed computing also makes it possible to deal with more complex QDE models. For example, ILP-QSI works on a 65-node Beowulf cluster. QML-Morven tests all its experiments on a cluster with nine nodes. QML systems need the corresponding modifications and additional software to adapt themselves to the parallel computing environment, so that they can effectively utilize the computing resources. Theoretically, all the execution components proposed in section 2 can be parallelized, but parallelising the following two is practical: (i) Parallelising the model search algorithm. The model search algorithm is a computationally expensive component in QML. When the model space is large, parallelising this component will significantly accelerate the whole computational process. (ii) Parallelising the qualitative simulator. When a qualitative simulator is used as the model verification component, and the simulation process is time consuming, it is natural to parallelize the simulator. Some research has already been carried out to achieve this, for example, the work of Platzner *et al.* (1997) and JMorven (Bruce & Coghill, 2005).

https://doi.org/10.1017/S0269888909990348 Published online by Cambridge University Press

7.5 Dealing with hybrid dynamic systems

The modelling of the real-world dynamic systems often involves more than one QDE, and the system will switch from one QDE to another under certain conditions, that is, they are hybrid dynamic systems (Alur *et al.*, 2000), or more precisely qualitative hybrid dynamic systems. Although much work has been carried out on identifying quantitative models of hybrid dynamic systems (Juloski *et al.*, 2005), there is little research about learning qualitative structures of complex hybrid systems. Such learning tasks are more challenging because they involve not only identifying multiple QDEs, but also the complicated transitions between these QDEs. For example, learning qualitative structures of genetic regulatory networks modelled by (de Jong *et al.*, 2004) has to fulfil both of the above-mentioned tasks.

Some preliminary research has been carried out in this field, such as the work of MISQ-RT (Ramachandran *et al.*, 1994), but there are still a lot of unsolved problems. One problem is the analysis of the influence of incomplete qualitative data on the learning, which is analogous to the one performed in learning singular qualitative models. Another problem is how to detect the switching thresholds of the hybrid dynamic systems, especially hybrid biological systems. This has been the subject of a preliminary study by Drulhe *et al.* (2006).

8 Conclusions

In this paper, the research field of QML was comprehensively reviewed: first a general structure of QML was proposed, then all the prominent QML systems were described in detail with reference to this structure. Their characteristics were summarized and compared, and their limitations were also pointed out. Finally, several applications of QML were briefly introduced, and possible future directions were explored. QML, as a very useful and powerful tool for the problems in many disciplines, has been an active research field for more than two decades. As Price *et al.* (2006) pointed out, the challenges and promises of QR were greater than ever, and we believe that the future of QML is equally promising.

Acknowledgements

We appreciate the support from the National Natural Science Foundation of China under the Grant nos. 60433020, 60673099 and 60773095. WP is financially supported by a joint scholarship of the University of Aberdeen and China Scholarship Council. WP and GMP are supported by the CRISP project (*Combinatorial Responses In Stress Pathways*) funded by the BBSRC (BB/F00513X/1) under the Systems Approaches to Biological Research (SABR) Initiative.

Appendix A: The U-Tube system

This system consists of a hollow tube in the shape of a 'U' and liquid in the tube, as shown in Figure 9. In this figure, the left and right tank of the U-Tube are labelled as tanks 1 and 2, and the

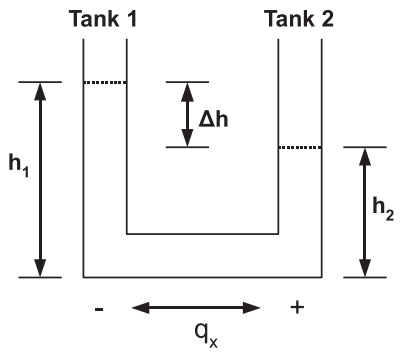


Figure 9 The U-Tube system

https://doi.org/10.1017/S0269888909990348 Published online by Cambridge University Press

$$
\begin{array}{l}
DERIV(h_1, q_x) \\
DERIV(h_2, -q_x) \\
ADD(h_2, \Delta h, h_1) \\
M^+(\Delta h, q_x) \\
MINUS(q_x, -q_x)
\end{array}
$$

Figure 10 The QSIM model for the U-Tube system

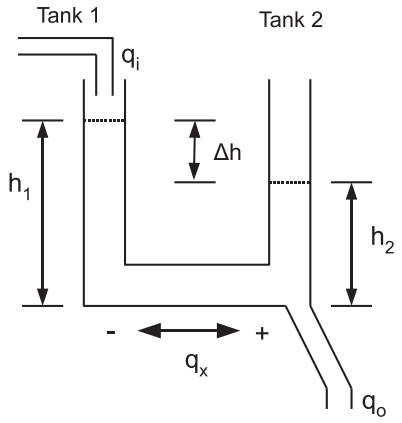


Figure 11 The coupled tanks

$$
\begin{array}{l}
DERIV\ (h_1, h_1') \\
DERIV\ (h_2, h_2') \\
ADD\ (h_2,\ \Delta h,\ h_1) \\
M^+\ (\Delta h,\ q_x) \\
M^+\ (h_2,\ q_o) \\
ADD\ (h_2',\ q_o,\ q_x) \\
ADD\ (q_x,\ h_1',\ q_i)
\end{array}
$$

Figure 12 The QSIM model for the coupled tanks

levels of the liquid in these two tanks are denoted as h_1 and h_2, respectively. Δh is the difference of h_1 and h_2. Liquid can flow freely from one tank to another, and the flow from the tank 1 to tank 2 is denoted as q_x. The QSIM model for this system is shown in Figure 10.

Appendix B: The coupled tanks

The coupled tanks system can be seen as a U-Tube (see Appendix A) with an input and output, as shown in Figure 11. The inflow is denoted by q_i and the outflow is denoted by q_o. The QSIM model for this system is shown in Figure 12.

Appendix C: The cascaded tanks

The cascaded tanks system is shown in Figure 13. There are two tanks (labelled as tanks 1 and 2), and tank 1 is located in a higher position than tank 2. Each tank has an inflow and an outflow, and the outflow of tank 1 is the inflow of tank 2. The QSIM model of this system is shown in Figure 14. In this figure, q_i is the inflow of tank 1; q_x is the outflow of tank 1 and also the inflow of tank 2; q_o is the outflow of tank 2.

https://doi.org/10.1017/S0269888909990348 Published online by Cambridge University Press

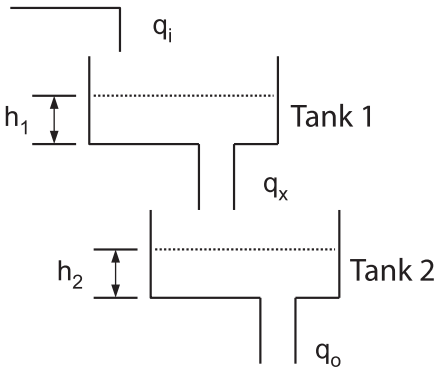


Figure 13 The cascaded tanks

$$
\begin{array}{|l|}
\hline
DERIV\ (h_1,h_1') \\
DERIV\ (h_2,h_2') \\
M^+\ (h_1,\ q_x) \\
M^+\ (h_2,\ q_o) \\
ADD\ (h_2',\ q_o,\ q_x) \\
ADD\ (q_x,\ h_1',\ q_i) \\
\hline
\end{array}
$$

Figure 14 The QSIM model for the cascaded tanks

References

Abe, S. 1993. A qualitative system idenditification method. In *Proceedings of the Seventh International Workshop on Qualitative Reasoning about Physical Systems*, 1–10. Orcas Island, Washington.

Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T., Ho, P., Nicolin, X., Olivero, A. & Sifakis, J. 2000. Discrete abstractions of hybrid systems. In *Proceedings of the IEEE*, **88**, 971–984. Pennsylvania University, Philadelphia, PA. IEEE Press.

Bellazzi, R., Ironi, L., Guglielmann, R. & Stefanelli, M. 1998. Qualitative models and fuzzy systems: an integrated approach for learning from data. *Artificial Intelligence in Medicine* **14**, 5–28.

Bergadano, F. & Gunetti, D. 1996. *Inductive Logic Programming From Machine Learning to Software Engineering*. MIT Press.

Bhaskar, R. & Nigam, A. 1990. Qualitative physics using dimensional analysis. *Artificial Intelligence* **45**, 73–111.

Blackman, R. & Tukey, J. 1958. *The measurement of Power Spectra*. Dover Publications Inc.

Bohte, Z. 1991. *Numerical Method*. The Society of Mathematicaians, Physicists and Astronomers of Slovenia.

Bradley, E., Easley, M. & Stolle, R. 2001. Reasoning about nonlinear system identification. *Artificial Intelligence* **133**, 139–188.

Bradley, E. & Stolle, R. 1996. Automatic construction of accurate models of physical systems. *Annals of Mathematics and Artificial Intelligence* **17**, 1–28.

Bratko, I., Muggleton, S. & Varšek, A. 1991. Learning qualitative models of dynamic systems. In *Proceedings of the 8th International Workshop on Machine Learning*, Birnbaum, L. and Collins, G. (eds). Morgan Kaufmann.

Bratko, I. & Šuc, D. 2003. Learning qualitative models. *AI Magazine* **240**(4), 107–119.

Bruce, A. M. 2007. *JMorven*: *A Framework for parallel non-constructive qualitative reasoning and fuzzy interval simulation*. PhD thesis, Department of Computing Science, Univeristy of Aberdeen.

Bruce, A. M. & Coghill, G. M. 2005. Parallel fuzzy qualitative reasoning. In *Proceedings of the 19th International Workshop on Qualitative Reasoning*. Graz, 110–116.

Burnet, F. M. 1959. *The Clonal Selection Theory of Acquired Immunity*. Cambridge University Press.

Camacho, R. 2000. *Inducing Models of Human Control Skills using Machine Learning Algorithms*. PhD thesis, University of Porto.

Cellier, F. E. 1991. General system problem solving paradigm for qualitative modeling. In *Qualitative Simulation Modeling and Analysis, Advances in Simulation* **5**, 51–71. Springer-Verlag.

Coghill, G. M. 1992. *Vector Envisionment of Compartmental Systems*. Master's thesis, University of Glasgow.

Coghill, G. M. 1996. *Mycroft: A Framework for Constraint based Fuzzy Qualitative Reasoning*. PhD thesis, Heriot-Watt University.

https://doi.org/10.1017/S0269888909990348 Published online by Cambridge University Press

Coghill, G. M. & Chantler, M. J. 1994. Mycroft: a framework for qualitative reasoning. In *Second International Conference on Intelligent Systems Engineering*, 43–48, Hamburg-Harburg, Germany.
Coghill, G. M., Garrett, S. & King, R. D. 2004. Learning qualitative metabolic models. In *European Conference on Artificial Intelligence (ECAI'04)*. Valencia, Spain, 445–449.
Coghill, G. M., Srinivasan, A. & King, R. D. 2008. Qualitative system identification from imperfect data. *Journal of Artificial Intelligence Research* **32**, 825–877.
Coiera, E. 1989a. Generating qualitative models from example behaviours. Technical Report DCS Report 8901, Department of Computer Science, University of New South Wales, Sydney, Australia.
Coiera, E. 1989b. Learning qualitative models from example behaviours. In *Proceedings of the Third Workshop on Qualitative Physics*. Stanford, 45–51.
de Almeida, C., Ozyamak, E., Miller, S., de Moura, A., Booth, I. & Grebogi, C. 2008. Modelling of methylglyoxal detoxification pathway in enteric bacteria. In *Abstract Book of the 9th International Conference on Systems Biology*, 170. Goteborg.
de Castro, L. N. & Timmis, J. 2002. An artificial immune network for multimodal function optimization. In *Proceedings of IEEE Congress on Evolutionary Computation (CEC'02)*. IEEE Press, 674–699.
de Castro, L. N. & Von Zuben, F. J. 2000. The clonal selection algorithm with engineering applications. In *Proceedings of GECCO,Workshop on Artificial Immune Systems and Their Applications*. Las Vegas, USA, 36–39.
de Castro, L. N. & Von Zuben, F. J. 2002. Learning and optimization using the clonal selection principle. In *IEEE Transactions on Evolutionary Computation, Special Issue on Artificial Immune Systems*. IEEE Press, **6**, 239–251.
de Jong, H., Gouze, J.-L., Hernandez, C., Page, M., Sari, T. & Geiselmann, J. 2004. Qualitative simulation of genetic regulatory networks using piecewise-linear models. *Mathematical Biology* **66**, 301–340.
Drulhe, S., Ferrari-Trecate, G., de Jong, H. & Viari, A. 2006. Reconstruction of switching thresholds in piecewise-affine models of genetic regulatory networks. *Lecture Notes in Computer Science* **3927**, 184–199. Springer-Verlag.
Džeroski, S. & Todorovski, L. 1993. Discovering dynamics. In Proc. Tenth *International Conference on Machine Learning*. 97–103. Morgan Kaufman.
Džeroski, S. & Todorovski, L. 1994. Discovering dynamics: From inductive logic programming to machine discovery. *Journal of Intelligent Information Systems* **3**, 1–20.
Ferguson, G. P., Totemeyer, S., MacLean, M. J. & Booth, I. R. 1998. Methylglyoxal production in bacteria: suicide or survival? *Archives of Microbiology* **170**(4), 209–218.
Forbus, K. D. 1997. Qualitative reasoning. In *The Computer Science and Engineering Handbook*, 715–733. CRC-Press.
Gerçeker, R. K. & Say, A. C. C. 2006. Using polynomial approximations to discover qualitative models. In *Proceedings of the 20th Annual Workshop on Qualitative Reasoning (QR06)*, 64–74.
Goldberg, D. E. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley.
Hau, D. T. & Coiera, E. W. 1993. Learning qualitative models of dynamic systems. *Machine Learning* **26**, 177–211.
Iwasaki, Y. & Simon, H. A. 1986. Causality in device behavior. *Artificial Intelligence* **29**, 3–32.
Juloski, A. L., Heemels, W. P. M. H., Ferrari-Trecate, G., Vidal, R., Paoletti, S. & Niessen, J. H. G. 2005. Comparison of four procedures for the identification of hybrid systems. In *Proceedings of the Hybrid Systems: Computation and Control (HSCC-05)*, Morari, M., Thiele, L. & Rossi, F (ed.), Lecture Notes in Computer Science **3414**, 354–369. Springer-Verlag.
Kay, H., Rinner, B. & Kuipers, B. 2000. Semi-quantitative system identification. *Artificial Intelligence* **119**, 103–140.
Keppens, J. & Shen, Q. 2001. On compositional modelling. *Knowledge Engineering Review* **16**(2), 157–200.
Khoury, M., Guerin, F. & Coghill, G. M. 2007. Learning dynamic models of compartment systems by combining symbolic regression with fuzzy vector envisionment. In *Genetic and Evolutionary Computation Conference (GECCO07)*, Thierens, D. (ed.), 2887–2894. ACM Press.
King, R. D., Garrett, S. M. & Coghill, G. M. 2005. On the use of qualitative reasoning to simulate and identify metabolic pathways. *Bioinformatics* **21**(9), 2017–2026.
Koza, J. R. 1992. *Genetic Programming: On the Programming of Computers by means of Natural Evolution*. MIT Press.
Kraan, I., Richards, B. L. & Kuipers, B. 1991. Automatic abduction of qualitative models. In *Proceedings of the Fifth International Workshop on Qualitative Reasoning about Physical Systems*, 295–301.
Kuipers, B. 1986. Qualitative simulation. *Artificial Intelligence* **29**, 289–338.
Kuipers, B. 1994. *Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge*. MIT Press.
Kuipers, B. & Kassirer, J. P. 1984. Causal reasoning in medicine: Analysis of a protocol. *Cognitive Science: A Multidisciplinary Journal* **8**(4), 363–385.
Ljung, L. 1999. *System Identification—Theory For the User*, 2nd edition. Prentice Hall.
Lotka, A. J. 1925. *Elements of Physical Biology*. Williams & Wilkins Co.

https://doi.org/10.1017/S0269888909990348 Published online by Cambridge University Press

McCreath, E. 1999. *Induction in First Order Logic From Noisy Training Samples and Fixed Sample Sizes*. PhD thesis, University of New South Wales.

Michaelis, L. & Menten, M. 1913. Die kinetik der invertinwirkung. *biochemische zeitschrift* **49**, 333–369.

Mitchell, T. 1997. *Machine Learning*. McGraw Hill.

Morgan, A. 1988. *Qualitative Behaviour of Dynamic Physical Systems*. PhD thesis, University of Cambridge.

Muggleton, S. 1995. Inverse entailment and Progol. *New Generation Computing, Special issue on Inductive Logic Programming* **1314**, 245–286. Springer-Verlag.

Muggleton, S. 1996. Learning from positive data. Lecture Notes in AI, 358–376.

Muggleton, S. & Feng, C. 1990. Efficient induction of logic programs. In *Proceedings of the 1st Conference on Algorithmic Learning Theory*. Ohmsma, Tokyo, Japan, 368–381.

Muggleton, S., King, R. D. & Sternberg, M. J. 1992. Protein secondary structure prediction using logic-based machine learning. *Protein Engineering* **5**(7), 647–657.

Muggleton, S. & Raedt, L. D. 1994. Inductive logic programming: Theory and methods. *Journal of Logic Programming* **19–20**, 629–679.

Pang, W. & Coghill, G. M. 2007a. Modified clonal selection algorithm for learning qualitative compartmental models of metabolic systems. In *Genetic and Evolutionary Computation Conference (GECCO07)*, Thierens, D. (ed.), 2887–2898. ACM Press.

Pang, W. & Coghill, G. M. 2007b. Advanced experiments for learning qualitative compartment models. In *The 21st Annual Workshop on Qualitative Reasoning*. Aberystwhyth.

Pang, W. & Coghill, G. M. 2009. An immune-inspired approach to qualitative system identification of the detoxification pathway of methylglyoxal. In *Proceeding of 8th International Conference on Artificial Immune Systems (ICARIS 2009)*, Andrews, P. S. and Timmis, J. (eds), Lecture Notes in Computer Science **5666**, 151–164. Springer-Verlag.

Papadimitriou, C. H. & Steiglitz, K. 1982. *Combinatorial optimization: algorithms and complexity*. Prentice-Hall.

Pawlak, Z. 1991. *Rough Sets: Theoretical Aspects of Reasoning About Data*. Kluwer Academic Publishing.

Platzner, M., Rinner, B. & Weiss, R. 1997. Parallel qualitative simulation. *Simulation Practice and Theory—International Journal of the Federation of European Simulation Societies* **5**(7–8), 623–638.

Plotkin, G. D. 1971. A further note on inductive generalisation. *Machine Intelligence* **6**, 101–124.

Price, C., Trave-Massuyes, L., Milne, R., Ironi, L., Forbus, K., Bredeweg, B., Lee, M., Struss, P., Snooke, N., Lucas, P., Cavazza, M. & Coghill, G. 2006. Qualitative futures. *The Knowledge Engineering Review* **21**(4), 317–334.

Ralston, A. & Rabinowitz, P. 2001. *A first Course in Numerical Analysis*, 2nd edition. Dover Publications.

Ramachandran, S., Mooney, R. J. & Kuipers, B. J. 1994. Learning qualitative models for systems with multiple operating regions. In *the Eighth International Workshop on Qualitative Reasoning about Physical Systems (QR-94)*. Nara.

Rebolledo, M. 2006. Rough intervals: enhancing intervals for qualitative modeling of technical systems. *Artificial Intelligence* **170**, 667–685.

Richards, B. L., Kraan, I. & Kuipers, B 1992. Automatic abduction of qualitative models. In *National Conference on Artificial Intelligence*, 723–728. AAAI.

Richards, B. L. & Mooney, R. J. 1992. Learning relations by pathfinding. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*. San Jose, 50–55.

Richards, B. L. & Mooney, R. J. 1995. Automated refinement of first-order horn-clause domain theories. *Machine Learning* **19**(2), 95–131.

Say, A. C. C. 1992. *Qualitative System Identification*. PhD thesis, Boğaziçi University.

Say, A. C. C. & Kuru, S. 1996. Qualitative system identification: deriving structure from behavior. *Artificial Intelligence* **83**, 75–141.

Selman, B., Levesque, H. & Mitchell, D. 1992. A new method for solving hard satisfiability problems. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*. San Jose, 440–446.

Shen, Q. & Leitch, R. 1993. Fuzzy qualitative simulation. *IEEE Transactions on Systems, Man, and Cybernetics* **23**(4), 1038–1061.

Shoup, T. E. 1979. *A Practical Guide to Computer Methods for Engineers*. Prentice-Hall.

Srinivasan, A. 1999. *The Aleph Manual*. `http://web.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/aleph.html`

Srinivasan, A. & King, R. D. 2008. Incremental identification of qualitative models of biological systems using inductive logic programming. *Journal of Machine Learning Research* **9**, 1475–1533.

Todorovski, L. 2003. *Using domain knowledge for automated modeling of dynamic systems with equation discovery*. PhD thesis, Faculty of Electrical Engineering and Computer Science, University of Ljubljana.

Todorovski, L. & Džeroski, S. 1997. Declarative bias in equation discovery. In *Proceedings of the 14th International Conference on Machine Learning*. Morgan Kaufmann, 376–384.

https://doi.org/10.1017/S0269888909990348 Published online by Cambridge University Press

Todorovski, L., Džeroski, S., Srinivasan, A., Whiteley, J. & Gavaghan, D. 2000. Discovering the structure of partial differential equations from example behavior. In *Proceedings of the 17th International Conference on Machine Learning*. Morgan Kaufmann, 991–998.

Valdés-Pérez, R. E. 1994. Conjecturing hidden entities by means of simplicity and conservation laws: machine discovery in chemistry. *Artificial Intelligence* **65**(2), 247–280.

Varšek, A. 1991. Qualitative model evolution. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, Sydney, Australia.

Vatcheva, I., de Jong, H., Bernard, O. & Mars, N. J. 2005. Experiment selection for the discrimination of semi-quantitative models of dynamical systems. *Artificial Intelligence* **170**, 472–506.

Wiegand, M. 1991. *Constructive Qualitative Simulation of Continuous Dynamic Systems*. PhD thesis, Heriot-Watt university.

Zadeh, L. A. 1965. Fuzzy sets. *Information and Control* **8**, 338–353.

https://doi.org/10.1017/S0269888909990348 Published online by Cambridge University Press