Computational Discovery of Scientific Models: Guiding Search with Knowledge and Data

Pat Langley

Department of Computer Science
University of Auckland

Silicon Valley Campus
Carnegie Mellon University

Thanks to K. Arrigo, G. Bradshaw, S. Borrett, W. Bridewell, S. Dzeroski, H. Simon, L. Todorovski, and J. Zytkow for their contributions to this research, which was partly funded by NSF Grant No. IIS-0326059 and ONR Grant No. N00014-11-1-0107.

The Scientific Enterprise

Science is a unique collection of activities distinguished by some important characteristics:

- Systematic collection and analysis of *observations*
- Formal statement of *theories*, *laws*, and *models*
- Use of the latter to *explain* and *predict* the former
- Use of observations to *evaluate* theorized structures

Moreover, science can apply these ideas to *any* area of enquiry, in principle, even, to *science itself*.

Philosophy of Science

One discipline – *philosophy of science* – has studied science itself since the 19th Century, including the:

- character of scientific observations and experiments
- structure of scientific theories, laws, and models
- nature of scientific explanations and predictions
- evaluation of scientific theories, models, and laws

However, philosophers of science have typically avoided one important topic: *scientific discovery*.

Mystical Views of Scientific Discovery

Philosophers largely ignored scientific discovery, believing it to be immune to logical analysis. Popper (1934) wrote:

> *The initial stage, the act of conceiving or inventing a theory, seems to me neither to call for logical analysis nor to be susceptible of it … My view may be expressed by saying that every discovery contains an 'irrational element', or 'a creative intuition'…*

He was not alone in this view. Hempel and many others believed discovery was inherently irrational and beyond understanding.

However, advances made by two fields – *cognitive psychology* and *artificial intelligence* – in the 1950s suggested otherwise.

Scientific Discovery as Problem Solving

Simon (1966) offered another view – that scientific discovery is a variety of *problem solving* that involves:

- *Search* through a space of connected *problem states*
- Generated from earlier states by mental *operators*
- Guided by *heuristics* that keep the search tractable

Heuristic search had been implicated in many cases of human problem solving, such as proving theorems and playing chess.

This idea offered a powerful new approach to understanding the rational character of scientific discovery.

But it also suggested ways to *automate* this mysterious process.

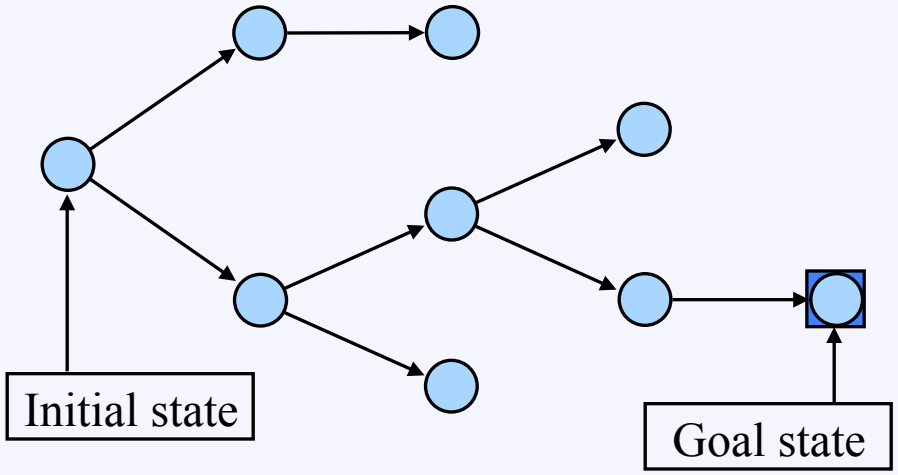
Heuristic Search in a Problem Space

Heuristic search is analogous to the traversal of a physical maze.

States in the problem space map onto locations in the maze.

Operators for producing new states map onto steps through the maze.

Solutions correspond to paths from the maze entrance to its exit (goal).



The initial state and the operators *implicitly* define a problem space.

Heuristics aid search by favoring likely choices and rejecting others to make solution finding tractable.

An Early Response

For my CMU dissertation research, I adapted Simon's ideas on scientific discovery, developing a computer program that:

- Carried out search in a problem space of theoretical terms;
- Using operators that combined old terms into new ones;
- Guided by heuristics that noted regularities in data; and
- Applied these recursively to formulate higher-level relations.

The result was *Bacon* (Langley, 1981), an early AI system that rediscovered laws from the history of physics and chemistry.

I named the system after Sir Francis Bacon because it adopted a data-driven approach to discovery.

Bacon on Kepler's Third Law

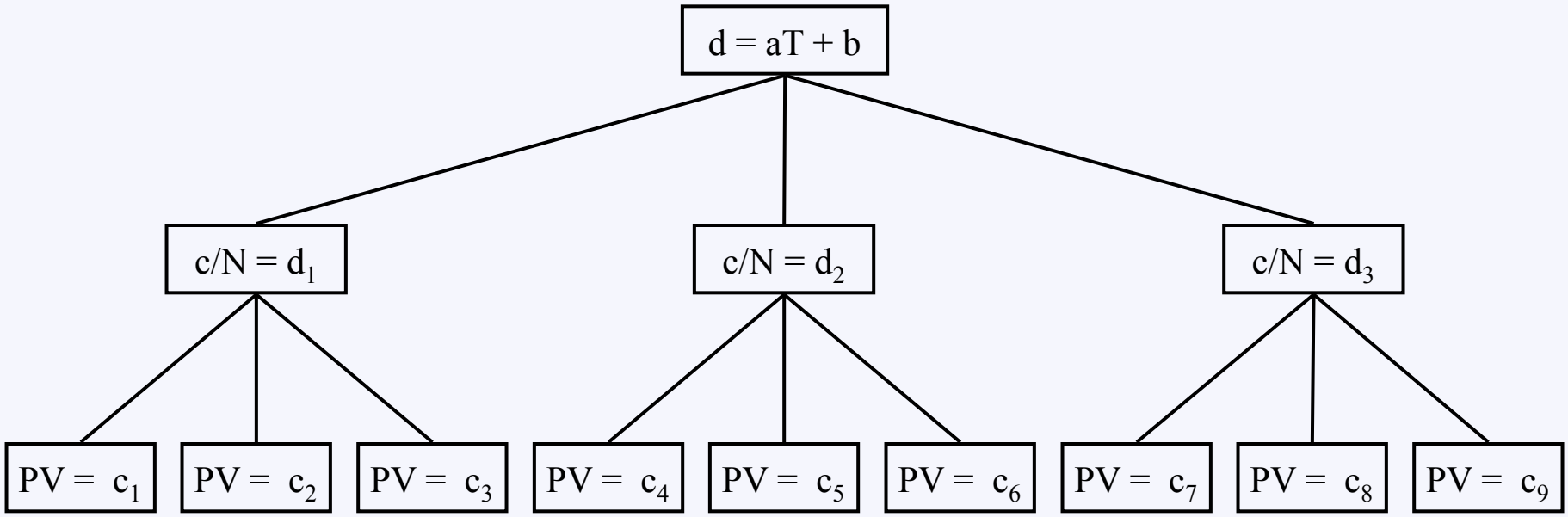
The Bacon system carried out heuristic search, through a space of numeric terms, looking for constants and linear relations.

moon	d	p	d/p	d^2/p	d^3/p^2
A	5.67	1.77	3.20	18.15	58.15
B	8.67	3.57	2.43	21.04	51.06
C	14.00	7.16	1.96	27.40	53.61
D	24.67	16.69	1.48	36.46	53.89

This table shows its progression from the distance and period of Jupiter's moons to a term with nearly constant value.

Bacon on the Ideal Gas Law

Bacon rediscovered the ideal gas law, PV = aNT + bN, in three stages, each at a different level of description.



Parameters for laws at one level became dependent variables in laws at the next level, enabling discovery of complex relations.

Some Laws Discovered by Bacon

Basic algebraic relations:

- Ideal gas law $PV = aNT + bN$
- Kepler's third law $D^3 = [(A - k) / t]^2 = j$
- Coulomb's law $FD^2 / Q_1Q_2 = c$
- Ohm's law $TD^2 / (LI - rI) = r$

Relations with *intrinsic properties*:

- Snell's law of refraction $\sin I / \sin R = n_1 / n_2$
- Archimedes' law $C = V + i$
- Momentum conservation $m_1V_1 = m_2V_2$
- Black's specific heat law $c_1m_1T_1 + c_2m_2T_2 = (c_1m_1 + c_2m_2)\,T_f$

Initial Responses to Bacon

Responses to the Bacon work were mixed, with some agreeing it clarified important aspects of scientific discovery.

But others claimed that *the real* key to discovery, which Bacon did not address, instead lay in:

- Deciding which variables to measure and relate
- Determining which problem space to search
- Selecting which scientific problem to address

Others held that Bacon only did what it was programmed to do, and thus did not really 'discover' anything.

We only claimed the system offered insights into the operation of scientific discovery, with much remaining to be done.

Ensuing Systems for Law Discovery

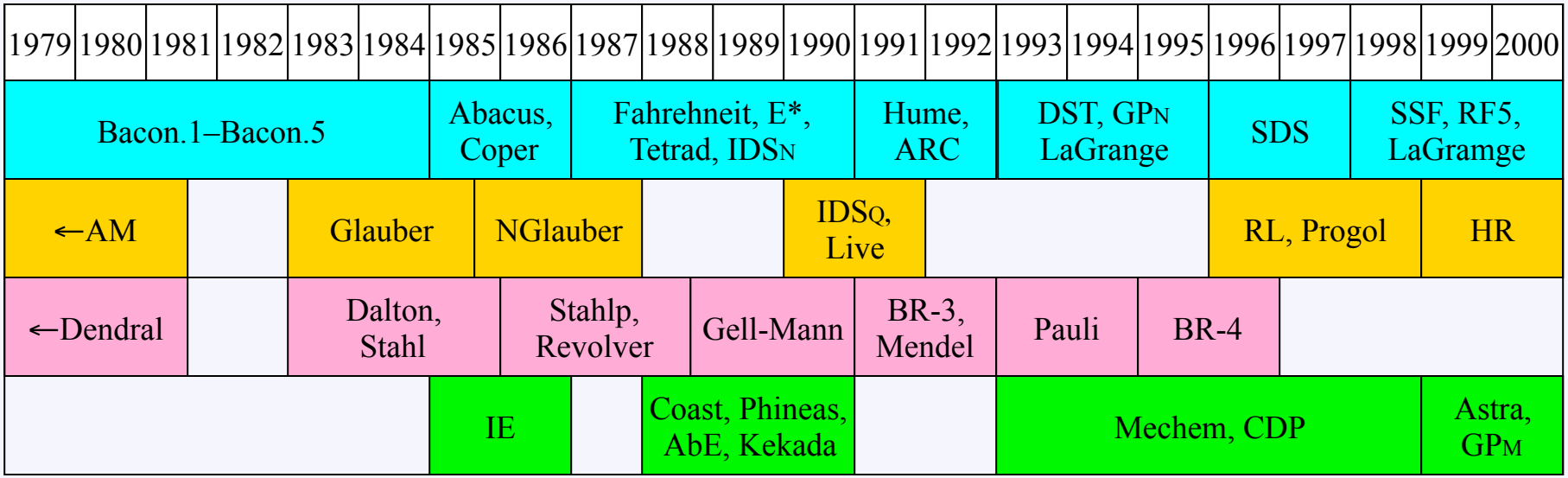
Indeed, Bacon inspired other AI systems for law discovery like:

- ABACUS (Falkenhainer, 1985) and ARC (Moulet, 1992)
- Fahrenheit (Zytkow, Zhu, & Hussam, 1990)
- COPER (Kokar, 1986) and E* (Schaffer, 1990)
- IDS (Nordhausen & Langley, 1990)
- Hume (Gordon & Sleeman, 1992)
- DST (Murata, Mizutani, & Shimura, 1994)
- SSF (Washio et al., 1997) and LaGramge (Todorovski et al., 2006)
- GP (Koza et al., 2001) and Eureqa (Schmidt & Lipson, 2009)

These relied on different methods but also searched for explicit mathematical laws that matched data.

Other Research on Discovery (from 1979 to 2000)

Interest in computational discovery spread to other aspects of science, including qualitative laws and explanatory models.



Research in this tradition has continued to the present, in some cases producing new scientific results.

Successes of Computational Scientific Discovery

AI systems of this type have helped to discover new knowledge in many scientific fields:

- reaction pathways in catalytic chemistry (Valdes-Perez, 1994, 1997)
- qualitative chemical factors in mutagenesis (King et al., 1996)
- quantitative laws of metallic behavior (Sleeman et al., 1997)
- quantitative conjectures in graph theory (Fajtlowicz et al., 1988)
- qualitative conjectures in number theory (Colton et al., 2000)
- temporal laws of ecological behavior (Todorovski et al., 2000)
- models of gene-influenced metabolism in yeast (King et al., 2009)

Each of these has led to publications in the *refereed literature of the relevant scientific field*.

The Data Mining Movement

During the 1990s, a new paradigm known as *data mining and knowledge discovery* emerged that:

- Emphasized the availability of large amounts of data;
- Used computational methods to find regularities in the data;
- Adopted heuristic search through a space of hypotheses;
- Initially focused on commercial applications and data sets.

Most work used notations invented by computer scientists, unlike work on scientific discovery, which used *scientific formalisms*.

Data mining has been applied to scientific data, but the results seldom bear a resemblance to scientific *knowledge*.

Discovering Explanatory Models

The early stages of any science focus on *descriptive laws* that *summarize* empirical regularities.

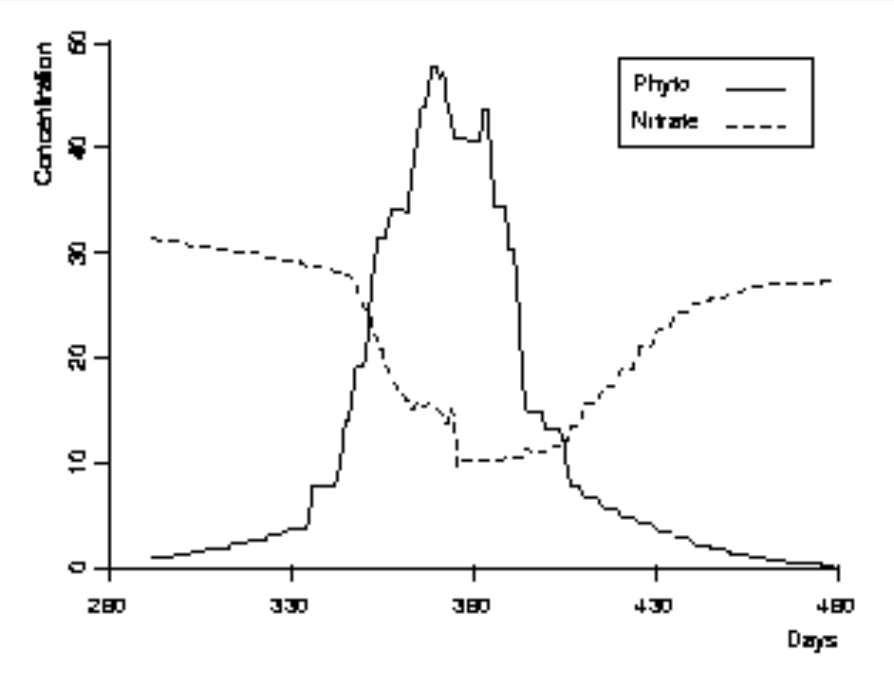
Mature sciences instead emphasize the creation of *models* that *explain* phenomena in terms of:

- Inferred *components* and *structures* of entities
- Hypothesized *processes* about entities' interactions

Explanatory models move beyond description to provide deeper accounts linked to theoretical constructs.

Can we develop computational systems that address this more sophisticated side of scientific discovery?

An Example: The Ross Sea Ecosystem



Formal accounts of ecosystem dynamics are often cast as sets of differential equations.

Here four equations describe the concentrations of phytoplankton, zooplankton, nitrogen, and detritus in the Ross Sea over time.

Such models can match observed variables with some accuracy.

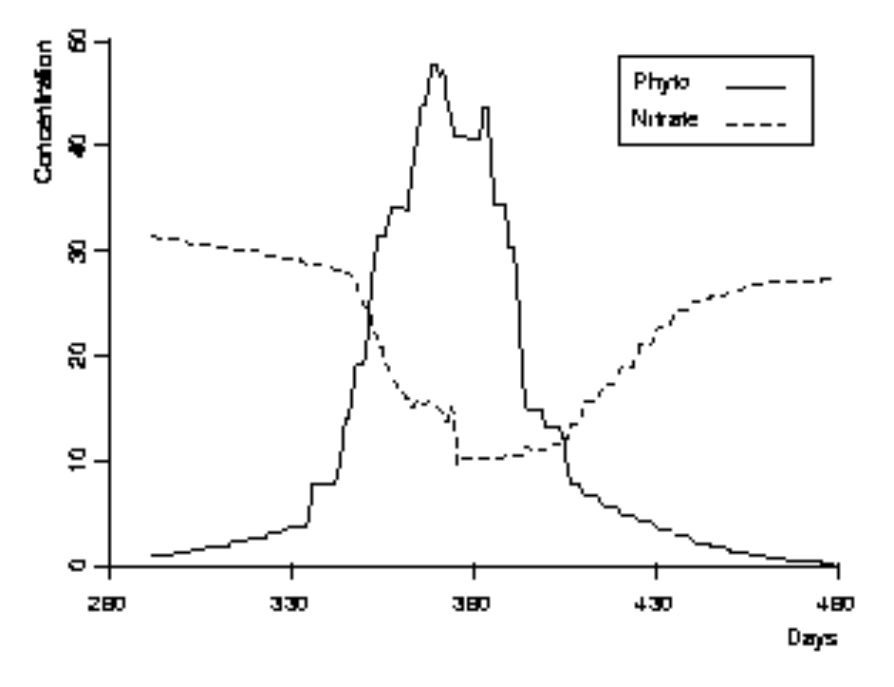
$$d[phyto,t,1] = -0.307 \times phyto - 0.495 \times zoo + 0.411 \times phyto$$

$$d[zoo,t,1] = -0.251 \times zoo + 0.615 \times 0.495 \times zoo$$

$$d[detritus,t,1] = 0.307 \times phyto + 0.251 \times zoo + 0.385 \times 0.495 \times zoo - 0.005 \times detritus$$

$$d[nitro,t,1] = -0.098 \times 0.411 \times phyto + 0.005 \times detritus$$

A Deeper Account of Ross Sea Dynamics



As phytoplankton uptakes nitrogen, its concentration increases and the nitrogen decreases. This continues until the nitrogen is exhausted, which leads to a phytoplankton die off. This produces detritus, which gradually remineralizes to replenish nitrogen. Zooplankton grazes on phytoplankton, which slows the latter's increase and also produces detritus.

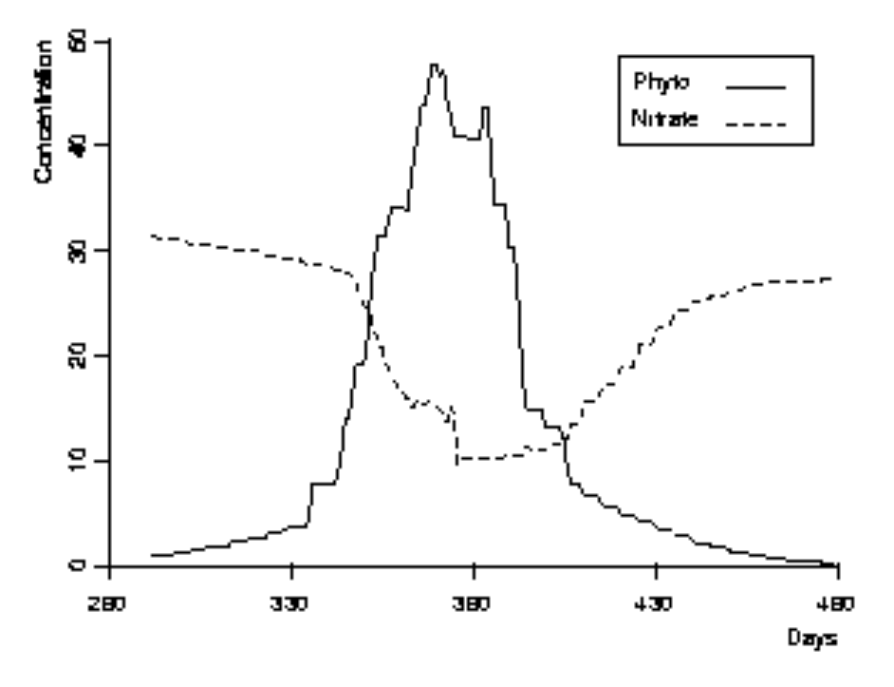
$$d[phyto,t,1] = -0.307 \times phyto - 0.495 \times zoo + 0.411 \times phyto$$

$$d[zoo,t,1] = -0.251 \times zoo + 0.615 \times 0.495 \times zoo$$

$$d[detritus,t,1] = 0.307 \times phyto + 0.251 \times zoo + 0.385 \times 0.495 \times zoo - 0.005 \times detritus$$

$$d[nitro,t,1] = -0.098 \times 0.411 \times phyto + 0.005 \times detritus$$

Processes in Ross Sea Dynamics



As phytoplankton uptakes nitrogen, its concentration increases and the nitrogen decreases. This continues until the nitrogen is exhausted, which leads to a phytoplankton die off. This produces detritus, which gradually remineralizes to replenish nitrogen. Zooplankton grazes on phytoplankton, which slows the latter's increase and also produces detritus.

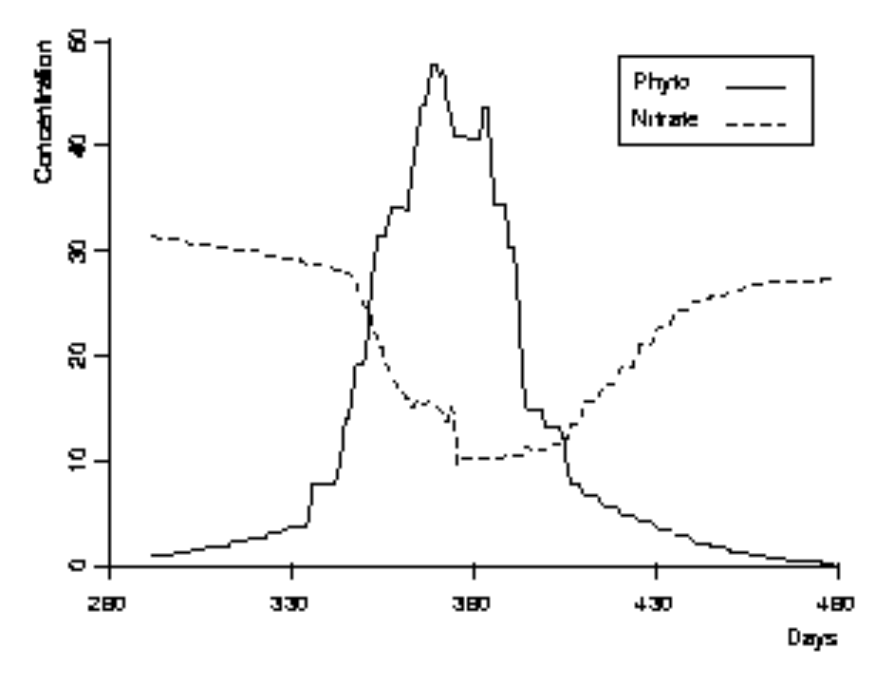
$$d[phyto,t,1] = -0.307 \times phyto - 0.495 \times zoo + 0.411 \times phyto$$

$$d[zoo,t,1] = -0.251 \times zoo + 0.615 \times 0.495 \times zoo$$

$$d[detritus,t,1] = 0.307 \times phyto + 0.251 \times zoo + 0.385 \times 0.495 \times zoo - 0.005 \times detritus$$

$$d[nitro,t,1] = -0.098 \times 0.411 \times phyto + 0.005 \times detritus$$

Processes in Ross Sea Dynamics



As phytoplankton uptakes nitrogen, its concentration increases and the nitrogen decreases. This continues until the nitrogen is exhausted, which leads to a phytoplankton die off. This produces detritus, which gradually remineralizes to replenish nitrogen. *Zooplankton grazes on phytoplankton, which slows the latter's increase and also produces detritus.*

d[phyto,t,1] = − 0.307 × phyto − 0.495 × zoo + 0.411 × phyto

d[zoo,t,1] = − 0.251 × zoo + 0.615 × 0.495 × zoo

d[detritus,t,1] = 0.307 × phyto + 0.251 × zoo + 0.385 × 0.495 × zoo − 0.005 × detritus

d[nitro,t,1] = − 0.098 × 0.411 × phyto + 0.005 × detritus

A Process Model for the Ross Sea

model Ross_Sea_Ecosystem

variables: phyto, zoo, nitro, detritus
observables: phyto, nitro

process phyto_loss(phyto, detritus)
  equations: d[phyto,t,1] = −0.307 × phyto
             d[detritus,t,1] = 0.307 × phyto

process zoo_loss(zoo, detritus)
  equations: d[zoo,t,1] = −0.251 × zoo
             d[detritus,t,1] = 0.251 × zoo

process zoo_phyto_grazing(zoo, phyto, detritus)
  equations: d[zoo,t,1] = 0.615 × 0.495 × zoo
             d[detritus,t,1] = 0.385 × 0.495 × zoo
             d[phyto,t,1] = −0.495 × zoo

process nitro_uptake(phyto, nitro)
  equations: d[phyto,t,1] = 0.411 × phyto
             d[nitro,t,1] = −0.098 × 0.411 × phyto

process nitro_remineralization(nitro, detritus)
  equations: d[nitro,t,1] = 0.005 × detritus
             d[detritus,t,1] = −0.005 × detritus

We can reformulate such an account by restating it as a *quantitative process model*.

Such a model is equivalent to a standard differential equation model, but it makes explicit assumptions about the processes involved.

Each process indicates that certain terms in equations must stand or fall together.

A New Discovery Task

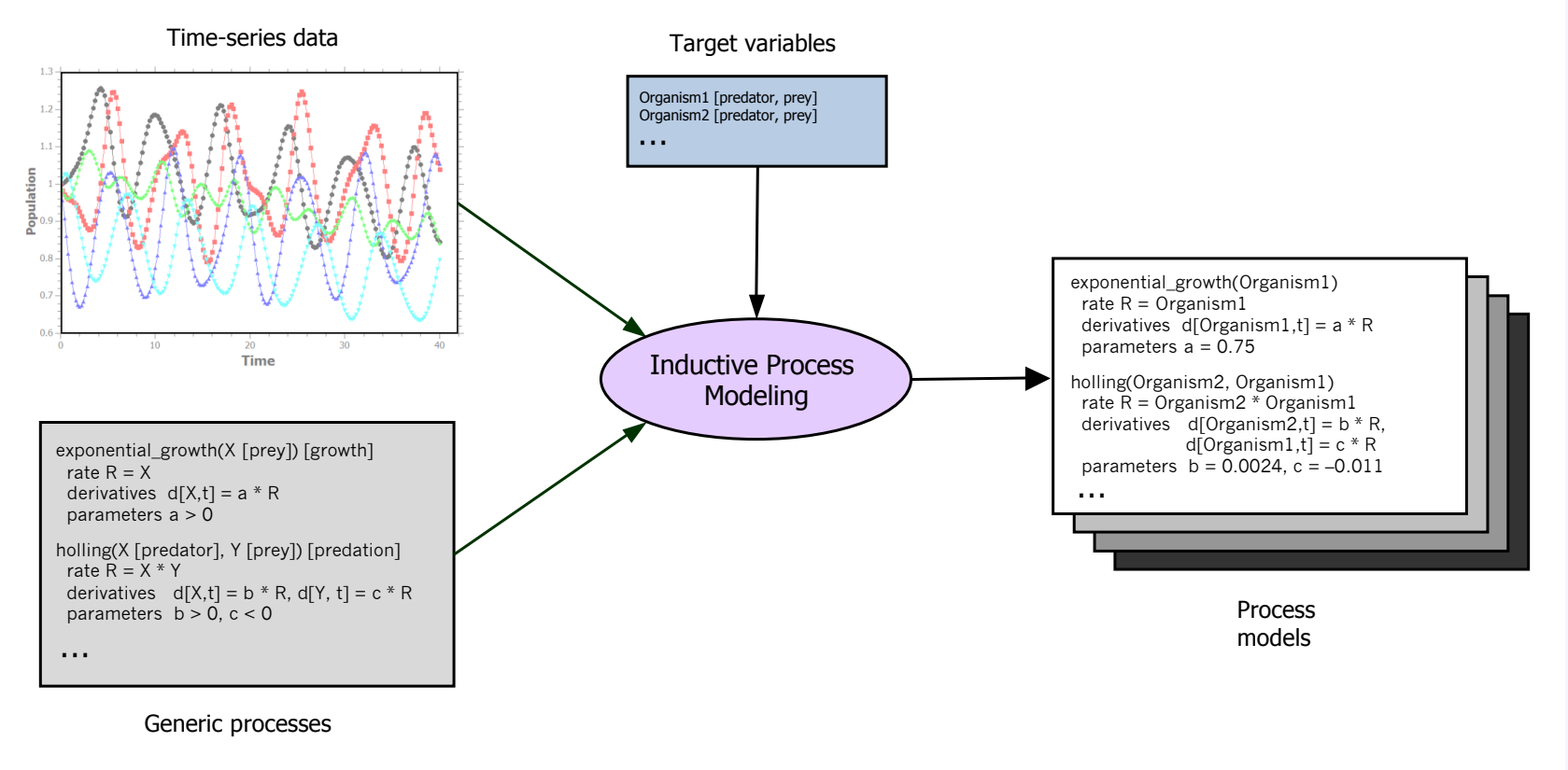
We can define the task of discovering such explanatory process models as:

- *Given:* A set of entities with associated variables
- *Given:* Times series for some of these variables
- *Given:* Knowledge about processes that might occur
- *Find:* Quantitative process models that explain the observed time series and predict new observations

We have referred to this class of computational discovery tasks as *inductive process modeling* (Bridewell et al., 2008).

Inductive Process Modeling

Inductive process modeling constructs explanations of time series from background knowledge (Langley et al., *ICML-2002*).



Models are stated as sets of *differential equations* organized into higher-level *processes*.

Some Generic Processes

process exponential_loss(S, D)
  variables: S{species}, D{detritus}
  parameters: α [0, 1]
  equations: d[S, t, 1] = $-1 \times \alpha \times S$
             d[D, t, 1] = $\alpha \times S$

generic process grazing(S1, S2, D)
  variables: S1{species}, S2{species}, D{detritus}
  parameters: ρ [0, 1], γ [0, 1]
  equations: d[S1, t, 1] = $\gamma \times \rho \times S1$
             d[D ,t, 1] = $(1 - \gamma) \times \rho \times S1$
             d[S2, t, 1] = $-1 \times \rho \times S1$

generic process nutrient_uptake(S, N)
  variables: S{species}, N{nutrient}
  parameters: τ [0, ∞], β [0, 1], μ [0, 1]
  conditions: $N > \tau$
  equations: d[S, t, 1] = $\mu \times S$
             d[N, t, 1] = $-1 \times \beta \times \mu \times S$

process remineralization(N, D)
  variables: N{nutrient}, D{detritus}
  parameters: π [0, 1]
  equations:
    d[N, t, 1] = $\pi \times D$
    d[D, t, 1] = $-1 \times \pi \times D$

process constant_inflow(N)
  variables: N{nutrient}
  parameters: ν [0, 1]
  equations: d[N, t, 1] = ν

Our aquatic ecosystem library contains about 25 generic processes, including ones with alternative functional forms for loss and grazing processes.

These form the *building blocks* from which to compose models.

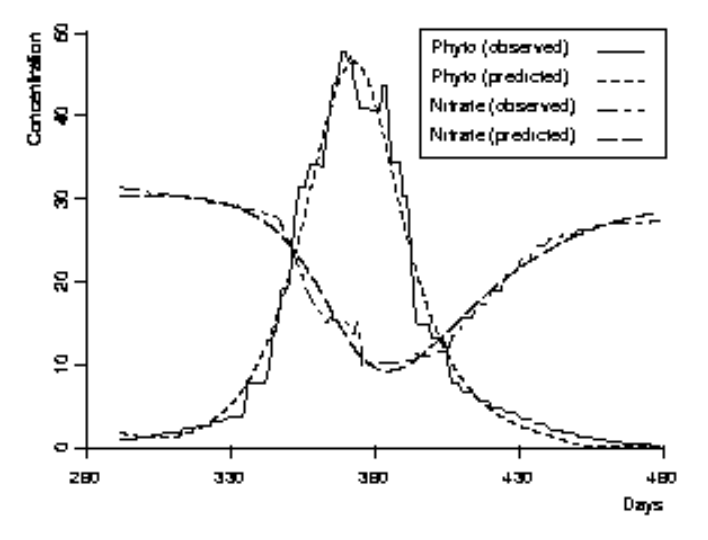
The SC-IPM System

We have reported SC-IPM (Bridewell & Langley, 2010), a system for inductive process modeling that:

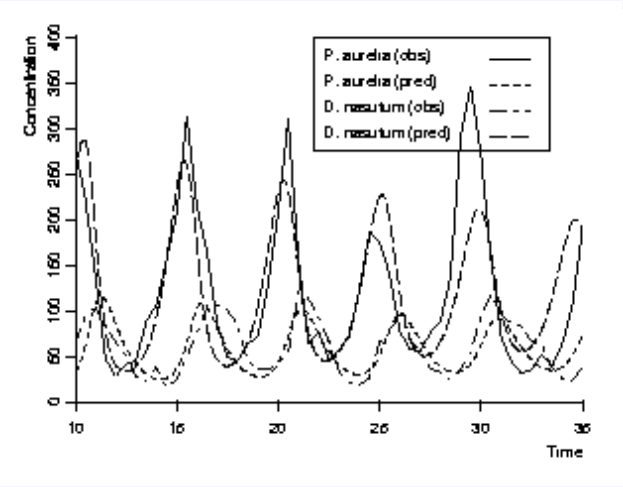
1. Uses background knowledge to generate *process instances*;
2. Combines them to produce possible *model structures*, rejecting ones that violate known constraints;
3. For each candidate model structure:
   a. Carries out gradient descent search through parameter space to find good coefficients;
   b. Invokes random restarts to decrease chances of local optima;
4. Returns the parameterized model with lowest squared error or a ranked list of models.

We have also presented encouraging results with SC-IPM on a variety of scientific data sets.

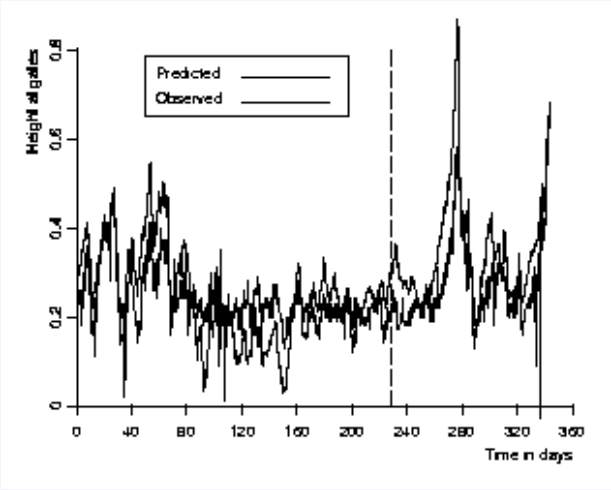
Some SC-IPM Successes



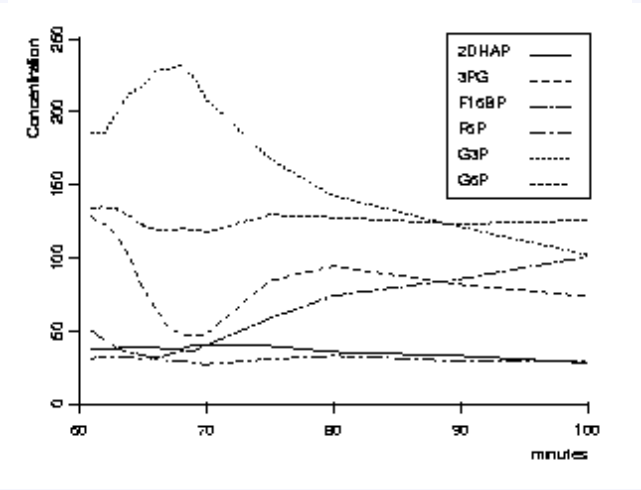
aquatic ecosystems



protist dynamics



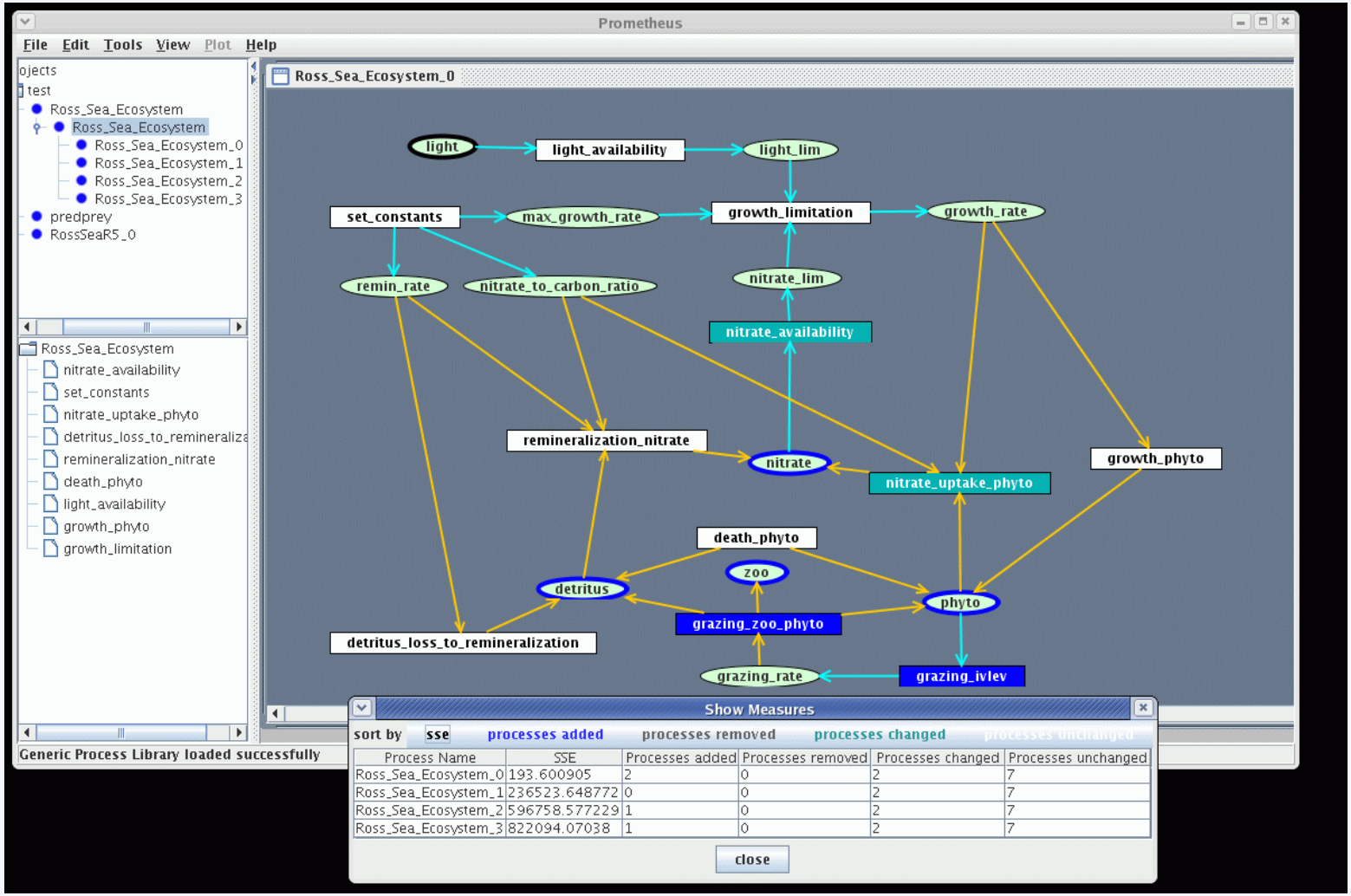
hydrology



biochemical kinetics

The PROMETHEUS System

We have embedded these ideas in PROMETHEUS, an interactive system for process model construction (Bridewell et al., *IJHCS*, 2007).



Extensions to Inductive Process Modeling

In addition, we have extended the basic framework to support:

- Inductive revision of quantitative process models
  - Asgharbeygi et al. (*Ecological Modeling*, 2006)
- Hierarchical generic processes that constrain search
  - Todorovski, Bridewell, Shiran, and Langley (*AAAI-2005*)
- An ensemble-like method that mitigates overfitting effects
  - Bridewell, Bani Asadi, Langley, and Todorovski (*ICML-2005*)
- An EM-like method that estimates missing observations
  - Bridewell, Langley, Racunas, and Borrett (*ECML-2006*)

These extensions make the modeling framework more robust along a number of fronts.

Critiques of SC-IPM

Despite these successes, the SC-IPM system suffers from four key drawbacks, in that it:

- Evaluates *full model structures*, so disallows heuristic search;
- Requires *repeated simulation* to estimate model parameters;
- Invokes *random restarts* to reduce chances of local optima;
- Despite these steps, it can still find poorly-fitting models.

99.99 percent of CPU time

As a result, SC-IPM does not scale well to complex modeling tasks and it is not reliable.

In recent research, we have developed a new framework that avoids these problems (Langley & Arvay, *AAAI-2015*).

A New Process Formalism

SC-IPM allowed processes with only algebraic equations, only differential equations, and mixtures of them.

In our new modeling formalism, each process P must include:

- A *rate* that denotes P's speed / activation on a given time step;
- An *algebraic equation* that describes P's rate as a *parameter-free* function of known variables;
- One or more *derivatives* that are proportional to P's rate.

This notation has important mathematical properties that assist model induction.

A Sample Process Model

Consider a process model for a simple predator-prey ecosystem:

```
exponential_growth[aurelia]
  rate         r = aurelia
  parameters   A = 0.75
  equations    d[aurelia] = A * r

exponential_loss[nasutum]
  rate         r = nasutum
  parameters   B = -0.57
  equations    d[nasutum] = B * r

holling_predation[nasutum, aurelia]
  rate         r = nasutum * aurelia
  parameters   C = 0.0024
               D = -0.011
  equations    d[nasutum] = C * r
               d[aurelia] = D * r
```

Each derivative is proportional to the algebraic rate expression.

A Sample Process Model

Consider a process model for a simple predator-prey ecosystem:

```
exponential_growth[aurelia]
  rate         r = aurelia
  parameters   A = 0.75
  equations    d[aurelia] = A * r

exponential_loss[nasutum]
  rate         r = nasutum
  parameters   B = -0.57
  equations    d[nasutum] = B * r

holling_predation[nasutum, aurelia]
  rate         r = nasutum * aurelia
  parameters   C = 0.0024
               D = -0.011
  equations    d[nasutum] = C * r
               d[aurelia] = D * r
```

This model compiles into a set of differential equations

```
d[aurelia] = 0.75 * aurelia – 0.011 * nasutum * aurelia
d[nasutum] = 0.0024 * nasutum * aurelia – 0.57 * nasutum
```

Some Generic Processes

Generic processes have a very similar but more abstract format:

```
exponential_growth(X [prey]) [growth]
  rate         r = X
  parameters   A = (> A 0.0)
  equations    d[prey] = A * r

exponential_loss(X [predator]) [loss]
  rate         r = predator
  parameters   B = (< B 0.0)
  equations    d[prey] = B * r

holling_predation(X [predator], Y [prey]) [predation]
  rate         r = X * Y
  parameters   C = (> C 0.0)
               D = (< D 0.0)
  equations    d[predator] = C * r
               d[prey] = D * r
```

As before, these are *building blocks* for constructing models.

RPM: Regression-Guided Process Modeling

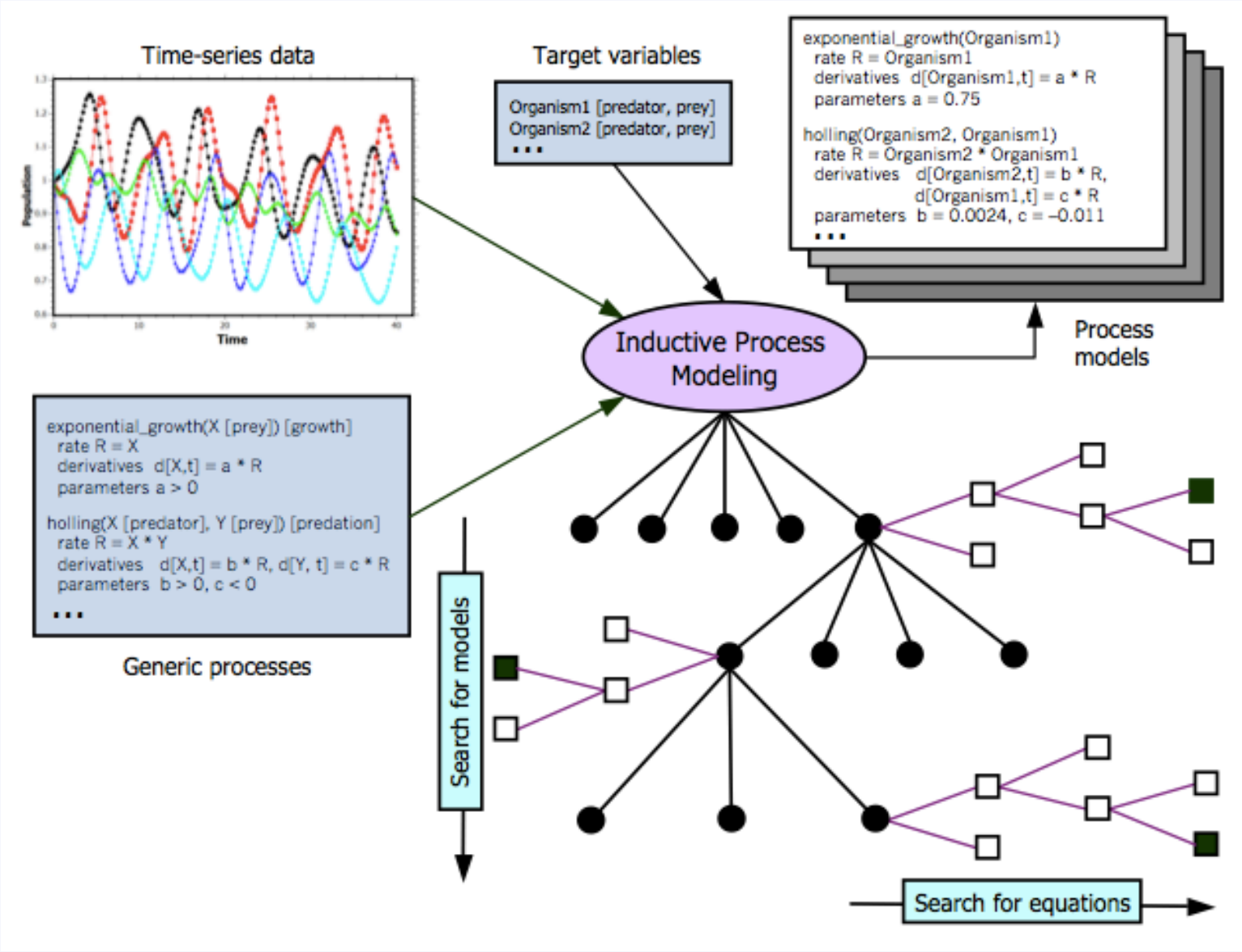
This suggests a new approach to inducing process models that our *RPM* system implements:

- Generate all process instances consistent with type constraints
- For each process P, calculate the *rate* for P on each time step
- For each dependent variable X,
  - Estimate *dX/dt* on each time step with center differencing,
  - For each subset of processes with up to *k* elements,
    - Find a regression equation for dX/dt in terms of process rates
    - If the equation's r^2 is high enough, retain for consideration
  - Add the equation with the highest r^2 to the process model

Assumes all variables observed
Rate expressions are parameter free

This approach factors the model construction task into a number of tractable components.

Two-Level Heuristic Search in RPM



Heuristics for Model Induction

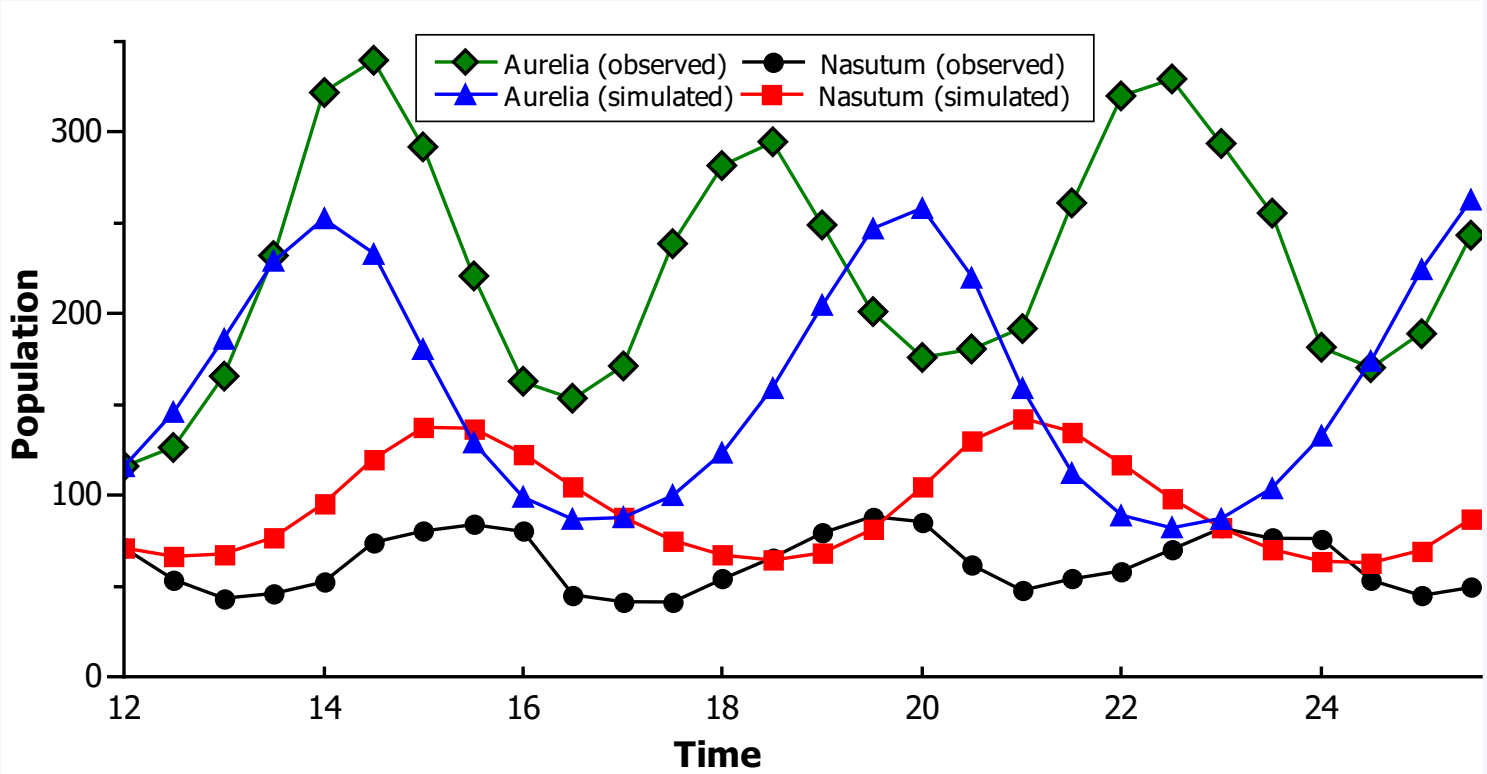
RPM uses four heuristics to guide its search through the space of process models:

- A model may include only one process instance of each type;
- Parameters must obey numeric constraints in generic processes;
- If an equation for one variable includes a process P, then P must appear in equations for other variables that P mentions;
- Incorporate variables that participate in more processes earlier than less constrained ones.

These heuristics reduce substantially the amount of search that RPM carries our during model induction.

Behavior on Natural Data

RPM matches the main trends for a simple predator-prey system.

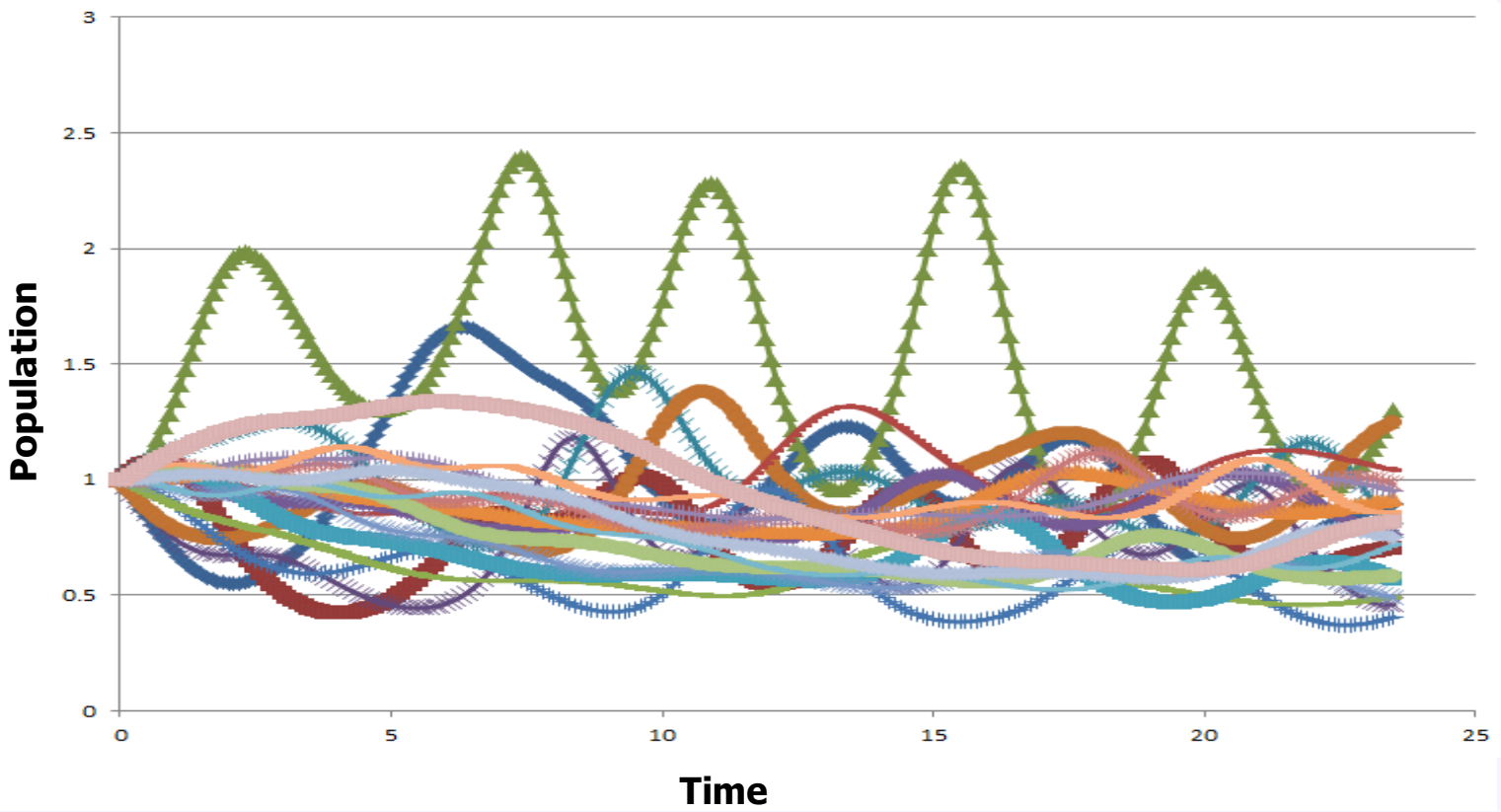


$d[aurelia] = 0.75 * aurelia - 0.11 * nasutum * aurelia \; [r^2 = 0.84]$

$d[naustum] = 0.0024 * nasutum * aurelia - 0.57 * nasutum \; [r^2 = 0.71]$

Behavior on Complex Synthetic Data

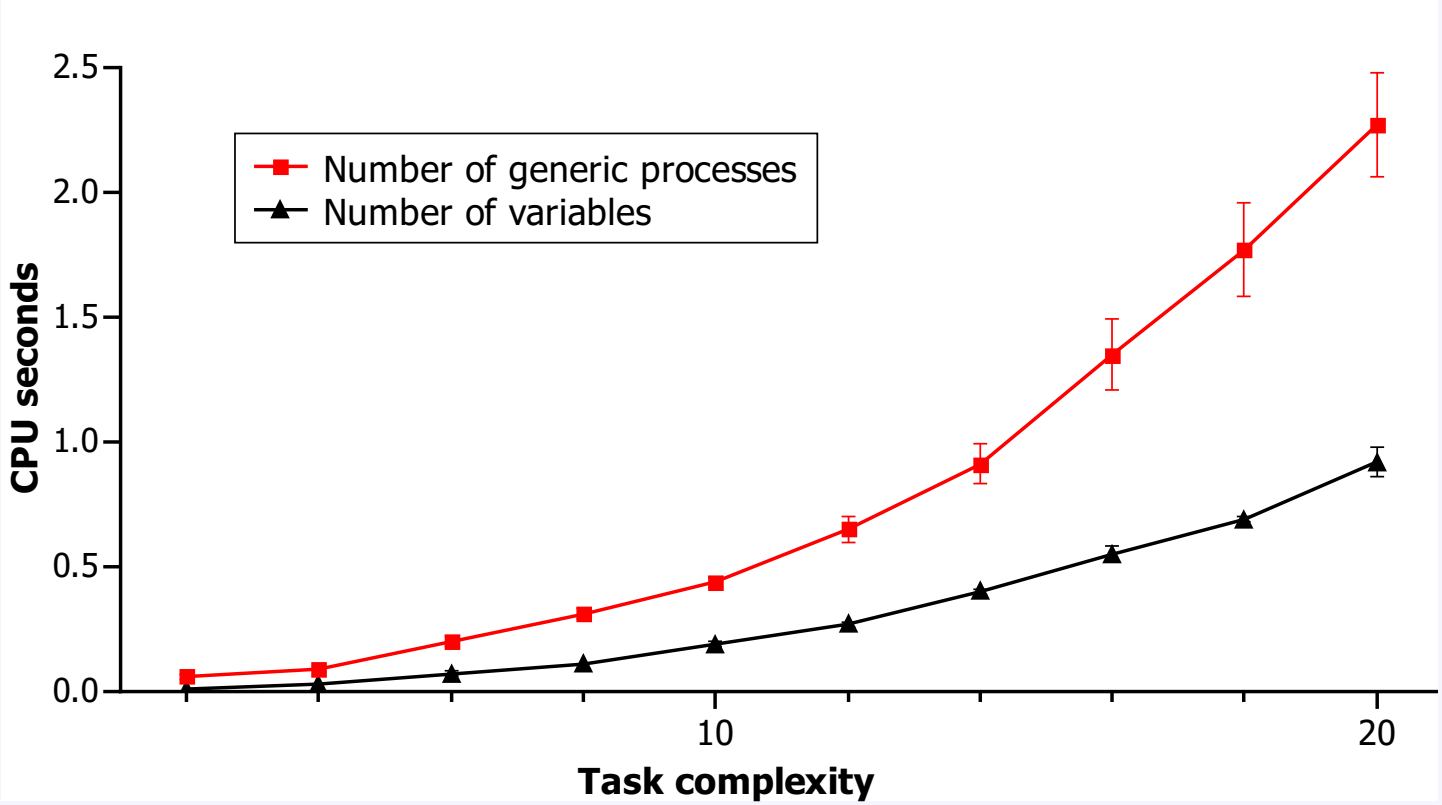
RPM also finds an accurate model for a 20-organism food chain.



This suggests the system scales well to difficult modeling tasks.

Handling Noise and Complexity

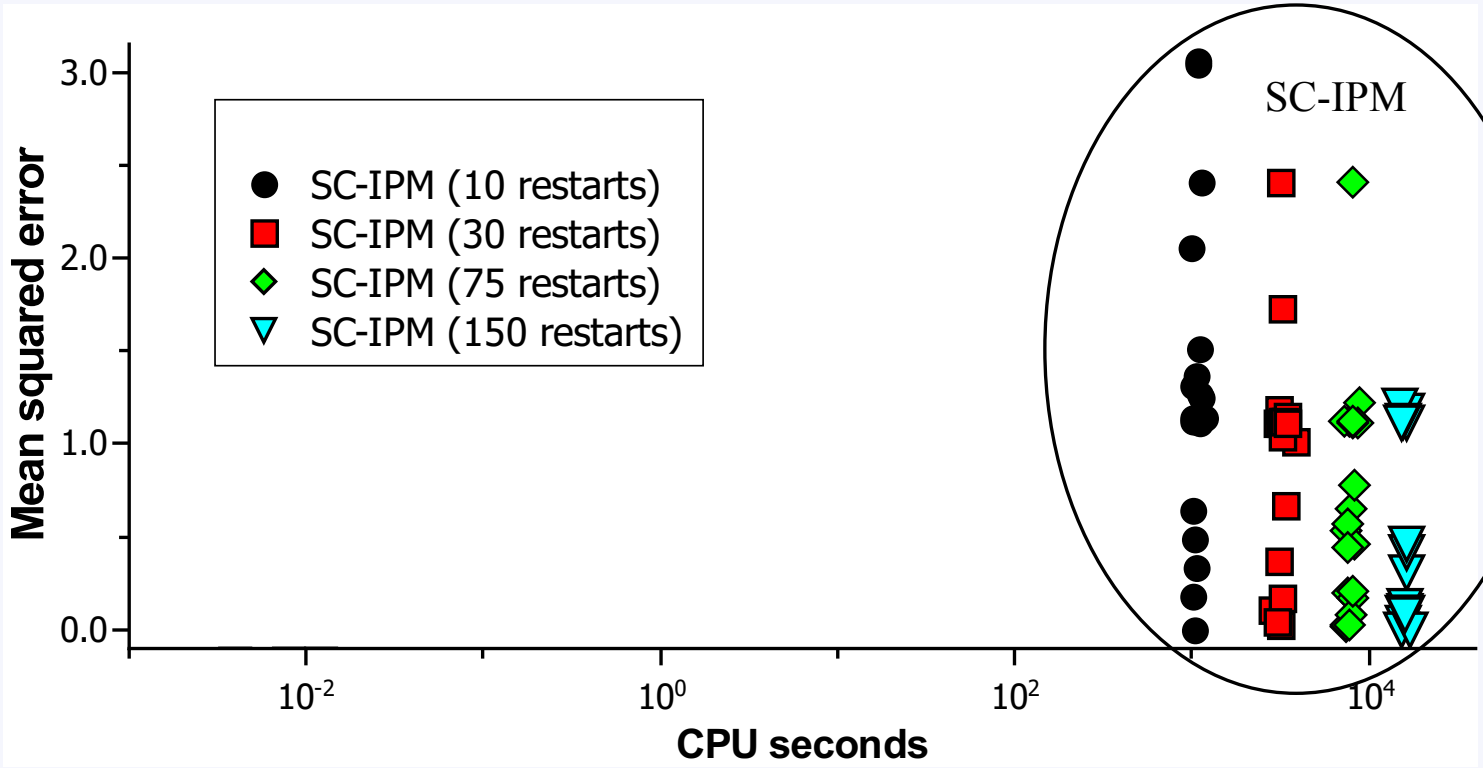
With smoothing, RPM can handle 10% noise on synthetic data.



The system also scales well to increasing numbers of generic processes and variables in the target model.

RPM and SC-IPM

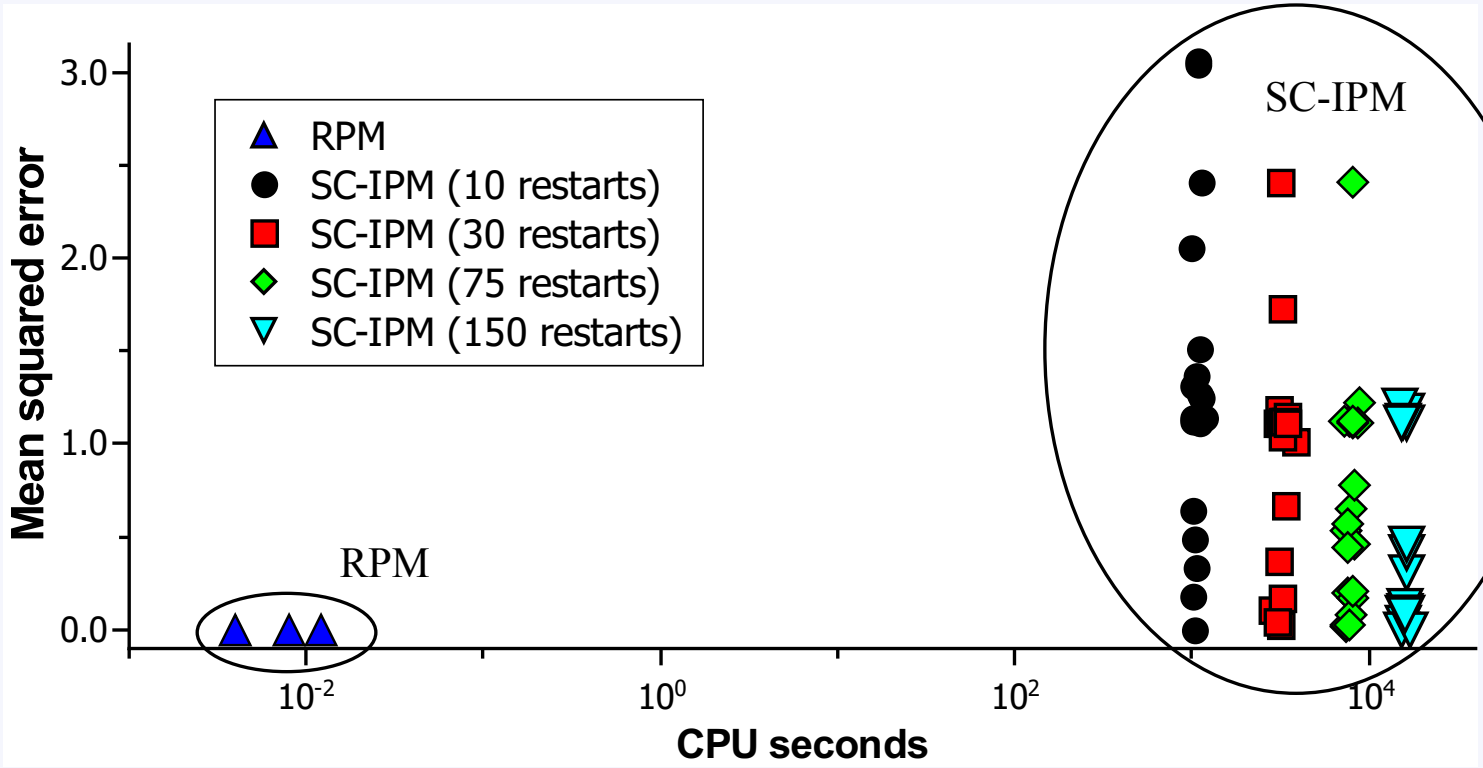
We compared RPM to SC-IPM, its predecessor, on synthetic data for a three-variable predator-prey ecosystem.



SC-IPM finds more accurate models with more restarts, but also takes longer to find them.

RPM and SC-IPM

We compared RPM to SC-IPM, its predecessor, on synthetic data for a three-variable predator-prey ecosystem.



RPM found accurate models far more reliably than SC-IPM and, at worst, ran *800,000 faster* than the earlier system.

Related and Future Research

Our approach builds on ideas from earlier research, including:

- Qualitative representations of scientific models (Forbus, 1984)
- Inducing differential equations (Todorovki, 1995; Bradley, 2001)
- Heuristic search and multiple linear regression

Our plans for extending the RPM system include:

- Replacing greedy search for models with beam search
- Adding heuristic search through the equation space
- Handling parametric rate expressions (e.g., using LMS)
- Dealing with unobserved variables (e.g., iterative optimization)

Together, these should extend RPM's coverage and usefulness.

Summary Remarks

Inductive process modeling is a novel and promising approach to discovering scientific models that:

- Incorporates a formalism that is familiar to many scientists
- Utilizes background knowledge about the problem domain
- Produces meaningful results from moderate amounts of data
- Generates models that explain, not just describe, observations
- Can scale well both to many processes and complex models

Although our work has focused on ecological modeling, the key ideas extend to other domains.

For more information, see *http://www.isle.org/process/* .

eScience and Discovery Informatics

The *escience* movement champions the use of computers to aid the scientific enterprise, emphasizing two themes:

- Creation and simulation of complex explanatory models
  - E.g., differential equation models for meteorology and biology
  - However, most such models are constructed *manually*
- Collection, storage, and mining of scientific data sets
  - E.g., learned classifiers in astronomy and planetology
  - But such analyses make no contact with scientific theory

Science is about the *relation between* theory and data, and work on computational scientific discovery offers a way to join them.

This idea is central to the emerging field of *discovery informatics*.

Big Data and Scientific Discovery

Digital collection and storage have led to rapid growth of data in many areas.

The *big data* movement seeks to capitalize on this content, but, in science at least, must address *three* distinct issues:

- Scaling to large and heterogeneous *data sets*
- Scaling to large and complex *scientific models*
- Scaling to large *spaces of candidate models*

Handling large data sets has been widely studied and poses the fewest challenges.

We need far more work on the second two issues, for which the methods of computational scientific discovery are well suited.

Concluding Remarks

Scientific discovery does not involve any mystical or irrational elements; we can study and even partially automate it.

Our explanation of this fascinating set of processes relies on:

- Carrying out search through a space of laws or models;
- Utilizing operators for generating structures and parameters;
- Guiding search by data and by knowledge about the domain.

Systems discover laws and models stated in the formalisms and concepts familiar to scientists.

This paradigm has already started to aid the scientific enterprise, and its importance will only grow with time.

Publications on Computational Scientific Discovery

Bridewell, W., & Langley, P. (2010). Two kinds of knowledge in scientific discovery. *Topics in Cognitive Science*, *2*, 36–52.

Bridewell, W., Langley, P., Todorovski, L., & Dzeroski, S. (2008). Inductive process modeling. *Machine Learning*, *71*, 1–32.

Bridewell, W., Sanchez, J. N., Langley, P., & Billman, D. (2006). An interactive environment for the modeling and discovery of scientific knowledge. *International Journal of Human-Computer Studies*, *64*, 1099–1114.

Dzeroski, S., Langley, P., & Todorovski, L. (2007). Computational discovery of scientific knowledge. In S. Dzeroski & L. Todorovski (Eds.), *Computational discovery of communicable scientific knowledge*. Berlin: Springer.

Langley, P. (2000). The computational support of scientific discovery. *International Journal of Human-Computer Studies*, *53*, 393–410.

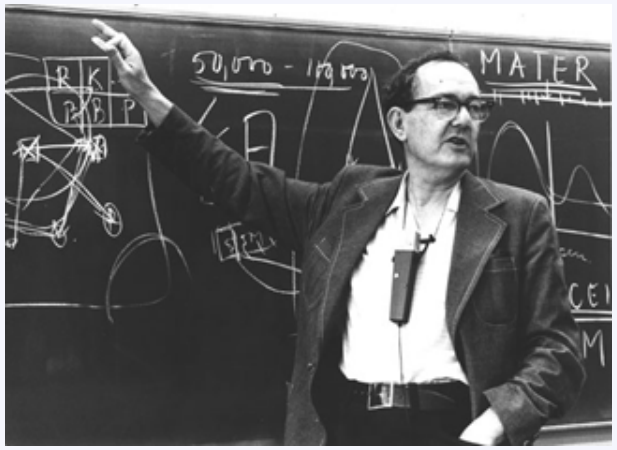
Langley, P., & Arvay, A. (2015). Heuristic induction of rate-based process models. *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. Austin, TX: AAAI Press.

Langley, P., Simon, H. A., Bradshaw, G. L., & Zytkow, J. M. (1987). *Scientific discovery: Computational explorations of the creative processes*. Cambridge, MA: MIT Press.

Langley, P., & Zytkow, J. M. (1989). Data-driven approaches to empirical discovery. *Artificial Intelligence*, *40*, 283–312.

In Memoriam

In 2001, the field of computational scientific discovery lost two of its founding fathers.



Herbert A. Simon
(1916 – 2001)

Jan M. Zytkow
(1945 – 2001)

Both were interdisciplinary researchers who published in computer science, psychology, philosophy, and statistics.

Herb Simon and Jan Zytkow were excellent role models for us all.