Ecological Modelling 245 (2012) 136–165

Contents lists available at SciVerse ScienceDirect

Ecological Modelling

journal homepage: www.elsevier.com/locate/ecolmodel

The influence of parameter fitting methods on model structure selection in automated modeling of aquatic ecosystems

Darko Čerepnalkoski [a,c,*], Katerina Taškova [b,c], Ljupčo Todorovski [e], Nataša Atanasova [f,g], Sašo Džeroski [a,c,d]

[a] Department of Knowledge Technologies, Jožef Stefan Institute, Jamova cesta 39, Ljubljana, Slovenia
[b] Computer Systems Department, Jožef Stefan Institute, Jamova cesta 39, Ljubljana, Slovenia
[c] Jožef Stefan International Postgraduate School, Jamova cesta 39, Ljubljana, Slovenia
[d] Centre of Excellence for Integrated Approaches in Chemistry and Biology of Proteins, Jamova cesta 39, Ljubljana, Slovenia
[e] Faculty of Administration, University of Ljubljana, Ljubljana, Slovenia
[f] Center for Marine and Environmental Research (CIMA), University of Algarve, Faro, Portugal
[g] International Center for Coastal Ecohydrology, Faro, Portugal

ARTICLE INFO

Article history:
Available online 5 July 2012

Keywords:
Aquatic ecosystems
Dynamical systems
Equation discovery
Process-based modeling
Parameter estimation
Meta-heuristic optimization

ABSTRACT

Modeling dynamical systems involves two subtasks: structure identification and parameter estimation. ProBMoT is a tool for automated modeling of dynamical systems that addresses both tasks simultaneously. It takes into account domain knowledge formalized as templates for components of the process-based models: entities and processes. Taking a conceptual model of the system, the library of domain knowledge, and measurements of a particular dynamical system, it identifies both the structure and numerical parameters of the appropriate process-based model. ProBMoT has two main components corresponding to the two subtasks of modeling. The first component is concerned with generating candidate model structures that adhere to the conceptual model specified as input. The second subsystem uses the measured data to find suitable values for the constant parameters of a given model by using parameter estimation methods. ProBMoT uses model error to rank model structures and select the one that fits measured data best.

In this paper, we investigate the influence of the selection of the parameter estimation methods on the structure identification. We consider one local (derivative-based) and one global (meta-heuristic) parameter estimation method. As opposed to other comparative studies of parameter estimation methods that focus on identifying parameters of a single model structure, we compare the parameter estimation methods in the context of repetitive parameter estimation for a number of candidate model structures. The results confirm the superiority of the global optimization methods over the local ones in the context of structure identification.

1. Introduction

Building models of ecosystems is at the heart of ecological modeling. Models of aquatic ecosystems, in particular, typically encompass the trophic relations among the species that are present in the system. Aquatic ecosystems are dynamical systems, the modeling of which involves several stages (Luenberger, 1979). When devising a model of an aquatic ecosystem, the first task is to decide on the structure of the model. Starting with a conceptual model of the system of interest and using knowledge about modeling aquatic ecosystems, the modeling expert selects the most suitable model structure for the observed ecosystem. This is the structure identification phase of the modeling process.

Once the model structure has been specified, the modeler has to determine suitable values for the numerical parameters of the model. To this end, he can use different techniques for estimating the values of the parameters from measured data about the behavior of the system. The data include measurements of the factors that drive the changes of the ecosystem, as well as the quantities that determine the state of the system. This is the parameter estimation phase of the modeling process.

Tools for automated modeling of dynamical systems simultaneously tackle both structure identification and parameter estimation

* Corresponding author at: Department of Knowledge Technologies, Jožef Stefan Institute, Jamova cesta 39, Ljubljana, Slovenia.
E-mail addresses: darko.cerepnalkoski@ijs.si (D. Čerepnalkoski), katerina.taskova@ijs.si (K. Taškova), ljupco.todorovski@fu.uni-lj.si (L. Todorovski), naatanasova@ualg.pt (N. Atanasova), saso.dzeroski@ijs.si (S. Džeroski).

0304-3800/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.ecolmodel.2012.06.001

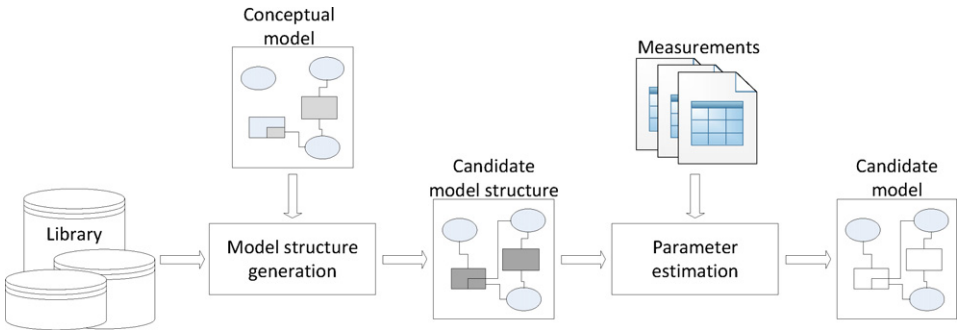


Fig. 1. Automated modeling with ProBMoT.

(Džeroski and Todorovski, 1993; Todorovski and Džeroski, 1997; Todorovski, 2003). We have developed ProBMoT—a Process-Based Modeling Tool, that uses a knowledge-based approach to solving the structure identification task. Background knowledge about modeling aquatic ecosystems is captured into a library of model fragments. Each model fragment is a small piece of structure that captures a single interaction in the system. ProBMoT then composes process-based models of the complete system under study by using the library of domain knowledge and a conceptual model, which constraint the space of candidate model structures.

Current tools for automated process-based modeling, such as LAGRAMGE 2.0 (Todorovski and Džeroski, 2006) and HIPM (Todorovski et al., 2005), use the local optimization method ALG-717 (Bunch et al., 1993) to solve the parameter estimation task for each candidate model structure. Local optimization algorithms are unsuitable for estimating the parameters of nonlinear models with many parameters. The parameter space of such a model forms a large and multimodal landscape and finding the global optimum presents too hard of a problem for a local algorithm. Global optimization methods (for an overview, see Horst et al., 2000) and in particular meta-heuristic methods are more robust in finding solutions in large and complex parameter spaces.

Global optimization methods have already been successful when used for parameter estimation in ecological and environmental models. Deterministic global optimization has been employed, among others, for parameter optimization of a groundwater model (Finley et al., 1998). Genetic algorithms and genetic programing have been extensively used in parameter calibration of lake ecosystem models (Whigham and Recknagel, 2001; Gilboa et al., 2009; Cao et al., 2008). Other global optimization algorithms used in calibrating ecological models include simulated annealing (Matear, 1995) and particle swarm optimization (Afshar et al., 2011). Some comparisons of the performance of different global optimization methods applied on the same task have also been performed (Athias et al., 2000; Zhang et al., 2009). One recent study (Tashkova et al., 2012) has performed a comparison among local and global optimization methods for estimating the parameters of a single model of the trophic relations in a lake ecosystem.

All these studies assume a single known model structure and focus on the task of parameter estimation. In contrast, when solving the structure identification task, automated modeling tools solve a series of parameter estimation tasks, one for each model structure. In this paper, we study the influence of the method for parameter estimation on the solution of the structure identification task. More specifically, we compare the performance of ProBMoT when using the local optimization method ALG-717 and a global method for parameter estimation, which is based on ant-colony optimization called Differential Ant Stigmergy Algorithm (DASA) (Korošec et al., 2012).

We compare the performance of ProBMoT/DASA and ProBMoT/ALG on a number of tasks of modeling phytoplankton dynamics in 21 data sets stemming from four lake ecosystems. We are particularly aimed at answering three central questions related to the performance of structure identification:

Best Model Improvement What is the difference between the errors of the best models obtained with ProBMoT/DASA and ProBMoT/ALG?
Overall Model Improvement What is the overall difference between the errors of the models obtained with ProBMoT/DASA and ProBMoT/ALG?
Individual Model Improvement What is the difference between the errors of each candidate model structure obtained with ProBMoT/DASA and ProBMoT/ALG?

The remainder of the paper is organized as follows. In Section 2, we introduce ProBMoT, a tool for automated modeling of dynamical systems from domain knowledge and observation data. Next, in Section 3, we describe the experimental setup for empirical evaluation and comparison of the local and global optimization methods used as parameter estimators within ProBMoT. In Section 4, we present and analyze the results of the empirical evaluation. Finally, Section 5 concludes the paper and presents directions for further research.

2. ProBMoT—Process-Based Modeling Tool

In this section, we present ProBMoT (Process-Based Modeling Tool), a tool for automated modeling of dynamical systems such as aquatic ecosystems. ProBMoT approaches modeling by using both data and domain knowledge about the given system. The general domain knowledge about the considered class of systems, e.g., lake food webs, is formulated in a library of domain knowledge. Using the components of the library and a conceptual model of the system at hand, candidate model structures are generated. Then, a parameter optimization method uses the available data from the system to estimate the numerical parameters of each model structure resulting in a completely specified candidate model. The model whose behavior matches the measurements best is the result of the automated modeling process. The procedure of automated modeling with ProBMoT is depicted in Fig. 1.

2.1. Process-based models

Process-based models (Bridewell et al., 2008) are a way of representing dynamical systems. They consist of two basic types of elements: entities and processes. Entities correspond to the actors of the observed system. These actors are involved in processes that explain how entities interact, as well as what is the influence of the

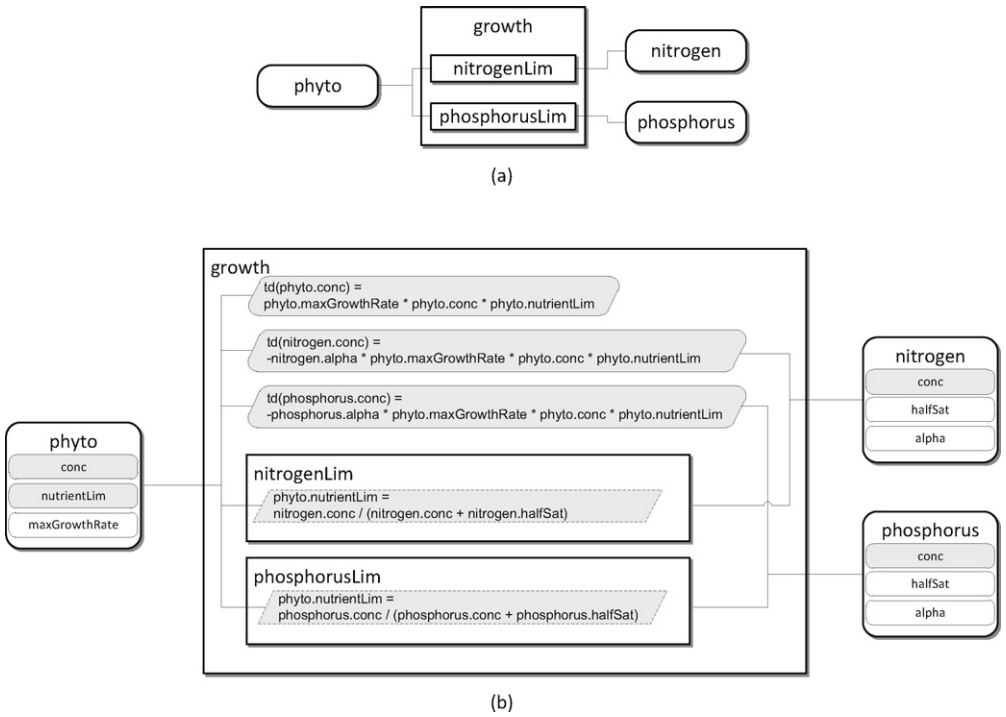


Fig. 2. An example model of trophic relation in a lake ecosystem. (a) Qualitative model and (b) quantitative model.

interactions on the involved entities themselves. When we deal with equation-based models, entities correspond to the variables and processes to arithmetical expressions (equation fragments) in the models.

The state of the system is represented as a set of entities. Each entity corresponds to one object that appears in the system. If we consider a simple lake ecosystem, nutrients such as *phosphorus* and *nitrogen*, as well as *phytoplankton* would be represented as entities. The phenomena that occur in the system would be described by the processes of the model. Each process in the model corresponds to a single phenomenon in the system. For instance, the *growth* of *phytoplankton* limited by *nitrogen* and *phosphorus* would be one process. This process has two arguments, the *phytoplankton* and the set of nutrients consisting of *phosphorus* and *nitrogen*.

Some processes that occur in the system can be more easily described in terms of several smaller and simpler processes. We consider such processes as component processes, because they act as components of the larger, more complex processes. In our simple lake ecosystem, the phytoplankton *growth* process can be defined in terms of two component processes that capture the limitation by nitrogen and phosphorus of the phytoplankton growth. The relations between the entities and processes in the system are depicted in Fig. 2(a).

In addition to the purely qualitative relations depicted in Fig. 2(a), process-based models provide quantitative information about the phenomena they capture. Each entity is described with one or more properties that are fixed, called constants, and properties that can change with time, called variables. Thus, *phytoplankton* can be specified by giving its *concentration*, *nutrient limitation* (both variables) and *maximal growth rate* (constant), whereas *nitrogen* can be specified by its *concentration* (variable) and the *half saturation* constant and *alpha* constant (stoichiometric ratio between algae biomass and the nutrient). The same variables and constants appear in *phosphorus*. Processes provide quantitative descriptions of the relation they represent as one or more equations. Furthermore, an equation can contain only variables and constants of the entities that participate in the corresponding process. For example, *nitrogen limitation*, which involves the *nitrogen* and *phytoplankton* entities, can only contain references to the variables and constants specified in either *nitrogen* or *phytoplankton*. These quantitative relations are depicted in Fig. 2(b).

Table 1 gives the representation of the simple lake ecosystem conceptualized in Fig. 2 in the formalism of process-based models. It contains three entities: *nitrogen*, *phosphorus* and *phytoplankton*. Each entity, as noted earlier, is specified with a list of its variables and constants, which are the properties of the entity that are important in the given context. For each variable, we provide its aggregation function, which specifies the method of aggregation of the influences of different processes on that variable. The *nutrientLim* variable of *phytoplankton*, for example, specifies that if there are several different processes influencing that variable, these influences are to be multiplied. Constants are defined in a straightforward manner, by assigning a numerical value to them.

Table 1 also presents the main process of the system, the growth of phytoplankton, limited by nitrogen and phosphorus, together with its two component processes that capture the limitation by the nutrients: *nitrogenLim* and *phosphorusLim*. The limitation by the nutrients is formulated as a Monod (saturation) function and captured within the nutrientLim variable. The *growth* process specifies a nutrient limited model for growth captured in the time-derivative (td) equations for the concentration of *phytoplankton*, *nitrogen* and *phosphorus*.

For each variable in the model, we compile one equation which will have that variable on its left hand side. The equation is compiled by combining all equations in the model that influence that variable, i.e., all equations which have the variable on the left hand side. The aggregation function for combing the equations is given in the definition of that variable. For instance, the variable phyto.nutrientLim (the variable nutrientLim of the entity phyto) is influenced by two equations, those in the processes nitrogenLim and phosphorusLim. Having in mind that the aggregation of the influences is performed by multiplication, we obtain the following equation for *phyto.nutrientLim*:

Table 1
A simple model of a lake ecosystem comprising nitrogen, phosphorus and phytoplankton.

```
model simpleLakeModel;
entity nitrogen {
  vars:
    conc {aggregation:sum};
  consts:
    halfSat = 3.38,
    alpha = 10.11;
}
entity phosphorus {
  vars:
    conc {aggregation:sum};
consts:
    halfSat = 0.05,
    alpha = 0.09;
}
entity phyto {
  vars:
    conc {aggregation:sum},
    nutrientLim{aggregation:product};
  consts:
    maxGrowthRate = 2.99;
}
process growth(phyto, [nitrogen, phosporus]) {
  processes:
    nitrogenLim(phyto, nitrogen),
    phoshporusLim(phosphorus, nitrogen);
  equations:
    td(phyto.conc) = phyto.maxGrowthRate
    * phyto.nutrientLim * phyto.conc,
    td(nitrogen.conc) = -nitrogen.alpha
      *phyto.maxGrowthRate*phyto.nutrientLim*phyto.conc,
    td(phosphorus.conc) = -phosphorus.alpha
      *phyto.maxGrowthRate*phyto.nutrientLim*phyto.conc;
}
process nitrogenLim(phyto, nitrogen) {
  equations:
    phyto.nutrientLim =
    nitrogen.conc / (nitrogen.conc + nitrogen.halfSat);
}
process phosphorusLim(phyto, phosphorus) {
  equations:
    phyto.nutrientLim = phosphorus.conc /
    (phosphorus.conc + phosphorus.halfSat);
}
```

$$phyto.nutrientLim = \frac{nitrogen.conc}{nitrogen.conc + nitrogen.halfSat} \cdot \frac{phosphorus.conc}{phosphorus.conc + phosphorus.halfSat} \quad (1)$$

By compiling the equations for all variables, we get a system of ordinary differential equations (ODEs) which is a quantitative model of the system. This model can then be used to perform quantitative analysis such as model simulation and parameter calibration.

2.2. Libraries of domain knowledge

Different entities in the system can have common properties. If we compare the nutrients *nitrogen* and *phosphorus* from Table 1, we can see that they share similarities with respect to their variables and constants. This is to be expected, since they are both nutrients. Properties which hold for more entities are specified in objects which we call entity templates. Entity templates are used for specifying common properties of entities. The idea is that the template captures some general knowledge that holds for many different cases and can be reused when dealing with different specific scenarios. Nevertheless, an entity template is an incomplete entity specification in that it only contains partial information for an entity. This information, however, is general and can be used for creating more than one entity.

Table 2
A library of domain knowledge for the simple lake ecosystem.

```
library SimpleLakeLibrary;
template entity Nutrient {
  vars:
    conc {aggregation:sum};
  consts:
    halfSat {range: <0,15>},
    alpha {range: <0,inf>};
}
template entity Phytoplankton {
  vars:
    conc {aggregation:sum},
    nutrientLim{aggregation:product};
  consts:
    maxGrowthRate {range: <0.05,3>};
}
template process
Growth(pp : Phytoplankton, ns : Nutrient<1, inf>) {
  processes:
    NutrientLimitation(pp, <n:ns>);
  equations:
    td(pp.conc) =
    pp.maxGrowthRate * pp.nutrientLim * pp.conc,
    td(<n:ns>.conc) = -n.alpha * pp.maxGrowthRate
    * pp.nutrientLim * pp.conc;
}
template process
NutrientLim(pp : Phytoplankton, n : Nutrient) {
  equations:
    pp.nutrientLim = n.conc / (n.conc * n.halfSat);
}
```

Similarly, if we compare the processes *nitrogenLim* and *phosphorusLim* from Table 1, we can see that they have equations that adhere to the same general pattern, which is reasonable because they both represent processes of saturated Monod type nutrient limitation. Therefore, it makes sense to try to group such similar processes within some more general concepts. For this reason, we create process templates – objects that represent parameterized recipes for creating processes. They can be seen as incomplete processes, i.e., processes that only contain some general information and lack some specific information.

The set of process and entity templates relevant to the domain in question are collected into a library of domain knowledge. Table 2 shows the domain knowledge library which corresponds to the simple lake ecosystem model given in Table 1. This library defines two entity templates, *Nutrient* and *Phytoplankton*. They contain the appropriate definitions for variables and constants which were present in the model in Table 1. One important difference is that a constant in the library is not bound to a particular numerical value, but instead specifies a range of allowed values, allowing to be reused for different entities with different values for each entity.

In addition to the two entity templates, the simple lake ecosystem library defines two process templates: *Growth* and *NutrientLim*. The *Growth* process template defines a recipe for defining processes of phytoplankton growth limited by a set of nutrients. Hence, the *Growth* process template has two arguments, the first of type *Phytoplankton* named *pp*, which represents the phytoplankton whose growth is being modeled, and the second a set of type *Nutrient* named *ns*, which represents the nutrients that limit the growth of the phytoplankton. The specification *Nutrient<1, inf>* denotes a set of *Nutrients* which has to contain at least one element and can contain arbitrarily many elements. The *Growth* process contains a component process of type *NutrientLim* for each *Nutrient* from the set of nutrients ns denoted by the declaration: *NutrientLimitation(pp, <n:ns>)*. It also contains one time-derivative equation for

Table 3
A simple lake model specified by using the simple lake library.

```
model simpleLakeModel : SimpleLakeLibrary;
entity nitrogen : Nutrient {
  vars:
    conc;
  consts:
    halfSat = 3.38,
    alpha = 10.11;
}
entity phosphorus : Nutrient {
  vars:
    conc;
  consts:
    halfSat = 0.05,
    alpha = 0.09;
}
entity phyto : Phytoplankton {
  vars:
    conc,
    nutrientLim;
  consts:
    maxGrowthRate = 2.99;
}
process growth(phyto, [nitrogen, phosphorus]) : Growth {
  processes:
    nitrogenLim, phosphorusLim;
}
process nitrogenLim(phyto, nitrogen) : NutrientLim {}
process phosphorusLim(phyto, phosphorus) : NutrientLim {}
```

the concentration of the phytoplankton, and one equation for each nutrient from the set of nutrients *ns*.

Having defined entity and process templates, we can use them to create suitable entity and process instances. Every new instance acquires all of the properties which were specified in the template. Properties which are characteristic for the particular instance itself are specified within the instance definition. Using the library from Table 2, the simple lake model can be represented as given in Table 3. It is evident that using the templates defined in the library we obtain a terse specification of the model. Instead of providing a complete specification of the entities and processes in the model itself, we create instances of the templates from the library. In the simple lake model, *nitrogen* and *phosphorus* are instances created using the *Nutrient* template, whereas *phyto* is created using the *Phytoplankton* template. The *growth* process, on the other hand, is created from the process template *Growth*, having *phyto* as the first argument and the set of *nitrogen* and *phosphorus* as the second.

2.3. Organizing entities and processes into hierarchies

Entity templates can be further arranged into hierarchies. We put more common properties in the entity templates which are higher in the hierarchy. This enables the entity templates which are lower in the hierarchy to inherit the properties of their ancestors and provides a cleaner and more reusable design of the entity templates. Fig. 3(a) shows how the entity templates can be arranged into a hierarchy in the simple lake library. Note that *Nutrient* and *Phytoplankton* shared the common variable concentration (*conc*) which has now moved to the *EcosystemEntity*. *Nutrient* and *Phytoplankton* are now subtypes of *EcosystemEntity* and thus inherit its properties, i.e., they inherit *conc*.

In the model in Table 3, nitrogen and phosphorus limitation had the same functional form – that of a Monod function. In practice, however, they can have any of a number of different functional forms such as Monod2 or exponential functions. We can arrange the limitation processes into a hierarchy like the one presented in Fig. 3(b) (Todorovski et al., 2005). The process template *NutrientLim* is at the root of the hierarchy. It does not supply and functional form by itself, but rather serves to organize the different nutrient limitation functions which are provided. Each of the different limitation functions has *NutrientLim* as their ancestor in the process hierarchy. In this manner, we have formalized the alternatives for the conceptual process of nutrient limitation.

Using this approach, we can construct a library with hierarchies of entity types and process alternatives. Table 4 gives the hierarchical form of the simple lake library while the corresponding model is given in Table 5. Note that this model is the same as the model in Table 3, with the exception of the limitation processes which now have different forms.

2.4. Conceptual models

The model presented in Table 5 has a fully specified structure and fully specified parameters. Instead of committing to specific processes, we can build a model which contains some process templates at higher levels of the hierarchy. We will refer to the latter as conceptual processes. Conceptual processes do not have a specific functional form, but rather state the semantics of the relation without committing to a particular equation. In the hierarchy of process templates, conceptual processes are process templates which are higher in the hierarchy, i.e., have children process templates beneath them. We call a model with conceptual processes a conceptual model. Because a conceptual model contains processes which are not bound to particular equations it is not a complete model. As such, it cannot be translated to a set of differential equations, and simulated. A conceptual model is an abstraction for a whole class of specific models.

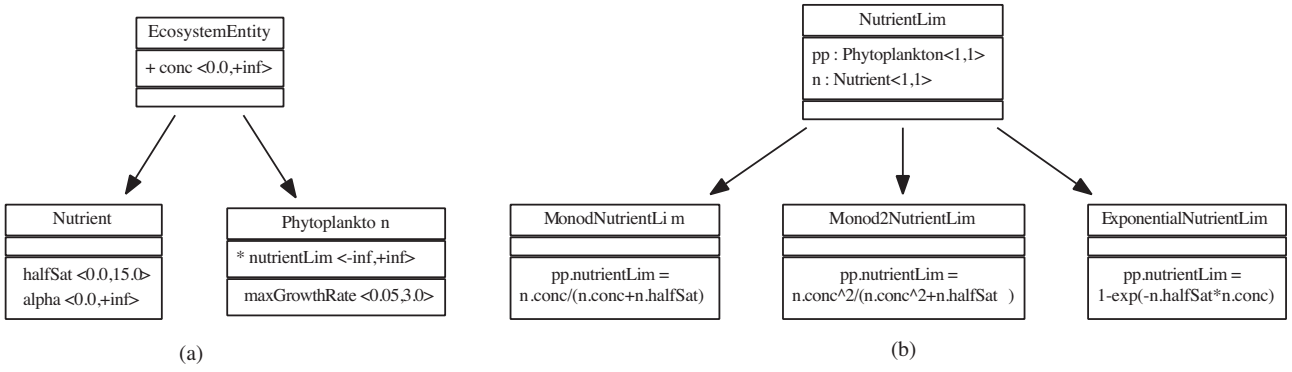


Fig. 3. Hierarchies of templates in the simple lake library. (a) Hierarchy of entity templates and (b) hierarchy of process templates.

Table 4
A hierarchical version of the simple lake library.

```
library SimpleLakeLibrary;
template entity EcosystemEntity {
  vars:
    conc {aggregation:sum};
}
template entity Nutrient : EcosystemEntity {
  consts:
    halfSat {range: <0,15>},
    alpha {range: <0,inf>};
}
template entity Phytoplankton : EcosystemEntity {
  vars:
    nutrientLim{aggregation:product};
  consts:
    maxGrowthRate {range: <0.05,3>};
}
template process
Growth(pp : Phytoplankton, ns : Nutrient<1, inf>) {
  processes:
    NutrientLim(pp, <n:ns>);
  equations:
    td(pp.conc) =
    pp.maxGrowthRate * pp.nutrientLim * pp.conc,
    td(<n:ns>.conc) = -n.alpha * pp.maxGrowthRate
    * pp.nutrientLim * pp.conc;
}
template process
NutrientLim(pp : Phytoplankton, n : Nutrient) {}
template process MonodNutrientLim : NutrientLim {
  equations:
    pp.nutrientLim = n.conc / (n.conc + n.halfSat);
}
template process Monod2NutrientLim : NutrientLim {
  equations:
    pp.nutrientLim =
    n.conc*n.conc / (n.conc*n.conc + n.halfSat);
}
template process ExponentialNutrientLim : NutrientLim {
  equations:
    pp.nutrientLim = 1 - exp(-n.halfSat * n.conc);
}
```

Table 5
A simple lake model using the hierarchical library.

```
model simpleLakeModel : SimpleLakeLibrary;
entity nitrogen : Nutrient {
  vars:
    conc;
  consts:
    halfSat = 3.38,
    alpha = 10.11;
}
entity phosphorus : Nutrient {
  vars:
    conc;
  consts:
    halfSat = 0.05,
    alpha 0.09;
}
entity phyto : Phytoplankton {
  vars:
    conc,
    nutrientLim;
  consts:
    maxGrowthRate = 2.99;
}
process growth(phyto, [nitrogen, phosphorus]) : Growth {
  processes:
    nitrogenLim, phosphorusLim;
}
process nitrogenLim(phyto, nitrogen) : MonodNutrientLim {}
process phosphorusLim(phyto, phosphorus) :
ExponentialNutrientLim {}
```

Table 6
A conceptual model for the simple lake ecosystem.

```
conceptual model simpleConceptualModel: simpleLakeLibrary;
entity nitrogen : Nutrient {
  vars:
    conc;
  consts:
    halfSat = null,
    alpha = null;
}
entity phosphorus : Nutrient {
  vars:
    conc;
  consts:
    halfSat = null,
    alpha = null;
}
entity phyto : Phytoplankton {
  vars:
    conc,
    nutrientLim;
  consts:
    maxGrowthRate = null;
}
process growth(phyto, [nitrogen, phosphorus]) : Growth {
  processes:
    nitrogenLim, phosphorusLim;
}
process nitrogenLim(phyto, nitrogen) : NutrientLim {}
process phosphorusLim(phyto, phosphorus) : NutrientLim {}
```

A conceptual model does not need to specify the values for each parameter. On the contrary, it is common practice for a conceptual model to omit the values of the numerical parameters, except the ones that are known to hold for certain. A constant parameter which is unspecified in the conceptual model is assigned the special value *null*. A conceptual model for the lake ecosystem is presented in Table 6. The *nitoregenLim* and *phosphorusLim* process do not specify any alternative formulation of nutrient limitation but are rather declared as general *NutrientLim* processes. In addition, all of the parameters present in the model are left unspecified.

2.5. Generation of specific candidate model structures

A conceptual process is an abstraction for the processes that are beneath it in the process hierarchy. When we substitute one of the specific processes for the conceptual process, we say that it has acquired a specific functional form. The process nitrogenLim from the conceptual model in Table 6 is declared as a conceptual process of type NutrientLim. Using the library from Table 4, we can conclude that nitrogenLim can acquire three different functional forms: MonodNutrientLim, Monod2NutrientLim or ExponentialNutrientLim. Similarly, phosphorusLim also has the type NutrientLim, which means that it can also acquire the same three different forms independently of nitrogenLim. Hence, the whole conceptual model can be specified in $3 \times 3 = 9$ different ways. In other words, we say that it generates 9 specific models. ProBMoT generates all specific models for a given conceptual model, which are obtained by using all possible substitution rules. These specific models have fully specified model structure, but can have unspecified parameters. The values of the unspecified parameters are determined by parameter estimation.

2.6. Parameter estimation

Parameter estimation is the final step in automated modeling of dynamical systems (Gershenfeld, 1999). Given a model structure for the observed system and measured data, its goal is to estimate the model parameters in order to minimize the discrepancy

between the measurements and model predictions. In particular, we use the maximum-likelihood estimator introduced by R. A. Fisher in 1912 (Pfanzagl, 1994) that maximizes the probability of observing the given data if a given model is chosen. The likelihood function depends on the probability of the measurements in the data set. Assuming that the measurements follow independent normal distributions with constant variance, the maximum-likelihood parameter estimation maps into a nonlinear least-squares estimation of the parameters, which minimizes the sum of squared errors between the observed values and the values predicted by the model. Due to its intuitive appeal and simplicity, least-squares estimation is commonly used for parameter estimation in nonlinear models.

The most basic form of least-squares estimation uses the sum of squared errors (*SSE*) for expressing the model error, which sums up the squares of the differences between the measured values y_i and the predicted values $\hat{y}_i$, at each time point i.

$$SSE = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{2}$$

Root mean squared error (*RMSE*) divides the *SSE* by the number of time points N and takes the square root, making the measurement units and scale comparable to the ones of the observed output variable.

$$RMSE = \sqrt{\frac{1}{N} SSE} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2} \tag{3}$$

Parameter estimation leads to challenging optimization tasks that typically require advanced meta-heuristic approaches for global optimization, such as evolutionary or swarm-based methods. Ecological models are typically nonlinear and have many parameters; the measurements are sparse and imperfect due to noise. All these constraints can lead to identifiability problems, i.e., the inability to uniquely identify the unknown model parameters, making parameter estimation an even harder optimization task.

2.6.1. Local and global optimization methods

Classical approaches to nonlinear continuous optimization are mainly local optimization methods (such as direct-search and derivative-based methods) that rapidly converge to the optimum, provided that the search is started from an initial point that is in close proximity of the optimum. As these methods do not have mechanism to escape from the local optima, they only guarantee local convergence. Derivative-based methods are an adequate choice for smooth and unimodal objective functions, but they can fail if the landscape is discontinuous, non-smooth, multi-modal or ill-conditioned. The disadvantage of the direct-search method is that they become less efficient for high-dimensional problems. Therefore, it is recommended to use global optimization approaches that are more robust regarding the dimensionality and the landscape characteristic of the search space.

Global optimization approaches can be divided into deterministic (exact) and stochastic (probabilistic). The deterministic methods (e.g., branch and bound, interior-point, and cutting planes) can locate the global optima and assure their optimality, but do not guarantee that they can solve any type of global optimization problems in finite time. Stochastic methods, on the other hand, rely on probabilistic search rules to find good solutions (Törn et al., 1999). They can locate the neighborhood of the global optima relatively quickly, but their efficiency comes at the cost of not being able to guarantee global optimality. In the last two decades, special attention has been given to meta-heuristics: These are general-purpose algorithms that can find acceptable solutions in a reasonable time-frame in complex and large search domain. Most meta-heuristics are inspired by natural processes such as evolution (evolutionary algorithms) or social behavior of biological organisms, e.g., ant colony optimization (Dorigo and Stützle, 2004).

Our study includes comparison of two optimization methods: the differential ant-stigmergy algorithm (DASA), a recently developed meta-heuristic method for global optimization, and the derivative-based algorithm 717 (ALG-717) updated with random restarts to cope with the challenge of multiple local optima.

2.6.2. Algorithm 717 (ALG-717)

Algorithm 717 (ALG-717) is a set of modules for solving the parameter estimation problem in nonlinear regression models, including the nonlinear least-squares, maximum likelihood, and some robust fitting problems (Bunch et al., 1993). The basic method is a generalization of NL2SOL—an adaptive nonlinear least-squares algorithm, which uses a model/trust-region technique for computing trial steps along with an adaptive choice for the Hessian model. Since ALG-717 is not a global search algorithm, we wrapped the original procedure in a loop of restarts with randomly chosen initial points, providing in some way a simple global search. The number of restarts was set to use a number of function evaluations comparable to that of the other method (DASA). We used the module for constraint (on parameter bounds) optimization with user-supplied routines for the first and second-order derivatives of the objective function.

2.6.3. The differential ant-stigmergy algorithm (DASA)

The DASA algorithm was proposed by Korošec et al. (2012). It is a version of an Ant Colony Optimization (ACO) meta-heuristic, designed to successfully cope with high-dimensional continuous optimization problems. The rationale behind the algorithm is in memorizing the move in the search space that improves the current best solution, and using it in further searches. The algorithm uses pheromones as a means of communication between ants (a phenomenon called stigmergy), combined with graph representation of the search space.

The most important property of DASA is that it transforms the problem into a graph-search problem by fine-grained discretization of the continuous domain of the parameters' differences, unlike the common way of discretizing parameters values. The parameters' differences assigned to the graph vertices are used to navigate through the search space.

3. Experimental setup

The aim of our study is to compare the influence of two established methods for parameter estimation, ALG-717 and DASA, on the overall process of automated modeling of aquatic ecosystems. The study addresses the task of modeling phytoplankton dynamics in four different aquatic ecosystems. First, we developed a library for modeling aquatic ecosystems, which we used as domain knowledge. Second, we obtained data from four ecosystems, relevant for modeling phytoplankton dynamics. Finally, we formulated conceptual models for each ecosystem. The resulting experimental setup consisted of 21 tasks of modeling phytoplankton dynamics, where a task is defined by the combination of a conceptual model and a data set. Each task was given to ProBMoT using both ALG-717 and DASA as parameter estimation methods, yielding a total of 42 experiments. The

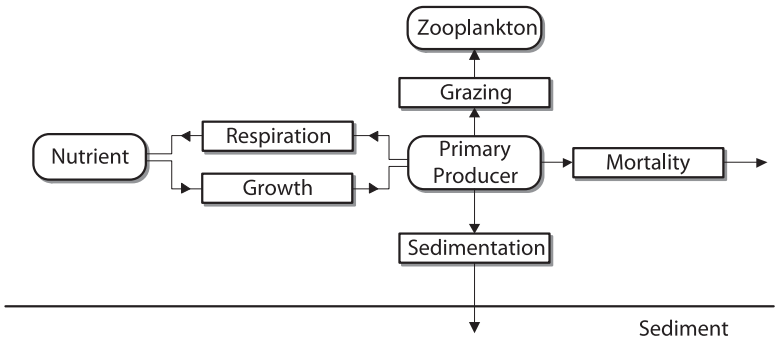


Fig. 4. The generalized scheme of processes for modeling conceptual models, underlying the library of domain knowledge in aquatic ecosystems.

details of the experimental setup are given in the following subsections.

3.1. A library for modeling aquatic ecosystems

Using the formalism for representing domain knowledge presented in Section 2, we developed a library for modeling aquatic ecosystems. The library is based on the work for process-based modeling by Atanasova et al. (2006b), where a similar library was developed using a different formalism. The entire library is given in Table 12 in Appendix A.

The generalized conceptual model for modeling aquatic ecosystems underlying the library is given in Fig. 4. The rounded rectangles represent the entity types and include *Nutrient*, *Primary Producer* and *Zooplankton*. The boxes represent the interactions between the entities in the form of process types. The library is constructed around the primary producer as a central entity. It contains processes suitable for modeling the dynamics of the primary producer. The main processes that affect the dynamics are growth, respiration, mortality, sedimentation and grazing. In the following subsections, we briefly present the modeling knowledge encoded in the library.

3.1.1. The growth of primary producers

The growth process positively influences the concentration of the primary producer and can be stated as:

$$pp.conc' = pp.growthRate \cdot pp.conc \tag{4}$$

where *pp.conc* is the concentration of the primary producer and *pp.growthRate* is the primary producer growth rate. The growth rate itself can be formulated as a limited growth rate, influenced by temperature, light and nutrient limitation functions:

$$pp.growthRate = pp.maxGrowthRate \cdot pp.tempGrowthLim \cdot pp.lightLim \cdot pp.nutrientLim \tag{5}$$

where *pp.maxGrowthRate* is the maximal growth rate under optimal conditions, *pp.tempGrowthLim* is the temperature influence on the growth rate, *pp.lightLim* is the influence of light on the growth rate, and *pp.nutrientLim* is the product of the limitation functions of all nutrients that are relevant for the growth of the primary producer.

Each type of limitation can be modeled with several different limitation functions that are present in the library. The influence of temperature on the growth can be modeled with two linear functions and an exponential function. In addition to this, temperature limitation can be turned off, by setting it to 1, which yields 4 alternatives for modeling the temperature influence. Similarly, light influence can be turned off or modeled with Monod or the optimal light limitation function, whereas nutrient limitation (for each nutrient) can be turned off or modeled with one of the Monod, Monod2 and exponential functions.

If all limitations are 'turned off' (equal to 1), then the *pp.growthRate* coefficient equals to *pp.maxGrowthRate*, i.e., is constant and the growth process is formulated as non limited, i.e., exponential growth function.

3.1.2. Respiration

The respiration of a primary producer decreases its mass, i.e., the phytoplankton mass. It can be expressed as an exponential decay or can be temperature-influenced. In the case where respiration is temperature-influenced, the influence of temperature can be expressed in terms analog to the influence of temperature on phytoplankton growth, i.e., no limitation at all or limitation in the form of one of two linear functions and one exponential limitation function.

3.1.3. Mortality

The mortality process represents non-predatory mortality and is usually included in scenarios where there is no grazing process included. The mortality can be expressed as an exponential decay of phytoplankton (first order kinetics) or second order kinetics. The process may be temperature-limited. The temperature limitation functions in the library include two linear functions and one exponential function.

3.1.4. Grazing

The grazing process included in the library is formulated for zooplankton filter-feeders as:

$$pp.conc' = -zoo.amxFiltrationRate \cdot zoo.tempGrowthLim \cdot zoo.phytoLim \cdot zoo.conc \cdot pp.conc \tag{6}$$

where *zoo.maxFiltrationRate* is the maximal filtration rate coefficient, *zoo.conc* is the zooplankton concentration and *pp.conc* is the concentration of the phytoplankton. The temperature influence on grazing is specified trough the *zoo.tempGrowthLim*, term which contains a linear or an exponential temperature limitation function option.

3.1.5. Sedimentation

Loss of phytoplankton biomass due to sedimentation is formulated by using the sedimentation rate coefficient, depth of the water column and the present concentration of the phytoplankton.

Table 7
Summary of measured data for the modeled ecosystems.

	Bled	Glumsø	Kasumigaura	Venice
Environmental influence	Temperature Light	Temperature Light	Temperature Light	Temperature
Nutrients	Phosphorus Nitrogen Silica	Phosphorus Nitrogen	Phosphorus Nitrogen	Phosphorus Nitrogen Ammonia
Primary producer	Phytoplankton	Phytoplankton	Phytoplankton (as Chl-a)	Algal biomass of *Ulva rigida*
Zooplankton	Zooplankton (*Daphnia hyalina*)	Zooplankton	None	None
Years	1995–2002	1973/1974 1974/1975	1986–1992	Loc. 0: 1985/1986 Loc. 1, 2, 3: 1990/1991
Number of data sets	8	2	7	4

The process may be temperature-influenced, where the temperature functions include two linear functions and one exponential function.

3.2. Data sets

The data used for this study comes from four different ecosystems, namely Lake Bled in Slovenia, Lake Glumsø in Denmark, Lake Kasumigaura in Japan, and the Lagoon of Venice in Italy. For each aquatic ecosystem, a number of physical, chemical and biological parameters were measured regularly for a sustained period of time in order to obtain the resulting data sets. Table 7 provides a summary of the data sets.

Lake Bled is a typical subalpine lake of glacial-tectonic origin. It occupies an area of 1.4 km^2 with a maximum depth of 30.1 m and an average depth of 17.9 m. The data set about the lake (obtained from the Slovenian Environmental Agency) comprises measurements of physical, chemical and biological parameters from 1995 to 2002 with a monthly frequency. The data used for modeling are as follows: temperature, light, dissolved inorganic nutrients in the lake (phosphorus, nitrogen and silica), and total phytoplankton biomass, and the zooplankton species *Daphnia hyalina*. The data were used as daily interpolated values between the measured points, obtained by using cubic spline interpolation (Atanasova et al., 2006c).

Lake Glumsø is situated in a sub-glacial valley in Denmark. It is a shallow lake with an average depth of about 2 m and a surface area of 266,000 m^2. The data set for Lake Glumsø includes daily measurements of water temperature, inorganic soluble nitrogen, soluble phosphorous, total phytoplankton, and zooplankton. In this case, we used two years of daily measurements from April 1973 to April 1974 and from October 1974 to October 1975 (Atanasova et al., 2008).

Lake Kasumigaura is a shallow lake in Japan with an average depth of 4 m. It has a volume of 662 million m^3 and a surface area of 220 km^2. The lake's dataset comprises measurements from 1986 to 1992 of: water temperature, global radiation, dissolved inorganic phosphorus, total phytoplankton, measured as chlorophyll-a (chl-a). The measurements were used as interpolated values between the actual measured values using linear interpolation. The actual frequency of the measurements is monthly (Atanasova et al., 2006a).

The Lagoon of Venice has a surface area of 550 km^2, with an average depth of less than 1 m. The data set used here includes weekly measurements for slightly more than one year at four different locations (0, 1, 2, and 3) in the Lagoon. Location 0 was sampled in 1985/1986, locations 1, 2, and 3 in 1990/1991. The sampled quantities are nitrogen in ammonia (nh), nitrogen in nitrate (no), phosphorus in orthophosphate, temperature, and algal biomass (*Ulva rigida*). Related modeling experiments with this data set were described by Atanasova (2006).

We assemble the data sets according to domains and years. This yields eight data sets from Lake Bled, two from Glumsø, seven from Kasumigaura and four from Venice, giving a total of 21 data sets.

3.3. Automated modeling tasks

In this study, we focus on the task of modeling phytoplankton dynamics using data from the domains presented in Section 3.2. For

Table 8
A summary of the modeling tasks per domain (ecosystem).

	Bled	Glumsø	Kasumigaura	Venice
Entities	Phosphorus Nitrogen Silica Phytoplankton Zooplankton	Phosphorus Nitrogen Phytoplankton Zooplankton	Phosphorus Nitrogen Phytoplankton Zooplankton	Phosphorus Nitrate Ammonia Phytoplankton
Conceptual processes	Growth Respiration Grazing Sedimentation	Growth Respiration Grazing Sedimentation	Growth Respiration Mortality Sedimentation	Growth Respiration Mortality Sedimentation
Candidate models	27216	3024	5832	18144

each domain (ecosystem), we prepare a conceptual model appropriate for modeling that particular case. The conceptual model is crafted according to the format presented in Section 2.4. It includes the entities for which we have measurements and the conceptual top-level processes appropriate for modeling phytoplankton dynamics: growth, respiration, mortality, sedimentation, and grazing by zooplankton. For two of the domains, Bled and Glumsø, there is data about zooplankton concentration. In these cases, we include a grazing process in the conceptual models. For Kasumigaura and Venice, where there is no data about zooplankton species, we include a natural mortality process instead. The conceptual models are outlined in Table 8.

For each domain, we tailor the general library presented in Section 3.1 to the requirements of the particular modeling task. We include processes which are needed for completing the conceptual model of the system. Based on previous analyses and previous knowledge about the domain, specific process alternatives which are appropriate for the given domain are taken into account, whereas unsuitable alternatives are discarded. Information about which processes are included in the pertinent libraries is given in Table 9.

For each modeling task, starting with a conceptual model and a library of domain knowledge, ProBMoT, as explained in Section 2.5, generated all specific candidate models structures. For the Bled domain it generated 27,216 candidate models. The Glumsø task yielded 3024 candidates, the Venice task 5832 and Kasumigaura task 18,144 candidate models. The constant parameters of each generated candidate model structure are fitted with the two parameter estimation approaches described in Section 2.6.

The data from the domains are divided into separate data sets for each year. In the case of Venice, where data were collected at different locations, one data set corresponds to one location where the measurements were taken. In our experiments, the goal is to construct a separate model for each data set. The model is an explanatory model of the system for the given time range.

4. Results and discussion

In this section, we first present the methodology for analyzing the influence of the parameter estimation method on the performance of ProBMoT. The analysis is based on the comparison of the performance of ProBMoT when using the ALG-717 (ProBMoT/ALG) and the DASA (ProBMoT/DASA) optimization algorithms for solving the parameter estimation task. In the second part of the section, we present and discuss the results of the empirical evaluation.

4.1. Evaluation methodology

Each run of ProBMoT generates a series of N candidate model structures and applies parameter estimation method to obtain the models m_i, $i = 1, \ldots, N$. For each candidate model structure ProBMoT calculates its error $RMSE(m_i)$ on the provided data set. The resulting vector of model errors is schematically presented in Fig. 5(a), where the number in the lower left corner of each box represents the sequential number of the model structure (as generated by ProBMoT) and the number in the center of the box represents the model error (for the set of parameter values estimated by ProBMoT). Let us denote the models obtained by ProBMoT/ALG and ProBMoT/DASA with m_i^A and m_i^D, respectively. To evaluate the impact of the optimization method on the ProBMoT performance, we thus compare the vectors $(RMSE(m_1^A), RMSE(m_2^A), \ldots, RMSE(m_N^A))$ and $(RMSE(m_1^D), RMSE(m_2^D), \ldots, RMSE(m_N^D))$ (Fig. 5(b)). We emphasize here three aspects of this comparison, depicted in Fig. 5(c–e), that correspond to the three central questions below.

Best Model Improvement. What is the difference between the errors of the best models obtained with ProBMoT/DASA and ProBMoT/ALG?

Most importantly, we want to compare the best model found by ProBMoT/DASA to the best model found by ProBMoT/ALG as depicted in Fig 5(c). This is the key aspect of the comparison, since the modeler focus his attention to the best model found by ProBMoT. Our hypothesis is that ProBMoT/DASA leads to the best model with lower error. Note that the best models found by ProBMoT/DASA and ProBMoT/ALG can differ not only in the parameter values but also it the model structure.

Overall Model Improvement. What is the overall difference between the errors of the models obtained with ProBMoT/DASA and ProBMoT/ALG?

Comparing the best models is an important aspect of comparison, but only provides a part of the whole picture. Here, we extend our attention from the best models to overall model comparison as depicted in Fig. 5(d). We first rank the models obtained by ProBMoT/ALG and ProBMoT/DASA with respect to their errors, i.e., we obtain two ranked lists of models, such that $RMSE(m_{a_1}^A) \leq RMSE(m_{a_2}^A) \leq \ldots \leq RMSE(m_{a_N}^A)$ and $RMSE(m_{d_1}^D) \leq RMSE(m_{d_2}^D) \leq \ldots \leq RMSE(m_{d_N}^D)$. We proceed with comparison of these two ranked lists as follows.

First, we check the extent to which ProBMoT/DASA outperforms ProBMoT/ALG, i.e., whether only the best ProBMoT/DASA model outperforms the best ProBMoT/ALG model or this holds for a wider range of models. We will be able to identify the ranges of models where ProBMoT/DASA is better and possibly where ProBMoT/ALG is better. Our hypothesis is that ProBMoT/DASA will outperform ProBMoT/ALG on the whole range of models.

Second, it will show whether the amount by which ProBMoT/DASA outperforms ProBMoT/ALG on the ith best model, as measured by the difference $RMSE(m_{a_i}^A) - RMSE(m_{d_i}^D)$, increases or decreases as we move towards lower ranked models. Clearly, lower ranked models will have larger model errors, but we are also interested in the way the model error increases. Slow increase in model error means that we have a number of candidate models which have virtually equal performance and choosing among them is subject to other non-trivial constraints. Very rapid increase in the model error makes the distinction between good and bad models clear and makes choosing a final model an easier task. It is preferable to have a large error increase at the beginning of the ranked list so one can clearly distinguish between a few good and a majority of bad models. Our hypothesis is that ProBMoT/DASA provides a more clear distinction between good and bad models than ProBMoT/ALG.

Finally, we measure of the overall performance of ProBMoT/ALG and ProBMoT/DASA, by summing up the errors of all candidate models:

$$Overall_A = \sum_{i=1}^{N} RMSE(m_i^A), \qquad Overall_D = \sum_{i=1}^{N} RMSE(m_i^D) \tag{7}$$

Individual Model Improvement. What is the difference between the errors of each candidate model structure obtained with ProBMoT/DASA and ProBMoT/ALG?

In the previous type of comparison, the models are ordered according to ascending *RMSE*. When we are comparing the ith best ProBMoT/ALG model $m_{a_i}(p_{a_i}^A)$ to the ith best ProBMoT/DASA model $m_{d_i}(p_{d_i}^D)$, we are in general comparing two different

Table 9
Outline of the process templates modeled in the specific libraries for each modeling task.

Process template	Bled	Glumsø	Kasumigaura	Venice
NutrientPrimaryProducerInteraction	+	+	+	+
LightInfluence	+	+	+	+
NoLightLim	–	–	–	+
LightLim	+	+	+	–
MonodLightLim	+	+	+	–
OptimalLightLim	+	+	+	–
NutrientInfluence	+	+	+	+
NoNutrientLim	–	–	–	–
NutrientLim	+	+	+	+
MonodNutrientLim	+	+	+	+
Monod2NutrientLim	+	+	+	+
ExponentialNutrientLim	+	+	+	+
Growth	+	+	+	+
GrowthRate	+	+	+	+
LimitedGrowthRate	+	+	+	+
TempGrowthInfluence	+	+	+	+
NoTempGrowthLim	–	–	–	–
TempGrowthLim	+	+	+	+
Linear1TempGrowthLim	+	+	+	+
Linear2TempGrowthLim	+	+	+	+
ExponentialTempGrowthLim	+	+	+	+
RespirationPP	+	+	+	+
ExponentialRespirationPP	+	+	+	–
TempRespirationPP	+	+	+	+
Temp1RespirationPP	+	+	+	+
Temp2RespirationPP	+	+	+	+
TempRespInfluence	+	+	+	+
NoTempRespLim	–	–	–	–
TempRespLim	+	+	+	+
Linear1TempRespLim	+	+	+	+
Linear2TempRespLim	+	+	+	+
ExponentialTempRespLim	+	+	+	+
MortalityPP	–	–	+	+
ExponentialMortalityPP	–	–	–	+
TempMortalityPP	–	–	+	+
Temp2MortalityPP	–	–	+	+
TempMortInfluence	–	–	+	+
NoTempMortLim	–	–	–	+
TempMortLim	–	–	+	+
Linear1TempMortLim	–	–	+	+
Linear2TempMortLim	–	–	+	+
ExponentialTempMortLim	–	–	+	+
Sedimentation	+	+	+	+
TempSedInfluence	+	+	+	+
NoTempSedLim	+	+	+	+
TempSedLim	+	+	+	–
Linear1TempSedLim	+	+	+	–
Linear2TempSedLim	+	+	+	–
ExponentialTempSedLim	+	+	+	–
FeedsOn	+	+	–	–
FeedsOnFiltration	+	+	–	–
PhytoLim	+	+	–	–
NoPhytoLim	–	–	–	–
MonodPhytoLim	+	+	–	–
Monod2PhytoLim	+	+	–	–
ExponentialPhytoLim	–	–	–	–

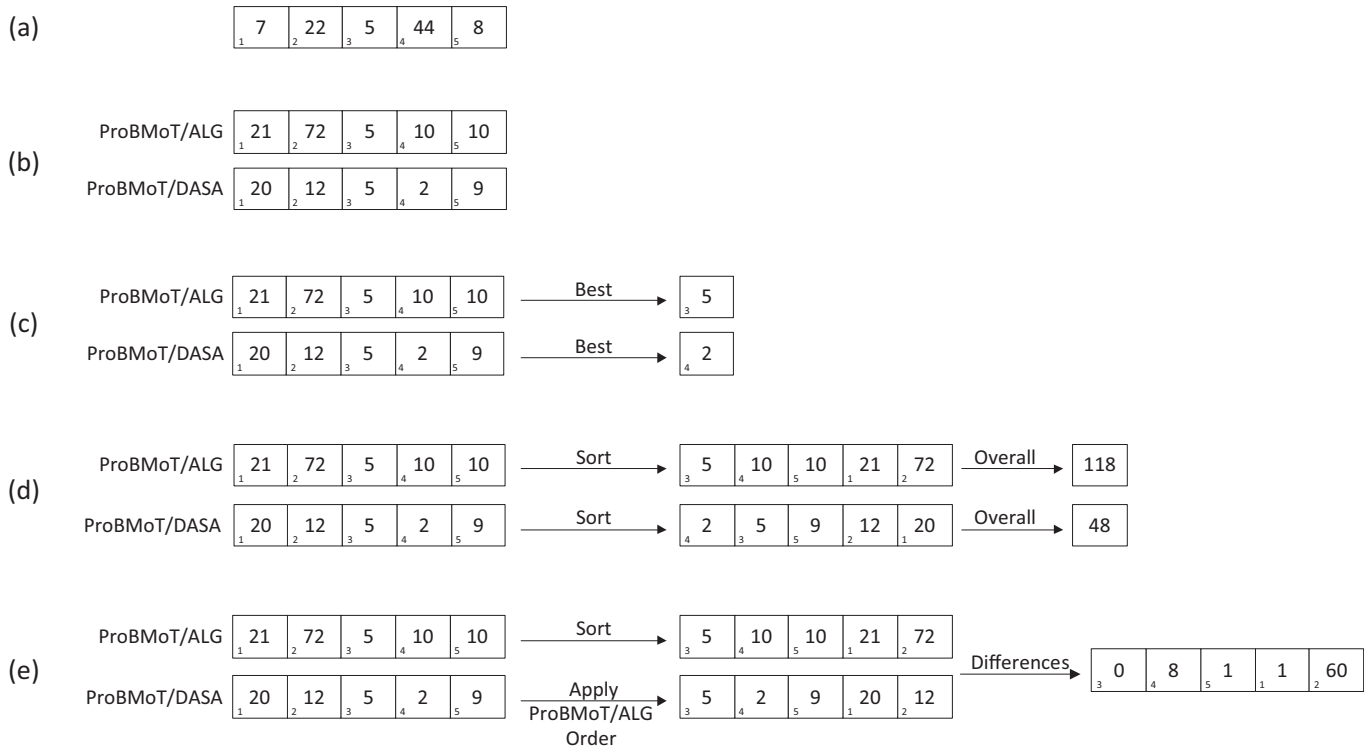


Fig. 5. Schematic representation of the evaluation methodology. (a) Vector of model errors; (b) ProBMoT/ALG and ProBMoT/DASA vectors of model errors; (c) Best Model Improvement; (d) Overall Model Improvement; (e) Individual Model Improvement.

modelstructures, i.e., $a_i \neq d_i$ in the general case. In the final and most stringent comparison, we compare the performance of ProBMoT/DASA and ProBMoT/ALG on the same model structure, i.e., $m_i(p_i^D)$ and $m_i(p_i^A)$, for $i = 1, \ldots, N$.

To perform this comparison, we use the ranking of models obtained by ALG to rank both list of models, i.e.: $RMSE(m^A_{a_1}) \leq RMSE(m^A_{a_2}) \leq \ldots \leq RMSE(m^A_{a_N})$ and $RMSE(m^D_{a_1}), RMSE(m^D_{a_2}), \ldots, RMSE(m^D_{a_N})$ as depicted in Fig. 5(e). We then observe the distribution of differences $RMSE(m^A_{a_i}) - RMSE(m^D_{a_i})$ for $i = 1, \ldots, n$, where the value of n ranges from 2 through 10, 100, 1000 (and 10,000, if $N > 10,000$) to N.

4.2. Best Model Improvement

Table 10 shows the *RMSE* values for the best models found with ProBMoT/ALG and ProBMoT/DASA on all data sets. Since the models selected by ProBMoT/DASA always have a lower *RMSE* than those selected by ProBMoT/ALG, we can conclude that DASA outperforms ALG on each modeling task.

Fig. 6 shows the simulations of the best ProBMoT/ALG and the best ProBMoT/DASA model on some of the tasks, along with the measured data. There is a considerable difference in the errors of the best models found on the Bled '97 task. This quantitative difference in the *RMSE* values is directly visible as a qualitative difference in the simulations. The ALG model simulation exhibits some resemblance to the measured data, whereas the DASA model simulation follows the dynamics of the phytoplankton concentration very closely, both in terms of time of rapid phytoplankton growth and peak amplitude.

The Glumsø '73 task gives similar results. The DASA model represents a close match of the original data, whereas the ALG model only gives a rough approximation, completely missing the main peak of phytoplankton concentration.

Table 10
Root mean squared error (RMSE) of the best models found with ProBMoT/ALG and ProBMoT/DASA.

	Bled								Glumsø	
	'95	'96	'97	'98	'99	'00	'01	'02	'73	'74
ProBMoT/ALG	0.650	0.362	0.800	0.802	1.282	2.439	0.683	0.771	0.099	0.074
ProBMoT/DASA	0.376	0.206	0.157	0.443	1.205	0.821	0.455	0.240	0.034	0.030

	Kasumigaura							Venice			
	'86	'87	'88	'89	'90	'91	'92	L0	L1	L2	L3
ProBMoT/ALG	1.685	0.775	0.674	0.782	0.648	0.819	0.981	138.417	221.619	111.924	55.987
ProBMoT/DASA	1.249	0.646	0.381	0.417	0.350	0.766	0.458	83.431	203.007	88.459	43.085

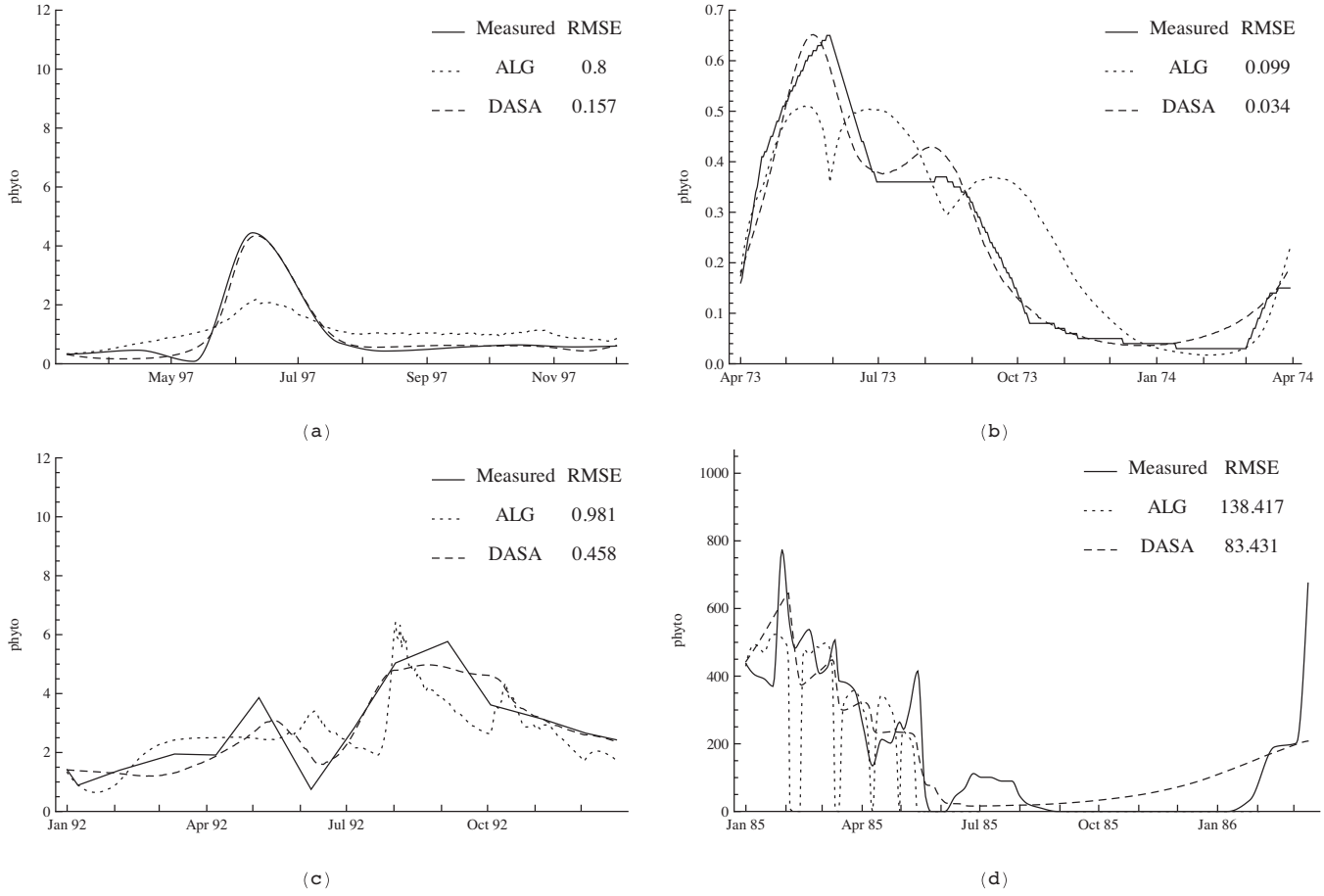


Fig. 6. Simulations of the best models found with ProBMoT/ALG and ProBMoT/DASA on several data sets. (a) Bled '97; (b) Glumsø '73; (c) Kasumigaura '92; (d) Venice 0.

The Kasumigaura '92 task is a more challenging task. On this task, ALG does not manage to find an admissible model which can be seen from the large error and the simulation, which follows the measured data very poorly. DASA does not excel on this task either, but still manages to roughly identify the two peaks in the phytoplankton concentration and their magnitude.

An even poorer performance of ALG is observed on the Venice data set at location 0. The model found by ALG produces erratic behavior and fails to capture any underlying dynamics. The DASA model represents a considerable improvement over ALG, despite of its poor overall quality. The simulations of the best models for all tasks are given in Appendix B. Our hypothesis that the best ProBMoT/DASA model outperforms the best ProBMoT/ALG model on each task is thus fully confirmed.

4.3. Overall Model Improvement

Fig. 7 shows the error profiles for both ALG and DASA on the tasks from Fig. 6. The profile curves on these tasks, as well as all the other tasks (presented in Appendix C) clearly show that DASA manages to find better models throughout the space of model structures. In other words, not only the best model found with ProBMoT/DASA is better than the best one found with ProBMoT/ALG, but also the second best model found with ProBMoT/DASA is better than the second best model found with ProBMoT/ALG, the third best model from ProBMoT/DASA is better than the third best model from ProBMoT/ALG, and so forth. The only exceptions are the few models at the very tail of the ranked list of models. In these cases, both ALG and DASA find poor parameter values, but in some cases the models with parameter values found with DASA have larger *RMSE* than those with parameters found with ALG. Our hypothesis that ProBMoT/DASA outperforms ProBMoT/ALG, not only on the best models for each task, but on the complete range of models (with the exception of the very few models at the end) is thus confirmed.

Table 11 presents the overall error of ProBMoT/ALG and ProBMoT/DASA on each task. It is clear that the total error of ProBMoT/DASA is smaller than the total error of ProBMoT/ALG on each task.

A closer inspection of the error profiles shows that the shapes of the profiles of ProBMoT/ALG and ProBMoT/DASA are very different. In general, the *RMSE* of the ProBMoT/ALG models increases rapidly with the increase in the rank number at the beginning of the error profile and remains steady from there on. Thus, ProBMoT/ALG clearly distinguishes a small number of good models from a large number of bad models. The opposite is true for the shape of the ProBMoT/DASA profile. A very large part of the models that appear at the beginning of the profile have indistinguishably similar *RMSE* values and a much smaller part of the models at the end have much larger *RMSE* values. Thus, DASA does not provide a useful distinction of good and bad models. This is contrary to our initial expectations, and our hypothesis that DASA will provide a more clear distinction of good and bad models is rejected. Within the conclusion

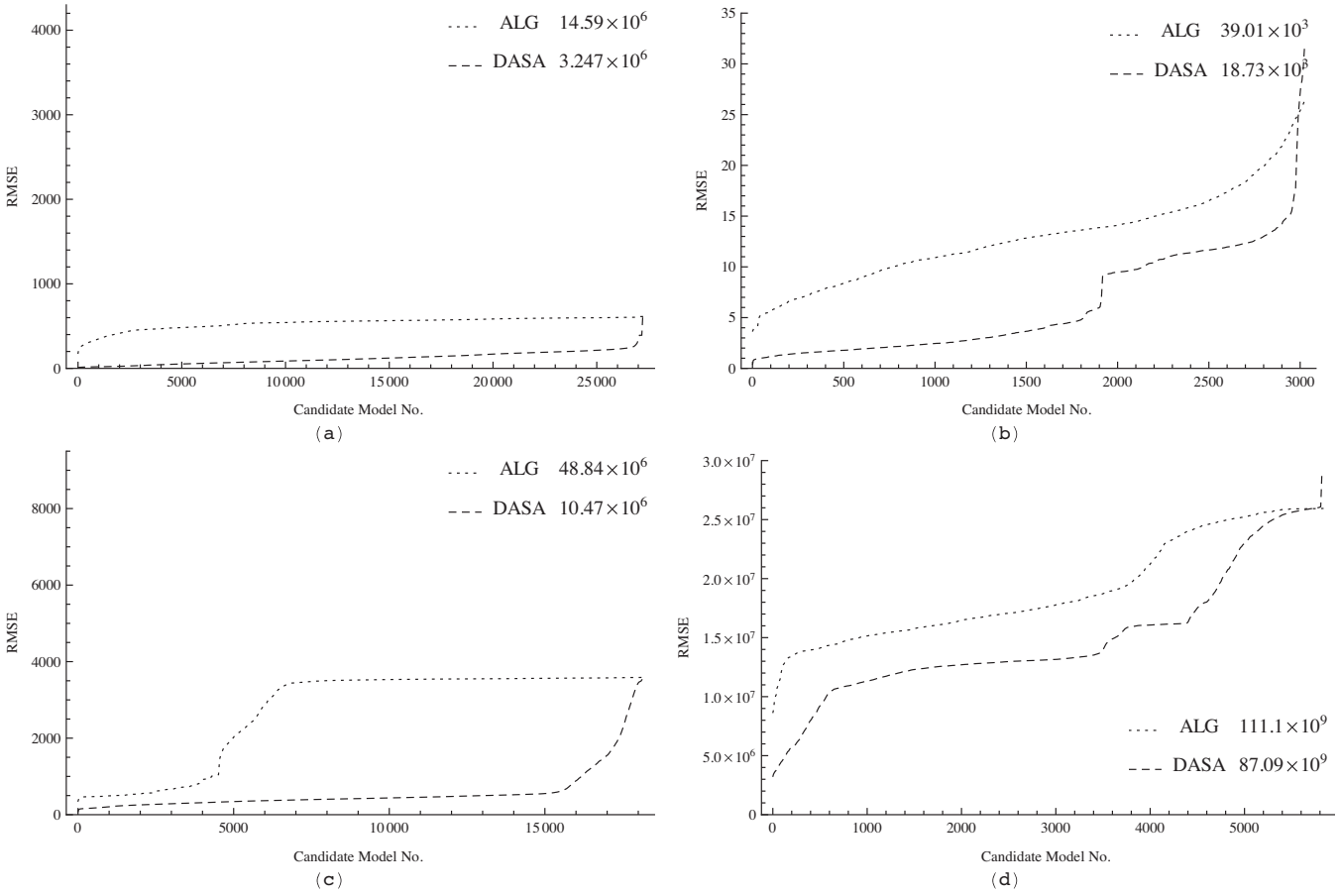


Fig. 7. Error profile curves of models found with ProBMoT/ALG and ProBMoT/DASA on several data sets. (a) Bled '97; (b) Glumsø '73; (c) Kasumigaura '92; (d) Venice 0.

Table 11
Total root mean squared error (total-RMSE) of all models generated with ProBMoT/ALG and ProBMoT/DASA.

	Bled ($\times 10^6$)								Glumsø ($\times 10^3$)	
	'95	'96	'97	'98	'99	'00	'01	'02	'73	'74
ProBMoT/ALG	23.44	22.48	14.59	27.01	107.6	46.64	39.32	29.89	39.01	38.07
ProBMoT/DASA	3.802	7.442	3.247	7.058	9.555	7.802	15.05	1.915	18.73	13.79

	Kasumigaura ($\times 10^6$)							Venice ($\times 10^9$)			
	'86	'87	'88	'89	'90	'91	'92	L0	L1	L2	L3
ProBMoT/ALG	128.1	42.52	36.74	30.75	39.6	30.26	48.84	111.1	152.0	108.4	9.234
ProBMoT/DASA	58.54	5.836	11.78	17.86	7.295	13.58	10.47	87.09	138.2	73.17	7.608

and further work, we examine the possible reasons for this outcome and formulate a new hypothesis related to this behavior of ProBMoT/DASA.

4.4. Individual Model Improvement

Fig. 8 shows the distributions of error differences between models fitted with DASA and with ALG for the tasks from Fig. 6. The first box-plot in each figure represents a summary of the differences of the two best models found with ALG. If these differences are above zero, DASA does not miss the two best solutions found by ALG. Next, the distributions of differences for the 10, 100, 1000 and 10,000 (if applicable) best models are shown. Lastly, the distribution of differences for all models is shown. In all cases, there is a clear shift of the distribution away from zero, which indicates that DASA overall finds better parameter values than ALG for the same model structures. Moreover, in the vast majority of cases, DASA manages to find better parameter values for the model structures for which ALG finds the best values. Notable exceptions are Kasumigaura '86 and Venice 3 where DASA does not manage to find better parameter values than ALG on several best structures identified by ALG. The distributions of error differences for all tasks are given in Appendix D.

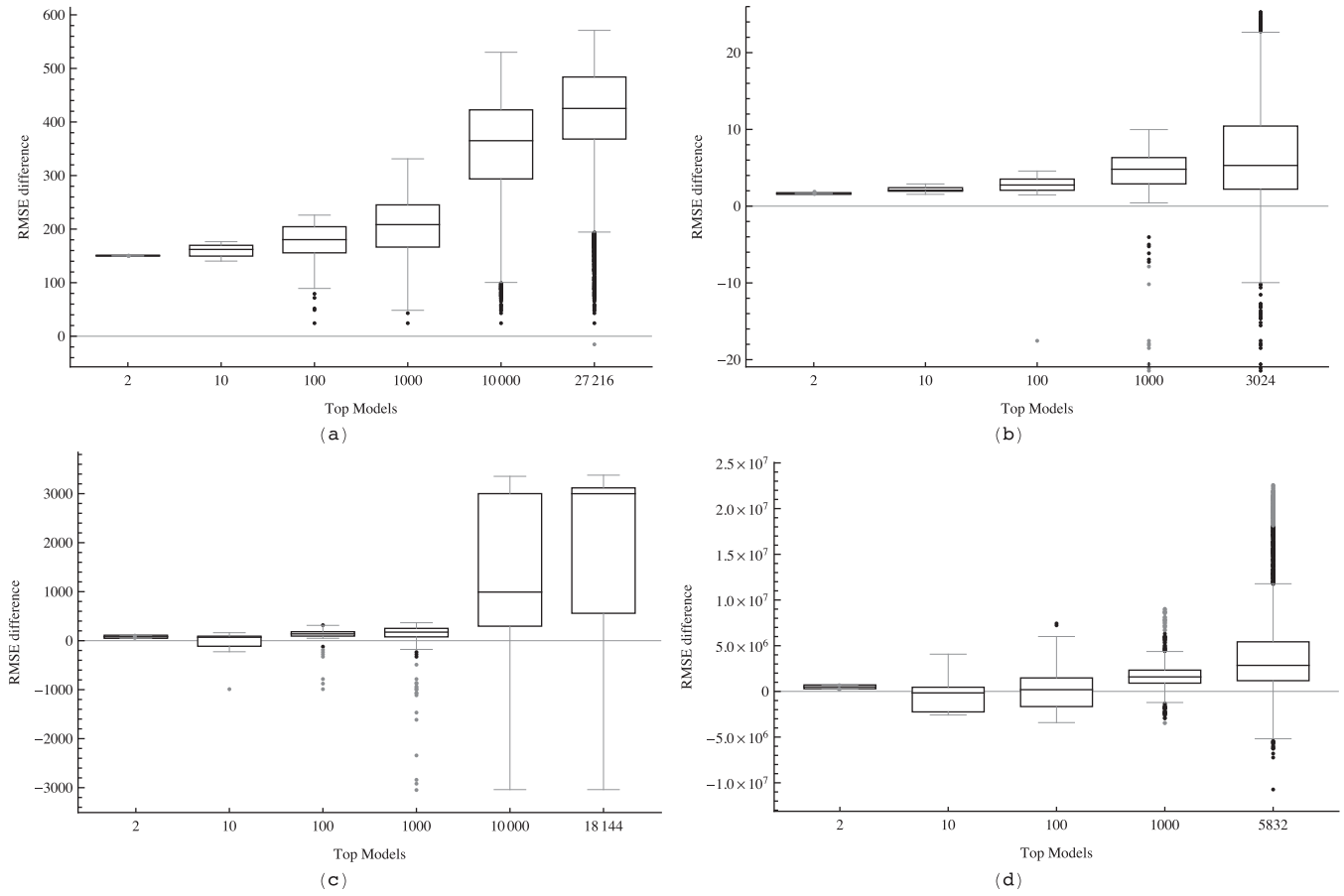


Fig. 8. Distributions of the model-wise differences of errors between ProBMoT/ALG and ProBMoT/DASA. (a) Bled '97; (b) Glumsø '73; (c) Kasumigaura '92; (d) Venice 0.

5. Conclusion and further work

In this paper, we presented ProBMoT, a tool for automated modeling of dynamical systems. ProBMoT uses process-based models that consist of entities and processes for representing dynamical systems. It integrates domain knowledge in the form of a library of entity and process templates. Using a conceptual model of the system, the library of domain knowledge and data for a particular system, it identifies both the structure and numerical parameters of a model for the system.

One major drawback of existing automated modeling approaches is the use of local search methods for parameter estimation. We investigated the effects of substituting DASA, a global search method for the previously used local search method ALG-717. For this purpose, we devised an extensive experimental setup which addressed 21 modeling tasks for four different aquatic ecosystems.

The results conclusively show that DASA outperforms ALG-717 on all modeling tasks. Not only ProBMoT using DASA manages to find better models for the systems, DASA manages to find better parameter values across the whole spectrum of model structures. Furthermore, refitting the best model structure found by ALG with DASA yields better results in almost all cases.

Automated modeling has so far focused mostly on discovering single year models of aquatic ecosystems and this is the approach taken in the present paper. The reason for this is that experiments with discovering models which hold for longer periods of time yielded poor results in earlier work. We conjecture that this is largely due to the poor parameter estimation when using local search methods. Using global search methods opens the opportunity to tackle long term modeling of dynamical systems with automated modeling tools.

One clue to support this lays in the error profile curves in Appendix C. Almost without exception, ALG manages to find good parameter values for very few model structures which can be seen by the shape of the error profile curve which increases rapidly at the beginning and reaches a plateau of models with high error values afterwards. The error profile curves of DASA instead show a plateau of models with low error values which are equally good, and poor parameter values for very few model structures which can be seen by the increase in error values near the end of the curve. This means that the small data sets used do not provide enough information to discriminate among the different model structures. Hence, we need to use longer, multi-year data sets, to narrow down the choice of model structures.

In further work, we will explore the avenues indicated above. We will apply ProBMoT/DASA to construct long term models of the dynamics of the aquatic ecosystems. We will also consider the use of other global methods for parameter estimation, such as differential evolution and particle swarm optimization.

One useful extension of ProBMoT would be automatic recommendation of suitable data sets from large data collections with ecosystem measurements. The type of data required by ProBMoT is directly dependent on the conceptual model that is used. The conceptual model specifies the set of entities that appear in the system. All exogenous variables in the entities need to be mapped to variables in the data set. Moreover, the entities in the conceptual model specify one or more system variables. The observed system variables also have to be mapped to variables in the data set. How

closely these observations are matched by the model predictions, as measured by the objective functions, is what guides the parameter estimation method to good solutions. The mappings of exogenous and observed system variables to variables in the data set are manually provided by the modeler. In order to automatically infer these mappings, the data collection should be annotated with the hierarchy of entity templates. In particular, the variables in the data sets should be annotated with variables from the entity templates in the library. Thus, a software tool, given a conceptual model can determine the data sets which contain all required variables in order to run ProBMoT. The modeler can then choose the most appropriate data set from this list of candidate data sets.

Appendix A. Aquatic ecosystems library

The complete knowledge library for aquatic ecosystems used in the experiments is presented in Table 12 .

Appendix B. Simulations of the best models

The simulations of the best models found by ProBMoT/ALG and ProBMoT/DASA are presented in Figs. 9–12 .

Fig. 9 presents the simulations of the best models found on the Bled domain.

Fig. 10 presents the simulations of the best models found on the Glumsø domain.

Fig. 11 presents the simulations of the best models found on the Kasumigaura domain.

Fig. 12 presents the simulations of the best models found on the Venice domain.

Appendix C. Error profiles

The error profiles of the models found by ProBMoT/ALG and ProBMoT/DASA are presented in Figs. 13–16.

Fig. 13 presents the error profiles of the models found on the Bled domain.

Fig. 14 presents the error profiles of the models found on the Glumsø domain.

Fig. 15 presents the error profiles of the models found on the Kasumigaura domain.

Fig. 16 presents the error profiles of the models found on the Venice domain.

Appendix D. Distributions of the model-wise differences

The distributions of the model-wise differences between ProBMoT/ALG and ProBMoT/DASA are presented in Figs. 17–20.

Fig. 17 presents the distributions of the model-wise differences on the Bled domain.

Fig. 18 presents the distributions of the model-wise differences on the Glumsø domain.

Fig. 19 presents the distributions of the model-wise differences on the Kasumigaura domain.

Fig. 20 presents the distributions of the model-wise differences on the Venice domain.

Table 12
The complete library of the domain knowledge for modeling aquatic ecosystems used for ProBMoT.

```
// ENTITIES

template entity EcosystemEntity {
  vars :
    conc {aggregation:sum; unit:"kg/m^3"; range:<0,inf>};
}

template entity Population : EcosystemEntity {
  vars:
    tempGrowthLim{aggregation:product},
    tempRespLim{aggregation:product},
    tempMortLim{aggregation:product},
    tempExcLim{aggregation:product},
    tempSedLim{aggregation:product};
}

template entity PrimaryProducer : Population {
  vars:
    nutrientLim{aggregation:product},
    lightLim{aggregation:product},
    growthRate;
  consts:
    maxGrowthRate { range: <0.05,3>; unit:"1/(day)"};
}

template entity Zooplankton : Population {
  vars:
    phytoLim{aggregation:sum},
    phytoSum{aggregation:sum};
  consts:
    maxFiltrationRate { range: <0.01, 15>; unit:"m3/(mgZoo*day)"},
    assimilationCoeff { range: <0,inf>; unit:"mgZoo/(mgAlgae)"};
}

template entity Nutrient : EcosystemEntity {
  consts:
    halfSaturation {range: <0,15>; unit:"mg/l"},
    alpha {range: <0,inf>; unit:"mgAlgaeBiomass/mgZooBiomass"};
}

template entity Environment {
  vars:
    temperature,light,flow;
  consts:
    volume,depth,area;
}

// PROCESSES

template process NutrientPrimaryProducerInteraction
(pp : PrimaryProducer, ns : Nutrient<1, inf>, env : Environment ) {
  processes:
    LightInfluence(pp, env), NutrientInfluence(pp, <n:ns>), Growth(pp, ns, env),
    RespirationPP(pp, ns, env);
}

// Temperature Growth Influence

template process TempGrowthInfluence(pop : Population, env : Environment) {}

template process NoTempGrowthLim : TempGrowthInfluence {
  equations:
    pop.tempGrowthLim = 1;
}

template process TempGrowthLim : TempGrowthInfluence {
  consts:
    refTemp { range: <10, 22>},
    minTemp { range: <0, 6>},
    optTemp { range: <15, 25>};
}

template process Linear1TempGrowthLim : TempGrowthLim {
  equations:
    pop.tempGrowthLim = env.temperature/refTemp;
}

template process Linear2TempGrowthLim : TempGrowthLim {
  equations:
    pop.tempGrowthLim = (env.temperature - minTemp)/(refTemp - minTemp);
}

template process ExponentialTempGrowthLim : TempGrowthLim {
  consts:
    theta { range: <1.06, 1.13>};
  equations:
    pop.tempGrowthLim = pow(theta, env.temperature - refTemp);
}

// Temperature Respiration Influence

template process TempRespInfluence(pop : Population, env : Environment) {}
```

Table 12 (*Continued*)

```
template process NoTempRespLim : TempRespInfluence {
  equations:
    pop.tempRespLim = 1;
}

template process TempRespLim : TempRespInfluence {
  consts:
    refTemp { range: <10, 22>},
    minTemp { range: <0, 6>},
    optTemp { range: <15, 25>};
}

template process Linear1TempRespLim : TempRespLim {
  equations:
    pop.tempRespLim = env.temperature/refTemp;
}

template process Linear2TempRespLim : TempRespLim {
  equations:
    pop.tempRespLim = (env.temperature - minTemp)/(refTemp - minTemp);
}

template process ExponentialTempRespLim : TempRespLim {
  consts:
    theta { range: <1.06, 1.13>};
  equations:
    pop.tempRespLim = pow(theta, env.temperature - refTemp);
}

// Temperature Mortality Influence

template process TempMortInfluence(pop : Population, env : Environment) {}

template process NoTempMortLim : TempMortInfluence {
  equations:
    pop.tempRespLim = 1;
}

template process TempMortLim : TempMortInfluence {
  consts:
    refTemp { range: <10, 22>},
    minTemp { range: <0, 6>},
    optTemp { range: <15, 25>};
}

template process Linear1TempMortLim : TempMortLim {
  equations:
    pop.tempRespLim = env.temperature/refTemp;
}

template process Linear2TempMortLim : TempMortLim {
  equations:
    pop.tempRespLim = (env.temperature - minTemp)/(refTemp - minTemp);
}

template process ExponentialTempMortLim : TempMortLim {
  consts:
    theta { range: <1.06, 1.13>};
  equations:
    pop.tempRespLim = pow(theta, env.temperature - refTemp);
}

// Temperature Sedimentation Influence

template process TempSedInfluence(pop : Population, env : Environment) {}

template process NoTempSedLim : TempSedInfluence {
  equations:
    pop.tempSedLim = 1;
}

template process TempSedLim : TempSedInfluence {
  consts:
    refTemp { range: <10, 22>},
    minTemp { range: <0, 6>},
    optTemp { range: <15, 25>};
}

template process Linear1TempSedLim : TempSedLim {
  equations:
    pop.tempSedLim = env.temperature/refTemp;
}

template process Linear2TempSedLim : TempSedLim {
  equations:
    pop.tempSedLim = (env.temperature - minTemp)/(refTemp - minTemp);
}

template process ExponentialTempSedLim : TempSedLim {
  consts:
    theta { range: <1.06, 1.13>};
  equations:
    pop.tempSedLim = pow(theta, env.temperature - refTemp);
}
```

Table 12 (*Continued*)

```
// Light Influence

template process LightInfluence(pp: PrimaryProducer, env: Environment) {}

template process NoLightLim : LightInfluence {
  equations:
    pp.lightLim = 1;
}

template process LightLim : LightInfluence {}

template process MonodLightLim : LightLim {
  consts:
    halfSat {range: <0, 200>};
  equations:
    pp.lightLim = env.light / (env.light + halfSat);
}

template process OptimalLightLim : LightLim {
  consts:
    optLight {range: <100, 200>};
  equations:
    pp.lightLim = env.light * exp(- env.light / optLight + 1) / optLight;
}

// Nutrient Influence

template process NutrientInfluence(pp : PrimaryProducer, n : Nutrient) {}

template process NoNutrientLim : NutrientInfluence {
  equations:
    pp.nutrientLim = 1;
}

template process NutrientLim : NutrientInfluence {}

template process MonodNutrientLim : NutrientLim {
  equations:
    pp.nutrientLim = n.conc / (n.conc + n.halfSaturation);
}

template process Monod2NutrientLim : NutrientLim {
  equations:
    pp.nutrientLim = n.conc * n.conc / (n.conc * n.conc + n.halfSaturation);
}

template process ExponentialNutrientLim : NutrientLim {
  consts:
    saturationRate { range: <0, 15>};
  equations:
    pp.nutrientLim = 1 - exp(-saturationRate * n.conc);
}

// Growth

template process Growth(pp : PrimaryProducer, ns : Nutrient<1, inf>, env : Environment) {
  processes:
    TempGrowthInfluence(pp, env), GrowthRate(pp, ns, env);
  equations:
    td(pp.conc) = pp.growthRate * pp.conc,
    td(<n:ns>.conc) = -n.alpha * pp.growthRate * pp.conc;
}

template process GrowthRate(pp : PrimaryProducer, ns : Nutrient<1, inf>, env: Environment) {}

template process LimitedGrowthRate : GrowthRate {
  equations :
    pp.growthRate = pp.maxGrowthRate * pp.tempGrowthLim * pp.lightLim * pp.nutrientLim;
}

// Respiration PP

template process RespirationPP(pp : PrimaryProducer, ns : Nutrient<1, inf>, env: Environment) {}

template process ExponentialRespirationPP : RespirationPP {
  consts:
    respRate {range: <0.0001, 2>};
  equations:
    td(pp.conc) = -respRate * pp.conc,
    td(<n:ns>.conc) = respRate * pp.conc;
}

template process TempRespirationPP : RespirationPP {
  processes:
    TempRespInfluence(pp, env);
}

template process Temp1RespirationPP : TempRespirationPP {
  consts:
    respRate {range: <0.0001, 1>};
  equations:
    td(pp.conc) = -respRate * pp.conc * pp.tempRespLim,
    td(<n:ns>.conc) = respRate * pp.conc * pp.tempRespLim;
}
```

Table 12 (*Continued*)

```
template process Temp2RespirationPP : TempRespirationPP {
  consts:
    respRate {range: <0.0001, 1>};
  equations:
    td(pp.conc) = -respRate * pp.conc * pp.conc * pp.tempRespLim,
    td(<n:ns>.conc) = respRate * pp.conc * pp.conc * pp.tempRespLim;

}

// Mortality PP

template process MortalityPP(pp : PrimaryProducer, env : Environment) {
  processes:
    TempMortInfluence(pp, env);
}

template process ExponentialMortalityPP : MortalityPP {
  consts:
    mortRate {range: <0.0001, 2>};
  equations:
    td(pp.conc) = -mortRate * pp.conc;
}

template process TempMortalityPP : MortalityPP {
  consts:
    mortRate {range: <0.0001, 2>};
  equations:
    td(pp.conc) = -mortRate * pp.conc * pp.tempMortLim;
}

template process Temp2MortalityPP : MortalityPP {
  consts:
    mortRate {range: <0.0001, 2>};
  equations:
    td(pp.conc) = -mortRate * pp.conc * pp.conc * pp.tempMortLim;
}

// Feeds On

template process FeedsOn(zoo:Zooplankton, pps:PrimaryProducer<1,inf>, env: Environment){
  processes :
    TempGrowthInfluence(zoo, env), PhytoLim(zoo, pps);
}

template process FeedsOnFiltration: FeedsOn{
  equations :
    td(zoo.conc) = zoo.assimilationCoeff * zoo.maxFiltrationRate * zoo.tempGrowthLim
    * zoo.conc * zoo.phytoSum * zoo.phytoLim,
    td(<pp:pps>.conc) = - zoo.maxFiltrationRate * zoo.tempGrowthLim * zoo.conc
    * pp.conc * zoo.phytoLim;
}

template process PhytoLim(zoo: Zooplankton, pps : PrimaryProducer<1,inf>) {}

template process NoPhytoLim : PhytoLim {
  equations:
    zoo.phytoLim = 1;
}

template process MonodPhytoLim : PhytoLim {
  consts:
    halfSaturation {range: <0, 20> };
  processes:
    Summation(zoo, pps);
  equations:
    zoo.phytoLim = zoo.phytoSum/ (halfSaturation+ zoo.phytoSum);
}

template process Monod2PhytoLim : PhytoLim {
  consts:
    halfSaturation {range: <0, 20> };
  processes:
    Summation(zoo, pps);
  equations:
    zoo.phytoLim = zoo.phytoSum * zoo.phytoSum / (zoo.phytoSum * zoo.phytoSum + halfSaturation);
}

template process ExponentialPhytoLim : PhytoLim {
  consts:
    saturationRate {range: <0, 5> };
  processes:
    Summation(zoo, pps);
  equations:
    zoo.phytoLim = 1 - exp(-saturationRate * zoo.phytoSum);
}

template process Summation(zoo : Zooplankton, pps: PrimaryProducer<1,inf>) {
  equations:
    zoo.phytoSum = <pp:pps>.conc;
}

template process Sedimentation(pop : Population, env: Environment) {
  processes:
    TempSedInfluence(pop, env);
  consts:
    sedimentationRate { range: <0.0001, 0.5>; unit:"1/(day)"};

  equations:
    td(pop.conc) = -(sedimentationRate / env.depth) * pop.conc * pop.tempSedLim;
}
```

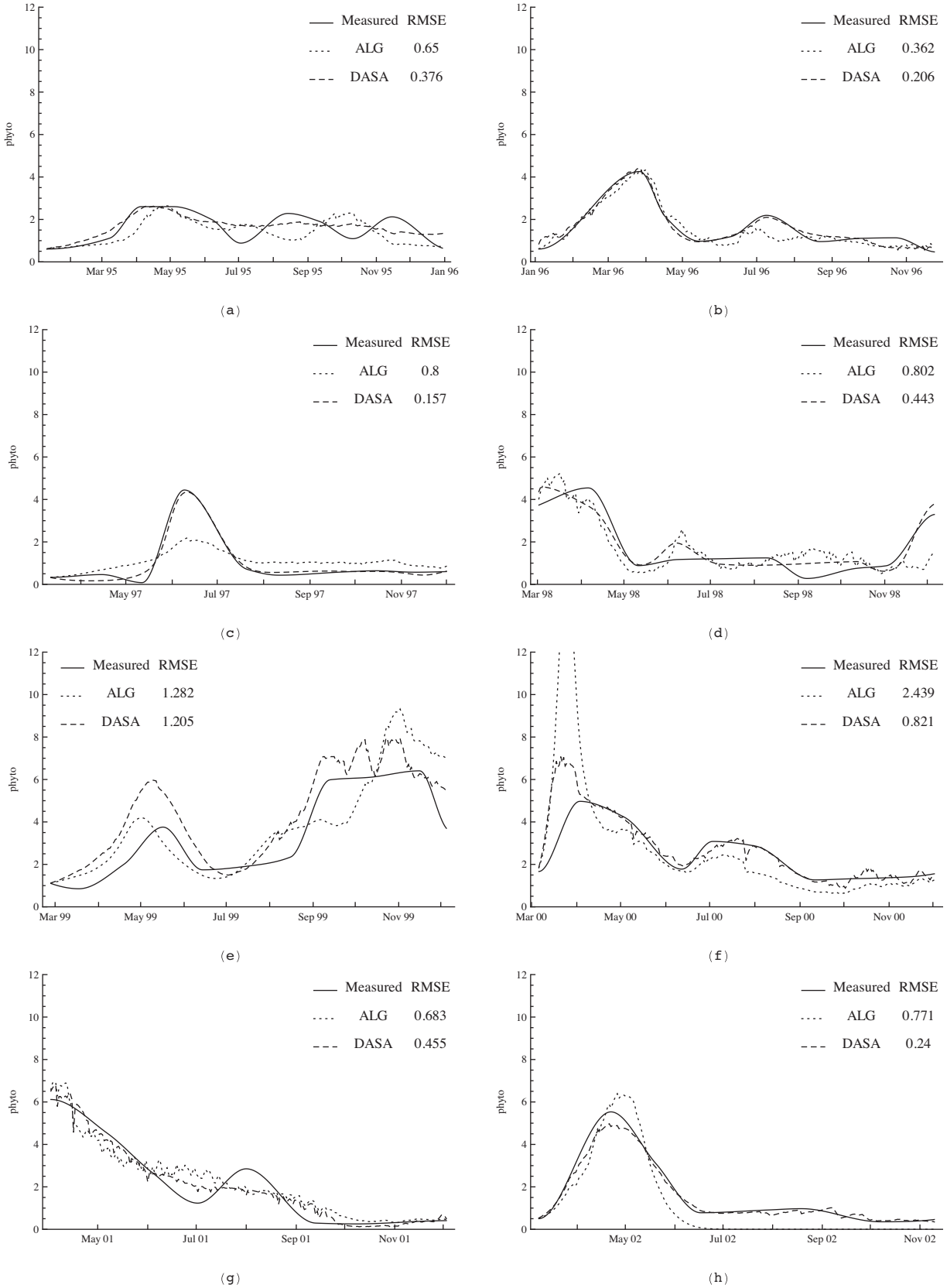


Fig. 9. Simulations of the best models found by ProBMoT/ALG and ProBMoT/DASA for each task for the Bled domain. (a) Bled '95; (b) Bled '96; (c) Bled '97; (d) Bled '98; (e) Bled '99; (f) Bled '00; (g) Bled '01; (h) Bled '02.

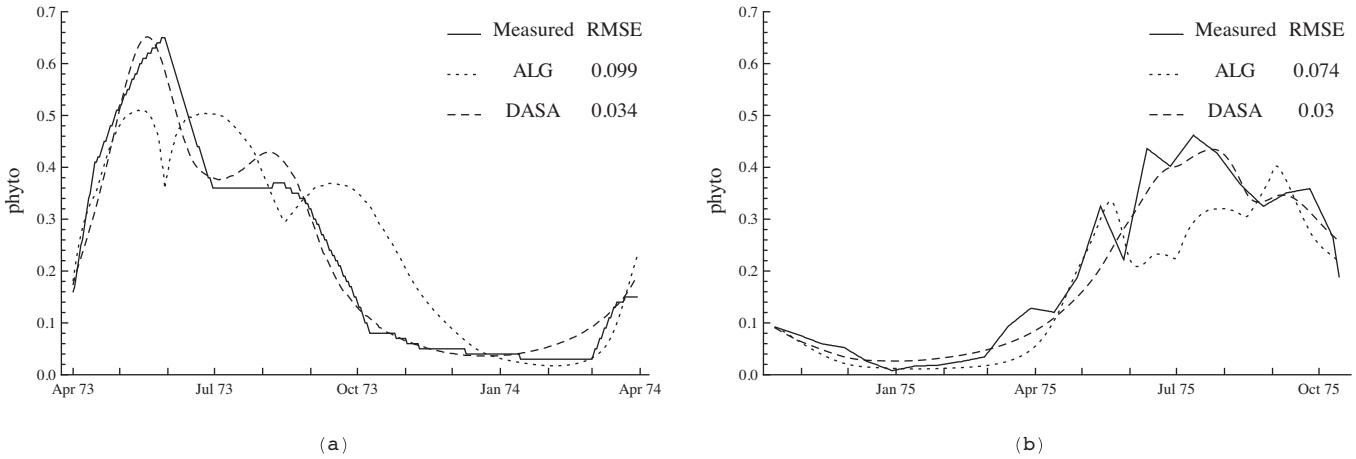


Fig. 10. Simulations of the best models found by ProBMoT/ALG and ProBMoT/DASA for each task for the Glumsø domain. (a) Glumsø '73 and (b) Glumsø '74.

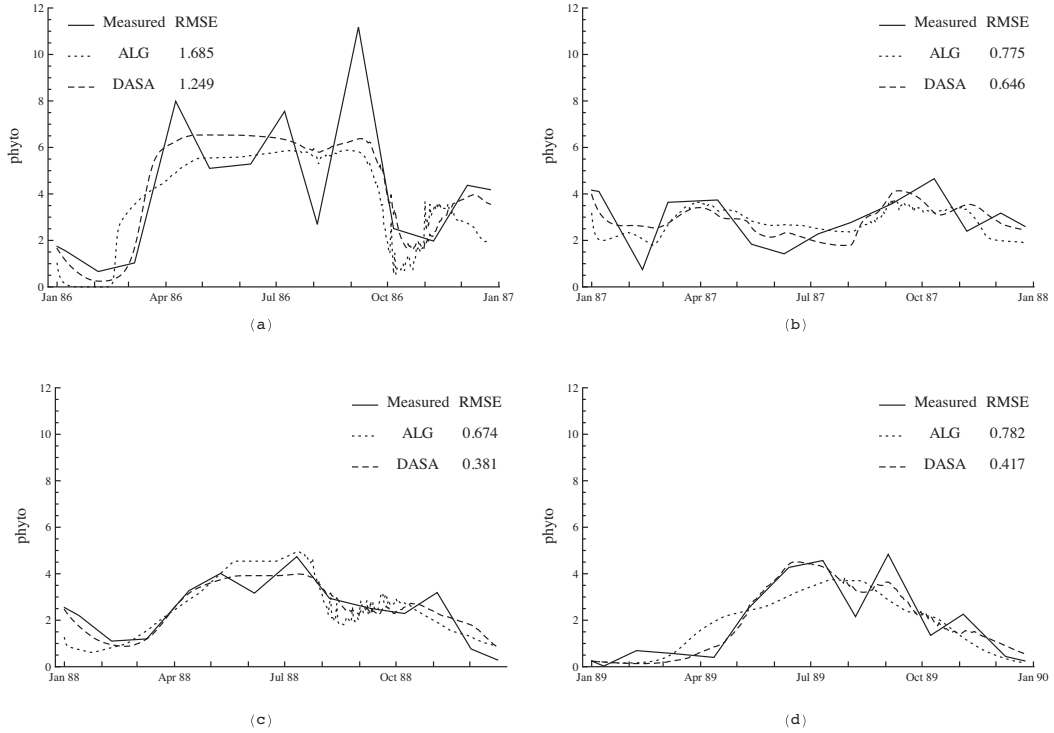


Fig. 11. Simulations of the best models found by ProBMoT/ALG and ProBMoT/DASA for each task for the Kasumigaura domain. (a) Kasumigaura '86; (b) Kasumigaura '87; (c) Kasumigaura '88; (d) Kasumigaura '89; (e) Kasumigaura '90; (f) Kasumigaura '91; (g) Kasumigaura '92.

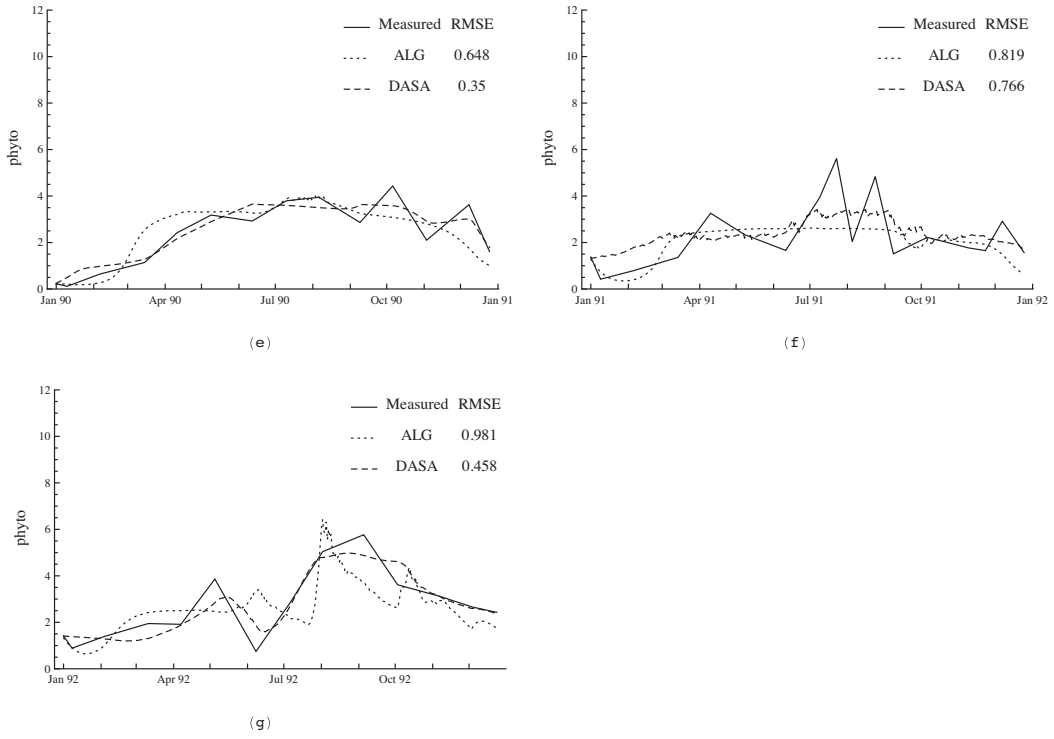


Fig. 11. (*Continued*).

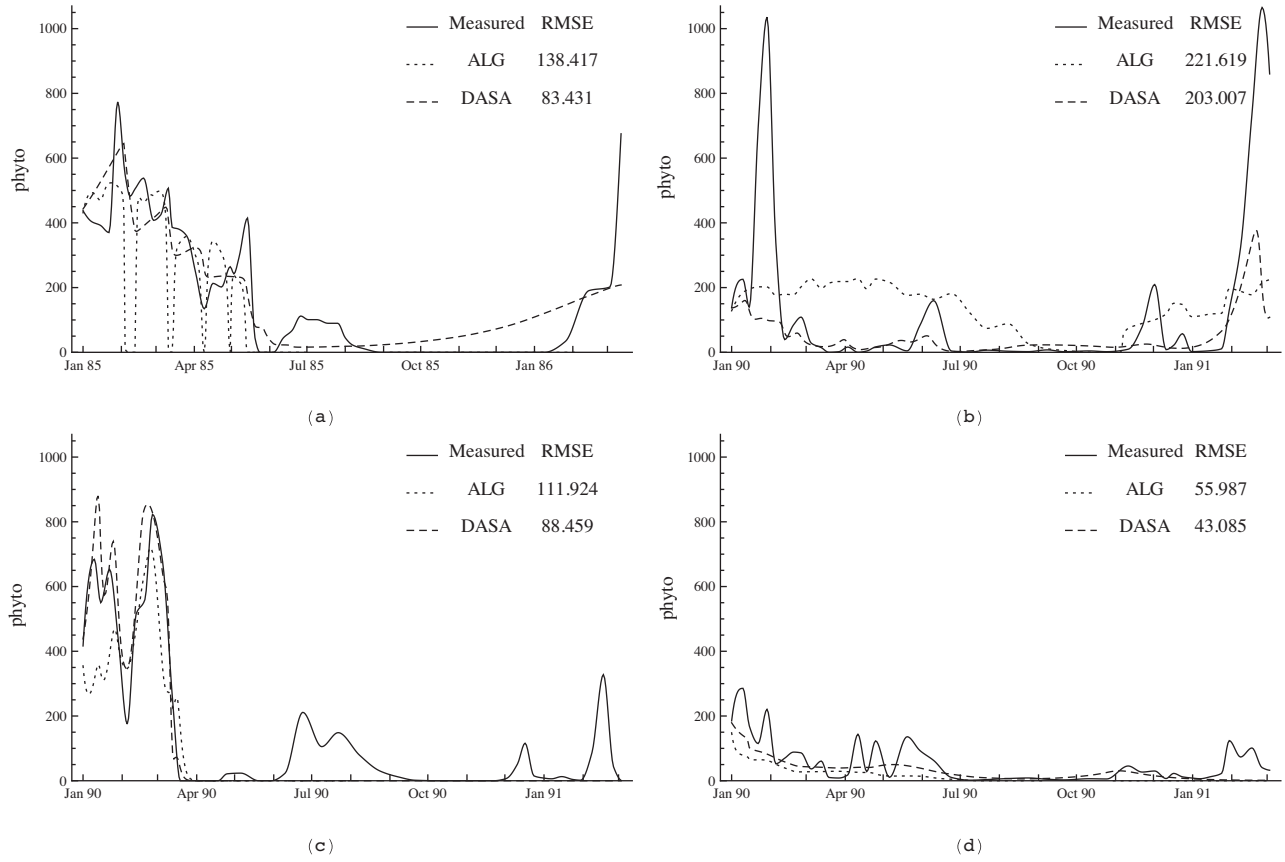


Fig. 12. Simulations of the best models found by ProBMoT/ALG and ProBMoT/DASA for each task for the Venice domain. (a) Venice Loc. 0; (b) Venice Loc. 1; (c) Venice Loc. 2; (d) Venice Loc. 3.

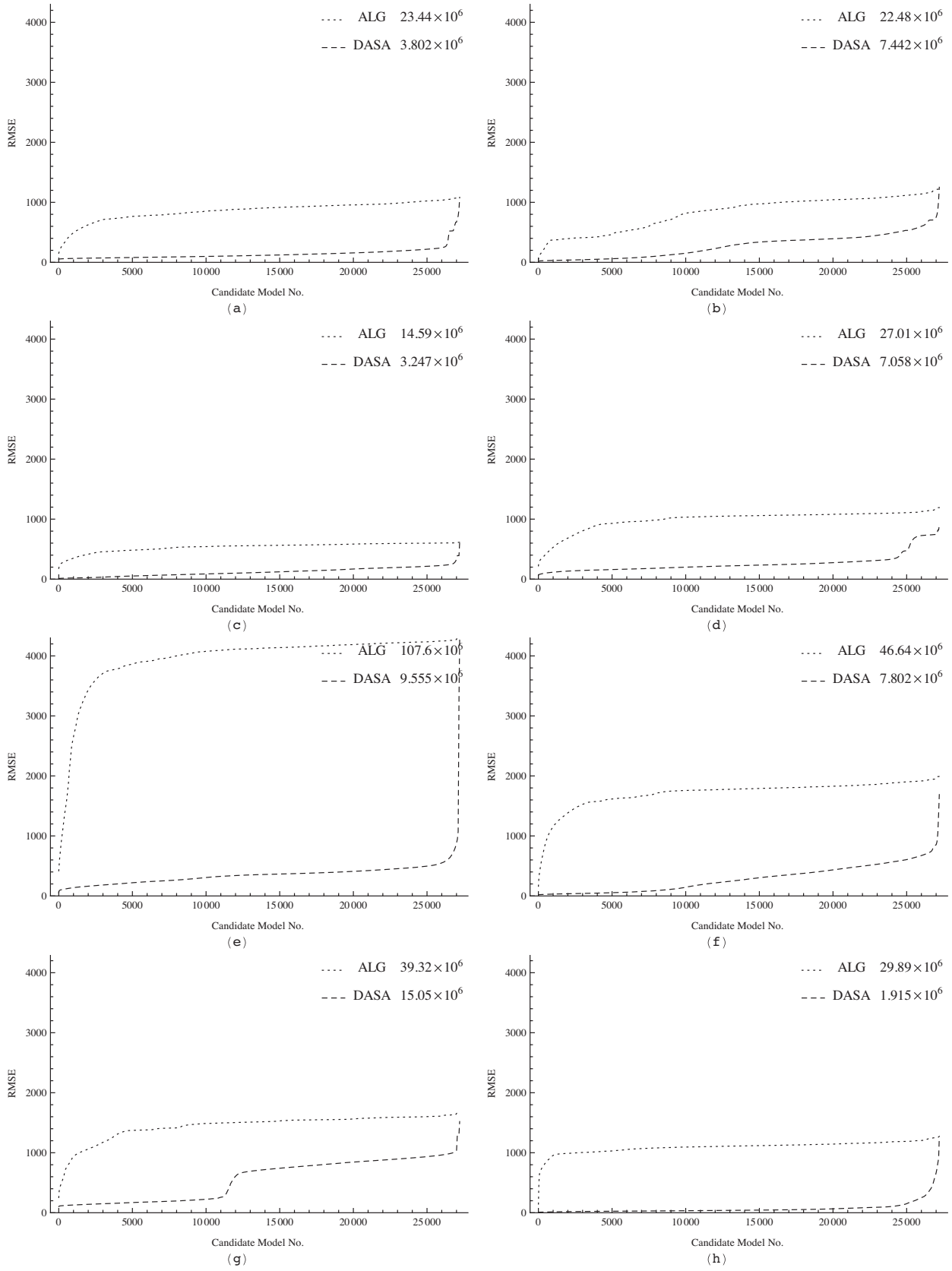


Fig. 13. Error profiles for ProBMoT/ALG and ProBMoT/DASA on all automated modeling tasks for the Bled domain. (a) Bled '95; (b) Bled '96; (c) Bled '97; (d) Bled '98; (e) Bled '99; (f) Bled '00; (g) Bled '01; (h) Bled '02.

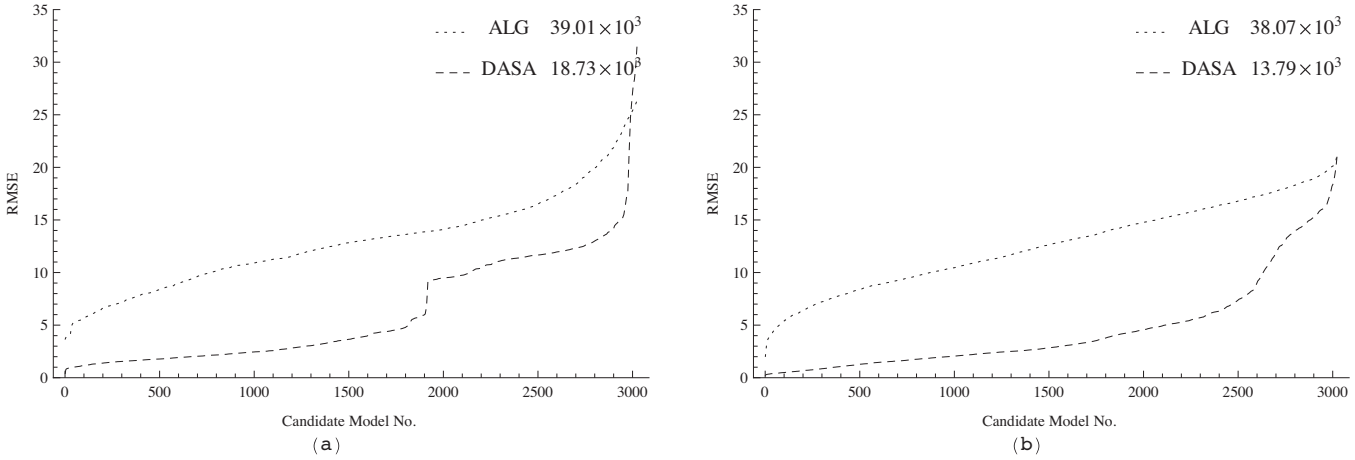


Fig. 14. Error profiles for ProBMoT/ALG and ProBMoT/DASA on all automated modeling tasks for the Glumsø domain. (a) Glumsø '73 and (b) Glumsø '74.

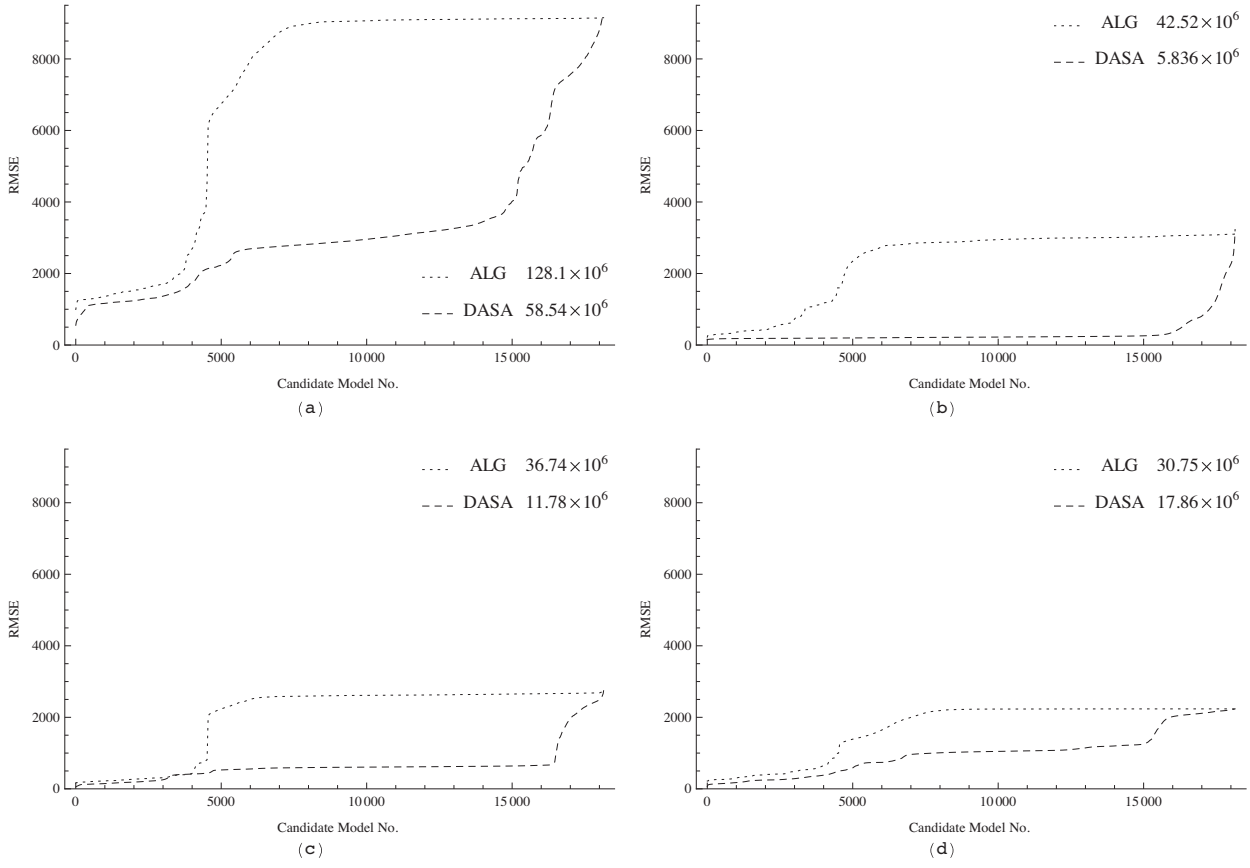
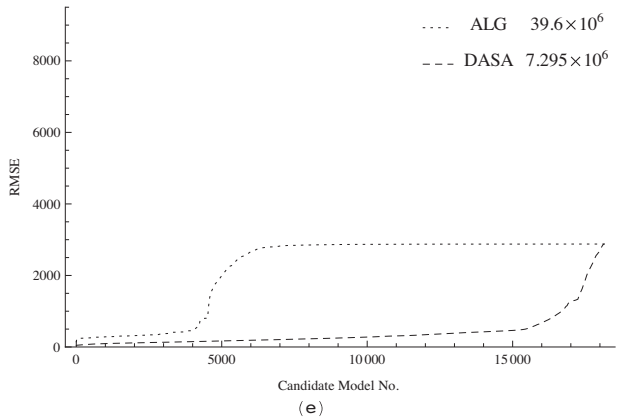
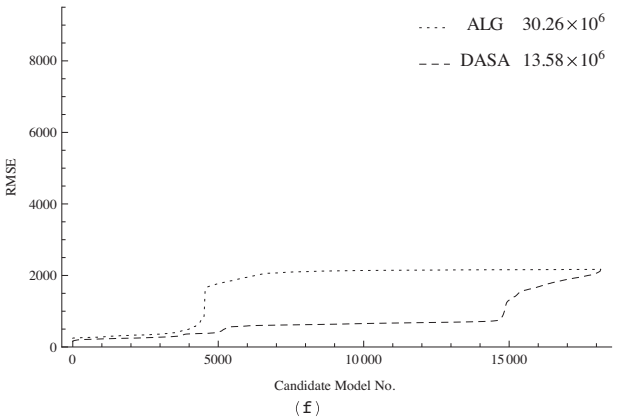


Fig. 15. Error profiles for ProBMoT/ALG and ProBMoT/DASA on all automated modeling tasks for the Kasumigaura domain. (a) Kasumigaura '86; (b) Kasumigaura '87; (c) Kasumigaura '88; (d) Kasumigaura '89; (e) Kasumigaura '90; (f) Kasumigaura '91; (g) Kasumigaura '92.





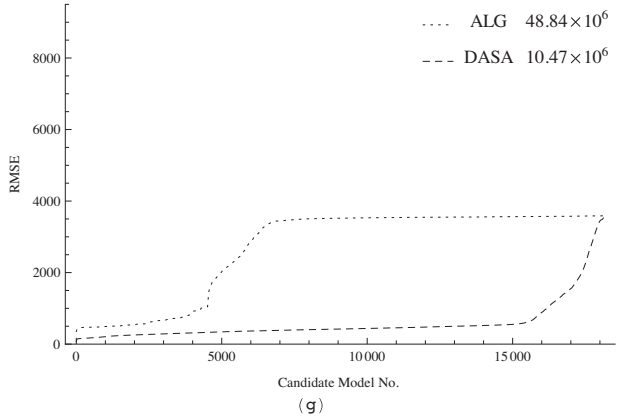
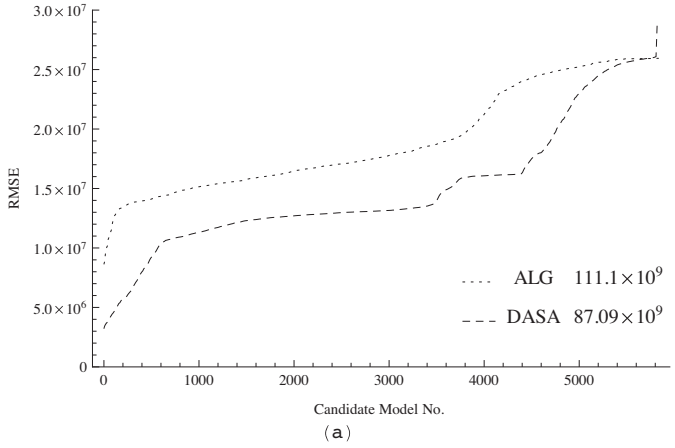
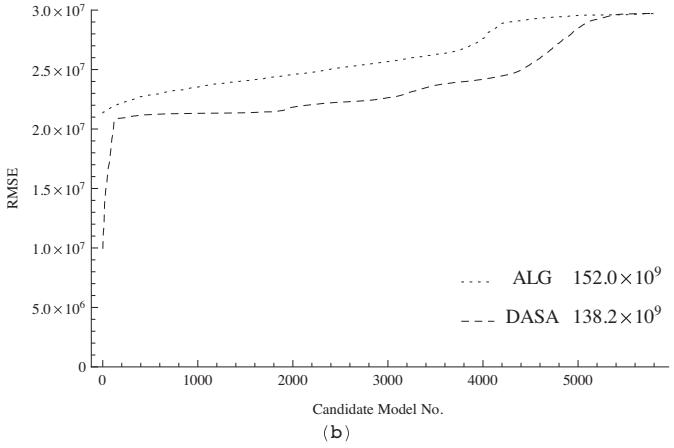
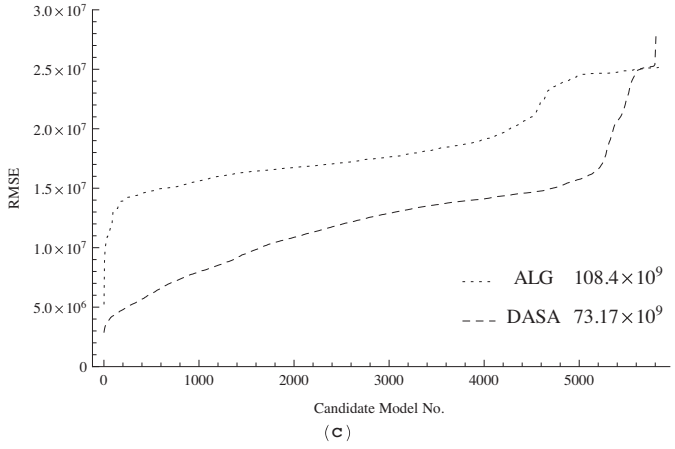


Fig. 15. (*Continued*).







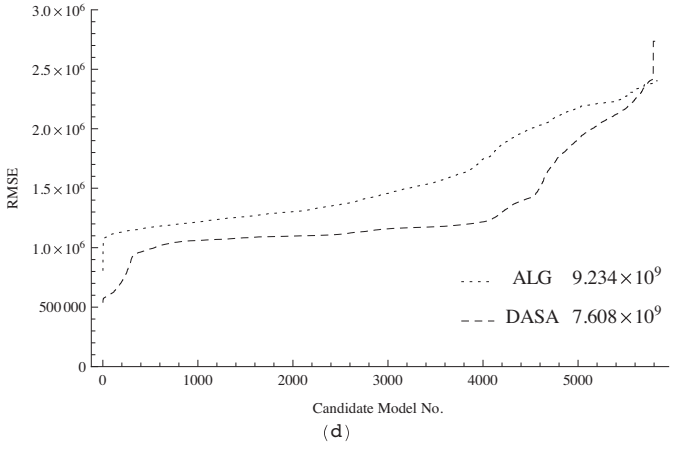


Fig. 16. Error profiles for ProBMoT/ALG and ProBMoT/DASA on all automated modeling tasks for the Venice domain. (a) Venice Loc. 0; (b) Venice Loc. 1; (c) Venice Loc. 2; (d) Venice Loc. 3.

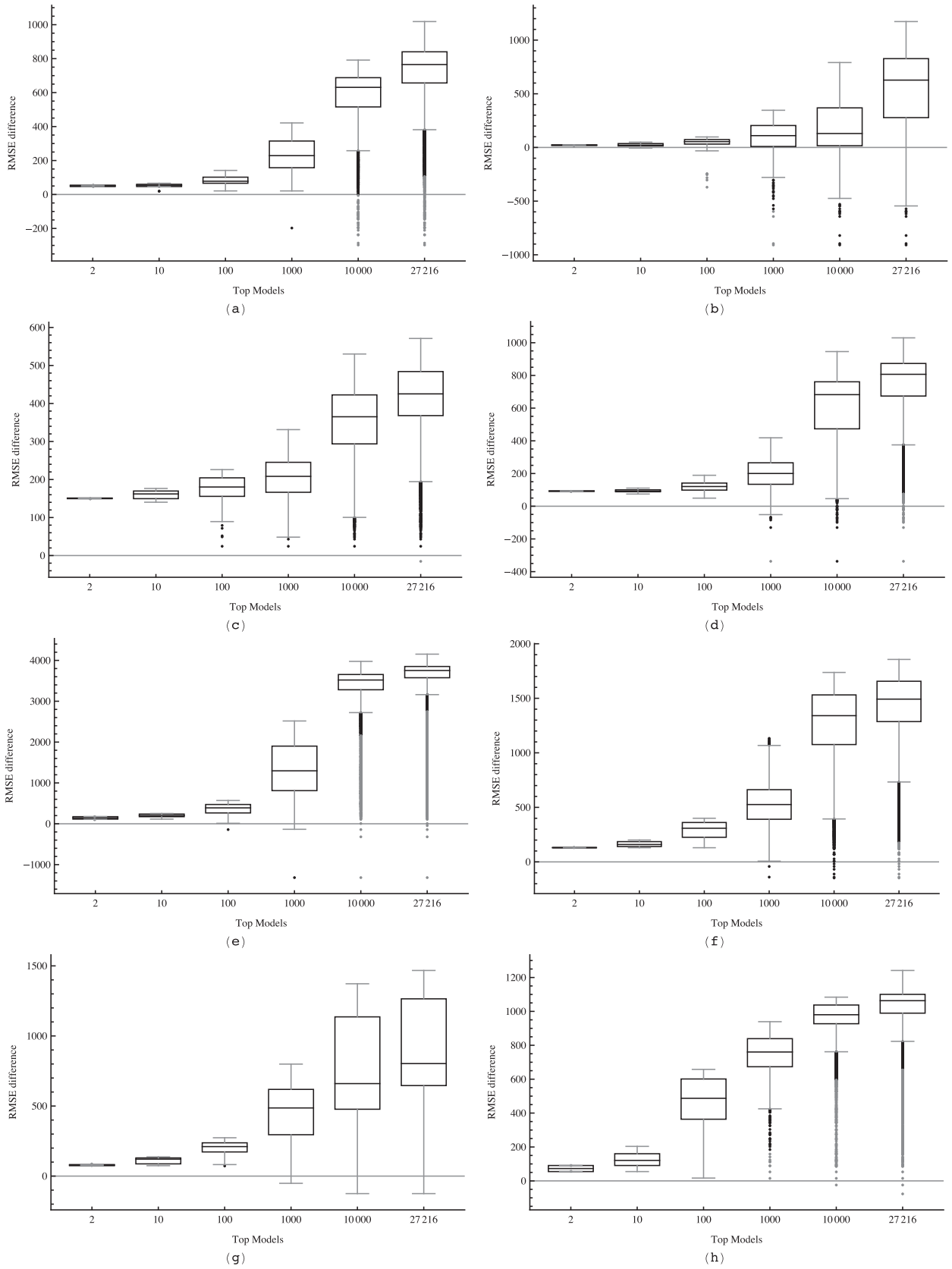


Fig. 17. Distribution of the model-wise differences of RMSE between ProBMoT/ALG and ProBMoT/DASA for all automated modeling tasks for the Bled domain. (a) Bled '95; (b) Bled '96; (c) Bled '97; (d) Bled '98; (e) Bled '99; (f) Bled '00; (g) Bled '01; (h) Bled '02.

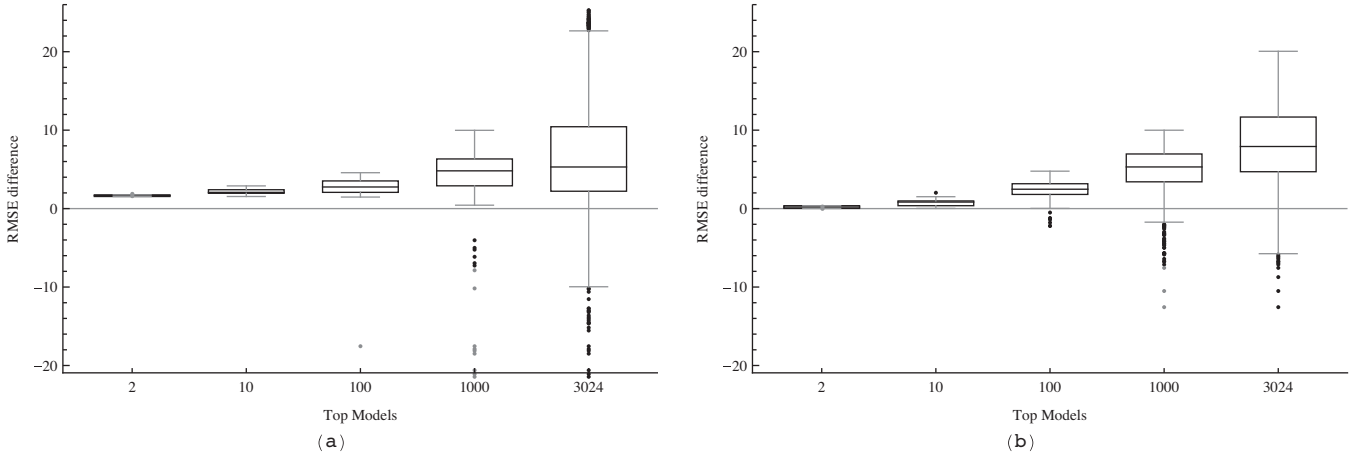


Fig. 18. Distribution of the model-wise differences of RMSE between ProBMoT/ALG and ProBMoT/DASA for all automated modeling tasks for the Glumsø domain. (a) Glumsø '73 and (b) Glumsø '74.

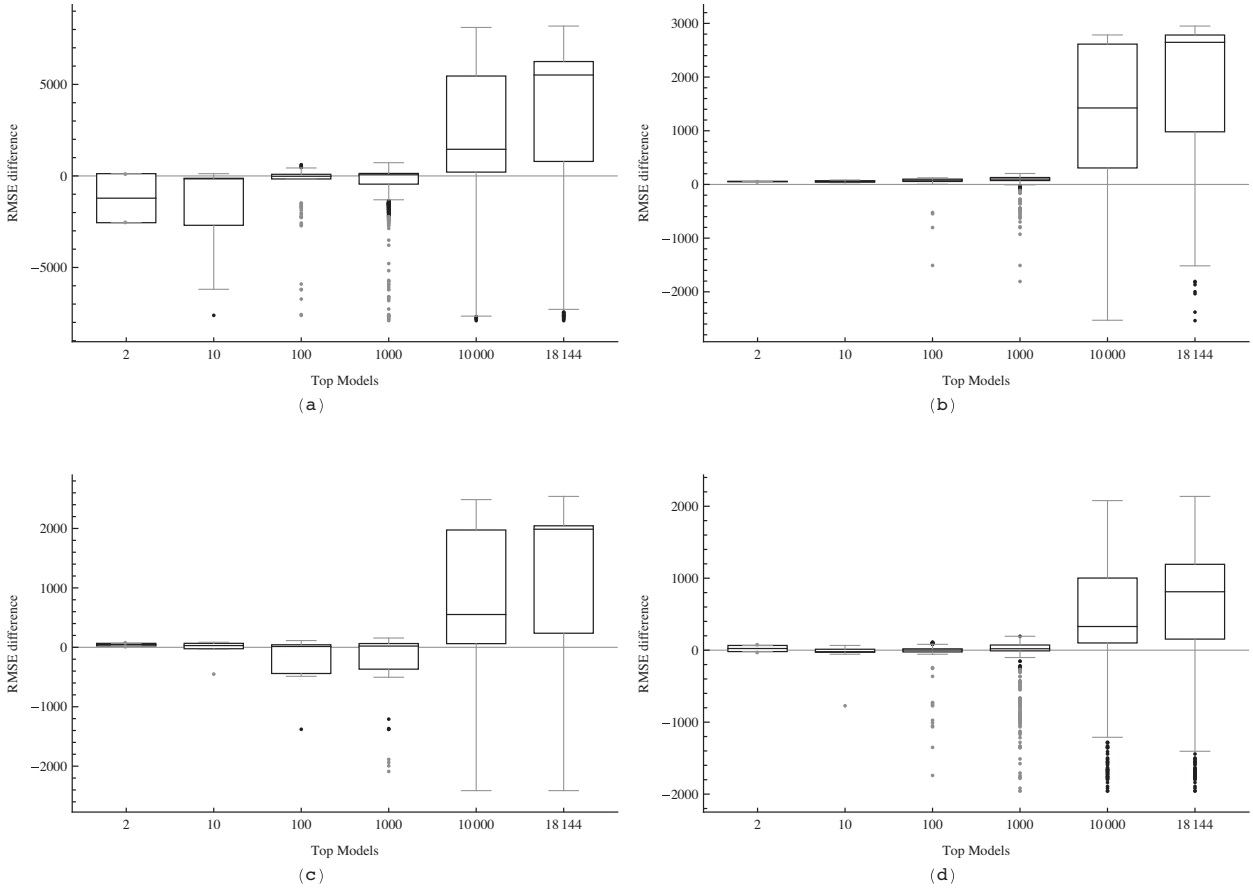


Fig. 19. Distribution of the model-wise differences of RMSE between ProBMoT/ALG and ProBMoT/DASA for all automated modeling tasks for the Kasumigaura domain. (a) Kasumigaura '86; (b) Kasumigaura '87; (c) Kasumigaura '88; (d) Kasumigaura '89; (e) Kasumigaura '90; (f) Kasumigaura '91; (g) Kasumigaura '92.

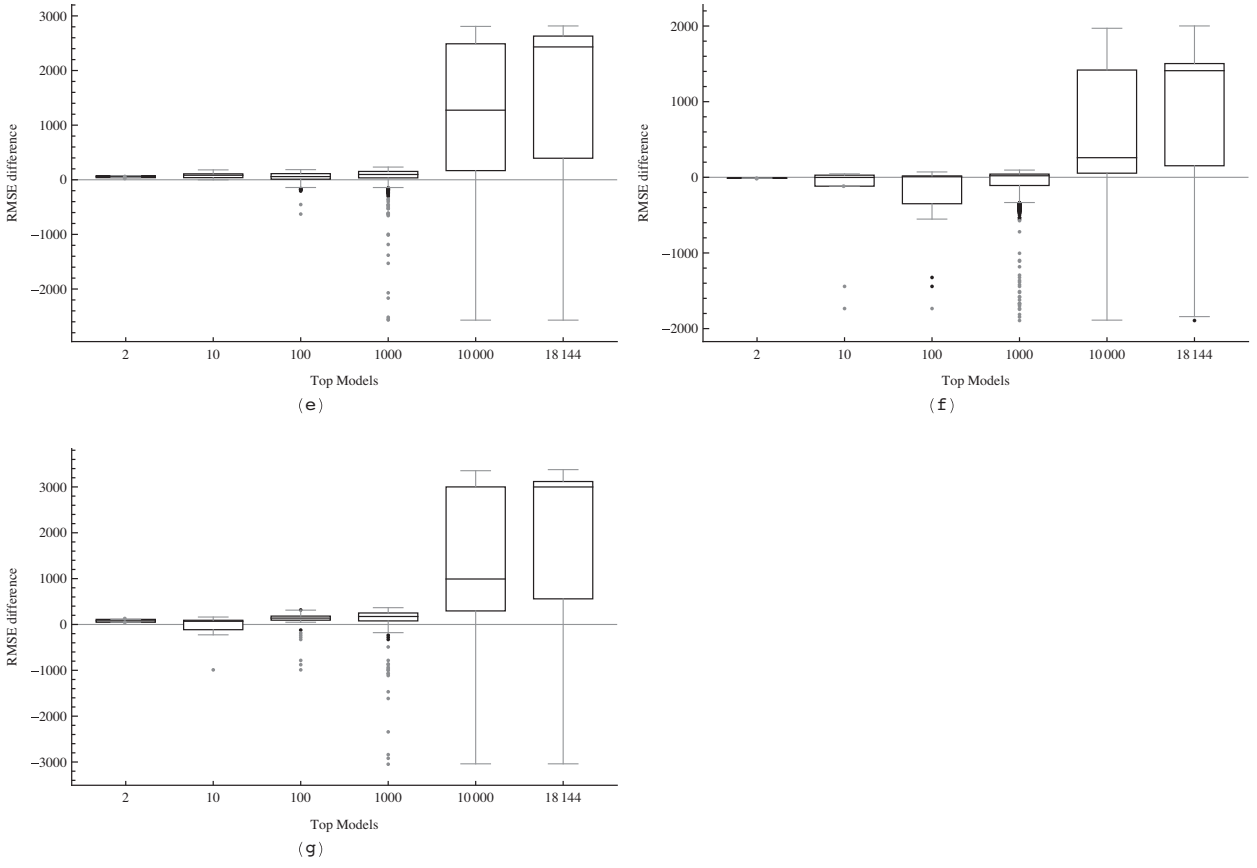


Fig. 19. (*Continued*)

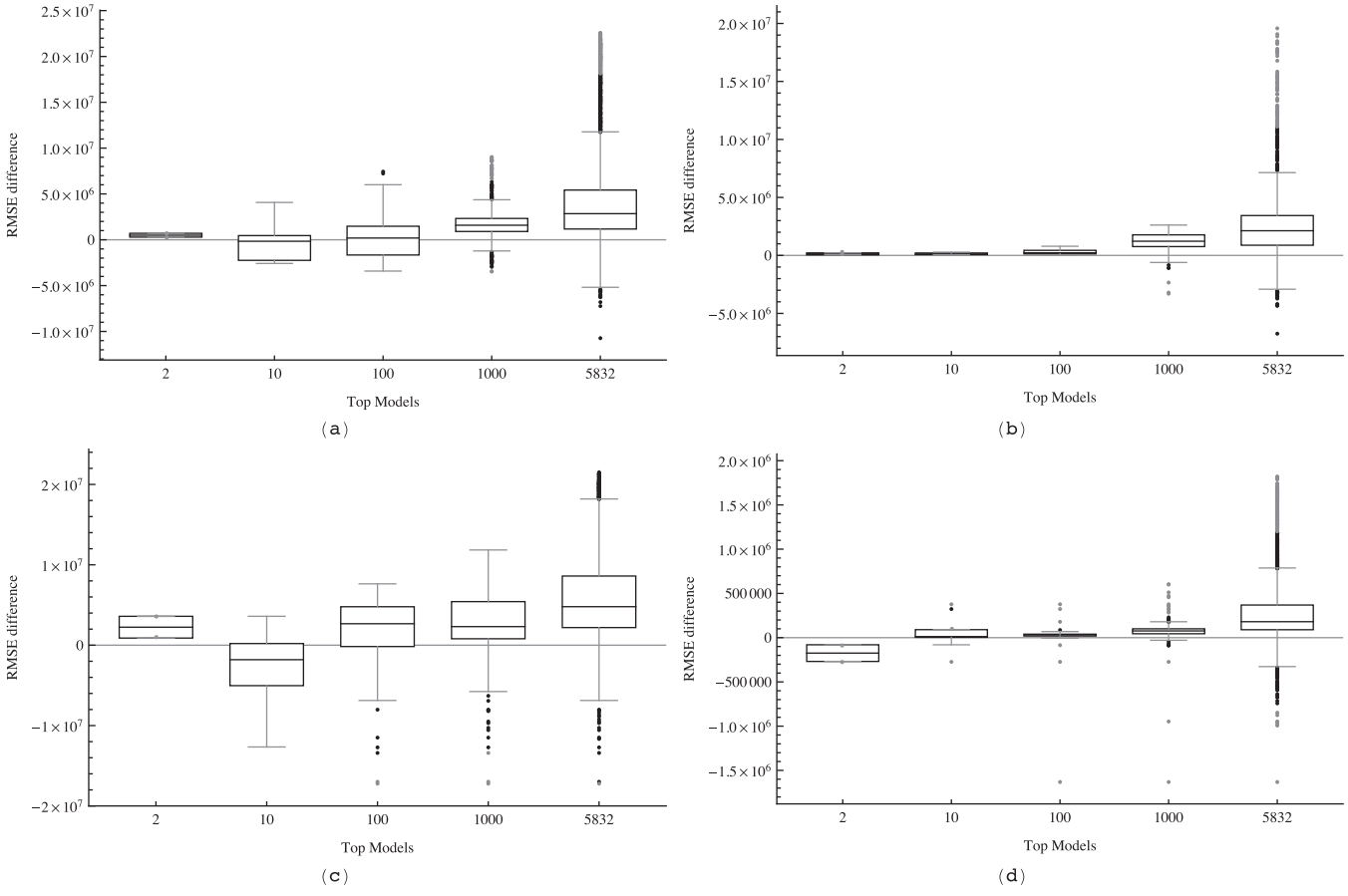


Fig. 20. Distribution of the model-wise differences of RMSE between ProBMoT/ALG and ProBMoT/DASA for all automated modeling tasks for the Venice domain. (a) Venice Loc. 0; (b) Venice Loc. 1; (c) Venice Loc. 2; (d) Venice Loc. 3.

References

Afshar, A., Kazemi, H., Saadatpour, M., 2011. Particle swarm optimization for automatic calibration of large scale water quality model (CE-QUAL-W2): application to Karkheh Reservoir, Iran. Water Resources Management 25 (10), 2613–2632.

Atanasova, N., 2006. Preparation and use of the domain expert knowledge for automated modeling of aquatic ecosystems. Ph.D. Thesis. Faculty of Civil and Geodetic Engineering, University of Ljubljana.

Atanasova, N., Recknagel, F., Todorovski, L., Džeroski, S., Kompare, B., 2006a. Computational assemblage of ordinary differential equations for chlorophyll-a using a lake process equation library and measured data of Lake Kasumigaura. In: Recknagel, F. (Ed.), Ecological Informatics. Springer, Berlin/Heidelberg, pp. 409–427.

Atanasova, N., Todorovski, L., Džeroski, S., Kompare, B., 2006b. Constructing a library of domain knowledge for automated modelling of aquatic ecosystems. Ecological Modelling 194 (1–3), 14–36.

Atanasova, N., Todorovski, L., Džeroski, S., Kompare, B., 2008. Application of automated model discovery from data and expert knowledge to a real-world domain: lake Glumso. Ecological Modelling 212 (1–2), 92–98.

Atanasova, N., Todorovski, L., Džeroski, S., Rekar Remec, v., Recknagel, F., Kompare, B., 2006c. Automated modelling of a food web in lake Bled using measured data and a library of domain knowledge. Ecological Modelling 194 (1–3), 37–48.

Athias, V., Mazzega, P., Jeandel, C., 2000. Selecting a global optimization method to estimate the oceanic particle cycling rate constants. Journal of Marine Research 58 (5), 675–707.

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

Bunch, D.S., Gay, D.M., Welsch, R.E., 1993. Algorithm 717: subroutines for maximum likelihood and quasi-likelihood estimation of parameters in nonlinear regression models. ACM Transactions on Mathematical Software 19 (1), 109–130.

Cao, H., Recknagel, F., Cetin, L., Zhang, B., 2008. Process-based simulation library SALMO-OO for lake ecosystems. part 2: multi-objective parameter optimization by evolutionary algorithms. Ecological Informatics 3 (2), 181–190.

Dorigo, M., Stützle, T., 2004. Ant Colony Optimization. MIT Press, Cambridge, MA.

Džeroski, S., Todorovski, L.,1993. Discovering dynamics. In: Proceedings of the Tenth International Conference on Machine Learning. Morgan Kaufmann, pp. 97–103.

Finley, J.R., Pintér, J.D., Satish, M.G., 1998. Automatic model calibration applying global optimization techniques. Environmental Modeling and Assessment 3 (1), 117–126.

Gershenfeld, N., 1999. The Nature of Mathematical Modeling. Cambridge University Press.

Gilboa, Y., Friedler, E., Gal, G., 2009. Adapting empirical equations to Lake Kinneret data by using three calibration methods. Ecological Modelling 220 (23), 3291–3300.

Horst, R., Pardalos, P., Thoai, N., 2000. Introduction to Global Optimization. Springer.

Korošec, P., Šilc, J., Filipič, B., 2012. The differential ant-stigmergy algorithm. Information Sciences 192, 82–97.

Luenberger, D.G., 1979. Introduction to Dynamic Systems: Theory, Models, and Applications. Wiley.

Matear, R.J., 1995. Parameter optimization and analysis of ecosystem models using simulated annealing: a case study at Station P. Journal of Marine Research 53 (4), 571–607.

Pfanzagl, J., 1994. Parametric Statistical Theory. Walter de Gruyter.

Tashkova, K., Šilc, J., Atanasova, N., Džeroski, S., 2012. Parameter estimation in a nonlinear dynamic model of an aquatic ecosystem with meta-heuristic optimization. Ecological Modelling 226, 36–61.

Todorovski, L., 2003. Using domain knowledge for automated modeling of dynamic systems with equation discovery. Ph.D. Thesis. University of Ljubljana.

Todorovski, L., Bridewell, W., Shiran, O., Langley, P.,2005. Inducing hierarchical process models in dynamic domains. In: Proceedings of the 20th National Conference on Artificial intelligence. AAAI Press, pp. 892–897.

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

Todorovski, L., Džeroski, S., 2006. Integrating knowledge-driven and data-driven approaches to modeling. Ecological Modeling 194 (1–3), 3–13.

Törn, A., Ali, M., Viitanen, S., 1999. Stochastic global optimization: problem classes and solution techniques. Journal of Global Optimization 14, 437–447.

Whigham, P., Recknagel, F., 2001. Predicting chlorophyll-a in freshwater lakes by hybridising process-based models and genetic algorithms. Ecological Modelling 146 (1–3), 243–251.

Zhang, X., Srinivasan, R., Bosch, D., 2009. Calibration and uncertainty analysis of the SW, model using Genetic Algorithms and Bayesian Model Averaging. Journal of Hydrology 374 (3–4), 307–317.