Decompositional, Model-based Learning and its Analogy to Diagnosis

Brian C. Williams and William Millar [†]
NASA Ames Research Center, MS 269-2
Moffett Field, CA 94305 USA
E-mail: {williams, millar}@ptolemy.arc.nasa.gov

Abstract

A new generation of sensor rich, massively distributed autonomous system is being developed, such as smart buildings and reconfigurable factories. To achieve high performance these systems will need to accurately model themselves and their environment from sensor information. Accomplishing this on a grand scale requires automating the art of large-scale modeling. To this end we have developed *decompositional, model-based learning (DML)*. DML takes a parameterized model and sensed variables as input, decomposes it, and synthesizes a coordinated sequence of "simplest" estimation tasks. The method exploits a rich analogy between parameter estimation and consistency-based diagnosis. Moriarty, an implementation of DML, has been applied to thermal modeling of a smart building, demonstrating a significant improvement in learning rate.

A new generation of sensor rich, massively distributed, autonomous systems is being developed, such as networked building energy systems, autonomous space probes, and biosphere-like life support systems, that have the potential for profound environmental and economic change (Williams & Nayak 1996). To achieve high performance, these *immobile robots* will need to develop sophisticated regulatory systems that accurately and robustly control their complex internal functions. To accomplish this immobots will exploit a vast nervous system of sensors to accurately estimate models of themselves and their environment on a grand scale. Handling these large scale model estimation tasks requires high-level reasoning methods that coordinate a large set of traditional adaptive processes. Decompositional, model-based learning (DML) and its implementation Moriarty address this problem, providing a high-level reasoning method that generates and coordinates a set of nonlinear estimation codes, by exploiting a rich analogy to ATMS-based prime implicant generation(de Kleer 1986) and consistency-based diagnosis (e.g., (de Kleer & Williams 1987)).

[†]Caelum Research Corporation.

Large-scale Model-Estimation

Moriarty emerged out of work on the *Responsive Environment*, an intelligent building control system developed within the Ubiquitous Computing project at Xerox PARC, and is currently being developed to support a biosphere-like habitat for Mars.

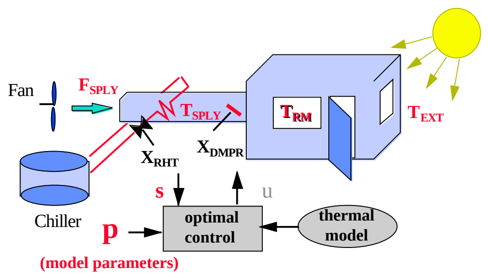

Figure 1: Office heating through an airduct.

Environmental control is made difficult by the overwhelming number of control variables, the tight coupling between control variables, and the slow response time of room temperature. Model-based control offers a natural solution, using a global model to adaptively predict where the optima lies. An informal evaluation of a non-linear, model-based controller using Xerox's responsive environment testbed suggests that energy savings in excess of 30% are conceivable (Zhang, Williams, & Elrod 1993).

The bottleneck is acquiring a model tailored to an individual building. Generic thermal models are available for complete buildings (e.g., the DOE2 simulator models), but these models are complex, containing on the order of thousands of equations and parameters for buildings with one hundred or more offices.

The technical challenge involves estimating the values of these parameters from sensor data, such as thermal conductance through walls and thermal capacity of an office's air space. An immobile robot can estimate its parameters by adjusting their values until the model best fits the sensor data. More precisely,

Definition 1 A *system model* is a set of algebraic equations $\mathbf{e}(\mathbf{c};\mathbf{v}) = \mathbf{0}$ over constants $\mathbf{c}$ and variables $\mathbf{v}$. An *embedded system model* is a system model with unknown parameters $\mathbf{p} \subset \mathbf{c}$, sensed variables $\mathbf{s} \subset \mathbf{v}$, and sensor data D. An *estimator* (for $\mathbf{p}$) is a function $f(\mathbf{x};\mathbf{p})$, where $\mathbf{x} \subset \mathbf{s}$, and such that $f(\mathbf{x};\mathbf{p}) = 0$ follows from the system model.

Given an embedded system model and estimator $f(\mathbf{x};\mathbf{p})$, we can estimate $\mathbf{p}$ by solving, for example, the non-linear optimization problem,

$$\mathbf{p}^* = arg \min_{\mathbf{p}} \sum_{\mathbf{x}_i \in D_{\mathbf{x}}} f(\mathbf{x}_i;\mathbf{p})^2.$$

A broad set of non-linear estimation codes are, of course, standard tools of the trade, used by physical scientists to extract parameters of physical phenomena. We make no contribution here to these basic algorithms. We simply note that the performance of these basic codes on non-linear physical models can quickly become inadequate as the dimension of the problem grows.

For example, consider office thermal modeling (Williams & Millar 1996) in figure 1. Heat flow into an office through a duct is regulated using a damper to control airflow via X_{dmpr}, and a radiator like device, called a reheat, to control air temperature via X_{rht}. Heat also flows from the sun, equipment, through walls and doorways. The thermal model of a single office consists of fourteen equations involving seventeen state variables and eleven parameters. About a third of the equations are nonlinear, such as,

$$F_{dmpr} = \left(\frac{\rho_{dmpr}(X_{dmpr})}{R_{dct}}\right)\sqrt{P_{dct}},$$

which relates air flow through the damper to duct pressure and duct air resistance as a function of damper position. Nine of the state variables are sensed, including temperature T, flow rate F, air pressure P, damper and reheat valve position X. Seven of the eleven parameter values are unknown and must be estimated. Estimating all seven parameters at once requires solving a 7-dimensional, nonlinear optimization problem involving a multi-modal objective space. Using arbitrary initial values, a Levenberg-Marquardt algorithm was applied repeatedly to this problem, but consistently became lost in local minima and did not converge after several hours.

Estimation Plans

To estimate parameters of higher dimensional systems a skilled modeler massages the physical equations into

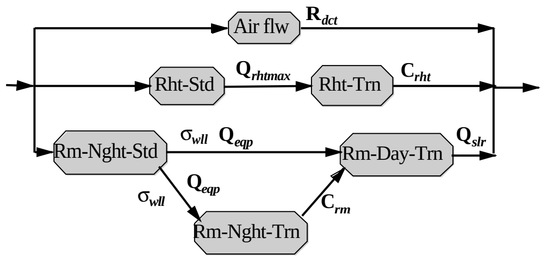

Figure 2: Model-estimation plan for a single office.

a tractable set of smaller parameter estimation tasks, coordinates their invocation and combines their results. For large modeling tasks that are an international priority, such as earth ecosystem modeling, an army of modelers can be employed at great cost. For the important, but more modest tasks performed by immobile robots, this army must be automated if high performance and robustness are to be achieved. The open research problem that we address through Moriarty is to automate the modeler's process of synthesizing these estimation plans. Moriarty automates key aspects of how a community of modelers decomposes, simplifies and coordinates large scale model estimation tasks, and generates code to perform the parameter estimation.

The problem of decomposing and coordinating estimation tasks is not one that has received significant attention in the statistics or machine learning literature. The classical estimation literature offers a vast number of techniques, many available as subroutines in Matlab or Splus, for online and offline estimation of various families of linear and nonlinear estimation problems, using a variety of different objective criteria. However these techniques do not generate the codes that setup and coordinate these subroutines. The Bayesian learning community makes extensive use of graphical models to strongly bias estimation, hence improving convergence (Buntine 1994; Spiegelhalter & Lauritzen 1990) and qualitative reasoning (QR) work exploits monotonicity constraints as a bias (Kay & Ungar 1993). However, research, for example, in graphical models and QR is only beginning to look at the model decomposition process (Shachter, Anderson, & Poh 1990; Stolle & Bradley 1996).

An estimation plan for the office model is shown above in figure 2. Each octagon in the diagram is an *estimation action* that is defined by an estimator of the form $y = f(\mathbf{x};\mathbf{p})$, applied to a subset of the sensors, parameters, and sensor data. The arc labels identify those parameters whose values are estimated in the preceding action. For example, the top octagon labeled "Air flw" produces an estimate for parameter

R_{dct}, the air resistance in the duct, using the estimator,

$$F_{ext} = (\rho_{lkg} + \rho_{dmpr}(X_{dmpr})) \frac{\sqrt{P_{dct}}}{R_{dct}}$$

derived from three equations in the office model pertaining to airflow. Note again that an estimator contains only *sensed* variables. This action exploits previously estimated parameters as constants (ρ_{lkg} and the coefficients of function $\rho_{dmpr}(X_{dmpr})$). The six estimators in the plan each contain only one or two unknown parameters, reducing the original seven dimensional problem to several problems of at most two dimensions.

Generating Estimation Plans

Consider a common sense account of the basic steps Moriarty uses to generate an estimation plan. First, Moriarty generates a set of possible estimation actions from the model. It then selects a subset of these actions sufficient to estimate all parameters, and coordinates the passing of estimated parameter values between these actions. Possible actions are generated in two steps: decomposition and simplification.

In the decomposition step, subsets of the model and sensors are identified that are sufficiently constrained to define an estimation problem, and are each turned into an estimator. For example, the estimator for the "Air flw" action described above was derived from the following air flow equations in $\mathbf{e}(\mathbf{c};\mathbf{v})$,

$$\begin{aligned} F_{ext} &= F_{lkg} + F_{dmpr}, \\ F_{lkg} &= \left(\frac{\rho_{lkg}}{R_{dct}}\right)\sqrt{P_{dct}}, \\ F_{dmpr} &= \left(\frac{\rho_{dmpr}(X_{dmpr})}{R_{dct}}\right)\sqrt{P_{dct}}. \end{aligned}$$

In the thermal example, Moriarty's decomposition step automatically generates eight possible estimation actions in total, each containing as few as one parameter or as many as all seven. Three of these actions are "Air flw" (with single parameter R_{dct}), "Rht" (with two parameters Q_{rhtmax} and C_{rht}), and "Rm" (with four parameters, σ_{wll}, Q_{eqp}, C_{rm} and $Q_{slr}(t)$). It is purely coincidental that the parameter sets for these actions are disjoint.

These three actions are sufficient to estimate all seven model parameters. If we were to produce an estimation plan based on this alone, the result would have a single action for each of the three primary parallel paths in figure 2. While any subset of the possible actions that estimate all seven parameters could be chosen to form a plan, the above three actions contain the fewest parameters individually.

The simplification step is optional, and is not developed here technically; however, we summarize its result here, as it makes the coordination step of our example more interesting. This step produces one or more simplified versions of each estimation action that contain fewer parameters, by identifying conditions on the data such that influences by one or more parameters become negligible. For example, consider the estimator for action "Rm":

$$\begin{aligned} \frac{dT_{rm}}{dt} &= \frac{C_0 F_{sply}\,(T_{sply} - T_{rm})}{C_{rm}} + \\ &\qquad \frac{Q_{eqp} + Q_{slr}(t) + \sigma_{wll}\,(T_{ext} - T_{rm})}{C_{rm}} \end{aligned}$$

This estimator is simplified by noticing that solar effect $Q_{slr}(t)$ is negligible at night time, when the sun is down (action "Rm-Nght"), while it is significant during the day (action "Rm-Day-Trn"). Action "Rm-Nght" is generated from "Rm" by restricting the data set to data taken at night. This allows $Q_{slr}(t)$ to be eliminated, reducing the number of parameters in the estimator from four to the three parameters σ_{wll}, Q_{eqp} and C_{rm}. Six additional estimators are generated – "Rm-Nght-Std", "Rm-Nght-Trn", "Rm-Day-Std", "Rm-Day-Trn", "Rht-Std" and "Rht-Trn" – by applying similar simplifications to estimators "Rm-Nght", "Rm-Day" and "Rht" that focus on transient and steady state behaviors. Six of the resulting estimators are those appearing in the plan of figure 2. Note that these simplified estimators are currently generated manually. The algorithm we are currently developing to automate this simplification step is based on *caricatural modeling*, described in (Williams & Raiman 1994).

Having generated these possible estimation actions, the coordination step selects and sequences a subset of the estimation actions to further simplify the search performed by each action. The parameter value estimated by an action, such as "Rm-Nght-Std", can be fed as a constant to a later action, such as "Rm-Day-Trn", reducing the dimensionality of the later action. Alternatively, an estimated parameter value can be exploited as an *initial* bias to the later action, by constraining its random restarts for the parameter based on the confidence in the previous estimate. The first policy more dramatically reduces the dimensionality of later actions, while the second policy can enable more refined estimates. We are currently evaluating the trade-offs. The plan in figure 2 treats passed parameter estimates as constants in later actions. For example, "Rm-Nght-Trn" is left only with C_{rm} as an unknown parameter.

Moriarty generates a "simplest" estimation plan using a greedy algorithm. This algorithm determines the next action to perform as follows: first, the action must have a minimal (but positive) number of unknown parameters; second, if two estimators have equal numbers of new parameters, then the action is selected that contains the fewest number of sensed variables. The first condition drives the convergence time downwards by reducing the dimensionality of the search space, while the second condition improves accuracy, by reducing the combined sensor error introduced.

The decomposition step automatically generates eight estimators F1-F8, characterized later in the experiments section. Invoking the greedy algorithm on these estimators generates a plan with three parallel actions, "Air-flw", "Rht" and "Rm", with one, two and four parameters, respectively. More interestingly, given the six estimators mentioned above under decomposition and simplification, the greedy algorithm generates the plan shown in figure 2, with each estimator containing at most two unknown parameters. The remaining sections turn to the details of decomposition.

Model Decomposition

The decomposition step determines a set of possible estimation actions that can be coordinated within an estimation plan. Estimation actions are generated by decomposing the model into a set of potentially overlapping sub-models, called *dissents*, and then by generating an estimator from each dissent. In this section we develop DML's concepts of *dissent* and *support*, and the role they play in the generation of estimation actions. As discussed later, dissent and support are inspired by the consistency-based diagnosis concepts of *conflict* and *environment*, respectively (de Kleer & Williams 1987; de Kleer, Mackworth, & Reiter 1992).

The office example is too complex for pedagogical purposes. Instead we introduce a modification of the standard polycell example from diagnosis. This example (figure 3) consists of five equations, one for each of the three multipliers M1, M2, M3 and two adders A1, A2. M1, for example, denotes the equation $x = a \times b$. The goal is to estimate unknown parameters b, d, f, s and t given sensed variables a, c, e, u and v. Variables x, y and z are dependent.

Recall that an estimation action solves an optimization problem that minimizes *disagreement* between a subset of the observables and a subset of the model. For a disagreement to exist the submodel must be *overdetermined*, given the sensed variables; that is, there must be at least two ways of uniquely determining the same value. We call an overdetermined submodel a *dissent*, and a submodel that uniquely determines a variable its *support*. More precisely,

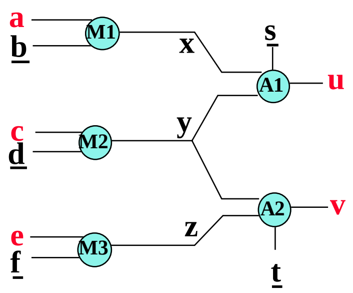

Figure 3: Polycell with $\mathbf{e} = \{M1, M2, M3, A1, A2\}$, $\mathbf{p} = \{b, d, f, s, t\}$ and $\mathbf{s} = \{a, c, e, u, v\}$.

Definition 2 Given a system model $S = \langle \mathbf{e}, \mathbf{s} \rangle$ with equations $\mathbf{e}(\mathbf{c}; \mathbf{v}) = \mathbf{0}$ and sensed variables $\mathbf{s}$, a *dissent* of S is a subsystem $\langle \mathbf{e_d}, \mathbf{s_d} \rangle$ of S, such that $\mathbf{e_d}(\mathbf{c}; \mathbf{v}) = \mathbf{0}$ is overdetermined given $\mathbf{s_d}$. $\langle \mathbf{e_d}, \mathbf{s_d} \rangle$ is a *minimal dissent* if no proper subsystem $\langle \mathbf{e}', \mathbf{s}' \rangle$ of $\langle \mathbf{e_d}, \mathbf{s_d} \rangle$ exists such that $\langle \mathbf{e}', \mathbf{s}' \rangle$ is a dissent of S.

Definition 3 Given system model $S = \langle \mathbf{e}, \mathbf{s} \rangle$ with variables $\mathbf{v}$, a *support* for variable $v_s \in \mathbf{v}$ is a subsystem $\langle \mathbf{e}', \mathbf{s}' \rangle$ of S, such that $\langle \mathbf{e}', \mathbf{s}' \rangle$ determines v_s, and no proper subsystem of $\langle \mathbf{e}', \mathbf{s}' \rangle$ determines v_s.

Note that a pair of support for variable v_s provides two means of determining v_s. Hence the union of the pair overdetermine v_s, and constitutes a dissent $\langle \mathbf{e}_{s1} \cup \mathbf{e}_{s2}, \mathbf{s}_{s1} \cup \mathbf{s}_{s2} \rangle$. We exploit this fact in the next section to generate dissents from a set of support.

The goal of decomposition is to generate a set of "simplest" estimation actions. A minimal dissent captures the intuition of "simplest" subproblem for two reasons. First, minimality increases convergence rate, by minimizing the number of parameters appearing in each dissent's equations $\mathbf{e}(\mathbf{c}; \mathbf{v}) = \mathbf{0}$, and hence the dimensionality of the optimization problem. In addition, minimality improves accuracy by minimizing the number of sensed variables $\mathbf{s}$ per estimation, and hence the amount of noise introduced.

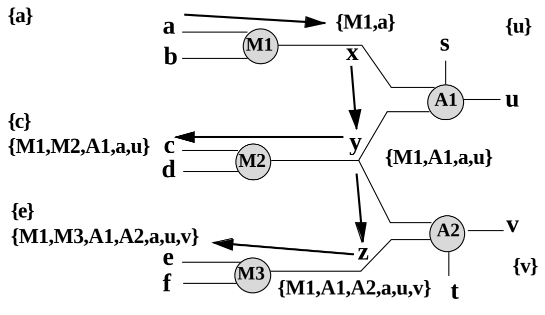

Figure 4: Examples of support in Polycell.

Returning to the example, figure 4 shows several examples of support for polycell. Note first the trivial support at each sensed variable: a sensed variable is its own support. Next, beginning with trivial support at a, u and v, and moving out towards c and e, a support is provided for each variable along the way. For example, y has support $\{M1, A1, a, u\}$. Note also that c and e each have pairs of support. The union of each pair comprises a (minimal) dissent, D1 and D2 respectively. A final dissent, not shown, results in D1–D3:

$$\begin{array}{lrcl} \text{D1:} & \{M1, M2, A1, a, c, u\} & \Rightarrow & b, d, s \\ \text{D2:} & \{M1, M3, A1, A2, a, e, u, v\} & \Rightarrow & b, f, s, t \\ \text{D3:} & \{M2, M3, A2, c, e, v\} & \Rightarrow & d, f, t \end{array}$$

To the left of the arrow is the set of equations and sensed variables for each dissent. To the right is the set of parameters mentioned in the dissent's equations, hence, those that can be estimated using that dissent.

Next, defining an estimation action represented by a minimal dissent $\langle \mathbf{e_d}, \mathbf{s_d} \rangle$ involves generating an estimator $\mathbf{y} = f(\mathbf{x}; \mathbf{p})$ and setting up a corresponding optimization problem. For example, dissent D1 consists of equations M1, M2, A1:

$$\begin{aligned} x &= a \times b \\ y &= c \times d \\ u &= x + y + s. \end{aligned}$$

Solving for u in D1 produces $u = a \times b + c \times d + s$, resulting in the estimation action,

$$\mathbf{p}^* = arg \min_{\mathbf{p}} \sum_{\langle a_i, c_i, u_i \rangle^T \in D} (u_i - [b \times a_i + d \times c_i + s])^2$$

where the parameters are $\mathbf{p} = \langle b, d, s \rangle^T$.

As discussed in the next section, to address tractability, Moriarty only generates those dissents composed of equations that are free of simultaneities. Solving this set of equations, to generate an estimator, is straightforward (see (Williams & Millar 1996)).

The Support Maintenance System

To generate support and (minimal) dissents we need conditions for identifying when a subsystem is uniquely determined or (minimally) over-determined, respectively. An appropriate choice of conditions hinges upon the computational price one is willing to pay.

A standard presumption, made by causal ordering research (e.g., see (Iwasaki & Simon 1986)), and frequently used for analyzing models of nonlinear physical systems, is that n independent model equations and exogenous variables uniquely determine n unknowns.

Assumption 1 Given system $\langle \mathbf{e}, \mathbf{s} \rangle$ with variables $\mathbf{v}$, let $\langle \mathbf{e}', \mathbf{s}' \rangle$ be any of its subsystems, $\mathbf{v}'$ its variables, $n = |\mathbf{e}'|$, $m = |\mathbf{s}'|$ and $l = |\mathbf{v}'|$. We assume that subsystem $\langle \mathbf{e}', \mathbf{s}' \rangle$ is (a) *overdetermined* if $n+m > l$, (b) *dissenting* if $n + m = l + 1$, (c) *uniquely determined* if $n + m = l$, and (d) *underdetermined* if $n + m < l$.

Note that this condition holds universally for linear systems, and is true of many *physical* systems.

The power of this condition is that it is trivial to evaluate; for example, it is far easier than identifying minimal dissents with a series of subset tests, or solving the system of equations. In addition, the condition doesn't require knowledge of the form of an equation, just the variables each equation interacts with. Moriarty uses this condition to decompose a graph of interactions into estimation subproblems, without restriction on, or further knowledge about, the form of the underlying equations. This is in the spirit of graphical learning methods, such as (Buntine 1994; Shachter, Anderson, & Poh 1990).

To generate minimal dissents, recall that a dissent is the union of two support for some variable. Hence minimal dissents can be generated from a set of support, by computing all such unions, and selecting those that satisfy condition (b). Furthermore, if the equations of the system are invertible, then it follows trivially that all dissents can be generated just from the support of the sensed variables $\mathbf{s}$.

To generate support we exploit an analogy to the concept of *prime implicant*, from propositional logic. A (theory) implicant for proposition p is a set of literals whose conjunction entails p, given propositional theory T. The implicant is prime if the set of literals is minimal under subset. While a support determines a variable's value, an implicant entails a proposition. The sensed variables of a support are analogous to the literals of the implicant, and the equations of a support are analogous to the subset of the theory that, together with the implicant, entails p.

An assumption-based truth maintenance system (de Kleer 1986) is an example of a restricted form of prime implicant algorithm that has proven useful in practice. The ATMS generates implicants that are comprised of a distinguished set of literals, called *assumptions,* given a theory that is horn. The implicants generated by the ATMS are referred to as *environments.*

Moriarty generates candidate support by exploiting the analogy to ATMS environments, and then uses condition (c) to quickly test that the candidate is uniquely determined, and hence constitutes a support. ATMS environments are generated by propagating them locally through a network of horn clauses, starting at the literals denoting assumptions, combining and then propagating a new set of environments after each traversal of a clause. Analogously, Moriarty generates a support by propagating them locally through the network of equations, starting at the sensed variables, combining and then propagating a new set of support after each traversal of an equation. A single path of propagation for polycell is shown in figure 4.

The propagation algorithm is called a support maintenance system (SMS), to highlight its analogy to an ATMS. The SMS uses the function *CreateDecomposition* to kick off propagation, by adding to each sensed variable v a trivial support $\{v\}$, indicating that v is independent. For polycell, this type of support is added to sensed variables a, c, e, u, and v.

```
CreateDecomposition($\langle \mathbf{e}, \mathbf{s} \rangle$)
  /* system model $\langle \mathbf{e}, \mathbf{s} \rangle$ */
  Initialize dissents to empty
  for each $s_i \in \mathbf{s}$
    $trivialSupport = \{\{\}, \{s_i\}\}$
    AddSupport($s_i$,trivialSupport)
  return dissents
```

After recording a new support, *AddSupport* tries to propagate the new support through successive local equations (partial results for propagating from a are shown in figure 4). For example, support $\{a\}$ propagates through $M1$ to x, creating new support $\{M1, a\}$. In addition, if a non-trivial support is being added to a sensed variable, then it is turned into a dissent. For example, when the support $\{M1, M2, A1, a, u\}$ is added to sensed variable c, the dissent $\{M1, M2, A1, a, c, u\}$ is generated.

```
AddSupport($v$,$\langle \mathbf{e}, \mathbf{s} \rangle$)
  /* variable $v$, support $\langle \mathbf{e}, \mathbf{s} \rangle$ */
  Add $\langle \mathbf{e}, \mathbf{s} \rangle$ to the support of $v$
  for each $e_i \in$ set of equations involving $v$
    if $e_i \notin \mathbf{e}$ then
      for each $co \in$ CausalOrientations($e_i$,$v$)
        Propagate($v$,$\langle \mathbf{e}, \mathbf{s} \rangle$,$co$)
  if $\mathbf{e} \neq \{\}$ and Sensed?($v$) then
    Add $\langle \mathbf{e}, \mathbf{s} \cup \{v\} \rangle$ to dissents
```

Propagation requires an equation that acts as a conduit, a variable to be propagated to (called the *effect*) and a set of variables that support this effect (called the *cause*). The function *CausalOrientations* generates the set of all ways the equation e can be oriented about a cause variable v to determine a new effect variable y, from complementary, cause variables $\mathbf{x}$. For polycell, given cause variable x and equation $A1$, y or u may be selected as effects, with the remaining variable and x acting as the cause.

```
CausalOrientations($e$,$v$)
  /* equation $e$, cause variable $v$ */
  $\mathbf{v}$ = set of variables in $e$
  return $\{\langle y, e, \mathbf{x} \rangle | y \in \mathbf{v}, \mathbf{x} = \mathbf{v} - \{y\}, v \in \mathbf{x}\}$
```

The core of the algorithm is *Propagate*, which passes a new support through an equation e to effect y. It uses the function *WeaveSupport* to take the union of a newly added support to a variable v, with a support for each of the other causes ($\mathbf{x} - \{v\}$), producing a composite subsystem c. Within WeaveSupport, S denotes a set of new support, S_2 is the set of support for one of the other cause variables, h, and S_c represents the set of all composite support constructed. Propagate then adds e to each composite c in S_c to produce candidate support S_w for effect v. For example, suppose Propagate is given new support $\{M1, a\}$ for x, and equation $A1$, with effect y and causes $\{u, x\}$. WeaveSupport selects the support $\{u\}$ for u, combines this with the new support to produce $\{M1, a, u\}$. Propagate then adds $A1$ to produce the new support $\{M1, A1, a, u\}$ for y. The effects of each propagate are indicated by arrows in figure 4.

Note that there are cases where a support being generated by Propagate or WeaveSupport is thrown away. First, to avoid circularities, propagate will not propagate through an equation mentioned in the new support. Likewise, it will not add a composite support to an effect if the effect is mentioned in that support. For WeaveSupport note that the union of a set of support may either be uniquely determined or overdetermined, depending on the sharing of variables and equations between the support being combined. The later case is ruled out by the "Overdetermined?" test, in the construction of S'_c.

```
Propagate($v$,$\langle \mathbf{e}, \mathbf{s} \rangle$,$\langle y, e, \mathbf{x} \rangle$)
  /* equation $e$, $y$ its effect, $\mathbf{x}$ its causes,
    $v \in \mathbf{x}$, support $\langle \mathbf{e}, \mathbf{s} \rangle$ */
  if $e \notin \mathbf{e}$ then
    $S_w$ = WeaveSupport($v, \{\langle \mathbf{e}, \mathbf{s} \rangle\}, e, \mathbf{x}$)
    for each $\langle \mathbf{e}_w, \mathbf{s}_w \rangle \in S_w$
      if $y \notin$ set of variables in $\mathbf{e}_w$ then
        AddSupport($y$,$\langle \mathbf{e}_w \cup \{e\}, S_w \rangle$)
end
```

```
WeaveSupport($v$,$S$,$e$,$\mathbf{x}$)
  /* equation $e$, its causes $\mathbf{x}$, $v \in \mathbf{x}$,
    & its supporters $S$ */
  if $\mathbf{x}$ is empty, then
    return $S$
  else
    $h$ = a variable in $\mathbf{x}$
    $R = \mathbf{x} - \{h\}$
    if $h = v$, then
      return WeaveSupport($\phi$,$S$,$e$,$R$)
    else
      $S_2 = \{\langle \mathbf{e}, \mathbf{s} \rangle | \langle \mathbf{e}, \mathbf{s} \rangle \in$ Support($h$)$, e \notin \mathbf{e}\}$
```

```
$S_c = \{\langle \mathbf{e} \cup \mathbf{e}_2, \mathbf{s} \cup \mathbf{s}_2 \rangle |$
        $\langle \mathbf{e}, \mathbf{s} \rangle \in S, \langle \mathbf{e}_2, \mathbf{s}_2 \rangle \in S_2\}$
$S'_c = \{s | s \in S_c, \neg \mathit{Overdetermined?}(s)\}$
**return** $\mathit{WeaveSupport}(v, S'_c, e, R)$
```

The SMS is sound, since after weaving support the SMS checks and records support only if they are uniquely determined according to condition (c). With respect to completeness, we return to the analogy. For an ATMS, the set of environments generated by local propagation is incomplete for general clausal theories. However, it is complete for horn clause theories. Analogously, if a support is a system of simultaneous equations, then it will not be identified through the above local propagation algorithm, since the simultaneity represents a codependence between variables. The algorithm does, however, generate all support that are *simultaneity-free*. Soundness and completeness are evaluated further in (Williams & Millar 1996).

Analogy to Diagnosis

In addition to the analogy between support and prime implicants, there exists a strong analogy between model-estimation and consistency-based diagnosis, and their respective decomposition methods. Model-estimation and consistency-based diagnosis problems identify a set of *parameter values* and *modes*, respectively, that *minimize disagreement* between the model and the observables. In diagnosis, each mode is discrete and finite. Treating polycell as a diagnosis problem, the mode variables are M1, M2, M3, A1 and A2, each with the two modes OK and not-OK. In estimation, parameter values can be a mixture of continuous and discrete.

Both problems involve measuring a disagreement between model and observables. In estimation, the disagreement is usually a continuous error defined by a *Euclidean metric.* The error is to be minimized by the estimated parameters. In consistency-based diagnosis, logical disagreement can be framed using a *discrete metric*, which treats the distance between two values u and v as 1 unless they are equal, in which case the distance is 0. Diagnosis amounts to searching for a set of modes that bring the disagreement between model and observables to 0.

Turning to the decomposition process, a simplest diagnostic subproblem is a minimal subset of the component models and observed values that are inconsistent. This is loosely the concept of a *minimal conflict,* which is a minimal set of components, whose nominal models disagree with the observables (de Kleer & Williams 1987).* For example, assuming polycell has the values $a = 3$, $b = 2$, $c = 3$, $d = 2$, $e = 3$, $f = 2$, $u = 10$ and $v = 12$, then there are two minimal conflicts, $\{M1, M2, A1\}$, and $\{M1, M3, A1, A2\}$.

*The subtle, although important, difference is that a conflict doesn't include the subset of observed values that are involved in the inconsistency.

Shifting to estimation, we replace logical inconsistency (a conflict) with a system being overdetermined (a dissent). There is an important distinction between conflict and dissent. The inconsistency indicated by a conflict is unequivocal, while a dissent merely indicates the potential for error, hence, our use of a more mild term – "dissent" – in naming this form of disagreement. For example, polycell has three dissents, but only two conflicts. There is no conflict corresponding to dissent D3. The reason is that the potential for disagreement in D3, is not realized for the particular values assigned to the sensed variables in the diagnosis example, hence no conflict exists.

As we've already seen, this analogy continues down into the concepts of support, environment and their respective generation. This suggests the opportunity for developing a rich unification of large-scale model-based learning and diagnostic methods.

Experiments and Discussion

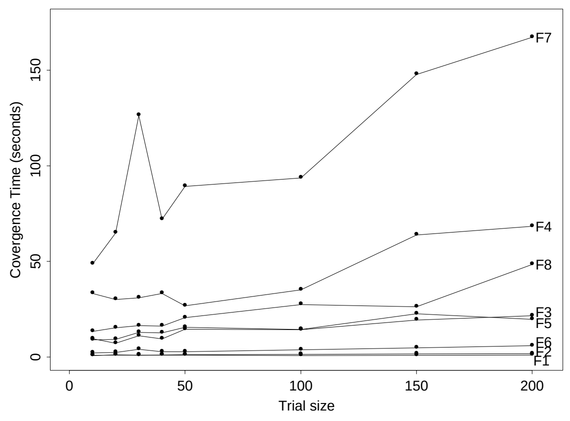

Figure 5: Convergence rate vs data size for the generated estimators F1–F8.

Moriarty implements all of DML except for the optional simplification step, using Mathematica for symbolic manipulation and Splus for statistical analysis. Consider Moriarty's performance on the office thermal example (figure 5) using dissent generation, but without the simplification step. Moriarty generates eight estimators, F1 – F8, in the decomposition step, and sequences them to produce $\langle F2, F1, F6 \rangle$, which correspond to the "Air-Flw," "Rht," and "Rm" estimation actions. We compare the performance of the eight estimators by running them against sensor data sets ranging in size from 10 to 200, shown above. The y axis denotes time required to converge on a final estimate of parameters involved in the dissent. The plot labeled F7 is for the original seven dimensional estimator, while plots F2, F1, and F6 are the three estimators in Moriarty's sequence. Higher dimensional estimators, like F7, tend to fail to converge given arbitrary initial conditions, hence ball-park initial parameter estimates were supplied to allow convergence in the higher dimensional cases. Decomposition leads to significant speed-up even when good initial estimates were available. For example, at trial size 200, the original estimator requires 166 seconds, while the total time to estimate all parameters using F2, F1, and F6 is under 9 seconds. This represents a speed up by a factor of 14.

In addition the sequence requires less data to converge. This is important for self-modeling systems that use online estimation to quickly track time-varying parameters. Employing the rule of thumb that the data set size should be roughly ten fold the dimension of the parameter space, F7 would require around 70 data points, while the F2, F1, F6 sequence requires only 40. The anomalous slow down in the convergence rate of F7 at 25 data points is attributed to insufficient data. Finally, although not included, parameter accuracy, measured by the confidence interval of each parameter is also improved using the generated sequence.

To summarize, a model-based approach is essential for regulating systems of the size of most immobile robots. Embodying an immobile robot with self-modeling capabilities requires the use of symbolic reasoning to coordinate a large set of autonomic estimation processes. This coordination should mimic the way in which a community of modelers decompose, simplify and coordinate modeling problems on a grand challenge scale. DML automates one aspect of this rich model decomposition and analysis planning process, by exploiting an analogy between model estimation, prime implicant generation and consistency-based diagnosis, and by introducing the concepts of dissent, support, and support maintenance system, analogous to conflicts, environment and the ATMS. Turning to the future, we are beginning to develop Moriarty to support a "biosphere-like" habitat, called a closed loop ecological life support system, and an *insitu* propellant plant for NASA's Mars exploration program.

Acknowledgements

Thanks to Sonia Leach and the Responsive Environment team at Xerox PARC – Zhang Ying, Scott Elrod and Joseph O'Sullivan for their support. Jim Kurien and Pandu Nayak provided valuable feedback.

References

Buntine, W. L. 1994. Operations for learning with graphical models. *Journal of Artificial Intelligence Research* 2:159–225.

de Kleer, J., and Williams, B. C. 1987. Diagnosing multiple faults. *Artificial Intelligence* 32(1):97–130.

de Kleer, J., and Williams, B. C. 1989. Diagnosis with behavioral modes. In *Proceedings of IJCAI-89*, 1324–1330.

de Kleer, J.; Mackworth, A.; and Reiter, R. 1992. Characterizing diagnoses and systems. *Artificial Intelligence* 56:197–222.

de Kleer, J. 1986. An assumption-based TMS. *Artificial Intelligence* 28(1):127–162.

Iwasaki, Y., and Simon, H. 1986. Theories of causal ordering: Reply to de Kleer and Brown. *Artificial Intelligence* 29:63–72.

Kay, H., and Ungar, L. 1993. Deriving monotonic function envelopes from observations. *Seventh International Workshop on Qualitative Reasoning.*

Shachter, R.; Anderson, S.; and Poh, K. 1990. Directed reduction algorithms and decomposable graphs. In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, 237–244.

Spiegelhalter, D., and Lauritzen, S. 1990. Sequential updating of conditional probabilities on directed graphical structures. *Networks* 20:579–605.

Stolle, R., and Bradley, E. 1996. Automatic construction of accurate models of physical systems. *Annals of Mathematics and Artificial Intelligence.*

Williams, B. C., and Millar, B. 1996. Automated decomposition of model-based learning problems. In *Proceedings of the Tenth International Workshop on Qualitative Reasoning, AAAI Technical Report WS-96-01*, 265–273.

Williams, B. C., and Nayak, P. P. 1996. Immobile robots: AI in the new millennium. *AI Magazine* 17(3):16–35.

Williams, B. C., and Raiman, O. 1994. Decompositional modeling through caricatural reasoning. In *Proceedings of AAAI-94*, 1199–1204.

Zhang, Y.; Williams, B. C.; and Elrod, S. 1993. Model estimation and energy-efficient control for building management systems. Technical report, Xerox PARC.