

# Influence-Based Model Decomposition For Reasoning About Spatially Distributed Physical Systems

Chris Bailey-Kellogg\*

*Dartmouth Computer Science Department  
6211 Sudikoff Laboratory  
Hanover, NH 03755*

Feng Zhao

*Xerox Palo Alto Research Center  
3333 Coyote Hill Road  
Palo Alto, CA 94304*

---

## Abstract

Many important science and engineering applications, such as regulating the temperature distribution over a semiconductor wafer and controlling the noise from a photocopier machine, require interpreting distributed data and designing decentralized controllers for spatially distributed systems. Developing effective computational techniques for representing and reasoning about these systems, which are usually modeled with partial differential equations (PDEs), is one of the major challenge problems for qualitative and spatial reasoning research.

This paper introduces a novel approach to decentralized control design, *influence-based model decomposition*, and applies it in the context of thermal regulation. Influence-based model decomposition uses a decentralized model, called an *influence graph*, as a key data abstraction representing influences of controls on distributed physical fields. It serves as the basis for novel algorithms for control placement and parameter design for distributed systems with large numbers of coupled variables. These algorithms exploit physical knowledge of locality, linear superposability, and continuity, encapsulated in influence graphs representing dependencies of field nodes on control nodes. The control placement design algorithms utilize influence graphs to decompose a problem domain so as to decouple the resulting regions. The decentralized control parameter optimization algorithms utilize influence graphs to efficiently evaluate thermal fields and to explicitly trade off computation, communication, and control quality. By leveraging the physical knowledge encapsulated in influence graphs, these control design algorithms are more efficient than standard techniques, and produce designs explainable in terms of problem structures.

*Key words:* Qualitative reasoning, spatial reasoning, model decomposition, spatially distributed systems, distributed control design.

---

## 1 Introduction

Many important science and engineering applications require interpreting data and designing decentralized controllers for spatially distributed systems. For example, recent advances in the fabrication of low-cost, large-scale arrays of microelectromechanical systems (MEMS) have enabled the construction of “smart matter” systems that integrate sensing, computation, and actuation at a fine grain. Recent MEMS applications include manipulation of parts with distributed arrays of cilia-like actuators [1] and stabilization of beams with piezoelectric materials that sense and counteract buckling [2].

Interpretation and control tasks for distributed physical systems encounter a number of challenges. In lumped parameter systems, i.e., systems modeled as ordinary differential equations, or ODEs, such as circuits, topology is the most important spatial property (e.g. in the device ontology commonly used in Qualitative Reasoning). However, in distributed parameter systems, i.e., systems modeled as partial differential equations, or PDEs, additional spatial properties are also relevant. For example, temperature fields are influenced by the geometry of the domain, spatial variations in material property, and boundary conditions. The additional complexity makes it much harder to apply analytic methods to determine closed-form solutions; instead, simulation and search on large-scale discretizations are often required. This presents both a challenge as well as opportunities for AI reasoning systems for analyzing and synthesizing distributed parameter systems. In fact, reasoning about spatially distributed systems has been proposed as one of the major challenge problems for the qualitative and spatial reasoning research community [3].

Data interpretation and control applications for distributed physical systems are limited by physical laws and physical hardware constraints. Sensors and actuators interact predominantly only with local regions of space. As a result, global interpretations must be extracted from collections of locally measured data, and global control laws must be enforced by local actuation rules. The difficulty of constructing local to global mappings and back is compounded

---

\* Corresponding author.

*Email addresses:* [cbk@cs.dartmouth.edu](mailto:cbk@cs.dartmouth.edu) (Chris Bailey-Kellogg),  
[fz@alum.mit.edu](mailto:fz@alum.mit.edu) (Feng Zhao).

*URLs:* <http://www.cs.dartmouth.edu/~cbk> (Chris Bailey-Kellogg),  
<http://www.parc.xerox.com/zhao> (Feng Zhao).

by nonlocal coupling between local nodes. In addition to strong interactions with local areas of space, sensors and actuators have weaker interactions with other areas of space. For example, the effect of a heating lamp controlling the temperature over a local region of a semiconductor wafer may diffuse to other regions and interfere with efforts to regulate temperature in those regions. Sensing and control designs must account for such coupling.

Data interpretation applications for spatially distributed systems are often challenged by the massive amount of data, either collected from arrays of sensors or produced by simulations running on fine-grained discretizations of models. Global analysis methods manipulating entire data sets quickly reach computational limitations as the size of the data sets increases. Instead, applications must rely on local methods that manipulate separate subsets of the data relatively independently. For example, domain decomposition methods [4] for solving partial differential equations form subregions of a discretization and iteratively combine and refine independent solutions for the subregions. Applications can also use data reduction and approximations to reason about a problem at multiple levels of abstraction. For example, multigrid methods [5] iterate between fine-grained and coarse-grained solutions to PDEs. Meteorologists use abstract structures such as isobars, pressure troughs, and pressure cells to reason about the underlying pressure data at a higher level of abstraction.

What makes it possible to overcome these challenges and design data interpretation and control applications for distributed physical systems? Physical properties such as continuity and locality give rise to regions of uniformity in spatially distributed data. Spatial Aggregation theory [6] proposes a uniform mechanism utilizing explicit representations of such knowledge, expressed as metrics, adjacency relations, and equivalence predicates, to uncover and exploit structures in physical data. Spatial Aggregation follows an imagistic reasoning [7] style, applying vision-like routines to manipulate multi-layer geometric and topological structures in spatially distributed data. Control design applications use similar techniques to navigate through imagistic representations of the effects of control actions [8].

This paper extends these ideas to capture a key abstraction, the *influence graph*, representing effects of decentralized controllers on distributed physical fields. The influence graph mechanism utilizes physical knowledge of locality, linear superposability, and continuity to manipulate structural descriptions of influences for a class of problems such as heat transfer, electrostatics, gravity, and incompressible fluid flow. The influence graph supports novel algorithms for control placement and parameter design for systems with large numbers of coupled variables [9–11]. The control placement design algorithms use structural knowledge to decompose a domain into regions so that controls in separate regions are maximally decoupled. The control parameter design

algorithms use structural knowledge to efficiently evaluate temperature fields and to explicitly trade off computation, communication, and control quality during optimization. By leveraging physical knowledge, these control design algorithms are more efficient than standard techniques and produce designs explainable in terms of problem structures. While we present it in terms of an application to decentralized thermal regulation, the influence graph mechanism provides a generic vocabulary for designing decentralized controls, in terms of effects of controls on a field and similarities in control effects. These techniques are appropriate for other control design problems requiring placement and parametric optimization of decentralized controls for distributed physical fields.

### 1.1 Problem Description

This paper considers a control design problem as a mapping from a spatial domain  $S^1$ , a behavioral model  $M$ , and a set of design constraints  $\Sigma$  to a set of control nodes  $C$  and their control parameters  $U$ . Formally, control design solves

$$S \times M \times \Sigma \rightarrow C \times U \quad (1)$$

We illustrate the design problem using a generic problem of temperature regulation for a piece of material [12], as shown in Figure 1. The design problem is to regulate the temperature distribution over the material to a desired profile, using a small number of point heat sources.

#### Definition 1 (Control Design for Thermal Regulation)

- $S$  specifies the geometry of the material, with its boundary denoted by  $\partial S$ .
- $M$  describes how heat diffuses in  $S$  and what happens at  $\partial S$ .
- $\Sigma = (T, \epsilon)$  where  $T$  is a desired temperature profile and  $\epsilon$  an error tolerance, both defined over the domain  $S$ . The design should minimize deviation from  $T$  and not allow it to exceed  $\epsilon$ .
- $C = \{c_1, \dots, c_n\} \subset S$  are point heat source locations.
- $U = \{u_1(t), \dots, u_n(t)\}$  are heat outputs from the corresponding heat sources as functions of time.

The design depends on the geometry  $S$ , the thermal process in the material (specified by  $M$ ), and the constraints  $\Sigma$ . There are two important components to the design: structure design (e.g. the number and location of heat sources)

---

<sup>1</sup> Later in the paper,  $S$  interchangeably refers to either the domain or a discretization of the domain, in slight abuse of notation.

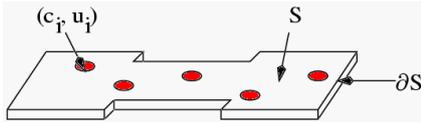


Fig. 1. Industrial heat treatment of a piece of material. The control objective is to achieve a specified temperature profile over the material by applying heat at a small number of locations, shown as dark circles.

that determines  $C$  and parametric design (e.g. heat source values) that determines  $U$ . Rather than addressing both design components simultaneously, which would yield a very large and complex design space, our approach is to design the structure first and then design the parameters. The structure is designed in a manner that actually aids the parametric design, by placing controls so that they minimally interfere with each other. This approach is particularly appropriate for applications where the structure design is performed once (to place controls), and the parametric design is performed repeatedly (e.g. for various desired profiles). In general, a design could iterate through the structure design and parametric design until satisfactory results are obtained. Most existing constrained optimization methods focus on parametric optimization. In contrast, our method uses structural information of a spatial physical field to guide both control placement as well as parametric design. While control placement design is performed at design time, parametric optimization can be performed off-line as well as online when tracking a time-varying temperature profile or when the system parameter drifts.

Our abstract problem statement describes a class of practical problems. Many such applications require decentralized control in order to ensure adaptivity, robustness, and scalability. Consider two different thermal regulation systems, represented in Figures 2 [13] and 3 [14]. Doumanidis developed the system in Figure 2 for rapid prototyping in thermal fabrication (i.e. welding). It includes a servodriven X-Y positioning table, upon which the parts to be joined are placed, a plasma-arc heat source, and an infrared camera providing temperature data. Doumanidis applies feedback control to a linearized model of the system whose parameters are estimated at run-time from temperature distributions [13]. Groups at Stanford and Texas Instruments developed the system in Figure 3 for rapid thermal processing for semiconductor curing, where a uniform temperature profile must be maintained to avoid defects. The control strategy is somewhat decentralized, providing separate power zones for three separate rings of heat lamps [14].

As another example, design of a “smart building” deploys networks of sensors and actuators to regulate temperature and other environmental parameters. Using a decentralized design is appealing since it allows the network to overcome failures in individual control elements and to scale up without incurring the complexity that increases exponentially with the number of controls. Optimal placement and control of these sensors and actuators can save energy

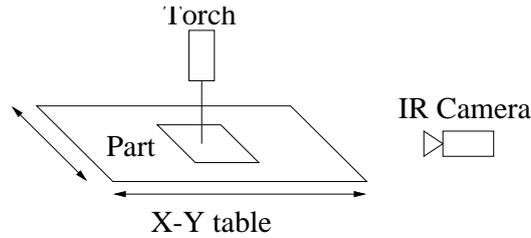


Fig. 2. Rapid thermal processing system welds by moving a part on a positioning table based on feedback from an infrared camera.

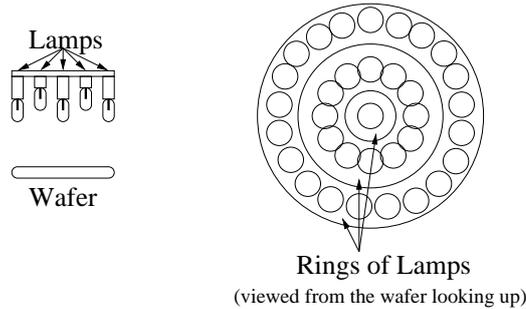


Fig. 3. Rapid thermal processing for semiconductor manufacturing maintains a uniform temperature distribution by independent control to separate rings of heat lamps.

as well as maximizing occupant comfort. The design challenge is to achieve the global control objective through appropriate combinations of local control actions.

In this paper, we only consider the control problem, assuming that the system is fully observable (e.g. obtaining sufficient temperature data from an infrared camera). The observability problem can be addressed similarly to the controllability problem; see Section 6.2 for further discussion.

## 1.2 Overview of the Approach

This paper presents influence-based model decomposition and applies it to a case study application to the design of decentralized controls for thermal regulation. Structures uncovered in physical fields serve as the basis for algorithms that design control placements and control actions. Influence graph-based design mechanisms support explicit trade-offs between factors such as amount of computation, amount of communication, and resulting control quality. Influence graphs allow explanation of and meta-level reasoning about the resulting designs, in terms of the physical knowledge they represent. While the case study is firmly grounded in the thermal regulation domain, the influence graph abstracts a generic set of reasoning mechanisms for control design problems requiring placement and parametric optimization of decentralized controls for

distributed physical fields.

Our approach stems from work by Abelson *et al.* [15] underlining the importance of incorporating intelligence into scientific computing. *Intelligent simulation* uses a mixture of symbolic, numeric, and geometric techniques to automatically prepare, perform, and interpret simulations. Yip *et al.* [7] further refined this idea to specify a class of *imagistic reasoning* systems, in which the input is an image-like representation of spatially distributed physical data, the output is a high-level description of the data, and perception-like routines are used to perform the mapping. Yip and Zhao [6] then introduced Spatial Aggregation (SA) as a realization of imagistic reasoning unifying the mechanisms underlying a number of successful imagistic problem solvers, including KAM [16], which interprets the behaviors of Hamiltonian systems, MAPS [17], which designs control laws based on a geometric analysis of the state equations of a dynamical system, and HIPAIR [18], which analyzes the kinematics of fixed-axis mechanisms.

Figures 4 and 5 overview our approach. The first task is to determine the effects of controllers on a field — in this case study, point heat sources yielding heat flow in a piece of material (Figure 4) — and encapsulate the control effects in an influence graph. Analysis of the structures in an influence graph drives control placement design and parametric optimization as follows (depicted in Figure 5). Control probes yield a sampled influence graph representation. These control probes are clustered based on similarities of their effects as represented in the influence graph. In the example, 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 equivalence 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. Controls are placed in the regions and optimized by adjusting their outputs in response to their effects on the field.

The rest of the paper is organized as follows. Section 2 describes models of distributed heat flow and their computational representation in the Spatial Aggregation Language. Section 3 presents the construction of influence graphs for decentralized control problems. Influence graph-based control design algorithms are elaborated in two separate sections: Section 4 presents algorithms for determining control placements, while Section 5 presents algorithms for determining control actions. Finally, Section 6 discusses the broader applicability of the approaches described here.

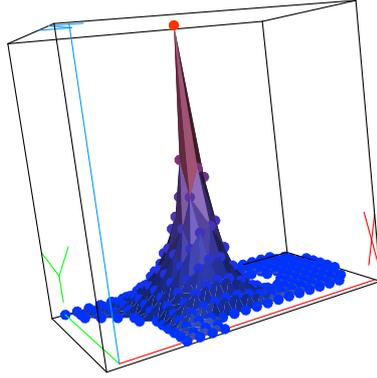


Fig. 4. Graphical representation of the effects of a single controller (point heat source) on a two-dimensional field (the vertical axis represents temperature). The influence graph mechanism encapsulates, analyzes, and manipulates the effects of many such individual controllers.

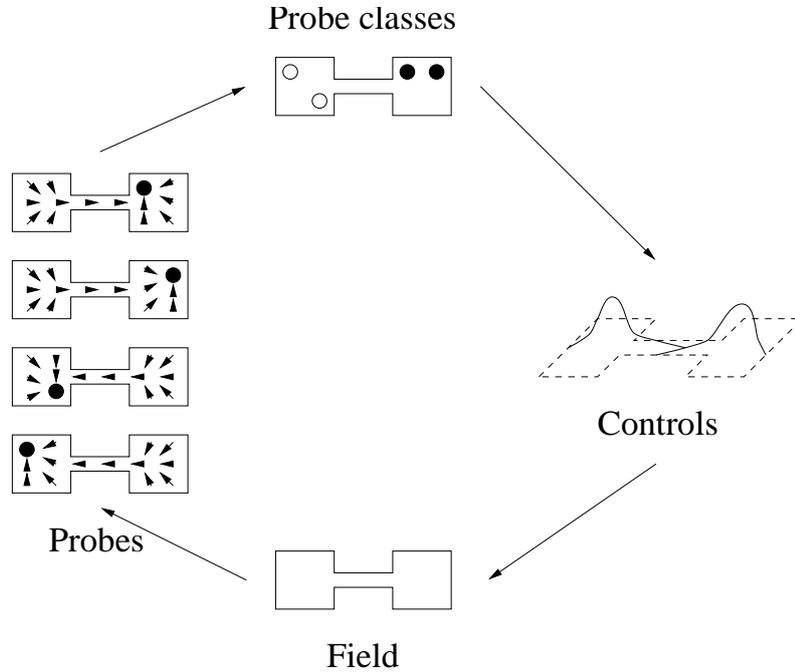


Fig. 5. Overview of decentralized control design by influence-based model decomposition. Control probes form a sampled influence graph which is analyzed for similarities in effects (e.g. the flows due to the probes in the left half of the dumbbell are similar). Clustering probes yields a decomposition of the field into regions to be separately controlled. Individual controls are parametrically optimized based on their influences on the field.

## 2 Spatially-Distributed Models

The physical process of heat diffusion is modeled by partial differential equations (PDEs). In particular, this case study focuses on steady-state (asymptotic)

otic) temperature distributions, modeled by the Laplace equation:

$$\nabla \cdot \nabla k \phi + \dot{Q} = 0, \quad (2)$$

subject to *boundary conditions* specifying heat flow characteristics at the boundary of the domain. Here  $\nabla$  is the spatial derivative (e.g.  $\frac{\partial}{\partial x} \hat{i} + \frac{\partial}{\partial y} \hat{j}$  for a 2-D Euclidean space),  $\phi$  is the temperature,  $k$  is the material conduction coefficient, and  $\dot{Q}$  is the source value representing heat per unit time and volume. Intuitively, this equation models heat diffusion as a smoothing process, reducing sharp spatial variations in temperature.

Given the description of a piece of material (geometry, boundary conditions, and material properties), solving the heat equation yields the resulting temperature profile over the material. However, for non-trivial descriptions, it is often infeasible to find a closed-form solution to the equation. Thus scientists and engineers turn to solving equations for discretized representations in order to extract the behavior of a system. For example, the finite difference method (Figure 6(a)) approximates the spatial derivatives in the heat equation by comparing temperatures among points in a grid, and the finite element method [19] (Figure 6(b)) minimizes an error term derived from conservation of energy over the elements in a mesh. Both methods lead to systems of the following form:

$$A\vec{\phi} = \vec{u} \quad (3)$$

$A$  is the *capacitance* matrix formed from the discretization,  $\vec{\phi}$  is a vector whose elements represent the resulting temperature distribution at the nodes in the discretization, and  $\vec{u}$  is the contribution from heat sources and boundary conditions. In terms of our control design terminology (Eq. 1),  $S$  specifies the locations for the discretization of  $\vec{\phi}$ ,  $M$  the capacitance matrix  $A$ ,  $C$  the positions of the heat sources with respect to the indexing of  $\vec{\phi}$ , and  $U$  the heat source outputs  $\vec{u}$  over time.

An important point about the discretization of the heat equation by finite differences or finite elements is that it yields a set of equations relating temperature  $\vec{\phi}$  to a linear combination of heat source outputs  $\vec{u}$ , via their contributions in  $A^{-1}$ . This is not to say that there is no nonlinearity in spatial variables. In fact, the heat conduction coefficient  $k$  can vary nonlinearly in the spatial domain.

Furthermore, note that the effects of boundary conditions (what happens at the edges of the domain) are wrapped up into the system and treated the same as heat sources. For example, if the boundary conditions specify a non-zero constant value (i.e. the temperature in the domain has little impact on

the temperature outside, which remains constant), the impact can be treated similarly to a heat source/sink, and added in as a separate entry in  $\vec{u}$ . If the boundary conditions have a term involving  $\vec{\phi}$  (e.g. an insulated domain with gradient boundary conditions), the impact is factored directly into the capacitance matrix  $A$ .

The capacitance matrix is sparse: entry  $(i, j)$  in  $A$  is non-zero if and only if  $i$  and  $j$  are neighbors in the discretization. This property makes the system amenable to solution by *relaxation* methods, which start with a guess at the resulting temperature profile and iteratively improve the guess until the process converges. The original system (3) is rewritten into a form with  $\vec{\phi}$  on both sides of the equation:

$$\vec{\phi} = B\vec{\phi} + \vec{v} \tag{4}$$

For example, one possible rewriting (Jacobi) sets  $B = D^{-1}(L + U)$  and  $\vec{v} = D^{-1}\vec{u}$ , where  $A$  has been separated into  $A = D - L - U$ , with  $D$  a diagonal matrix,  $L$  a lower-triangular matrix, and  $U$  an upper-triangular matrix [5]. Then successive approximations to the solution are formed by using the current value of  $\vec{\phi}$  on the right-hand side of (4) to compute a new value of  $\vec{\phi}$  on the left-hand side:

$$\begin{aligned} \vec{\phi}^{(1)} &= B\vec{\phi}^{(0)} + \vec{v} \\ \vec{\phi}^{(2)} &= B\vec{\phi}^{(1)} + \vec{v} \\ &\dots \\ \vec{\phi}^{(n)} &= B\vec{\phi}^{(n-1)} + \vec{v} \end{aligned}$$

The convergence rate of this technique depends on the form of the rewriting from (3) to (4); common approaches include the Jacobi method, which simultaneously updates all members of  $\vec{\phi}$ , the Gauss-Seidel method, which progressively updates elements of  $\vec{\phi}$ , and successive overrelaxation, which extends this approach to mix in differing proportions of old and new  $\vec{\phi}$  values [12].

The Spatial Aggregation Language (SAL) [20,6], summarized in Table 1, provides a concise set of data types and operators at a level of abstraction appropriate for computing with distributed physical fields. The algorithms in this paper utilize SAL vocabulary and extend it with influence graph operations. In the heat control domain, a piece of material is discretized and represented by a set of point objects (sensor locations or points for which the heat equation is to be solved). Temperatures are represented by a scalar field mapping points to temperatures; heat flows are represented by the corresponding gradient vector field. Points are related in neighborhood graphs (ngraphs) encoding an appropriate adjacency relationship (e.g. 4-adjacency for the finite difference grid of

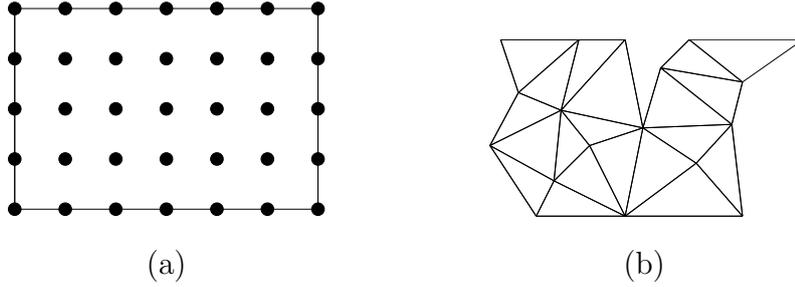


Fig. 6. Discretized distributed representations support numerical solution of the heat equation by solving systems of equations for (a) diffusion in a regular finite difference grid or (b) conservation properties in a finite element mesh.

Figure 6(a) or a triangulated neighborhood in the finite element mesh of Figure 6(b)). The heat equation can be solved by local relaxation rules on such neighborhood graphs. Structures in fields are extracted as equivalence classes of neighboring objects according to some similarity measure (e.g. similarity in temperature value or gradient vector direction). These equivalence classes are abstracted to primitive objects at a higher level of abstraction (e.g. isothermal regions or curves from groups of points with similar temperature value or gradient flow curves from groups of points with similar vector directions).

To emphasize the distributed nature of the algorithms in this paper, many of them are presented graphically, depicting data flow among processing elements (e.g. field nodes connected by ngraph edges). More traditional pseudocode also helps illustrate the high-level computational structure of the algorithms.

### 3 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. This section introduces the *influence graph* mechanism to represent and manipulate such dependencies. By explicitly representing the dependencies of field nodes on control nodes, influence graphs support the control design techniques discussed in the rest of this paper.

#### 3.1 Thermal Hill

A heat source influences the temperature distribution in a field through heat propagation. Figure 7(a) shows that the steady-state influence of a source on a field forms a “thermal hill”: the temperature decays away from the source. When multiple sources affect a thermal field, their thermal hills interact,

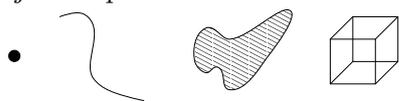
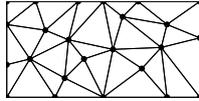
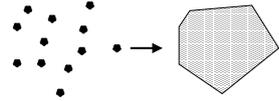
<ul style="list-style-type: none"> <li>• <i>Primitive Objects</i> represent locations and structures in spatial data.</li> </ul> <p>Example: • </p>
<ul style="list-style-type: none"> <li>• <i>Compound Objects</i> combine primitive objects.</li> </ul> <ul style="list-style-type: none"> <li>• <i>Spaces</i> group objects.</li> </ul> <p>Example (points and curves): </p> <ul style="list-style-type: none"> <li>• <i>Fields</i> associate objects and features.</li> </ul> <p>Example (points and temperatures): </p> <ul style="list-style-type: none"> <li>• <i>Ngraphs</i> relate nearby objects.</li> </ul> <p>Example (Delaunay triangulation): </p> <ul style="list-style-type: none"> <li>• <i>Equivalence classes</i> group similar objects.</li> </ul> <p>Example (points with similar vector directions): </p>
<ul style="list-style-type: none"> <li>• <i>Means of Abstraction</i> connect compound objects at one level of abstraction and primitive objects at the next.</li> </ul> <p>Example (points to region bounded by convex hull): </p>

Table 1  
Components of the Spatial Aggregation Language.

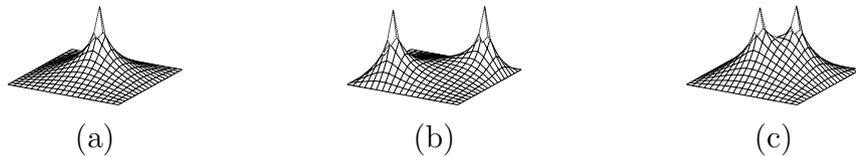


Fig. 7. Steady-state thermal hills around sources. The vertical axis represents temperature value which peaks at source location(s). (a) A single source. (b) Two fairly independent sources. (c) Two tightly coupled sources.

jointly affecting the temperature distribution (Figures 7(b) and Figure 7(c)). Figure 4 shows a hill in a more complex domain.

The structure of these thermal hills exposes quite a bit about the influence of a heat source on the temperature field. Temperature decays away from the source at different rates in different directions, due to different constraints from geometry, boundary conditions, and material properties. Similarly, thermal hills from heat sources at different locations have different shapes. Thermal hills expose the *locality* of the effects of a heat source: 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 order to take advantage of these and other properties at a high level of abstraction, influence graphs serve as an abstract, domain-independent representation encoding this knowledge. As discussed in Section 6, influence graphs can represent controls and fields other than from the thermal regulation domain.

### 3.2 Influence Graph Construction

An *influence graph* records the dependencies between control nodes and spatial objects in a field as edge weights in a graph.

**Definition 2 (Influence Graph)** An influence graph  $\mathcal{I} = (S, C, E, w)$  where

- $S$  is a set of field nodes.
- $C \subset S$  is a set of control nodes.
- $E = C \times S$  is a set of edges from control nodes to field nodes.
- $w : E \rightarrow \mathbb{R}$  is an edge weight function with  $w(e)$  for  $e = (c, s)$  the field value at  $s$  given a unit control value at  $c$ .

We use the following notational shortcuts:

- $\mathcal{I} : C \times S \rightarrow \mathbb{R}$  such that  $\mathcal{I}(c, s) \mapsto w(e)$  when  $e = (c, s) \in E$ .
- $\mathcal{I} : C \rightarrow (S \rightarrow \mathbb{R})$  such that  $\mathcal{I}(c) \mapsto \{(s, \mathcal{I}(c, s)) \mid s \in S\}$  when  $c \in C$ .

Hence, the graph edges record a normalized influence from each control node to each field node. The thermal hills in the last subsection (e.g. Figure 7(a)) are 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 (e.g. using the relaxation mechanism discussed in the previous section), experiment, or even explicit inversion of a capacitance matrix. In the case of solving for an influence graph by relaxation, the computation requires  $O(mn)$  work where  $m$  is the number of probes and  $n$  the number of field nodes; the constant in big  $O$  depends on the relaxation method. This work can be distributed, computing influence separately for each probe. An influence graph then serves as a high-level interface caching important information about spatially distributed physical systems; in this case, the information indicates dependencies of field values upon controls.

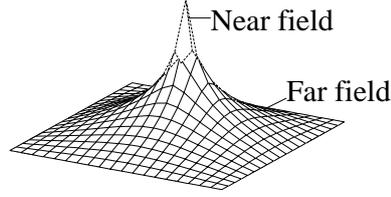


Fig. 8. An influence hill partitions a field into near and far fields — the near field is strongly influenced by the control and the far field isn't.

### 3.3 Influence Graph Properties

Why is an influence graph useful? As Figures 7 and 4 showed for thermal hills, influence strengths vary in different directions and from different locations, depending on geometry and field properties. Influence graphs encapsulate these variations in dependence for use by other reasoning mechanisms. For example, control placement design will exploit the constraints on heat flow indicated by directions of flow.

Influence graphs also encapsulate locality of control effects. Locality can be used to distinguish between a *near field* and *far field* relative to the amount of influence exerted by a control. For example, Figure 8 shows the near and far fields based on the thermal hill from a heat source.

**Definition 3 (Near and Far Field)** For an influence graph  $\mathcal{I} = (S, C, E, w)$ , the near field of a control  $c \in C$  is the set  $N_c = \{s \in S \mid \mathcal{I}(c, s) > \varepsilon_c\}$ ; the far field is  $S - N_c$ .

The required amount of influence  $\varepsilon_c$  (which can depend on the control location  $c$ ) is specified for example by a fixed influence threshold, a threshold proportional to the peak value, or a threshold based on the “knee” of the influence hill for the control. Our implementation of the algorithm presented below uses a proportional threshold. Control parameter design will leverage the locality encapsulated in influence graphs, and extracted in terms of near/far fields, to support more independent reasoning about control actions taken by decentralized controls.

In many distributed physical phenomena, despite nonlinearities in the spatial variables (e.g. non-uniform conduction characteristics or irregular geometry), solutions can be linearly superposed as discussed near Eq. 3.

**Definition 4 (Linear Superposability in the Heat Equation)** Heat equation solutions are linear superposable: if  $\vec{\phi}_1$  and  $\vec{\phi}_2$  are solutions to the heat equation (Eq. 3) with  $\vec{u}_1$  and  $\vec{u}_2$  respectively, and  $c$  is a constant, then

- (Scalability):  $c\vec{\phi}_1$  is a solution to the heat equation with  $c\vec{u}_1$ .

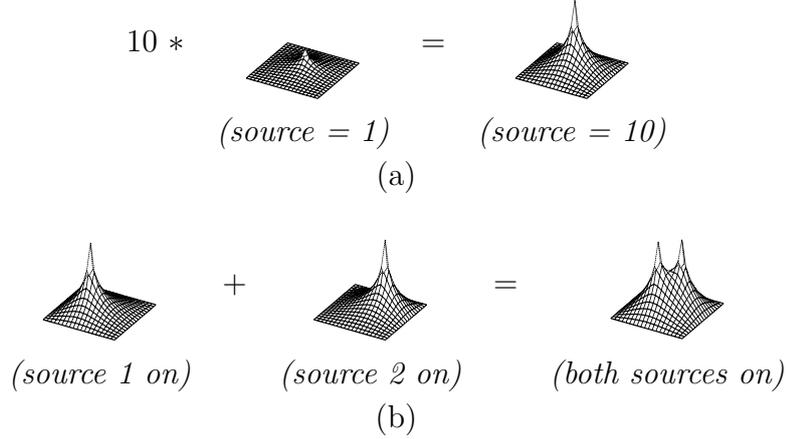


Fig. 9. Linear superposability of thermal hills. (a) Scalability: scaled thermal hill is equivalent to thermal hill with source value scaled. (b) Additivity: sum of thermal hills is equivalent to thermal hill with both sources active.

- (Additivity):  $\vec{\phi}_1 + \vec{\phi}_2$  is a solution to the heat equation with  $\vec{u}_1 + \vec{u}_2$ .

Since influence graphs represent solutions to the heat equation at unit control values, this property means that the effects of controls can be combined through a superposition of influences.<sup>2</sup> For example (see Figure 9), given the thermal hill for a heat source at one control value, the temperature field resulting from a different control value is simply an appropriately scaled version of the original thermal hill. Similarly, given the thermal hills for two separate heat sources, the temperature field resulting from both heat sources is simply the sum of the two thermal hills. Influence graphs encode the crucial dependency information, while hiding other possibly nonlinear effects. Control parameter design will exploit linear superposability to efficiently evaluate fields through addition and subtraction of appropriately scaled hills.

### 3.4 Simple Structures in Influence Graphs

A common representation of the structure of a field is with iso-contours, or curves of equal field value. For example, Figure 10 shows some iso-contours for the hill of Figure 4. The contours are essentially loops around the hill at the same “altitude” (influence value). The steepness of the hill can be judged by examining the distance between adjacent contours at various points. This in turn indicates the rate of influence decay in different directions. The iso-contours can be computed from the influence field  $\mathcal{I}(c)$  due to a given control  $c$  by well-established techniques such as the marching squares algorithm.

Gradient vectors are a dual representation to iso-contours: while iso-contours

<sup>2</sup> Recall that non-zero boundary conditions are to be treated as separate influences.

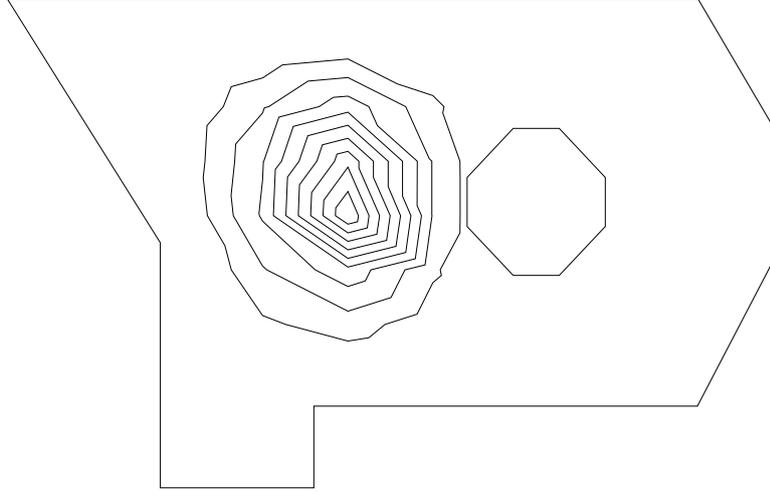


Fig. 10. Iso-influence contours for material and heat source. The contours are loops around an influence hill at equal value.

loop around a hill, gradient vectors point up it. For example, Figure 11 shows the gradient vector directions for the hill in Figure 4. The lengths of these vectors, not shown here, indicate the local steepness of the hill. The directions of the vectors indicate the directions of steepest ascent up the hill. Local operations on fields and neighborhood graphs (refer again to Table 1) support local estimation of gradient vectors by approximating the spatial derivatives. For example the centered differences approach estimates a second derivative by the difference between two adjacent estimates of the first derivative:

$$\frac{\partial \phi}{\partial x} \approx \left( \frac{\vec{\phi}_{i+1} - \vec{\phi}_i}{x_{i+1} - x_i} - \frac{\vec{\phi}_i - \vec{\phi}_{i-1}}{x_i - x_{i-1}} \right) / \left( \frac{x_{i+1} - x_i}{2} + \frac{x_i - x_{i-1}}{2} \right) \quad (5)$$

where  $i - 1, i, i + 1$  are horizontally-adjacent points (a similar equation approximates the derivative with respect to  $y$ ).

#### 4 Control Placement Design

The first design task considered here is that of designing a control placement. For the thermal domain, control placement design uses a description of a material's geometry, conduction properties, boundary conditions, and design constraints, in order to place heat sources. The placement of the heat sources affects their ability to achieve a desired temperature distribution by parametric adjustment. For example, if all the heat sources are clumped at one end of the material, they cannot adequately control the area at the other end of the material. Similarly, such a clumping makes it hard to individually determine

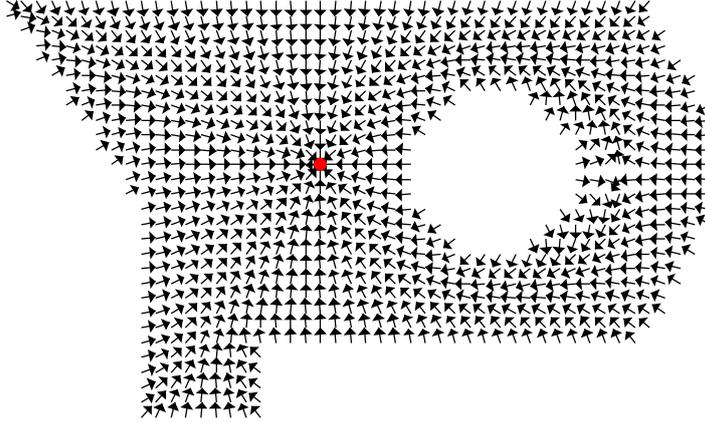


Fig. 11. Gradient vector directions for material and heat source. The vectors point up an influence hill. Vector magnitudes are normalized to 1.0 here, in order for the directions to be apparent.

control actions, since the actions taken by one control strongly affect the necessary actions of another (e.g. heat from one source affects the area another source is trying to control). Furthermore, as discussed in the introduction, in order to scale up to massive sets of distributed controls, it is necessary to reason about controls as independently as possible.

Based on this insight, the design objective considered here is that of placing controls so that they minimally interfere with each other. The approach to achieving this objective is to decompose a problem domain  $S$  into a set  $P = \{R_1, R_2, \dots, R_n\}$  of *decoupled, atomic* subregions  $R_i$ , and then independently design controls (placement and parameters) for the separate subregions. Intuitively, regions are considered decoupled if the exact control design in one region is fairly independent of the exact control design in another, and a region is considered atomic if it needs no further decomposition — control design for the region yields adequate control of the region.

More specifically, we seek a decomposition maximizing a quality score utilized by Shi and Malik [21] for image segmentation. 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:

$$q = \prod_{R \in P} \sum_{c \in R} \frac{\sum_{r \in R} \mathcal{I}(c, r)}{\sum_{s \in S} \mathcal{I}(c, s)} \quad (6)$$

For each control node in a region, divide its influence on nodes in its own region by its total influence. Summing that over each region yields an estimate of the

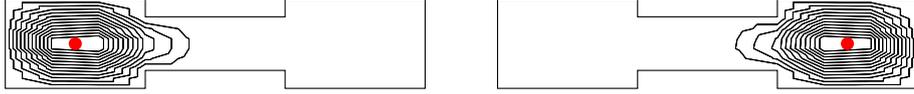


Fig. 12. Temperature fields exhibit structures in response to heat source probes. The narrow channel in the dumbbell constrains the heat flows from the two probes.

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.

**Definition 5 (Control Placement Design)** Control placement design for spatial domain  $S$ , behavioral model  $M$ , and design constraints  $\Sigma$  (Eq. 1) yields a number  $n$  and set of control locations  $C = \{c_1, c_2, \dots, c_n\}$  maximizing decomposition quality (Eq. 6).

Influence graphs are used to perform this decomposition; refer again to Figure 5 for an overview. We now step through the key ideas necessary for the decomposition algorithm, to be given in Table 2.

#### 4.1 Control Probes

For a temperature field to exhibit structures, heat sources must be applied; then an influence graph can be constructed. For example, Figure 12 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 algorithms in the following subsections are based on the response of temperature fields to such *control probes*.

**Definition 6 (Control Probe)** A control probe is a sample control placed in a domain in order to estimate effects of other potential controls. This yields an influence graph  $\mathcal{I} = (S, C, E, w)$  for probes  $C$  in a domain  $S$ .

The number and placement of control probes affect the structures uncovered in temperature fields, and thus the quality of the resulting control design. Probe locations can be chosen either statically or dynamically. For example, static probe locations can be chosen at random or based on the size of the field discretization (e.g. every 10 units). Dynamic probe locations can be chosen in order to gather more information about inadequately explored regions or to disambiguate inconsistent interpretations. This allows potentially better results at the expense of more implementation complexity and run-time cost.

Probes serve as representatives for the effects of arbitrarily-placed controls. That is, we assume that a control at some un-probed location will have similar

effects to controls at “nearby” probed locations, where nearness is measured in terms of amount of influence, rather than only geometry.

**Definition 7 (Primary Controls)** *For an influence graph  $\mathcal{I} = (S, C, E, w)$ , the primary controls for a location  $s \in S$  is the set of controls  $\{c \in C \mid \mathcal{I}(c, s) > \varepsilon_s\}$ .*

While we used a fixed threshold for  $\varepsilon_s$  in our implementation, other possible implementations could be based on the near field or on a standard deviation above the average influence from all controls. Note that nodes can have multiple primary controls.

The quality of the approximation of controls at arbitrary locations by representative primary controls depends on geometry and material properties. Since the influence graph encapsulates the effects of geometry and material conditions, it provides a natural mechanism for reasoning about approximation quality. In particular, experimental results presented later in this section show the trade-off between number of probes (and thus approximation quality, assuming that the quality of an approximation for a location improves with more, closer probes) and resulting quality of control design.

By taking control probes as representatives of control placement effects on a field, the problem of decomposing the domain into regions can be reformulated into one of partitioning probes into equivalence classes. Each equivalence class of probes serves as a representative for the effects of its *controlled region*, the set of nodes for which the probes are primary controls. A good decomposition produces probe classes whose controlled regions are decoupled from the controlled regions of other classes, and which have no acceptable subclasses.

#### 4.2 Evaluating Control Decoupling

The first criterion for evaluating a decomposition is that each region be decoupled from other regions. In terms of control probe equivalence classes, decoupling will be evaluated by considering independence of control placement and independence of control parameters.

To evaluate independence of control placement, consider the influence gradient vectors induced by a set of probes; Figure 13 shows an example for two probes. While the flows are different in direction near the locations of the two probes, they are quite similar in direction 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 comparing the angular difference between gradient vectors

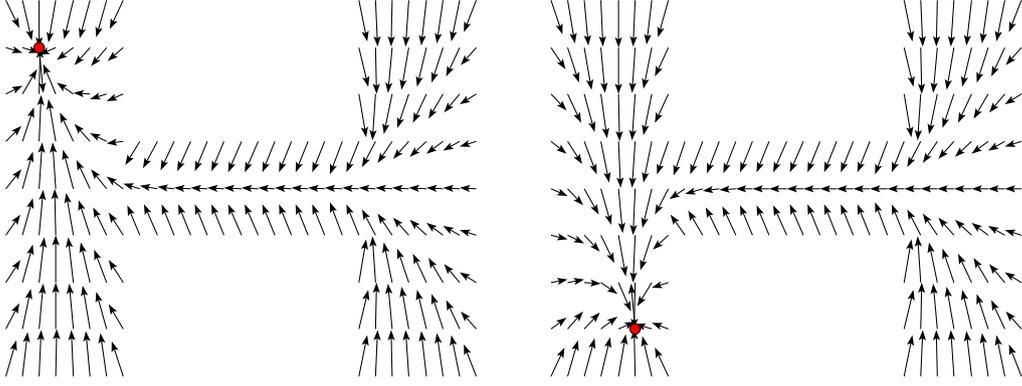


Fig. 13. Similarity of flows in the far fields of control probes suggests indistinguishability of control placement. Vector directions in the right half of the dumbbell are very similar for both probe locations.

produced by different probes:

$$\text{place\_sim}(c_1, c_2) = \sum_{s \in S} \nabla \mathcal{I}(c_1, s) \cdot \nabla \mathcal{I}(c_2, s), \quad (7)$$

where  $\nabla \mathcal{I}(c, s)$  is the gradient vector field for the influence from  $c$ , evaluated at  $s$  (see Section 3.4), and  $\cdot$  represents dot-product of the vectors. This measure, for which larger values indicate more similar vector fields, evaluates the indistinguishability of exact 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, recall the distinction between a probe's near field and its far field: the far field is only weakly influenced by the probe, and thus can be effectively decomposed from it. Alternatively, it makes sense to group together probes that have significant overlap in their near fields. This overlap can be measured by element-wise comparing influence value differences and summing the results.

$$\text{param\_sim}(c_1, c_2) = m_{12} - \sum_{s \in S} |\mathcal{I}(c_1, s) - \mathcal{I}(c_2, s)|, \quad (8)$$

where  $m_{12} = \sum_{s \in S} |\mathcal{I}(c_1, s)| + \sum_{s \in S} |\mathcal{I}(c_2, s)|$  inverts the metric so that more similar influence yields a larger number.

**Definition 8 (Decoupling)** *Two sets of controls  $P_i$  and  $P_j$  of a partition are decoupled if  $\forall c_1 \in P_i, c_2 \in P_j : (\text{place\_sim}(c_1, c_2) < \varepsilon_{\text{place}}) \wedge (\text{param\_sim}(c_1, c_2) < \varepsilon_{\text{param}})$ .*

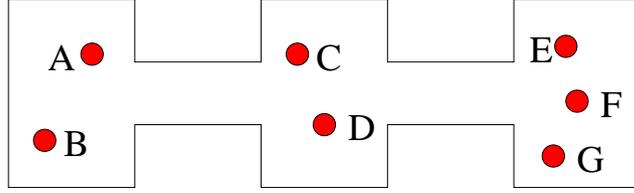


Fig. 14. Control probe placement with potential non-atomic partition  $\{\{A, B, C, D\}, \{E, F, G\}\}$  and atomic decomposition  $\{\{A, B\}, \{C, D\}, \{E, F, G\}\}$ .



Fig. 15. Overlapping near fields indicate probe class atomicity. (a) Non-atomic class (the probes are decoupled). (b) Atomic class (the probes are not decoupled).

### 4.3 Evaluating Region Atomicity

Each region in a partition must be decomposed far enough. For example, in Figure 14 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  $\{\{A, B\}, \{C, D\}\}$ .

**Definition 9 (Atomic Decomposition)** *A set  $P \subset S$  in a partition is atomic if for all  $P_1, P_2 \subset P$  with  $P_1 \cap P_2 = \emptyset$ ,  $P_1$  and  $P_2$  are not decoupled, according to Definition 8.*

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 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. A more efficient and empirically effective method is to allow grouping of pairs of probes only if their near fields sufficiently overlap, as shown in Figure 15.

```

function cluster_probes( $\mathcal{I} = (S, C, E, w)$ ,  $G \subset C \times C$ )
  Let  $P = \{\{c\} \mid c \in C\}$ 
  Repeat:
    Let  $nbr_{ij} = (P_i \neq P_j) \wedge (\exists c_1 \in P_i, c_2 \in P_j : (c_1, c_2) \in G) \wedge \mathbf{atomic}(P_i \cup P_j)$ 
    Let  $sim_{ij} = (\alpha \mathbf{place\_sim}(P_i, P_j) + \mathbf{param\_sim}(P_i, P_j))$  if  $nbr_{ij}$  is true
      or 0 otherwise
    Let  $best_i = \arg \max_j sim_{ij}$ 
    Let  $merge = \{P_i \cup P_j \mid best_i = j \wedge best_j = i\}$ 
    Let  $keep = \{P_i \mid \exists j : best_i = j \wedge best_j \neq i\}$ 
    Set  $P$  to  $merge \cup keep$ 
  Until  $merge = \emptyset$ 
  Return  $P$ 

```

Table 2

Algorithmic description of probe clustering.

#### 4.4 Control Probe Partitioning

Based on these criteria, control probes can be clustered into decoupled, atomic equivalence classes. We introduce an algorithm for performing this clustering, using the SAL neighborhood graph and classification mechanisms. Start with each probe in its own class, and form a neighborhood 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 2 provides pseudocode for this algorithm, and Figure 16 shows the data flow. Figure 17 illustrates a sample probe neighborhood graph, Figure 18 depicts sample influence gradients for two probes, and Figure 19 shows the controlled regions for equivalence classes of probes after the merging process.

#### 4.5 Performance

The influence-based decomposition algorithm has 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.

One important question about the design algorithm is the impact of the number of control probes on the effectiveness of the resulting design. To test this, 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

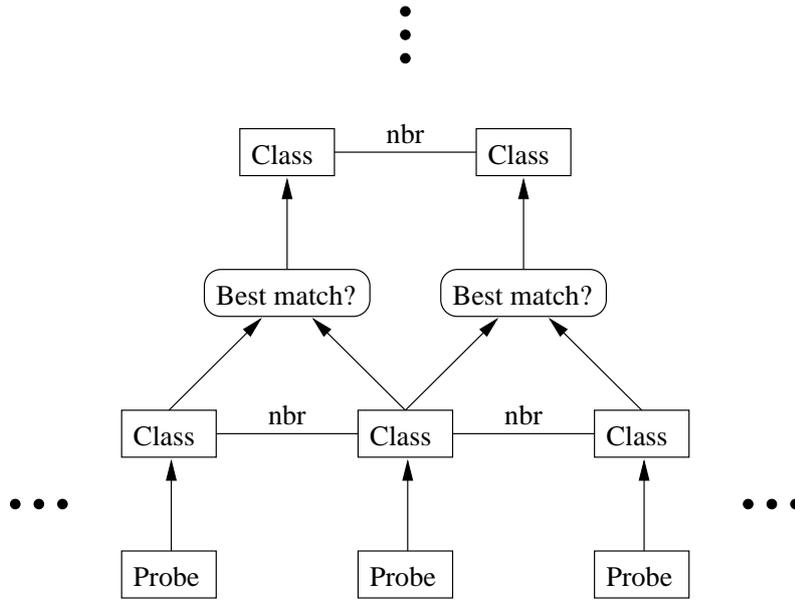


Fig. 16. Data flow for probe clustering algorithms: repeatedly merge best-match pairs of classes.

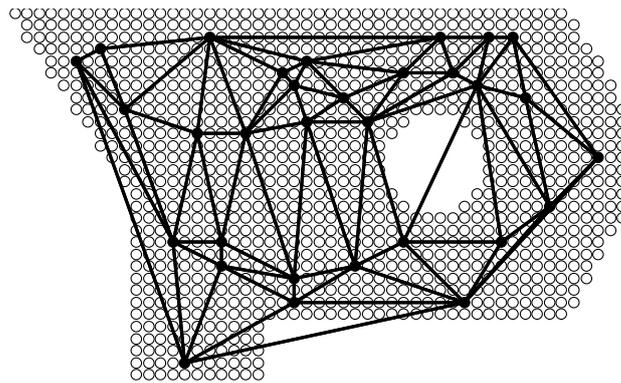


Fig. 17. Probe merging example: probe neighborhood graph.

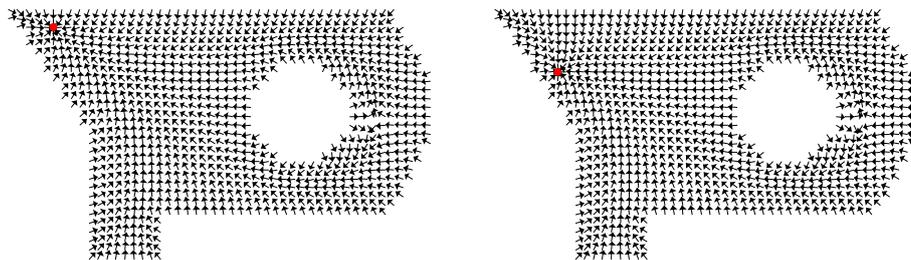


Fig. 18. Probe merging example: influence gradient vectors from two probes.

number of probes and error/variance.

Data for three sample problems are given here: a plus-shaped piece of material, a P-shaped piece of material, and an anisotropic (non-uniform conduction

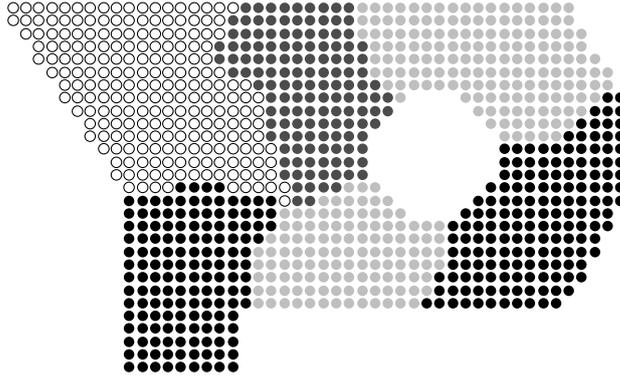


Fig. 19. Probe merging example: region decomposition after merging.

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 probe-based algorithm described above was used to decompose the input domain. The near field of each probe was set to all nodes with influence at least 10 percent of the peak. The probe neighborhood graph was a Delaunay triangulation. Similarity measures between probe classes compared only flow vector direction differences as in Eq. 7. Merging was performed until four classes remained.

#### 4.5.1 Decomposition Quality

The goal of the 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 all regions (decomposed) and has no internally independent regions (atomic). Recall that Eq. 6 provides a metric indicating how well this goal is achieved.

To provide a baseline for comparing the performance of our approach, we implemented the technique of Shi and Malik [21]: apply spectral partitioning to a matrix equivalent to an influence graph with edges weights modulated by total influence from corresponding control nodes. Intuitively, this approach partitions the influence graph, removing edges so that the resulting connected regions maximize internal influence (atomic) and minimize external influence (decomposed). Shi and Malik showed that this yields a partition that generally reaches a nearly-optimal decomposition according to Eq. 6.

Table 3 provides raw performance data: the average and standard deviation of the decomposition quality over the set of trial runs. Figure 20 illustrates the variation in error and standard deviation with respect to different numbers of control probes.

	Spectral	Infl4	Infl8	Infl16	Infl32
Plus					
error	1.0	0.97	1.06	1.21	1.17
std dev	n/a	0.30	0.17	0.12	0.10
P					
error	1.0	0.68	0.77	0.86	1.11
std dev	n/a	0.063	0.15	0.12	0.066
Bar					
error	1.0	0.56	0.65	0.76	0.85
std dev	n/a	0.26	0.10	0.073	0.024

Table 3

Performance data for decomposition quality: relative average and standard deviation of decomposition quality. Spectral uses spectral partitioning to decompose a full influence graph, while Infl4–Infl32 use the probe-based method with 4–32 randomly placed probes.

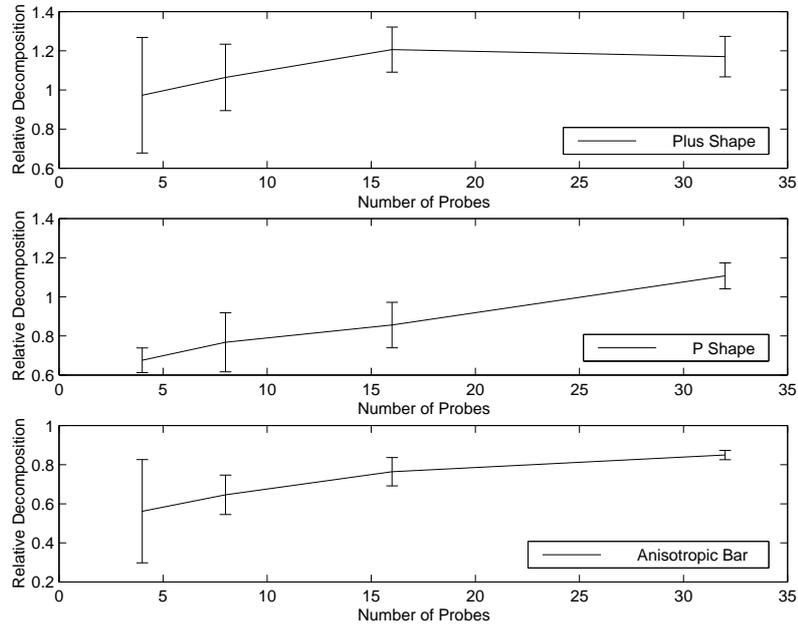


Fig. 20. Performance data indicate that the influence-based decomposition algorithm supports trading decomposition quality for computation. Decompositions achieve quality comparable to spectral partitioning, but with an influence graph for a small number of probes rather than a full influence graph.

For all three problems in Figure 20, the average quality 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 the amount of computation versus the resulting decomposition quality. With enough probes, the decomposition quality is roughly equivalent to that of spectral partitioning. It is worth noting that spectral partitioning requires computation of a matrix corresponding to a full influence graph (from every node to every other node), rather than just influence from a small number of probes. The spectral partitioning approach also requires solving a general eigenvalue problem for the influence matrix.

#### 4.5.2 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.

Recall that the goal of decomposition is to determine atomic, decoupled regions, each to be controlled separately. To generate a simple control placement from a given partition, we placed controls in the “center of influence” of each region of the partition. The center of influence is like the center of mass, but weighted with total influence from the probes, rather than mass, at each point.

To compare the performance of our approach to a standard technique, we implemented a simulated annealing [22] design algorithm. A configuration consists of a heat source placement; each step moves one source and tests whether or not the move improves the ability of the design to meet the desired temperature profile. The annealing process was run for 100 steps, requiring computation equivalent to placing 100 probes.

Table 4 provides the raw performance data, including the average error (sum of squared difference between actual temperature profile and desired temperature profile) and the standard deviation in the error over the set of trial runs. Figure 21 illustrates the variation in error and standard deviation with respect to different numbers of control probes.

As with decomposition quality, the average and standard deviation of control quality tend to improve with the number of probes. With enough probes,

	Anneal	Infl4	Infl8	Infl16	Infl32
Plus					
error	1.0	1.25	1.07	1.03	1.0
std dev	0.014	0.083	0.069	0.028	0.029
P					
error	1.0	1.16	1.17	1.03	0.99
std dev	0.014	0.077	0.124	0.014	0.027
Bar					
error	1.0	1.42	1.11	1.0	0.99
std dev	0.011	0.24	0.13	0.085	0.050

Table 4

Performance data for decomposition-based control design: relative average and standard deviation of error. Anneal is a simulated-annealing based optimizer, while Infl4–Infl32 use the influenced-based decomposition method with 4–32 randomly-placed probes.

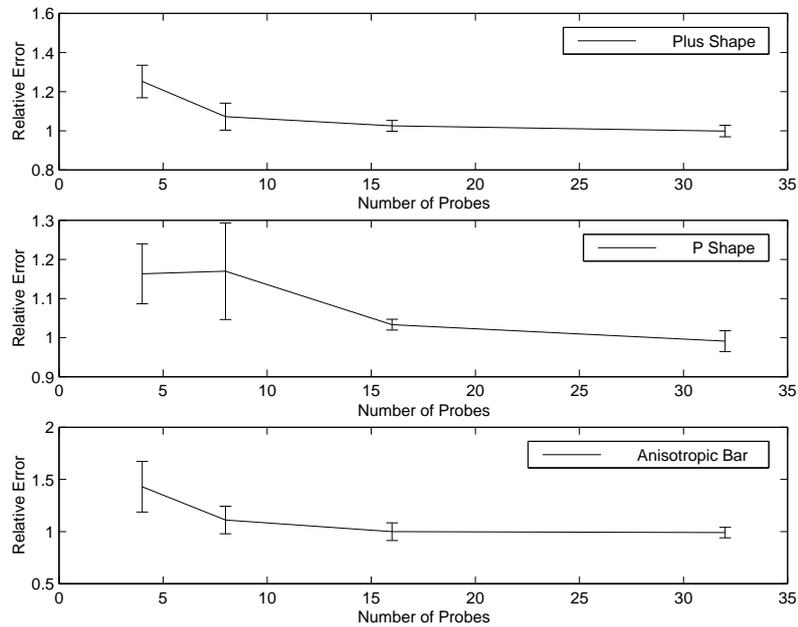


Fig. 21. Performance data indicate that decomposition-based control placement design supports trading control quality for computation. Designs achieve performance comparable to simulated annealing, but with a small, fixed number of field evaluations.

the quality is commensurate with that of simulated annealing. A major difference is that the decomposition-based approach uses a small, fixed number of function evaluations. In both cases, only the global control placement was designed; local placement adjustments could somewhat reduce the error.

#### 4.6 Discussion

The control placement design algorithm decomposes a domain based on influence graph structures for a set of control probes. It uses influence graphs to measure similarity of effects on the field from classes of probes. Based on these measures, it identifies minimal equivalence classes of probes that are mutually independent with respect to control placement and parameter design.

In order to provide a consistent basis for comparison with other algorithms (i.e. spectral partitioning and simulated annealing) which require a fixed number of partitions, the performance data presented here applied probe merging until a desired number of regions was reached. However, our framework has the advantage of a metric, probe class atomicity, that indicates the best number of partitions.

Spectral partitioning and other graph partitioning algorithms [23–26] could be used directly for control design: partition a field into subfields and place a control in each subfield. However, these approaches account for topology and perhaps geometry, but not material properties. This section presented spectral partitioning on the influence graph (rather than on the domain) as a baseline for decomposition performance data. As previously mentioned, our approach is much more efficient, using influences for only a small number of probes, rather than for every node. The influence-based partition process could be implemented by encoding probe similarity metrics in graph edges and then applying a graph partitioning algorithm. However, the merging algorithm presented here is simple, efficient, and effective.

The influence-based design algorithm searches the design space in a much different manner from that of other combinatorial optimization algorithms, such as genetic algorithms [27] and simulated annealing [22]. Rather than searching the space of all possible combinations of source locations, our approach combines results from a small set of control probes, develops a global description of the domain, and partitions it appropriately. We explicitly form 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 con-

strained influence flows through a field.

Both decomposition quality and control quality increase somewhat asymptotically with the number of probes. The major computational cost is in computing probe influences, rather than in merging probe classes. This suggests a modified control design algorithm that iteratively increases the number of probes, checks the resulting decomposition at each step, and halts when the quality stabilizes. This algorithm avoids dependence on a fixed number of probes and follows a trade-off curve between computation and control quality.

## 5 Control Parameter Design

Given a control placement design, the next task is to optimize control parameters in order to satisfy the design objectives. Consider the control objective of maintaining a specified temperature distribution for an extended period of time, as is the case for rapid thermal prototyping [14]. This task can be broken into two parts: design-time computation of *set points* around which the heat source outputs will vary, and run-time feedback control of the actual heat source outputs based on local, linearized models derived at those set points. This section considers the computation of the set-point heat source values, leaving the feedback control to standard engineering techniques (e.g. as in [14] or [13]).

**Definition 10 (Control Parameter Design)** Control parameter design for a set of controls  $C = \{c_1, c_2, \dots, c_n\}$  with respect to a desired set-point temperature profile  $T$  and allowable error  $\epsilon$  yields set-point heat outputs  $U = \{u_1, u_2, \dots, u_n\}$  minimizing the deviation from  $T$  and ensuring that it is less than  $\epsilon$ .

Control parameter design requires simultaneous optimization of many parameters (the heat source values). While algorithms for multi-parameter optimization exist [28], they are computationally expensive for large problems and difficult to parallelize for distributed applications. This section demonstrates that structural knowledge, in the form of the influence graph, significantly improves the performance of a basic decentralized optimization algorithm.

A simple decentralized optimization algorithm repeatedly tests adjustments to control outputs and chooses those that minimize the error (e.g. sum of squared difference of the resulting temperature from the desired profile). Table 5 and Figure 22 summarize the algorithm and its data flow. Remember that the optimization processes are decentralized, so that each heat source adjusts itself independently, taking a step towards what it thinks minimizes error. The desired temperature  $T$  is represented as  $\{(s_i, a_i) \mid s_i \in S\}$  where  $a_i$  is the

```

function decentralized_opt( $\mathcal{I} = (S, C, E, w), T, \epsilon$ )
  Let  $U = \{(u_1, 0), (u_2, 0), \dots, (u_n, 0)\}$ 
  Repeat:
    For each  $c_i \in C$ :
      Let  $\Delta_i = \{\delta_i, -\delta_i\}$ 
      Increment  $u_i$  by  $\arg \min_{\delta \in \Delta_i} \|T - \mathbf{temp}(\mathcal{I}, U|_{u_i=u_i+\delta})\|$ 
  Until  $\|T - \mathbf{temp}(\mathcal{I}, U)\| < \epsilon$ 
  Return  $U$ 

```

Table 5

Algorithmic description of decentralized optimization algorithm.

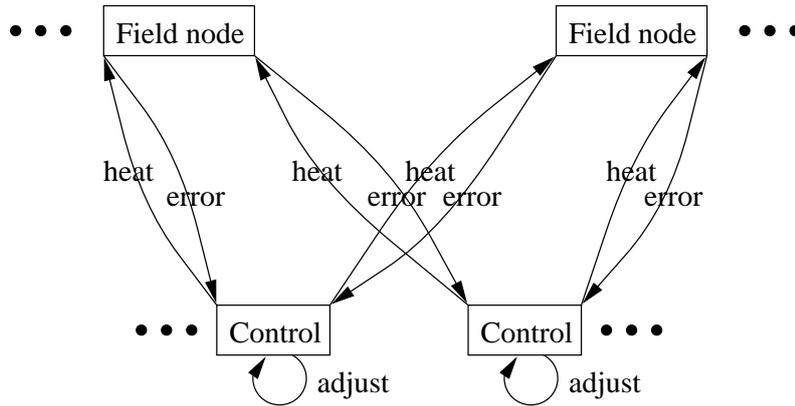


Fig. 22. Data flow for the basic decentralized optimization algorithm: adjust control values based on the error in the field resulting from different heat outputs.

temperature at  $s_i$ . We assume a function **temp** that evaluates the temperature field given a set of parameter values for an influence graph (e.g. by relaxation, as in Section 2). Our implementation uses a set  $\Delta_i$  of control adjustments proportional to the current control value; other sets are possible.

In the next three sections, the influence graph mechanism will be used (1) to avoid redundant computation during field evaluation, (2) to reduce communication among sources and field nodes, and (3) to support cooperation among local optimization processes for the sources.

### 5.1 Efficient Field Evaluation

During each step of an iterative optimization process, the field is evaluated using a relatively expensive, iterative relaxation method, as discussed in Section 2. However, recall that an influence graph caches the dependence of field nodes on normalized sources, and that the field is determined by a linear superposition of source effects. Thus the field value for a node can be calculated by summing together the weights of influence graph edges coming into the

```

function decentralized_opt_efficient_field_evaluation( $\mathcal{I} = (S, C, E, w), T, \epsilon$ )
  Let  $U = \{(u_1, 0), (u_2, 0), \dots, (u_n, 0)\}$ 
  Let  $F = \{(s_1, 0), (s_2, 0), \dots, (s_n, 0)\}$ 
  Repeat:
    For each  $c_i \in C$ :
      Let  $\Delta_i = \{\delta_i, -\delta_i\}$ 
      Let  $dF_\delta = \{(s, \delta \mathcal{I}(c_i, s)) \mid s \in S\}$  for  $\delta \in \Delta_i$ 
      Increment  $u_i$  by  $\arg \min_{\delta \in \Delta_i} \|T - (F + dF_\delta)\|$ 
      Increment  $F$  by  $dF_\delta$  for chosen  $\delta$ 
  Until  $\|T - F\| < \epsilon$ 
  Return  $U$ 

```

Table 6

Algorithmic description of decentralized optimization algorithm incorporating efficient field evaluation.

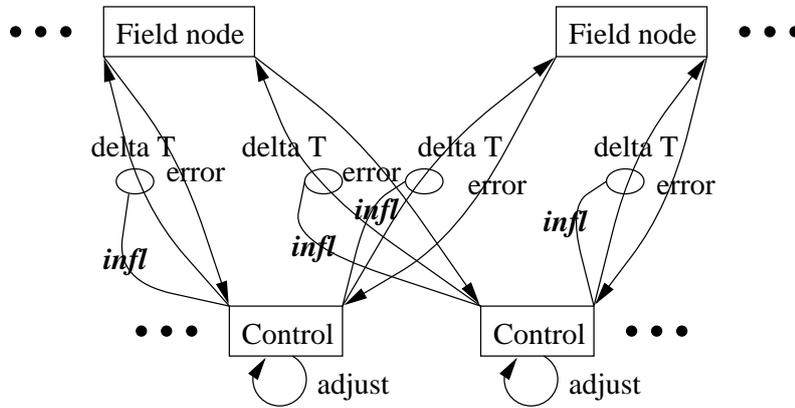


Fig. 23. Data flow for efficient field evaluation: adjust temperatures based on changes in source values and influence graph information.

node, scaled by the control source values. This computation is extremely fast and results in a drastic speed-up in computation.

Table 6 summarizes the field-evaluation algorithm, and Figure 23 illustrates the data flow for the modified optimization algorithm. To determine the impact of a different heat output, a source calculates the resulting temperature change for each field node, based on influence graph edge weights.

The influence graph essentially pre-computes and caches the inverse of the capacitance matrix of the field (Eq. 3). An important distinction is that it does this *in a decentralized fashion*, without ever forming a global matrix for the temperature field or the sources. This representation is particularly efficient when sources are sparse.

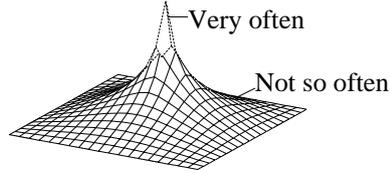


Fig. 24. Influence structure supports reduced communication between control node and field nodes: a control should communicate more frequently with field nodes it strongly influences.

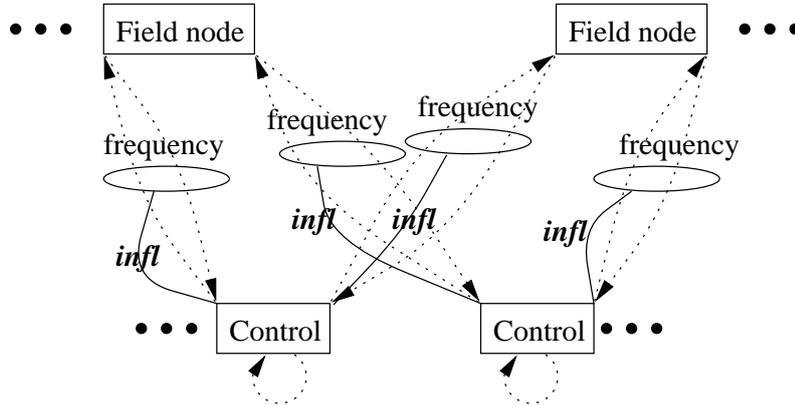


Fig. 25. Data flow for reduced communication optimization: modulate frequency of source-field communication by influence strength.

## 5.2 Reduced Communication

At each optimization step, a source must estimate the error caused by an adjustment to the source value, with respect to the current state of the temperature field. The source can consult the entire temperature field for the current error, and then adjust the values throughout the field when it changes, but that requires much communication. Alternatively, it can consider only a local region assigned to it (e.g. the region for which it was designed in Section 4), but that ignores the influence on the other regions. Better yet, a source can communicate more frequently with those field nodes it most strongly affects, as shown in Figure 24. If a source only weakly affects a temperature node, we need not assign it much blame/credit for the error at that node.

Table 7 summarizes the new field-evaluation algorithm, and Figure 25 illustrates the new data flow during control optimization. The frequency of source-field communication is proportional to the amount of influence. Decreasing frequency decreases overall communication costs, but increases the potential for error due to underestimated source effects.

There are several possible strategies for establishing source node to field node communication. The most basic method, used in Tab. 7, computes communication frequency as a function of the weight along the influence graph edge.

```

function decentralized_opt_reduced_communication( $\mathcal{I} = (S, C, E, w), T, \epsilon$ )
  Let  $U = \{(u_1, 0), (u_2, 0), \dots, (u_n, 0)\}$ 
  Let  $F = \{(s_1, 0), (s_2, 0), \dots, (s_n, 0)\}$ 
  Let  $n = 0$ 
  Repeat:
    For each  $c_i \in C$ :
      Let  $\Delta_i = \{\delta_i, -\delta_i\}$ 
      Let  $V_i = \{s \in S \mid n = 0 \pmod{\alpha/\mathcal{I}(c_i, s)}\}$ 
      Let  $dF_\delta = \{(s, \delta \mathcal{I}(c_i, s)) \mid s \in V_i\} \cup \{(s, 0) \mid s \notin V_i\}$  for  $\delta \in \Delta_i$ 
      Increment  $u_i$  by  $\arg \min_{\delta \in \Delta_i} \|T - (F + dF_\delta)\|$ 
      Increment  $F$  by  $dF_\delta$  for chosen  $\delta$ 
      Increment  $n$ 
  Until  $\|T - F\| < \epsilon$ 
  Return  $U$ 

```

Table 7

Algorithmic description of decentralized optimization algorithm incorporating reduced-communication field evaluation.

This requires each source to communicate with each field node (some more frequently than others). A more qualitative method forms equivalence classes of field nodes based on influence (iso-influences) for each source, and treats the regions equivalently with respect to communication frequency. Now communication paths only exist between sources and regions. An even more qualitative method forms equivalence classes of field nodes based on which source has the strongest influence, again treating regions equivalently with respect to communication frequency. With this assignment, each source communicates only with its own region and with other sources, which pass information on to their regions.

### 5.3 Joint Optimization

While the decentralized optimization algorithm seeks to independently optimize sources, in reality there is *coupling*: the heat from one source affects the temperature throughout the entire field and thus influences the actions taken by other sources (refer again to Figures 7(b) and (c)). Independent optimization of coupled sources might require more iterations to converge, as the sources make seemingly independent choices which they later find to be wrong due to dependencies. Even worse, sources might converge to sub-optimal values, where no independent actions help, but cooperative actions would.

As a particular example of cooperative optimization, consider the “ridge problem” faced by optimization techniques. An example manifestation of the ridge problem in the temperature control domain, illustrated in Figure 26, occurs when independently increasing the value of one source increases the total er-

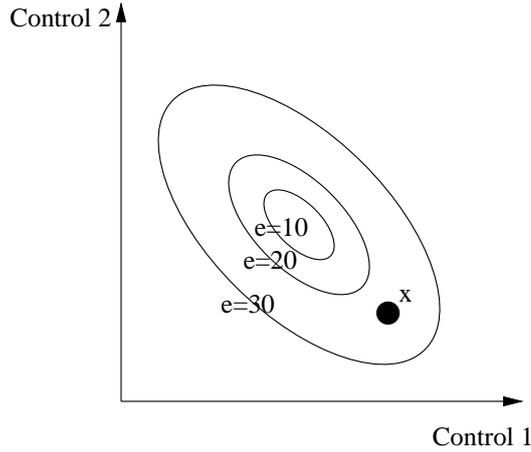


Fig. 26. Ridge problem in control optimization: starting at  $x$ , independently increasing or decreasing either control output increases the error, but jointly decreasing control 1 and increasing control 2 decreases the error.

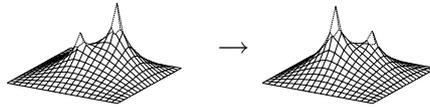


Fig. 27. Supervisor control shifts control value from one control to another.

ror and independently decreasing the value of another source also increases the total error, but jointly increasing the one and decreasing the other decreases the total error. This is due to coupling between the areas influenced by the sources: the joint modification maintains a similar temperature profile in the overlap area and benefits other areas. By cooperatively optimizing, the optimizer walks along the ridge in the error landscape.

Joint optimization can be programmed by incorporating *supervisors* into the decentralized optimization algorithm. A supervisor is a control node whose action is to shift control output from one control to another, as in Figure 27. Supervisors can be placed, for example, between pairs of very close controls, or between pairs of controls whose influence hills are highly overlapping. Since a supervisor's action shifts control value from one control to another, its influence is simply the influence difference between the two nodes. Figure 28 summarizes the data flow for this approach. The pseudocode is the same as before, except that the control adjustments  $\Delta_i$  for a control  $c_i$  supervising controls  $c_j$  and  $c_k$  shift an amount  $\delta_i$  of control between  $c_j$  and  $c_k$ , so that either  $u_j$  is incremented by  $\delta_i$  and  $u_k$  decremented by  $\delta_i$ , or else  $u_j$  is decremented by  $\delta_i$  and  $u_k$  is incremented by  $\delta_i$ .

Supervisors implement source cooperation and help avoid optimization ridges by shifting heat from one source to the other based on the error profile in the field. This approach could be extended to the addition of supervisor controls that shift heat among groups of sources rather than between pairs. Note that a

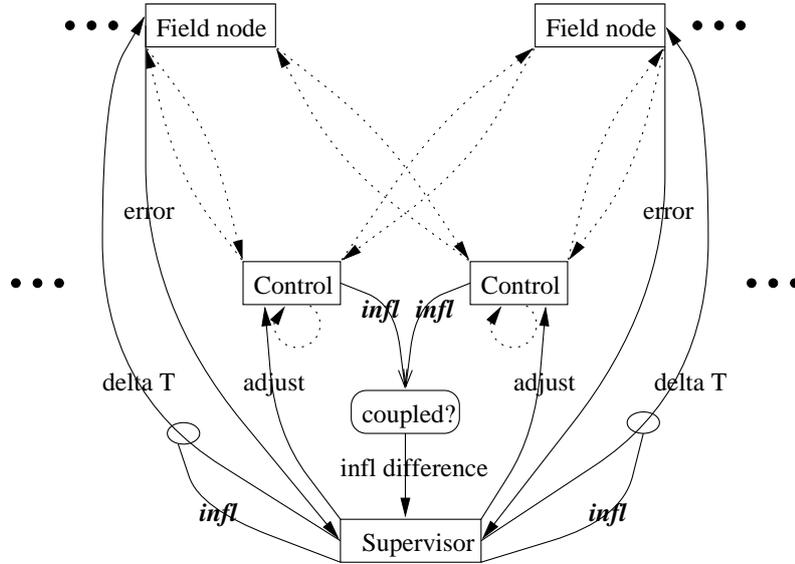


Fig. 28. Data flow for joint optimization: form supervisor nodes for tightly-coupled sources; optimize supervisors, shifting heat from one source to another based on errors in the field.

supervisor only needs to be established for a group of sources that are tightly coupled. Further extensions could let supervisors check for cooperation less frequently or recognize a potential need for cooperation (for example, too much heat near one source and not enough near the other) before attempting an expensive adjustment.

#### 5.4 Performance

The control parameter design algorithms, when applied to several problems, result in competitive designs and run-time performance.

For a distributed optimization problem with  $m$  sources and  $n$  field objects, the basic algorithm requires on the order of  $KLmn$  units of computation, where  $K$  and  $L$  are the numbers of iterations for the optimization and relaxation processes respectively.  $K$  and  $L$  depend on properties of the problem including the size of the field, the number of controls, the material properties, and the geometry; in the test cases below,  $K$  is roughly between 10 and 100, while  $L$  is roughly between 100 and 1000. Using the influence graph to eliminate repeated relaxation, the algorithm scales as  $Kmn$ . Exploiting the communication structure, the cost is reduced to  $KmC_n$  for a smaller  $C_n$ , the number of field objects with which each source communicates, possibly independent of  $n$ . By cooperating among the local optimizers, the number of iterations  $K$  is further reduced.

#### 5.4.1 *Efficient Field Evaluation*

As expected, the influence graph mechanism results in enormous savings during repeated decentralized field evaluations. For example, in our implementation, it takes about 49 seconds to iteratively solve for the temperature in a field with about 1000 nodes, while it takes less than 0.02 seconds using the influence graph. The speed-up would be similar for any implementation — it illustrates the vast savings obtained by simply summing influences rather than re-solving the linear equations each time.

#### 5.4.2 *Reduced Communication*

Influence graphs significantly reduce communication during source optimization. Table 8 summarizes results for steady-state parametric design on a regular 20x20 discretized thermal field. While the domain evaluated here is square, similar results hold for other shapes — the important factor is the locality of the thermal hills encapsulated in the influence graph. Data for three problems are provided: four sources near the corners of the grid, four sources near the center of the grid, and sixteen sources tiled over the grid. These three problems exhibit varying thermal hill shapes and thus varying ability to reduce communication. Three performance results are shown for each test: the number of iterations for convergence, the total source-field node communication, and the average squared error across the thermal field. Actual run-time is roughly proportional to the number of communications.

The first two optimizers evaluated (Gauss-Newton and Broyden-Fletcher-Golfarb-Shanno) are Matlab-based implementations of two standard multi-parameter optimization algorithms (Simplex search optimization is not included because it fails to converge within 300 steps on all of these tests.) Note that the Matlab algorithms are not decentralized; in order to compare the amount of communication, each source is considered to communicate with each non-boundary field node each iteration. The influence-based optimizers use an implementation with varying amounts of communication: Inff1 updates each field object based on each source every iteration, while Inff2–Inff4 update field objects with frequency proportional to influence, with different constants of proportionality. Performance numbers are relative to Gauss-Newton (lower is better).

These results show that on representative multi-parameter optimization problems, the structure-based decentralized optimizers compete well with the centralized optimization techniques in both speed and error, while greatly reducing the amount of communication among distributed optimization processes. Figure 29 charts the trade-off between communication and error in the four influence-based optimizers on these problems. Naturally, error increases as communication decreases, but there is quite a long flat area where the com-

	GN	BFGS	Infl1	Infl2	Infl3	Infl4
4-corner						
iterations	1.0	.7368	1.105	1.0	.8947	1.0
communication	1.0	.7368	1.105	.4293	.0947	.0465
error	1.0	1.0	1.0	1.004	1.125	1.243
4-center						
iterations	1.0	.7	1.2	1.05	.95	1.55
communication	1.0	.7	1.2	.8103	.2693	.2002
error	1.0	1.0	1.0	1.001	1.047	1.134
16-tiled						
iterations	1.0	3.804	.6429	.875	.8214	1.339
communication	1.0	3.804	.6429	.5573	.2051	.1346
error	1.0	1.0	1.003	1.015	1.022	1.157

Table 8

Performance data for communication reduction in optimization: relative number of iterations, number of communications, and resulting error for different optimization methods for representative problems. GN (Gauss-Newton) and BFGS (Broyden-Fletcher-Golfarb-Shanno) are Matlab-based centralized optimizers. Infl1–Infl4 are influence graph-based decentralized optimizers with communication rates proportional to influence, with varying constants of proportionality. Values are relative to Gauss-Newton.

munication decreases without a serious impact on the error. In problems with larger domains, there will be even fewer field nodes strongly influenced by a source (depending on geometry and material properties), providing even greater potential savings.

### 5.4.3 Joint Optimization

Influence graphs also support cooperative source optimization. Table 9 provides data for representative problems with tight coupling among sources due to material properties and source spacing. The sources are placed in four different configurations on a 20x20 grid: a pair of sources at the edge, four sources tightly packed near a corner, eight sources in a line across the middle, and sixteen sources tightly packed near the center. The results from the two (centralized) Matlab optimizers are provided for reference; the first influence-based optimizer does not cooperatively optimize, while the second one places a supervisor between each neighboring pair of sources. BFGS fails to converge for the 8-line test case.

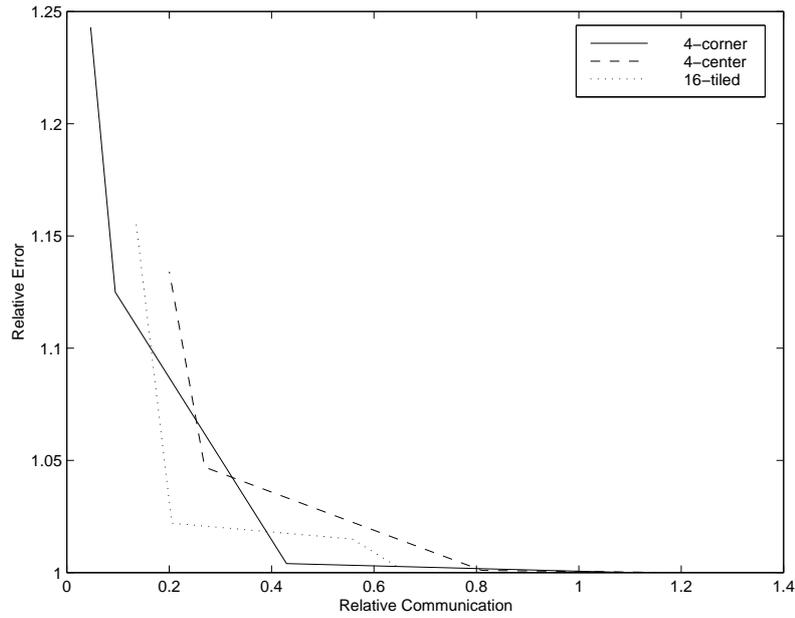


Fig. 29. Performance data indicate that influence graphs support trading optimization quality for amount of communication. In the flat area, amount of communication is greatly reduced with little impact on error.

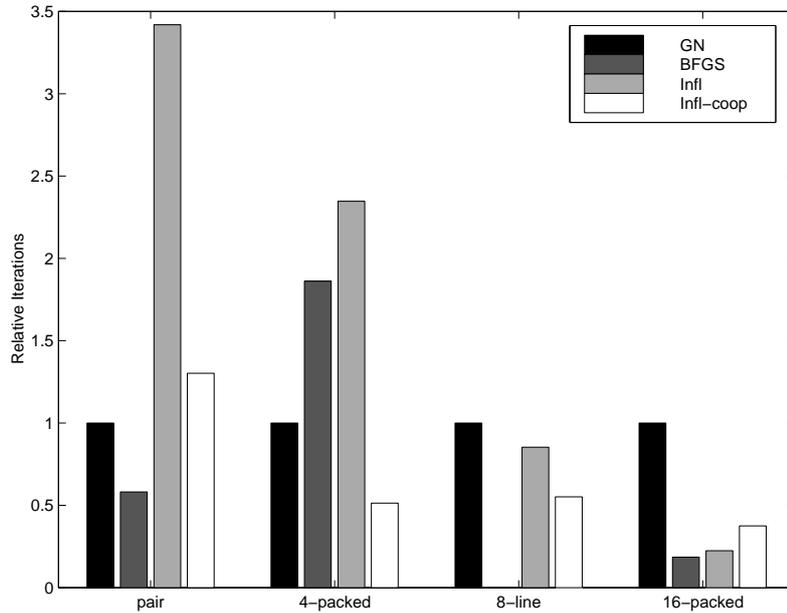


Fig. 30. Performance data indicate that influence graphs support cooperative optimization: Infl-coop uses supervisors for pairs of tightly-coupled sources and generally requires fewer iterations than does the standard optimizer Infl. The centralized Matlab optimizers GN (Gauss-Newton) and BFGS (Broyden-Fletcher-Golfarb-Shanno) are provided for reference.

	GN	BFGS	Infl	Infl-coop
pair				
iterations	1.0	0.5814	3.419	1.302
error	1.0	1.0	1.0	1.0
4-packed				
iterations	1.0	1.861	2.347	0.5139
error	1.0	1.0	1.0	1.0
8-line				
iterations	1.0	n/a	0.8529	0.5515
error	1.0	n/a	1.007	1.003
16-packed				
iterations	1.0	0.1859	0.2244	0.3750
error	1.0	1.004	1.054	1.009

Table 9

Performance data for cooperative optimization: number of iterations and resulting error for different optimization methods for representative problems. GN (Gauss-Newton) and BFGS (Broyden-Fletcher-Golfarb-Shanno) are Matlab-based centralized optimizers. Infl is the standard decentralized optimizer, while Infl-coop uses influence graph-based joint optimization. Values are relative to Gauss-Newton.

Figure 30 illustrates the convergence rate of the different algorithms. Both influence-based optimizers find or come very close to the optimal error, but the use of cooperation generally results in much faster convergence. In the final test case (16 sources tightly packed), the cooperative optimization method takes somewhat longer. This is most likely due to the implementation of only pairwise cooperation — the tight coupling of so many sources might benefit from hierarchical supervision of larger groups of sources.

### 5.5 Discussion

Influence graphs support control parameter design by encoding structural dependencies among control sources and spatial fields. This information allows efficient evaluation of fields in terms of scaled sums of influences. It also supports trading off among computation, communication, and control quality based on amount of influence. While the optimization algorithm was based on a very simple decentralized updating process, its results are competitive with several standard centralized optimization algorithms.

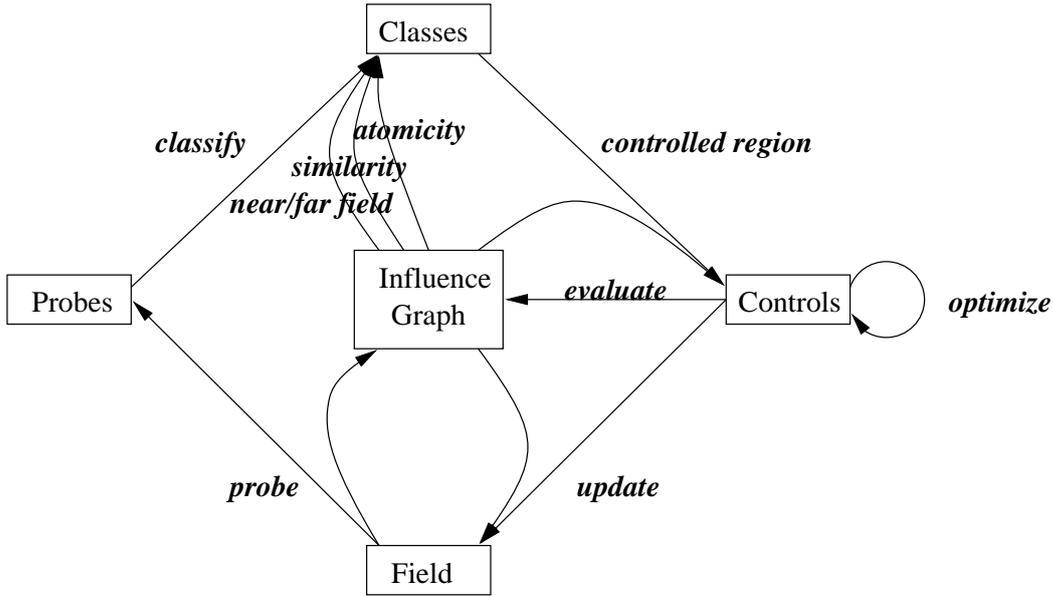


Fig. 31. The influence graph mechanism support decentralized control design with a set of generic data types and operators for partitioning probes and optimizing parameters.

## 6 Conclusion

This paper has presented the influence-based model decomposition to decentralized control design and a case study application. The influence graph mechanism supports decentralized control design utilizing structural descriptions uncovered in physical data. The influence graph-based control design algorithms decompose a problem domain into minimally-coupled subregions, efficiently evaluate fields, and compare parametric design trade-offs. The algorithms are efficient and yield explainable designs.

While we have concentrated on the specific application of decentralized control of a thermal field, many of these techniques generalize to other, similar application areas. The influence graph mechanism developed in this paper provides a generic framework, illustrated in Figure 31, for decentralized control design. Probing a field allows extraction of a structural representation (the influence graph) of the effects of controls on the field. Control probes are classified based on similarity and atomicity measures, and the controlled regions of the equivalence classes yield a decomposition of the original field. Actual controls are placed based on this decomposition. Parametric optimization adjusts the control actions of these controls, based on their effects on the field as encapsulated in an influence graph. This generic framework is applicable to a variety of decentralized control design domains. The remainder of this section discusses the conditions that make this approach successful and potentially appropriate for other design problems.

The control design algorithms rely on the encoding in influence graphs of physical field dependencies. Static influence graphs might not be realistic for some physical processes; for example, when material properties vary with temperature. In such cases, it might be necessary to reason with sets of influence graphs; for example, different influence graphs for different temperature bins.

One key piece of physical knowledge leveraged by the control design algorithms is that of locality. Control placement design strives for decoupling, placing controls so that they interfere as little as possible. This allows control parameter optimization to individually optimize the resulting controls. In addition, control parameter optimization separately considers a control’s effects on strongly-influenced nodes and on weakly-influenced nodes. Certain problems, such as heat transfer with highly conductive materials, may not possess strong locality; such problems are less amenable to these approaches.

Control placement design forms groups of control probes with similar effects on the field. This technique relies on continuity of effects: nearby controls have similar effects, unless there are particular constraints due to geometry and material properties. The goal of the control probes is to uncover these constraints. 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.

Many physical processes (e.g. heat conduction, gravity, electrostatics, and incompressible fluid flow) obey linear superposition of solutions. The influence graph-based optimization process uses this property to evaluate fields efficiently, based on sums of influences. The influence graph encapsulates other possibly nonlinear irregularities in physical fields, exposing linear dependence on control values.

### *6.1 Related Work*

Influence-based model decomposition uses explicit representations of physical knowledge in order to reason about physical systems. Much research in Qualitative Reasoning has also studied how to apply high-level representations of physical systems and domain knowledge in order to predict, diagnose, reconfigure, and tutor [29–32]. Many of the results in that community have centered around three main ontologies: the device ontology [30], which propagates qualitative constraints along topological connections between devices such as the elements of a circuit; the process ontology [29], which generates possible qualitative temporal evolutions of processes such as fluid flow between containers, based on specifications of interaction; and the constraint ontology [31], which

simulates qualitative differential equations describing the evolution of a system. High-level languages such as the Compositional Modeling Language [33] support the specification and compilation of domain knowledge. These ontologies tend to deal only with a system’s topology, abstracting away its rich spatial properties.

Some recent research has extended these approaches to utilize spatial information. Qualitative spatial reasoning systems use abstract descriptions of shape and topology as the basis for inferring behaviors of systems. For example, the Region-Connection Calculus [34] represents topological relations, such as *overlaps* and *is-disconnected-from*, while Rajagopalan’s extremal point representation [35] supports relative orientation and position descriptions. Qualitative physical fields [36] extend Qualitative Process Theory [29] to include qualitative spatio-temporal processes; for example, modeling heat flow between topologically connected sunny and shaded regions and inferring the evolution of warm and cold regions. Recognizing that topology is often not sufficient for complex tasks, the Metric Diagram / Place Vocabulary (MD/PV) theory [37] incorporates problem-specific metric information between special entities (places) in a domain. Similarly, the Spatial Semantic Hierarchy [38,39] discovers “interesting” locations in the construction of mappings between topological and metric maps for robot navigation.

Spatial simulation research in the diagrammatic reasoning community also leverages knowledge of physical systems in order to predict behaviors over time. WHISPER [40] represents objects in a pixel-occupancy array in order to solve problems in a blocks world; similar models have been used to simulate fluid flow from low-level “molecular” interactions [41]. The analogical simulation framework [42] employs a multi-level symbolic/visual representation of a system and has been used to simulate rigid body kinematic behaviors.

Our work differs from most of the above related work in that, in addition to *using* structural descriptions of physical phenomena to reason about systems, we also support *automatic generation* of the structural descriptions of physical phenomena from data or simulations (“predicate extraction” in the taxonomy of Chandrasekaran [43]).

The key idea of decomposing large models of physical systems based on some notion of influence has also been used successfully in areas such as qualitative reasoning, Bayesian nets, image processing, and molecular dynamics. For example, in the area of parameter estimation, Williams and Millar developed a decomposition algorithm that determines for each unknown variable in a model a minimally overdetermined subset of constraints [44]. 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. For qualitative simulation, Clancy

introduced an algorithm that generates an envisionment of a model expressed as a qualitative differential equation, once a partition of the model is given by the modeler [45]. 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 [21,46]. In the probabilistic reasoning community, Friedman et al. have introduced a method to decompose a large Bayesian belief net into weakly-interacting components by examining the dependency structure in the net [47]. Finally, 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 [48].

A large body of engineering literature explores methods for modeling spatially distributed physical systems and elaborating the consequences of these models. Since closed-form analytical solutions are often impossible, engineers typically use techniques such as finite differences and finite elements [19] to represent a system’s governing partial differential equations in terms of matrices on an appropriate discretization. They then apply iterative algorithms [28] to solve the resulting sets of equations. Advanced techniques such as domain decomposition [4] and multigrid methods [5] achieve additional efficiency in convergence or parallelizability of computation.

Our approach differs from these traditional techniques in a number of ways. We provide operators and data types at a level of abstraction appropriate for the tasks, not requiring coercion of a program into a matrix form. Our approach builds only local models and elaborates the consequences of these models through local interaction rules. It makes explicit where and how physical knowledge and domain-specific assumptions are being used, in order to avoid the fragility often associated with numerical methods. In combination with multi-layer descriptions of a system, the explicit use of physical knowledge allows high-level explanation of results.

Much engineering research has also studied the design of decentralized control actions for spatially distributed phenomena. One approach is to simplify (e.g. linearize) the model of a system and apply traditional engineering techniques (e.g. linear-quadratic-gaussian control or Kalman filters) to the design [49]. Another approach is to apply local control methods at the individual controllers and then use hierarchical techniques to exchange information necessary for global control [50]. Market-based methods [51] allow individual controllers to negotiate commodities representing control parameters in order to reach a global solution [52]. Recently, hierarchical constrained optimization has been developed for optimally allocating actuator forces for a planar array of airjets given a global desired trajectory of an object being controlled [53]. The method allocates actuation forces to groups of airjets of various spatial scales,

according to a user specified grouping of actuators into modules, say, using knowledge of the physical layout of the airjet table.

Design techniques for decentralized control placement have also been studied in the engineering community. Different metrics can be used to estimate the quality of a control design. For example, a control design can be evaluated in terms of effectiveness for specific vibration modes [54], required control energy [55], or error with respect to a desired control profile over a family of expected disturbances [56]. The controller placement is then computed by combinatorial optimization of the metric; for example, by greedy search [57,58], genetic algorithms [55,54,59], or simulated annealing [60].

In contrast to these parametric and structural design techniques, we seek to use domain knowledge to automatically extract and exploit high-level structural descriptions of physical phenomena in the design process. This yields principled methods 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 higher-level reasoning about and explanation of the design decisions. For example, the structural description is used to automatically decouple a region into relatively independently controllable subregions.

## 6.2 Future Work

Empirical evidence was provided demonstrating that the structure-based design algorithms perform at least as well as standard approaches, and also support explicit trade-offs between criteria such as communication and control quality. However, no mathematical proofs *guaranteeing* properties of the control design were presented. It would be interesting to see which control theoretic properties of the algorithms, if any, can be stated in analytic forms.

The control placement design algorithms seek to decompose a field in order to place controls that minimally interfere with each other. Other criteria are also important, and could be combined with this approach. For example, if there is exactly one desired temperature profile, its characteristics could be used to steer control placement to locations where most heat is required.

The control parameter design algorithms deal with temperature regulation by varying control output around some set point. The algorithms naturally extend to timing-varying control, where the goal is to track some desired profile over time. By discretizing controls in time as well as in space, the same properties of locality and linearity hold, and the same trade-offs between communication and control quality can be made. However, additional design criteria (e.g. total error over time vs. maximum error at any point) become important.

Furthermore, timing-varying control opens up new avenues of control design; for example, achieving decentralized control with one moving control rather than with a set of stationary controls.

The control design algorithms also explicitly focus on the thermal regulation domain. Since many other systems, including electrostatics, gravity, and incompressible fluid flow, obey the same model, it would be interesting to study application of the techniques developed here to those domains. Extending this work to address wave phenomena (governed by a model different from the diffusion equation) remains as a future research topic.

The control design algorithms only address one side of the picture: placement and optimization of controls. Equally important is the dual problem of sensor placement. In some cases, such as heat control with readily-available data from an infrared camera, the sensor placement problem need not be addressed. In other cases, however, it has a great impact on the control design. For example, controls might have access only to data from nearby sensors, and sensors might not even be available in some parts of the domain. One simple approach to sensor placement is to co-locate sensors with actuators. Then sensor information could be propagated to controls, perhaps using methods similar to the reduced-communication optimization algorithm in order to trade off between communication frequency and accuracy.

A more sophisticated approach to sensor placement requires reasoning about the information available at various sensor locations; that is, what each potential location reveals about the effects of the controls. A mechanism dual to the influence graph could be defined to encode distributed representations of information available at potential sensor locations. This mechanism could then be utilized to place sensors so as to maximize coverage and minimize overlap, just as with the control placement algorithm.

A related consideration is that of the effect of noise on sensors and the resulting control actions. Standard control techniques could be applied to smooth data over time in order to reduce the effects of noise. However, more powerful techniques could use influence graphs and the information graphs proposed above in order to reason about potential error in the data. For example, outlying data points could be identified by building up a model of the influence hill for a control, and noticing when a data point does not conform to the appropriate shape. The influence graph mechanism could also be extended, in a manner similar to the reduced communication optimization algorithm, to reason about the effects of sensor uncertainty. That is, there is a curve trading off error in sensor state and error in control analogous to the curve trading off frequency of sensor state update and error in control.

Enabled by advances in microelectronics and microfabrication, a new gener-

ation of sensor-actuator-rich systems, ranging from smart buildings to self-diagnosing printers to airplanes steered by micro-flaps, calls for a scalable and principled approach to the massively distributed data interpretation and control design problem. There are many challenges in programming such data interpretation and control applications, and existing tools are not sufficient. The Spatial Aggregation Language and influence-based model decomposition represent a step towards the development of powerful programming environments for these tasks: they provides high-level data types and operators in a decentralized framework, they use explicit representations of physical knowledge, and they bridge local and global representations through multiple layers of abstraction. The control design algorithms presented here also exemplify many of the characteristics desirable for such applications: by decomposing and decentralizing they are scalable, by reasoning in terms of spatial structures they provide explainable design decisions, and by utilizing physical knowledge they expose trade-offs among desirable design properties.

## Acknowledgment

This work benefitted substantially from conversations with members of the Intelligent Simulation Group at Ohio State (Xingang Huang, Shiou Loh, Jeff May, and Iván Ordóñez) and a number of researchers at Xerox PARC (Patrick Cheung, Marcus Fromherz, John Gilbert, Warren Jackson, John Lamping, Rachel Lau, and Mark Yim), along with valuable suggestions from the reviewers. This work was supported in part by NSF grant CCR-9457802, ONR grant N00014-97-1-0599, a Sloan Research Fellowship, and DARPA contract F33615-99-C-3611. CBK is currently supported by grants to Bruce Randall Donald, Dartmouth Computer Science Department, from the NSF, Microsoft Research, and the Department of Justice.

## References

- [1] K.-F. Böhringer, B. Donald, N. MacDonald, Programmable vector fields for distributed manipulation, with applications to MEMS actuator arrays and vibratory parts feeders, *International Journal of Robotics Research* .
- [2] A. Berlin, MEMS-based active structural strengthening technology, in: *Proceedings of Government Microcircuit Applications Conference*, 1994.
- [3] K. Forbus, *Diagrammatic Reasoning: Cognitive and Computational Perspectives*, AAAI Press, 1995, Ch. Qualitative spatial reasoning: frameworks and frontiers.

- [4] T. Chan, T. Mathew, Domain Decomposition Algorithms, Vol. 3 of Acta Numerica, Cambridge University Press, 1994, pp. 61–143.
- [5] W. Briggs, A Multigrid Tutorial, Lancaster Press, 1987.
- [6] K. Yip, F. Zhao, Spatial aggregation: theory and applications, Journal of Artificial Intelligence Research 5.
- [7] K. Yip, F. Zhao, E. Sacks, Imagistic reasoning, ACM Computing Surveys 27(3).
- [8] F. Zhao, Intelligent simulation in designing complex dynamical control systems, in: Tzafestas, Verbruggen (Eds.), Artificial intelligence in industrial decision making, control, and automation, Kluwer Academic Publishers, 1995.
- [9] C. Bailey-Kellogg, F. Zhao, Spatial aggregation: modeling and controlling physical fields, in: Proceedings of Qualitative Reasoning Workshop, 1997.
- [10] C. Bailey-Kellogg, F. Zhao, Qualitative analysis of distributed physical systems with applications to control synthesis, in: Proceedings of AAAI, 1998.
- [11] C. Bailey-Kellogg, F. Zhao, Influence-based model decomposition, in: Proceedings of AAAI, 1999.
- [12] Y. Jaluria, K. Torrance, Computational Heat Transfer, Hemisphere Publishing, 1986.
- [13] C. Doulamidis, In-process control in thermal rapid prototyping, IEEE Control Systems .
- [14] T. Kailath, et al., Control for advanced semiconductor device manufacturing: A case history, in: W. Levine (Ed.), The Control Handbook, CRC Press, 1996.
- [15] H. Abelson, et al., Intelligence in scientific computing, CACM 32(5).
- [16] K. Yip, KAM: A system for intelligently guiding numerical experimentation by computer, MIT Press, 1991.
- [17] F. Zhao, Extracting and representing qualitative behaviors of complex systems in phase spaces, Artificial Intelligence 69(1-2) (1994) 51–92.
- [18] L. Joskowicz, E. Sacks, Computational kinematics, Artificial Intelligence 51 (1991) 381–416.
- [19] O. Zienkiewicz, R. Taylor, The Finite Element Method, McGraw-Hill, 1989.
- [20] C. Bailey-Kellogg, F. Zhao, K. Yip, Spatial aggregation: language and applications, in: Proceedings of AAAI, 1996.
- [21] J. Shi, J. Malik, Normalized cuts and image segmentation, in: Proceedings of CVPR, 1997.
- [22] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller, Equation of state calculations by fast computing machines, Journal of Chemical Physics 21 (1953) 1087–1092.

- [23] B. Kernighan, S. Lin, An effective heuristic procedure for partitioning graphs, *The Bell System Technical Journal* (1970) 291–308.
- [24] H. Simon, Partitioning of unstructured problems for parallel processing, *Computing Systems in Engineering* 2 (1991) 135–148.
- [25] G. Miller, S.-H. Teng, W. Thurston, S. Vavasis, Automatic mesh partitioning, in: A. George, J. Gilbert, J. Liu (Eds.), *Sparse matrix computations: graph theory issues and algorithms*, Springer-Verlag, 1993.
- [26] B. Hendrickson, R. Leland, A multilevel algorithm for partitioning graphs, Tech. Rep. SAND93-2339, Sandia National Laboratories (1993).
- [27] J. Holland, *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, 1975.
- [28] W. Press, B. Flannery, S. Teukolsky, W. Vetterling, *Numerical Recipes: the Art of Scientific Computing*, Cambridge University Press, 1986.
- [29] K. Forbus, Qualitative process theory, *Artificial Intelligence* 24.
- [30] J. de Kleer, J. Brown, A qualitative physics based on confluences, *Artificial Intelligence* 24.
- [31] B. Kuipers, Qualitative simulation, *Artificial Intelligence* 29.
- [32] D. Weld, J. de Kleer (Eds.), *Readings in Qualitative Reasoning about Physical Systems*, Morgan Kaufmann, 1990.
- [33] B. Falkenheimer, K. Forbus, Compositional modeling: Finding the right model for the job, *Artificial Intelligence* 51 (1991) 95–143.
- [34] Z. Cui, A. Cohn, D. Randell, Qualitative simulation based on a logical formalism of space and time, in: *Proceedings of AAAI*, 1992.
- [35] R. Rajagopalan, A model for integrated qualitative spatial and dynamical reasoning about physical systems, in: *Proceedings of AAAI*, 1994.
- [36] M. Lundell, A qualitative model of physical fields, in: *Proceedings of AAAI*, 1996.
- [37] K. Forbus, P. Nielsen, B. Faltings, Qualitative spatial reasoning: the CLOCK project, *Artificial Intelligence* 51.
- [38] B. Kuipers, T. Levitt, Navigation and mapping in large-scale space, *AI Magazine* 9(2).
- [39] B. Kuipers, A hierarchy of qualitative representations for space, in: *Proceedings of 10th International Workshop on Qualitative Reasoning*, 1996.
- [40] B. Funt, Problem solving with diagrammatic representations, *Artificial Intelligence* 13.
- [41] F. Gardin, B. Meltzer, Analogical representations of naive physics, *Artificial Intelligence* 38.

- [42] B. Chandrasekaran, N. Narayanan, Towards a theory of commonsense visual reasoning, in: K. Nori, C. Madhavan (Eds.), *Foundations of Software Technology and Theoretical Computer Science*, Springer, 1990.
- [43] B. Chandrasekaran, Diagrammatic representation and reasoning: some distinctions, in: *AAAI Fall Symposium Series: Diagrammatic Reasoning*, 1997.
- [44] B. Williams, B. Millar, Automated decomposition of model-based learning problems, in: *Proceedings of 10th International Workshop on Qualitative Reasoning*, 1996.
- [45] D. Clancy, Model decomposition and simulation: a component based qualitative simulation algorithm, in: *Proceedings of AAAI*, 1997.
- [46] P. Felzenszwalb, D. Huttenlocher, Image segmentation using local variation, in: *Proceedings of CVPR*, 1998.
- [47] N. Friedman, D. Koller, A. Pfeffer, Structured representation of complex stochastic systems, in: *Proceedings of AAAI*, 1998.
- [48] F. Zhao, An  $O(N)$  algorithm for three-dimensional n-body simulations, Tech. Rep. AI-TR-995, MIT AI Lab (1987).
- [49] N. Sandell Jr., P. Varaiya, M. Athans, M. Safonov, Survey of decentralized control methods for large scale systems, *IEEE Trans. on Automatica Control* 23(2).
- [50] J. How, S. Hall, Local control design methodologies for a hierarchic control architecture, in: *Proceedings of AIAA Guidance, Navigation, and Control Conference*, 1992.
- [51] M. Wellman, A market-oriented programming environment and its applications to distributed multicommodity flow problems, *Journal of Artificial Intelligence Research* .
- [52] O. Guenther, T. Hogg, B. Huberman, Market organizations for controlling smart matter, in: *Proceedings of the International Conference on Computer Simulation and Social Sciences*, 1997.
- [53] W. Jackson, M. Fromherz, A. Berlin, D. Biegelsen, P. Cheung, Hybrid problems in smart matter control, in: *Proceedings of AAAI Spring Symposium on Hybrid Systems and AI*, 1999.
- [54] H. Furuya, R. Haftka, Combining genetic and deterministic algorithms for locating actuators on space structures, in: *Proceedings of AIAA 36th Structures, Structural Dynamics, and Materials Conference and Adaptive Structures Forum*, 1995.
- [55] A. Dhingra, B. Lee, Optimal placement of actuators in actively controlled structures, *Engineering Optimization* 23 (1994) 99–118.
- [56] S. Hakim, M. Fuchs, Optimal actuator placement with minimum worst case distortion criterion, in: *Proceedings of AIAA 36th Structures, Structural Dynamics, and Materials Conference and Adaptive Structures Forum*, 1995.

- [57] R. Skelton, M. DeLorenzo, Selection of noisy actuators and sensors in linear stochastic systems, *Journal of Large Scale Systems, Theory and Applications* 4 (1981) 109–136.
- [58] R. Haftka, H. Adelman, Selection of actuator locations for static shape control of large space structures by heuristic integer programming, *Computers and Structures* 20.
- [59] X. Liu, D. Begg, R. Fishwick, Genetic approach to optimal topology / controller design of adaptive structures, *International Journal for Numerical Methods in Engineering* 41.
- [60] G.-S. Chen, R. Bruno, M. Salama, Optimal placement of active / passive members in truss structure using simulated annealing, *AIAA Journal* 29.