University of Ljubljana
Faculty for Computer and Information Science

Ljupčo Todorovski

Declarative Bias in Equation Discovery

M.Sc. Thesis

Thesis Supervisor: professor Ivan Bratko

Ljubljana, 1998

Abstract

In the areas of machine learning dealing with very expressive and large hypothesis spaces, declarative bias is used to define and restrict the search space for the target concept. Although equation discovery systems deal with potentially infinite hypothesis spaces, only few (if any) make use of declarative bias to specify the form of the discovered equations.

The thesis presents a declarative bias formalism for equation discovery. We use context free grammars to specify the form of discovered equations. These grammars can make use of functions defined on the basis of domain specific knowledge, in addition to common mathematical operators. The formalism is implemented in the equation discovery system LAGRAMGE that heuristically searches the equation space defined by the context free grammar.

LAGRAMGE is applied to several equation discovery problems in artificial and real world domains. Among others, the complex model of an inverted pendulum that consists of a cart and two poles was successfully reconstructed from numerically simulated data. LAGRAMGE was also used for automated modeling of phytoplankton growth in the Danish lake Glumsoe. The discovered model is suitable for accurate short and long term prediction of the phytoplankton concentration in the lake.

Contents

Chapter 1

Introduction

1.1 Motivation

The term bias refers to any kind of basis for choosing one generalization over another, other than strict consistency with the observed training examples [Mitchell 1980]. The importance of bias was proved in some early machine learning and pattern recognition research. It was observed that stronger language bias leads to better performance of learned concept on test cases not available to the learning system during the learning process.

All machine learning systems use some kind of language bias to specify the hypothesis space. There is a trend in machine learning to make language bias declarative, i.e., to make the definition of the hypothesis space a parameter or input of the learning system. This trend has been most obvious in areas of machine learning concerned with very expressive (and therefore large) hypothesis languages, such as subsets of first order logic. Many systems thus exist within the area of inductive logic programming (ILP) that implement and use various forms of declarative bias (for an overview see [Nédellec et al. 1996]).

The hypothesis space used in equation discovery systems is the space of all equations that can be constructed given a fixed set of arithmetical operators, functions and variables. The number of such equations is infinite. However, from the theory of approximations it is known that an infinite number of polynomial curves (piecewise polynomial, splines) provide a good fit to any data set. Therefore, some kind of strong language bias, based on specific knowledge about the problem at hand, is necessary to prevent degradation of the discovered equations' accuracy on unseen data. On the other hand, the bias must be weak enough to allow an equation that fits the training data with acceptable accuracy.

State of the art equation discovery systems employ non declarative language bias or some kind of parametrized weak bias, such as choosing a subset of possible operators and functions, and limiting equation depth (see e.g. [Zembowitz and Żytkow 1992, Todorovski 1993, Džeroski and Todorovski 1993]). None of the existing equation discovery systems use strong declarative language bias, based on specific knowledge about the domain at hand, to specify the form and restrict the space of possible equations.

Declarative language bias can limit the search in a potentially huge hypothesis space to a manageable level. Also, declarative language bias can be viewed as a way of providing domain knowledge to the learning system, so that it can adapt better to the task at hand. It is thus clearly desirable to have a declarative bias facility in a system for equation discovery, where the search space is potentially huge and the problems faced very diverse.

1.2 Related work

On the declarative bias side, the work presented in the thesis is related to learning systems in the area of inductive logic programming (ILP), which often use declarative language bias [Nédellec et al. 1996]. A variety of formalisms for declarative bias have been used in ILP systems such as clause templates/schemata in MOBAL [Morik et al. 1993], clause sets in TRACY [Bergadano and Gunetti 1995] or combinations of both in CLAUDIEN [Dehaspe and DeRaedt 1995]. [Cohen 1992] uses grammars to describe the clauses in the hypothesis language of the ILP system GRENDEL. Grammars have been also used to bias search in genetic programming [Whigham 1995].

On the equation discovery side, the presented work is related to equation discovery systems, such as BACON [Langley et al. 1987], EF [Zembowitz and Żytkow 1992], E* [Schaffer 1993], LAGRANGE [Todorovski 1993, Džeroski and Todorovski 1993] and GOLDHORN [Križman et. al 1995]. BACON, EF and E* are concerned with the problem of discovering equations involving one or two input variables. To discover equations involving more than two variables they ask the user for specific additional experiments in the domain at hand. Except for LAGRANGE and GOLDHORN, none of the mentioned systems can deal with differential equations.

All the mentioned systems include some form of bias: the particular biases used are however either hard-coded or parametrized and thus much less flexible that the declarative bias facility described in the thesis. E* chooses among several (six) built-in simple equation templates. EF provides a declarative bias facility for specifying the (sub)set of built-in functions for introducing new terms, but these can only be combined by polynomial regression, producing equations that are linear in their parameters. The degree of polynomials considered in regression is specified as parameter. The depth of new terms which can be introduced by multiplication is a parameter in LAGRANGE, as is the number of additive terms involved in an equation. GOLDHORN considers the same equation space as LAGRANGE, that is the space of multivariate polynomials on the system variables.

1.3 Contributions of the thesis

In the thesis, a declarative bias formalism for equation discovery is developed. We use context free grammars to specify the form of the discovered equations. These grammars can make use of common arithmetical operators. Additionally, we can provide domain specific knowledge to the equation discovery system with incorporating user-defined functions in the grammar.

The formalism is implemented in an equation discovery system called LAGRAMGE[1]. The hypothesis space of LAGRAMGE is ordered with a refinement operator on derivation trees, which allows the use of different search strategies for equation discovery. The equations considered can be non-linear in both the system variables and constant parameters. The search heuristic used is based on the fit of equations to measured data, but can also take into account the length of equations. LAGRAMGE can find both ordinary and differential equations and it can also take into account more than one measurement table of the system variables.

The features of LAGRAMGE were empirically tested on several synthetic and real life domains. Experiments in synthetic domains vary from the reconstruction of a simple ecosystem model to the reconstruction of a complex dynamic system of two poles on cart. LAGRAMGE was also used on the problem of modeling and predicting phytoplankton growth in Lake Glumsoe, Denmark. The discovered model can be used for accurate prediction of the phytoplankton concentration.

1.4 Organization of the thesis

The thesis is organized as follows.

Chapter 2 defines the task of equation discovery in the presence of declarative bias and describes the developed declarative bias formalism. The use of the formalism is illustrated on the problem of equation discovery in a simple ecosystem domain. The chapter concludes with a discussion on the hypothesis space complexity.

A detailed description of the equation discovery system LAGRAMGE and its' implementation is given in Chapter 3. A refinement operator on derivation trees and an algorithm for enumerating derivation trees are presented. Both are used in LAGRAMGE for exploring the hypothesis space of equations. After a description of the procedures for constant parameters' fitting, the heuristic functions and the stopping criteria used, Chapter 3 concludes with a description of the top-level algorithm and a section on LAGRANGE's implementation in the C programming language.

The results of applying LAGRAMGE in three synthetic and two real life domains are presented in Chapter 4. The experimental domains vary from the reconstruction of a simple ecosystem model to modeling and prediction of phytoplankton growth in Danish lake Glumsoe.

Finally, Chapter 5 concludes the thesis with a summary of the presented work and directions for further work.

[1] This is a deliberate misspelling of the name of the equation discovery system LAGRANGE, the predecessor of LAGRAMGE. The letter N is replaced with M, so that the second part of the acronym reads GRAM as in grammar. Namely, declarative bias based on grammars is used in LAGRAMGE

Chapter 2

Declarative bias in equation discovery

Equation discovery systems explore the hypothesis space of all equations that can be constructed given a set of arithmetical operators, functions and variables, searching for an equation that fits the input data best. The number of all possible equations is infinite and an infinite number of polynomial curves provide a good fit to any data set. Therefore, some kind of language bias is necessary for restricting the hypothesis space.

In this chapter, we present a declarative bias formalism for equation discovery. This formalism allows the user of an equation discovery system to restrict the hypothesis space on the basis of expert background knowledge for the domain at hand. It makes use of context free grammars to specify the form of the discovered equations. These grammars can make use of common arithmetical operators. Additionally, we can provide a domain specific knowledge to the equation discovery system with incorporating user-defined functions in the grammar.

The rest of the chapter is organized as follows. First, we define the task of equation discovery in the presence of a declarative language bias. We present a formalism for declarative bias based on context free grammars. The use of the presented formalism is illustrated on the problem of equation discovery in a simple ecosystem domain. Finnally, the chapter concludes with a study of the reduction of the hypothesis space complexity when using declarative language bias.

2.1 Problem definition

The problem of equation discovery, as addressed in the thesis can be defined as follows.
Given:

- context free grammar $G = (N, T, P, S)$ and
- input data $D = (V, v_d, M)$, where

- $V = \{v_1, v_2, \ldots v_n\}$ is a set of domain variables,
- $v_d \in V$ is the dependent variable and
- M is a set of one or more measurement tables. Each measurement table is a table of measured values of the domain variables at successive time points:

time	v_1	v_2	$\ldots$	v_n
t_0	$v_{1,0}$	$v_{2,0}$	$\ldots$	$v_{n,0}$
t_1	$v_{1,1}$	$v_{2,1}$	$\ldots$	$v_{n,1}$
t_2	$v_{1,2}$	$v_{2,2}$	$\ldots$	$v_{n,2}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
t_m	$v_{1,m}$	$v_{2,m}$	$\ldots$	$v_{n,m}$

Find an equation that expresses the dependent variable v_d in terms of variables from V. This equation is expected to minimize the discrepancy between the measured and calculated values of the dependent variable. The equation can be:

- differential, i.e., of the form $dv_d/dt = \dot{v_d} = E$[1], or
- ordinary, i.e., of the form $v_d = E$.

where E is an expression that can be derived from the context free grammar G.

2.2 The declarative bias formalism

2.2.1 Context free grammars

The syntax of the expressions on the right hand side of the equation is prescribed with a context free grammar [Chomsky 1986, Hopcroft and Ullman 1979]. A context free grammar contains a finite set of variables (also called nonterminals or syntactic categories) each of which represents expressions or phrases in a language (in equation discovery, nonterminals represent sets of expressions that can appear in the equations). The expressions represented by the nonterminals are described in terms of nonterminals and primitive symbols called terminals. The rules relating the nonterminals among themselves and to terminals are called productions.

The original motivation for the development of context free grammars was the description of natural languages. For example, a simple grammar for deriving sentences consists of the productions *sentence* $\to$ *noun verb*, *noun* $\to$ *spider*, *noun* $\to$ *snake*, and *verb* $\to$ *attacks*. Here *sentence*, *noun* and *verb* are nonterminals, while words that actually appears in sentences (i.e. *spider*, *attacks*) are terminals. The sentences *spider attacks* and *snake attacks* can be derived with this grammar.

[1]Note that in the case of discovering differential equation $\dot{v_d} \notin V$. If $\dot{v_d} \in V$, the problem can be reformulated as discovery of ordinary equation.

We denote a context free grammar as a tuple $G = (N, T, P, S)$ [Hopcroft and Ullman 1979]. N and T are finite disjoint setsof nonterminals and terminals, respectively. P is a finite set of productions and $S \in N$ is the starting nonterminal. Productions in P are of the form $A \to \alpha$, where $A \in N$ is called the left side and $\alpha \in (N \cup T)^*$ the right side of the production. The set of all productions with nonterminal A on their left side is termed P_A.

The grammar used to describe the declarative bias for equation discovery has several symbols with special meanings. The terminal *const* $\in T$ is used to denote a constant parameter in an equation that has to be fitted to the input data. The terminals v_i are used to denote variables from the input domain D. Finally, the nonterminal $v \in N$ denotes any variable from the input domain. Productions connecting this nonterminal symbol to the terminals v_i are attached to v automatically, i.e., $\forall v_i \in V : v \to v_i \in P$.

The only restriction on the grammar G is that the right sides of the productions in P have to be expressions that are legal in the C programming language. This means that we can use all C built-in operators and functions in the grammar. Additional functions, representing background knowledge about the addressed domain can be used, as long as they are defined in conjunction with the grammar. Note that the derived equations may be non-linear in both the constant parameters and the system variables.

2.2.2 Generating expressions from the grammar

Expressions are derived from the grammar $G = (N, T, P, S)$ using derivation trees. A derivation (or parse) tree is a tree that satisfies the following constraints:

1. Every vertex has a label:
   (a) interior nodes have labels from the set of nonterminals N,
   (b) leaf nodes have labels from the set of all symbols $T \cup N$.
2. The label of the root is S.
3. If vertex n has label A and vertices $n_1, n_2, \ldots n_k$ are the children of vertex n in order from left to right, with labels $A_1, A_2, \ldots A_k$, respectively, then $A \to A_1 A_2 \ldots A_k$ must be production in $P_A \subseteq P$.

The derivation tree generates an expression, if all its leaf nodes are labelled with terminal symbols. If we read the labels of the leaf nodes from left to right, we get the expression derived by the tree. The algorithm for deriving expressions from a context free grammar is given in Table 2.1. For simplicity, we make no difference between the label of a vertex and the vertex itself.

Finally, we define the height of the derivation tree as the maximal length of a path from the root to a leaf node.

```
procedure GenerateExpression(G)
    begin with a derivation tree 𝒯 consisting of the node S
    repeat
        choose a nonterminal leaf node A in 𝒯
        choose a production p = A → A_1 A_2 ... A_k ∈ P_A
        expand A with the successors A_1, A_2, ... A_k
    until all leaf nodes in 𝒯 are terminals
    print out the leaf nodes in 𝒯 in order from left to right
```

Table 2.1: Algorithm for deriving expressions from a context free grammar

2.2.3 An example - simple ecosystem domain

For example, consider the grammar for the aquatic ecosystem domain (see Section 4.1.1) defined in Table 2.2. The productions $v \to nut \mid phyt \mid zoo$ connecting the nonterminal v with the domain variables are automatically added to the grammar. The definition of the background knowledge function `monod` in the C programming language is also attached to the grammar.

```c
double monod(double c, double v) {
        return(v / (v + c));
}
```

$$
\begin{aligned}
N &= \{E, F, Y, v\} \\
T &= \left\{+, const, \cdot, \mathtt{monod}, (, \underline{,},), nut, phyt, zoo\right\} \\
P &= \left\{\begin{array}{lcl} E & \to & const \mid const \cdot F \mid E + const \cdot F \\ F & \to & v \mid Y \mid v \cdot Y \\ Y & \to & \mathtt{monod}(const, v) \\ v & \to & nut \mid phyt \mid zoo \end{array}\right\} \\
S &= E
\end{aligned}
$$

Table 2.2: Example context free grammar G used as declarative bias for equation discovery

The definition of the function `monod` is based on expert background knowledge in population dynamics [Monod 1942, Crispi and Mosetti 1993]. This function is used to describe population growth where `v` is a nutrient variable and `c` a saturation constant. The monod function is incorporated in expressions through the production $Y \to \mathtt{monod}(const, v)$. Productions for the nonterminal F can be used to combine domain variables and `monod` terms in multiplicative terms. Finally, linear combinations of these multiplicative terms are constructed with productions for the starting nonterminal E. Note that the derived expressions are not necessarily linear in either the constant parameters or the system variables, because of the presence of a constant parameter and a variable in the denominator of the monod term.

Figure 2.1 shows the derivation tree for the expression $const \cdot phyt + const \cdot phyt \cdot \texttt{monod}(const, nut) + const \cdot zoo \cdot \texttt{monod}(const, phyt)$ generated with the grammar from Table 2.2. The height of the derivation tree is five (equal to the length of the path from the root node to the leaf node nut).

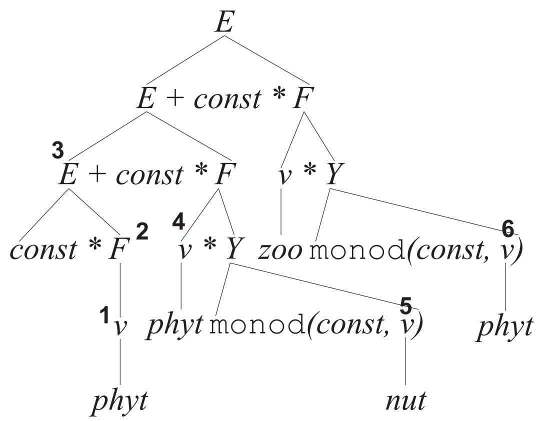

Figure 2.1: Derivation tree for the expression $const \cdot phyt + const \cdot phyt \cdot \texttt{monod}(const, nut) + const \cdot zoo \cdot \texttt{monod}(const, phyt)$

2.3 Hypothesis space complexity

The hypothesis space for equation discovery in the presence of declarative bias is the set of all possible differential or ordinary equations, whose right hand sides can be derived with the context free grammar G. Once we choose the dependent variable and the form of the equation we want to discover (differential or ordinary) the equation is uniquely defined with the expression on its right hand side. As we showed in the previous section, the expressions are derived from the context free grammar using derivation trees. Therefore, the hypothesis space of equations is equivalent to the set of all derivation trees.[2]

2.3.1 Number of derivation trees

Let $n_G(A, h)$ be the number of derivation trees of grammar G with root symbol A and height h, and $N_G(A, h)$ the number of derivation trees with root symbol A and height up to h. Then, we can use the following recursive formulas to calculate these numbers:

[2]This is clearly true for unambiguous context free grammars. A context free grammar such that some expression has two derivation trees is said to be ambiguous. But it also holds for LAGRAMGE, because it searches through the space of derivation trees, rather than expressions. The expressions with more than one derivation tree will be considered more than once in the process of equation discovery with LAGRAMGE.

- $A \in T$:

$$n_G(A,h) = \begin{cases} 1 & : \ h = 0 \\ 0 & : \ h \geq 1 \end{cases}$$

- $A \in N$:

$$n_G(A,h) = \begin{cases} 0 & : \ h = 0 \\ \text{number of productions } A \rightarrow w : w \in T^* & : \ h = 1 \\ \sum_{A \rightarrow A_1 \ldots A_k \in P_A} \left[\prod_{i=1}^{k} N_G(A_i, h-1) - \prod_{i=1}^{k} N_G(A_i, h-2)\right] & : \ h \geq 2 \end{cases}$$

- $N_G(A,h) = \sum_{h_i=0}^{h} n_G(A, h_i)$

If we limit the height of the derivation trees to h_{max}, then the hypothesis space complexity, i.e., the number of expressions that can be derived using the grammar G is $N_G(S, h_{max})$.

2.3.2 Reduction of the hypothesis space complexity

Using the formulas above, we can compare the number of expressions that can be derived using the example grammar G from Table 2.2 and the number of all expressions involving common arithmetical operators. The latter can be derived using the universal grammar U:

$$\begin{array}{lcl} E & \rightarrow & E + F \mid E - F \mid F \\ F & \rightarrow & F \cdot T \mid F/T \mid T \\ T & \rightarrow & \mathit{const} \mid v \mid (E) \end{array}$$

Consider the second, most complicated, equation $\dot{phyt} = phyt \cdot nut/(k_N + nut) - r_P \cdot phyt - zoo \cdot phyt/(k_P + phyt)$ of the simple ecosystem domain from Section 4.1.1. The expression on the right hand side of this equation can be derived using the grammar G from Table 2.2 with a derivation tree of height five (see Figure 2.1). On the other side, using the universal grammar for deriving the same expression, a derivation tree of height nine is necessary. A higher tree is needed because of the lack of the `monod` function definition in the universal grammar, so the monod term has to be derived using the productions for addition, division and putting in parenthesis.

Table 2.3 lists the number expressions that can be derived using grammars G and U for different heights h of derivation trees. It is clear from the table that by using the background function `monod` in the context free grammar definition G the complexity of the hypothesis space for equation discovery is reduced by a factor of order 10^{35}.

We also compare the number of derivation trees with the number of equations considered during the search process in the equation discovery systems LAGRANGE [Todorovski 1993, Džeroski and Todorovski 1993] and GOLDHORN [Križman et. al 1995].

The language bias of equation discovery systems LAGRANGE and GOLDHORN is the space of multivariate polynomials on system variables and their time derivatives. The equations for the simple ecosystem domain can be rewritten in the form of multivariate polynomials in many different ways. For the purpose of comparison, the one is chosen that minimizes the degree of the polynomials and the number of additive terms involved

h	$N_G(E,h)$	$N_U(E,h)$
1	1	0
2	1	0
3	1	7
4	121	36
5	**1831**	7300
6	27481	14674005
7	412231	$2.3607 \cdot 10^{12}$
8	6183481	$5.5481 \cdot 10^{21}$
9	92752231	$\mathbf{3.8267 \cdot 10^{38}}$
10	$1.3913 \cdot 10^{9}$	$1.2462 \cdot 10^{68}$

Table 2.3: Hypothesis space size for grammars G and U

in the equations, i.e, minimizes the number of equations considered by Lagrange and GoldHorn in the process of equation discovery:

$$\begin{aligned} k_N \cdot \dot{nut} + nut \cdot \dot{nut} &= -nut \cdot phyt \\ \dot{phyt} &= -\dot{nut} - r_P \cdot phyt - \dot{zoo} - r_Z \cdot zoo \\ k_P \cdot \dot{zoo} + phyt \cdot \dot{zoo} &= phyt \cdot zoo - p_Z \cdot zoo - k_P \cdot zoo - r_Z \cdot zoo \cdot phyt \end{aligned}$$

We can observe from these equations, that the degree of polynomials is two and the maximal number of additive terms involved in equations is five. Given an initial set of system variables and their derivatives $I = \left\{nut, phyt, zoo, \dot{nut}, \dot{phyt}, \dot{zoo}\right\}$ with $|I| = 6$, we can compute that the number of required new terms introduced with multiplication is $\binom{6+2-1}{2} = 21$. That gives a total number of 27 terms that can be used in linear combinations. The number of all equations that involve at most five of these 27 terms can be computed in the following way:

$$\binom{27}{1} + \binom{27}{2} + \binom{27}{3} + \binom{27}{4} + \binom{27}{5} = 101583.$$

Note that this number of considered equations suffices for discovering all three equations, because Lagrange and GoldHorn discover implicit equations where none of the system variables is distinguished as dependent. Thus, hypothesis space, when using declarative bias, is reduced by a factor of $101583/(3 \cdot 1831) = 18.49$.

Table 2.4 summarizes the comparison of hypothesis space complexities for the problem of equation discovery in the aquatic ecosystem domain from Section 4.1.1.

Language bias	Hypothesis space complexity	Factor of reduction
context free grammar G	$3 \cdot 1831 = 5493$	-
all arithmetical expressions	$3 \cdot 3.8267 \cdot 10^{38} = 1.14801 \cdot 10^{39}$	$2.09 \cdot 10^{35}$
LAGRANGE/GOLDHORN	101583	18.49

Table 2.4: Comparison of hypothesis space complexities for the problem of equation discovery in the aquatic ecosystem domain

Chapter 3

LAGRAMGE

The declarative bias formalism defined in the previous chapter is implemented in the equation discovery system LAGRAMGE that takes as input a context free grammar and a set of measured data, then (heuristically) searches the equation space defined by the grammar.

The rest of the chapter is organized as follows. In the first section, we present a refinement operator on derivation trees and an algorithm for enumerating derivation trees up to an h_{max}. Both are used in LAGRAMGE for exploring the hypothesis space of equations. The use of the refinement operator is illustrated on the example from the previous chapter. In the following sections, the procedures for constant parameters' fitting, the heuristic functions and the stopping criteria used in LAGRAMGE are presented. After a brief description of the top-level algorithm, we conclude the chapter with a section on LAGRANGE's implementation in the C programming language.

3.1 Ordering the hypothesis space

In order to allow the use of different search strategies for exploring the hypothesis space [Dehaspe and DeRaedt 1995], we define a refinement operator on derivation trees. It orders the derivation trees according to their height. The defined refinement operator is complete and thus can be also used for exhaustive search through the hypothesis space. The completeness means that all derivation trees can be generated starting with the shallowest tree and applying the refinement operator.

On the other hand, the defined refinement operator is not optimal, in the sense that each derivation tree can be a refinement of two or more different trees. Therefore, the use of this refinement operator for exhaustive search will cause us to consider some of the derivation trees more then once. To avoid this, an algorithm for enumerating derivation trees is presented that enumerates each derivation tree exactly once.[1]

We should also note here that the refinement operator on derivation trees, defined in this section, does not define any generality relation. The refinement operators on clauses

[1] If the context free grammar is ambiguous, expressions with more then one derivation tree will be still considered more then once.

in first order logic are based on *lgg* (least general generalization) relation of clauses. This property allows different optimizations of the search in ILP systems based on pruning the unpromising branches of the hypothesis spaces. These optimizations can not be used in LAGRAMGE's search process.

In the following subsections we use the notion of height of a production, and an ordering of the productions according to it. The height $h(p)$ of a production p is the minimal height of a derivation tree with production p used to expand its root node. The height of the production $p = A \to A_1 A_2 \ldots A_k$ can be calculated as follows:

$$h(p) = 1 + \max_{i=1}^{k} \{h(A_i)\}, \text{ where } h(A) = \begin{cases} \min_{q \in P_A}\{h(q)\} & ; \quad A \in N \\ 0 & ; \quad A \in T \end{cases}.$$

Using the height of productions we order the productions for each nonterminal A, i.e. the sets P_A, in ascending order. Furthermore, $p_{A,i}$ denotes the i-th production in the ordered set P_A.

Consider the grammar in Table 2.2 again. From the equation for calculating the height of a production, it is evident that $h(p) = 1$ iff $p = A \to w : w \in T^*$. In the example grammar there are four such productions, one of them is for the nonterminal E and the rest are for the nonterminal v. Thus, $h(p) = 2$ iff $p = A \to w : w \in (T \cup \{E, v\})^*$. We continue until the heights of all productions in P are calculated. The heights of the productions of the example grammar are given in Table 3.1.

h	productions with height h
1	$E \to const; v \to nut; v \to phyt; v \to zoo$
2	$Y \to \texttt{monod}(const, v); F \to v$
3	$F \to Y; F \to v \cdot Y; E \to const \cdot F; E \to E + const \cdot F$

Table 3.1: Heights of productions of the example grammar

3.1.1 Refinement operator on derivation trees

We define a refinement operator on derivation trees of the context free grammar G with the algorithm in Table 3.2. Given a derivation tree $\mathcal{T}$, the algorithm generates a refinement of $\mathcal{T}$. Optionally, the height of the refinement can be limited by the parameter h_{max}. The algorithm chooses a nonterminal node from the derivation tree (line 1) and tries to replace the production used to expand that node with the next production in the ordered set of productions for the same nonterminal, taking care that the height limit h_{max} is not exceeded (lines 2-7). To expand each remaining nonterminal leaf node in the tree, the algorithm uses the first production in the ordered set of productions for the nonterminal in that node (lines 8-12).

procedure `RefinementOperator`$(G, \mathcal{T}, h_{max})$
  choose a nonterminal node A in $\mathcal{T}$
  let $p_{A,i}$ be the production applied at that node
  $l :=$ length of the path from A to the root node of $\mathcal{T}$
  delete the subtrees of node A
  if $i + 1 \leq |P_A|$ **and** $l + h(p_{A,i+1}) \leq h_{max}$ **then**
    let $p_{A,i+1} = A \rightarrow A_1 A_2 \ldots A_k$
    expand the node A with the successors $A_1, A_2, \ldots A_k$
    repeat
      choose nonterminal leaf node B in $\mathcal{T}$
      let $p_{B,1} = B \rightarrow B_1 B_2 \ldots B_j \in P_B$
      expand the node B with the successors $B_1, B_2, \ldots B_j$
    until all leaf nodes in $\mathcal{T}$ are terminals
    print out the leaf nodes in $\mathcal{T}$ in order from left to right
  endif

Table 3.2: Refinement operator on derivation trees

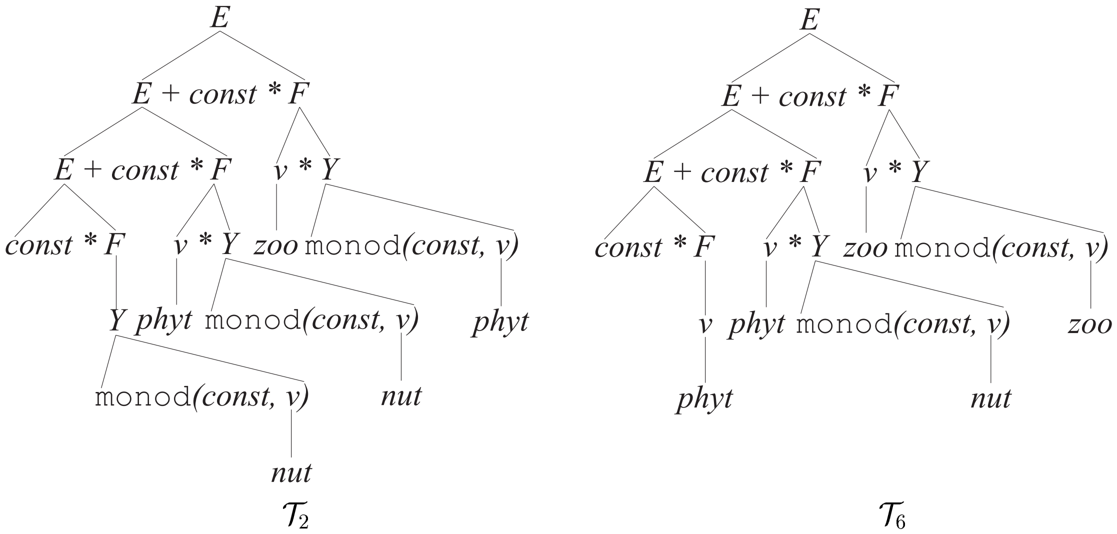

Figure 3.1: Two refinements of the derivation tree from Figure 2.1

The branching factor of the defined refinement operator varies for each derivation tree. It is equal to the number of internal (non leaf) nodes of the derivation tree in the case $h_{max} = \infty$, but can be smaller when some finite height limit is applied.

The trees $\mathcal{T}_2$ and $\mathcal{T}_6$ on Figure 3.1 are two refinements of the derivation tree from example in the previous chapter (see Figure 2.1). There are six possible refinements for that derivation tree. We can get these refinements by the following replacements of the productions in the appropriate nodes of the original derivation tree:

$\mathcal{T}_1$	$v \to phyt \mapsto^\rho v \to zoo$	$\mathcal{T}_2$	$F \to v \mapsto^\rho F \to Y$
$\mathcal{T}_3$	$E \to const * F \mapsto^\rho E \to E + const * F$	$\mathcal{T}_4$	$v \to phyt \mapsto^\rho v \to zoo$
$\mathcal{T}_5$	$v \to nut \mapsto^\rho v \to phyt$	$\mathcal{T}_6$	$v \to phyt \mapsto^\rho v \to zoo$

3.1.2 Enumerating derivation trees

procedure `EnumerateDerivationTrees`(G, h_{max})
1 begin with the derivation tree $\mathcal{T}$ consisting of node $S^{(0)}$
2 $exit$:= **false**
3 **repeat**
4 let α be the expression generated by $\mathcal{T}$
5 **if** $\alpha \in T^*$ **then**
6 print out α
7 delete the subtrees of the ancestor of rightmost leaf node in $\mathcal{T}$
8 **else**
9 let $A^{(i)}$ be the leftmost nonterminal leaf node in α
10 l := length of the path from $A^{(i)}$ to the root node of $\mathcal{T}$
11 **if** $i + 1 \leq |P_A|$ **and** $l + h(p_{A,i+1}) \leq h_{max}$ **then**
12 let $p_{A,i+1} = A \to A_1 A_2 \ldots A_k$
13 change label of the node $A^{(i)}$ to $A^{(i+1)}$
14 expand the node $A^{(i)}$ with the successors $A_1^{(0)}, A_2^{(0)}, \ldots A_k^{(0)}$
15 **else**
16 **if** $A^{(i)}$ is the root node of $\mathcal{T}$ **then**
17 $exit$:= **true**
18 **else**
19 delete the subtrees of the ancestor of node $A^{(i)}$
20 **endif**
21 **endif**
22 **endif**
23 **until** $exit$

Table 3.3: Enumeration of derivation trees

The algorithm for enumerating derivation trees of the context free grammar G with height limited with a maximal value h_{max} is given in Table 3.3. The algorithm starts with the tree consisting of one node labelled $S^{(0)}$ (line 1). Beside a nonterminal or terminal symbol, the label of node contains also an index of the production currently used to expand that node. Nodes that are not yet expanded have index 0. The algorithm expands the nonterminal leaf nodes of the derivation tree in order from left to right (line 9). If all the leaf nodes are terminals, then the generated expression is enumerated (lines 5 and 6). Otherwise, the algorithm backtracks through internal nodes of the tree and replaces the current production used in the node with the next one from the ordered set of productions for the same nonterminal, if the height limit h_{max} is not exceeded (lines 7-14 and 19). The algorithm stops when an attempt to backtrack beyond the root node is made (lines 16 and 17).

3.2 Constant parameters' fitting

Expressions generated by the context free grammar G contain one or more special terminal symbols *const*, used to denote a constant parameter in an equation that has to be fitted to the measurement data. In general, the generated expressions are non-linear in these parameters, so a non-linear fitting method $\mathcal{F}$ is applied to determine the values of these parameters. The fitting method minimizes the value of the error function $Error(\mathbf{c})$, i.e., if $\mathbf{c}$ is the vector of constant parameters in expression E, then the result of the fitting algorithm is a vector of parameter values $\mathbf{c}^*$, such that:

$$\mathbf{c}^* = \mathrm{argmin}_{\mathbf{c} \in R^{dim(\mathbf{c})}} \{Error(\mathbf{c})\}.$$

The error function $Error(\mathbf{c})$ is the sum of squared errors function, defined in the following manner:

- for a differential equation of the form $dv_d/dt = \dot{v_d} = E$:

$$Error(\mathbf{c}) = \sum_{i=0}^{m} \left[v_{d,i} - \left(v_{d,0} + \int_{t_0}^{t_i} E(\mathbf{c}, v_1, \ldots v_n) dt \right) \right]^2,$$

- for an ordinary equation of the form $v_d = E$:

$$Error(\mathbf{c}) = \sum_{i=0}^{m} (v_{d,i} - E(\mathbf{c}, v_{1,i}, \ldots v_{d-1,i}, v_{d+1,i}, \ldots v_{n,i}))^2,$$

where m is the size of the measurement table and $v_{j,i}$ the value of the system variable v_j at time t_i. When more then one measurement table is provided in the input domain, the error function is calculated as the sum of the error functions of all measurement tables.

Note that in the case of calculating the error function for differential equations we use the integral of the expression on the right hand side of the equation instead of the derivative of the dependent variable. This is done because the numerical error involved in algorithms for numerical integration is in general smaller than the numerical error involved in numerical calculation of the derivatives [Križman et. al 1995, Bohte 1987]. We use a simple trapezoid formula for numerical integration [Bohte 1987]:

$$\int_{t_{i-1}}^{t_i} E(\mathbf{c}, v_1, \ldots v_n)dt \approx \frac{t_i - t_{i-1}}{2} \left[E(\mathbf{c}, v_{1,i-1}, \ldots v_{n,i-1}) + E(\mathbf{c}, v_{1,i}, \ldots v_{n,i})\right]$$

The downhill simplex and Levenberg-Marquardt [Press et al. 1986] algorithms are used for minimization of the error function $Error(\mathbf{c})$.

3.3 Heuristic functions

After fitting, the value of the heuristic function $\mathcal{H}$ of the expression is evaluated. It is based on the sum of squared errors value SSE calculated by the fitting method:

$$SSE(E) = Error(\mathbf{c}^*) = \min_{\mathbf{c} \in R^{dim(\mathbf{c})}} \{Error(\mathbf{c})\}.$$

An alternative heuristic function MDL can be used, that takes into account the length l of E:

$$MDL(E) = SSE(E) + \frac{l}{10 \cdot l_{max}} \sigma_{v_d},$$

where l_{max} is the length of the largest expression generated by the grammar with height of the derivation tree up to h_{max} and σ_{v_d} is the standard deviation of the measured values for the dependent variable v_d.

3.4 Stopping criteria

Within LAGRAMGE three different stopping criteria $\mathcal{C}$ can be used:

1. **false**

   LAGRAMGE stops when all trees have been considered (exhaustive search) or no improvment can be achieved by applying new search steps (heuristic search strategies).

2. $\mathcal{H}(E) \leq T_{\mathcal{H}}$,

   where E is the first expression with a best value of the heuristic function $\mathcal{H}$ among the expressions considered so far, and T is a user defined threshold.

3. $elapsed\_time > T_t$.

   LAGRAMGE stops when a user defined CPU time limit T_t is exceeded.

```
   procedure LAGRAMGE(D, G, h_max, S, w_b, F, H, C)
1     switch (S)
2     case exhaustive:
3        let T be the shallowest derivation tree
4        Q := ∅
5        repeat
6           fit the constant parameter values of expression E,
                  generated by T, with F
7           evaluate expression E according to H
8           Q := Q ∪ {T}
9           retain the w_b best derivation trees in Q
10          T := next enumerated derivation tree in G
11       until all derivation trees enumerated or C
12    case beam_search:
13       let T_0 be the shallowest derivation tree
14       Q := {T_0}
15       repeat
16          Q_1 := Q
17          R := set of all refinements of derivation trees in Q
18          fit the constant parameter values of expressions,
                  generated by trees in R, with F
19          evaluate expressions, generated by trees in R, according to H
20          Q := Q_1 ∪ R
21          retain the w_b best derivation trees in Q
22       until Q = Q_1 or C
23    endswitch
24    print out expressions generated by trees in Q
```

Table 3.4: Top-level LAGRAMGE algorithm

3.5 Top-level algorithm

In addition to the domain definition D and grammar G, LAGRAMGE takes as input the maximal height h_{max} of the derivation trees used for deriving expressions, search strategy $\mathcal{S}$, the beam width w_b, a fitting method $\mathcal{F}$, a heuristic function $\mathcal{H}$ and a stopping criterion $\mathcal{C}$. The top level algorithm of LAGRANGE is given in Table 3.4.

On the basis of the value of parameter $\mathcal{S}$ (line 1) LAGRAMGE uses the exhaustive (lines 2-11) or beam search (lines 12-22) procedure on the hypothesis space of derivation trees defined by the context free grammar G.

3.6 Implementation

LAGRAMGE is implemented in the C programming language using the Gnu `gcc` compiler. It was developed on a Hewlett Packard workstation working under the HP-UX operation system. It was successfully compiled and run on Hewlett Packard, Sun and PC (under the Windows 95/NT operating systems) workstations.

The process of equation discovery with LAGRAMGE is broken in two phases. In the first phase a context free grammar is compiled for use in LAGRAMGE with the command:

`lgc` *gramm-file-name*

where a parameter *grammar-file-name* is the name of the file that contains the context free grammar along with the definitions of the background knowledge functions used. The format of the file is described in Appendix A. The second, main phase starts LAGRAMGE on the domain at hand:

`lg -v` *variable-name* `[-h` *time-step*`] [-c` *conf-file-name*`]` *domain-file-name* `[`*domain-file-name* `...]`

The command accepts the following switches and parameters:

`-v` *variable-name* - determines a dependent variable from the domain. The use of this switch is obligatory, because one of the system variables has to be distinguished as dependent.

`-h` *time-step* - provides a time step between two successive measurements for discovering differential equations. If this switch is omitted, ordinary equations are sought by LAGRAMGE.

`-c` *conf-file-name* - provides the name of the file with the LAGRAMGE's parameters. If this switch is omitted the default configuration file `default.cfg` is used:

```
search_strategy: beam
fitting_algorithm: amoeba
heuristic_function: sse
stopping_criterion: none
max_derivation_tree_depth: 5
beam_width: 25
```

By default, LAGRAMGE uses beam search strategy with beam width of 25, uses the downhill simplex method for constant parameters' fitting, the *SSE* heuristic function, the stopping criterion $\mathcal{C} = \textbf{false}$ and maximal height of derivation trees $h_{max} = 5$. For more information on the format of the configuration parameters' file see Appendix A.

domain-file-name(s) - one or more file names. Each file contains a table of measurements of the domain variables at successive time points.

The output of LAGRAMGE is the list of best equations found, along with their heuristic function values, the number of considered derivation trees and the elapsed CPU time.

Chapter 4

Experiments with Lagramge

Lagramge was applied to three synthetic and two real world problem of equation discovery. The problems were chosen to illustrate the capabilities of Lagramge. They all concern the behavior of dynamic systems (differential equations were thus sought by Lagramge) and involve non-linearities. These can be non-linearities in the system variables or parameters, or non-linearities arising from the use of logical conditions in the model of the dynamic system. The two poles on cart is a complex system that was out of reach (complexity-wise) of existing machine discovery systems and the phytoplankton growth domain is a real domain, which is not too complex judged by the equations structure, but was not successfully modeled by existing discovery systems. Two of the domains involve background knowledge in the form of predefined functions that can be used in the equations.

For the synthetic domains, the experimental data were obtained by simulating the original equations for ten randomly chosen initial states for a certain number of time steps. Data for real life domains were obtained with measurements made on the real system or its laboratory replica. For all problems, the data were accompanied with an appropriate bias, i.e., grammar, for the problem at hand. Derivation tree height was limited, the limit varying between 4 and 6. In all the experiments all combinations of search strategies, constant parameters' fitting, heuristic functions and stopping criteria (except for the time limit stopping criterion) were considered. The beam width was set to 25, and the stopping criterion had the threshold value 10^{-5}, when used. All experiments were run on Hewlett-Packard K-200 workstation working under HP-UX operating system.

In the rest of the chapter, we give brief description for each of the experimental domains, along with a declarative bias used and summary of results achieved with Lagramge. For the tables with complete results of experiments in synthetic domains see Appendix B.

4.1 Experiments in synthetic domains

4.1.1 Aquatic ecosystem

A simple system of differential equations describes the evolution of the concentrations of nutrient *nut*, phytoplankton *phyt* and zooplankton *zoo* in an aquatic environment [Crispi and Mosetti 1993]:

$$\begin{aligned}\dot{nut} &= -\frac{nut \cdot phyt}{k_N + nut}\\ \dot{phyt} &= \frac{nut \cdot phyt}{k_N + nut} - r_P \cdot phyt - \frac{phyt \cdot zoo}{k_P + phyt}\\ \dot{zoo} &= \frac{phyt \cdot zoo}{k_P + phyt} - r_Z \cdot zoo.\end{aligned}$$

It was simulated with the parameters $k_N = k_P = 1$, $r_P = r_Z = 0.05$, from ten randomly selected initial states, for 100 time steps of 0.1. Figure 4.1 shows the evolution of the concentrations of nutrient, phytoplankton and zooplankton through time.

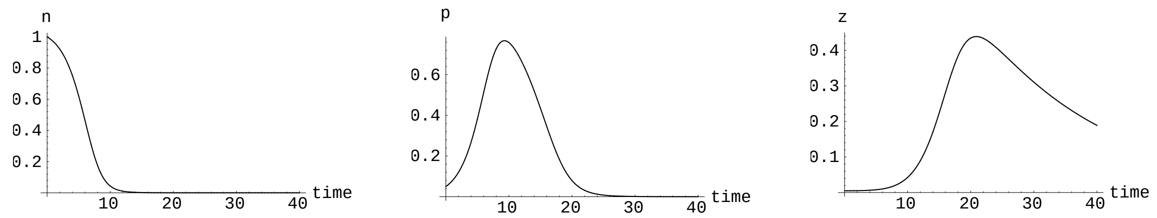

Figure 4.1: Evolution of the aquatic ecosystem

The grammar from Table 2.2 was used. Note that the `monod` function is defined, which represents background knowledge about population growth that comes from the area of ecological modeling. This function is used to describe population growth where `v` is a nutrient variable and `c` a saturation constant. The number of derivation trees $N_G(E, h)$ with height below or equal to h is given in the Table 2.3.

In the experiment with beam search, downhill simplex fitting, *SSE* heuristic and without stopping criterion applied, LAGRAMGE successfully rediscovered the correct equations in about 50 minutes of CPU time, visiting 1525 expressions. LAGRAMGE failed to rediscover the equation for $\dot{phyt}$ when using *MDL* heuristic function, and discovered equation similar to, but simpler than the original one when the stopping criterion was applied. The equation for $\dot{zoo}$ was successfully rediscovered in only one experiment. This is due to the slow dynamic of the zooplankton at the beginning of the evolution of the ecosystem (see

Figure 4.1). As mentioned before, we simulated the original equations for only first 100 time steps of 0.1, which gives 10 time units of simulation trace for training data.

LAGRAMGE rediscovered equation of the aquatic ecosystem domain in their original comprehensible form. Equation discovery systems LAGRANGE and GOLDHORN are also capable of rediscovering these equations, but the will be rediscovered in the incomprehensible implicit form of multivariate polynomials.

4.1.2 Piecewise-linear circuit

A simple piecewise-linear circuit [Baghious and Jarry 1993] which exhibits a chaotic attractor similar to that of the Lorenz equations can be described by the following three state equations:

$$\begin{aligned} \dot{x} &= \sigma(y - x) \\ \dot{y} &= \mu_1(r - z) - qy \\ \dot{z} &= \mu_2 x - bz \end{aligned}$$

where $\mu_1 = \operatorname{sgn} x$, $\mu_2 = \operatorname{sgn} y$.

We simulated the equations for 100 time steps of $h = 0.1$ from ten randomly chosen initial states. The parameter values were $\sigma = 1.2$, $r = 7$, $q = 0.1$ and $b = 0.1$. The chaotic behavior of the system is shown in Figure 4.2.

The grammar from Table 4.1 was used in the experiments. Note that the `If` function is defined as background knowledge. This example illustrates the possibility to use logical conditions in the grammars and equations.

```
double If(double v, double e1, double e2) {
return((v > 0) ? e1 : e2);
}
```

$$\begin{aligned} \mathit{IfE} &\rightarrow E \mid \mathtt{If}(v, E, E) \\ E &\rightarrow \mathit{const} \mid \mathit{const} \cdot v \mid E + \mathit{const} \cdot v \end{aligned}$$

h	1	2	3	4
$N_G(\mathit{IfE}, h)$	0	4	154	1900

Table 4.1: Declarative bias and hypothesis space complexity for piecewise-linear circuit domain

In the experiment with beam search, Levenberg-Marquardt fitting, SSE heuristic and threshold stopping criterion the correct equations were discovered in about 12 minutes of CPU time, considering 1606 derivation trees. LAGRAMGE successffully rediscovered original equations with both exhaustive and beam search strategy. The search space was reduced by factor 3 when the beam search was used. In the experiments with *MDL* heuristic function the discovered equations for $\dot{y}$ and $\dot{z}$ was similar to, but simpler then the original ones.

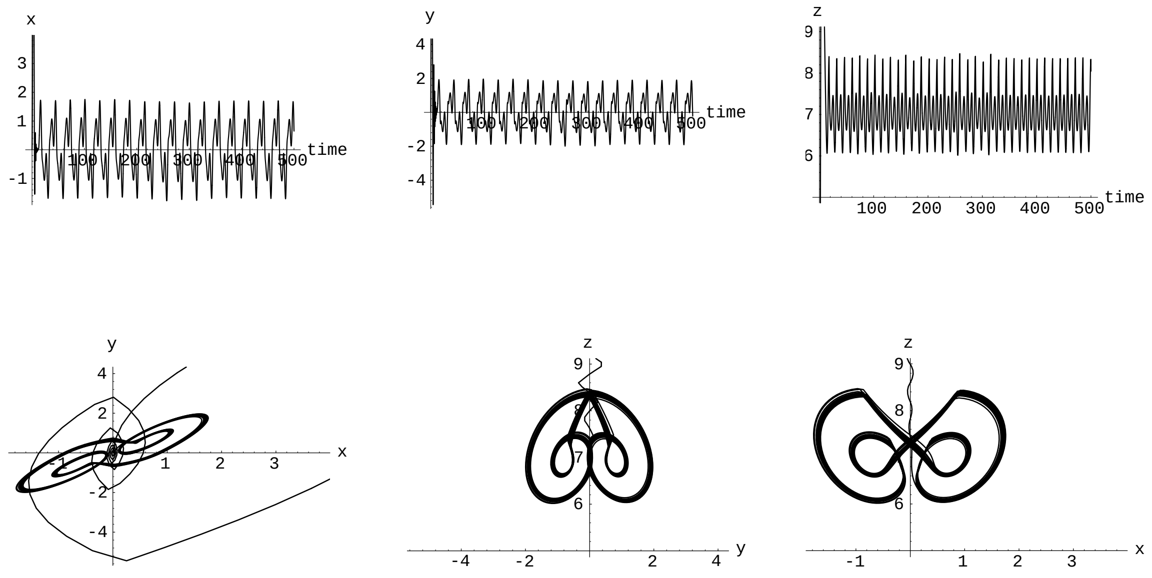

Figure 4.2: Chaotic behavior of the piecewise-linear circuit

4.1.3 Two poles on cart

The task of balancing the inverted pendulum (pole on cart) is a standard benchmark problem for testing classical and modern approaches to the control of dynamic systems [Geva and Sitte 1993, Urbančič and Bratko 1994]. The inverted pendulum consists of a cart that can move along a horizontal track, and a pole hinged on top of the cart, so that it can rotate in the vertical plane defined by the track and its fixed point. A force parallel to the track can be applied to the cart. The inverted pendulum is the most complicated system whose dynamics were successfully modeled by LAGRANGE [Todorovski 1993, Džeroski and Todorovski 1993].

The two poles on cart system complicates the inverted pendulum by hinging a second pole on top of the first (see Figure 4.3). The top pole can rotate in the vertical plane around the point where it is attached to the bottom pole. The system variables in this case are x, the distance of the cart from the origin point on the track, φ, the inclination angle of the bottom pole relative to the vertical line through its fixed point, and ϑ, the inclination angle of the top pole. The dynamics of this system can be described by a system of second order differential equations.

$$2(M + m_1 + m_2)\ \ddot{x} + l_1(m_1 + 2m_2)\ \ddot{\varphi}\cos\varphi + m_2 l_2\ \ddot{\vartheta}\cos\vartheta =$$
$$2F + (m_1 + 2m_2) l_1\ \dot{\varphi}^2 \sin\varphi + m_2 l_2\ \dot{\vartheta}^2 \sin\vartheta$$

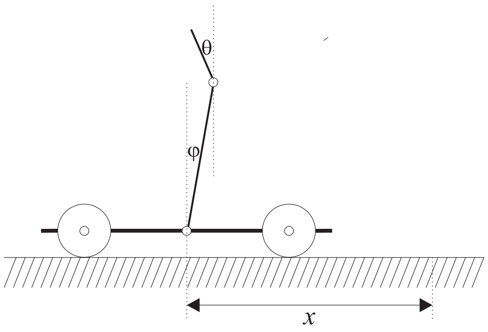

Figure 4.3: Two poles on cart

$$
\begin{gathered}
(m_1 + 2m_2)\ \ddot{x}\cos\varphi + 2l_1(m_1/3 + m_2)\ \ddot{\varphi} + m_2 l_2\ \ddot{\vartheta}\cos(\vartheta - \varphi) = \\
m_2 l_2\ \dot{\vartheta}^2 \sin(\vartheta - \varphi) + (m_1 + 2m_2) g\ \sin\varphi \\
\ddot{x}\cos\vartheta + l_1\ \ddot{\varphi}\cos(\vartheta - \varphi) + 2/3\ \ddot{\vartheta} = g\ \sin\vartheta - l_1\ \dot{\varphi}^2 \sin(\vartheta - \varphi)
\end{gathered}
$$

In the equations, M, m_1 and m_2 are the masses of the cart, the bottom, and the top pole, respectively, l_1 and l_2 the lengths of the poles, F the force applied to the cart, and $g = 9.81$ is the gravitational acceleration.

Ten simulations runs were performed, each of 50 time steps of $h = 0.01$, starting from a randomly chosen initial state. No control force was applied, i.e., $F = 0$. The parameters' values were chosen as follows: $M = 2[kg]$, $m_1 = 1[kg]$, $m_2 = 0.5[kg]$, $l_1 = 1[m]$, $l_2 = 0.5[m]$.

The grammar below was used for $\ddot{x}$. The grammar for $\ddot{\varphi}$ replaces the production for $\mathit{Facc_1}$ with $\mathit{Facc_1} \to \ddot{x} \cdot \cos A \mid \ddot{x}$, while the grammar for $\ddot{\vartheta}$ replaces the production for $\mathit{Facc_2}$ with $\mathit{Facc_2} \to \ddot{x} \cdot \cos A \mid \ddot{x}$.

$$
\begin{array}{lcl}
E & \to & \mathit{Eacc} + \mathit{Evel} \\
\mathit{Eacc} & \to & \mathit{const} \cdot \mathit{Facc_1} + \mathit{const} \cdot \mathit{Facc_2} \\
\mathit{Facc_1} & \to & \ddot{\varphi} \cdot \cos A \mid \ddot{\varphi} \\
\mathit{Facc_2} & \to & \ddot{\vartheta} \cdot \cos A \mid \ddot{\vartheta} \\
\mathit{Evel} & \to & \mathit{Evel} + \mathit{const} \cdot \mathit{Fvel} \mid \mathit{const} \cdot \mathit{Fvel} \\
\mathit{Fvel} & \to & \mathit{Vvel_2} \cdot \sin A \mid \mathit{Vvel_2} \mid \sin A \\
\mathit{Vvel_2} & \to & \dot{\varphi} \cdot \dot{\varphi} \mid \dot{\vartheta} \cdot \dot{\vartheta} \\
A & \to & \varphi \mid \vartheta \mid \varphi - \vartheta
\end{array}
$$

h	1	2	3	4	5	6
$N_G(E,h)$	0	0	0	0	176	2122

Table 4.2: Declarative bias and hypothesis space complexity for two poles on cart domain

While the above grammar looks complicated, it can be constructed easily with a modest knowledge of physics, considering the fact that the target equations are equations of Lagrangian dynamics.

In the experiment with beam search, Levenberg-Marquardt fitting, *SSE* heuristic and without stopping criterion applied, LAGRAMGE discovered the correct equations in about 9 minutes of CPU time, visiting 1840 expressions. LAGRAMGE successfully rediscovered original equations with both exhaustive and beam search strategy. The search space was reduced by factor 3 when the beam search was used. In the experiments with *MDL* heuristic function the discovered equations for $\ddot{\varphi}$ and $\ddot{\vartheta}$ was similar to, but simpler then the original ones.

The equations for two poles on cart are beyond capacities of the existing equation discovery systems LAGRANGE and GOLDHORN because of the enormous hypothesis space complexity of the domain. A calculation, similar to the one from Section 2.3.2, shows that LAGRANGE and GOLDHORN have to consider 185,111,450,355 different equations for this domain. LAGRAMGE successfully identified the dynamics of a system much more complicated than systems considered so far in equation discovery, due to the use of stong declarative language bias in the form of a context free grammar.

4.2 Experiments in real domains

4.2.1 Surge tank

A problem of modelling water behavior in a surge tank is considered in this section. A surge tank is a container usually connected to one or more pressure pipes that conduct water to the turbines of a hydro-power plant. It is placed as close as possible to the place in the plant's pipeline where a formation of a surge is expected. Surge pressure is transformed to water movement in the tank, resulting in oscillations of the water level in it [Križman et. al 1995, Karalič 1995].

The measurements of the water level h used in the experiments were done in a laboratory replica model of a surge tank. The best theoretical model is known to be [Križman et. al 1995]:

$$\ddot{h} = -0.05608h - 3.75771\dot{h}|\dot{h}|.$$

This model can be derived from the grammar shown in Table 4.3.

$$\begin{array}{lcl} E & \to & const \mid const \cdot F \mid E + const \cdot F \\ F & \to & T \mid T \cdot T \\ T & \to & v \mid \texttt{fabs}(v) \end{array}$$

h	1	2	3	4	5
$N_G(E,h)$	1	1	1	41	841

Table 4.3: Declarative bias and hypothesis space complexity for surge tank domain

In all the experiments an equation equivalent to the theoretical equation was discovered by LAGRAMGE. When using *SSE* heuristic function the best equation discovered by LAGRAMGE was $\ddot{h} = -3.06302 \cdot 10^{-5} - 0.0569615h - 3.42417\dot{h}|\dot{h}|$, and $\ddot{h} =$

${}_{t}\Delta_m = 6.979$
${}_{l_{SSE}}\Delta_m = 15.054$
${}_{l_{MDL}}\Delta_m = 13.906$
${}_{g}\Delta_m = 37.063$
${}_{g_{BEST}}\Delta_m = 8.282$
${}_{f}\Delta_m = 18.245$
${}_{f_{BEST}}\Delta_m = 6.075$

Table 4.4: Comparison of different equations for surge tank domain

$-0.056333h - 3.36659\dot{h}|\dot{h}|$, when using *MDL* heuristic function. First equation was discovered with beam search after considering 601, and second after considering 576 expressions (approximately 3.5 minutes of CPU time for each equation). The same equations were discovered when using exhaustive search, different parameter optimization procedures and with or without threshold stopping criterion applied.

We compared the performance of discovered equations with the performance of equations discovered by GoldHorn and Fors [Karalič 1995]. The equations were simulated starting with initial state $h(0) = -0.216$, $\dot{h}(0) = 0.06253988333$. As a measure of model's fitness we use the difference between the behavior according to the equation and measured behavior. The difference measure ${}_{a}\Delta_{b}$ between behaviors a and b is defined as [Karalič 1995]:

$$ {}_{a}\Delta_{b} = \sum_{i=0}^{N} |h_a(t_i) - h_b(t_i)|, $$

witch represents the numerical aproximation of the area between two behaviors, when they are plotted on the same graph.

The comparison between the following equations was done: theoretical equation (t), equation discovered with Lagramge (l_{SSE} and l_{MDL}), equations discovered with GoldHorn from unfiltered and filtered data (g and g_{BEST}) and equations discovered with Fors (f is best equation discovered according to Fors heuristic function and f_{BEST} is the equation according to the difference measure used for this comparison). The results of comparison are summarized in Table 4.4.

We can see from the table that equation discovered with *MDL* heuristic function is better, which was expected for the real domain with noisy measurement data. Since we do not use any filtering of the data, we can also observe that Lagramge discovered much better equation then the one discovered by GoldHorn. However, GoldHorn equation discovered with using its sophisticated filtering methods is more accurate. Similar comparison stands for Fors. Both equations discovered with Lagramge are better then the best equation discovered with Fors, according to the value of Fors's heuristic function. Equations discovered by Fors were additionally evaluated according to the difference from

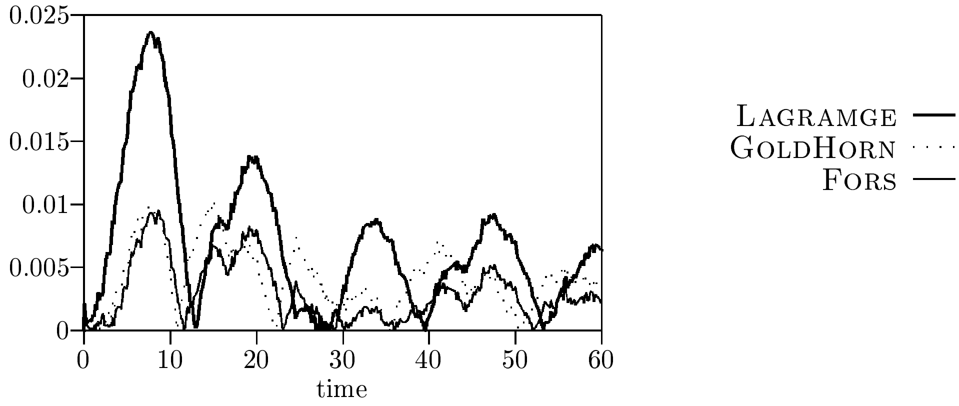

Figure 4.4: Absolute errors of best equations

the measured behavior. Best equation selected that way, is better then the LAGRANGE's. The absolute differences of the best equations' behaviors (f_{BEST}, g_{BEST} and l_{MDL}) from the measured behavior are plotted on the graph in Figure 4.4.

4.2.2 Phytoplankton growth in Lake Glumsoe

The final problem we consider is a real world problem of modeling the dynamics of phytoplankton growth in the Danish lake Glumsoe [Jørgensen et al. 1986]. Lake Glumsoe is situated in a sub-glacial valley in Denmark. It is shallow with average depth of about 2 m and its surface area is 266,000 m^2. For several years, it was receiving mechanically-biologically treated waste water from a community with 3,000 inhabitants and a surrounding area which was mainly agricultural with almost no industry. The high nitrogen and phosphorus concentration in the treated waste water has caused hypereutrophication. The lake contained no submerged vegetation, probably due to the low transparency of the water and oxygen deficit at the bottom of the lake.

Concentrations of phytoplankton (*phyt*), zooplankton (*zoo*), soluble nitrogen (*nitro*) and soluble phosphorus (*phosp*) were considered relevant for modeling the phytoplankton growth. State variables were measured at 14 distinct time points, over a period of two months. The amount of measured data itself was far too small for equation discovery, so additional processing was applied to obtain a dataset suitable for equation discovery [Kompare 1995, Demšar 1996]. First, dotted graphs of the measurements were plotted and given to three human experts to draw a curve that, in their own opinion, described the dynamic behavior of the observed system variable between the measured points. A properly plotted expert curve can be regarded as an additional source of reliable data. Curves drawn by the human experts were then smoothed with Besier splines. Finally, three data sets were obtained by sampling the splines derived from each of the three human expert' approximations at regular time intervals with time step $h = 0.1$ day. The

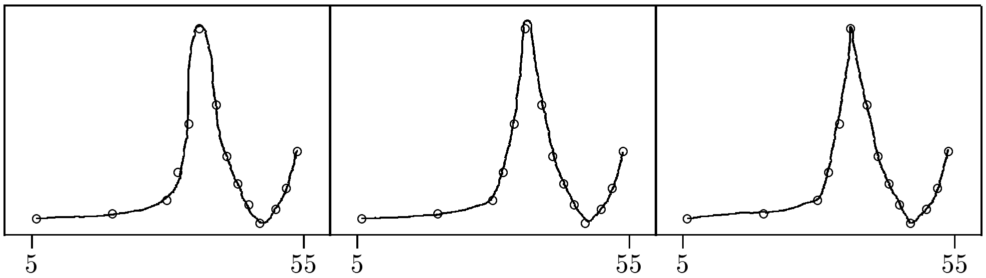

Figure 4.5: Phytoplankton growth as seen by three domain experts

dynamic behavior of the phytoplankton as represented by each of the three data sets is shown in Figure 4.5.

The grammar given in Table 4.5 was used in the experiments. It was constructed by taking into account ecological background knowledge on algal growth [Monod 1942, Crispi and Mosetti 1993]. Phosphorus and nitrogen are nutrients for phytoplankton and can thus appear in `monod` terms (productions for nonterminals Y and v_Y). Other terms describe the decay of phytoplankton ($-const \cdot phyt$) and the feeding of zooplankton on phytoplankton ($-const \cdot phyt \cdot zoo$). At the maximum derivation depth 4 used in our experiments, 72 equations can be derived from the grammar. The values of the constant parameters in the equations specified by the grammar are constrained to be positive.

```
double monod(double c, double v) {
        return(v / (v + c));
}
```

$$
\begin{array}{lcl}
E & \rightarrow & F \cdot phyt - const \cdot phyt \mid F \cdot phyt - const \cdot phyt - const \cdot phyt \cdot zoo \\
F & \rightarrow & const \cdot Y \cdot Y \mid const \cdot Y + const \cdot Y \mid const \cdot Y \\
Y & \rightarrow & \mathrm{monod}(const, v_Y) \mid v_Y \\
v_Y & \rightarrow & phosp \mid nitro
\end{array}
$$

Table 4.5: Context free grammar used for the Lake Glumsoe domain

In a first set of experiments we used the 'leave one out' testing method: Lagramge was given two sets of data for equation discovery, and the best equation discovered was then tested on the remaining data set. The equation was tested on the task of predicting phytoplankton growth.

In experiments with the *MDL* heuristic function (all possible 72 equations were considered), the best equation discovered by Lagramge was chosen that satisfied the constraints for the parameters' values. The three equations obtained had the same structure:

$$\dot{phyt} = const_1 \cdot phyt \cdot \frac{phosp}{const_2 + phosp} - const_3 \cdot phyt.$$

The structure of the equations discovered makes sense from an ecological point of view. It tells us that phosphorus is a limiting factor for phytoplankton growth in the lake.

Training data sets	$const_1$	$const_2$	$const_3$	r
1, 2	0.617	0.101	0.442	0.9994
1, 3	0.763	0.0797	0.592	0.9989
2, 3	0.383	0.444	0.155	0.9996

Table 4.6: Constant parameters' values and correlation coefficients for equations discovered by Lagramge

We used the obtained equations for predicting the phytoplankton concentration in the lake on the testing set and then calculated the correlation coefficient between the measured and predicted values. The constant parameter values, as well as calculated correlation coefficients are shown in Table 4.6. It can be seen that all equations give accurate short term predictions for phytoplankton growth. Note, however, the differences in the values of the equation coefficients, which indicate that experts approximated the dynamics of phytoplankton growth in quite different ways. Furthermore, we tested the robustness of the predictor on increasing the prediction period. The summary of the results (correlation coefficients between measured and predicted values) for prediction periods of one, two and five days are given in Table 4.7.

Training data sets	1 day	2 days	5 days
1, 2	0.9836	0.9413	0.7243
1, 3	0.9849	0.9552	0.8137
2, 3	0.9853	0.9566	0.7267

Table 4.7: Correlation coefficients between the actual phytoplankton concentration and the concentration predicted by the discovered equations for different prediction time periods.

Finally, we compared the accuracy of the obtained predictor with the accuracies of two simple predictors: no-change and same-change. The no-change predictor predicts that the value of the variable at the next time point will be the same as the present value ($\hat{phyt}(t+h) = phyt(t)$). The same-change predictor predicts the same change of the value of the variable, as the change in previous time step ($\hat{phyt}(t+h) - phyt(t) = phyt(t) - phyt(t-h)$). The graphs in Figure 4.6 show the dependence of correlation coefficients between the measured values and values predicted by the three different predictors for increasing prediction period for all data sets.

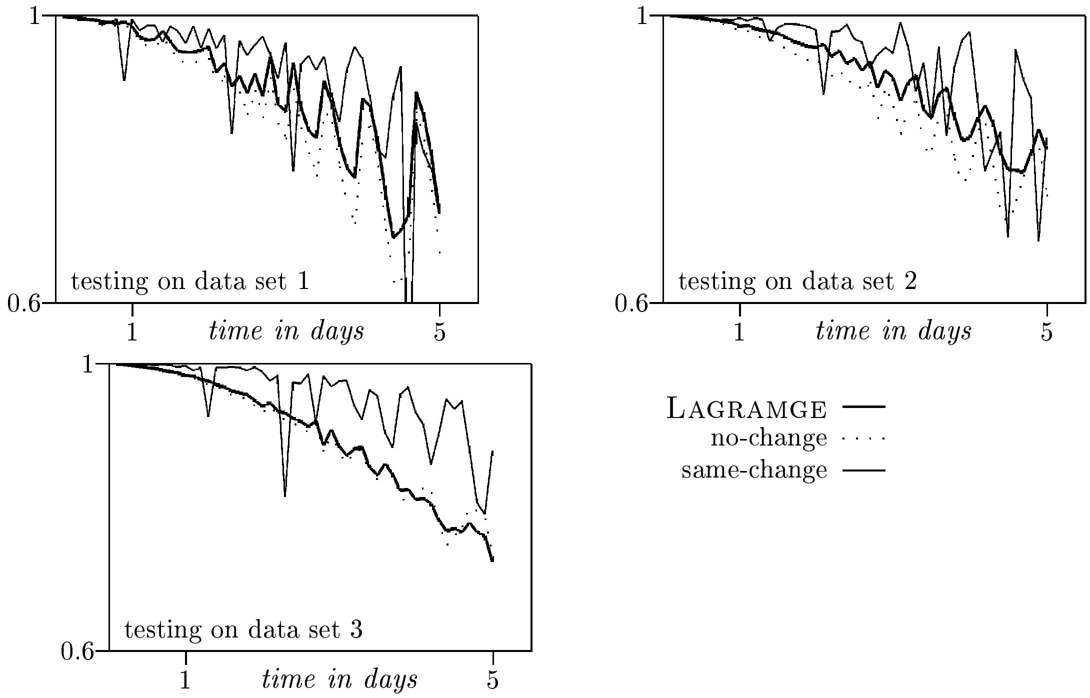

Figure 4.6: Dependence of correlation coefficients between the measured values and values predicted with the three different predictors on increasing prediction period for all experimental data sets

The graphs show that the accuracy of the predictions decreases as the prediction time increases, which could be expected. The performance and robustness of all predictors is comparable. The same-change predictor has better performance than the one obtained with LAGRAMGE, especially on the third data set, but the LAGRAMGE predictor is more robust, i.e., has smaller oscillations of performance. The no-change predictor has the lowest accuracy on all data sets.

Recently, a new data set was obtained by applying a more sophisticated smoothing method to the graph plotted by the first human expert (Figure 4.7). The new data set also includes the measurements of the temperature of the water in the lake (*temp*). Due to the fact that the second bloom might not be described by the same model, the last portion of the data was not taken into account.

The grammar used in the experiments with the new data set was the same as the one used in previous experiments (see Table 4.5), except that the new production $v_Y \rightarrow temp$ was added to allow the use of temperature in the monod terms. The best equation discovered by LAGRAMGE that satisfies the constraint for the constant parameters' values was:

$$\dot{phyt} = 0.553 \cdot temp \cdot phyt \cdot \frac{phosp}{0.0264 + phosp} - 4.35 \cdot phyt - 8.67 \cdot phyt \cdot zoo.$$

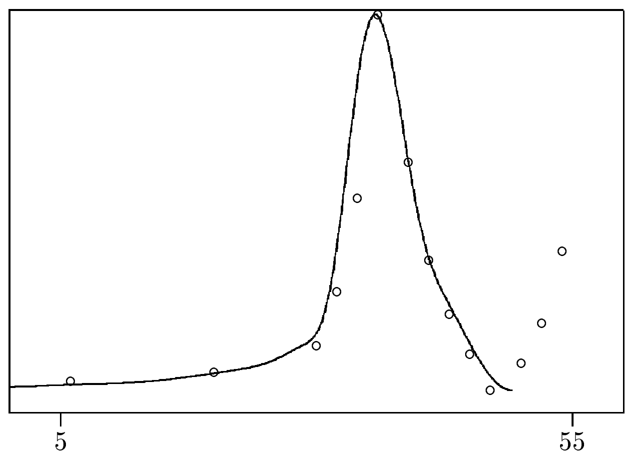

Figure 4.7: New data set for phytoplankton growth

Note that the structure of the equation discovered is similar to the structure of equations discovered on three data sets. It tells us that phosphorus and water temperature are the limiting factors for phytoplankton growth in the lake.

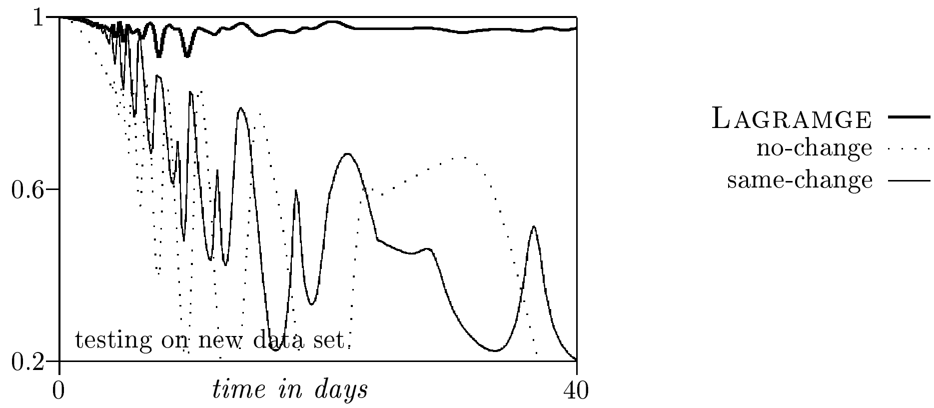

Figure 4.8: Accuracy and robustness of predictor obtained from new data set

The graph in Figure 4.8 shows the correlation coefficients between the measured values and the values predicted with three different predictors for new data set as the prediction period increased. We can observe a significant improvement of both accuracy and robustness of the predictor obtained with LAGRAMGE. The predictor is now also suitable for long-term prediction of phytoplankton growth in Lake Glumsoe.

Chapter 5

Summary and further work

We presented an equation discovery system LAGRAMGE that uses declarative language bias for restricting the hypothesis space of equations. The main advantage of declarative versus built-in language bias is that the user is allowed to adapt the learning system to the domain at hand. The language biases used in existing equation discovery systems restrict their hypothesis spaces to some manageable small hypothesis space of equations, which often affects the performance of the system (the correct equation is out of the hypothesis space) as well as the comprehensibility of discovered equations (equations equivalent to the expected ones are discovered, but they are written in different form). However, if the constraint of comprehensibility is applied, one may lose some accuracy on the training data, but if accuracy is paramount the expert can use a different, less restrictive grammar (e.g. specifying polynomial equations as in LAGRANGE and GOLDHORN). Another major point of using background knowledge is the possibility to focus the search in the space of possible equations, which allows discovery of equations with complex structure that were out of reach for previous equation discovery systems. Finally, restricting the hypothesis space prevents the overfitting of the equation to the training data and the degradation of the model's accuracy on test cases not available to the learning system during the learning process.

If plenty of measurement data are available, less restrictive bias (more general equation space) can be used. On the other hand, when less data are available, more background knowledge should be used to restrict the space of equations and compensate for the lack of data. While classical identification methods consider one equation structure and other equation discovery methods consider a large space of equations regardless of the amount of data available, LAGRAMGE offers the possibility to trade-off between measurement data and background knowledge. The size of the data set is one aspect of the data quality. The trade-off mentioned above applies to the overall quality of data. For example, where noise would cause a nonsense model to be selected based on fit, background knowledge can restrict the space of models to avoid such models. LAGRAMGE in itself has no special noise handling procedures for discovery from noisy data. Note, however, that background knowledge reduces the effect of noise, as discussed above.

Within the declarative language bias formalism presented in the thesis, background knowledge in the form of function definitions can be used along with common arithmetical operators and functions built in the C programming language. The hypothesis space of

LAGRAMGE is a set of equations, such that the expressions on their right hand sides can be derived from a given context free grammar. It is ordered with a refinement operator on derivation trees, which allows the use of different search strategies for equation discovery.

The equations considered can be non-linear in both the system variables and the constant parameters. LAGRAMGE thus uses non-linear optimization procedures, such as downhill simplex and Levenberg-Marquardt, to fit equation parameters. The search heuristic used is based on the fit of equations to measured data, but can also take into account the length of equations (MDL heuristic function). LAGRAMGE can find both ordinary and differential equations and can also take into account more than one measurement table of the system variables.

In the experiments with synthetic domains, LAGRAMGE reconstructed from simulated data three models of dynamic systems. Two of them are out of reach for existing machine discovery systems, e.g. LAGRANGE [Todorovski 1993, Džeroski and Todorovski 1993] and GOLDHORN [Križman et. al 1995]. One of them includes piece-wise linear equations, i.e., logical conditions. In the simple ecosystem domain, LAGRAMGE makes use of background knowledge defining a building block useful in ecological modeling, stemming from the monod expression. The equations equivalent to the ones for the simple ecosystem domain can be successfully reconstructed with LAGRANGE and GOLDHORN, but in the incomprehensible form of multivariate polynomials. The pole on cart domain was the most complicated domain still within reach of LAGRANGE. The two poles on cart domain, however, is well out of its reach, due to the size of the hypothesis space. The declarative bias facility of LAGRAMGE enabled us to specify a grammar based on some physics knowledge about Lagrangian dynamics, which greatly reduced the equation space. It can be also observed from the experiments that great reductions of the number of considered equations can be achieved with using the beam search strategy.

Experiments with real life domains include the problem of modeling the oscillation of water height in a surge tank and the problem of prediction of phytoplankton growth in Danish lake Glumsoe. Accuracy of the discovered model for the surge tank domain is smaller, but comparable to the models discovered by GOLDHORN and FORS. While the first domain was successfully modeled by existing equation discovery systems, GOLDHORN failed to produce satisfactory equations in the second domain [Demšar 1996]. LAGRAMGE discovered sensible and comprehensible equation from the ecological expert's point of view. The structure of the discovered equation tells us that phosphorus is a limiting factor for phytoplankton growth in the lake, along with temperature. The obtained equaiton can also be used to produce accurate predictions of the phytoplankton growth in the lake.

One direction for further work is to extend the declarative bias formalism to allow constraints of the constant parameter values according to domain knowledge. More sophisticated parameter fitting procedures would have to be used to fit constant parameters of the equations according to these constraints.

Other directions for further work concern background knowledge and declarative bias acquisition. This might include the creation of domain specific grammar libraries, e.g., for the domain of ecological modeling, as well as the use of intermediate concepts constructed by decomposition of real functions [Demšar et al. 1997].

Acknowledgements

I would like to thank my thesis supervisor professor Ivan Bratko for accepting me in his research workgroup and supporting my postgraduate study.

I am especially grateful to Sašo Džeroski for his guidance and helpful comments through all stages of my research work in the area of machine learning and equation discovery.

Great thanks to Boris Kompare for providing crucial ecological background knowledge for Lake Glumsoe domain and to professor Sven-Erik Jørgensen for providing Lake Glumsoe data.

Thanks to Vito Robar and Anamarija Rožić-Hristovski for reading and debugging the manuscript.

A great amount of the work, presented in the thesis, was done at the Institute for Biomedical Informatics at Faculty of Medicine in Ljubljana. Thanks to the staff of the institute for providing stimulating research environment and tolerating all the network and computer facilities overloading and faults during my research work.

Bibliography

[Baghious and Jarry 1993]
Baghious, E. H. and Jarry, P. (1993) „Lorenz attractor“ from differential equations with piecewise-linear terms. *International Journal of Bifurcation and Chaos*, 3(1): 201–210.

[Bergadano and Gunetti 1995]
Bergadano, F. and Gunetti, D. (1995) *Inductive logic programming: from Machine Learning to Software Engineering.* MIT Press, Cambridge, MA.

[Bohte 1987]
Bohte, Z. (1987) *Numerical methods.* Cankarjeva založba, Ljubljana, Slovenia. In Slovene.

[Chomsky 1986]
Chomsky, N. *Knowledge of Language: Its Nature, Origin and Use.* Preager Publishers, New York, NY.

[Cohen 1992]
Cohen, W. (1992) Compiling prior knowledge into an explicit bias. In *Proc. Ninth International Conference on Machine Learning*, pages 102–110. Morgan Kaufmann, San Mateo, CA.

[Crispi and Mosetti 1993]
Crispi, G. and Mosseti, R. (1993) Adjoint estimation of aquatic ecosystem parameters. *Coenoses*, 8(1): 11–14.

[Dehaspe and DeRaedt 1995]
Dehaspe, L. and De Raedt, L. (1995) A declarative language bias for concept-learning algorithms. *The Knowledge Engineering Review*, 7(3): 251–269.

[Demšar 1996]
Demšar, D. (1996). Experiments in automated modeling of ecological processes in Lake Glumsoe. BSc Thesis, Faculty of Computer and Information Science, University of Ljubljana, Slovenia. In Slovene.

[Demšar et al. 1997]
Demšar, J., Zupan, B., Bohanec, M. and Bratko, I. (1997) Constructing intermediate

concepts by decomposition of real functions. In *Proc. Ninth European Conference on Machine Learning*. Springer, Berlin.

[Džeroski and Todorovski 1993]
Džeroski, S. and Todorovski, Lj. (1993) Discovering dynamics In *Proc. Tenth International Conference on Machine Learning*, pages 97–103. Morgan Kaufmann, San Mateo, CA.

[Geva and Sitte 1993]
Geva, S. and Sitte, J. (1993) A cartpole experimental benchmark for trainable controllers. *IEEE Control Systems*, 13(5): 40–51.

[Hopcroft and Ullman 1979]
Hopcroft J. E. and Ullman, J. D. (1979) *Introduction to automata theory, languages and computation.* Addison-Wesley, Reading, MA.

[Jørgensen et al. 1986]
Jørgensen, S., Kamp-Nielsen, L., Chirstensen, T., Windolf-Nielsen, J. and Westergaard, B. (1986) Validation of a prognosis based upon a eutrophication model. *Ecological modelling*, 32: 165–182.

[Karalič 1995]
Karalič, A. (1995) *First Order Regression.* PhD Thesis, Faculty of Computer and Information Science, University of Ljubljana, Slovenia.

[Karalič and Bratko 1997]
Karalič, A. and Bratko, I. (1997) First order regression. *Machine Learning*, 26: 147–176.

[Kompare 1995]
Kompare, B. (1995). The use of artificial intelligence in ecological modeling. PhD Thesis, Royal Danish School of Pharmacy, Copenhagen, Denmark.

[Križman et. al 1995]
Križman, V., Džeroski, S. and Kompare, B. (1995) Discovering dynamics from measured data. *Electrotechnical Review*, 62: 191–198.

[Langley et al. 1987]
Langley, P., Simon, H. and Bradshaw, G. (1987) Heuristics for empirical discovery. In Bolc, L., editor, *Computational Models of Learning.* Springer, Berlin.

[Mitchell 1980]
Mitchell, T. (1980) The need for biases in learning generalizations. Technical Report TR-117, Department of Computer Science, Rutgers University.

[Monod 1942]
Monod, J. (1942). *Recherches sur la croissance des cultures bacteriennes.* Hermann, Paris. In French.

[Morik et al. 1993]
Morik, K., Wrobel, S., Kietz, J.-U. and Emde, W. (1993) *Knowledge Acquisition and Machine Learning: Theory, Methods and Applications.* Academic Press.

[Nédellec et al. 1996]
Nédellec, C., Rouveirol, C., Adé, H., Bergadano, F. and Tausend, B. (1996) Declarative bias in ILP. In De Raedt, L., editor, *Advances in Inductive Logic Programming*, pages 82–103. IOS Press, Amsterdam.

[Press et al. 1986]
Press, W. H., Flannery, B. P., Teukolsky, S. A. and Vetterling, W. T. (1986) *Numerical Recipes.* Cambridge University Press, Cambridge, MA.

[Schaffer 1993]
Schaffer, C. (1993) Bivariate scientific function finding in a sampled, real-data testbed. *Machine Learning*, 12: 167–183.

[Todorovski 1993]
Todorovski, Lj. (1993) *Modeling of dynamic systems with equation discovery.* BSc Thesis, Faculty of Computer and Information Science, University of Ljubljana, Slovenia. In Slovene.

[Urbančič and Bratko 1994]
Urbančič, T. and Bratko, I. (1994) Learning to control dynamic systems. In Michie, D., Spiegelhalter, D. and Taylor, C., editors, *Machine Learning, Neural and Statistical Classification.* Ellis Horwood, Chichester.

[Whigham 1995]
Whigham, P. (1995) Grammatically-based genetic programming. In *Proc. ML'95 Workshop on Genetic Programming: From Theory to Real-World Applications*, pages 33–41. Lake Tahoe, CA.

[Zembowitz and Żytkow 1992]
Zembowitz, R. and Żytkow, J. (1992). Discovery of equations: experimental evaluation of convergence. In *Proc. Tenth National Conference on Artificial Intelligence.* Morgan Kaufmann, San Mateo, CA.

Appendix A

LAGRAMGE input files

A.1 Grammar files

$$
\begin{array}{lcl}
\mathit{GrammarFile} & \rightarrow & \mathit{FunctionDefinitions}\ \mathit{Productions} \\[1em]
\mathit{FunctionDefinitions} & \rightarrow & \text{“}\%\{\text{“}\ \mathbf{function-definitions-in-C}\ \text{“}\%\}\text{“} \mid \epsilon \\[1em]
\mathit{Productions} & \rightarrow & \mathit{Production}\ \mathit{Productions} \mid \mathit{Production} \\[1em]
\mathit{Production} & \rightarrow & \mathit{LeftSide}\ \text{“}\!\rightarrow\!\text{“}\ \mathit{RightSide}\,\text{“};\text{“} \\
\mathit{LeftSide} & \rightarrow & \mathit{Nonterminal} \\
\mathit{RightSide} & \rightarrow & \mathit{Symbols}\ \text{“}|\text{“}\ \mathit{RightSide} \mid \mathit{Symbols} \\[1em]
\mathit{Symbols} & \rightarrow & \mathit{Symbol}\ \mathit{Symbols} \mid \mathit{Symbol} \\
\mathit{Symbol} & \rightarrow & \mathit{NonTerminal} \mid \mathit{Terminal} \\[1em]
\mathit{NonTerminal} & \rightarrow & \mathit{CapitalLetter} \mid \mathit{String} \\
\mathit{Terminal} & \rightarrow & \mathit{String} \\[1em]
\mathit{String} & \rightarrow & \mathit{String}\ \mathit{AlphaNum} \mid \epsilon \\[1em]
\mathit{AlphaNum} & \rightarrow & \mathit{Letter} \mid \mathit{Digit} \mid \mathit{Misc} \\
\mathit{Letter} & \rightarrow & \text{“}\mathbf{a}\text{“} \mid \text{“}\mathbf{A}\text{“} \mid \text{“}\mathbf{b}\text{“} \mid \text{“}\mathbf{B}\text{“} \mid \ldots \mid \text{“}\mathbf{z}\text{“} \mid \text{“}\mathbf{Z}\text{“} \\
\mathit{CapitalLetter} & \rightarrow & \text{“}\mathbf{A}\text{“} \mid \text{“}\mathbf{B}\text{“} \mid \ldots \mid \text{“}\mathbf{Z}\text{“} \\
\mathit{Digit} & \rightarrow & \text{“}\mathbf{0}\text{“} \mid \text{“}\mathbf{1}\text{“} \mid \ldots \mid \text{“}\mathbf{9}\text{“} \\
\mathit{Misc} & \rightarrow & \text{“}+\text{“} \mid \text{“}-\text{“} \mid \ldots
\end{array}
$$

A.2 Configuration files

$$
\begin{array}{lcl}
\mathit{ConfigurationFile} & \rightarrow & \mathit{ParameterSettings} \\
\mathit{ParameterSettings} & \rightarrow & \mathit{ParameterSetting}\ \mathit{ParameterSetting} \mid \mathit{ParameterSetting} \\
\mathit{ParameterSetting} & \rightarrow & \mathbf{parameter - name} \text{ “:“ } \mathbf{parameter - value}
\end{array}
$$

Parameter name	Possible values
search_strategy	beam exhaustive
fitting_algorithm	amoeba lm
heuristic_function	sse mdl
stopping_criterion	none error_treshold time_limit
max_derivation_tree_depth	positive integer number
beam_width	positive integer number
error_treshold	positive real number
time_limit	positive real number

A.3 Domain files

$$
\begin{array}{lcl}
\mathit{DomainFile} & \rightarrow & \mathit{VarNames}\ \mathit{Measurements} \\
\\
\mathit{VarNames} & \rightarrow & \mathit{VarName}\ \mathit{VarNames} \mid \mathit{VarName} \\
\mathit{VarName} & \rightarrow & \mathit{Letter}\ \mathit{String} \\
\\
\mathit{Measurements} & \rightarrow & \mathit{Measurement} \text{ “;“ } \mathit{Measurements} \mid \mathit{Measurement} \text{ “;“} \\
\mathit{Measurement} & \rightarrow & \mathit{Measurement}\ \mathit{Number} \mid \mathit{Number} \\
\\
\mathit{String} & \rightarrow & \mathit{String}\ \mathit{AlphaNum} \mid \epsilon \\
\\
\mathit{Number} & \rightarrow & \mathit{Sign}\ \mathit{Digits} \mid \mathit{Sign}\ \mathit{Digits} \text{ “.“ } \mathit{Digits} \\
 & \mid & \mathit{Sign}\ \mathit{Digits} \text{ “.“ } \mathit{Digits} \text{ “}\mathbf{E}\text{“ } \mathit{Sign}\ \mathit{Digits} \\
\mathit{Digits} & \rightarrow & \mathit{Digits}\ \mathit{Digit} \mid \mathit{Digit} \\
\\
\mathit{AlphaNum} & \rightarrow & \mathit{Letter} \mid \mathit{Digit} \mid \text{“\_“} \\
\mathit{Letter} & \rightarrow & \text{“}\mathbf{a}\text{“} \mid \text{“}\mathbf{A}\text{“} \mid \text{“}\mathbf{b}\text{“} \mid \text{“}\mathbf{B}\text{“} \mid \ldots \mid \text{“}\mathbf{z}\text{“} \mid \text{“}\mathbf{Z}\text{“} \\
\mathit{Digit} & \rightarrow & \text{“}\mathbf{0}\text{“} \mid \text{“}\mathbf{1}\text{“} \mid \ldots \mid \text{“}\mathbf{9}\text{“} \\
\mathit{Sign} & \rightarrow & \text{“}+\text{“} \mid \text{“}-\text{“} \mid \epsilon
\end{array}
$$

Appendix B

Results of the experiments

In the appendix we give a tabular presentation of the outcomes for all the experiments in the synthetic domains from Chapter 4. The experiments were performed with exhaustive and beam search strategy with beam width set to 25, both algorithms for constant parameters' fitting, *SSE* and *MDL* heuristic function, and with and without threshold stopping criterion (threshold was set to 10^{-5}). Each cell contains the following information:

- the success of the experiment in sense that the discovered equation is:
  - OK - equivalent to the original was discovered
  - SIMPLER - similar to but simpler than the original
  - DIFFERENT - different from the original
- CPU time elapsed for the experiment
- number of expressions considered

B.1 Aquatic ecosystem

Stopping criterion	Heuristic function	downhill simplex		Levenberg-Marquardt	
		exhaustive	beam (25)	exhaustive	beam (25)
none	*SSE*	OK 4234.12/1831	OK 851.49/475	OK 11598.70/1831	OK 1475.84/375
	MDL	OK 4247.26/1831	OK 309.49/248	OK 11623.00/1831	OK 685.81/254
$T_{\mathcal{H}} = 10^{-5}$	*SSE*	OK 1.79/11	OK 3.31/20	OK 2.40/11	OK 2.29/13
	MDL	OK 4233.84/1831	OK 309.08/248	OK 11617.10/1831	OK 678.19/254

Results of experiments for equation $\dot{nut} = -phyt \cdot nut/(k_N + nut)$

Stopping criterion	Heuristic function	downhill simplex		Levenberg-Marquardt	
		exhaustive	beam (25)	exhaustive	beam (25)
none	*SSE*	OK 4401.17/1831	OK 913.40/532	OK 12150.90/1831	DIFFERENT 2410.66/581
	MDL	DIFFERENT 4410.95/1831	DIFFERENT 389.23/300	DIFFERENT 12124.70/1831	DIFFERENT 859.55/293
$T_{\mathcal{H}} = 10^{-5}$	*SSE*	SIMPLER 1216.79/746	SIMPLER 469.94/355	SIMPLER 2601.29/746	SIMPLER 1242.91/347
	MDL	DIFFERENT 4402.09/1831	DIFFERENT 386.10/300	DIFFERENT 12163.40/1831	DIFFERENT 858.26/293

Results of experiments for equation
$\dot{phyt} = phyt \cdot nut/(k_N + nut) - r_P \cdot phyt - zoo \cdot phyt/(k_P + phyt)$

Stopping criterion	Heuristic function	downhill simplex		Levenberg-Marquardt	
		exhaustive	beam (25)	exhaustive	beam (25)
none	*SSE*	DIFFERENT 5150.92/1831	OK 1167.32/518	DIFFERENT 8381.44/1831	DIFFERENT 854.96/456
	MDL	SIMPLER 5156.20/1831	SIMPLER 444.35/335	SIMPLER 8455.39/1831	SIMPLER 632.16/335
$T_{\mathcal{H}} = 10^{-5}$	*SSE*	DIFFERENT 2.28/13	SIMPLER 57.64/58	DIFFERENT 3.19/13	DIFFERENT 133.91/52
	MDL	DIFFERENT 2.30/13	SIMPLER 57.30/57	DIFFERENT 3.13/13	DIFFERENT 135.22/51

Results of experiments for equation $\dot{zoo} = zoo \cdot phyt/(k_P + phyt) - r_Z \cdot zoo$

Stopping criterion	Heuristic function	downhill simplex		Levenberg-Marquardt	
		exhaustive	beam (25)	exhaustive	beam (25)
none	*SSE*	DIFFERENT 5808.27/6336	OK 1331.57/1842	DIFFERENT 2348.17/6336	DIFFERENT 543.18/1840
	MDL	DIFFERENT 6117.25/6336	DIFFERENT 1267.23/1668	DIFFERENT 1927.04/6336	DIFFERENT 670.08/1668
$T_{\mathcal{H}} = 10^{-5}$	*SSE*	DIFFERENT 3688.67/3947	SIMPLER 1182.79/1694	DIFFERENT 1432.17/3947	DIFFERENT 577.92/1692
	MDL	DIFFERENT 4673.8/6336	DIFFERENT 1329.34/1668	DIFFERENT 1608.39/6336	DIFFERENT 580.62/1668

Summary of results for aquatic ecosystem domain

B.2 Piecewise-linear circuit

Stopping criterion	Heuristic function	downhill simplex		Levenberg-Marquardt	
		exhaustive	beam (25)	exhaustive	beam (25)
none	*SSE*	OK 1538.73/1900	OK 322.77/510	OK 991.19/1900	OK 224.47/513
	MDL	SIMPLER 3334.50/1900	SIMPLER 228.56/410	SIMPLER 1487.47/1900	SIMPLER 160.08/410
$T_{\mathcal{H}} = 10^{-5}$	*SSE*	OK 1536.57/1900	OK 326.27/510	OK 983.54/1900	OK 215.26/513
	MDL	SIMPLER 1555.96/1900	SIMPLER 123.72/410	SIMPLER 985.47/1900	SIMPLER 300.86/410

Results of experiments for equation $\dot{x} = \sigma(y - x)$

Stopping criterion	Heuristic function	downhill simplex		Levenberg-Marquardt	
		exhaustive	beam (25)	exhaustive	beam (25)
none	*SSE*	OK 1454.25/1900	OK 316.86/498	OK 991.90/1900	OK 237.66/498
	MDL	SIMPLER 2209.62/1900	SIMPLER 266.82/441	SIMPLER 1486.88/1900	SIMPLER 208.01/441
$T_{\mathcal{H}} = 10^{-5}$	*SSE*	OK 1463.91/1900	OK 335.65/498	OK 1213.47/1900	OK 220.40/498
	MDL	SIMPLER 1466.45/1900	SIMPLER 145.55/441	SIMPLER 983.47/1900	SIMPLER 193.50/441

Results of experiments for equation $\dot{y} = \mu_1(r - z) - qy$

Stopping criterion	Heuristic function	downhill simplex		Levenberg-Marquardt	
		exhaustive	beam (25)	exhaustive	beam (25)
none	*SSE*	OK 1902.08/1900	OK 557.37/603	OK 992.04/1900	OK 302.17/595
	MDL	SIMPLER 2851.97/1900	SIMPLER 326.95/473	SIMPLER 1912.57/1900	SIMPLER 201.35/472
$T_{\mathcal{H}} = 10^{-5}$	*SSE*	OK 1894.55/1900	OK 566.89/603	OK 1062.09/1900	OK 293.45/595
	MDL	SIMPLER 1889.38/1900	SIMPLER 198.95/473	SIMPLER 1041.59/1900	SIMPLER 200.97/472

Results of experiments for equation $\dot{z} = \mu_2 x - bz$

Stopping criterion	Heuristic function	downhill simplex		Levenberg-Marquardt	
		exhaustive	beam (25)	exhaustive	beam (25)
none	*SSE*	OK 4895.06/5700	OK 1197.00/1611	OK 2975.13/5700	OK 764.30/1606
	MDL	SIMPLER 8396.09/5700	SIMPLER 822.33/1324	SIMPLER 4886.92/5700	SIMPLER 569.44/1323
$T_{\mathcal{H}} = 10^{-5}$	*SSE*	OK 4895.03/5700	OK 1228.81/1611	OK 3259.10/5700	OK 729.11/1606
	MDL	SIMPLER 4911.79/5700	SIMPLER 468.22/1324	SIMPLER 3010.53/5700	SIMPLER 695.33/1323

Summary of results for piecewise-linear circuit domain

B.3 Two poles on cart

Stopping criterion	Heuristic function	downhill simplex		Levenberg-Marquardt	
		exhaustive	beam (25)	exhaustive	beam (25)
none	*SSE*	OK 1692.47/2112	OK 317.68/612	OK 803.48/2112	OK 129.7/612
	MDL	DIFFERENT 1899.39/2112	DIFFERENT 434.42/569	DIFFERENT 826.64/2112	DIFFERENT 208.97/569
$T_{\mathcal{H}} = 10^{-5}$	*SSE*	OK 662.83/867	OK 237.51/503	OK 353.66/867	OK 98.43/503
	MDL	DIFFERENT 1746.22/2112	DIFFERENT 448.47/569	DIFFERENT 634.67/2112	DIFFERENT 236.27/569

Results of experiments for equation

$$2(M + m_1 + m_2)\ \ddot{x} + l_1(m_1 + 2m_2)\ \ddot{\varphi}\cos\varphi + m_2 l_2\ \ddot{\vartheta}\cos\vartheta = 2F + (m_1 + 2m_2)l_1\ \dot{\varphi}^2\sin\varphi + m_2 l_2\ \dot{\vartheta}^2\sin\vartheta$$

Stopping criterion	Heuristic function	downhill simplex		Levenberg-Marquardt	
		exhaustive	beam (25)	exhaustive	beam (25)
none	*SSE*	OK 1907.85/2112	OK 530.15/619	OK 838.54/2112	OK 270.14/619
	MDL	SIMPLER 2183.68/2112	SIMPLER 506.79/528	SIMPLER 602.07/2112	SIMPLER 233.76/528
$T_{\mathcal{H}} = 10^{-5}$	*SSE*	OK 917.00/968	OK 407.07/580	OK 305.98/968	OK 219.47/580
	MDL	SIMPLER 1490.36/2112	SIMPLER 310.53/528	SIMPLER 490.91/2112	SIMPLER 178.46/528

Results of experiments for equation

$$(m_1 + 2m_2)\ \ddot{x}\cos\varphi + 2l_1(m_1/3 + m_2)\ \ddot{\varphi} + m_2 l_2\ \ddot{\vartheta}\cos(\vartheta - \varphi) = m_2 l_2\ \dot{\vartheta}^2\sin(\vartheta - \varphi) + (m_1 + 2m_2)g\ \sin\varphi$$

Stopping criterion	Heuristic function	downhill simplex		Levenberg-Marquardt	
		exhaustive	beam (25)	exhaustive	beam (25)
none	*SSE*	OK 2207.95/2112	OK 483.74/611	OK 706.15/2112	OK 143.34/609
	MDL	SIMPLER 2034.18/2112	SIMPLER 326.02/571	SIMPLER 498.33/2112	SIMPLER 227.35/571
$T_{\mathcal{H}} = 10^{-5}$	*SSE*	OK 2108.84/2112	OK 538.21/611	OK 772.53/2112	OK 260.02/609
	MDL	SIMPLER 1437.22/2112	SIMPLER 570.34/571	SIMPLER 482.81/2112	SIMPLER 165.89/571

Results of experiments for equation
$\ddot{x}\cos\vartheta + l_1\ \ddot{\varphi}\cos(\vartheta-\varphi) + 2/3\ \ddot{\vartheta} = g\ \sin\vartheta - l_1\ \dot{\varphi}^2\sin(\vartheta-\varphi)$

Stopping criterion	Heuristic function	downhill simplex		Levenberg-Marquardt	
		exhaustive	beam (25)	exhaustive	beam (25)
none	*SSE*	OK 5808.27/6336	OK 1331.57/1842	OK 2348.17/6336	OK 543.18/1840
	MDL	DIFFERENT 6117.25/6336	DIFFERENT 1267.23/1668	DIFFERENT 1927.04/6336	DIFFERENT 670.08/1668
$T_{\mathcal{H}} = 10^{-5}$	*SSE*	OK 3688.67/3947	OK 1182.79/1694	OK 1432.17/3947	OK 577.92/1692
	MDL	DIFFERENT 4673.80/6336	DIFFERENT 1329.34/1668	DIFFERENT 1608.39/6336	DIFFERENT 580.62/1668

Summary of results for two poles on cart domain

Appendix C

Omejevanje prostora hipotez pri odkrivanju enačb

C.1 Uvod

Vsi programi za avtomatsko učenje uporabljajo neko omejitev prostora hipotez. Večina teh programov ima vnaprej določen prostor hipotez, na katerega ima uporabnik zelo malo (če sploh kaj) vpliva. V zadnjem času se je pojavila tendenca v avtomatskem učenju, da postaja prostor hipotez vhodni (uporabniško določen) parameter učnega sistema. Tendenca je še najbolj opazna v področjih avtomatskega učenja, ki imajo opravka z zelo velikimi prostori hipotez, kot je recimo induktivno logično programiranje (prostor hipotez je v tem primeru podmnožica stavkov v logiki prvega reda). Veliko programov za induktivno logično programiranje dovoli uporabniku določanje in omejevnje prostora hipotez (glej pregled v [Nédellec et al. 1996]).

Prostor hipotez v programih za avtomatsko odkrivanje enačb je množica vseh enačb, ki jih lahko sestavimo iz določenega nabora aritmetičnih operatorjev, funkcij in spremenljivk. Množica teh enačb je neskončna. Iz teorije približkov je znano, da obstaja neskončno število polinomskih krivulj (odsekoma polinomske krivulje, zlepki), ki so lahko dober približek poljubnim podanim podatkom. Programi za avtomatsko odkrivanje enčb morajo omejiti prostor možnih enačb, da bi preprečili zmanjševanje natančnosti odkritih enačb na novih podatkih, ki niso na voljo v fazi učenja. Po drugi strani, mora biti prostor hipotez dovolj velik, da zajame enačbe, ki dajo dovolj dober približek podanim podatkom.

Obstoječi programi za odkrivanje enačb imajo vnaprej opisan prostor enačb. Uporabnik lahko preko določenih parametrov vpliva na kompleksnost odkritih enačb, ali pa poda množico operatorjev in funkcij iz katerih so sestavljene enačbe (glej [Zembowitz and Żytkow 1992, Todorovski 1993, Džeroski and Todorovski 1993]).

V magistrskem delu smo razvili formalizem za opis prostora hipotez pri odkrivanju enačb. Formalizem temelji na kontekstno neodvisnih gramatikah in omogoča opis oblike enačb, ki naj bi jih program za odkrivanje enačb preiskal. V gramatikah poleg običajnih matematičnih operatorjev lahko uporabimo tudi funkcije, definirane na osnovi predznanja o posameznem problemu. Formalizem smo implementirali v programu za avtomatsko odkrivanje enačb LAGRAMGE, ki hevristično preiskuje prostor vseh možnih enačb, katerih oblika je podana s kontekstno neodvisno gramatiko.

Uporabnost programa LAGRAMGE smo prikazali na problemih odkrivanja enačb v umetnih in realnih domenah. Med drugim je LAGRAMGE uspešno rekonstruiral model invertiranega nihala sestavljenega iz vozička in dveh palic iz podatkov dobljenih z numerično simulacijo originalnega modela. Program LAGRAMGE smo tudi uporabili za avtomatsko modeliranje rasti rastlinskega planktona v jezeru Glumsoe na Danskem. Model, ki ga je LAGRAMGE odkril, natančno napoveduje koncentracijo rastlinskega planktona v jezeru, tako za krajša, kot za daljša časovna obdobja.

C.2 Formalizem za omejevanje prostora enačb

Nalogo odkrivanja enačb z omejevanjem prostora hipotez, kot jo obravnavamo v magistrskem delu, lahko definiramo z naslednjim predpisom:
Podano imamo:

- kontekstno neodvisno gramatiko $G = (N, T, P, S)$ in
- vhodne podatke $D = (V, v_d, M)$, pri čemer je V množica merjenih spremenljivk, $v_d \in V$ odvisna spremenljivka in M množica ene ali več tabel z meritvami spremenljivk iz V v nekem končnem časovnem intervalu.

Poiskati moramo enačbo, ki določa odvisnost spremenljivke v_d od vrednosti spremenljivk iz V. Enačba je lahko:

- diferencialna, oblike $\partial v_d / \partial t = \dot{v_d} = E$, ali
- navadna, oblike $v_d = E$,

pri čemer je E izraz, ki ga lahko izpeljemo iz kontekstno neodvisne gramatike $G = (N, T, P, S)$ [Hopcroft and Ullman 1979]. N in T sta neprazni disjunktni množici nekončnih in končnih simbolov. P je množica produkcij in S poseben nekončni simbol, ki ga imenujemo začetni simbol. Produkcije iz P so oblike $A \rightarrow \alpha$, pri čemer je $A \in N$ leva stran in $\alpha \in (N \cup T)^*$ desna stran produkcije.

Gramatika, ki jo uporabljamo za omejevanje prostora hipotez pri avtomatskem odkrivanju enačb, ima več simbolov s posebnim pomenom. Končni simbol $const \in T$ uporabljamo za označevanje konstantnega parametra v enačbi, katerega vrednost se prilagodi vhodnim podatkom o meritvi spremenljivk sistema. Končni simboli v_i označujejo posamezne spremenljivke iz V. Nekončni simbol $v \in N$ uporabljamo za označevanje poljubne spremenljivke obravnavanega sistema. Produkcije, ki povezujejo v s končnimi simboli v_i, se dodajo gramatiki G avtomatsko, t.j. $\forall v_i \in V : v \rightarrow v_i \in P$.

Gramatika G mora izpolnjevati en pogoj. Vse desne strani produkcij v P morajo biti sintaktično pravilni izrazi v programskem jeziku C. To pomeni, da lahko uporabljamo vse aritmetične operatorje in funkcije vgrajene v programskem jeziku C, opisu gramatike pa lahko dodamo še funkcije, ki jih definiramo na osnovi teoretičnega predznanja ekspertov iz področja uporabe.

Izraze generiramo iz gramatike G s pomočjo izpeljevalnih dreves z algoritmom, ki je podan v tabeli 2.1. Izpeljevalno drevo generira izraz takrat, ko so vsi njegovi listi označeni s končnimi simboli. Če preberemo oznake listov z leve proti desni, dobimo izraz, ki ga izpeljevalno drevo generira. Na koncu, definiramo še višino izpeljevalnega drevesa kot maksimalno dolžino poti iz korenskega vozlišča do enega izmed listov.

C.3 Lagramge

Izrazi, ki jih izpeljemo s pomočjo kontekstno neodvisne gramatike G vsebujejo enega ali več končnih simbolov *const*. Program Lagrmage uporablja nelinearno metodo za določanje vrednosti teh parametrov. Če je $\mathbf{c}$ vektor konstantnih parametrov v izrazu E, potem je rezultat nelinearne metode za določanje vrednosti parametrov vektor $\mathbf{c}^*$, za katerega velja $Error(\mathbf{c}^*) = \min_{\mathbf{c}\in R^{n_c}}\{Error(\mathbf{c})\}$. Funkcija napake $Error(\mathbf{c})$ je vsota kvadratov napak, definirana na naslednji način:

- za diferencialno enačbo oblike $\partial v_d/\partial t = \dot{v_d} = E$:

  $Error(\mathbf{c}) = \sum_{i=0}^{m}\left[v_{d,i} - \left(v_{d,0} + \int_{t_0}^{t_i} E(\mathbf{c}, v_1, \ldots v_n)\right)\right]^2$, and

- za običajno enačbo oblike $v_d = E$:

  $Error(\mathbf{c}) = \sum_{i=0}^{m}(v_{d,i} - E(\mathbf{c}, v_{1,i}, \ldots v_{d-1,i}, v_{d+1,i}, \ldots v_{n,i}))^2$,

pri čemer je m dolžina tabele meritev in $v_{j,i}$ vrednost spremenljivke v_j v času t_i. V primeru računanja funkcije napake za diferencialno enačbo uporabljamo integral izraza na desni strani enačbe namesto odvoda odvisne spremenljivke, zato ker je napaka pri numerični integraciji v splošnem manjša od napake, ki nastane pri numeričnem odvajanju. Za izračun integrala uporabljamo trapezno formulo za numerično integracijo. Kot nelinearne metode za določanje vrednosti konstantnih parametrov uporabljamo lahko algoritem spusta v dolino (*downhill simplex*) ali Levenberg-Marquardt metodo [Press et al. 1986].

V naslednjem koraku program Lagramge izračuna vrednost hevristične funkcije SSE za izraz E, ki je enaka vrednosti funkcije napake ($SSE(E) = Error(\mathbf{c}^*)$). Program Lagramge lahko uporablja tudi hevristično funkcijo MDL, ki upošteva tudi dolžino l izraza E:

$$MDL(E) = SSE(E) + \frac{l}{10 \cdot l_{max}}\sigma_{v_d},$$

pri čemer je l_{max} dolžina najdaljšega izraza, ki ga lahko izpeljemo iz gramatike G z izpeljevalnim drevesom višine največ h_{max}. Dolžina je enaka številu končnih simbolov v izrazu. MDL funkcija daje prednost krajšim enačbam pred daljšimi.

Program LAGRAMGE izčrpno ali hevristično (uporabljamo iskanje s snopom z nastavljivo širino snopa) poišče enačbo, ki ima najmanjšo vrednost izbrane hevristične funkcije v prostoru enačb, ki je določen s kontekstno neodvisno gramatiko G in maksimalno višino izpeljevalnih dreves h_{max}.

C.4 Poskusi s programom LAGRAMGE

Program LAGRAMGE smo preizkusili na problemih odkrivanja enačb v treh sintetičnih in dveh realnih domenah. Probleme smo izbrali tako, da prikažemo različne možnosti uporabe programa LAGRAMGE. V vseh domenah smo iskali diferencialne enačbe, ki vsebujejo nelinearnosti glede konstantnih parametrov in sistemskih spremenljivk.

V sintetičnih domenah smo vhodne podatke dobili s simulacijo originalnih enačb z desetimi različnimi, naključno izbranimi začetnimi stanji. Podatke za realne domene smo dobili iz meritev realnega sistema ali pa njegovega laboratorijskega modela. V dodatku B smo podali rezultate poskusov v sintetičnih domenah.

C.4.1 Enostaven ekosistem

Enostaven ekosistem opisuje evolucijo koncentracij hrane, rastlinskega in živalskega planktona v vodnem okolju. V poskusih s programom LAGRAMGE smo uporabili gramatiko iz tabele 2.2.

LAGRAMGE je odkril enačbe populacijske dinamike za enostavni ekosistem v njihovi originalni obliki, ki je razumljiva ekspertom iz področja ekologije. LAGRAMGE je potreboval 50 minut procesorskega časa in med iskanjem obravnaval 1525 različnih izrazov. Programa za avtomatsko učenje LAGRANGE in GOLDHORN sta tudi uspešno rekonstruirala iste enačbe, ampak v nerazumljivi obliki polinomov več spremenljivk.

C.4.2 Odsekoma-linearno vezje

Enostavno električno vezje [Baghious and Jarry 1993] s kaotičnim atraktorjem, ki je podoben atraktorju Lorenzovega sistema enačb lahko opišemo s tremi odsekoma-linearnimi diferencialnimi enačbami. V poskusih s programom LAGRAMGE smo uporabili gramatiko iz tabele 4.1, ki ilustrira možnost uporabe logičnih izrazov v enačbah.

LAGRAMGE je uspešno rekonstruiral enačbe električnega vezja v 12 minutah procesorskega časa in med tem obravnaval 1606 različnih izrazov.

C.4.3 Dve palici na vozičku

Invertirano nihalo je setavljeno iz vozička, ki se lahko giblje po vodoravnih tračnicah. Na vozičku je vrtljivo vpeta palica, ki je gibljiva v vertikalni ravnini, ki poteka skozi tirnice. S silo, ki je vzporedna s tračnicami, delujemo na voziček tako, da je ta čim bližje izhodiščni točki na tračnicah in palica v navpičnem položaju. Model invertiranega nihala je najbolj kompleksen model, ki ga je sistem za avtomatsko odkrivanje enačb LAGRANGE uspešno rekonstruiral [Todorovski 1993]. Sistem dveh palic na vozičku je invertirano nihalo z dvema palicama: druga palica je vrtljivo vpeta na vrhu prve. Zgornja palica lahko rotira v vertikalni ravnini, ki gre skozi tirnice, okrog točke v kateri je vpeta na spodnjo palico.

Rekonstrukcija enačb invertiranega nihala je zunaj dosega programov LAGRANGE in GOLDHORN. Prostor hipotez za ta dva programa zajame 185.111.450.355 enačb. V tabeli 4.2 je podana gramatika, ki smo jo uporabili za omejevanje prostora enačb pri modeliranju invertiranega nihala, in sicer za prvo enačbo. Čeprav je gramatika na prvi pogled zelo zapletena, jo lahko enostavno zgradimo na osnovi elementarnega znanja Lagrangeove metode za modeliranje dinamike mehanskih sistemov. V vseh poskusih z uporabo različnih metod za določanje vrednosti konstantnih parametrov, različnih hevrističnih metod in strategij iskanja, je LAGRAMGE uspešno rekonstruiral vse tri enačbe, ki opisujejo obnašanje invertiranega nihala. Najkrajši poskus je trajal približno 9 minut procesorskega časa. LAGRAMGE je med iskanjem najboljših enačb obravnaval 1840 različnih izrazov.

C.4.4 Vodostan

Domena vodostan zajema meritve realnega laboratorijskega modela vodostana, ki je sestavljen iz dveh posod povezanih z dolgo cevjo. Prva posoda je večja in je namenjena modeliranju vodnega zajetja. Druga posoda je izravnalna posoda, je manša in ima vgrajen odtočni ventil. Modelirali smo nihanje vode v manjši izravnalni posodi modela vodostana.

V poskusih s programom LAGRAMGE smo uporabili gramatiko, ki posnema obliko enačbe teoretično pridobljenega modela. Program LAGRAMGE je odkril enačbo, ki smo jo primerjali z enačbami pridobljenimi z uporabo programov GOLDHORN in FORS [Karalič 1995]. Rezultati primerjave so prikazani v grafu na sliki 4.4.

C.4.5 Jezero Glumsoe

V tej domeni smo obravnavali problem modeliranja rasti rastlinskega planktona v danskem jezeru Glumsoe [Jørgensen et al. 1986]. Na voljo smo imeli 14 meritev sistemskih spremenljivk koncentracije rastlinskega planktona (*phyt*), živalskega planktona (*zoo*), raztopljenega dušika (*nitro*), fosforja (*phosp*) in temperature vode (*temp*). Pomankanje intenzivnih meritev smo nadomestili z uporabo predznanja ekspertov iz področja ekologije. Najprej so trije eksperti točkovne grafe štirinajstih meritev povezali s krivuljami, ki po njihovem mnenju opisujejo obnašanje spremenljivk sistema med meritvenimi točkami. Tako dobljene podatke skupaj z gramatiko, definirano na osnovi teoretičnega predznanja iz populacijske dinamike, smo uporabili za odkrivanje enačb s programom LAGRAMGE. LAGRAMGE je odkril enačbo, ki je bila razumljiva ekspertom in je hkrati uporabna za dolgoročno napovedovanje koncentracije rastlinskega planktona v jezeru (glej graf na sliki 4.8).

C.5 Zaključki in nadaljnje delo

V magistrskem delu smo razvili program za avtomatsko odkrivanje enačb Lagramge, ki uporablja kontekstno neodvisne gramatike za omejevanje prostora hipotez. Kontekstno neodvisno gramatiko skupaj z uporabniško definiranimi funkcijami zgradimo na osnovi teoretičnega predznanja iz področja uporabe. Prostor hipotez, ki ga preiskuje program Lagramge, so enačbe, katerih desne strani lahko izpeljemo s pomočjo podane gramatike.

Glavna prednost uporabe formalizma za omejevanje prostora enačb je ta, da omogočimo uporabniku prilagajanje programa Lagramge področju uporabe. Obstoječi programi za avtomatsko odkrivanje enačb imajo nespremenljiv (relativno majhen t.j. obvladljiv) prostor enačb, kar lahko zmanjša njihovo uporabnost iz dveh razlogov. Prvi je ta, da je lahko enačba, ki pravilno opisuje sistem, zunaj vnaprej določenega prostora enačb. Drugi razlog je ta, da sistem sicer odkrije pravilno enačbo, ta pa je zapisana v obliki, ki ni razumljiva ekspertu iz področja uporabe. Krčenje prostora enačb lahko po drugi strani zmanša natančnost modelov, ki jih odkrije Lagrmage. Zato v primerih, ko je natančnost modela pomembna, lahko uporabimo gramatiko, ki zajame večji prostor možnih enačb (recimo polinome večih spremenljivk, tako kot Lagrange [Todorovski 1993] in GoldHorn [Križman et. al 1995]). Druga pomembna prednost omejevanja prostora hipotez je možnost zmanjševanja kompleksnosti prostora hipotez, kar omogoča programu Lagrange odkrivanje modelov, ki so zaradi kompleksnosti bili zunaj dosega obstoječih programov za odkrivanje enačb. Nenazadnje, omejevanje prostora hipotez preprečuje *overfitting* modela na učni množici in s tem tudi zmanševanje natančnosti modela na podatkih, ki niso bili na voljo v učnem procesu.

Lagramge lahko preiskuje prostor enačb definiran z gramatiko z izčrpnim ali hevrističnem iskanjem. Pri hevrističnem iskanju uporabljamo strategijo iskanja s snopom nastavljive širine. Lagramge lahko uporablja dve različni metodi za določanje vrednosti konstantnih parametrov in dve hevristični funkciji za ocenjevanje enačb. Lahko odkriva navadne in diferencialne enačbe iz ene ali več tabel z meritvami sistemskih spremenljivk.

Program Lagramge smo uporabili za odkrivanje enačb v štirih umetnih in dveh realnih domenah.

Formalizem za omejevanje prostora enačb bi lahko nadgradili s kosnstrukti, ki omogočajo omejevanje možnih vrednosti konstantnih parametrov, ki nastopajo v enačbah. Druga možna smernica za nadaljnje delo bi lahko bila izdelava knjižnic kontekstno neodvisnih gramatik za različna področja uporabe programa Lagramge.

Izjava

Izjavljam, da sem magistrsko delo samostojno izdelal pod vodstvom mentorja prof. dr. Ivana Bratka. Izkazano pomoč drugih sodelavcev sem v celoti navedel v zahvali.

Ljupčo Todorovski