18th World IMACS / MODSIM Congress, Cairns, Australia 13-17 July 2009
http://mssanz.org.au/modsim09

Innovative Autocalibration Techniques Using High Performance Computing

Markiyan Sloboda[1], Vimal Sharma[1], Simon Hood[1], David Swayne[1], William Booty[2], David Lam[2] and Isaac Wong[2]

[1]Computing Research Laboratory for the Environment, University of Guelph, Guelph NG2W1, CANADA
[2]Aquatic Ecosystem Management Research Division, Canada Centre for Inland Waters, 867 Lakeshore Road, Burlington L7R4A6, CANADA dswayne@uoguelph.ca

Abstract: We compare the computational effectiveness of two different approaches to the autocalibration of environmental models: algorithms based on gridding the input parameter set; and genetic or evolutionary algorithms (GAs). Dramatic steps were taken to reduce the total run-time per simulation of the selected complex model in the calibration phase: a model that takes a few seconds per run can be calibrated simply, but one with several minutes or hours of computer time per run offers limited scope for matching fitness criteria. Complex models are difficult to calibrate. A model can be calibrated for its particular objective or purpose, using either algorithmic approach. There are gains from machine learning, over GAs for some situations which we will enumerate.

Coarse-grained parallelization offers a limited scope for increasing the number of simulations that can be run in acceptable elapsed wall-clock time, but some hard limits do exist. Our high-performance computer platforms are based on interconnecting clusters of processors, connected by high speed (gigabit) Ethernet. For a cluster of n processors, the hard lower bound on the fraction by the elapsed time can be reduced is $1/n$, limiting the utility of increasing n for moderately complex models. Fortunately, the communication overhead in this class of architecture is demonstrably nearly constant with n for a moderately large number (<100) of processors, and it has a relatively minor negative impact on speedup. Our parallelization of two well-known algorithms, Shuffled Complex Evolution (SCE) and Dynamically Dimensioned Search (DDS) was relatively simple. The algorithms were run asynchronously on disjoint subsets of the parameter search space for a fixed number of steps on n-1 processors, after which they reported to a master node which determined the redistribution of computational effort depending on the current value of the fitness function being used to monitor the success of the calibration. Extensive experiments were performed using classical test functions and environmental models of different time-complexity, whose run times varied from a few seconds to significant fractions of an hour. Limiting the numbers of interprocessor communication steps required in this master-slave interaction may increase the overall numbers of simulations, but this compromise does not change the order of the speedup, for practical numbers of processors (N<30).

An alternative to a parallel GA is to apply machine learning using a “gridded” approach. This would appear at first glance to be a “brute force” approach, but we note that the required numbers of simulations to “learn” the behaviour of the parameter space in the calibration grows logarithmically with the size of that space, and that this class of approach has the potential to be competitive with GAs. The impact on multiple simultaneous calibrations of several watersheds can be significant, even to the point of making feasible a problem that, with other calibration schemes and GAs is often intractable. Since the calculation is derived from the fitness test, simulations cannot be easily reused in the GA approach; as a result, a lot of computation is discarded. By contrast, for machine learning approaches, every simulation is preserved in a database, and the comparison of the fitness function to observation is not required during simulation. If a fitness test is applied and it dictates a second or subsequent iteration is necessary, it is included in the database for potential future use. From this approach, we also hypothesize that a “table-top” agent for a specific watershed nonpoint source (NPS) model, for instance, is certainly worth the attempt to implement machine learning. This approach may be the only reasonable calibration strategy for multiply-connected models or for producing Bayesian Network model approximations for dealing with uncertain inputs.

Practical tests of hypotheses on communication overhead that confirm our assertions have been conducted using SAC-SMA, WATFLOOD, AGNPS and SWAT using data from published test results and from typical Southern Canadian watersheds. Our test runs make use of the Shared Hierarchical Academic Research Computer Network (SHARCNET), located in several institutions in Southern Ontario, Canada.

Keywords: *environmental models, high performance computing, calibration, genetic algorithms, machine learning*

1. INTRODUCTION AND PROBLEM STATEMENT

1.1 Genetic Algorithms

Our group has conducted, for a number of years, experiments in utilizing high performance computing for autocalibration. In that time we have developed an understanding of the limits posed by coarse grained parallel computing. We have experimented with two algorithms: the Shuffled Complex Evolution (SCE) algorithm of Duan (2003) and the Dynamically Dimensioned Search (DDS) algorithm (Tolson and Shoemaker, 2007). Both algorithms are of the evolutionary type. SCE is currently a favoured algorithm in the literature, and DDS has many desirable properties, including the ability to continue searching in the calibration parameter space beyond local optima for promising solutions to the calibration problem. It apparently outperforms SCE in a number of test cases (Tolson et al. 2007). Both algorithms have a number of properties which limit their usefulness.

Firstly, they are sequential in nature. We have parallelized both of them successfully, but they still present a significant computational burden. Parallelization may result in a slower (or faster!) algorithm. Thus, there may be issues with parallelization that were not contemplated in the serial algorithm. Even though this is not the experience, it may be the case that performance is less predictable.

Secondly, they do not lead to the "reuse" of simulations, which once performed are usually no longer part of the computation. Reports in the literature indicate that large numbers of simulations are required for the agricultural non-point source (NPS) SWAT model (SWAT 2005): there can be of the order of 50,000-250,000 simulations involved to explore full parameter sets (Tanakamaru and Burges, 1996; Franchini et al., 1998), and even with a 100 processor configuration, a model which takes (say) 12 minutes per individual run will require, for function evaluations alone, 100 hours of wall-clock time. For example Eckhardt and Arnold (2001) reported using 18,000 simulations to calibrate on 18 parameters. DDS has had some better results: in our work with Tolson, we reported that a 13 parameter SWAT calibration required 3000 simulations using DDS.

Thirdly, even though a GA can find more than a single solution parameter set, there is no guarantee that it can find all of them, particularly if more than one needs to be found. That is, each useful calibration parameter set is a new problem which may be less easy to find than its predecessor. This is equally true for parallel and for sequential versions of these algorithms. GAs are often tested on pathological functions, where hill-climbing is tested and properties of GAs are tested on solution spaces that may be far less forgiving than the situation in model calibration.

A fourth limiting property of GAs is that the fitness function chosen to evaluate simulations for their closeness as measured against observation cannot be separately evaluated from the test simulations. This function (or functions) drives the simulation, and is typically intimately bound with the iteration, as it drives the trajectory towards a potential solution. If new observations require recalibration, there is no possibility for reuse of the simulations already performed.

1.2 Machine Learning

There is one possible alternative to GAs that with which we are gaining experience. If the input parameter space is gridded, it is possible to perform several simulations which are independent of any fitness function or observation. Initial consideration of this approach would rule it out on the grounds of the impossibly large numbers of simulations. For 10 parameters, each uniformly distributed on [0,1], the numbers of simulations that would blanket the whole space, to a single decimal digit, is 10^{10}.

To overcome this very real problem, we will use a machine learning approach, and accept that we might miss a solution. If we agree to be, say, 99% certain that we have a solution on the discretized space that is at most 1% in error, it can be shown that we require only 2800 simulations. That estimate is based on Probably Approximately Correct (PAC) learning[1]. The total number of examples E required to approximately represent an hypothesis H with probability 1- δ to within a small positive ε has been shown to satisfy

$$E \geq (\ln(H) - \ln(\delta))/\varepsilon$$

[1] For an exposition of PAC learning, including the details of this theorem, *Artificial Intelligence: a Modern Approach (2nd ed.) by Russell and Norvig, (Prentice-Hall) pp.668-670* is an excellent resource.

The proof of the theorem concerning this lower bound leads one to believe that success on a single iteration is not only possible, but probable. Multiple iterations can reduce the probability of missing an acceptable solution to very nearly zero. PAC learning can be used to estimate a useful lower bound on the number of simulations required to successfully calibrate a model. The method is admittedly crude: there is no genetic component or extrapolation involved; just a randomly generated, but organized, systematic search for an acceptable set of calibration parameters.

The calculation of the simulations does not involve the calculation of the requisite fitness function, such as the *coefficient of determination* or *index of agreement* or *coefficient of efficiency* as defined in Legates and McCabe (1999). This processing step does not have to be executed until the necessity arises to fit a particular set of observations. Furthermore, there is no need to throw away simulations about a particular watershed, so long as the physical parameters and the time-scale of the simulation remain within the scope of the original assumptions. This is a form of machine learning in its finest form: accumulating results to further fill in the available information. If multiple iterations are performed, we save all of the simulations in the case re-calibration is needed at a later step, say in the co-calibration of an NPS model with a receiving reservoir or lake.

What is being proposed is *not* a one-step process, but rather an iterative one. Successive iterations at precision level of k digits after promising regions at k-1 digits have been identified can be explored (and the results saved for possible future use). In order to determine promising regions, we have implemented searching using so-called *association rules* using the R-Project's[2] software tool *arules* (Borgelt, 2004). The *apriori* algorithm is employed to generate a set of regions defined by parameters which give promising values for the fitness function of choice, and the simulation proceeds within the boundaries of this region. Other schemes have been proposed, based on cluster analysis, but these have not yet been tested.

2. Coarse-grained Parallel Autocalibration

2.1 Our Platforms

We are fortunate to have access to a number of parallel machines using the Canadian high performance computing resource called SHARCNET. Including the University of Guelph, SHARCNET[3] is "a consortium of Canadian academic institutions who share a network of high performance computers. "

2.2 Hypothesis for Total Time taken for Genetic Algorithm Computation

We are assuming that the type of model being used does not lend itself to being internally parallelized. Therefore, we are gaining speedup with an upper bound as compared to its single simulation time T, a total computation time of kT/*n*, where k is the number of required simulations. This is a bound for parallel Shuffled Complex Evolution (P-SCE) and parallel Dynamically Dimensioned Search (P-DDS), because there is a need to *gather* the fitness functions of separate parallel search complexes and to restart the system based on best or near-best estimates of where the best solution (according to fitness) is likely to be found. Embarrassingly Parallel DDS has the least communication overhead, but it is not zero. In this section we explore the behaviour of our three parallel algorithms (P-SCE, P-DDS and EP-DDS).

If the total time required for operation is unity and α is computation time required, we derive the following relationship. For the total wall-clock time T, we say

T = Computation time $(1 - \alpha)$ + Communication time (α)

For *n* processors, the relative unit wall-clock time can be expressed as:

$$T_n = \frac{1}{n} + \alpha\left(\frac{n-1}{n}\right)$$

Figure 1 shows the different trends for various values of α. Each of these trends represents the total time taken with increasing number of processors for a given value of α.

This means if a problem is computationally expensive and computation time is the dominating factor in the above equation, total time taken will follow one of the trend lines. If the communication time becomes

[2] http://www.r-project.org

[3] http://www.sharcnet.org

significant or comparable to computation time, the time taken by adding processors will jump to various α trend lines. Thus we can predict if the algorithm will improve in efficiency for a particular problem by adding more processors or not.

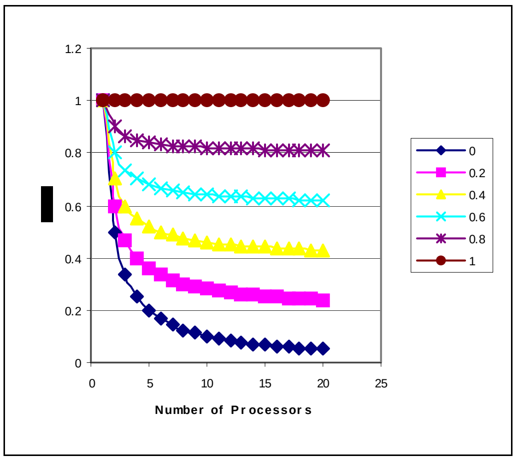


Figure 1: Different trends for taken for various values of α.

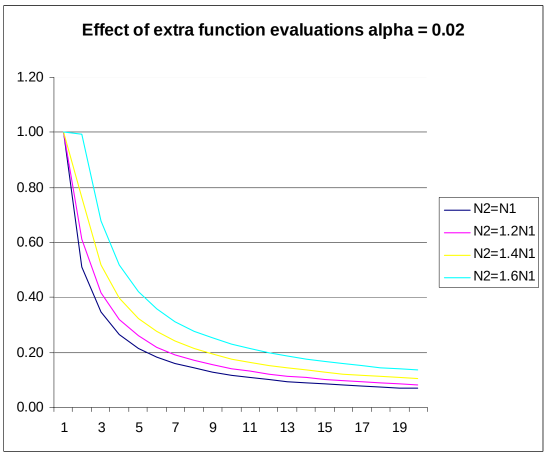


Figure 2: Effect of extra function evaluations as a result of parallelization.

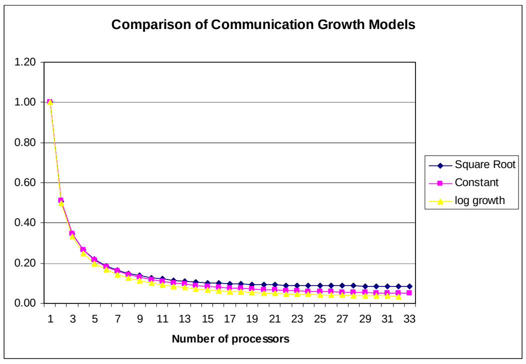


Figure 3: comparison of constant communication with logarithmic and square-root growth.

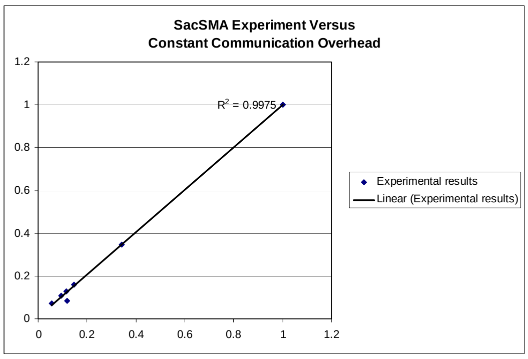


Figure 4: Sample validation of communication over-head (SAC-SMA on Leaf River Example). The optimal value of the α parameter is 0.02 for this case .

3. Gridded Machine-Learning

3.1 Motivation

Our strategic objective is to develop a suite of communicating models suitable for a very large river basin. The two areas in which we are working are: the Great Lakes Basin (especially Lake Ontario) and the Lake Winnipeg Basin. We are very far from our objective on Lake Winnipeg's sources, but we hope to achieve some success in modelling within five years. Problems abound: Lake Winnipeg is the receiver of several hundred smaller watersheds and many unique features ("exotic" rivers passing through to the lake, large non-contributing areas whose impact carries a probability rather than a certainty, and several other properties for which model solutions are not yet available).

Our preferred route is to develop reasonable models based on SWAT and other tools, to assist in assembling a very large Bayesian Network approximation. To have any chance at modelling a significant number of the features we will be faced with extensive calibration: a daunting task. In using a sophisticated model like SWAT to model watersheds for input to a larger assessment of the Basin, the calibration has to be both automatic and extremely efficient. It is unlikely, based on our Great Lakes modelling experiments that uncalibrated simulations will be sufficient, and it is equally unlikely that we can afford to execute the numbers of simulation required to use a GA for calibration.

3.2 Example using explicit gridding

We are working for the moment on the watersheds which present nutrient loading issues to Lake Ontario, within the Great Lakes Basin. We illustrate the explicit PAC algorithm (a single step), through an example taken from the Raisin River, in Eastern Ontario, Canada. The Raisin River flows into the St. Lawrence River, which drains the Laurentian Great Lakes. Various modelling experiments have been performed on this watershed, and it has been extensively monitored. The watershed's length is 150 km, and its surface area measures 546 km^2. We calibrated only the flow, during the period 1985-1994, and we validated the results on 1995-2004 data, at the Williamstown Gauging Station, shown on the map. To produce a tractable example for this paper, only four of the most sensitive SWAT parameters were chosen (Table 1). The range and steps for the parameters are in the rightmost two columns. Experience has taught us that these parameters incorporate the overwhelming impact of the Winter season in Southern Ontario, and, as a result, are the most sensitive ones in calibrating flows in this region. A coarse gridding was selected, to minimize turn-around time for the experiment, as access to the SHARCNET grid becomes restrictive as problem size escalates.

The normal SWAT 2005 run-time for the Raisin river watershed, for a four-year simulation is about 15 minutes on an Intel Core 2 Duo CPU, T7250 @ 2.00 GHz, with 2GB of RAM. By eliminating DISK I/O, cleaning memory after each use, and running inside a wrapper to control inputs and collect statistics, we were successful in reducing the run-time to 4.5 minutes.

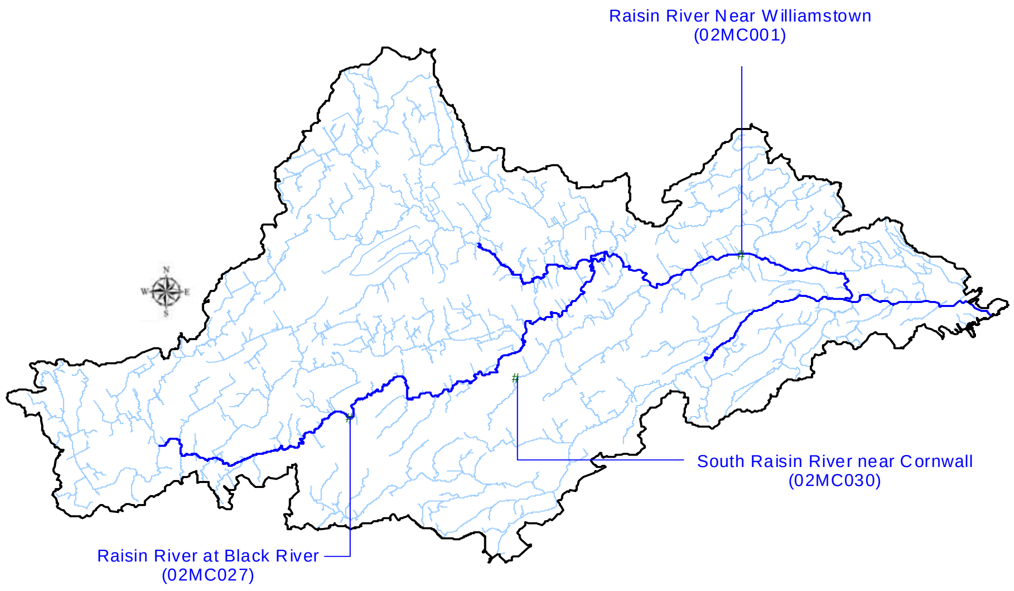

Raisin River Near Williamstown
(02MC001)
South Raisin River near Cornwall
(02MC030)
Raisin River at Black River
(02MC027)

Figure 5: Raisin River, Southeastern Ontario

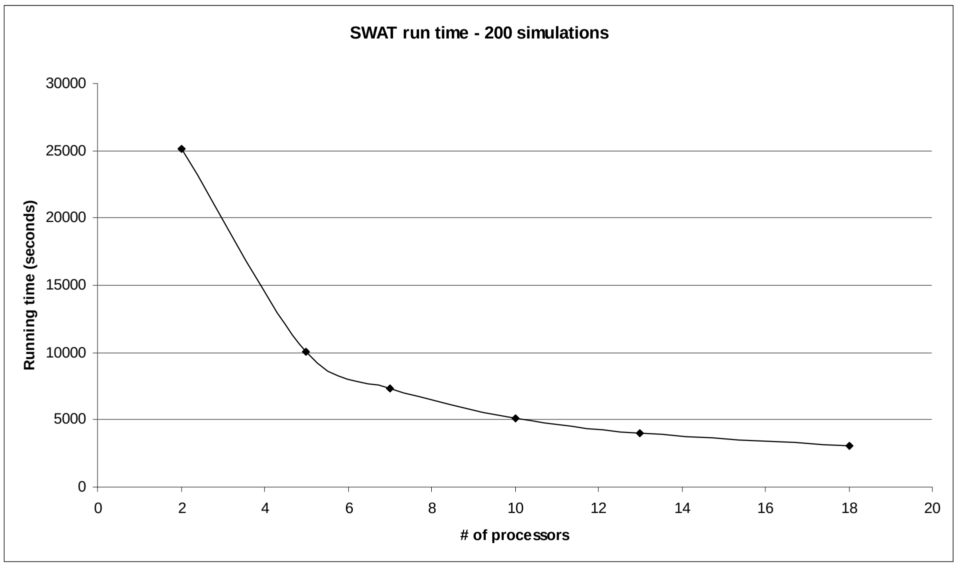
The PAC theorem predicted a minimum 200 simulations would be needed to capture the behaviour of the fitness function (in this case the coefficient of determination r^2) with probability of 95%. In that, we were successful, as Figure 6 and Table 2 illustrate.

Hydrology Parameter	Name	Range	Step
SFTMP	Snowfall temperature (°C)	0.3-0.7	0.1
SMTMP	Snow melt base Temperature (°C)	0.9-4.5	0.9
TIMP	Snow pack temperature lag factor	0.05-0.25	0.05
ESCO	Soil evaporation compensation	0.1-0.9	0.2

Table 1: Raisin River table of actuator parameters for preliminary SWAT trial

In order to have a compatible FORTRAN with the reference model obtained from the SWAT web site, SWAT was run on an older SHARCNET MPI cluster made of dual Itanium2 processors running at 0.9GHz, with 2GB memory, connected via Gigabit Ethernet. We are currently porting the whole configuration to a faster machine, but the vagaries of FORTRAN implementations restricted our experiment for this paper.

Calibrated values: SFTMP – 1.5; SMTMP – 0.9; TIMP – 0.25; ESCO – 0.7;
Flow calibration R-square: 87%, validation R-Square: 89%



#proc	Run time (sec)
2 –	25,128
5 –	10,063
7 –	7,357
10 –	5,072
13 –	4,046
18 –	3,047

Figure 6: experimental results: optimal parameters and time to completion of 200 simulations for various processor configurations.

Using a servlet we wrote to examine the results of the 200 simulations, the following results for the value of r^2 were obtained. Raw results were first "roughened" or put in ranges. The arules package was then used to generate results or consequences according to the implications of the antecedents. Three numbers are included in the output: *support, confidence* and *lift.*

Conditions (antecedent)	Result (consequent)	Support	Confidence	Lift
{SFTMP=1 - 1.5,SMTMP=3.6 - 4.5}	{R2=0.36 - 0.6}	0.075	0.75	4.29
{SFTMP=1 - 1.5,TIMP=0.05 - 0.1}	{R2=0.36 - 0.6}	0.075	0.75	4.29
{SMTMP=3.6 - 4.5,TIMP=0.05 - 0.1}	{R2=0.36 - 0.6}	0.075	1	5.71
{SMTMP=3.6 - 4.5,TIMP=0.15 - 0.25}	{R2=0.6 - 0.69}	0.075	0.75	5
{SMTMP=1.8 - 3.6,ESCO=0.9 - 0.9}	{R2=0.69 - 0.77}	0.075	1	5.71
{SMTMP=0.9 - 1.8,ESCO=0.3 - 0.5}	{R2=0.77 - 0.84}	0.075	0.6	3.43
{SMTMP=0.9 - 1.8,ESCO=0.1 - 0.3}	{R2=0.77 - 0.84}	0.075	0.75	4.29
{SMTMP=0.9 - 1.8,TIMP=0.05 - 0.1}	{R2=0.77 - 0.84}	0.1	1	5.71
{SMTMP=0.9 - 1.8,TIMP=0.1 - 0.15}	{R2=0.84 - 0.87}	0.075	0.6	3
{SMTMP=0.9 - 1.8,ESCO=0.9 - 0.9}	{R2=0.84 - 0.87}	0.075	0.75	3.75
{SMTMP=0.9 - 1.8,TIMP=0.15 - 0.25}	{R2=0.87 - 0.88}	0.1	0.67	5.33

Table 2: arules output for rules where the consequent is a range of values for r^2.

S**upport** is the number of rangified transactions that include all items in the antecedent and consequent parts of the rule. (The support is sometimes expressed as a percentage of the total number of records in the database.) **Confidence** is the ratio of the number of transactions that include the consequent as well as the antecedent (namely, the support) to the number of transactions that include all items in the antecedent. In the last line of Table 1, 2/3 of the rules with the given antecedent give r^2 values between 0.87 and 0.88. **Lift** is the ratio of Confidence to Expected Confidence, where Expected Confidence is the number of transactions that include the consequent divided by the total number of transactions.

Referring to the last row of the table: in this case, for 2/3 of the examples for which SMTMP and TIMP lie, respectively, in the range 0.9-1.8 and 0.15-0.25, r^2 is in the range 0.87 – 0.88; 10% of the rules involve the three parameters in the specified range; and the lift value of 5.333 is a ratio that gives us information about the increase in probability of the "then" (consequent) given the "if" (antecedent) part. The range of values for the fitness function gives a criterion for subdividing the parameter search, and the confidence is an indicator of the smoothness of the region in which a maximum value is likely to be found. Multiple regions of high value for the fitness function have to be treated separately. A rule for finding the optimum value within a decreasingly small range is the objective of the search in this space.

5. Conclusions and Indications for Further Work

We are having modest success with a quite primitive search strategy. In terms of being able to re-use expensive model simulations, we assert that this strategy is quite effective for achieving a good calibration fit to observations. In our example (Figure 6), the validation step revealed r^2 = 89%, significantly better than the manual calibration results which preceded this analysis.

Machine learning has the potential to provide feedback where more calculations are necessary. All of the simulations are still available since the simulation step and the evaluation step are decoupled. It outperforms manual calibration thus far, and appears to be the only way to effectively marry multiple watersheds. A single calibration for multiple watersheds or combinations of NPS watershed models and lake circulation is an example of multiobjective calibration. For all practical purposes, it becomes intractable: individually, we get the watershed correct and the lake reasonably correct, but when we combine them the results are generally unsatisfactory. The machine learning approach can potentially adapted as a discrete optimization problem for the multiple calibration situation.

References

Bingner, R. L. and F.D. Theurer (2007), AGNPS Web Site. *Internet* at http://www.ars.usda.gov/Research/docs.htm?docid=5199

Borgelt, Christian (2004), Apriori — Finding Association Rules/Hyperedges with the Apriori Algorithm. http://fuzzy.cs.uni-magdeburg.de/~borgelt/apriori.html

Burnash, R.J.C., R.L. Ferral, R.A. McGuire (1973), A Generalized Streamflow Simulation System - Conceptual Modeling for Digital Computers, U.S. Department of Commerce, National Weather Service and State of California, Department of Water Resources.

Eckhardt, K., and J.G. Arnold (2001), Automatic calibration of distributed catchment model. Journal of Hydrology 251. pp. 103–109.

Franchini, M., G. Galeati, S. Berra (1998), Global optimization techniques for the calibration of conceptual rainfall-runoff models. Hydrological Sciences Journal 43. pp. 443-458.

Kouwen, N., (1988) "WATFLOOD: A Micro-Computer based Flood Forecasting System based on Real-Time Weather Radar, Canadian Water Resources Journal, 13(1):62-77.

Legates, D., and McCabe, J. (1999) Evaluating the use of "Goodness of Fit" measures in hydrological and hydroclimatic model validation. Water Resources Research 35. pp. 233-241.

Sharma, V., D.A. Swayne, D.C. Lam, W. Schertzer (2006), Auto-calibration of Hydrological Models Using High Performance Computing. iEMSs: 3rd Biennial meeting of the International Environmental Modelling and Software Society, Burlington, Vermont, USA.

Sharma, V., D.A. Swayne, D.C. Lam, W. Schertzer (2006), Parallel Shuffled Complex Evolution Algorithm for Calibration of Hydrological Models. iEMSs: 3rd Biennial meeting of the International Environmental Modelling and Software Society, Burlington, Vermont, USA.

SWAT 2005 (2005), http://www.brc.tamus.edu/swat/soft_model.html

Tanakamaru, H., and S.J. Burges (1996), Application of global optimization to parameter estimation of the tank model, paper presented at Proceedings of the International Conference on Water Resources & Environment Research: Towards the 21st century, Vol. II. Kyoto, Japan. Oct. 29-31, 1996.

Tolson, B.A. and C.A. Shoemaker (2007), The dynamically dimensioned search (DDS) algorithm for computationally efficient watershed model calibration. Water Resources Research 43, 1413

Tolson, Bryan A., V. Sharma and D.A. Swayne (2007), Parallel Implementations of the Dynamically Dimensioned Search (DDS) Algorithm Proc. 6th International Symposium on Environmental Software Systems. IFIP Conference Series. 10pp.