Processor Speed Control With Thermal Constraints

Almir Mutapcic, Stephen Boyd, *Fellow, IEEE*, Srinivasan Murali, David Atienza, Giovanni De Micheli, *Fellow, IEEE*, and Rajesh Gupta, *Fellow, IEEE*

***Abstract*—We consider the problem of adjusting speeds of multiple computer processors, sharing the same thermal environment, such as a chip or multichip package. We assume that the speed of each processor (and associated variables such as power supply voltage) can be controlled, and we model the dissipated power of a processor as a positive and strictly increasing convex function of the speed. We show that the problem of processor speed control subject to thermal constraints for the environment is a convex optimization problem. We present an efficient infeasible-start primal-dual interior-point method for solving the problem. We also present a distributed method, using dual decomposition. Both of these approaches can be interpreted as nonlinear static control laws, which adjust the processor speeds based on the measured temperatures in the system. We give numerical examples to illustrate performance of the algorithms.**

***Index Terms*—Convex optimization, distributed control, primal-dual interior-point methods, temperature-aware processor control.**

I. Introduction

We consider a multiprocessor system, in which many processors share a common thermal environment, e.g., many processor cores on a single chip, or processors on separate chips in a multichip package. We assume that the speed of each processor (along with associated variables such as power supply voltage) can be varied over a range. The speed of each processor affects its power dissipation, which in turn affects the overall temperature distribution of the system. The goal is to adjust the speeds (and associated variables) and to obtain the maximum total processing capability while respecting limits on the temperature at various points in the system. A variation on this problem is to choose the processor speeds to minimize the maximum temperature in the system while meeting a given required total processing capability.

In this paper, we show that the problem of static processor speed control with thermal constraints can be posed as a convex optimization problem. We then give two methods to solve the problem. The first method is an efficient primal-dual interior-point method, which is extremely fast and can solve a problem instance with a hundred processors in tens of milliseconds and can scale to much larger problems. This method can be warm-started to track the temperature changes due to other thermal sources beyond our control and therefore can be considered as a complex nonlinear control law. The second method is based on solving a dual problem of the (primal) processor speed problem. The benefit of this approach is that it gives a distributed method, where each processor adjusts its speed based only on temperature measurements at nearby points. This method too can be interpreted as a nonlinear feedback control law, which is in addition distributed.

Processor speed control with power and thermal constraints has been a topic of extensive research in the last few years, see, e.g., the surveys [1]–[3] and the references therein. Here, we briefly discuss some of the results. Several authors have applied formal control techniques to derive various feedback control laws for the processor speeds [4]–[9]. In particular, Donald and Martonosi propose an effective proportional-integral (PI) controller for thermal management in [3]. Skadron *et al*. have implemented a thermal modeling tool for electronic devices, called HotSpot; using it they provide and simulate a temperature-aware processor system in [10]. (For more details and references about thermal modeling, see the Appendix.) A more general approach of optimization with thermal constraints described as partial differential equations is given in [11]. Other approaches for temperature control of systems and devices can be implemented using fuzzy controllers [12] or thermal compensation circuits [13], [14]. Pruhs and coauthors formulate the processor speed control problems with power, thermal, and task precedence constraints as scheduling optimization problems and present heuristic algorithms to solve them [15], [16]. Recently, the authors in [17] use a variational approach to derive an analytical optimal solution for a single processor speed control with thermal constraints. In [18], energy aware task scheduling in real-time systems is posed as a convex optimization problem and solved using the ellipsoid method. For some work on using convex optimization for multiprocessor frequency assignment, and some experimental results, see [19] and [20].

The main contributions of the present paper are as follows. We describe the first highly efficient interior-point algorithm, and the first distributed algorithm (with convergence proof), for

Manuscript received September 10, 2008. First published December 22, 2008; current version published September 04, 2009. This work was supported in part by Focus Center Research Program Center for Circuit and System Solutions Award 2003-CT-888, in part by Jet Propulsion Laboratory Award I291856, in part by Precourt Institute on Energy Efficiency, in part by Army Award W911NF-07-1-0029, in part by National Science Foundation Award ECS-0423905 and Award 0529426, in part by Defense Advanced Research Projects Agency Award N66001-06-C-2021, in part by National Aeronautics and Space Administration under Award NNX07AEIIA, and in part by Air Force Office of Scientific Research Award FA9550-06-1-0514 and Award FA9550-06-1-0312. This paper was recommended by Associate Editor N. K. Jha.

A. Mutapcic and S. Boyd are with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305 USA (e-mail: almirm@stanford.edu; boyd@stanford.edu).

S. Murali and G. De Micheli are with the Integrated Systems Laboratory (LSI), Ecole Polytechnique Federale de Lausanne (EPFL), 1015 Lausanne, Switzerland (e-mail: srinivasan.murali@epfl.ch; giovanni.demicheli@epfl.ch).

D. Atienza is with the Embedded Systems Laboratory (ESL), Ecole Polytechnique Federale de Lausanne (EPFL), 1015 Lausanne, Switzerland,, and also with the Department of Computer Architecture and Automation (DACYA), Complutense, University of Madrid (UCM), 28015 Madrid, Spain (e-mail: david.atienza@epfl.ch).

R. Gupta is with the Department of Computer Science and Engineering, University of California, San Diego, CA 92093 USA (e-mail: rgupta@ucsd.edu).

Digital Object Identifier 10.1109/TCSI.2008.2011589

solving the problem of processor speed control with thermal constraints. In this paper, we emphasize algorithm and system aspects of the problem; we do not devote much attention to physical and circuit-level modeling. In particular, the numerical examples given in this paper are chosen for simplicity and clarity and are primarily meant to illustrate the algorithms and their performance.

We outline the rest of the paper. In Section II, we describe our power and thermal model and provide the problem formulation. In Section III, we present a custom infeasible-start primal-dual algorithm, together with its convergence properties, implementation details, and some algorithm enhancements. In Section IV, we describe our dual decomposition method and prove convergence and show how we can recover a primal solution (optimal processor speeds) from the dual problem solution. In Sections V and VI, we present some numerical results for the algorithms and provide a comparison with a decentralized PI controller method. In Section VII, we list some variations and extensions of the problem and the proposed algorithms. We conclude the paper in Section VIII.

II. Problem Formulation

A. Power Dissipation Model

We consider n processors, sharing the same environment, where processor i operates at speed $s_i \in [s_{\min}, s_{\max}]$. For example, these could be n processor cores collocated on the same chip or n processor chips in a multichip package. Each processor can change its frequency and possibly also associated variables such as supply voltage, in order to regulate its speed. The power dissipation of processor i is a function of its speed variable:

$$p_i = \phi_i(s_i), \qquad i = 1, \ldots, n \tag{1}$$

where $\phi_i : [s_{\min}, s_{\max}] \to \mathbf{R}_+$ is a strictly increasing convex function. (The monotonicity assumption means the faster we run the processor, the more power it consumes; the convexity assumption means that the energy efficiency of the processor, in Joules per operation, decreases with increasing speed.) We also assume that the functions ϕ_i are differentiable and that $\phi_i'(s_{\min})$ and $\phi_i'(s_{\max})$ are well defined. We use $p = (p_1, \ldots, p_n)$ and $s = (s_1, \ldots, s_n)$ to denote vectors of processor powers and speeds. For future reference we define

$$p = \phi(s) \in \mathbf{R}^n \tag{2}$$

where $\phi(s) = (\phi_1(s_1), \ldots, \phi_n(s_n))$ denotes componentwise application of functions $\phi_1, \ldots, \phi_n$ to the vector s.

One common power dissipation function is $\phi_i(s_i) = \beta_i s_i^{\alpha_i}$, where $\alpha_i > 1$ and $\beta_i > 0$. In particular, in the well-known "cube" model (i.e., $\alpha_i = 3$),the power is proportional to the cube of processor speed [15].

B. Thermal Model

We observe the temperature (in °C) at m points in the system and denote the temperature vector as $T = (T_1, \ldots, T_m) \in \mathbf{R}^m$. The ambient temperature is denoted $T_{\text{amb}} \in \mathbf{R}$. We assume that $m > n$, since we will always have at least one temperature measurement at or near each processor.

In this paper, we focus on a steady-state thermal model, which does not capture any thermal transients. This is justified when power updates are carried out on a time scale exceeding ten or so milliseconds, since the thermal time constants for single chip and multichip packages are in the range of milliseconds [21]. For faster power updates, a dynamic thermal model will be needed. The methods in this paper can be extended to handle such problems, but for simplicity, we focus here on the steady-state model.

We use a linear model of the form

$$T = Gp + T_{\text{other}} + T_{\text{amb}}\mathbf{1} = G\phi(s) + T_{\text{other}} + T_{\text{amb}}\mathbf{1} \tag{3}$$

where $G \in \mathbf{R}_+^{m \times n}$, $\mathbf{1}$ denotes the vector of ones, and $T_{\text{other}} \in \mathbf{R}^m$. The matrix G maps the vector of processor powers into a temperature rise vector; $T_{\text{other}} \in \mathbf{R}^m$ is the contribution to the temperature due to other (uncontrollable or fixed) heat sources.

For future use, we give some properties of the matrix G. The entry G_{ij} has units of °C/W and gives the temperature rise at the point i due to 1 W of power dissipated by processor j. The matrix G is elementwise nonnegative ($G_{ij} \geq 0$) and, in theory, is always dense, i.e., $G_{ij} > 0$. However, it can be very well approximated by a sparse matrix by truncating small entries to zero. In this case, the nonzeros in the jth column of G correspond to the points that are thermally affected by the jth processor. The matrix G can be found by finite-element analysis (see the Appendix) or by direct measurements in an existing system, e.g., using the sensors described in [22].

C. Optimization Problem

The total processing capability (throughput) of the system is given by the sum of the processor speeds

$$U(s) = s_1 + \cdots + s_n = \mathbf{1}^T s. \tag{4}$$

This is a simple and a common choice for the overall utility derived by the system. Later, we will comment on the use of more complex system utility functions. We should also mention that processor speed itself need not give a good measure of processing utility; a more sophisticated utility model would include cache effects, threading, and other phenomena. For simplicity, though, we will use a utility based only on processor speeds in this paper.

The thermal constraint for the system is that no observed temperature in the system can exceed a given maximum temperature $T_{\max}$

$$T = G\phi(s) + T_{\text{other}} + T_{\text{amb}}\mathbf{1} \leq T_{\max}\mathbf{1} \tag{5}$$

where $\leq$ denotes componentwise inequality. If there is a temperature sensor at (or in practice, near) each heat source, then $T \leq T_{\max}\mathbf{1}$ will imply that temperatures *everywhere* are less or equal to $T_{\max}$, by the maximum principle for the heat equation, which states that the maximum temperature will always be located at a point where there is a power (heat) source.

The processor speed control problem is to choose processor speeds s (between the given limits) so as to maximize the overall system utility, subject to the thermal constraint

$$\text{maximize} \quad U(s)$$

$$\begin{aligned} \text{subject to} \quad & G\phi(s) + T_{\text{other}} + T_{\text{amb}}\mathbf{1} \le T_{\max}\mathbf{1} \\ & s_{\min} \le s \le s_{\max}. \end{aligned} \tag{6}$$

The problem variables are $s \in \mathbf{R}^n$; the problem data are $G \in \mathbf{R}_+^{m\times n}$, $T_{\text{amb}}, T_{\max} \in \mathbf{R}$, $T_{\text{other}} \in \mathbf{R}^m$, $s_{\min}, s_{\max} \in \mathbf{R}^n$, and the functions are $\phi_1, \ldots, \phi_n$.

The problem (6) is a convex optimization problem since the objective is a linear function, and all the constraint functions are convex; see, e.g., [23, Ch. 4]. To see that the thermal constraint (5) is convex, i.e., the components of its left-hand side are convex functions of s, we note that the matrix G is nonnegative, so each entry of T is a nonnegative-weighted linear combination of convex functions, and therefore convex. The problem (6) is readily solved using standard interior-point methods. Moreover, by exploiting the particular structure of the problem, we will see that interior-point methods can solve very large-scale instances of the problem, very efficiently.

For future use, we define the *temperature slack vector* $z \in \mathbf{R}^m$ as

$$z = T_{\max}\mathbf{1} - T = T_{\max}\mathbf{1} - G\phi(s) - T_{\text{other}} - T_{\text{amb}}\mathbf{1}. \tag{7}$$

The slack z_i gives the slack or margin in the thermal constraint $T_i \le T_{\max}$. The thermal constraint (5) then means that every element of z is nonnegative, i.e., $z \ge 0$. Note that z is readily found from the temperature measurements. For future use, we also define nonnegative slacks between the processor speeds and the lower and the upper speed limits as

$$z_l = s - s_{\min}, \qquad z_u = s_{\max} - s. \tag{8}$$

We finish this section with an important assumption. We assume that problem (6) is strictly feasible, which means that when all processors operate at minimum speed (i.e., $s = s_{\min}$), the thermal constraint (5) is satisfied strictly (i.e., with a positive margin):

$$G\phi(s_{\min}) + T_{\text{other}} + T_{\text{amb}}\mathbf{1} < T_{\max}\mathbf{1}. \tag{9}$$

This assumption is just Slater's condition for the problem (6) and, among other things, guarantees that strong duality holds; see [23, Sec. 5.2.3].

III. A Primal-Dual Interior-Point Method

In this section, we present an infeasible-start primal-dual interior-point method [23, Sec. 11.7], [24], [25, Ch. 19] to solve problem (6). The primal-dual interior-point method uses Newton's method, applied to a suitably modified form of the optimality conditions, which we describe shortly.

We start with the optimality conditions for (6). Let $\lambda \in \mathbf{R}_+^m$ be the dual variables associated with the thermal constraint (5), and $\lambda_l \in \mathbf{R}_+^n$ and $\lambda_u \in \mathbf{R}_+^n$ the dual variables associated with the lower and the upper speed limits, respectively. The Lagrangian function [see, e.g., [23, Ch. 5] of problem (6) is

$$\begin{aligned} L(s,\lambda,\lambda_u,\lambda_l) =& \mathbf{1}^T s - \lambda^T(G\phi(s) + T_{\text{other}} + T_{\text{amb}}\mathbf{1} - T_{\max}\mathbf{1}) \\ & - \lambda_u^T(s - s_{\max}) - \lambda_l^T(-s + s_{\min}) \\ =& \mathbf{1}^T s + \lambda^T z + \lambda_u^T z_u + \lambda_l^T z_l. \end{aligned}$$

The Karush–Kuhn–Tucker (KKT) optimality conditions for the problem are

$$\begin{aligned} \mathbf{1} - \operatorname{diag}(\nabla\phi(s))G^T\lambda - \lambda_u + \lambda_l &= 0 \\ \operatorname{diag}(\lambda)z &= 0 \\ \operatorname{diag}(\lambda_u)z_u &= 0 \\ \operatorname{diag}(\lambda_l)z_l &= 0 \\ \lambda, \lambda_u, \lambda_l &\ge 0 \\ z, z_u, z_l &\ge 0. \end{aligned}$$

A. Primal-Dual Search Direction

Next, we explicitly specify the slack variables in terms of the speeds and modify the complementary slackness conditions to obtain

$$\begin{aligned} \mathbf{1} - \operatorname{diag}(\nabla\phi(s))G^T\lambda - \lambda_u + \lambda_l &= 0 \\ G\phi(s) + T_{\text{other}} + T_{\text{amb}}\mathbf{1} - T_{\max}\mathbf{1} + z &= 0 \\ s - s_{\max} + z_u &= 0 \\ s_{\min} - s + z_l &= 0 \\ \operatorname{diag}(\lambda)z &= \sigma\mu\mathbf{1} \\ \operatorname{diag}(\lambda_u)z_u &= \sigma\mu\mathbf{1} \\ \operatorname{diag}(\lambda_l)z_l &= \sigma\mu\mathbf{1} \\ \lambda, \lambda_u, \lambda_l &\ge 0 \\ z, z_u, z_l &\ge 0 \end{aligned}$$

where a *centering parameter* $\sigma \in [0,1]$ and a *duality measure* $\mu > 0$ are parameters that set the accuracy of the approximation. (These parameters define a point on the so-called central path and describe the biased search directions in the primal-dual methods [24, Ch. 1].) In our case, the duality measure μ is defined by

$$\mu = \frac{(\lambda^T z + \lambda_u^T z_u + \lambda_l^T z_l)}{(m + 2n)}$$

which is the average value of the pairwise product between the slack and the dual variables. The goal is to reduce this duality measure as close to zero as possible.

The modified optimality conditions can be compactly written as

$$\begin{aligned} & r(s,\lambda,\lambda_u,\lambda_l,z,z_u,z_l) \\ & \quad = \begin{bmatrix} \mathbf{1} - \operatorname{diag}(\nabla\phi(s))G^T\lambda - \lambda_u + \lambda_l \\ G\phi(s) + T_{\text{other}} + T_{\text{amb}}\mathbf{1} - T_{\max}\mathbf{1} + z \\ s - s_{\max} + z_u \\ s_{\min} - s + z_l \\ \operatorname{diag}(\lambda)z - \sigma\mu\mathbf{1} \\ \operatorname{diag}(\lambda_u)z_u - \sigma\mu\mathbf{1} \\ \operatorname{diag}(\lambda_l)z_l - \sigma\mu\mathbf{1} \end{bmatrix} \\ & \quad = 0 \end{aligned}$$

where we require $\lambda, \lambda_u, \lambda_l, z, z_u, z_l \ge 0$.

The primal-dual search direction is the Newton step for solving the nonlinear equations $r(s,\lambda,\lambda_u,\lambda_l,z,z_u,z_l) = 0$. If $y = (s,\lambda,\lambda_u,\lambda_l,z,z_u,z_l)$ denotes the current point, the Newton step $\Delta y = (\Delta s, \Delta\lambda, \Delta\lambda_u, \Delta\lambda_l, \Delta z, \Delta z_u, \Delta z_l)$ is characterized by the linear equations

$$r(y+\Delta y) \approx r(y) + Dr(y)\Delta y = 0.$$

Therefore, we can find the Newton step from

$$Dr(y)\Delta y = -r(y)$$

which can be written out explicitly as shown at the bottom of the page (10), where

$$\begin{aligned}
H_s &= -\nabla^2\phi(s)\mathrm{diag}(G^T\lambda)\\
D_s &= \mathrm{diag}(\nabla\phi(s))\\
r_d &= \mathbf{1} - D_sG^T\lambda - \lambda_u + \lambda_l\\
r_p &= G\phi(s) + T_{\mathrm{other}} + T_{\mathrm{amb}}\mathbf{1} - T_{\max}\mathbf{1} + z\\
r_u &= s - s_{\max} + z_u\\
r_l &= s_{\min} - s + z_l\\
r_{\mathrm{centp}} &= \mathrm{diag}(\lambda)z - \sigma\mu\mathbf{1}\\
r_{\mathrm{centu}} &= \mathrm{diag}(\lambda_u)z_u - \sigma\mu\mathbf{1}\\
r_{\mathrm{centl}} &= \mathrm{diag}(\lambda_l)z_l - \sigma\mu\mathbf{1}.
\end{aligned}$$

B. Primal-Dual Algorithm

At each iteration of the algorithm, we solve the Newton system (10) to find a new search direction and then choose the step length, so the inequalities $\lambda, \lambda_u, \lambda_l, z, z_u, z_l \geq 0$ are strictly satisfied and the duality measure μ, together with the primal, dual, and centrality residuals, are properly reduced. It is important for the progress of the algorithm to balance out these various aims and as the primal and dual variables approach optimality, we equally need to drive the residuals to zero. Our infeasible primal-dual interior-point algorithm for solving problem (6) loosely follows the IPF algorithm in [24, p. 166], and it proceeds as follows:

given

$$\epsilon > 0, \sigma \in (0,1)$$

Initialize: s with $s_{\min} < s < s_{\max}$; $z = \mathbf{1}, \lambda = \mathbf{1}$

while $\mu > \epsilon$

1) Compute search direction Δy by solving (10).
2) Find a step length $\alpha \in (0,1]$ using the line search method described next.
3) Update: $y := y + \alpha\Delta y$.

The line search in step 2 is a standard backtracking line search, based on the reduction of the duality measure μ and norm of the residuals and modified to ensure that $\lambda, \lambda_u, \lambda_l, z, z_u, z_l > 0$. We denote the current iterate as $s, \tilde{\lambda} = (\lambda, \lambda_u, \lambda_l)$ and $\tilde{z} = (z, z_u, z_l)$, and the next iterate as $s^+, \tilde{\lambda}^+$ and $\tilde{z}^+$, i.e.,

$$s^+ = s + \alpha\Delta s, \quad \tilde{\lambda}^+ = \tilde{\lambda} + \alpha\Delta\tilde{\lambda}, \quad \tilde{z}^+ = \tilde{z} + \alpha\Delta\tilde{z}.$$

With the new dual and slack iterates $\tilde{\lambda}^+$ and $\tilde{z}^+$, we associate the new duality measure

$$\mu^+ = \frac{(\tilde{\lambda}^+)^T\tilde{z}^+}{(m+2n)}.$$

We first compute the largest positive step length, not exceeding one, that gives $\tilde{\lambda}^+ \geq 0$ and $\tilde{z}^+ \geq 0$, i.e.,

$$\begin{aligned}
\alpha^{\max} &= \sup\{\alpha \in (0,1] \,|\, \tilde{\lambda} + \alpha\Delta\tilde{\lambda} \geq 0, \tilde{z} + \alpha\Delta\tilde{z} \geq 0\}\\
&= \min\{1, \min\{-\tilde{\lambda}_i/\Delta\tilde{\lambda}_i \,|\, \Delta\tilde{\lambda}_i < 0\}, \ldots\\
&\quad \min\{-\tilde{z}_i/\Delta\tilde{z}_i \,|\, \Delta\tilde{z}_i < 0\}\}.
\end{aligned}$$

We start the backtracking with $\alpha = 0.99\alpha^{\max}$ and multiply α by $\rho \in (0,1)$ until we have

$$\|r_d\|_2 \leq \left(\frac{\|r_d^0\|_2}{\mu^0}\right)\beta\mu^+$$

$$\|(r_p, r_u, r_l)\|_2 \leq \left(\frac{\|\,(r_p^0, r_u^0, r_l^0)\,\|_2}{\mu^0}\right)\beta\mu^+ \tag{11}$$

$$\mathrm{diag}(\tilde{\lambda})\tilde{z} \geq \gamma\mu^+, \qquad \mu^+ \leq (1 - 0.01\alpha)\mu \tag{12}$$

where $\beta > 0$ and $\gamma \in (0,1)$ are the backtracking parameters, and $r_d^0, r_p^0, r_u^0, r_l^0, \mu^0$ are residual values and the duality measure given the initial starting points, respectively. The criteria in (11) enforce a decrease in the dual and primal residuals, while the criteria in (12) enforce reduction in the centrality residual and mandatory decrease in the duality measure.

Common choices for the algorithm and backtracking parameters are $\sigma = 0.5$, $\beta = 5$, $\gamma = 0.05$, and $\rho = 0.85$. We take the tolerance to be $\epsilon = 10^{-8}$.

The most expensive part of computing the primal-dual search direction is solving the linear system (10). Next, we present an efficient method for solving these equations.

C. Solving the Newton System

The linear system (10) has $5n + 2m$ linear equations in $5n + 2m$ variables, which can be solved directly for small n and m (e.g., using the PLU factorization and the forward/backward solve steps). However, more efficient solution methods can be obtained by exploiting structure in the problem, which we do next.

$$\begin{bmatrix}
H_s & -D_sG^T & -I & I & 0 & 0 & 0\\
GD_s & 0 & 0 & 0 & I & 0 & 0\\
I & 0 & 0 & 0 & 0 & I & 0\\
-I & 0 & 0 & 0 & 0 & 0 & I\\
0 & \mathrm{diag}(z) & 0 & 0 & \mathrm{diag}(\lambda) & 0 & 0\\
0 & 0 & \mathrm{diag}(z_u) & 0 & 0 & \mathrm{diag}(\lambda_u) & 0\\
0 & 0 & 0 & \mathrm{diag}(z_l) & 0 & 0 & \mathrm{diag}(\lambda_l)
\end{bmatrix}
\begin{bmatrix}
\Delta s\\ \Delta\lambda\\ \Delta\lambda_u\\ \Delta\lambda_l\\ \Delta z\\ \Delta z_u\\ \Delta z_l
\end{bmatrix}
= -\begin{bmatrix}
r_d\\ r_p\\ r_u\\ r_l\\ r_{\mathrm{centp}}\\ r_{\mathrm{centu}}\\ r_{\mathrm{centl}}
\end{bmatrix} \tag{10}$$

We note that the matrix $Dr(y)$ is highly structured since most of the blocks are diagonal. We also note that n will be a small or a modest number, probably at most in thousands [26]. (Current multiprocessor systems have n small, no more than a hundred. For example, Intel has recently demonstrated a prototype multiprocessor system with 80 processor cores [27]). Also recall that we usually have $m \gg n$, which will guide our choice of the pivoting order during block elimination of the system (10).

We first eliminate the variables $\Delta z = -GD_s\Delta s - r_p$, $\Delta z_u = -\Delta s - r_u$, and $\Delta z_l = \Delta s - r_l$ to obtain

$$\begin{bmatrix} H_s & -D_sG^T & -I & I \\ -\mathrm{diag}(\lambda)GD_s & \mathrm{diag}(z) & 0 & 0 \\ -\mathrm{diag}(\lambda_u) & 0 & \mathrm{diag}(z_u) & 0 \\ \mathrm{diag}(\lambda_l) & 0 & 0 & \mathrm{diag}(z_l) \end{bmatrix} \begin{bmatrix} \Delta s \\ \Delta\lambda \\ \Delta\lambda_u \\ \Delta\lambda_l \end{bmatrix} = - \begin{bmatrix} r_d \\ r_{\mathrm{centp}} - \mathrm{diag}(\lambda)r_p \\ r_{\mathrm{centu}} - \mathrm{diag}(\lambda_u)r_u \\ r_{\mathrm{centl}} - \mathrm{diag}(\lambda_l)r_l \end{bmatrix}.$$

Next we eliminate the dual variables

$$\begin{aligned} \Delta\lambda &= \mathrm{diag}(\lambda/z)(GD_s\Delta s + r_p) - \mathrm{diag}(z^{-1})r_{\mathrm{centp}} \\ \Delta\lambda_u &= \mathrm{diag}(\lambda_u/z_u)(\Delta s + r_u) - \mathrm{diag}\left(z_u^{-1}\right) r_{\mathrm{centu}} \\ \Delta\lambda_l &= \mathrm{diag}(\lambda_l/z_l)(-\Delta s + r_l) - \mathrm{diag}\left(z_l^{-1}\right) r_{\mathrm{centl}} \end{aligned}$$

to obtain the linear system

$$H\Delta s = r_s \tag{13}$$

where $H \in \mathbf{R}^{n\times n}$ and $r_s \in \mathbf{R}^n$ are given by

$$\begin{aligned} H &= \nabla^2\phi(s)\mathrm{diag}(G^T\lambda) + D_sG^T\mathrm{diag}(\lambda/z)GD_s \\ &\quad + \mathrm{diag}(\lambda_u/z_u) + \mathrm{diag}(\lambda_l/z_l) \\ r_s &= r_d + D_sG^T(\mathrm{diag}(z^{-1})r_{\mathrm{centp}} - \mathrm{diag}(\lambda/z)r_p) \\ &\quad + \mathrm{diag}\left(z_u^{-1}\right) r_{\mathrm{centu}} - \mathrm{diag}(\lambda_u/z_u)r_u \\ &\quad - \mathrm{diag}\left(z_l^{-1}\right) r_{\mathrm{centl}} + \mathrm{diag}(\lambda_l/z_l)r_l. \end{aligned}$$

The matrix H is a symmetric $n \times n$ positive semidefinite (PSD) matrix (since each of the summation terms is a symmetric PSD matrix) and thus the preferred method to solve (13) is via Cholesky factorization [23, App. C3]. Since n is modest (say, not more than 1000 or so) and H is generally dense (even when G is sparse), the cost of solving (13) is $(1/3)n^3$ flops. What dominates is forming the matrix H, specifically forming the subcomponent matrix

$$Y = D_sG^T\mathrm{diag}(\lambda/z)GD_s.$$

In the dense case, the cost of forming Y is mn^2 flops. When G is sparse, we can exploit the sparsity to form Y faster [28]. In summary, the flop count per iteration is approximately mn^2 when G is dense and can be as small as n^3 when G is sparse.

D. Convergence of the Method

Convergence properties of the primal-dual interior-point methods as applied to convex optimization problems have been investigated in numerous works and summarized in [23, Sec. 11.7], [24], [29]. Various theoretical results have been shown, such as polynomial-time complexity results for the total iteration count and the total computational cost, for similar algorithms. In practice, primal-dual interior-point algorithms typically converge to a very high accuracy in a few tens of iterations (between 20 and 100) regardless of the problem dimensions. In extensive computational experiments with the primal-dual algorithm described earlier, we found no case in which more than 36 iterations were required to achieve accuracy of $\epsilon = 10^{-8}$.

E. Warm-Starting

Suppose that we solve an instance of the processor speed control problem (6), and subsequently the problem data changes slightly. For example, T_{other} changes due to variations in the external (uncontrollable) power sources. In this case, we can reuse our knowledge of the previous optimal solution to initialize (i.e., *warm-start*) the primal-dual method when solving the perturbed problem [30], [31]. Extensive numerical tests show that with warm-starting the number of iterations to achieve $\epsilon = 10^{-4}$ accuracy drops down from around 23 to around 2–4 iterations.

We list some applications of the warm-start technique:

1) *Efficient Generation of Optimal Tradeoff Curves:* We can efficiently solve the problem as we sweep one parameter over some range, e.g., computing the optimal tradeoff between $T_{\max}$ and U.
2) *Tracking Changes in T_{other} and T_{amb}:* We can recompute the optimal processor speeds as the ambient temperature or temperature due to other heat sources, change. We can interpret this tracking of optimal speeds as a complicated nonlinear feedback law, where we reoptimize as the problem parameters change.
3) *Adjusting to Changes in G:* We can take into account changes in G, which can occur due to variation of the thermal conductance as a function of temperature. To model this effect, we can update G based on the current temperatures, reoptimize processor speeds (using warm-start), then update G again, and so on. In [19], experimental results show that this iterative procedure converges quickly (takes about 3–4 reoptimization steps).

IV. Dual Distributed Method

In the previous section, we interpreted a warm-starting primal-dual interior-point method as a complex nonlinear control law for the processor speed control given the temperature constraints, which can track changes in the problem data. In this section, we use the method of dual decomposition applied to problem (6) to derive a simpler nonlinear feedback law, which is in addition distributed.

A. Dual Problem

We start by deriving a dual problem for (6). The partial Lagrangian function for maximizing $U(s) = \mathbf{1}^Ts$ is given by

$$\begin{aligned} L(s,\lambda) &= \mathbf{1}^Ts - \lambda^T(G\phi(s) + T_{\mathrm{other}} + T_{\mathrm{amb}}\mathbf{1} - T_{\max}\mathbf{1}) \\ &= \sum_{i=1}^{n}\left(s_i - g_i^T\lambda\phi_i(s_i)\right) + \lambda^T(T_{\max}\mathbf{1} - T_{\mathrm{other}} - T_{\mathrm{amb}}\mathbf{1}) \end{aligned} \tag{14}$$

where $g_i \in \mathbf{R}^m$ is the ith column of G and $\lambda \in \mathbf{R}^m_+$ are the dual variables associated with the thermal constraint (5), as used in Section III. The dual variable $\lambda_i \geq 0$ can be viewed as the price of temperature violation at node i. The dual function is

$$g(\lambda) = \sum_{i=1}^{n} \psi_i \left(g_i^T \lambda\right) + \lambda^T (T_{\max}\mathbf{1} - T_{\text{other}} - T_{\text{amb}}\mathbf{1})$$

where

$$\psi_i(y) = \max_{s_i \in [s_{\min}, s_{\max}]} (s_i - y\phi_i(s_i)).$$

(For brevity, we omit the index i from the components of vectors $s_{\min}$ and $s_{\max}$, and we follow this convention throughout this section.) The functions ψ_i are convex, since by definition they are pointwise maximums of affine functions [23, Ch. 3]. Then the dual function $g(\lambda)$ is also convex since it is a sum of convex functions with linear terms.

The dual problem is

$$\begin{array}{ll} \text{minimize} & g(\lambda) \\ \text{subject to} & \lambda \geq 0 \end{array} \tag{15}$$

where the optimization variables are the dual variables $\lambda \in \mathbf{R}^m$. This is a convex optimization problem since it is a minimization of a convex function over a convex set (positive orthant). Let $\lambda^\star$ denote an optimal solution of the dual problem (15).

Before we proceed to give an algorithm for solving (15), we will give some properties of the dual function g and the functions ψ_i. We can readily find the value of the function ψ_i, which is attained at the optimal solution $s_i^\star(y)$ given by

$$\begin{aligned} s_i^\star(y) &= \arg \max_{s_i \in [s_{\min}, s_{\max}]} (s_i - y\phi_i(s_i)) \\ &= \begin{cases} \phi_i^{'-1}(\frac{1}{y}) & \phi_i^{'-1}(1/y) \in [s_{\min}, s_{\max}] \\ s_{\min} & \phi_i^{'-1}(1/y) \leq s_{\min} \\ s_{\max} & \phi_i^{'-1}(1/y) \geq s_{\max}. \end{cases} \end{aligned} \tag{16}$$

The function value is

$$\psi_i(y) = \begin{cases} s_i^\star(y) - y\phi_i\left(s^\star(y)\right) & s_i^\star(y) \in [s_{\min}, s_{\max}] \\ s_{\min} - y\phi_i(s_{\min}) & s_i^\star(y) \leq s_{\min} \\ s_{\max} - y\phi_i(s_{\max}) & s_i^\star(y) \geq s_{\max}. \end{cases}$$

Because $s^\star(G^T\lambda)$ is the unique maximizer of $L(s, \lambda)$ in (14), it follows that $s^\star(G^T\lambda^\star)$ is equal to an optimal solution $s^\star$ of the primal problem (6) given that the strong duality holds, which is true because of the assumption (9). (For more details, see [23, Sec. 5.5.5]). In other words, if prices are optimal, they lead to optimal speeds.

The function ψ_i is differentiable with

$$\psi_i^{'}(y) = \begin{cases} -\phi_i(s_i^\star(y)) & s_i^\star(y) \in [s_{\min}, s_{\max}] \\ -\phi_i(s_{\min}) & s_i^\star(y) \leq s_{\min} \\ -\phi_i(s_{\max}) & s_i^\star(y) \geq s_{\max}. \end{cases}$$

The gradient of the dual function is then given by

$$\begin{aligned} \nabla g(\lambda) &= \sum_{i=1}^{n} \psi_i^{'}\left(g_i^T\lambda\right) g_i + (T_{\max}\mathbf{1} - T_{\text{other}} - T_{\text{amb}}\mathbf{1}) \\ &= T_{\max}\mathbf{1} - G\phi(s^\star(G^T\lambda)) - T_{\text{other}} - T_{\text{amb}}\mathbf{1} \\ &= z(s^\star(G^T\lambda)) \end{aligned}$$

which is precisely the temperature slack at each location, evaluated for speeds $s^\star(G^T\lambda)$.

B. Distributed Algorithm

We describe a distributed algorithm for solving the dual problem (15) based on the projected gradient method with smoothing of the speeds. The given solution method is often called the dual decomposition method [32]–[34], [35, ch. 6].

In the algorithm, we start with any positive λ and repeatedly carry out the update

$$\lambda := (\lambda - \alpha\nabla g)_+ \tag{17}$$

where $\alpha > 0$ is the step size, and $(x)_+$ denotes the entrywise nonnegative part of the vector x (i.e., projection onto the nonnegative orthant). The full algorithm proceeds as follows.

given

$$\alpha > 0, \theta = (0, 1]$$

Initialize: s with $s_{\min} \leq s \leq s_{\max}$; $\lambda > 0$ (e.g., $\lambda = \mathbf{1}$)

repeat

1) Compute processor speeds, given current prices, using (16), and smooth out.

$$s := \theta s^\star(G^T\lambda) + (1 - \theta)s.$$

2) Measure (or compute) temp. slacks at the sensors.

$$z := T_{\max}\mathbf{1} - T.$$

3) Update temperature prices.

$$\lambda := (\lambda - \alpha z)_+.$$

The parameter θ acts to smooth out the sequence of speeds generated. When $\theta = 1$, the algorithm reduces to the classical projected subgradient method for the dual problem. We will show that for small enough α, s and λ converge to $s^\star$ and $\lambda^\star$, respectively. Also note that the temperature slack z can have negative entries during the algorithm execution, but at the optimum will be nonnegative.

The given algorithm is completely distributed (decentralized). Each processor updates its speed based on its previous speed and price information obtained from the neighboring sensors (i.e., sensors for which its power is affecting their temperature), while each sensor updates its price based on its local (measured or calculated) temperature slack. We can interpret the algorithm as a simple nonlinear feedback control law.

The method described here is one of the simplest projected subgradient dual algorithms. Far more sophisticated methods are described and analyzed in, e.g., [34], [36], [37].

C. Convergence Proof

In this section, we give a convergence proof for the distributed algorithm presented in Section IV-B, when the smoothing parameter θ is 1. (For an analysis of the algorithm for $\theta < 1$, see

[38, Sec. 3.2], [39]). The algorithm is based on the projected gradient method and a standard result is that the projected gradient method converges for a small enough fixed stepsize $\alpha > 0$; see, e.g., [35], [38], [40]. We give the convergence proof here for completeness, since we can explicitly evaluate the constants and estimate the rate of the convergence.

The distributed algorithm is given by the repeated updates of the form

$$\lambda^{(k+1)} = \left(\lambda^{(k)} - \alpha \nabla g\left(\lambda^{(k)}\right)\right)_+.$$

Let $\lambda^\star$ be an arbitrary optimal point. Then we have

$$(\lambda^\star - \alpha \nabla g(\lambda^\star))_+ = \lambda^\star$$

which comes from the (optimality) condition that $-\nabla g(\lambda^\star)$ supports $\mathbf{R}^m_+$ at $\lambda^\star$.

We consider the squared distance from the current point to an optimal point, and we show that this distance decreases with each iteration. We have

$$\begin{aligned}
&\left\|\lambda^{(k+1)} - \lambda^\star\right\|_2^2 \\
&\quad = \left\|\left(\lambda^{(k)} - \alpha \nabla g\left(\lambda^{(k)}\right)\right)_+ - (\lambda^\star - \alpha \nabla g(\lambda^\star))_+\right\|_2^2 \\
&\quad \le \left\|\lambda^{(k)} - \alpha \nabla g\left(\lambda^{(k)}\right) - \lambda^\star - \alpha \nabla g(\lambda^\star)\right\|_2^2 \\
&\quad = \left\|\lambda^{(k)} - \lambda^\star\right\|_2^2 \\
&\qquad - 2\alpha \left(\nabla g\left(\lambda^{(k)}\right) - \nabla g(\lambda^\star)\right)^T \left(\lambda^{(k)} - \lambda^\star\right) \\
&\qquad + \alpha^2 \left\|\nabla g\left(\lambda^{(k)}\right) - \nabla g(\lambda^\star)\right\|_2^2
\end{aligned}$$

where the inequality comes from the fact that a projection onto a convex set is contractive, i.e., after projection the distance between the projected points cannot increase.

Since g is a convex function, we always have

$$\left(\nabla g\left(\lambda^{(k)}\right) - \nabla g(\lambda^\star)\right)^T \left(\lambda^{(k)} - \lambda^\star\right) \ge 0. \tag{18}$$

To proceed with the proof, we need to derive a lower bound on the inner product in (18) and an upper bound for the quantity $\|\nabla g(\lambda^{(k)}) - \nabla g(\lambda^\star)\|_2$.

We obtain the lower bound

$$\begin{aligned}
&\left(\nabla g\left(\lambda^{(k)}\right) - \nabla g(\lambda^\star)\right)^T \left(\lambda^{(k)} - \lambda^\star\right) \\
&\quad = \left(G\phi(s^\star(G^T\lambda^\star)) - G\phi\left(s^\star\left(G^T\lambda^{(k)}\right)\right)\right)^T \left(\lambda^{(k)} - \lambda^\star\right) \\
&\quad \ge \sigma_{\min}(G)(\min_i \phi_i'(s_{\min})) \left\|\lambda^{(k)} - \lambda^\star\right\|^2
\end{aligned}$$

where $\sigma_{\min}(G)$ is the smallest singular value of G. Defining

$$M = \sigma_{\min}(G)(\min_i \phi_i'(s_{\min})) \tag{19}$$

we have

$$M \left\|\lambda^{(k)} - \lambda^\star\right\|^2 \le \left(\nabla g\left(\lambda^{(k)}\right) - \nabla g(\lambda^\star)\right)^T \left(\lambda^{(k)} - \lambda^\star\right). \tag{20}$$

The upper bound is given by the Lipschitz condition on the gradient ∇g,

$$\begin{aligned}
\|\nabla g(\lambda_1) - \nabla g(\lambda_2)\| &= \|G\phi(s^\star(G^T\lambda_1)) - G\phi(s^\star(G^T\lambda_2))\| \\
&\le \|G\|\|\phi(s^\star(G^T\lambda_1)) - \phi(s^\star(G^T\lambda_2))\| \\
&\le \|G\|(\max_i \phi_i'(s_{\max}))\|\lambda_1 - \lambda_2\|.
\end{aligned}$$

Defining

$$L = \sigma_{\max}(G)(\max_i \phi_i'(s_{\max})) \tag{21}$$

where $\sigma_{\max}(G) = \|G\|$ is the largest singular value of G, we have

$$\left\|\nabla g\left(\lambda^{(k)}\right) - \nabla g(\lambda^\star)\right\|_2^2 \le L \left\|\lambda^{(k)} - \lambda^\star\right\|_2^2. \tag{22}$$

Combining the bounds (20) and (22) with the inequality mentioned earlier, we get

$$\left\|\lambda^{(k+1)} - \lambda^\star\right\|_2^2 \le (1 - 2\alpha M + \alpha^2 L) \left\|\lambda^{(k)} - \lambda^\star\right\|_2^2 \tag{23}$$

which proves the convergence of the algorithm given that the stepsize satisfies

$$0 < \alpha < 2M/L. \tag{24}$$

The convergence rate is linear with the rate determined by the constant $K = 1 - 2\alpha M + \alpha^2 L$. This constant is minimized with the stepsize choice $\alpha = M/L$ and is equal to $K^\star = 1 - M/L$. In practice

$$\frac{M}{L} = \frac{\sigma_{\min}(G)(\min_i \phi_i'(s_{\min}))}{\sigma_{\max}(G)(\max_i \phi_i'(s_{\max}))} = \frac{1}{\kappa(G)} \frac{\min_i \phi_i'(s_{\min})}{\max_i \phi_i'(s_{\max})} \tag{25}$$

can be very small since it is related to the inverse of the condition number of $G(\kappa(G))$ and the minimum and maximum slope of the functions ϕ_i.

V. Numerical Example

In this section, we present some numerical results for a synthetic problem instance in order to illustrate performance of the proposed algorithms: the primal-dual algorithm in Section III and the distributed algorithm in Section IV. The example is meant only to illustrate our methods; in particular, it is not meant to be a realistic model of any real multicore processor.

As our example, we consider a chip with a 10×10 array of processors and a 55×75 grid of thermal sensors, which could be actual hardware sensors or nodes in the finite-element thermal model. Thus, our problem dimensions are $n = 100$ and $m = 4125$. The locations of processors and sensors are shown in Fig. 1. We take $T_{\text{amb}} = 40\,^\circ\text{C}$ (the ambient temperature

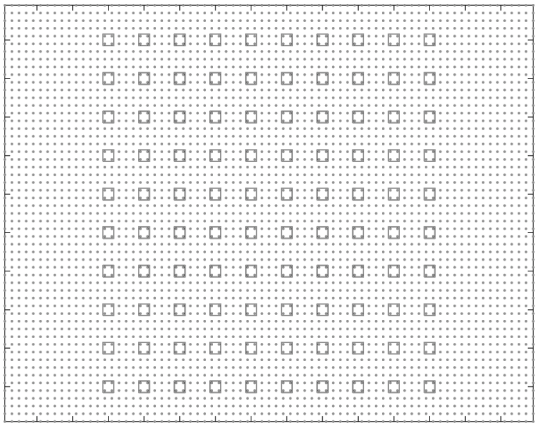

Fig. 1. Locations of processors (squares) and thermal sensors (dots) on the chip.

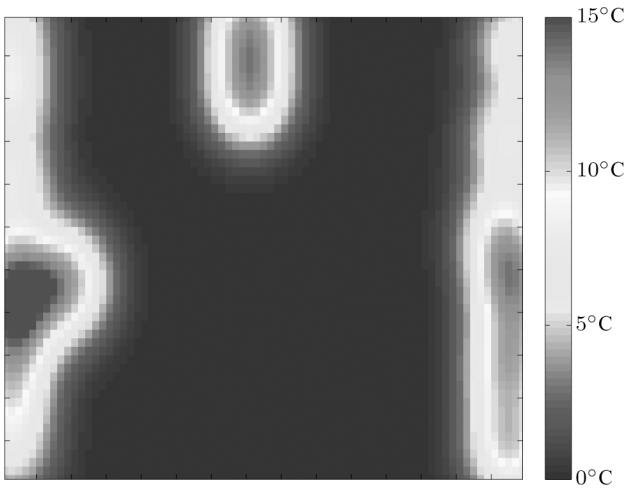

Fig. 2. Map of temperature rise due to other uncontrollable sources.

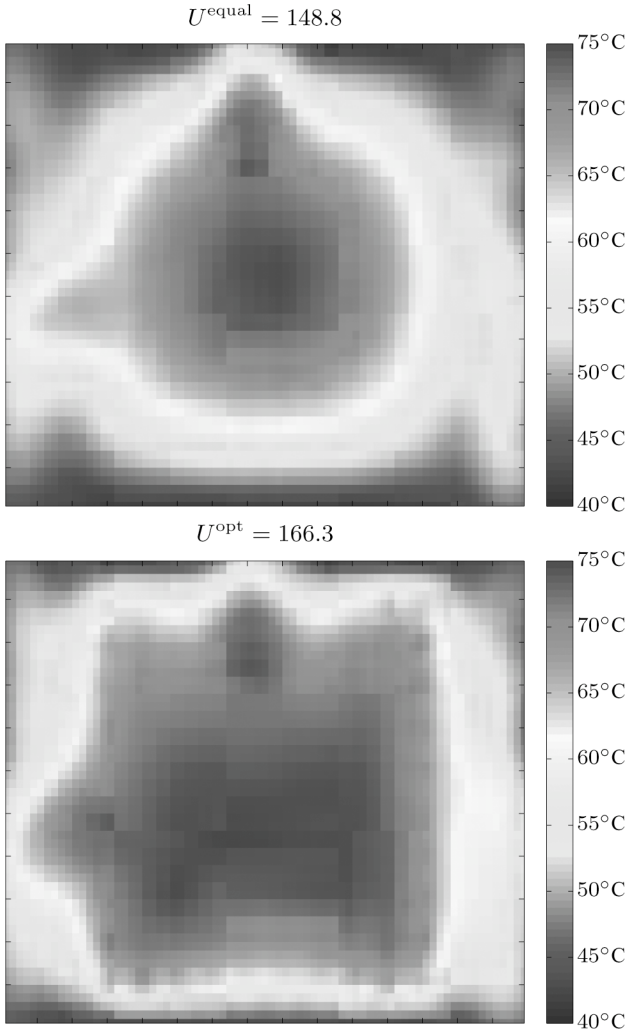

Fig. 3. *Top*: Temperature distribution with all speeds equal. *Bottom*: Temperature distribution with optimal speed allocation.

in a closed system environment), $T_{\max} = 75\,^{\circ}\mathrm{C}$, and we use the synthetically generated temperature map in Fig. 2 as T_{other}. The matrix G was generated using the finite-element model, as described in the Appendix, with thermal conductance $k = 1$ between all the nodes, and thermal conductance $k = 10$ between all boundary nodes and the external environment. We take $s_{\min} = \mathbf{1}$ and $s_{\max} = 3 \cdot \mathbf{1}$. We take the power functions to all be the same, $p_i = \phi_i(s_i) = s_i^3$, following the well-known cube speed-power model.

We implemented the algorithms in Matlab and performed numerical simulations on a 2.8-GHz Athlon CPU, with 1 GB of RAM, running Linux. We used the CVX package [41] to verify correctness of the results obtained by our algorithms.

A. Comparison With Equal-Speed Scheme

We compare performance of the primal-dual algorithm versus a simple suboptimal scheme for (6), in which we set all processor speeds to be equal and increase the speed until the thermal constraint becomes active. The equal-speed (left) and optimal (right) temperature distributions are shown in Fig. 3, together with the values of the system throughput. We note that the optimal speed allocation achieves about 12% throughput improvement while respecting the thermal constraint. A histogram of the optimal processor speeds is shown in Fig. 4, together with the value of the speed chosen for all the processors when the speeds are all equal.

Fig. 5 shows the optimal tradeoff curve between $T_{\max}$ and the maximum achievable throughput U^{opt} and U^{equal}, for the optimal primal-dual method and the equal-speed scheme, respectively. We note that the optimal scheme considerably outperforms the equal-speed scheme for all values of $T_{\max}$.

B. Performance of the Primal-Dual Method

Fig. 6 shows the plot of the duality measure μ and the residual $\|(r_p, r_d)\|_2$ versus the iteration number for a single run of the algorithm. We see that the primal-dual algorithm converges within 35 iterations.

Our simple Matlab implementation requires 2.3 s to solve the problem, using dense G (with all 412 500 entries). We truncated the entries in G with values below 0.15 and then rescaled each column to have the same column sum as the original matrix. (This preserves the total temperature rise from each source.) This results in a sparse matrix $\hat{G}$ with 27680 entries, i.e., a sparsity around 6.7%. This results in less than 1.3% error in the computed speeds. The time required to solve the problem, using the sparse thermal model with $\hat{G}$, was around 0.7 s. We estimate that

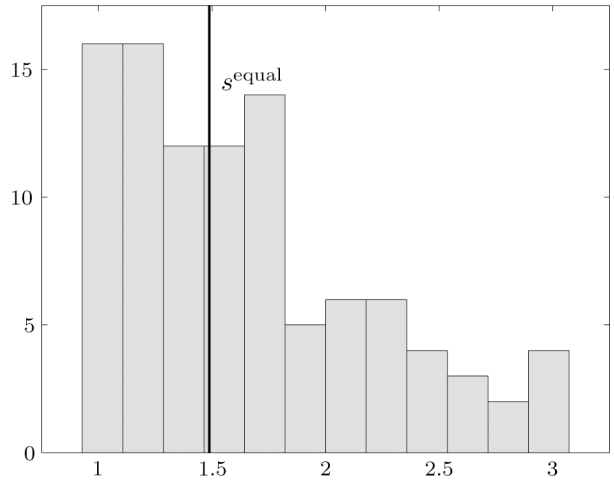

Fig. 4. Histogram of optimal processor speeds, together with the value of optimal equal speed s^{equal}.

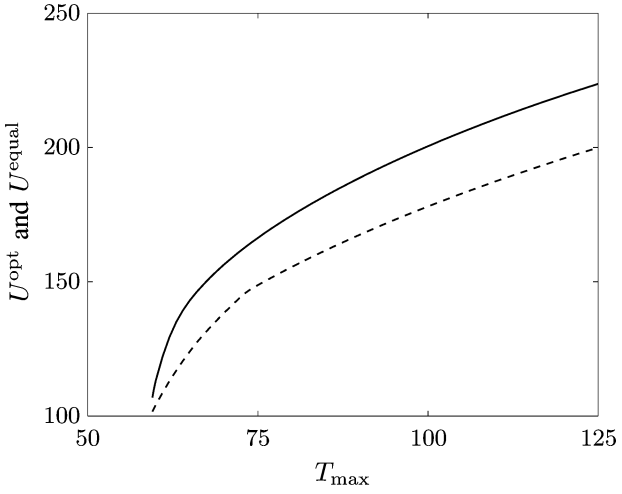

Fig. 5. Optimal tradeoff curve for $T_{\max}$ versus U^{opt} (solid curve) and tradeoff curve between $T_{\max}$ and suboptimal U^{equal} when all speeds are equal (dashed curve).

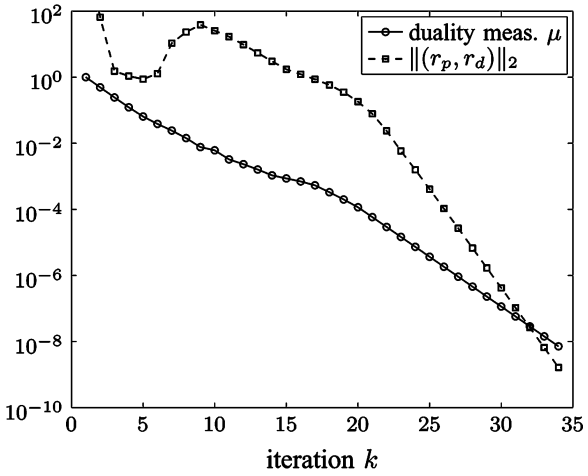

Fig. 6. Convergence of the primal-dual method: duality measure μ and norm of the residual $\|(r_p, r_d)\|_2$.

a *C* implementation of our algorithm would execute in around 0.1 s for the same sparse instance.

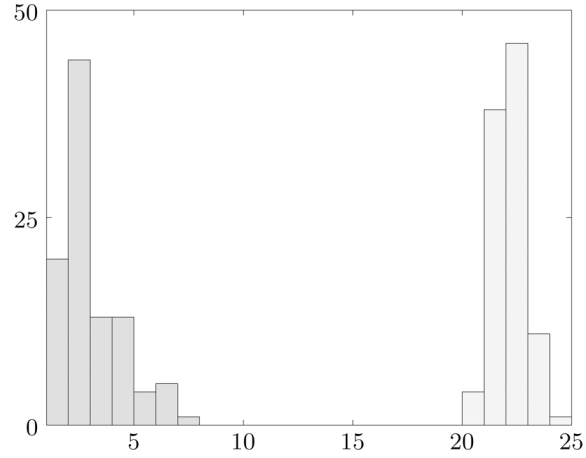

Fig. 7. Histogram of the total number of iterations needed for warm-start initialization (left, darker) versus cold-start initialization (right, lighter).

C. Warm-Starting the Primal-Dual Method

In this section, we experimentally verify some benefits of the warm-starting technique for the primal-dual method, as described in Section III-E.

We first investigate the benefit of warm-starting when the primal-dual method is used to track changes in the problem data. We consider the following experiment. Suppose that the uncontrollable power sources and the ambient temperature vary $\pm 1\%$ around their nominal values. We randomly generate 100 instances of these parameters, satisfying the aforementioned setup and solve the perturbed problems, using both the warm-start and cold-start techniques. In the warm-start technique, we initialize each new problem with the solution of the previous problem, while in the cold-start technique, we initialize the problem with speeds just above the minimum speed of the processors, e.g., $s = s_{\min} + 0.01$.

Fig. 7 shows the histograms of the number of iterations until each problem is solved within accuracy of $\epsilon = 10^{-4}$. We note that the warm-start technique performs very well and solves each instance of the problem within 1–7 iterations (typically 2 iterations), as compared to about 20–25 iterations needed by the cold-start. These results confirm that the primal-dual method coupled with warm-starting can act as an effective (though complex) nonlinear control law, which tracks changes in the problem parameters. Our Matlab implementation can execute the warm-start optimization in around 100 ms; we estimate that a *C* implementation would execute in around 10 ms.

As the second experiment, we investigate the use of warm-starting to efficiently generate optimal tradeoff curves between competing problem parameters. The optimal tradeoff between $T_{\max}$ and U^{opt} in Fig. 5 was generated using the warm-starting technique. The first point was obtained using the cold-start and then subsequently we used warm-starting to obtain remaining 100 points on the curve. The primal-dual method required 30 iterations to obtain the first point, after which it needed only about 2–3 iterations for each new point. In our example, the total tradeoff curve was generated using about 280 total iterations, while the cold-start method required close to 2180 iterations. Therefore, warm-starting can be used to efficiently perform tradeoff analysis and explore the problem design space.

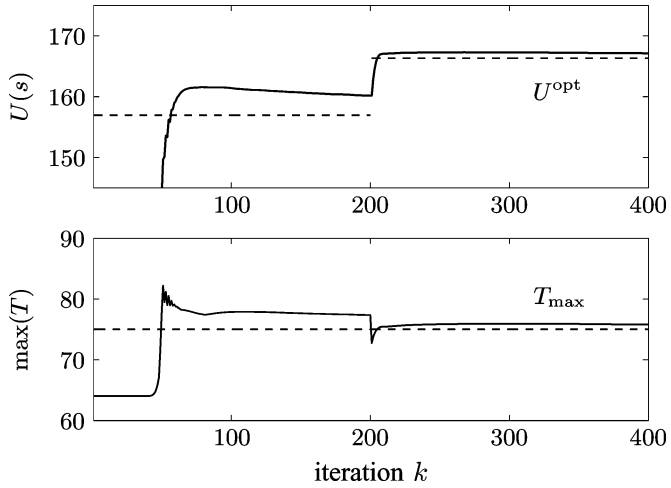

Fig. 8. Convergence of the control law with no smoothing $\theta = 1$, $\alpha = 10^{-4}$, and starting from $\lambda = 0.11$. *Top*: Primal objective value. *Bottom*: Maximum temperature on the chip versus $T_{\max}$ (dashed line).

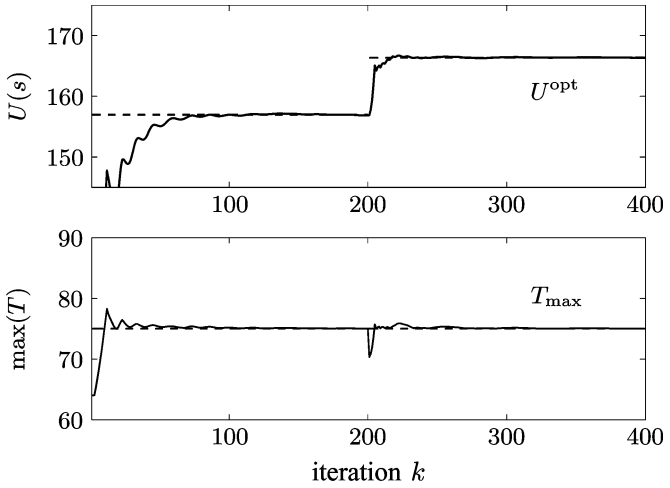

Fig. 9. Convergence of the control law with smoothing $\theta = \alpha = 0.03$ and starting from $\lambda = \mathbf{1}$. *Top*: Primal objective value. *Bottom*: Maximum temperature on the chip versus $T_{\max}$ (dashed line).

D. Performance of the Distributed Algorithm

In this section, we investigate performance of the distributed algorithm when used as a simple distributed control law, which tracks changes in the problem parameters.

We test the algorithm, using the problem setup described at the beginning of Section V. For our problem instance, we found constants $M = 1.48$ and $L = 1057.1$, which imply that the proposed algorithm will converge as long as $0 < \alpha < 2.8 \times 10^{-3}$. However, extensive numerical experiments suggest that the algorithm when used with smoothing actually converges for larger values of α, up to around 0.05.

We investigate how well the distributed algorithm tracks changes in the ambient temperature and the power supplied by uncontrollable external sources. We introduce the following event: at iteration $k = 200$, we randomly perturb the ambient temperature and the powers of the external sources by $\pm 10\%$ around their initial values. Figs. 8 and 9 show convergence of the maximum temperature and the system throughput, for the distributed method with no smoothing ($\theta = 1$) and one with smoothing (where we set $\theta = \alpha = 0.03$), respectively. We compare the convergence curves versus the optimal values obtained by the primal-dual method. We note that the distributed algorithm with smoothing reacts very well to the changes, much better than the standard method without smoothing (which takes a longtime to converge to the optimal values). In our computational experiments, we also found that the method with smoothing was much more robust to more aggressive stepsizes and initialization from different λ.

VI. Cell Processor Case Study

As a more realistic example, we consider the first-generation cell processor [42], a nine core system jointly developed by Sony, Toshiba, and International Business Machines Corporation. The processor features eight *Synergistic Processing Elements* (SPEs), to which the workloads are assigned by a *Power Processor Element* (PPE). The fastest operation of each core was clocked at 5.6 GHz with a 1.4 V supply at 56 °C under laboratory conditions; however, the cores are commercially usually run at 3.2 GHz [42].

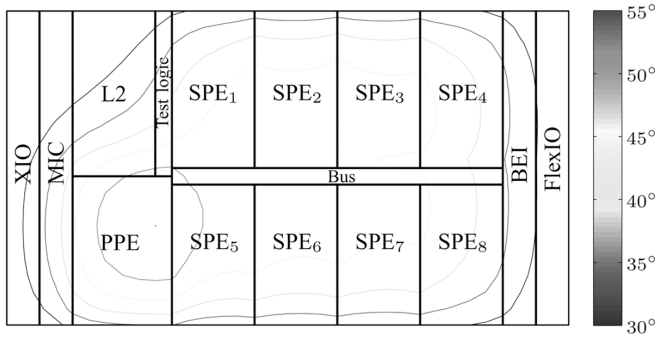

Fig. 10. Cell processor floorplan and thermal contour map during a regular loading of the processor.

In [43], the authors show that due to an aggressive low-cost thermal design of the cell processor, the PPE exhibits a thermal hot spot during standard operation when all the cores ran at 3.2 GHz. Using the setup in [43] and our simplified thermal model (see the Appendix), we reproduce these results in Fig. 10, which shows the processor floorplan overlaid with a thermal contour map. (In this case, the maximum die temperature is 60 °C, measured at the PPE.) As a solution to this uneven processor heating, the processor implements a sophisticated thermal sensing and system cooling techniques. Its ten local digital thermal sensors provide early warnings of any temperature increase, and a linear thermal sensor measures the die's global temperature.

A. Speed Assignment for SPEs

We use the centralized method (see Section III) to find an optimal speed assignment for the processor, where we set $s_{\min} = 2$ GHz and $s_{\max} = 5$ GHz based on the data-sheet and take the power functions to follow the cube speed-power model, i.e., $p_i = \phi_i(s_i) = s_i^3$. We let $T_{\text{amb}} = 20\,^{\circ}\text{C}$ and $T_{\max} = 60\,^{\circ}\text{C}$ (a safe operating temperature for the given speed limits).

We obtain the optimal speeds

$$s_{\text{SPE}} = (3.77, 3.62, 3.58, 3.90, 3.47, 3.88, 3.46, 4.10)\ \text{GHz}$$

and $s_{\text{PPE}} = 4.46$ GHz. The total processing capability of the processor in the (nominal) equal-speed case is $U_{\text{nom}} = 9 \times$

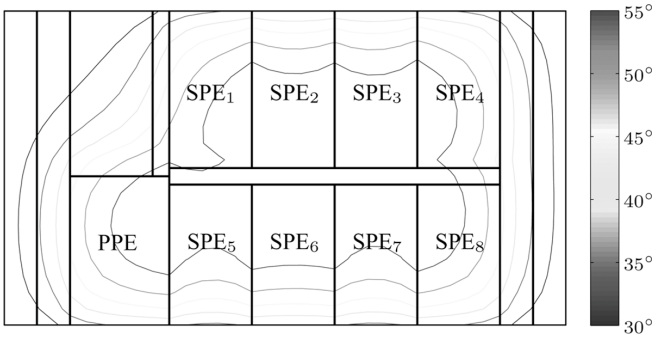

Fig. 11. Thermal contour map given an optimal speed assignment for the processor.

$3.2 = 28.8$ GHz, while the optimal result is $U_{\text{opt}} = 34.3$ GHz, which is a 19% increase. The resulting temperature profile is shown in Fig. 11. We note that the temperature is now more evenly distributed across the system. Our methodology can be used to quickly calculate optimal speed assignment policies given various operating conditions, thus exploring the performance limits of the processor.

B. Dynamic Thermal Management

We investigate a simple dynamic thermal management of the processor, using the dual distributed method described in Section IV. We present some comparative results versus a decentralized PI control [3], [5], in which individual PI controllers govern each core, typically selecting the hottest of the available temperature sensors as the input signal. This decentralized control was shown to achieve large gains over suboptimal schemes such as, e.g., the equal-speed scheme. In particular, we use the decentralized PI controller proposed by Donald and Martonosi in [3, Sec. 4.1], parametrized by the proportional gain $K_p = 0.0107$ and the integral term K_i tuned to give the best performance for the following setup.

We let T_{amb} vary sinusoidally between 5 °C and 35 °C as

$$T_{\text{amb}} = 20 + 15\sin(2\pi k/1000)\ ^\circ\text{C}$$

where k is the iteration number. We use the dual distributed control law with smoothing $(\theta = \alpha = 0.03)$ and PI controller to compute processor speeds given the varying T_{amb} such that $T \leq T_{\max} = 60$ °C while maximizing the total processing capability. We initialize both of the methods from $s = s_{\min} = 2$ GHz. Since the PI controller does not handle speed constraints directly, we modify the algorithm to clip the speeds if they go past their limits.

For comparison purposes, we compute U^{opt} at each iteration, using warm-started centralized method from Section III. The performance results are shown in Fig. 12. We observe that the dual-distributed method, after a brief settling time, perfectly tracks the optimal speed assignments. The PI controller also performs very well; however, at some instances it is up to 8% suboptimal to the distributed method (which obtains globally optimal performance). In contrast, the PI controller has a faster settling time and it was tuned not to overshoot $T_{\max}$, while the distributed method briefly overshoots $T_{\max}$ by 1.6 °C.

Figs. 13 and 14 show processor speed allocations versus the iteration number for the dual distributed and PI control scheme,

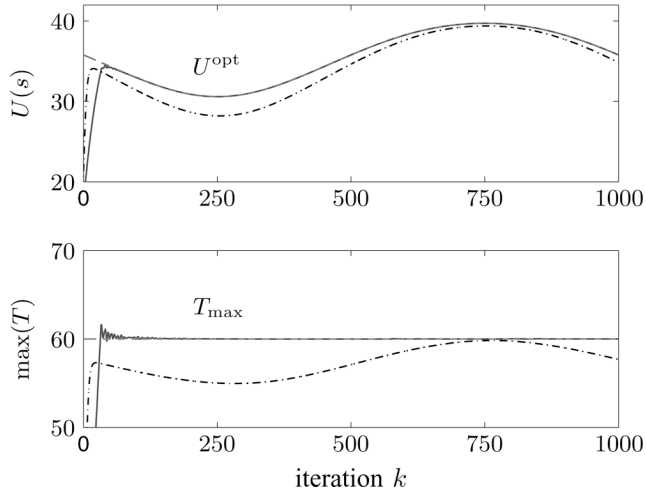

Fig. 12. Dynamic thermal tracking using the dual-distributed method (solid line) and the PI control law (dash-dotted line). *Top*: Obtained throughput versus the optimal value (dashed line). *Bottom*: Maximum temperatures on the chip versus $T_{\max}$ (dashed line).

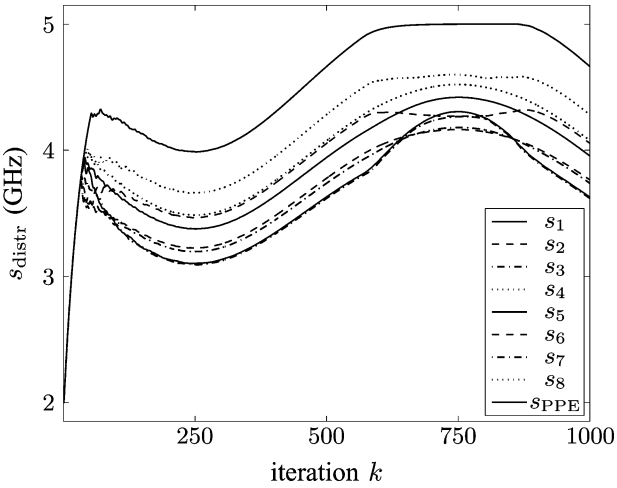

Fig. 13. Processor speeds versus iteration using the dual-distributed algorithm.

respectively. We note that the PI controller due to its decentralized nature does not take into account temperature cross-coupling between cores, while the dual-distributed method handles this cross-coupling through the dual variables and at some instances switches relative order of core speeds.

We make some final comments, concerning the comparison of our proposed method and the PI controller in [3]. Our first comment is that the methods are not as different as they appear, since our dual-distributed method can be interpreted as a nonlinear feedback control law. Moreover, both methods have a low computational complexity. One difference is that our method directly takes into account constraints, whereas [3] handles constraints indirectly. On the other hand, historically, decentralized PI control has worked very well in a wide variety of real applications.

VII. Extensions

We also list some variations and extensions of the processor speed control problem (6) and the proposed algorithms. First, we can easily substitute a nonlinear concave differentiable

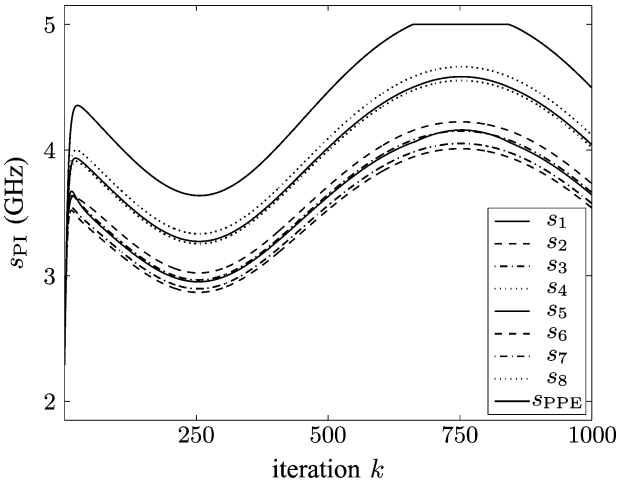

Fig. 14. Processor speeds versus iteration using the PI controller.

utility function for the total throughput $U(s) = \mathbf{1}^T s$ used in this paper. For example, we can use a logarithmic utility, $U^{\log}(s) = \sum_{i=1}^n \log s_i$, as is commonly used in economics and networking. This utility function will achieve proportional fairness, at least when the lower and upper processor speed bounds are not active. Both the primal-dual and dual decomposition methods are readily modified to handle such problems.

A second extension is to the case in which active cooling is used. Here, we have some additional actuators, also within our control, that can pump heat out of the environment, up to some given maximum rate. Active cooling can be modeled by adding the term $-Bu$ to the model for T, where B is an elementwise nonnegative matrix and u is the nonnegative vector of cooling rates. If there is no cost to using cooling, we would obviously operate the cooling at maximum rate. But we can subtract a cost (say, in energy) of using the cooling or limit its total. The result is still a convex optimization problem.

Third, we can turn the problem formulation around and minimize the maximum temperature subject to obtaining a specified level of throughput, or some other convex constraint on the speed variable. This problem can be formulated as the following optimization problem (in epigraph form):

$$\begin{array}{ll} \text{minimize} & T \\ \text{subject to} & G\phi(s) + T_{\mathrm{other}} + T_{\mathrm{amb}}\mathbf{1} \le T \\ & s_{\min} \le s \le s_{\max} \\ & w(s) \in \mathcal{W} \end{array} \tag{26}$$

where the variables are $T \in \mathbf{R}$ and $s \in \mathbf{R}^n$. The problem data are $G \in \mathbf{R}^{m\times n}$, $T_{\mathrm{other}} \in \mathbf{R}^m$, $T_{\mathrm{amb}} \in \mathbf{R}$, $s_{\min}, s_{\max} \in \mathbf{R}^n$, the functions $\phi_1, \ldots, \phi_n$, and the speed constraint set $\mathcal{W}$, which specifies the system workload. A simple example, in which a given throughput is required, is $\mathcal{W} = \{s \,|\, \mathbf{1}^T s = U^{\mathrm{req}}\}$.

This is a convex optimization problem, when the workload constraint set is defined by a set of linear equalities and convex inequalities. We can construct an infeasible-start primal-dual interior-point method for (26) similar to the one proposed in this paper. For more details on solving a particular instance of such a problem using geometric programming, see [44].

A fourth extension is to the dynamic case. In this paper, we have ignored the thermal dynamics, assuming that the speed control takes place on a slower time scale than the thermal time constant of the system. But the same basic ideas used here for the static case can be used to handle the case where the system thermal dynamics are taken into account.

VIII. Conclusion

In this paper, we have presented two algorithms for adjusting speeds of multiple processors subject to common thermal constraints. The first algorithm is an efficient implementation of an infeasible-start primal-dual interior-point method, while the second one is a distributed method based on dual decomposition. Both algorithms can be interpreted as nonlinear control laws that can track changes in the problem data and therefore can be used for real-time processor speed control in modern multiprocessor systems. Numerical simulations on the examples shown (and others) verify that both algorithms perform well in practice. In the future, we will experimentally verify performance of these algorithms, implemented on a real multiprocessor system.

Our final comments concern the question of how the control laws proposed in this paper might be used in a real multicore processor. First, it seems to us that unless n is substantial (say, tens), the benefits of sophisticated speed control would not justify the cost. If the methods were deployed, we would imagine using aggressively simplified thermal models (which might, indeed, be formed or updated using system identification [45] in actual operation), which would reduce the computational burden associated with the control law. This would also bring the update time down to below the millisecond range, allowing fast speed updates if needed. We hope that future research will address these issues.

Appendix

Thermal modeling of electronic (and other) devices is a well-developed field. For example, finite-difference and finite-element methods are discussed in [10], [11], [46], [47]; Green's function methods are considered in [48]. In this appendix, we show how the simple thermal model (3) used in this paper can be derived from the finite-difference method.

The steady-state temperature distribution over a region $\mathcal{R}$, with boundary isothermal at temperature T_{amb}, is governed by the Poisson (steady-state heat) equation

$$\nabla \cdot (k(x)\nabla T(x)) - P(x) = 0, \quad T(x) = T_{\mathrm{amb}} \text{ for } x \in \partial\mathcal{R}. \tag{27}$$

Here, $T(x)$ is the temperature, $k(x)$ is the thermal conductivity, and $P(x)$ is the power density of the heat source(s), at location $x \in \mathbf{R}^3$. (Here we ignore nonlinear terms in the thermal model, such as temperature dependence of thermal conductivity.)

We approximate the partial differential equation (27), using finite-difference discretization with m sample points, and a special (ground) node, which represents the ambient environment, assumed to be isothermal, with temperature T_{amb}. Heat flows along edges of a directed graph (typically a mesh) with $m+1$ nodes (the sample points and the ground node), and l directed edges between the points. We associate temperatures with the

nodes, and heat flow with the edges, with positive heat flow meaning heat flow in the direction of the edge, and negative heat flow meaning heat flow in the opposite direction. The graph can represent a 1-D, 2-D, or 3-D physical system and can include nodes for the substrate, packaging and heat sinks, effects of the outside environment, and so on.

We let $T_i \in \mathbf{R}$ denote the temperature at the ith node. Then the vector $T = (T_1, \ldots, T_m) \in \mathbf{R}^m$ gives the (discrete) temperature distribution of the system. We let $P = (P_1, \ldots, P_m) \in \mathbf{R}^m_+$ denote the powers injected into the nodes (due to the processors and other power sources). We have

$$P = Bp + P_{\rm other}$$

where $B \in \mathbf{R}^{m\times n}_+$ gives the distribution of the processor powers $p = (p_1, \ldots, p_n)$ into the nodes, and $P_{\rm other} \in \mathbf{R}^m_+$ is the power injected into the nodes due to other sources. (B_{ij} gives the fraction of the power dissipated by the processor j into the node i.) Typically, B and $P_{\rm other}$ are sparse.

With each edge we associate a thermal conductance $k_i \in \mathbf{R}_+$, which is (roughly) the average of $k(x)$ over the region between the nodes connected by the ith edge. The thermal conductivity vector is then $k = (k_1, \ldots, k_l) \in \mathbf{R}^l_+$.

Let $A \in \mathbf{R}^{m\times l}$ denote the reduced node-incidence matrix for the graph, defined as

$$A_{ij} = \begin{cases} +1, & \text{edge } j \text{ goes to node } i \\ -1, & \text{edge } j \text{ goes from node } i \\ 0, & \text{otherwise.} \end{cases}$$

Each column of A describes a directed edge. If the edge goes between two sample nodes, the column has exactly two nonzero entries, one $+1$ and one -1. If the edge goes to or from the ground node, the column has only one nonzero entry.

The discrete analog of the steady-state heat equation (27) is then

$$(AKA^T)(T - T_{\rm amb}\mathbf{1}) - P = 0 \tag{28}$$

where $K = \mathrm{diag}(k) \in \mathbf{R}^{l\times l}$. The matrix $AKA^T \in \mathbf{R}^{m\times m}$ is a *weighted Laplacian* matrix and is positive definite when A is full rank (which occurs when the graph, including the ground node, is connected). We solve (28) to obtain

$$\begin{aligned} T &= (AKA^T)^{-1}P + T_{\rm amb}\mathbf{1} \\ &= (AKA^T)^{-1}Bp + (AKA^T)^{-1}P_{\rm other} + T_{\rm amb}\mathbf{1} \end{aligned}$$

which has the form (3), with

$$G = (AKA^T)^{-1}B, \quad T_{\rm other} = (AKA^T)^{-1}P_{\rm other}.$$

It is well known that $(AKA^T)^{-1}$ has all positive entries (since the graph is connected), so the matrix G also has all nonnegative entries.

REFERENCES

[1] S. Irani and K. Pruhs, "Algorithmic problems in power management," *S1GACT News*, vol. 36, no. 2, pp. 63–76, 2005.

[2] K. Skadron, M. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, "Temperature- aware computer systems: Opportunities and challenges," *IEEE Micro.*, vol. 23, no. 6, pp. 52–61, Nov./Dec. 2003.

[3] J. Donald and M. Martonosi, "Techniques for multicore thermal management: Classification and new exploration," in *Proc. 33rd Int. Symp. Compt. Archit. (ISCA 2006)*, 2006, pp. 78–88.

[4] D. Brooks and M. Martonosi, "Dynamic thermal management for high-performance microprocessors," in *Proc. 7th Int. Symp. High-Performance Comput. Archit. (HPCA)*, Jan. 2001, pp. 171–182.

[5] K. Skadron, T. Abdelzaher, and M. Stan, "Control-theoretic techniques and thermal-RC modeling for accurate and localized dynamic thermal management," in *Proc. 8th Int. Symp. High-Performance Comput. Archit. (HPCA)*, 2002, pp. 17–28.

[6] E. Rohou and M. Smith, "Dynamically managing processor temperature and power," in *Proc. 2nd Workshop Feedback-Directed Optimization*, Nov. 1999, pp. 66–73.

[7] J. Srinivasan and S. Adve, "Predictive dynamic thermal management for multimedia applications," in *Proc. 17th Int. Conf. Supercomputing (ICS 2003)*, Jun 2003, pp. 109–120.

[8] M. Huang, J. Renau, S.-M. Yoo, and J. Torrellas, "A framework for dynamic energy efficiency and temperature management," in *Proc. 33rd ACM/IEEE Int. Symp. Microarchitecture (MICRO 2000)*, 2000, pp. 202–213.

[9] W. Hung, Y. Xie, N. Vijaykrishnan, M. Kandemir, and M. Irwin, "Thermal-aware allocation and scheduling for systems-on-chip," in *Proc. Des. Autom. Test Eur. (DATE 2005)*, 2005, pp. 898–899.

[10] K. Skadron, M. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan, "Temperature-aware microarchitecture: Modeling and implementation," *ACM Trans. Archit. Code Optim.*, vol. 1, no. 1, pp. 94–125, 2004.

[11] K. Ghobadi, "A heat-transfer optimization problem," Master's Thesis, McMaster University, Hamilton, ON, Aug. 2006.

[12] C.-F. Juang and C.-H. Hsu, "Temperature control by chip-implemented adaptive recurrent fuzzy controller designed by evolutionary algorithm," *IEEE Trans. Circuits Syst I, Reg. Papers*, vol. 52, no. 11, pp. 2376–2384, Nov. 2005.

[13] W. Daasch, C. Lim, and G. Cai, "Design of VLSI CMOS circuits under thermal constraint," *IEEE Trans. Circuits Syst II, Exp. Briefs*, vol. 49, no. 8, pp. 589–593, Aug. 2002.

[14] D. Ma and C. Zhang, "Thermal compensation method for CMOS digital integrated circuits using temperature-adaptive DC-DC converter," *IEEE Trans. Circuits Syst II, Exp. Briefs*, vol. 53, no. 11, pp. 1284–1288, Nov. 2006.

[15] N. Bansal, T. Kimbrel, and K. Pruhs, "Dynamic speed scaling to manage energy and temperature," in *Proc. 45th IEEE Symp. Found. Comput. Sci. (FOCS 2004)*, Washington, DC, 2004, pp. 520–529.

[16] K. Pruhs, R. van Stee, and P. Uthaisombut, "Speed scaling of tasks with precedence constraints," in *Proc. 3rd Workshop Approximation Online Algorithms, LNCS*, 2005.

[17] R. Rao, S. Vrudhula, C. Chakrabarti, and N. Chang, "An optimal analytical solution for processor speed control with thermal constraints," in *Proc. 2006 Int. Symp. Low Power Electron. Des.*, 2006, pp. 292–297.

[18] R. Jejurikar and R. Gupta, "Energy aware task scheduling with task synchronization for embedded real time systems," *IEEE Trans. Comput.-Aided Design of Integr. Circuits Syst.*, vol. 25, no. 6, pp. 1024–1037, Jun. 2006.

[19] S. Murali, A. Mutapcic, D. Atienza, R. Gupta, S. Boyd, and G. De Micheli, "Temperature-aware processor frequency assignment for MP-SoCs using convex optimization," in *Proc. CODES/ISSS 2007*, Oct. 2007, pp. 111–116.

[20] S. Murali, A. Mutapcic, D. Atienza, R. Gupta, S. Boyd, L. Benini, and G. De Micheli, "Temperature control of high-performance multi-core platforms using convex optimization," in *Proc. DATE 2008*, Mar. 2008, pp. 110–115.

[21] R. Darveaux, I. Turlik, L.-T. Hwang, and A. Reisman, "Thermal stress analysis of a multichip package design," *IEEE Trans. Compon., Hybrids, Manuf. Technol.*, vol. 12, no. 4, pp. 663–672, Dec. 1989.

[22] P. Chen, M.-C. Shie, Z.-Y. Zheng, Z.-F. Zheng, and C.-Y. Chu, "A fully digital time-domain smart temperature sensor realized with 140 FPGA logic elements," *IEEE Trans. Circuits Syst I: Reg. Papers*, vol. 54, no. 12, pp. 2661–2668, Dec. 2007.

[23] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge: Cambridge University Press, 2004.
[24] S. Wright, *Primal-Dual Interior-Point Methods*. Philadelphia: SIAM, 1997.
[25] J. Nocedal and S. Wright, *Numerical Optimization*, 2nd ed. New York: Springer-Verlag, 2006.
[26] S. Borkar, "Thousand core chips-a technology perspective," in *Proc. Des. Autom. Conf. DAC 2007*, Jun. 2007, pp. 746–749.
[27] S. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, P. Iyer, A. Singh, T. Jacob, S. Jain, S. Venkataraman, Y. Hoskote, and N. Borkar, "An 80-Tile 1.28 TFLOPS network-on-chip in 65 nm CMOS," in *Proc. Int. Solid-State Circuits Conference (ISSCC 2007)*, , pp. 98–589.
[28] T. Davis, *Direct Methods for Sparse Linear Systems*. Philadelphia: SIAM, 2006.
[29] Y. Ye, *Interior Point Algorithms: Theory and Analysis*. New York: John Wiley, 1997.
[30] E. Yildirim and S. Wright, "Warm-start strategies in interior-point methods for linear programming," *SIAM J.Optim.*, vol. 12, no. 3, pp. 782–810, 2002.
[31] J. Gondzio and A. Grothey, "Reoptimization with the primal-dual interior point method," *SIAM J. Optim.*, vol. 13, no. 3, pp. 842–864, 2003.
[32] G. B. Dantzig and P. Wolfe, "Decomposition principle for linear programs," *Oper. Res.*, vol. 8, pp. 101–111, 1960.
[33] N. Z. Shor, *Minimization Methods for Non-Differentiable Functions*. New York: Springer-Verlag, 1985.
[34] A. Nedić and A. Ozdaglar, "Approximate primal solutions and rate analysis for dual subgradient methods," MIT Press, Cambridge, MA, LIDS Tech. Rep. 2753, Mar. 2007.
[35] D. Bertsekas, *Nonlinear Programming*, 2nd ed. Belmont: Athena Scientific, 1999.
[36] Y. Nesterov, "Primal-dual subgradient methods for convex problems," CORE Discussion Paper 2005/67, 2005 [Online]. Available: www.core.ucl.ac.be/services/psfiles/dp05/dp2005_67.pdf
[37] A. Nedic, "Subgradient methods for convex minimization," Ph.D. dissertaion, MIT, Cambridge, 2002.
[38] B. Polyak, "Introduction to optimization," Optimization Software, Inc., New York, 1987.
[39] H. Sherali and G. Choi, "Recovery of primal solutions when using subgradient optimization methods to solve Lagrangian duals of linear programs," *Oper. Res. Lett.*, vol. 19, pp. 105–113, 1996.
[40] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*, ser. Applied Optimization. Boston, MA: Kluwer, 2003, vol. 87.
[41] M. Grant, S. Boyd, and Y. Ye, "CVX: Matlab Software for Disciplined Convex Programming, Version 1.1," Aug. 2007 [Online]. Available: www.stanford.edu/~boyd/cvx/
[42] O. Takahashi, S. Cottier, S. Dhong, B. Flachs, and J. Silberman, "Power-conscious design of the Cell processor's synergistic processor element," *IEEE Micro.*, vol. 25, no. 5, pp. 10–18, Sept./Oct. 2005.
[43] D. Pham, S. Asano, M. Bolliger, M. Day, H. Hofstee, C. Johns, J. Kahle, A. Kameyama, J. Keaty, Y. Masubuchi, M. Riley, D. Shippy, D. Stasiak, M. Suzuoki, M. Wang, J. Warnock, S. Weitzel, D. Wendel, T. Yamazaki, and K. Yazawa, "The design and implementation of a first-generation cell processor," in *Proc. IEEE Int. Solid-State Circuits Conf.*, Feb. 2005, pp. 184–185.
[44] A. Mutapcic, S. Murali, S. Boyd, R. Gupta, D. Atienza, and G. De Micheli, "Optimized slowdown in real-time task systems via geometric programming," Jul. 2007 [Online]. Available: www.stanford.edu/ boyd/papers/gp_task_slowdown.html
[45] L. Ljung, *System Identification: A Theory for the User*, 2nd ed. Englewood Cliffs, NJ: Prentice Hall, 1998.
[46] C.-H. Tsai and S.-M. Kang, "Cell-level placement for improving substrate thermal distribution," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 19, no. 2, pp. 253–266, Feb. 2000.
[47] B. Goplen and S. Sapatnekar, "Efficient thermal placement of standard cells in 3D ICs using a force directed approach," in *Proc. Int. Conf. Comput.-Aided Design (ICCAD 2003)*, Nov. 2003, pp. 86–89.
[48] Y. Zhan and S. Sapatnekar, "High-efficiency Green function-based thermal simulation algorithms," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 26, no. 9, pp. 1661–1675, Sep. 2007.
[49] O. Semenov, A. Vassighi, and M. Sachdev, "Impact of self-heating effect on long-term reliability and performance degradation in CMOS circuits," *IEEE Trans. Devices Mater.*, vol. 6, no. 1, pp. 17–27, Mar. 2006.
[50] A. Coskun, T. Rosing, K. Whisnant, and K. Gross, "Static and dynamic temperature-aware scheduling for multiprocessor SoCs," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 16, no. 9, pp. 1127–1140, Sep. 2008.
[51] T. Rosing, K. Mihic, and G. De Micheli, "Power and reliability management of SoCs," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 15, no. 4, pp. 391–403, Apr. 2007.

Almir Mutapcic received the B.S. degree in electrical engineering from the University of Missouri-Rolla, in 1999, and the M.S. and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA, in 2002 and 2008, respectively.

His current research interests include convex optimization and its applications in distributed and robust optimization.

Stephen P. Boyd (S'82–M'85–SM'97–F'99) received the A.B. degree (*summa cum laude*) in mathematics from Harvard University, Cambridge, MA, in 1980, and the Ph.D. degree in electrical engineering and computer science from the University of California, Berkeley, in 1985.

He is currently the Samsung Professor of Engineering, in the Information Systems Laboratory, Electrical Engineering Department, Stanford University, Stanford, CA. He is the author or a coauthor of *Linear Controller Design—Limits of Performance* (Prentice-Hall, 1991), *Linear Matrix Inequalities in System and Control Theory* (SIAM, 1994), and *Convex Optimization* (Cambridge University Press, 2004). His current research interests include convex programming applications in control, signal processing, and circuit design.

Prof. Boyd is a Distinguished Lecturer of the IEEE Control Systems Society. He received an Office of Naval Research Young Investigator Award, a Presidential Young Investigator Award, and the 1992 American Automatic Control Council (AACC) Donald P. Eckman Award. He has also received the Perrin Award for Outstanding Undergraduate Teaching in the School of Engineering and an ASSU Graduate Teaching Award. In 2003, he received the AACC Ragazzini Education Award.

Srinivasan Murali received the M.S. and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA, in 2007.

He is a research scientist at EPFL, Lausanne, Switzerland, and CTO of iNoCs, Lausanne, Switzerland. His research interests include design of Networks on Chips, thermal modeling and reliability of multi-core systems.

Dr. Murali has been actively involved in several conferences (such as DATE, CODES-ISSS, NoC symposium, VLSI-SoC) as program committee member/session chair and is a reviewer for many leading conferences and journals. He received a best paper award in the DATE 2005 conference and the EDAA outstanding dissertation award in 2007 for his work on interconnect architecture design. He has over 30 publications in leading conferences and journals.

David Atienza received the M.Sc. degree from Complutense University of Madrid (UCM), Madrid, Spain, in 2001, and the Ph.D. degree from the Inter-University Micro-Electronics Center (IMEC), Leuven, Belgium, in 2005, both in computer science.

He is currently a Professor and the Director of the Embedded Systems Laboratory, Ecole Polytechnique Federale de Lausanne, Lausanne, Switzerland. He is also an Adjunct Professor in the Computer Architecture and Automation Department, UCM. His current research interests include design methodologies high-performance embedded systems and systems-on-chip (SoC), including new thermal management techniques for multiprocessor SoCs, dynamic memory management and memory hierarchy optimizations for embedded systems, novel architectures for logic and memories in forthcoming nanoscale electronics, networks-on-chip interconnection design, and low-power design of embedded systems. He is the author or coauthor of more than 100 publications in prestigious journals and international conferences and an Associate Editor of Elsevier's *Integration: The VLSI Journal*.

Prof. Atienza has been an elected member of the Executive Committee of the IEEE Council of Electronic Design Automation since 2008. He is an Associate Editor in the area of system-level design for the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS.

Giovanni De Micheli (S'79–M'79–SM'89–F'94) is currently a Professor and the Director of the Institute of Electrical Engineering, and the Integrated Systems Centre, Ecole Polytechnique Federale (EPF), Lausanne, Switzerland. He was a Professor of electrical engineering at Stanford University. He is the author of *Synthesis and Optimization of Digital Circuits* (McGraw-Hill, 1994) and a coauthor and/or coeditor of six other books and more than 300 technical articles. His current research interests include several aspects of design technologies for integrated circuits and systems, such as synthesis, hardware/software codesign and low-power design, as well as systems on heterogeneous platforms, including electrical, micromechanical, and biological components.

Prof. De Micheli is a Fellow of the Association for the Computing Machinery. He has been serving the IEEE in several capacities, namely: the Division 1 Director (2008–2009), a Cofounder and the President-Elect of the IEEE Council on Electronic Design Automation (2005–2007), the President of the IEEE Circuits and Systems (CAS) Society (2003), the Editor in Chief of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS (1987–2001). He was the Program Chair of the pHealth and VLSI System-on-Chip conferences in 2006. He was the Program and the General Chair of the Design Automation Conference in 1996–1997 and 2000, respectively. He has also chaired the Scientific Committee of the Canadian Society of Endocrinology and Metabolism, Neuchatel, Switzerland. He was a recipient of the 2003 IEEE Emanuel Piore Award for contributions to computer-aided synthesis of digital systems.

Rajesh Gupta (S'83–M'85–SM'97–F'04) received the B.Tech. degree in electrical engineering from the Indian Institute of Technology, Kanpur, India, in 1984, the M.S. degree in electrical engineering and computer science from the University of California (UC), Berkeley, in 1986, and the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, in 1994.

He was a Circuit Designer at Intel Corporation, Santa Clara, CA, where he was a member of three successful processor design teams. He is also a member the Computer Science faculty at the University of Illinois, Urbana-Champaign, and the UC Irvine. He is currently a Professor and a Holder of the QUALCOMM Endowed Chair in Embedded Microsystems, in the Department of Computer Science and Engineering, UC, San Diego. His current research interests include energy efficient and mobile computing issues in embedded systems. He is author or coauthor of more than 150 articles on various aspects of embedded systems and design automation. He is the holder of four patents on phase-locked loop (PLL) design, data-path synthesis, and system-on-chip modeling.

Prof. Gupta is a Distinguished Lecturer for the Association for Computing Machinery's Special Interest Group on Design Automation (ACM/SIGDA) and the IEEE Circuits and Systems (CAS) Society. He was the Founding Chair of the ACM/IEEE Conference on Models and Methods in Codesign (MEMOCODE) and the Founding Cochair of the ACM/IEEE/International Federation of Information Processing (IFIP) Conference on Codesign and System Synthesis (CODES+ISSS). He is the Editor-in-Chief of the *IEEE Design and Test of Computers* and is a member of the Editorial Boards of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS and the IEEE TRANSACTIONS ON MOBILE COMPUTING. He is the Vice President of Publications of the IEEE Council on Electronic Design Automation (CEDA). He was a recipient of the Chancellor's Fellow at the UC Irvine, the UCI Chancellor's Award for excellence in undergraduate research, the National Science Foundation CAREER Award, two Departmental Achievement Awards, and a Components Research Team Award at Intel.