Influence-Based Model Decomposition *

Christopher Bailey-Kellogg
Dartmouth College
6211 Sudikoff Laboratory
Hanover, NH 03755
cbk@cs.dartmouth.edu

Feng Zhao
Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304
zhao@parc.xerox.com

Abstract

Recent rapid advances in MEMS and information processing technology have enabled a new generation of AI robotic systems — so-called Smart Matter systems — that are sensor rich and physically embedded. These systems range from decentralized control systems that regulate building temperature (smart buildings) to vehicle on-board diagnostic and control systems that interrogate large amounts of sensor data. One of the core tasks in the construction and operation of these Smart Matter systems is to synthesize optimal control policies using data rich models for the systems and environment. Unfortunately, these models may contain thousands of coupled real-valued variables and are prohibitively expensive to reason about using traditional optimization techniques such as neural nets and genetic algorithms. This paper introduces a general mechanism for automatically decomposing a large model into smaller subparts so that these subparts can be separately optimized and then combined. The mechanism decomposes a model using an influence graph that records the coupling strengths among constituents of the model. This paper demonstrates the mechanism in an application of decentralized optimization for a temperature regulation problem. Performance data has shown that the approach is much more efficient than the standard discrete optimization algorithms and achieves comparable accuracy.

Introduction

The new-generation sensor-rich AI robotic systems present a number of challenges. First, these systems must reason about large amounts of sensor data in real time. Second, they must construct and reason about large models of themselves and the environment in order to rapidly determine optimal control response (Williams & Nayak 1996). This paper describes an efficient computational mechanism to automate one of the major tasks in reasoning about distributed physical systems: decomposition of large models for these systems into a set of submodels that can be separately optimized and then combined.

This paper makes important contributions to qualitative and model-based reasoning in two ways. (1) The paper introduces a novel graph formalization for a model decomposition problem upon which powerful graph partitioning algorithms can be brought to bear. The graph formalism is applicable to a large number of problems where the dependency information in a model of a distributed system can be either derived from numerical trial data or reconstructed from analytic descriptions commonly used in science and engineering. (2) The paper develops two efficient partitioning algorithms to decompose a large model into submodels. The first algorithm employs spectral partitioning to maximize intra-component dependencies (called the influences) while minimizing inter-component dependencies. The second algorithm determines weakly coupled groups of model components by systematically and efficiently examining the structural similarities exhibited by trial partitions. To illustrate the utility of these algorithms, this paper applies the decomposition algorithms to a distributed thermal control problem. Performance data has confirmed that the model decomposition algorithms have yielded an efficient control optimization algorithm that outperforms standard optimization algorithms such as genetic algorithms and simulated annealing. Our optimization algorithm exploits the locality in the decomposition to attain efficiency and is able to generate solutions that are interpretable in terms of problem structures. These contributions significantly extend our previous work (Bailey-Kellogg & Zhao 1998) on qualitative models and parametric optimization for large distributed physical systems.

Other researchers in qualitative reasoning, Bayesian nets, and image processing have also investigated the problem of using decomposition to efficiently model complex physical systems. Williams and Millar developed a decomposition algorithm for parameter estimation that determines for each unknown variable in a model a minimally overdetermined subset of constraints (Williams & Millar 1996). The algorithms of this paper identify similar dependencies among nodes of a net either from a constraint net or directly from numerical data, and then partition the dependency graph into nearly decoupled subsets. Clancy introduced an

* This paper also appears in the Proceedings of AAAI 1999.

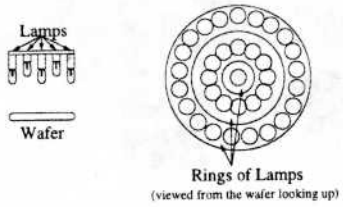

Figure 1: Rapid thermal processing for semiconductor manufacturing maintains a uniform temperature profile by independent control to separate rings of heat lamps.

algorithm for generating an envisionment of a model expressed as a qualitative differential equation, once a partition of the model is given by the modeler (Clancy 1997). Our influence-based decomposition algorithms can produce the model partitions required by Clancy's algorithm. Recent work in image segmentation has introduced measures of dissimilarity to decompose images, based on pixel intensity differences (Shi & Malik 1997; Felzenszwalb & Huttenlocher 1998). Friedman et al. in probabilistic reasoning have introduced a method to decompose a large Bayesian belief net into weakly-interacting components by examining the dependency structure in the net (Friedman, Koller, & Pfeffer 1998).

Many scientific and engineering applications have exploited similar insights in order to divide and conquer large computational problems. In the well-studied N-body problem, the interactions among particles are classified into near and far field so that they can be decomposed into a hierarchy of local interactions to achieve a linear-time speed-up (Zhao 1987). In engineering computation, domain decomposition techniques (Chan & Mathew 1994) have been developed to separately simulate submodels of large models, based on *connectivity* in the models. This paper utilizes a similar insight to formalize the task of model decomposition based on *influences* in the models. Furthermore, our approach explicitly represents the physical knowledge and structures that it exploits, so that higher-level reasoning mechanisms have an explainable basis for their decisions.

Problem Description

We develop the influence-based decomposition mechanism for large models typically arising from distributed sensing and control problems. For example, consider a distributed thermal regulation system for rapid thermal processing in semiconductor curing, where a uniform temperature profile must be maintained to avoid defects (Figure 1). The control strategy is decentralized, providing separate power zones for three rings of heat lamps (Kailath & others 1996). As a similar example, rapid prototyping in thermal fabrication can employ moving plasma-arc heat sources to control the temperature of parts to be joined (Doumanidis 1997).

Abstracting these real-world applications, this paper adopts as a running example the generic problem of decentralized temperature regulation for a piece of material. The temperature distribution over the material must be regulated to some desired profile by a set of individually controlled point heat sources. Many sensor-rich systems employ such decentralized control that accomplishes global objectives through local actions in order to ensure adaptivity, robustness, and scalability. For example, a smart building regulates its temperature using a network of sensors and actuators; decentralized control allows the network to overcome failures in individual control elements and to scale up without incurring exponential complexity. Rapid synthesis of optimal control policies for these distributed systems requires efficient methods for reasoning about large coupled models.

In particular, we focus on the control placement design task: to determine the number and location of heat sources, subject to a variety of structural constraints (e.g. geometry, material properties, and boundary conditions) and performance constraints (e.g. maximum output and maximum allowed error). While we only consider the placement design here, we have also studied the parametric optimization (in this example, the actual heat output) and reported it elsewhere (Bailey-Kellogg & Zhao 1998). The engineering community has applied various discrete optimization techniques (e.g. genetic algorithms in (Dhingra & Lee 1994) and simulated annealing in (Chen, Bruno, & Salama 1995)) to the control placement design problem. In contrast to these techniques, we seek to use domain knowledge to extract and exploit qualitative structural descriptions of physical phenomena in the design process. This yields a principled method for reasoning about designs and design trade-offs, based on an encapsulation of deep knowledge in structures uncovered for a particular problem. This in turn supports reasoning about and explanation of the design decisions.

The control placement design task will be used as a specific example illustrating our general mechanism for partitioning distributed models; the discussion section further discusses the generality of our approach. The goal here will be to design a placement that aids parametric optimization, by placing controls so that they minimally interfere with each other. This is particularly appropriate for applications where the placement design is performed once, and the parametric design is performed repeatedly (e.g. for various desired temperature profiles). The design approach taken here is to decompose a problem domain into a set of *decoupled, atomic* subregions, and then independently design controls for the separate subregions. Regions are considered decoupled if the exact control design in one region is fairly independent of the exact control design in another. A region is considered atomic if it needs no further decomposition — control design for the region yields adequate control of the region. Influence-based model decomposition provides a powerful high-level mechanism for achieving such designs.

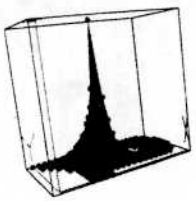

Figure 2: Thermal hill for a heat source.

Influence Graph

In order to design decentralized controls for a physical field, it is necessary to reason about the effects of the controls on the field. We previously introduced the *influence graph* (Bailey-Kellogg & Zhao 1998) to represent such dependencies. Figure 2 shows an example of influences in a temperature domain — a "thermal hill," with temperatures decaying away from the location of a heat source. When multiple sources affect a thermal field, their influences interact, yielding multiple peaks and valleys.

The influences in this example obey the locality principle: a heat source strongly affects nearby field nodes and only weakly affects further away field nodes, depending on the conduction properties of the material. In addition, despite nonlinearities in the spatial variables (e.g. non-uniform conduction characteristics or irregular geometry), influences from multiple heat sources can be linearly superposed to find joint influences. These properties are characteristic of a variety of physical phenomena. In order to take advantage of these and other properties, influence graphs serve as an abstract, domain-independent representation of this knowledge. The definition of influence graph assumes a discretized model, as in (Bailey-Kellogg & Zhao 1998), and as is common to standard engineering methods.

Definition 1 (Influence Graph) *An* influence graph *is a tuple* (F, C, E, w) *where*

- F *is a set of field nodes.*
- C *is a set of control nodes.*
- $E = C \times F$ *is a set of edges from control nodes to field nodes.*
- $w : E \rightarrow \mathcal{R}$ *is an edge weight function with* $w((c, f))$ *the field value at* f *given a unit control value at* c.

Hence, the graph edges record a normalized influence from each control node to each field node. A thermal hill (e.g. Figure 2) is a pictorial representations of the edge weights for an influence graph from one heat source to the nodes of a temperature field.

An influence graph is constructed by placing a control with unit value at each control location of interest, one at a time, and evaluating the field at field node locations of interest. The method of evaluation is problem-specific. For example, it could be found by numerical simulation, experimental data, or even explicit inversion of a capacitance matrix. An influence graph then serves as a high-level interface caching the dependency information. The following two sections use this dependency information in order to decompose models based on influences between their parts.

Graph Decomposition

As discussed in the introduction, many applications require decomposing models into smaller pieces, in order to reason more tractably with the components or to divide-and-conquer a problem. Many models can be formalized in terms of graphs describing the structural dependencies of components. Decomposing a model is then equivalent to partitioning the corresponding graph. Given a graph, a decomposition identifies subsets of vertices so that a metric such as the number of edges or total edge weight between vertices in different subsets is minimized. In particular, the decomposition of an influence graph partitions a model so that the connected components maximize internal influence and minimize external influence. For the running example of decentralized heat control, this decomposes a thermal field so that controls in one part are maximally independent from those in other parts.

The numerical analysis and computational geometry communities have developed a number of methods for partitioning graphs; these methods have varying costs and varying effectiveness. In particular, we concentrate here on spectral partitioning (Simon 1991), which examines the structure of a graph's *Laplacian matrix* encoding the connectivity between points. Specifically, entry (i, j) in the Laplacian matrix has value -1 if and only if there is an edge from node i to node j in the graph; entry (i, i) has value equal to the total number of edges from node i. It turns out that a good approximation to the optimal partitioning (minimum number of cut edges) can be achieved by separating nodes according to the corresponding values in the eigenvector for the first non-trivial eigenvalue of this matrix (the *Fiedler vector*). Intuitively, in a one-dimensional domain, this is similar to partitioning the domain by looking at the sign of a sine wave stretched over it. This technique can be extended to minimize the total weight of edges cut, and normalization of edge weights allows simultaneous optimization of both inter-partition dissimilarity and intra-partition similarity. Shi and Malik showed, in the context of image segmentation, that this approach yields a good estimate of the optimal decomposition (Shi & Malik 1997).

This novel formalization of control placement design in terms of influence graph partitioning provides a graph-based framework in which to develop design algorithms. The spectral partitioning algorithm serves as one instantiation of this framework, based on an all-to-all influence graph. The next section introduces an alternative approach that uses a less detailed model, decomposing a model using an influence graph from a set of representative nodes to all the other nodes.

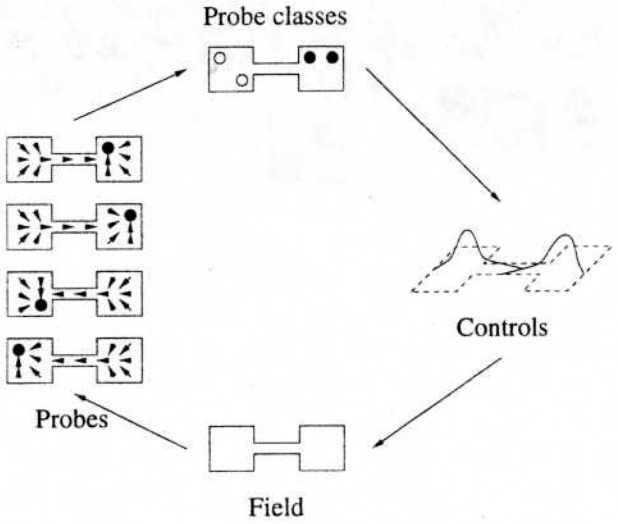

Figure 3: Overview of influence-based decentralized control design for thermal regulation.

Equivalence Class Partitioning

The equivalence class partitioning mechanism decomposes a model based on structural similarities exhibited in a field in response to the actions of sample control probes. Figure 3 overviews the mechanism. In the example shown, the geometric constraint imposed by the narrow channel in the dumbbell-shaped piece of material results in similar field responses to the two probes in the left half of the dumbbell and similar responses to the two probes in the right half of the dumbbell. Based on the resulting classes, the field is decomposed into regions to be separately controlled. In this case, the left half of the dumbbell is decomposed from the right half.

The following subsections detail the components of this mechanism.

Control Probes

For a temperature field to exhibit structures, heat sources must be applied; then an influence graph can be constructed. For example, Figure 4 shows the iso-influences resulting from two different heat source placements; in both cases, the structure of the contours indicates the constraint on heat flow due to the narrow channel. The control placement design algorithm is based on the response of temperature fields to such *control probes*. The number and placement of control probes affects the structures uncovered in a temperature field, and thus the quality of the resulting control design. Possible probe placement strategies include random, evenly-spaced, or dynamically-placed (e.g. in inadequately explored regions or to disambiguate inconsistent interpretations). Experimental results presented later in this paper illustrate the trade-off between number of probes and result quality, using the simple random probe placement strategy.

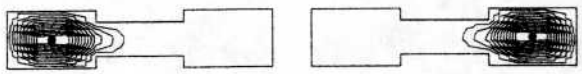

Figure 4: Temperature fields exhibit structures in response to heat source probes.

Probes serve as representatives for the effects of arbitrarily-placed controls. In particular, the probes that most strongly affect a location (e.g. influence greater than average or a fixed threshold) serve to approximate the effects that would be produced by a control placed at that location. The quality of the approximation of controls at arbitrary locations by representative probes depends on the effects of geometry and material properties. Since the influence graph encapsulates these effects, it provides a natural mechanism for reasoning about approximation quality.

Using control probes as representatives of control placement effects supports reformulation of the decomposition problem into that of partitioning probes into equivalence classes. Each equivalence class of probes serves as a representatives for a region of strongly-affected potential control locations, as discussed above. A good decomposition produces probe classes with regions decoupled from the regions of other classes, and which have no acceptable subclasses.

Evaluating Control Decoupling

The first criterion for evaluating a decomposition is that each region be *decoupled* from other regions; that is, that controls in one region have little effect on nodes in the other, and vice-versa. In terms of control probe equivalence classes, decoupling will be evaluated by considering independence of both control placement and control parameters.

To evaluate independence of control placement, consider the influence gradient vectors induced by a set of probes; Figure 5 shows a simple example for two probes.[1] While the flows are different near the locations of the two probes, they are quite similar far away from the probe locations. This similarity is due to constraints imposed by geometry and material properties; in this case, the narrow channel of the material effectively decouples the left and right halves. A numerical measure for the similarity is implemented, for example, by averaging the angular difference between gradient vectors produced by different probes. This measure evaluates the indistinguishability of control placement within the set of probe locations, and thus is correlated with a good decomposition into decoupled regions.

To evaluate independence of control parameters, we distinguish between nodes strongly and weakly influenced by a control (the *near field* and the *far field*, re-

[1] Recall that an influence graph is constructed from field values for unit controls. By influence gradient vectors, we mean vectors in the gradient field for a control — rate and direction of steepest change in field value.

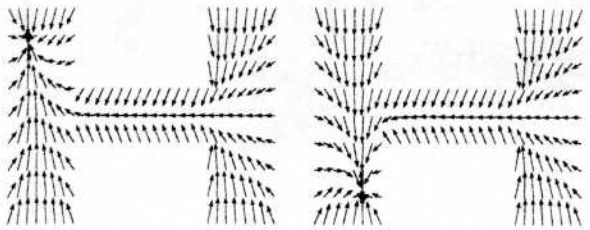

Figure 5: Similarity of flows due to control probes suggests indistinguishability of control placement.

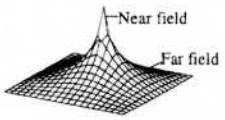

Figure 6: An influence hill partitions a field into near and far fields relative to a control.

spectively), due to locality in the domain (Figure 6). Possible near/far distinctions include a fixed influence threshold, a threshold proportional to the peak of the hill, or a threshold based on the "knee" of the hill. A probe only weakly affects its far field, and thus can be effectively decomposed from it. Alternatively, probes that have significant overlap in their near fields can be reasonably grouped together.

In addition to independence from any single control, a well-decoupled region must be independent from the combined effects of other controls. That is, for a set of probes, the *total* influence on its controlled region from controls for other probe sets must be sufficiently small.

Evaluating Region Atomicity

The second criterion for evaluating a decomposition is that each region be decomposed far enough. A region is considered *atomic* if none of its subregions are adequately decoupled from each other. For example, in Figure 7 a partition $\{\{A,B,C,D\},\{E,F,G\}\}$ achieves good decoupling, since the probes in the first class are relatively independent from those in the second class. However, it is not atomic, since $\{A,B,C,D\}$ can be further decomposed into $\{\{A,B\},\{C,D\}\}$.

One approach to ensuring atomicity of the classes of a decomposition is to recursively test subsets of probes to see if they result in valid decompositions. For example, by testing partitions of the class $\{A,B,C,D\}$ for independence, the partition $\{\{A,B\},\{C,D\}\}$ would be

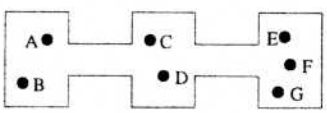

Figure 7: Control probe placement with potential non-atomic decomposition.

Given probes $P = \{p_1, p_2, \ldots, p_n\}$.

Form classes $C = \{\{p_1\},\{p_2\},\ldots,\{p_n\}\}$.

Repeat until stable
  For each neighboring $c_i, c_j \in C$
    If Forall k with $k \neq i$ and $k \neq j$
      decoupling of c_i and c_k and
      decoupling of c_j and c_k are better than
      decoupling of c_i and c_j
    And $c_i \cup c_j$ is *atomic*
    Then replace c_i and c_j in C with $c_i \cup c_j$

Table 1: Probe clustering algorithm.

uncovered. The test can use heuristics to avoid testing all possible subclasses. For example, just by examining overlap in influences in the class $\{A,B,C,D\}$, the partition $\{\{A,B\},\{C,D\}\}$ can be generated as a counterexample to the atomicity of $\{A,B,C,D\}$. If a class is already small, out-of-class probes can be used in such a test, and, if necessary, new probes can be introduced. For example, in an atomicity test for $\{A,B\}$, checking independence of $\{A,C\}$ from $\{B,D\}$ would show that $\{A,B\}$ is indeed atomic. An inexpensive and empirically effective method is to allow grouping of pairs of probes only if their near fields sufficiently overlap.

Probe Clustering Algorithm

Based on these criteria, control probes can be clustered into decoupled, atomic equivalence classes. One effective clustering method is greedily merging neighboring probe classes based on similarity. Start with each probe in its own class, and form a graph of classes based on proximity (e.g. Delaunay triangulation or nearness neighborhood). Then greedily merge neighboring pairs of classes that are most similar, as long as a region is strongly influenced by other regions, and until a merger would result in a non-atomic class. Table 1 provides pseudocode for this algorithm. Figure 8 illustrates a sample probe neighborhood graph, Figure 9 depicts some influence gradients for sample probes, and Figure 10 shows the controlled regions for equivalence classes of probes after the merging process. While more sophisticated clustering mechanisms could be applied, the results in the next section show this algorithm to be empirically effective on a set of example problems.

Implementation Results

The influence-based decomposition algorithms have proved effective in designing control placements for decentralized thermal regulation. The performance has been measured in two ways: quality of the decomposition, and ability of the resulting control design to achieve an objective.

Data for three sample problems are given here: a

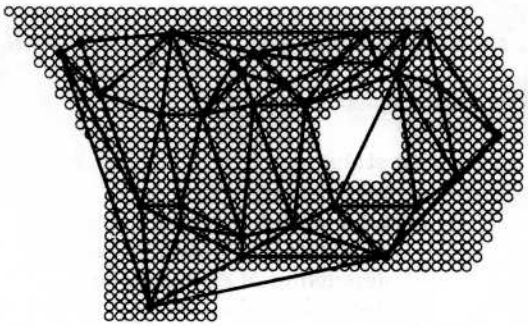

Figure 8: Probe clustering example: probe neighborhood graph.

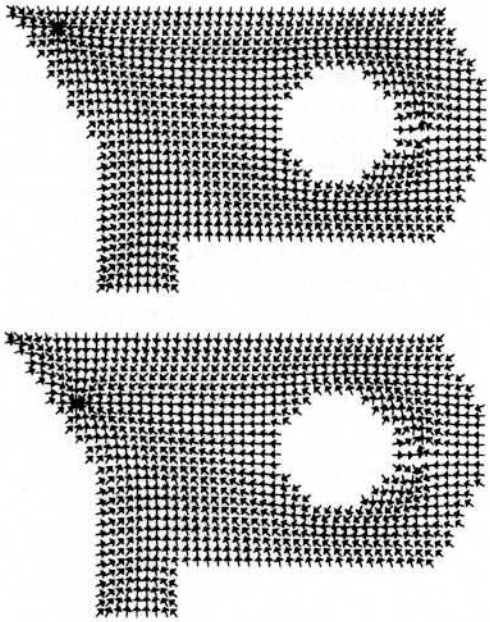

Figure 9: Probe clustering example: influence gradient vectors from two probes.

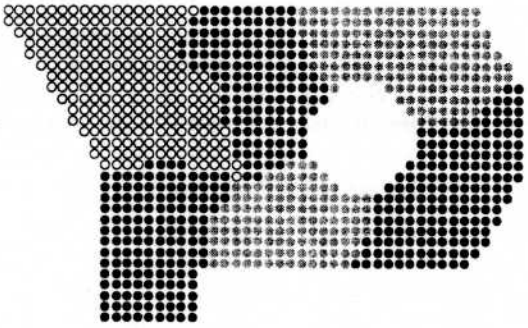

Figure 10: Probe clustering example: region decomposition after merging.

plus-shaped piece of material, a P-shaped piece of material, and an anisotropic (non-uniform conduction coefficient) bar. These problems illustrate different geometries, topologies (the P-shaped material has a hole), and material properties. Other problems have also been tested; the results are similar.

The decomposition algorithm forms groups of control probes with similar effects on the field. This requires that probes be dense enough, relative to conditions imposed by geometry and material properties, so that groups of probes with similar effects can be uncovered. Otherwise, each probe ends up in its own class, and the decomposition is too dependent on probe placement. To study the impact of the number of control probes on the effectiveness of the resulting design, different numbers of probes (4, 8, 16, and 32) were placed at random in a given domain, and results were averaged over a number of trial runs. While smarter probe placement techniques might yield more consistently effective designs, this approach provides a baseline and illustrates the trade-off between number of probes and error/variance. The probe clustering algorithm used a Delaunay triangulation probe neighborhood graph, a near field based on influence of at least 10 percent of peak, and a similarity measure based on flow vector direction. For comparison, merging was performed until four classes remained.

Decomposition Quality

The goal of a decomposition algorithm is to partition a domain into regions such that source placement and parametric optimization in each region is relatively independent of that in other regions (decomposed) and has no internally independent regions (atomic). The estimate of the quality of a decomposition used here is based on a corresponding formalization for image segmentation by Shi and Malik (Shi & Malik 1997): compare the total influence from each control location on locations in other regions (decomposed), and the amount of influence from that location on locations in its own region (atomic). To be more specific, define the decomposition quality q ($0 \leq q \leq 1$) for a partition P of a set of nodes S as follows (i is the influence):

$$q = \prod_{R \in P} \sum_{c \in R} \frac{\sum_{r \in R} i(c,r)}{\sum_{s \in S} i(c,s)}$$

For each control node, divide its influence on nodes in its own region by its total influence. Summing that over each region yields an estimate of the fraction of control output of any control location in the region that is used to control the other locations in that region. The quality measure is combined over all regions by taking the product of each region's quality.

Figure 11 compares the performance of the equivalence class partitioning mechanism with that of the spectral partitioning mechanism. It provides the average error and standard deviation in error over a number of trial runs, with respect to different numbers of control

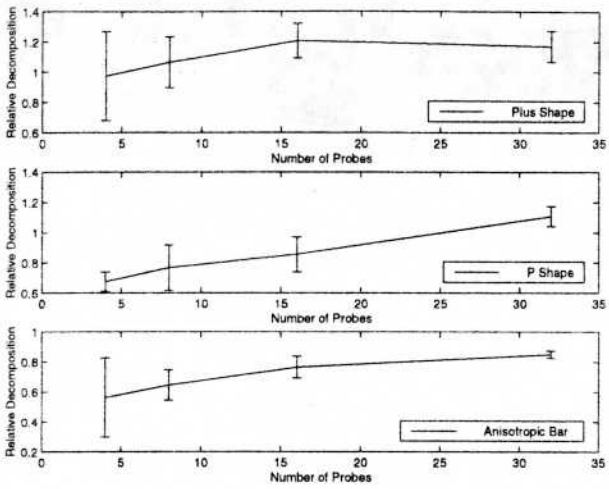

Figure 11: Performance data indicate that the probe clustering algorithm supports trading decomposition quality for computation. Results are relative to spectral partitioning.

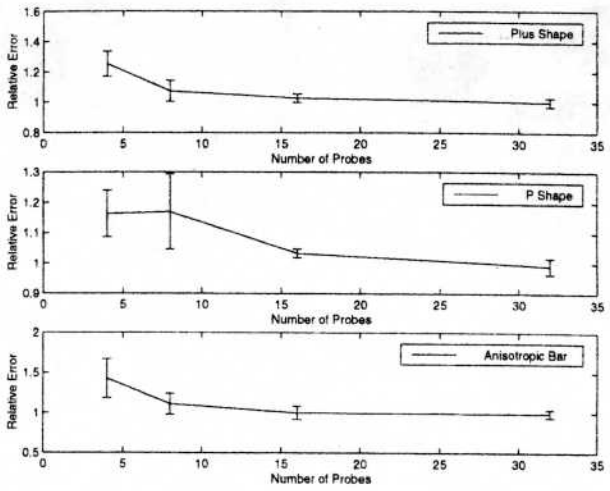

Figure 12: Performance data indicate that equivalence class-based control placement design supports trading control quality for computation. Results are relative to simulated annealing.

probes. For all three problems, the average quality for the equivalence class partitioning mechanism naturally decreases as the number of probes decreases. (There is a slight taper in the performance for the plus shape, due to statistical sampling.) Furthermore, the standard deviation of quality tends to increase as the number of probes decreases, since the partition is more sensitive to specific probe placements. The curve indicates a trade-off between amount of computation and resulting decomposition quality. With enough probes, the quality is commensurate with that of spectral partitioning.

Control Placement Quality

The ultimate measure of the control design algorithm is how well a design based on a decomposition can achieve a control objective. This section evaluates the ability of decomposition-based control designs to achieve a uniform temperature profile. This profile is better than other, non-uniform profiles at indicating the performance of a decomposition, since it does not depend as much on local placement adjustment and parametric optimization. Intuitively, if a decomposition clumps together sources, then some other region will not get enough heat and thus will have a large error.

Simulated annealing (Metropolis *et al.* 1953) serves as a baseline comparison for error; an optimizer was run for 100 steps. The decomposition-based control design used a simple approach: for each region of a decomposition, place controls in the "center of influence" (like the center of mass, but weighted with total influence from the probes, rather than mass, at each point). In both cases, only the global control placement was designed; local adjustment could somewhat reduce the error.

Figure 12 illustrates average error and standard deviation in error over a set of trial runs. The equivalence class algorithm was tested with different numbers of control probes; as shown above, the spectral partitioning algorithm would yield similar results. The error here is the sum of squared difference between actual temperature profile and desired temperature profile. As with decomposition quality, the average and standard deviation of control quality tend to improve with the number of probes; enough probes yields quality commensurate with that of simulated annealing.

Run-Time Performance

There are two implementation strategies to consider in terms of run-time performance. The *centralized* approach explicitly inverts the capacitance matrix describing the system (e.g. from a finite-element mesh), yielding influences from every potential control node to every field node. The matrix inversion constitutes the bulk of the run time. Spectral partitioning then performs an eigenvalue computation. Simulated annealing examines some number of configurations (in the tests above, 100) with respect to the effects encoded in this matrix. Our approach does simple comparisons between fields due to different probe locations.

The *decentralized* approach treats the system as a black box function to be computed for each configuration; the run time primarily depends on the number of function evaluations. The spectral method is not directly amenable to this approach. Simulated annealing requires the function to be evaluated for each configuration (in the tests above, 100). Our apprach requires the function to be evaluated for each probe (in the tests above, 4, 8, 16, or 32 times).

Discussion

This paper has developed mechanisms for automatically decomposing large models, based on structural representations of influences. This formalization in terms of influences differs from the formalisms in related engineering work, which typically use topology and geometry, and related AI work, which typically use constraint nets. Redescribing a model decomposition problem as an influence graph partitioning problem allows application of powerful, general algorithms.

We first introduced the influence graph mechanism in (Bailey-Kellogg & Zhao 1998) as a generic basis for representing and manipulating dependency information between control nodes and field nodes. The properties of physical fields that it encapsulates (locality, linear dependence, etc.) are exhibited by a variety of physical phenomena, from electrostatics to incompressible fluid flow. Note that these phenomena are all governed by diffusion processes; it remains future work to develop similar mechanisms for wave processes. Since they are based on influence graphs, the model decomposition formalism and algorithms developed here are applicable to this wide variety of models.

The influence graph provides a common interface to the model of a distributed physical system, whether it is derived from partial differential equations, simulation data, or actual physical measurements. In the case of physical measurements, sensor outliers can be detected by comparing the data with the expected value based on nearby points. Additionally, since the algorithms reason about the qualitative structures of influences, they are less sensitive to individual sensor errors.

The decomposition-based control design algorithms search a design space in a much different manner from that of other combinatorial optimization algorithms, such as genetic algorithms (Holland 1975) and simulated annealing (Metropolis *et al.* 1953). Rather than (perhaps implicitly) searching the space of all possible combinations of source locations, the influence-based decomposition approach divides and conquers a problem, breaking a model into submodels based on influences. This approach explicitly forms equivalence classes and structures in the domain, rather than implicitly representing them in terms of, for example, increased membership of highly-fit members in a population. Since design decisions are based on the influence structure of the field, this approach supports higher-level reasoning about and explanation of its results; for example, a design decision could be explained in terms of constrained influence flows through a field.

Conclusion

This paper has developed efficient influence-based model decomposition algorithms for optimization problems for large distributed models. Model decomposition is formalized as a graph partitioning problem for an influence graph representing node dependencies. The first algorithm applies spectral partitioning to an influence graph. The second algorithm decomposes a graph using structural similarities among representative control probes. Both algorithms reason about the structure of a problem using influences derived from either a constraint net or numerical data. Computational experiments show that the algorithms compare favorably with more exhaustive methods such as simulated annealing in both solution quality and computational cost.

Acknowledgments

The work is supported in part by ONR YI grant N00014-97-1-0599, NSF NYI grant CCR-9457802, and a Xerox grant to the Ohio State University.

References

Bailey-Kellogg, C., and Zhao, F. 1998. Qualitative analysis of distributed physical systems with applications to control synthesis. In *Proceedings of AAAI.*

Chan, T., and Mathew, T. 1994. *Domain Decomposition Algorithms,* volume 3 of *Acta Numerica.* Cambridge University Press. 61–143.

Chen, G.-S.; Bruno, R.; and Salama, M. 1995. Optimal placement of active / passive members in truss structure using simulated annealing. *AIAA Journal* 29.

Clancy, D. 1997. Model decomposition and simulation: a component based qualitative simulation algorithm. In *Proceedings of AAAI.*

Dhingra, A., and Lee, B. 1994. Optimal placement of actuators in actively controlled structures. *Engineering Optimization* 23:99–118.

Doumanidis, C. 1997. In-process control in thermal rapid prototyping. *IEEE Control Systems.*

Felzenszwalb, P., and Huttenlocher, D. 1998. Image segmentation using local variation. In *Proceedings of CVPR.*

Friedman, N.; Koller, D.; and Pfeffer, A. 1998. Structured represenation of complex stochastic systems. In *Proceedings of AAAI.*

Holland, J. 1975. *Adaptation in Natural and Artificial Systems.* The University of Michigan Press.

Kailath, T., et al. 1996. Control for advanced semiconductor device manufacturing: A case history. In Levine, W., ed., *The Control Handbook.* CRC Press.

Metropolis, N.; Rosenbluth, A.; Rosenbluth, M.; Teller, A.; and Teller, E. 1953. Equation of state calculations by fast computing machines. *Journal of Chemical Physics* 21:1087–1092.

Shi, J., and Malik, J. 1997. Normalized cuts and image segmentation. In *Proceedings of CVPR.*

Simon, H. 1991. Partitioning of unstructured problems for parallel processing. *Computing Systems in Engineering* 2:135–148.

Williams, B., and Millar, B. 1996. Automated decomposition of model-based learning problems. In *Proceedings of 10th International Workshop on Qualitative Reasoning.*

Williams, B., and Nayak, P. 1996. Immobile robots: AI in the new millenium. *AI Magazine* 17(3).

Zhao, F. 1987. An O(N) algorithm for three-dimensional n-body simulations. Technical Report AI-TR-995, MIT AI Lab.