A Divide and Conquer Approach to Learning from Prior Knowledge

Eric Chown ECHOWN@BOWDOIN.EDU
Department of Computer Science, Bowdoin College, 8650 College Station, Brunswick, ME 04011 USA

Thomas G. Dietterich TGD@CS.ORST.EDU
Department of Computer Science, Oregon State University, Corvallis, OR 97331-3202 USA

Abstract

A major challenge in producing large-scale simulations of the type used in ecosystem modeling is the problem of *model calibration*. This paper presents a method for solving a particularly difficult model calibration task that arose as part of a global climate change research project. An obvious approach to solving calibration problems is to formulate them as global optimization problems in which the goal is to find values for the free parameters that minimize the error of the model on training data. Unfortunately, this global optimization approach becomes computationally infeasible for many real applications. This paper presents a new divide-and-conquer method that analyzes the model to identify a series of smaller optimization problems whose sequential solution solves the global calibration problem. This paper argues that methods of this kind—rather than global optimization techniques—will be required in order for agents with large amounts of prior knowledge to learn efficiently.

1. Introduction

Computational models of natural systems are the subject of intense efforts in many scientific disciplines. For example, in global climate change studies, computational models of the ocean and atmosphere are constructed to predict the effects of increased atmospheric CO_2 on the world's climate. These models are currently influencing government policy decisions.

Ideally, every scientific model would be based on well-understood laws, much the way electronic circuit simulators are based on the laws of physics. In most modeling situations, however, it is necessary to introduce approximations either because computation with the exact laws is intractable or because sufficiently accurate detailed laws are not known. In many applications, these approximations take the form of large temporal or spatial scales. Coarse-resolution models are more tractable and also reduce the need for highly accurate fine-grained system laws. However, phenomena that occur at finer scales cannot be modeled directly and typically must be summarized by some "lumped" parameters. For example, instead of modeling the exact time delays in each transistor of a switching circuit, a modeler might assume that the delays all have the same fixed value, a parameter Δ. This kind of parameter is sometimes called a *conceptual* parameter, because it does not necessarily correspond to any quantity that can be measured. Instead, conceptual parameters summarize the detail that exists below the scale of the model.

The values of conceptual parameters are chosen by the modeler through a *calibration* process. This is a form of supervised learning. The modeler collects a set of training examples consisting of model inputs and desired outputs. The modeler then manually adjusts the values of the conceptual parameters to obtain a good fit between the model and the data. However, manual calibration by the modeler suffers from three potential problems: (a) it is very time-consuming, (b) it is tedious, so the modeler typically does not find the optimal calibration, and (c) it may lead to overfitting of the available training data.

In this paper, we describe an approach to automating the calibration of a very complex global vegetation model called MAPSS (Mapped Atmosphere-Plant-Soil System (Neilson, 1995)). We show that standard methods are not able to handle this problem. Then we present a new divide-and-conquer approach to model calibration. The paper concludes with a discussion of the general lessons for machine learning—particularly for the problem of learning in the presence of prior knowledge.

2. An Example Calibration Problem

2.1 The Structure of MAPSS

The goal of the MAPSS system is to predict the consequences of global climate change on the distribution of plant ecosystems worldwide. The inputs to MAPSS are the climate (precipitation, temperature, etc.) and soil properties at a given site. The key outputs from MAPSS are predictions for the amount of vegetation and a biome classification. The amount of vegetation is measured by the *leaf area index* (or LAI). This is the ratio of leaf surface area to the ground surface area. The model predicts LAI separately for trees, grasses, and shrubs; it outputs a three-tuple: $\langle LAI_{tree}, LAI_{grass}, LAI_{shrub}\rangle$. The biome classification produced by MAPSS categorizes the site into one of 74 ecosystem types (or *biomes*) such as "tall grass prarie", "desert", "conifer forest", and so on. This classification is performed by a set of rules that consider the LAI values and the temperature profile of the site.

MAPSS is structured as a search procedure wrapped around a water-flow simulation model. The search procedure conducts a kind of generalized binary search to find values for the LAI 3-tuple such that the water-flow simulation model achieves water balance and sunlight balance. A site is in water balance if the sum of the water flows entering and leaving the soil (summed over an entire year) is zero and if the amount of water in the soil is always non-negative in each month. The soil can temporarily store water, but it cannot accumulate water from one year to another. A site is in sunlight balance if the amount of sunlight reaching the ground (as determined by tree LAI) can support the amount of grass growing there (as specified by grass LAI).

The water flow simulation model works by simulating the flow of water through the site. Water falls as precipitation (in the form of rain or snow). Precipitation either evaporates, runs off into streams, is stored as snow, or infiltrates into the soil. Soil water either percolates to deeper soil layers or is taken up by the roots of plants and transpired back into the atmosphere. MAPSS also simulates competition for sunlight by estimating the amount of light that reaches the forest floor given the current tree LAI.

Once the final LAI values have been computed, classification rules are applied to determine the biome class for the site, and the model terminates. To predict a vegetation map for the United States, this procedure is applied separately to each of 70,000 sites.

MAPSS is controlled by 65 parameters that determine the rates of water flows, plant growth, and light penetration. The key approximations in MAPSS involve the spatial scale (10km-squares), the depth scale (dividing the soil into 3 layers), and the time scale (1 month time-step). Roughly half of the parameters are "conceptual" and require calibration. We have focused on calibrating a set of 20 parameters specifying water flow rates. Of these one group of 12 parameters specify the rate at which water crosses the boundaries between the three soil layers. Theses include six parameters that describe saturated soil water flow and six parameters that describe unsaturated flow. The other eight parameters specify the rates that different vegetation forms transpire water; in the case of trees, the rate depends on the climate zone.

This review of MAPSS shows that it is much more complicated than typical physical models. The combination of the internal competition for water and sunlight and the internal search for LAI values that achieve water balance means that it is impossible to "invert" MAPSS and solve for the values of its conceptual parameters in terms of the model inputs and outputs. Hence, calibration of MAPSS requires some form of search for good parameter values.

2.2 The MAPSS Calibration Problem

To calibrate MAPSS, our collaborators started by building an input climate data set based on 1,211 weather stations in the U.S. and interpolating it to the 70,000 sites. Working from this data, we ran MAPSS (using the parameter values manually chosen by Neilson) to produce predicted outputs for LAI and RUNOFF (the total amount of water that runs off from the site). This gave us 70,000 input-output pairs for MAPSS. The calibration problem is to use these training examples to automatically recover the parameter values that were used to generate the output data.

The obvious approach to model calibration is to formulate it as a search problem. Let Θ be a vector containing the parameters to be calibrated. For a particular site s, define the error function

$$\begin{aligned} J(s,\Theta) &= (\widehat{LAI}_{tree} - LAI_{tree})^2 + \\ &\quad (\widehat{LAI}_{grass} - LAI_{grass})^2 + \\ &\quad (\widehat{LAI}_{shrub} - LAI_{shrub})^2 + \\ &\quad (RU\widehat{NO}FF - RUNOFF)^2, \end{aligned}$$

where $\widehat{LAI}$ and $RU\widehat{NO}FF$ are the LAI and $RUNOFF$ values predicted by the model using parameters Θ. The goal of the search is to find a value for Θ that minimizes $J(s,\Theta)$ summed over all sites s.

We refer to this approach to calibration as the *global search* approach, because it attempts to calibrate *all* of the parameters simultaneously. This use of "global search" should not be confused with "global optimization" algorithms, which guarantee finding a global minimum in a large search space.

In principle, we could use all 70,000 sites for calibration, but this is not necessary. We are only attempting to fit 20 parameters, and each site provides inputs and outputs for 12 months of the year, so there is no shortage of data! Moreover, the cost of evaluating $J(s, \Theta)$ once for all 70,000 sites is approximately 30 CPU hours on a SPARCstation 2. Hence, we drew a training sample of 82 sites chosen to include examples of each of the biomes. Within each biome, we chose a mix of sites that were centrally located and sites that were located near the boundaries. This ensured that we had solid positive examples of each class and also good "near miss" examples to help locate the decision boundaries between biomes.

2.3 Attempts to Calibrate MAPSS

We investigated three search algorithms for minimizing J: gradient descent, Powell's method, and simulated annealing. Gradient descent is difficult to apply to this problem, because MAPSS' search for LAI values makes it impossible to obtain a closed-form expression for the gradient of J with respect to Θ. We tried computing the gradient numerically, but gradient descent was unable to make much progress in reducing J from good starting points. Unfortunately, Powell's method (Brent, 1973), a search technique that can be applied when no gradient information is available, behaved the same as gradient descent. It was unable to make much progress in minimizing J.

The difficulty with both gradient descent and Powell's method is that they assume that the error surface J can be locally approximated by a plane. Both methods estimate the slope vector of the plane and move along that direction. Unfortunately, MAPSS is highly nonlinear as a result of (a) the competition for light and water between grasses and woody vegetation, (b) the many threshold effects (e.g., from unsaturated to saturated water flow), and (c) the nonlinear form of the model equations (e.g., most are exponential).

These search failures led us to try simulated annealing. We employed the adaptive simulated annealing code distributed by Lester Ingber (1995). Simulated annealing makes no assumptions about the shape of the error surface, but as a consequence it can be exceedingly slow. We have successfully applied it to calibrate the soil water sub-part of MAPSS, which has 12 parameters. After approximately 6 weeks of CPU time (746,000 evaluations of J), it found parameter values quite close to the target values. However, despite several attempts, lasting several months, we have been unable to successfully calibrate all 20 parameters via simulated annealing. The results that were obtained were not much better than those found by Powell's method. While in theory simulated annealing should work in such a case, it too needs to be calibrated to work effectively on a particular problem, and such a calibration is not practical for a search space as complex and computationally expensive as MAPSS.

Another family of methods for solving calibration problems are set-intersection methods. Set-intersection methods take each training data point and compute the set of possible modifications consistent with the prior knowledge that would account for that data point. The sets from each data point can then be intersected to find a small set of possible modifications that are consistent with all of the training examples. Then some method can be applied to choose one of these modifications, or all of the modifications can be retained to be processed when more data becomes available. One example of this approach is the Incremental Version Space Merging (IVSM) method (Hirsh, 1994). The version space formalism makes it easy to describe a large set of potential modifications compactly.

Closely related methods have been developed in the system identification and control literature (Combettes, 1993; Bertsekas & Rhodes, 1971; Deller et al., 1993). These methods address the problem of determining the state of a system given a model of the system and some observations about the state. Each observation can be analyzed to compute the set of states consistent with that observation, and those sets can then be intersected to refine the hypothesis concerning the unknown state. Klepper and Hendrix (1994) have applied these methods to calibrate ecosystem models.

We investigated applying set-intersection methods to MAPSS. However, it is not possible to compute the set of parameter values consistent with a given data point in closed form (even approximately), because of the great complexity and non-linearity of the MAPSS model. Hence, we were not able to apply set-intersection methods.

A third—and much less developed—approach to model calibration is the decomposition approach. This approach attempts to decompose the overall learning problem into independent subproblems. The one example that we know of is the work of Williams and Millar (1996). They study a model of an office building that has 14 algebraic equations with seven free parameters. A full seven-dimensional calibration of these free parameters using the Levenberg-Marquardt algorithm did not converge after several CPU hours, but they were able to decompose the model into three independent submodels involving one, two, and four parameters. These models could be calibrated in just a few seconds of CPU time.

When Ron Neilson performed the initial calibration of MAPSS, he employed a simple kind of decomposition method. The method of Williams and Millar cannot be applied directly, because it assumes that it is possible to solve the model in closed form for any parameter in terms of the other quantities in the model. The remainder of this paper describes the new decomposition method that we developed based on program analysis techniques.

3. A Divide-and-Conquer Calibration Method

The key idea of our divide-and-conquer method is to decompose the overall problem of calibrating MAPSS into a series of subproblems that can be solved sequentially to calibrate all of the parameters. Each subproblem is defined by a set of parameters to be calibrated and a set of training examples (sites) to be used for calibrating those parameters. The hope is that each subproblem is small enough (i.e., concerns a small enough number of parameters) that it can be solved quickly by existing search methods.

This scheme will only work if there are sites where only a subset of the parameters controls MAPSS' computations. For example, if a site is dry enough throughout the year, then the soil will only exhibit unsaturated water flow. Consequently, only the unsaturated water flow parameters will be involved in the water balance calculations of MAPSS; the values of the saturated water flow parameters will be irrelevant. The fact that MAPSS contains a large number of threshold conditions suggests that there will be many "operating regions" of MAPSS where only a subset of the parameters will be important.

Assuming we can identify these operating regions, the calibration process can start with the operating region involving the smallest number of parameters. Once they have been calibrated, their values are known. Next, we can choose the operating region involving the smallest number of remaining unknown parameters and calibrate them. In this way, we can sequentially calibrate all of the parameters by solving a series of small calibration problems.

To implement this approach, we must solve three problems. First, we must divide the model into good operating regions where only small subsets of parameters are relevant. Second, we must divide the sites (training examples) into those corresponding to each operating region. Third, we must find an optimization method capable of solving the individual calibration problems for each operating region.

We now consider these three problems in turn.

3.1 Dividing the Model

Formally, an operating region corresponds to a control path through the program. A control path is simply the sequence of statements executed by the program in a given run. The parameters relevant to that control path are all of the parameters involved in the computations along that path (including the conditions tested in `if` statements). Hence, the problem of identifying good operating regions reduces to the problem of identifying control paths through the model that involve only a small number of relevant parameters.

For many models this will not be a problem. Indeed with MAPSS our knowledge of the model made the relevant divisions clear. However, this research represents a step towards developing a completely automatic methodology so we set attempted to solve this problem in an algorithmic fashion.

Unfortunately, identifying relevent control paths is difficult for two reasons. First, many models, including MAPSS, are specified as large and unwieldy computer programs. Second, such programs will often contain a number of loops. In principle there are an infinite number of ways that programs can execute. This means we will be searching for specific control paths in what is essentially an infinite space.

We address the first difficulty by rewriting essential portions of the model in a declarative single-assignment, loop-free programming language. Automatic analysis of this declarative version is possible through the straightforward application of partial evaluation techniques as described below. This step is the weakest link in our attempt to build a general methodology. As noted already, however, in many cases this step will be irrelevant.

To solve the second problem we take advantage of the fact that we are in a supervised learning task. In a calibration task we will know all of the targets and can use them to highly constrain our search for good control paths. For example, MAPSS operates by searching the LAI space using water balance to constrain the search. Since we already know the LAI values this search is irrelevant. With MAPSS, this left us with a core model which we were able to make completely loop free. The search space can be cut down further by the realization that most control paths can be almost completely ignored. As soon as a candidate control path requires more than a threshold number of parameters it, and all of its offshoots, can be ignored.

We implemented a partial evaluator for our declarative version of MAPSS. The declarative-MAPSS code consists of a mixture of `if` and assignment statements. The evaluator takes as input the declarative-MAPSS code, a list of known parameters, and a threshold M specifying the maximum number of relevant unknown parameters that a path can contain. The partial evaluator then symbolically evaluates the program (Futamura, 1971).

An assignment statement is processed by storing its value in the symbol table and by analyzing the expression to determine whether it contains any new unknown parameters. If so, these unknown parameters are added to the set of unknown parameters for the current path. If the size of this set exceeds M, then exploration of this path is terminated.

At each `if` statement, the partial evaluator attempts to infer whether the condition is true or false. If the value can be inferred, then only the `then` or `else` branch is taken (as appropriate). If the value of the condition cannot be inferred, the current path splits into two subpaths. These are explored recursively in depth-first fashion. Any new unknown parameters that appeared in the condition expression are added to the set of unknown parameters for both subpaths.

For example, consider the code fragment shown in Table 1. Suppose `Drain_rate` and `Drain_rate2` are parameters and `soil_water` is an internal variable computed by previous assignment statements. A partial evaluation of this fragment (with $M = 1$ and no known parameters) proceeds as follows. Initially, there is a single path. However, the condition of the first `if` statement cannot be evaluated, so two paths are created. One with `soil_water` < 5.2 and one with `soil_water` ≥ 5.2. At this point, the set of relevant parameters for both paths is empty.

First, the `then` clause is evaluated, and the value `drainage = 0` is stored in the symbol table. Next, the program simplifies the assignment `extra_water` := `soil_water`−`drainage` to `soil_water`. Then the second `if` statement is processed. The condition can be evaluated to be true, so the `then` statement is evaluated, and the value of `drainage2` is set to zero. The last statement is then evaluated to infer that `final_water` is equal to `soil_water`. This first complete path is output. It involves no unknown parameters, so it is

Table 1. Fragment of MAPSS Infiltration Code

```
IF  (soil_water < 5.2)
   THEN  drainage = 0
   ELSE  drainage = soil_water * Drain_rate
extra_water = soil_water - drainage
IF  (extra_water < 5.2)
   THEN  drainage2 = 0
   ELSE  drainage2 = extra_water * Drain_rate2
final_water = extra_water - drainage2
```

not useful for calibration purposes.

Control then returns to the last choice point, which was the `else` branch of the first `if` statement. The value of `drainage` is set to the expression `soil_water` − `drainage`. Because this expression involves an unknown parameter, the number of unknown parameters on this path is set to 1. The condition of the second `if` statement is analyzed. In this path, it cannot be evaluated, so the path is split into two subpaths. In the first path, `extra_water` is asserted to be less than 5.2. The `then` branch is processed to assert that `drainage2` is zero, and the last assignment statement infers that `final_water` is equal to `extra_water`. This produces the second path, which has one relevant parameter.

Control then returns to the `else` branch of the second `if` statement. The assignment statement in this branch involves a second unknown parameter, `Drain_rate2`, so this path is terminated, because the number of unknown parameters exceeds M.

From this example, we can see that the partial evaluator returns the list of all paths having no more than M relevant unknown parameters. By starting with $M = 1$ and increasing, we implement depth-first iterative deepening and find the path with the smallest set of unknown parameters (breaking ties arbitrarily). We stop with the first path that has at least one unknown parameter.

3.2 Dividing the Training Examples

3.2.1 Identifying Training Examples

Once we have found a path with a small number of unknown parameters, we must identify a set of training examples that will be processed by this path in the final calibrated model. Those training examples can then be used to calibrate the unknown parameters of the path. We will say that such training examples *belong to* the path.

Unfortunately, the unknown parameters determine which training examples will take which paths. For example, in the code from Table 1, the value of

`Drain_rate` can determine whether the condition of the second `if` statement is satisfied.

A natural approach to solving this problem is to use an EM-style algorithm (Dempster et al., 1977). An EM approach would start with an initial value for the unknown parameters Θ. It would then take each training example and compute the probability that that example belonged to each path (the "E-step"). Then, holding those probabilities fixed, it would modify the value of Θ to minimize the expected error along each path (the "M-step").

Aside from the fact that this would require developing a probabilistic interpretation of MAPSS, there are three difficulties with this EM approach. First, the M-step requires a global optimization, because many parameters are shared across the paths. Second, both the E- and M-steps require considering all possible paths through the model, and hence, they do not scale to large models. Third, EM is a local search procedure, and it can easily get stuck in local maxima. The challenge in designing an algorithm for identifying training examples is to find a method that can work with only one path at a time and that can tolerate a high degree of noise throughout the process.

Our algorithm works somewhat like EM. It maintains a set of candidate models and alternates between two phases. In the "filtering phase", the candidate models are applied to filter the training data and identify training examples that have a high probability of belonging to the selected path. In the "calibration phase", these chosen examples are employed to improve the candidate models. We call this the *Data Gathering Algorithm*.

3.2.2 The Data Gathering Algorithm

Let Θ^k be the unknown parameters involved in the chosen path. At each point in the data gathering algorithm, there is a collection of N current models, $\Theta_1^k, \ldots \Theta_N^k$, and a set of N current training examples. For MAPSS we choose N to be 40. It was chosen to be large enough to overcome noise in the process, small enough to be computationally tractable, and large enough to support the final calibration of the path. We have experimented with other values with similar results.

Each filtering phase begins by evaluating each of the N examples on each of the N models and computing the prediction error $J(s, \Theta_i^k)$. The $N/2$ models with the lowest prediction error are selected to act as a filtering set. Then training examples are drawn at random from the training set one at a time. Each example is given as input to the $N/2$ models in the filtering set. Each model determines whether the example is sent down the target path. An example is said to have "passed" the filter, if at least V_{min} of the models agree that it belongs to the target path. Training examples are drawn until N examples have passed the filter. This completes the filtering phase.

In each calibration phase, the training examples are randomly assigned to the current models so that each model is assigned exactly one example. A quick, 1000-step simulated annealing search is then performed to calibrate each model with respect to its one assigned example. This produces N updated models, which are used for the next filtering phase.

The data gathering algorithm is initialized by generating N random parameter vectors. Half of these are randomly chosen to be the first filtering set. The algorithm terminates when the filtering phase finds N examples that all of the models in the filtering set unanimously agree belong to the target path.

There are three key aspects to this algorithm. First, the use of N current models produces the "ensemble effect" that makes their voted decisions more accurate than the decisions of any single model. Second, the use of only half of the models in the filtering set makes the filtering process robust to bad models. Third, the technique of training each model on only one example ensures that a "bad" example (i.e., one that does not belong to the target path) will only corrupt *one* of the N models instead of potentially all of them.

Table 2. Relative Accuracy in Site Selection

V_{min}	Random	Run 2	Run 3	Run 4
32	39.2	3.7	6.8	4.8
36	28.8	19.0	4.3	0.3

Table 2 shows the results of two runs of this algorithm for the most difficult path—finding examples of unsaturated water flow. The numbers show the percentage of false positives (current examples that do not belong to the target path) generated at each iteration. In the first iteration, most of the selected training examples do not belong to the target path, but by the fourth iteration, around 90% of the examples are correct for $V_{min} = 32$, and 99% for $V_{min} = 36$. As shown, the algorithm typically terminates within five iterations.

3.3 Calibrating the Path

Once a set of examples has been found that are believed to belong to the path, we apply simulated annealing to calibrate the parameters for the path. We begin the simulated annealing procedure using the parameter values from the best model found in the last filtering phase.

For the second and succeeding paths, we also permit previously-calibrated parameters to change their values slightly. This is controlled through the parameter temperature mechanism of ASA. Each parameter has a "temperature" that controls how rapidly the value of that parameter will be varied during simulated annealing. (This is in addition to the overall temperature that controls whether retrograde "moves" will be accepted during the search.) Parameters that were calibrated in previous iterations are given very low parameter temperatures. The temperatures are reduced exponentially as a function of the number of previous paths in which the parameter was already calibrated.

The progress of the calibration is shown in Tables 3–4. Each row corresponds to one path, and all parameters involved in that path are shown in that row. Parameters that appear for the first time are the key parameters being calibrated in that row. Parameters that were calibrated in previous iterations are also being calibrated, but at much lower parameter temperatures. More than 1,500,000 total evaluations of J were required, with the bulk of the evaluations coming in iterations 7 and 8. The results agree closely with the target values except for the top soil level saturated drainage parameters, which are evidently somewhat underconstrained by the model and the data.

4. Lessons for Learning with Prior Knowledge

Learning in the presence of prior knowledge is one of the big open problems in machine learning. Over the past two decades, many methods have been developed. It is important to realize, however, that most of these pursue the global search approach. Take FOCL (Pazzani & Kibler, 1992), for example. The prior knowledge (in the form of a Horn clause theory and intensionally- and extensionally-defined predicates) guides a FOIL-like algorithm for growing a set of decision rules. In KBANN (Towell et al., 1990), a fixed neural network structure constrains a gradient descent search. In the work of Clark and Matwin (1993), a qualitative process model constrains a rule-learning algorithm. In Cohen's (1992) work, a grammar constrains the space of horn-clause theories.

As intelligent systems acquire very large knowledge bases, these global approaches will not scale. It is unreasonable to use all new examples to rebuild the entire knowledge base of the agent while using the old knowledge base just to constrain the process. Instead, learning methods are needed that can identify only the subpart of the knowledge base that requires revision. Then relevant training examples must be identified so that a localized learning algorithm can be applied. This situation is very similar to the calibration problem described in this paper. Calibrating with all of the training examples was not feasible, so subparts were identified and calibrated individually.

This paper has shown that global search is not a viable method for calibrating models as complex as MAPSS. The paper has introduced a new approach that analyzes the structure of the model to identify a series of tractable subproblems that can be solved sequentially. Program analysis tools were applied to identify these subproblems. An iterative, statistical procedure was developed for finding relevant training examples. And an existing simulated annealing implementation was then employed to solve the individual subproblems. This combination of techniques was able to automatically calibrate the 20 most important parameters of the MAPSS model. The biggest remaining challenge for this work is to test its generality on additional models; its success on a model as complex as MAPSS is an extremely promising sign in that regard.

Acknowledgements

The authors thank Ron Neilson and Jesse Chaney for their support and assistance in carrying out this research. The authors gratefully acknowledge the support of the NSF under grants IRI-9204129 and ASC-9503976.

References

Bertsekas, D. P., & Rhodes, I. B. (1971). Recursive state estimation for a set-membership description of uncertainty. *IEEE Transactions on Automatic Control*, *AC-16*, 117–128.

Brent, R. P. (1973). *Algorithms for minimization without derivatives.* Englewood Cliffs, NJ: Prentice Hall.

Clark, P., & Matwin, S. (1993). Using qualitative models to guide inductive learning. *Proceedings of the Tenth International Conference on Machine Learning* (pp. 49–56). San Francisco, CA: Morgan Kaufmann.

Cohen, W. W. (1992). Compiling prior knowledge into an explicit bias. *Proceedings of the Ninth International Conference on Machine Learning* (pp. 102–110). San Francisco, CA: Morgan Kaufmann.

Table 3. Calibration Progress of Transpiration and Slope Parameters. The Target row shows the "correct" values of the parameters. The Range row indicates the legal range of values for this parameter.

Run	Grass	Tree1	Tree2	Tree3	Shrub	GrassS	TreesS	ShrubS
1	4.265					3.792		
2	4.249	3.748				1.417	1.001	
3	4.259		3.502			4.999	1.003	
4	4.278			2.750		3.496	1.032	
5	4.252				9.254	2.248		3.002
6					9.246			3.002
7	4.249					1.998		
8	4.251					2.072		
Target	4.25	3.75	3.5	2.75	9.25	2.00	1.00	3.00
Range	2–8.5	1.5–5	1.5–5	1.2–4	4–15	0–5	0–5	0–5

Table 4. Calibration Progress of Drainage Parameters. The first six parameters are parameters for unsaturated drainage, the last six for saturated drainage.

Run	Deep1	Deep2	Mid1	Mid2	Top1	Top2	Deep1	Deep2	Mid1	Mid2	Top1	Top2
1	0.786	11.476	0.286	2.744								
2	0.704	11.607	0.125	2.337								
3	0.194	9.971	0.491	2.994								
4	0.443	10.934	0.597	2.939								
5	0.371	10.293	0.640	5.051								
6	0.443	10.933	0.604	2.944	0.834	2.530						
7	0.274	9.754	0.603	3.644			0.849	11.447				
8	0.201	9.982	0.504	3.003			0.857	10.128	0.816	3.087	0.613	0.719
Target	0.20	10.00	0.50	3.00	0.80	2.50	0.80	10.00	0.80	3.00	0.50	1.0
Range	0–1	1–20	0–1	1–7.5	0–1	1–7.5	0–1	1–20	0–1	1–5	0–1	0.5–3

Combettes, P. L. (1993). The foundations of set theoretic estimation. *Proceedings of the IEEE*, *81*, 182–208.

Deller, J. R., Nayeri, M., & Odeh, S. F. (1993). Least-square identificatioin with error bounds for real-time signal processing and control. *Proceedings of the IEEE*, *81*, 815–849.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum-likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, *B39*, 1–38.

Futamura, Y. (1971). Partial evaluation of computation process—an approach to a compiler-compiler. *Systems, Computers, Controls*, *2*, 45–50.

Hirsh, H. (1994). Generalizing version spaces. *Machine Learning*, *17*, 5–46.

Ingber, L. (1995). *Adaptive simulated annealing [software package]* (Technical Report). California Institute of Technology.

Klepper, O., & Hendrix, E. M. T. (1994). A method for robust calibration of ecological models under different types of uncertainty. *Ecological Modelling*, *74*, 161–182.

Neilson, R. P. (1995). A model for predicting continental scale vegetation distribution and water balance. *Ecological Applications*, *5*, 362–385.

Pazzani, M., & Kibler, D. (1992). The utility of knowledge in inductive learning. *Machine Learning*, *9*, 57–94.

Towell, G. G., Shavlik, J. W., & Noordewier, M. O. (1990). Refinement of approximate domain theories by knowledge-based artificial neural networks. *Proceedings of the Eighth National Conference on Artificial Intelligence*. Menlo Park, CA: AAAI Press.

Williams, B. C., & Millar, B. (1996). Automated decomposition of model-based learning problems. *The Tenth International Workshop on Qualitative Reasoning* (pp. 265–273). Cambridge, MA: AAAI Press/MIT Press.