Reasoning About and Optimizing Distributed Parameter Physical Systems Using Influence Graphs

Christopher Bailey-Kellogg **Feng Zhao**
Xerox Palo Alto Research Center
3333 Coyote Hill Road, Palo Alto, CA 94304 U.S.A.
{kellogg,zhao}@parc.xerox.com

Abstract

We develop the *influence graph* mechanism for reasoning about and optimizing decentralized controls for distributed parameter physical systems. Distributed parameter systems, such as air flow around an airplane wing, temperature over a semiconductor wafer, and noise from a photocopy machine, are common physical phenomena. The influence graph mechanism encodes the structural dependency information in a distributed parameter system and exploits the information to (1) alleviate redundant computation and (2) reduce communication and support cooperation among local control processes. Using the mechanism, we obtained a dramatic computational speed-up in optimizing control design for a distributed temperature field.

Introduction

Reasoning about and optimizing spatially distributed physical fields such as fluid flow, temperature, and acoustic waves are challenging for a number of reasons. First, a distributed parameter field is conceptually harder to reason about and model than a lumped parameter system such as a circuit. Spatial topology, metric, material properties, and physical laws all come into play, in addition to the combinatorial structures. Second, numerical methods for optimizing distributed systems are prohibitively expensive for large, irregular geometric domains and highly non-uniform phenomena. When a physical realization of control mechanisms employs a set of distributed sensors and actuators, it is necessary that the reasoning and optimization process be implemented in a *decentralized* way to ensure adaptivity and robustness. The practical impact of such mechanisms is enormous. For instance, the drag on an airplane can be reduced by analyzing and controlling the air flow around the wings. Temperature in a "smart" building can be regulated to maximize occupant comfort while minimizing energy consumption (Berlin 1994; Williams & Nayak 1996).

In this paper, we develop the influence graph mechanism for synthesizing distributed control schemes for physical fields. The mechanism finds a feasible control design for a physical field, specifying control actions that meet given design objectives. It extends the spatial aggregation (SA) framework (Bailey-Kellogg, Zhao, & Yip 1996; Yip & Zhao 1996) by explicitly encoding and managing dependencies among spatial objects. This work has significantly extended the previously developed mechanisms for control structure design (Bailey-Kellogg & Zhao 1997) to address distributed optimization of controllers.

The SA influence graph mechanism differs from existing numerical design and optimization methods in several important ways. Our objective is to construct a qualitative physics model for physical fields so that behaviors of the fields can be inferred using a small number of operations on a discrete representation and explained in terms of object interaction and evolution. In particular, the influence graph serves as a means for calculating, explaining, and exploiting dependencies in physical fields. The influence graph makes explicit how physical knowledge, such as locality and linear superposability of control, can be used to improve design techniques. Finally, SA encourages a decentralized mindset, manipulating fields through local interaction rules rather than through global models.

The remainder of the paper proceeds as follows. First we introduce the problem of control optimization for distributed parameter systems. We then develop the influence graph as a mechanistic device to encapsulate dependencies in physical fields. We present algorithms that use influence graphs to help manage the computational complexity in control optimization. We provide experimental evidence to demonstrate the effectiveness of the mechanism. Finally, we discuss related work.

The Problem: Control Optimization

As an example of control optimization for a distributed parameter system, consider the temperature regulation problem for a piece of material (Jaluria & Torrance 1986), as shown in Figure 1[1]. As part of the manufacturing process, the temperature distribution over the

[1]A similar problem arises from temperature control in semiconductor manufacturing, in which the oven temperature over a surface of semiconductor wafer must be regulated by a set of spatially distributed heating lamps to ensure high yield. This is a challenging control problem in rapid thermal processing (RTP) of semiconductor wafers because temper-

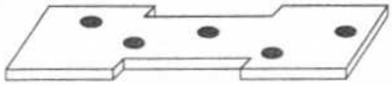
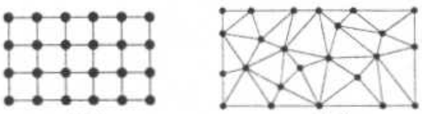


Figure 1: Industrial heat treatment of a metal sheet. The control objective is to achieve a specified temperature profile over the material by applying heat at a small number of locations, shown as dark circles.



(a) (b)

Figure 2: A field modeled by a network of local spatial objects: (a) A regular finite difference grid of objects; (b) A finite element mesh of triangle objects.

sheet must be regulated at some desired profile to minimize damage to the material. We consider two types of control problem: the *steady-state* problem, where a particular temperature distribution must be maintained over an extended period of time, and the *transient* problem, in which a desired temperature profile must be tracked over time. Control designs depend on the thermal process in the metal sheet, which in turn depends on physical laws, geometry, material properties, and boundary conditions, in addition to control actions.

Spatial Aggregation (SA) provides a field ontology for representing distributed parameter physical fields. For example, the temperature field over the sheet of metal is described by a network of spatial objects encoding geometric location and temperature information (Figure 2). A neighborhood graph (N-graph) encodes spatial adjacencies among the objects. For temporal problems, objects are also indexed by time. Local constraints such as those derived from approximations to field derivatives or laws of conservation govern the evolution of the objects. The field values are determined by a local relaxation method that iteratively updates spatial objects using the local constraints. SA also provides operators for constructing and transforming these objects across space and over time. The SA framework serves as a computational substrate upon which the influence graph mechanism has been developed and implemented.

Given a description of the field to be controlled, the task is to design a control strategy that effectively steers the physical process to meet the desired criteria. To determine the parameters of the distributed control, one needs to search the large design space subject to structural and performance constraints:

- Structural constraints: geometry, physical properties, boundary conditions.
- Performance constraints: desired profile, optimality conditions on solutions, restriction on control sources (e.g. maximum heat output).

ature non-uniformity often leads to chip defects (Kailath & others 1996).

(a) (b) (c)

Figure 3: Steady-state thermal hills around sources. The vertical axis represents temperature value. (a) A single source. (b) Two fairly independent sources. (c) Two tightly coupled sources.

In the thermal control problem, the control objective is to establish a particular temperature distribution over the entire field, using a small set of discrete heat sources, subject to constraints on the maximum source output and acceptable temperature fluctuations. This global control objective can be formulated locally by constraining each thermal object to have a temperature within some error tolerance of its desired temperature. The available control authority consists of point sources. For the transient heat control problem, source values are discretized over time, so that each source at a time instant is separately optimized. If desired, additional constraints can relate the source value at one time to the value at the next time.

Previous work (Bailey-Kellogg & Zhao 1997) has addressed the design of control structure for a distributed parameter problem; i.e. where to place a set of control sources. This paper addresses the design of control parameters; i.e. the rate of heat output from the sources. In particular, the optimization task is *distributed* among the sources, so that each source attempts to regulate temperature in a local neighborhood and sources cooperate to seek a global optimum. This style of decentralized optimization is necessary to support the applications discussed in the introduction, such as "smart" buildings, where vast networks of sensors and actuators interact with a spatially distributed physical environment.

Influence Graph

A heat source influences the temperature distribution in a field through heat propagation. Figure 3(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, jointly affecting the temperature distribution. The interaction necessitates sharing of information among sources during control parameter optimization, depending on the coupling strength (Figures 3(b) and Figure 3(c)).

We introduce the *influence graph* to record the dependencies between control sources and spatial objects in

Figure 4: Iso-contours for an influence hill.

the field. For a design problem with field nodes F and source nodes S, an influence graph is a triple (V, E, w) with vertices $V = F \cup S$, edges $E = S \times F$, and edge weights $w : E \rightarrow \mathcal{R}$ such that $w((s, f))$ is the field value at f given a unit source at s. Hence, the graph edges record a normalized influence from each source to each field node; Figure 3(a) can be considered a pictorial representation of the edge weights for an influence graph from one source. An influence graph is constructed by placing a unit source at each control location one at a time and evaluating the field values at all the locations of interest, using an iterative relaxation method. The graph needs to be constructed only once for a given control placement and is used repeatedly for later parametric optimization processes.

Why is an influence graph useful? In many distributed physical phenomena, despite nonlinearities in the spatial variables such as non-uniform conduction characteristics and irregular geometries, the field is linearly dependent upon control sources and boundary conditions[2]. The effects of sources can be combined through a superposition of influence hills. Influence graphs encode the crucial dependency information, while hiding other possibly nonlinear effects.

An influence graph is also useful because it explicates the *locality* of source effects on a physical field. Thermal hills decay away from sources; correspondingly, influence graph edges have weaker links for further away field nodes. Spatial aggregation extracts structures such as "iso-influences" that group field objects into equivalence classes based on roughly equal influence from a given source. Figure 4 shows iso-contours for the thermal hill of Figure 3(a); the iso-influence regions are the contiguous field nodes between the iso-contours. The applicability of the assumptions that a problem possesses locality and is linear in control is discussed in the Discussion section.

Influence graphs also describe dependencies in the transient heat problem. Transient heat sources have thermal hills, as shown in Figure 5. Here temperature values for spatiotemporal thermal points depend on source values at locations and time instants. Once again, the temperature values depend linearly on influences from sources. The locality of source effects applies in both space and time.

[2] Boundary conditions can be treated the same as sources.

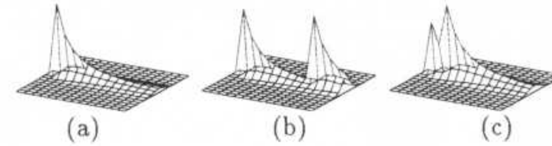


Figure 5: Transient thermal hills around sources. The vertical axis represents temperature value; time flows to the right; a constant slice through the source location has been taken in one spatial dimension. (a) A single source. (b) Two fairly independent sources. (c) Two tightly coupled sources.

Optimization Using Influence Graph

We will use the problem of thermal field regulation to develop the influence graph mechanism. To synthesize a control strategy for the distributed thermal field, the design process requires simultaneous optimization of many parameters (the source values). While algorithms for multi-parameter optimization exist (Press *et al.* 1986), they are computationally expensive for large problems and difficult to parallelize for distributed applications.

We will show how structural knowledge, in the form of the influence graph, significantly improves the performance of a basic decentralized optimization algorithm. The basic optimization algorithm repeatedly adjust each source's value in the direction that minimizes error[3]. Remember that the optimization processes are decentralized — in this basic algorithm, each source adjusts itself independently, taking a step towards what it thinks minimizes error. Figure 6 shows the data flow in this algorithm: each source node tells each field node a possible heat output, receives from the field node the resulting error at that node, and updates its heat output to minimize error. 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.

Efficient Field Evaluation

During each step of an iterative optimization process, the field is evaluated using the relatively expensive, iterative relaxation method on the spatial objects. 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 spatial object can be calculated by summing together the weights of influence graph edges coming into the node, scaled by the control source values. This computation is extremely fast and results in a drastic speed-up in computation. The data flow for the modified optimization algorithm

[3] Control node positions can also be optimized in this manner.

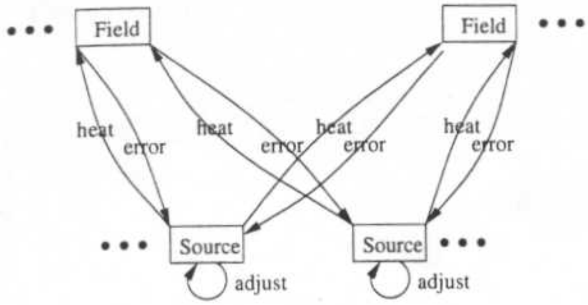


Figure 6: Data flow in the basic decentralized optimization algorithm: sources adjust themselves based on the error in the field resulting from different heat outputs.

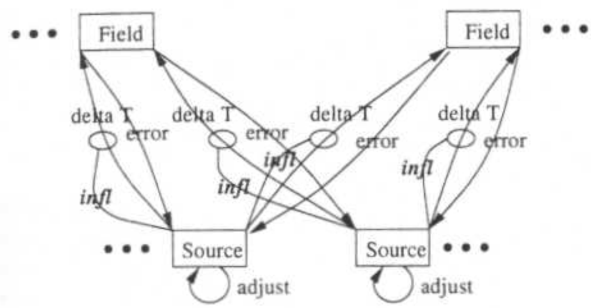


Figure 7: Data flow using efficient field evaluation: sources inform field nodes of temperature updates based on influence graph information.

is illustrated in Figure 7. Now 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. 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.

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. by the structure design algorithm in (Bailey-Kellogg & Zhao 1997)), but that ignores the influence on the other regions. Better yet, a source can communicate with those field nodes it most strongly affects more frequently. If a source only weakly affects a temperature node, we need not assign it much blame/credit for the error at that node. As shown in

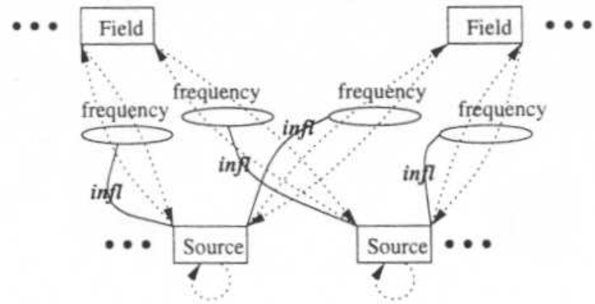


Figure 8: Data flow for reduced communication: frequency of source-field communication is modulated by influence strength.

Figure 8, the frequency of source-field communication is a function of the amount of influence. Decreasing frequency decreases overall communication costs, but increases the potential for error due to underestimated source effects.

We have developed several strategies for establishing source node to field node communication. The most basic method computes communication frequency as a function of the weight along the influence graph edge. 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.

Joint Optimization

While we want 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 3(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 occurs when independently increasing the value of one source increases the total error 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[4]. 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.

The following steps incorporate joint source optimization into our decentralized framework:

1. Establish a neighborhood graph linking a source to other sources with which it will cooperate. This neighborhood graph can be based on spatial proximity. Better yet, it can utilize information encoded in the influence graph: if there is a region strongly influenced by multiple sources, then those sources are coupled and can cooperate to optimize that region.
2. Form new sources ("supervisors") that seek to optimize the distribution of heat between a pair of neighboring sources. The actions available to a supervisor source are to shift heat from one of the sources to the other. Influence graph edge weights for the supervisor are set from the influence difference for the sources it supervises.
3. Add the supervisors to the set of sources being optimized by the decentralized optimization algorithm.

Figure 9 summarizes the data flow for this algorithm: if two sources are tightly coupled, based on influence information, then a supervisor is formed for them; the supervisor is optimized similarly to other sources, but its actions affect the supervised sources. 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 group supervisor sources that shift heat among groups of sources rather than between pairs. However, our experience has not found that extension necessary. Note that supervisors only need to be established between pairs 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 thoroughly testing.

Performance

The 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. Using the influence graph to speed up the field evaluation, 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, we further reduce the number of iterations k.

As expected, the influence graph mechanism results in enormous savings during repeated decentralized field evaluations. For example, in an implementation using the C++-based SA library on a 100-MHz Pentium system with Linux and gcc, 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. Since the field evaluation must be performed at each iteration, the savings add up quickly.

Influence graphs significantly reduce communication during source optimization. Table 1 summarizes results for steady-state parametric design on a regular 20x20 discretized thermal field (transient design has similar performance). The first two optimizers (Gauss-Newton and Broyden-Fletcher-Golfarb-Shanno) are Matlab-based implementations of two standard multi-parameter optimization algorithms (Simplex search optimization was omitted because it failed to converge within 300 steps on all of these tests.) Note that the Matlab algorithms are not decentralized. The SA-based optimizers use an implementation with default parameters and varying amounts of communication: SA1 updates each field object based on each source every iteration, while SA2–SA4 update field objects with frequency proportional to influence, with different constants of proportionality. The optimizers were run on three different tests: four sources near the corners of the grid, four sources near the center of the grid, and sixteen sources tiled over the grid. Three performance results are shown for each test: the number of

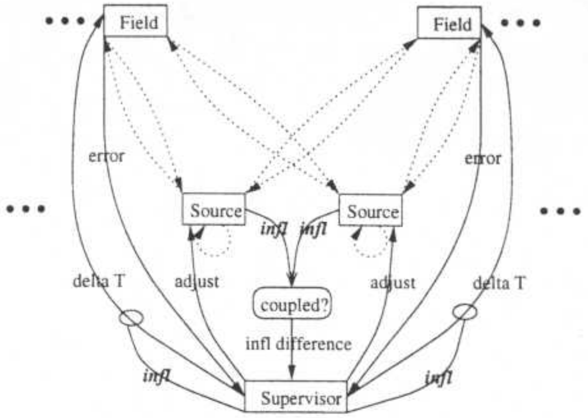


Figure 9: Data flow for joint optimization: supervisor nodes are formed for tightly-coupled sources; optimization of the supervisors shifts heat from one source to another based on errors in the field.

[4] Generally, if jointly adjusting two tightly-coupled sources in the same direction decreases the error, then so does independently adjusting one or the other.

	GN	BFGS	SA1	SA2	SA3	SA4
4-corner						
# iter	19	14	21	19	17	19
	1.0	.7368	1.105	1.0	.8947	1.0
# comm	24624	18144	27216	10572	2332	1144
	1.0	.7368	1.105	.4293	.0947	.0465
error	.2028	.2028	.2028	.2037	.2281	.252
	1.0	1.0	1.0	1.004	1.125	1.243
4-center						
# iter	20	14	24	21	19	31
	1.0	.7	1.2	1.05	.95	1.55
# comm	25920	18144	31104	21004	6980	5188
	1.0	.7	1.2	.8103	.2693	.2002
error	.3459	.3459	.3459	.3463	.3621	.3922
	1.0	1.0	1.0	1.001	1.047	1.134
16-tiled						
# iter	56	213	36	49	46	75
	1.0	3.804	.6429	.875	.8214	1.339
# comm	290304	1104192	186624	161790	50535	39061
	1.0	3.804	.6429	.5573	.2051	.1346
error	.1164	.1164	.1167	.1181	.1189	.1347
	1.0	1.0	1.003	1.015	1.022	1.157

Table 1: Performance data for communication reduction in optimization: number of iterations, number of communications, and resulting error for different optimization methods for representative problems

iterations for convergence, the total source-field node communication[5], and the average squared error across the thermal field. Actual run-time is roughly proportional to the number of communications. Below the raw performance numbers are performance relative to Gauss-Newton (lower is better).

These results show that on representative multi-parameter optimization problems, the SA 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 10 charts the trade-off between communication and error in the four SA optimizers on these problems. Naturally, error increases as communication decreases, but there is quite a long flat area where the communication decreases without serious impact on the error. In problems with larger domains, there will be even fewer field nodes strongly influenced by a source (depending of course on geometry and material properties), providing even greater potential savings.

Finally, influence graphs also support cooperative source optimization. Table 2 provides data for three representative problems with tight coupling among sources: four steady-state sources tightly packed near the edge of a 20x20 grid, sixteen steady-state sources tightly packed near the center of a 20x20 grid, and one transient source controlling 15 time steps in a 5x5 grid. The results from the two (centralized) Matlab optimizers are provided for reference; the first SA optimizer does not cooperatively optimize, while the second one places a supervisor between each neighboring pair of sources. Both SA optimizers find or come very close to the optimum, but the use of cooperation results in much faster convergence. Figure 11 illustrates this point.

[5] In the Matlab-based routines, essentially each source communicates with each non-boundary field node each iteration.

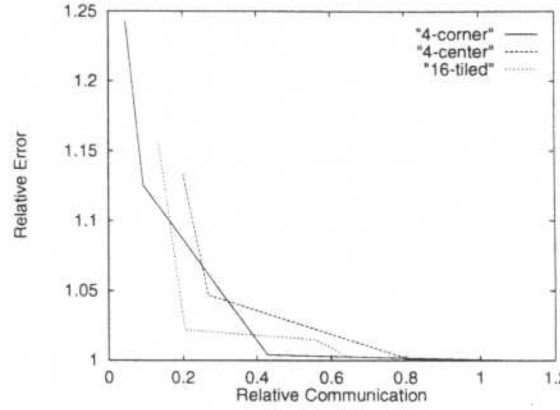


Figure 10: Influence graphs support trading optimization quality for amount of communication. In the flat area, amount of communication is greatly reduced with little impact on error.

	GN	BFGS	SA	SA-coop
4-packed				
# iter	19	96	78	28
error	.6725	.6725	.6737	.6725
16-packed				
# iter	56	207	167	70
error	.3357	.3357	.3371	.3357
15-time				
# iter	94	226	64	25
error	.1189	.1190	.1189	.1189

Table 2: Performance data for cooperative optimization: number of iterations and resulting error for different optimization methods for representative problems.

Discussion

The influence graph explicitly encodes the structural dependencies among control sources and spatial fields. The graph mechanism allows us to explore the design tradeoffs among computation, communication, and control quality in a principled way. Additionally, the causal information encoded in the graph can be used to automatically generate explanations for why higher-level control decisions are made.

The influence graph based design techniques rely on two important pieces of physical knowledge: locality and linear superposability of control. Locality makes it possible to decouple a field and separately consider a source's effects on strongly-influenced nodes and on weakly-influenced nodes. We note that certain problems, such as heat transfer with highly conductive materials, may not possess strong locality; such problems are less amenable to a decompositional approach. Many physical processes (e.g. heat conduction, gravity, electrostatics, and incompressible fluid flow) are linear in control. This fact makes it possible, for example, to calculate temperature as a sum of influences from different heat sources. Influence graphs encapsulate other possi-

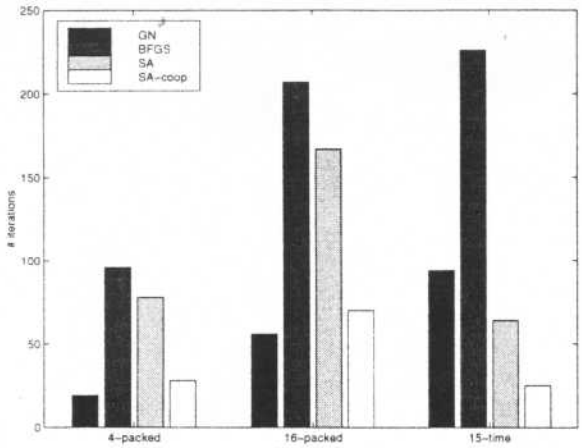


Figure 11: Influence graphs support cooperative optimization: SA-coop uses supervisors for pairs of tightly-coupled sources and requires far fewer iterations than does the standard SA optimizer. The centralized Matlab optimizers GN and BFGS are provided for reference.

bly nonlinear irregularities in physical fields, exposing linear dependence on source values.

While traditional qualitative physics ontologies focus on lumped parameter models of physical systems (DeKleer & Brown 1984; Forbus 1984; Kuipers 1986), the spatial aggregation based influence graph mechanism aims at establishing an ontology for reasoning about distributed parameter physical phenomena that accounts for spatial and physical properties. More recently, Forbus et al. (Forbus, Nielsen, & Faltings 1991) and Lundell (Lundell 1996) have developed representational frameworks for distributed parameter physical fields. SA encodes both qualitative structures and quantitative dependencies of physical fields and can be regarded as a more refined mechanism for reasoning about spatial phenomena.

Unlike control design for lumped parameter linear systems, few analytic design techniques have been developed for distributed control of large physical fields (Sandell Jr. *et al.* 1978). In practice, the design is often accomplished by brute-force numerical computation. Among the recent experimental work in decentralized control, Doumanidis (Doumanidis 1997) addressed the problem of control parameter optimization for distributed parameter systems. He introduced a first-order approximation to local effects of heat sources during the optimization. However, his approach ignored effects of geometries and the use of structural knowledge in guiding decentralized optimization.

The influence graph design algorithms essentially decompose optimization into decentralized optimizers and then account for interaction among the distributed processes. Many other interesting papers have studied problem decomposition and interaction among subproblems. For instance, the parti-game algorithm decomposes high-dimensional state-spaces for learning control strategies (Moore & Atkeson 1995). Similarly, Bradley and Zhao presented several methods for synthesizing nonlinear control laws in phase spaces (Bradley & Zhao 1993); their methods partition phase spaces into manageable subspaces. Bertsekas uses adaptive aggregation in dynamic programming to group states and their dependencies into meta-states according to residual errors (Bertsekas & Castanon 1989). Multigrid (Briggs 1987) and domain decomposition (Chan & Mathew 1994) algorithms support efficient calculation of field values by exploiting the structure of the global matrix for a system. Williams and Millar (Williams & Millar 1996) and Clancy and Kuipers (Clancy & Kuipers 1997) present two different methods for decomposing large models and reasoning about the interactions of sub-components. Our work differs from these in that it uses structural knowledge of distributed parameter fields in the form of influence graphs to guide decomposition and cooperation among decentralized optimization processes.

Conclusions

This paper has developed the powerful influence graph for reasoning about and optimizing controls for distributed parameter physical systems. The influence graph advances the state-of-the-art in qualitative physics and spatial reasoning by identifying and encoding structural dependency knowledge in physical fields. The mechanism has been used in a decentralized optimizer to avoid redundant computation, reduce communication needs, and cooperate among local control processes. It has demonstrated a significant computational advantage on a problem of decentralized control optimization for a thermal field.

Acknowledgments

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

References

Bailey-Kellogg, C., and Zhao, F. 1997. Spatial aggregation: modeling and controlling physical fields. In *Proc. 11th International Workshop on Qualitative Reasoning*.

Bailey-Kellogg, C.; Zhao, F.; and Yip, K. 1996. Spatial aggregation: language and applications. In *Proceedings of AAAI*.

Berlin, A. 1994. MEMS-based active structural strengthening technology. In *Proceedings of Government Microcircuit Applications Conference*.

Bertsekas, D., and Castanon, D. 1989. Adaptive aggregation methods for infinite horizon dynamic programming. *IEEE Trans. on Automatic Control* 34.

Bradley, E., and Zhao, F. 1993. Phase-space control system design. *IEEE Control Systems* 13(2):39-47.

Briggs, W. 1987. *A Multigrid Tutorial.* Lancaster Press.

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

Clancy, D. J., and Kuipers, B. 1997. Model decomposition and simulation: A component based qualitative simulation algorithm. In *Proceedings of AAAI.*

DeKleer, J., and Brown, J. 1984. A qualitative physics based on confluences. *Artificial Intelligence* 24.

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

Forbus, K.; Nielsen, P.; and Faltings, B. 1991. Qualitative spatial reasoning: the CLOCK project. *Artificial Intelligence* 51.

Forbus, K. 1984. Qualitative process theory. *Artificial Intelligence* 24.

Jaluria, Y., and Torrance, K. 1986. *Computational Heat Transfer.* Hemisphere Publishing.

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

Kuipers, B. 1986. Qualitative simulation. *Artificial Intelligence* 29.

Lundell, M. 1996. A qualitative model of physical fields. In *Proceedings of AAAI.*

Moore, A., and Atkeson, C. 1995. The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces. *Machine Learning* 21.

Press, W. H.; Flannery, B. P.; Teukolsky, S. A.; and Vetterling, W. T. 1986. *Numerical Recipes: the Art of Scientific Computing.* Cambridge University Press.

Sandell Jr., N.; Varaiya, P.; Athans, M.; and Safonov, M. 1978. Survey of decentralized control methods for large scale systems. *IEEE Trans. on Automatica Control* 23(2).

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

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

Yip, K. M., and Zhao, F. 1996. Spatial aggregation: theory and applications. *J. Artificial Intelligence Research* 5.