Available online at www.sciencedirect.com

SCIENCE DIRECT®

Ecological Modelling 170 (2003) 129–140

ECOLOGICAL MODELLING

www.elsevier.com/locate/ecolmodel

Learning population dynamics models from data and domain knowledge

Sašo Džeroski*, Ljupčo Todorovski

Department of Intelligent Systems, Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia

Abstract

This paper is concerned with integrating knowledge-based modeling or modeling from first principles, with data-driven or automated modeling of dynamic systems. The approach presented here includes methods for equation discovery: unlike mainstream system identification methods, which work under the assumption that the form of the equations is known, equation discovery systems explore a space of possible equation structures. We propose a formalism for representing knowledge about processes in population dynamics domains, and a method to transform such knowledge into an operational form that could be used by equation discovery systems. We also describe the extensions of the equation discovery system LAGRAMGE necessary to incorporate this kind of knowledge in the process of equation discovery.

Keywords: Equation discovery; Data-driven modeling; Knowledge-based modeling; Machine learning; Population dynamics; Process-based modeling; Structural modeling

1. Introduction

Most of the work in scientific discovery (Langley et al., 1987) is concerned with assisting the empirical approach to modeling of physical systems. Following this approach, the observed system is modeled on a trial-and-error basis to fit observed data. None or a very limited portion of the available domain (or background) knowledge about the observed system is used in the modeling process. This is especially the case in domains where a limited amount of knowledge is expressed in the form of mathematical laws, such as biology, medicine, and other life sciences.

The empirical approach is in contrast to the theoretical approach to modeling, where the basic physical processes involved in the observed system are first identified. A human expert then uses the domain knowledge about the identified processes to write down a proper structure of the model in the form of differential equations. Finally, the values of the constant parameters of these equations are fitted against the observed data using standard system identification methods (Ljung, 1993).

The focus of this paper is on integrating background knowledge from the domain of use, supplied by a domain expert, into the process of automated modeling of population dynamics with equation discovery. The equation discovery system LAGRAMGE (Todorovski and Džeroski, 1997) has made initial steps toward this integration. It allows the user to define the space of possible equations using a context-free grammar, written on the basis of the user background knowledge about the domain at hand. However, one can argue that it is difficult to encode a context-free grammar from expert knowledge.

In this paper, we propose a formalism for encoding population dynamics modeling knowledge that is

* Corresponding author. Fax: +386-1-425-1038.
E-mail address: saso.dzeroski@ijs.si (S. Džeroski).

0304-3800/$ – see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/S0304-3800(03)00221-7

more accessible to human experts. It allows an automated generation of a grammar for equation discovery. The generated grammar is context dependent (and not context free as in LAGRAMGE), so LAGRAMGE 2.0 was developed that, among other improvements, allows the use of context-dependent constraints in the grammar specifying the space of possible equations.

The paper is organized as follows. The basics of population dynamics modeling are introduced in Section 2. The formalism for encoding the population dynamics modeling knowledge, and the process of its transformation into a grammar for equation discovery are presented in Section 2. Section 3 gives a brief introduction to different approaches to the problem of equation discovery, especially differential equations. The focus is on methods for incorporating background knowledge from the domain of use in the process of equation discovery. The necessary improvements of LAGRAMGE are presented in Section 4. The experimental evaluation of LAGRAMGE 2.0 is given in Section 5. The last Section 6 summarizes the paper and gives directions for further work.

2. Population dynamics modeling

Population ecology studies the structure and dynamics of populations, where each population is a group of individuals of the same species inhabiting the same area. In this paper, we consider modeling the dynamics of populations, especially the dynamics of change of their density. We consider models of predator–prey population dynamics, where the interaction between predator and prey is antagonistic in the sense that it causes increase of the predator population and decrease of the prey population. The models take the form of systems of differential equations (Murray, 1993).

2.1. A generalized Volterra–Lotka model

Consider a simple model based on two populations, foxes and rabbits. The latter are grazing on grass, and the foxes are carnivores that hunt rabbits. We assume that rabbits are the only food of foxes, unlimited supply of grass is available to the rabbits and ignore seasonal changes. Under these assumptions, if the rabbit population is large, the fox population grows rapidly. However, this causes many rabbits to be eaten, thus diminishing the rabbit population to the point where the food for foxes is not sufficient. Consequently, the fox population decreases, which causes faster growth of the rabbit population.

The oscillatory behavior of the two population densities can be modeled by the Volterra–Lotka population dynamics model (Murray, 1993). This model can be generalized to the following schema:

$$\dot{N} = \text{growth\_rate}(N) - \text{feeds\_on}(P, N) \quad (1)$$

$$\dot{P} = \text{feeds\_on}(P, N) - \text{decay\_rate}(P) \quad (2)$$

where N is the prey (rabbit) population density and P is the predator (fox) population density, while $\dot{N} = \mathrm{d}N/\mathrm{d}t$ and $\dot{P} = \mathrm{d}P/\mathrm{d}t$ denote the time derivatives or rates of change of the two populations.

The generalized schema above allows for relaxing the unrealistic assumptions made in the original Volterra–Lotka model. These assumptions include exponential growth of the prey population and unlimited predation capacity of the predator population. By relaxing these assumptions, as described below, we can build models of predator–prey population dynamics with different complexities.

The term growth_rate(N) defines the model of the prey population growth in absence of predation. Two models of single population growth are usually used (Murray, 1993): (a) growth_rate(N) $= aN$; and (b) growth_rate(N) $= aN(1 - N/K)$. The first model (a) assumes that the population growth is exponential and unlimited. However, real-world environments typically have some carrying capacity for the population, which limits the density of the population. In such cases, logistic growth model (b) can be used as an alternative, where K is a constant, determining the carrying capacity of the environment.

The second assumption made in simple population models is that the predation rate is proportional to the densities of predator and prey populations (feeds_on(N, P) $= bPN$). As for population growth, this means that the predation growth is exponential and unlimited. Again, in some cases the predators have limited predation capacity. When the prey population density is small, the predation rate is proportional to it, but when the prey population becomes abundant, the predation capacity saturates to some limit

value (feeds_on(N, P) = $bPs(N)$). Three alternative terms are often used to model the predator saturation response to the increase of the prey density: (a) $s(N) = AN/(N + B)$, (b) $s(N) = AN^2/(N^2 + B)$, and (c) $s(N) = A(1 - e^{-BN})$, where A is the limit value of the predation capacity saturation, and B is the constant, which determines the saturation rate (Murray, 1993).

The modeling knowledge about population growth and saturation presented here can be very useful as background knowledge for automated modeling of ecological systems with equation discovery.

2.2. Encoding of domain and modeling knowledge

The knowledge about modeling population dynamics can be divided in two types. The first type is domain-specific knowledge about the populations and their role in the food chain. In the Volterra–Lotka model of population dynamics, this knowledge is represented by the single fact that foxes feed on rabbits. This type of knowledge can be expected from a biologist without any experience in mathematical modeling of population dynamics.

The second type of knowledge is domain-independent knowledge about population dynamics modeling, of the kind presented above. This type of knowledge is independent of the particular populations and food chain in a given domain, but is specific to the area of population dynamics modeling. It can be provided by, for example, a mathematician with some population dynamics modeling experience.

When modeling a new system of populations, domain-specific knowledge about the populations and food chains can be provided by a biologist familiar with the domain, but not necessarily with population dynamics modeling. We can re-use the area specific, but domain-independent modeling knowledge across different domains, although it has been provided by somebody who is not necessarily familiar with the ecosystem structure and the interactions involved in the new domain. By combining the two types of knowledge, we can obtain a space of possible population dynamics models (systems of differential equations) specified by a grammar. Measured data about the behavior of the populations can then be used to select among these models through the process of equation discovery.

Table 1
Description of a simple Volterra–Lotka population dynamics domain consisting of two populations: foxes (predator) and rabbits (prey)

```
domain(Volterra_Lotka).
population(Volterra_Lotka, fox).
population(Volterra_Lotka, rabbit).
feeds_on(Volterra_Lotka, fox, rabbit).
```

2.2.1. Domain-specific knowledge

We first provide a specification of the food chain in the domain: for our example rabbits and foxes domain it is given in Table 1. We use three first-order predicates: the `domain` (`domain_name`) predicate is used to specify the name of the domain at hand; each population in the domain is specified using the predicate `population` (`domain_name`, `population_name`); finally, we use the predicate `feeds_on(domain_name, predator_population, prey_population)`to specify each predator–prey interaction between two populations. For now, only predator–prey interactions can be specified. However, the formalism can be easily generalized to allow other types of interactions between populations, such as parasitism, competitive exclusion and symbiosis (Murray, 1993).

Note that by using the predicates `population` and `feeds_on`, the user is allowed to specify an arbitrary number of populations and predator–prey interactions between them.

2.2.2. Domain-independent modeling knowledge

The second part of the modeling knowledge is domain independent: for our example rabbits and foxes domain, it is given in Table 2. We use the predicate `template(process_name,input_variable, process_model_template)` to specify a set of alternative models for population dynamics processes

Table 2
Templates with alternative sub-expressions used for modeling the processes of saturation and population growth

```
template(saturation, X, (X)).
template(saturation,X,(X/(X+const[0:]))).
template(saturation,X,(X*X/(X*X+const[0:]))).
template(saturation,X,(1-exp(-const[0:]*X))).

template(growth,X,(const*X)).
template(growth,X,(const*X*(1-X/const[0:]))).
```

like population growth and saturation. Note that the symbol const is used to specify a constant parameter, whose value has to be fitted against measured data.

A constraint of the form [L:U] can be assigned to each constant parameter specifying that the value v of the constant parameter should be within the interval $L \leq v \leq U$. Omitting the U (L) value in this constraint means that there is no upper (lower) bound on the constant parameter value. For example, the symbol const[0:] means that the constant parameter should be non-negative.

2.2.3. More complex food chains (lattices)

In the Volterra–Lotka example, we are dealing with a very simple food chain with two populations and one predation (`feeds_on`) link. In general, a population dynamics domain will have many populations and the graph consisting of predation links will be a lattice. We can have a situation where one predator population can prey on several other populations.

For example, we can have three populations of foxes, rabbits, and pheasants, where foxes can feed on both rabbits and pheasants as alternative food sources. This will be represented by the facts `feeds_on(fox, rabbit)` and `feeds_on(fox, pheasant)`. The two predation process are happening in parallel and are combined additively, which means that there are two `feeds_on` terms in the differential equation for the (predator) population of foxes.

Similarly, we can have a situation where several predator populations prey on a single prey population. An example situation is represented by the facts `feeds_on(fox,rabbit)` and `feeds_on(wolf, rabbit)`. Again, the two predation processes are happening in parallel and are combined additively, i.e. there are two `feeds_on` terms in the differential equation for the (prey) population of rabbits.

In contrast to alternative food sources, a population may depend on several food sources at the same time. For example, phytoplankton needs both phosphorus and nitrogen at the same time to achieve optimal growth. In our formalism for representing domain knowledge, this would be represented as `feeds_on(phytoplankton, [nitrogen, phosphorus])`. The processes of phytoplankton feeding on nitrogen and phosphorus combine multiplicatively, i.e. the `feeds_on` term for phytoplankton contains a product of the two terms for the processes of feeding on nitrogen and phosphorus. Considering the food chains as graphs, the above `feeds_on` fact would correspond to something like an AND node in AND/OR graphs (Bratko, 2001). Alternative food sources for a predator population, on the other hand, would correspond to OR nodes.

2.2.4. Transforming the background knowledge into a grammar

We have written a program in the programming language Prolog (Bratko, 2001) that transforms a given food chain with nodes and links as outlined above to a grammar that derives expressions for the differential equations.

The top level productions of the grammar follow the generalized Schema (1) and (2). For each prey population that is not a predator, a growth term is added on the right-hand of the equation. For all other populations, a decay term is added. In the equation for a predator population, a positive term is added for each outgoing `feeds_on` link. In the equation for a prey, a negative term is added for each incoming `feeds_on` link.

Productions are then added for each of the basic processes (growth, decay, predation) taking place in the domain. For each `feeds_on(X, Y)` link, a production specifies the form the associated term can take: the predator density `X` is multiplied by the term `nutrient(Y)`. Productions for `nutrient(Y)` allow for different saturation terms, based on the templates specified in Table 2. For each `feeds_on(X, [Y1,...,Yn])` link, the product `nutrient(Y1)` $*\ldots*$ `nutrient(Yn)` is used instead of the `nutrient(Y)` term above.

The above assumes that we know exactly what predation processes are taking place in our domain. A useful extension might be to consider the given food chain as the set of possible processes, not all of which need take place. In such a case, we can allow any of the feeds-on, growth, and decay terms go to zero, essentially specifying that the process in question is not taking place (or has negligible effect in the domain overall). Also, for `feeds_on(X,[Y1,...,Yn])` links, we can allow only a subset of the given set of `n` potential prey populations (nutrients) be relevant to the predator, and have appropriate productions for each of the non-empty subsets of `[Y1,...,Yn]`. Our Prolog program already implements these extensions,

although our experiments with deriving grammars from domain knowledge and using these grammars for equation discovery did not use these extensions.

The above notions are illustrated by the following example.

2.2.5. Transforming the background knowledge into a grammar: an example

Using the definitions of the background knowledge from Tables 1 and 2, a grammar for equation discovery can be automatically generated. The process of transformation of the knowledge into grammar is automated using the predator–prey model schema. The grammar for the example population dynamics domain, consisting of rabbits and foxes, is given in Table 3.

The starting non-terminal symbol in the grammar `Volterra_Lotka` is used to generate the system of two differential equations of the population dynamics model, using the generalized Schema (1) and (2). The growth of the prey population of rabbits in the absence of predation is modeled using the non-terminal symbol `growth(rabbit)` with two alternative productions, reflecting the two `template` predicates for growth from Table 2. The third non-terminal symbol `feeds_on(fox, rabbit)` models the predation of foxes on rabbits. The predation rate is always proportional to the density of the fox population and the non-terminal `nutrient(rabbit)` is used to introduce the model of predator response to the increase of rabbit population density (the productions reflect the templates from Table 2). The terminal symbols fox and rabbit represent the system variables (population densities).

Strictly speaking, the grammar in Table 3 is not context free. The production for the starting symbol `Volterra_Lotka` generates two `feeds_on(fox, rabbit)` symbols, one in the first and another one in the second equation. In context-free grammar, these two non-terminal symbols can generate two different expressions. In population dynamics models, however, these two expressions have to be the same. The use of context-dependent constraints can overcome this limitation of context-free grammars.

3. Equation discovery

Equation discovery is the area of machine learning that develops methods for automated discovery of quantitative laws, expressed in the form of equations, in collections of measured data (Langley et al., 1987). It is related to the area of system identification. However, mainstream system identification methods work under the assumption that the structure of the model, i.e. the form of the equations, is known and are concerned with determining the values of the constant parameters in the model (Ljung, 1993). Equation discovery systems, on the other hand, aim at identifying both an adequate structure of the equations and appropriate values of the constant parameters.

3.1. Background knowledge and language bias

Equation discovery systems search through the space of possible equation structures. Most of the equation discovery systems emulate the empirical approach to scientific discovery: different equation

Table 3
A grammar for equation discovery constructed from the background knowledge in Tables 1 and 2

```
Volterra_Lotka ->
  time_deriv(rabbit) = growth(rabbit) - feeds_on(fox,rabbit)
  time_deriv(fox)=feeds_on(fox,rabbit)-const* fox

feeds_on(fox,rabbit)->const* fox* nutrient(rabbit)

nutrient(rabbit) -> rabbit
nutrient(rabbit)->rabbit/(rabbit+const[0:])
nutrient(rabbit)->rabbit* rabbit/(rabbit* rabbit+const[0:])
nutrient(rabbit)->1-exp(-const[0:]* rabbit)

growth(rabbit)->const* rabbit
growth(rabbit)->const* rabbit* (1-rabbit/const[0:])
```

structures are generated and fitted against measured data. However, some of the possible equation structures may be inappropriate for modeling the observed system. For example, consider the case where the measured variables of the observed system are not dimensionless. In that case some algebraic combinations of the system variables, such as addition or subtraction of mass and energy, are not valid. Beyond this simple example, there are also more sophisticated inconsistencies of equation structures with some background knowledge from the domain of the observed system.

Different equation discovery systems explore different spaces of possible equations, or in other words they use different language biases. One possibility is to use some pre-defined (built-in) language bias that restricts the space of possible equation structures to some reasonably small class, such as polynomials or trigonometric functions, like in LAGRANGE (Džeroski and Todorovski, 1995). In this case, the user can only influence the space of possible equations in a limited way and cannot use domain-specific knowledge in the process of equation discovery.

It is much better to use a declarative bias approach, where the user is allowed to influence or directly specify the space of possible equations. This approach provides users with a tool for incorporating their background knowledge about the domain at hand in the process of equation discovery. The use of background knowledge in the sense of a declarative language bias can avoid the problems of inconsistency of the discovered equations with the knowledge about the domain of the observed system, mentioned above.

Several equation discovery systems make use of domain-specific knowledge. In equation discovery systems that are based on genetic programming, the user is allowed to specify a set of algebraic operators that can be used. A similar approach has been used in the EF (Zembowicz and Żytkow, 1992) equation discovery system. The equation discovery system SDS (Washio and Motoda, 1997) effectively uses scale-type information about the dimensions of the system variables and is capable of discovering complex equations from noisy data.

However, expert users can usually provide much more modeling knowledge about the domain at hand than merely enumerating the algebraic operators to be used or (the scale-type of) dimensions of the measured system variables. In order to incorporate this knowledge in the process of equation discovery, we should provide the user with a more sophisticated declarative bias formalism. In LAGRAMGE (Todorovski and Džeroski, 1997), the formalism of context-free grammars has been used to specify the space of possible equations. Note here that context-free grammars are a far more general and powerful mechanism for incorporating domain-specific knowledge than the ones used in SDS (Washio and Motoda, 1997) and EF (Zembowicz and Żytkow, 1992).

The use of declarative bias in the form of a context-free grammar was crucial for modeling the phytoplankton growth in Lake Glumsoe in Denmark from real-world sparse noisy measurements (Džeroski et al., 1999). However, one can argue that it is difficult for the users of LAGRAMGE to express their knowledge about the domain in the form of a context-free grammar. In this paper, we present a formalism for encoding the domain knowledge at a higher, more user-friendly level, which can be automatically transformed to the operational form of grammars for equation discovery.

3.2. Discovery of differential equations

In this paper, we consider the problem of modeling dynamic systems, i.e. systems that change their state over time. Differential equations are the most common tool for modeling dynamic systems. LAGRANGE was the first equation discovery system that extended the scope of equation discovery systems to ordinary differential equations (Džeroski and Todorovski, 1995). The basic idea was to introduce the time derivatives of the systems variables through numerical differentiation and then search for algebraic equations. This simple approach has a major drawback: large errors are introduced by numerical differentiation.

The problem was partly resolved in the equation discovery system LAGRAMGE (Todorovski and Džeroski, 1997), where numerical integration is used instead of differentiation for the highest order derivatives. However, LAGRAMGE is only capable of discovering one differential equation for a single user-specified system variable at a time. In order to discover a system of simultaneous differential equations, LAGRAMGE has to be invoked several times, once for each system variable.

4. The equation discovery system LAGRAMGE 2.0

In order to use grammars like the one from Table 3 for equation discovery, we developed the equation discovery system LAGRAMGE 2.0, an improved version of LAGRAMGE 1.0 (Todorovski and Džeroski, 1997). Improvements were made in three directions. First, the context-dependent constraints have to be checked for each expression. Second, the grammar in Table 3 generates all model equations at once, therefore a system of simultaneous equations has to be discovered instead of discovering an equation for each system variable separately. Third, the constraints on the lower and upper bound of the values of the constant parameters have to be considered. The top-level algorithm of the LAGRAMGE 2.0 exhaustive search procedure is presented in Table 4.

The search procedure of LAGRAMGE 2.0 takes as an input a set of variables V, a data set D with measured time behaviors of the variables in V, a (context-dependent) grammar G, a set $V_d \subseteq V$ of dependent variables, and a parameter b that determines the number of best models (systems of equations) returned as output of LAGRAMGE.

The search space of LAGRAMGE is the set of parse trees that can be derived with the user provided grammar G. (A parse tree represents a set of simultaneous equations, i.e. a system of differential equations.) The search space is ordered according to the height of the parse trees using the refinement operator defined in (Todorovski and Džeroski, 1997). Starting with an empty parse tree, it can be repeatedly used to generate all parse trees.

Table 4
Outline of the search procedure of LAGRAMGE 2.0

```
    procedure LagramgeSearch(V, D, G, V_d, b)
1   Q ← {}
2   S ← enumerate all derivation trees in G
3   for each T in S do
4     if CheckConstraints(T, G) then
5       T.error ← 0.0
6       for each v in V_d
7         T.error ← T.error + Fit(v̇ = T · v, D)
8       endfor
9     endif
10  Q ← Q ∪ T
12  endwhile
11  return the b best parse trees in Q
```

Table 5
Examples of grammar productions with context-dependent constraints

```
E->A+B,B-A{A.1==A.2;}
E->A+B,B-A{A.1==A.2;B.1==B.2;}
E->A* E{A<=E;}
```

4.1. Context-dependent constraints

Each generated parse tree is first checked to see if it satisfies the context-dependent constraints in G (line 4 of the algorithm in Table 4). The user is allowed to specify an arbitrary number of context-dependent constraints for each production in the grammar. Examples of productions with context-dependent constraints are presented in Table 5.

In the first production, a single constraint `A.1==A.2` specifies that the first `(A.1)` and the second `(A.2)` occurrence of the symbol `A` on the right-hand side of the production should generate the same sub-expression. For example, the expression $a1 + b1$, $b2 - a2$ cannot be derived using that production ($A.1 -> a1$ is different from $A.2 -> a2$), whereas the expression $a1 + b1$, $b2 - a1$ can. However, the latter expression cannot be derived using the second production due to the second constraint `B.1==B.2`. $a1 + b1$, $b1 - a1$ is an example of an expression that can be derived using both productions.

The third production illustrates the use of a context-dependent constraint to avoid redundant generation of expressions that are equivalent due to the commutativity of multiplication. The context-free production $E -> A * E$ can generate both $a * b$ and $b * a$. On the other hand, using the context-dependent constraint $A <= E$ (where the operator $<=$ stands for lexicographic comparison), the second expression $b * a$ cannot be derived.

4.2. Simultaneous equations

In order to evaluate a system of simultaneous equations for the user provided set of dependent variables V_d, the sub-trees T_v of the generated tree T are identified for each dependent variable $v \in V_d$. Then the error of each equation of the form $\dot{v} = T \cdot v$ is evaluated (using the Fit function in line 7 of the algorithm in Table 4), where $T \cdot v$ denotes the

expression derived by the sub-tree T_v. The errors of the equations for all dependent variables are added together to obtain the error of the whole parse tree T (lines 5–8).

4.3. Constraints on the values of the constant parameters

The function Fit(*equation*, *D*) is used to fit the values of the constant parameters of the equation to the given data set *D*. The discrepancy between the measured data *D* and the data obtained by simulating the equation is used to evaluate the error of the equation. In LAGRAMGE 1.0, the downhill simplex algorithm (Press et al., 1986) was used to fit the values of the constant parameters (Todorovski and Džeroski, 1997). However, the downhill simplex algorithm cannot impose any constraints on parameter values. Because of this, we replaced the downhill simplex algorithm with the non-linear regression algorithm proposed in (Bunch et al., 1993). The latter allows the use of simple constraints specifying the lower and upper bounds on the values of the constant parameters.

5. Experiments

The goal of the experiments with LAGRAMGE, presented in this section, is to evaluate the effect of using the new type of background knowledge in the process of equation discovery. For that purpose, we compared the performance of LAGRAMGE 2.0 with the performance of LAGRAMGE 1.0 on the task of reconstructing different models of a simple aquatic ecosystem consisting of two populations of plankton (phytoplankton and zooplankton), as well as an inorganic nutrient (which we also treat as a population). The predator–prey food chain for this ecosystem is given in Table 6.

5.1. Experimental setup

Using the food chain description along with the domain-independent modeling templates from Table 2, the context-dependent grammar presented in Table 7 has been built using the algorithm described in Section 2.2. The grammar in Table 7 generates 32 different models, i.e. systems of three simultaneous equations, which are used in the experiments. The experimental evaluation of LAGRAMGE 2.0 consisted of attempting to reconstruct each of these 32 models from simulated data. Note that for inorganic nutrients, we assume no external inflow (no growth term); this is reflected in the equation for nut.

Using the grammar, we generated all 32 different model structures. In order to obtain simulation models, the values of the constant parameters have to be set. We used randomly generated values uniformly distributed on the $[0, 1]$ interval. We simulated each of the 32 obtained models from 10 different randomly selected initial states (initial values of nut, phyto and zoo) for 100 time steps of 1. Thus, 10 different behaviors were obtained of each of the 32 models.

In order to test the robustness of the approach to noise in the data, we added artificially generated random Gaussian noise to the behaviors. The noise was added at three different relative noise levels: 1, 5, and 10%. The relative noise level of l% means that we multiplied the original value x with $(1 + l \times G/100)$ to obtain a noisy value, where G is a normally distributed random variable with mean 0 and standard deviation 1.

Two evaluation criteria were used for evaluating the performance of the equation (re)discovery. First, the leave-one-out procedure was applied in order to estimate the error of discovered equations on test data, unseen during the discovery process. In each iteration of the leave-one-out procedure, 9 out of 10 behaviors were used to discover a system of differential equations with LAGRAMGE. The obtained differential

Table 6
Description of the aquatic environment population domain consisting of two populations (phytoplankton and zooplankton) and an inorganic nutrient (also treated as a population)

```
domain(aquatic).
population(aquatic, nut).
population(aquatic, phyto).
population(aquatic, zoo).
inorganic(_,nut).
feeds_on(aquatic, phyto, nut).
feeds_on(aquatic, zoo, phyto).
```

Table 7
A grammar for equation discovery in the aquatic ecosystem domain constructed from the background knowledge in Tables 6 and 2

```
aquatic->
  time_deriv(nut)=-feeds_on(phyto,nut)
  time_deriv(phyto)=growth(phyto)+feeds_on(phyto,nut) − feeds_on(zoo,phyto)
  time_deriv(zoo)=const∗ zoo+feeds_on(zoo,phyto)

feeds_on(phyto,nut)->const[0:]∗ phyto∗ nutrient(nut)
feeds_on(zoo,phyto)->const[0:]∗ zoo∗ nutrient(phyto)

nutrient(nut)->nut
nutrient(nut)->nut/(nut+const[0:])
nutrient(nut)->nut∗ nut/(nut∗ nut+const[0:])
nutrient(nut)->1-exp(-const[0:]∗ nut)

nutrient(phyto)->phyto
nutrient(phyto)->phyto/(phyto+const[0:])
nutrient(phyto)->phyto∗ phyto/(phyto∗ phyto+const[0:])
nutrient(phyto)->1-exp(-const[0:]∗ phyto)

growth(phyto)->const∗ phyto
growth(phyto)->const∗ phyto∗ (1-phyto/const[0:])
```

equations were then simulated using the initial state of the remaining (test) behavior. The simulation error was measured as the sum of squared differences between the simulated behavior and the test behavior. Second, the structure of the best model discovered by LAGRAMGE, was matched against the structure of the original model equations. The structure of the equations is obtained by abstracting the values of the constant parameters in them and replacing them with the generic symbol const.

5.2. Experimental results

We compared the performance of (1) LAGRAMGE 2.0 using the context-dependent grammar with the performance of (2) LAGRAMGE 2.0 using the grammar without constraints on the values of the constant parameters, and (3) LAGRAMGE 1.0 (where no context-dependent constraints, and no constraints on the values of the constant parameters can be used). The results of the comparison are summarized in Tables 8 and 9.

Before we discuss the experimental results, we should note here that the use of context-dependent constraints in the grammar reduces the space of possible models. The grammar in Table 7 generates 32 models when interpreted as a context-dependent grammar. On the other hand, when interpreted as a context-free grammar (by LAGRAMGE 1.0), this grammar generates 512 possible models. Therefore, using context-dependent constraints reduces the search space of LAGRAMGE by factor of 16.

Table 8
Performance of LAGRAMGE 2.0 (L2.0), LAGRAMGE 2.0 without applying constraints on constant parameters (L2.0-NCC) and LAGRAMGE 1.0 (L1.0)

Noise level (%)	Average test error			Structure reconstructed		
	L2.0	L2.0-NCC	L1.0	L2.0	L2.0-NCC	L1.0
0	0.0031	0.0020	0.0006	29	28	5
1	0.0083	*(1)	*(10)	8	6	0
5	0.1490	*(6)	*(13)	3	5	1
10	0.6187	*(6)	*(13)	4	5	2

Left hand side: average sum of squared errors on the test behavior. Right-hand side: number of successfully reconstructed original model structures.

Table 9
Win-tie-loss counts for comparison of the test error of (L2.0) with the test errors of L2.0-NCC and L1.0 (see caption of Table 8)

Noise level (%)	L2.0 vs. L2.0-NCC	L2.0 vs. L1.0
0	10-13-9	26-4-2
1	4-23-5	18-12-2
5	7-13-12	16-10-6
10	6-10-16	14-8-10

In the left hand side of Table 8 the average leave-one-out testing error of the 32 (re)discovered models is given. Note that the symbol $*(N)$ means that N out of 32 (re)discovered models could not be simulated (and therefore the average error could not be properly evaluated) due to the singularities, such as division by zero or unstable behavior of the discovered system of differential equations. Some of the singularities were caused by inappropriate values of the constant parameters (e.g. a negative saturation limit or carrying capacity). For this reason, LAGRAMGE 2.0 without applying constraints on the values of constant parameters often fails to discover a valid model. In models discovered by LAGRAMGE 1.0, some of the simulation failures are caused by inappropriate model structures, due to the lack of context-dependent constraints.

These results show one important aspect of the noise robustness of LAGRAMGE 2.0: at all noise levels it discovers models that can be simulated and have stable behaviors. This is due both to the context-dependent constraints and the constraints on the values of the constant parameters. This is very important: in our earlier experiments on modeling phytoplankton growth in Lake Glumsoe (Džeroski et al., 1999), we had to manually filter out the models discovered by LAGRAMGE 1.0 which had inappropriate values of the constant parameters (Džeroski et al., 1999).

In Table 9, the win-tie-loss counts for the comparison of the test error is presented. We counted the number of wins, ties, and losses in the following manner. For each of the 32 experiments, we compared the simulation error e_1 of LAGRAMGE 2.0 with the simulation error e_2 of the other two algorithms (L2.0-NCC and L1.0 in Table 9). Comparisons where the relative difference of the simulation errors is less than 10% (i.e. $90\% < e_2/e_1 < 110\%$) are considered ties.

The comparison of simulation errors shows clear performance improvement of LAGRAMGE 2.0 over LAGRAMGE 1.0 for all noise levels. On the other hand, models discovered by LAGRAMGE 2.0 without applying the constraints on the values of the constant parameters better fit noisy data than the ones generated with LAGRAMGE 2.0. This observation shows that models with inappropriate values of the constant parameters (or inappropriate structure for models discovered by LAGRAMGE 1.0) can sometimes fit the observed data better. However, these models do not make sense from biological point of view.

6. Discussion

We have presented an approach that allows for the representation and use of knowledge about processes in population dynamics systems and data about the behavior(s) of such systems, when trying to learn models that describe observed population behavior(s) in an automated fashion.

The formalism for encoding knowledge about population dynamics, allows us to encode two types of knowledge. The first is domain-specific knowledge about the predator–prey food chains in the domain and can be provided by a biologist without any experience in mathematical modeling. The second is modeling knowledge in the form of typical models or sub-models used for modeling different population dynamic processes, such as the growth of a population and the saturation of predation. The modeling knowledge is provided by a population modeling expert, not necessarily familiar with the domain at hand.

In both cases, we are dealing with high-level knowledge represented in first-order logic, which can be automatically transformed to the operational form of grammars used to guide the search for models in the process of equation discovery. This can be done for arbitrarily complex predator–prey models consisting of any number of populations and interactions between them. The proposed formalism can be easily extended with predicates for specifying other types of interactions between populations, such as parasitism, competitive exclusion, and symbiosis.

The grammars generated using the presented approach are context dependent and generate complete models, i.e. systems of simultaneous (differential) equations. The equation discovery system LAGRAMGE 2.0 was developed that makes use of such grammars.

Using context-dependent constraints reduces the space of possible models as compared to using purely context-free grammars (as in LAGRAMGE 1.0). Therefore, context-dependent constraints improve the efficiency of LAGRAMGE.

Experimental evaluation of LAGRAMGE 2.0 shows that both context-dependent constraints and constraints on the values of the constant parameters improves the noise robustness of LAGRAMGE in several ways. First, all the models (re)discovered by LAGRAMGE 2.0 at different noise levels can be simulated and generate stable behaviors. Second, all the models have clear interpretations from biological point of view. Finally, LAGRAMGE 2.0 (re)discovers the original model structure more often than LAGRAMGE 1.0. The experimental results should be further confirmed with experiments on the real-world observational data. These include modeling phytoplankton growth in the Danish lake, Glumsoe (Džeroski et al., 1999), predicting algae blooms in Lagoon of Venice (Džeroski et al., 1999) and modeling plankton population dynamics in the Japanese lake, Kasumigaura (Whigham, 2000).

Our approach is similar to the ECOLOGIC approach (Robertson et al., 1991) in the sense that it allows for representing modeling knowledge and domain-specific knowledge. However, in ECOLOGIC, the user has to select himself among the alternative models, whereas in our approach observational data is used to select among the alternatives. It is also related to process-based approaches to qualitative physics (Forbus, 1984). We can think of the food chain or domain-specific part of the knowledge as describing processes qualitatively, whereas the modeling part together with the data introduces the quantitative component.

Bradley et al. (2001) combine artificial intelligence techniques with traditional engineering methods for system identification, in order to automate the modeling of dynamic systems. They use meta-domain knowledge which allows for a unified representation of modeling knowledge, valid across different engineering domains, such as the mechanical, electrical or hydraulic domain. Their knowledge representation formalism is component-based, i.e. it incorporates knowledge about sub-models for individual components that can appear in the systems and about how these sub-models are combined into a model of the whole system. On the other hand, our representation of domain knowledge is process-based.

Currently, the presented formalism focuses on representing domain knowledge for population dynamics modeling. However, the presented formalism can be extended so knowledge about dynamic processes from other areas can be encoded and used for equation discovery. The knowledge should include ontology of typical processes in the area (such as predator–prey interaction and population growth in population dynamics) and templates of models typically used for modeling these processes. Finally, knowledge from different areas can be organized in the form of libraries of background knowledge for equation discovery.

References

Bradley, E., Easley, M., Stolle, R., 2001. Reasoning about nonlinear system identification. Artif. Intell. 133, 139–188.

Bratko, I., 2001. Prolog Programming for Artificial Intelligence, 3rd ed. Addison-Wesley, Reading, MA.

Bunch, D.S., Gay, D.M., Welsch, R.E., 1993. Algorithm 717; subroutines for maximum likelihood and quasi-likelihood estimation of parameters in nonlinear regression models. ACM Trans. Math. Software 19, 109–130.

Džeroski, S., Todorovski, L., 1995. Discovering dynamics: from inductive logic programming to machine discovery. J. Intell. Inf. Syst. 4, 89–108.

Džeroski, S., Todorovski, L., Bratko, I., Kompare, B., Križman, V., 1999. Equation discovery with ecological applications. In: Fielding, A.H. (Ed.), Machine Learning Methods for Ecological Applications. Kluwer Academic Publishers, Boston, MA, pp. 185–207.

Forbus, K., 1984. Qualitative process theory. Artif. Intell. 24, 85–168.

Langley, P., Simon, H.A., Bradshaw, G.L., Żythow, J.M., 1987. Scientific Discovery. MIT Press, Cambridge, MA.

Ljung, L., 1993. Modelling of industrial systems. In: Proceedings of Seventh International Symposium on Methodologies for Intelligent Systems. Springer, Berlin, pp. 338–349.

Murray, J.D., 1993. Mathematical Biology, 2nd ed. (Corrected). Springer, Berlin.

Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterlin, W.T., 1986. Numerical Recipes. Cambridge University Press, Cambridge, MA.

Robertson, D., Bundy, A., Muetzelfield, R., Haggith, M., Uschold, M., 1991. Ecologic: Logic-based Approaches to Ecological Modelling. MIT Press, Cambridge, MA.

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

Washio, T., Motoda, H., 1997. Discovering admissible models of complex systems based on scale-types and identity constraints. In: Proceedings of the 15th International Joint Conference on Artificial Intelligence, vol. 2. Morgan Kaufmann, San Francisco, CA, pp. 810–817.

Whigham, P.A., 2000. An inductive approach to ecological time series modelling by evolutionary computation. In: Abstract Book of the Second International Conference on Applications of Machine Learning to Ecological Modelling. Adelaide University, Adelaide, Australia, p. 35.

Zembowicz, R., Żytkow, J.M., 1992. Discovery of equations: experimental evaluation of convergence. In: Proceedings of the 10th National Conference on Artificial Intelligence. Morgan Kaufmann, San Francisco, CA, pp. 70–75.