Overview of Continuous Optimization Advances and Applications to Circuit Tuning

Andrew R. Conn and Chandu Visweswariah
IBM Thomas J. Watson Research Center
Yorktown Heights, NY 10598
{arconn,chandu}@watson.ibm.com

ABSTRACT

This paper surveys the state-of-the-art in continuous nonlinear optimization and makes the case that due to tremendous recent progress, larger and more complex problems can be solved than previously thought possible. The two basic paradigms, trust-region and line-search methods, are briefly described. In addition, various nonlinear optimization techniques are reviewed. The application of these nonlinear optimization methods to circuit sizing is presented by describing a pair of circuit sizing tools, one for dynamic tuning and one for tuning based on static timing analysis. Particular emphasis has been given to the customization of nonlinear optimization to the circuit sizing application.

1. NONLINEAR OPTIMIZATION

This section provides an overview of the state-of-the-art in continuous nonlinear optimization. It attempts to convince the reader of three important points. The first is that nonlinear optimization has been revolutionized in the past thirty years and a good application must make use of the latest improvements, both in terms of optimization algorithms and in linear algebra advances. Second, regrettably, nonlinear optimization is sufficiently complex that for the near future, and possibly always, software packages cannot be used effectively for tuning if the solver is just treated as a black box. Third, whenever possible, derivatives or gradients must be used to solve large problems accurately and efficiently. Although today there are more effective derivative-free methods [15] than the over-popular Nelder and Meade algorithm [28], they are really only suitable for problems with less than 100 variables. Even then methods that use derivatives are vastly superior.

Due to space restrictions, this paper will confine itself to the basic ideas of nonlinear optimization, which are fortunately simple. Suitable references are provided for the details. Typically, nonlinear problems are approximated by linear functions for constraints that are kept explicit and by quadratics for the objective or merit function and the iterates use these approximations. Quadratics are desirable because the first-order conditions require solving for the (possibly projected) gradient to be zero and the gradient of a quadratic is linear. Not surprisingly, all fast methods in some sense approximate Newton's method, which is one reason why derivatives are essential.

There are two fundamental approaches, namely *line-search* and *trust-region* methods. In line-search methods, a suitable descent direction is computed and an attempt is made to reduce some merit function in that direction. In trust-region methods, a local model of the merit function is created and "trusted" within a certain neighborhood. The model is then (approximately) optimized within the region. At the new point, the merit function is computed and the reduction in the merit function is compared to the predicted reduction according to the model. If the agreement is good, the trust region is grown and the point accepted. If the agreement is mediocre, the point is accepted and the trust region radius maintained. If the agreement is poor, the trust-region is reduced in size and the new point rejected. An elegant proof can be applied to bound the trust-region radius away from zero. In essence, as the trust region gets smaller, continuity guarantees that even a linear model of the merit function is sufficiently accurate to model reality, and therefore the trust region must ultimately grow and so is bounded away from zero. The proof relies on accurate gradients being available. Useful general references for both these methods are [27], [19] and [29]. A comprehensive text on trust-region methods is [12].

In either paradigm, the merit function may incorporate a measure of feasibility as well as the objective function (*penalty methods*) or feasibility may be maintained explicitly. In *active-set* methods, the goal is to to keep some of the constraints exactly satisfied using characterizations of optimality to iteratively determine which constraints should be so maintained. The idea is to eventually identify the correct active set at the solution so that optimization in the reduced space solves the problem. If there are m active or tight constraints in n-space, this reduced space is of dimension $n - m$, since it is equivalent to using the equalities to reduce the effective dimensionality of the problem.

ISPD'01, April 1-4, 2001, Sibinia, California, USA.

1.1 Penalty methods and sequential quadratic programming

Consider the generic optimization problem

$$\begin{array}{ll} \min\limits_{x\in\mathbf{R}^n} & f(x) \\ \text{s.t.} & c(x)=0, \\ \text{s.t. simple bounds} & l\le x\le u, \end{array} \tag{1}$$

where $c(x)\in\mathbf{R}^m$. The objective function $f(x)$ and the constraint functions $c_i(x),\ i=1\cdots m$ are assumed to be smooth. This formulation is general since any inequality can be converted to an equality by the addition of a slack variable. The simple bounds are usually treated separately since they are so special. One method uses a quadratic penalty function to handle the general equalities. The objective function and the equality constraints are replaced by a single unconstrained *merit function*

$$\min_{x\in\mathbf{R}^n,\mu\in\mathbf{R}} \quad \phi(x,\mu) \tag{2}$$

where $\phi(x,\mu)\stackrel{\text{def}}{=} f(x)+\frac{1}{2\mu}\sum_1^m[c_i(x)]^2$ and μ is known as the *penalty parameter*. The theory [18] shows that in the limit as μ approaches zero through positive values, under reasonable conditions, a minimizer of ϕ is a minimizer of f subject to the equalities. Intuitively this can be appreciated by recognizing that as μ becomes small it is essential to minimize the squared *penalty term*, which forces feasibility, but as f remains in the picture there is still an incentive for it to be at its optimal feasible value. Now consider some additional linear equalities $Ax=b$. These linear equalities could be treated differently rather than squaring them and adding them to ϕ. Given a point x^k that satisfies $Ax^k=b$, we would like to determine a direction d that maintains feasibility on the linear constraints. Create a local quadratic approximation for the penalty function along the direction d and iterate on the *quadratic programming* problem,

$$\begin{array}{ll} \min\limits_{d\in\mathbf{R}^n} & \frac{1}{2}d^TH^kd+d^T\nabla_x\phi(x^k,\mu) \\ \text{s.t.} & Ad=0, \end{array} \tag{3}$$

where H^k approximates the Hessian of ϕ. Such an approach is an example of sequential quadratic programming and can be formulated in a line-search or trust-region context. For the latter, an l_∞ trust-region constraint could be added to (3). Linear inequalities can also be incorporated directly in the quadratic programming subproblem. Alternatively, nonlinear constraints may be accounted for by linearizing them and adding the linearizations to (3) while modifying the Hessian to account for the curvature of these nonlinear constraints.

1.2 An augmented Lagrangrian method

Most of our own experience applying optimization to circuit tuning has been with a trust-region method that uses an augmented Lagrangian merit function. We will explain this in some detail. First consider the unconstrained case. The minimum of the model along the steepest descent direction within the trust region is called the Cauchy point. Convergence from any start point is assured as long as each iterate does at least as well as this point. Eventually the trust region is irrelevantly large, which guarantees a fast asymptotic convergence rate as long as the underlying model optimization is suitably chosen, for example, a safeguarded Newton-like method.

The generalization to simple bounds is straightforward. For instance, define the trust region as $\|.\|\le\Delta$, where Δ is the trust-region radius. Assuming the l_∞ norm, the trust region is a box. The feasible region corresponding to simple bounds is also a box. The intersection of two conformally oriented boxes is a box. Define a generalized Cauchy point as the minimum along the *projected* gradient path within the trust region, where the projection is with respect to the simple bounds. As in the unconstrained case, convergence from any start point can be guaranteed, provided each iterate does at least as well as the generalized Cauchy point. In practice, better convergence, and ultimately a satisfactory asymptotic convergence rate, is obtained by further reducing the model function. At each iteration of the further reduction, a subset of the variables that are at their simple or trust-region-dictated bounds at the generalized Cauchy point is fixed, and the resulting reduced problem is (approximately) solved in the smaller subspace while respecting the remaining simple and trust-region bounds.

This is the trust region basis for the kernel algorithm SBMIN [9] of LANCELOT [11]. After converting general inequalities to equalities using slack variables the objective function and general constraints are combined into a composite function, the *augmented Lagrangian function*,

$$\Phi(x,\lambda,\mu)=f(x)+\sum_{i=1}^m\lambda_ic_i(x)+\frac{1}{2\mu}\sum_{i=1}^m c_i(x)^2, \tag{4}$$

where the components λ_i of the vector λ are known as *Lagrange multiplier estimates*.

The constrained minimization problem (1) is now solved by finding approximate minimizers of Φ for a carefully constructed sequence of Lagrange multiplier estimates and penalty parameters. Note that Φ is a combination of the usual Lagrangian and the quadratic penalty term introduced above. The significance of this combination is that with an appropriate choice of the multipliers, it is no longer necessary for μ to converge to zero. In fact by differentiating (4) with respect to x and comparing coefficients with the usual Lagrangian, the first-order update for the multipliers can be obtained as

$$\lambda_i^+=\lambda_i+c_i(x^k)/\mu. \tag{5}$$

In order to solve large problems it is essential to exploit structure and use iterative linear algebra techniques. LANCELOT exploits a more general form of structure than sparsity called *group partial separability* [25]. The basic idea consists of writing the functions as sums of *nonlinear element functions* with large invariant subspaces — sparsity is just the special case where the invariant subspaces are along elementary vectors. Typically the element functions inherently involve many fewer variables than the original problem. An extreme example would be a separable problem, in which case the separate nonlinear element functions would involve mutually disjoint subsets of variables.

Truncated preconditioned conjugate gradient iterations are used to (approximately) solve the trust-region subproblem in the reduced space determined by the activities at the generalized Cauchy point. At the new point, a function evaluation is carried out. Then the trust-region is updated and a new sub-problem solved, until sufficient stationarity is obtained. Thus for fixed μ and λs the simple bounds minimization algorithm SBMIN is employed to find a sufficiently stationary approximate minimizer of Φ constrained explicitly by the simple bounds. Once this is done, if the

original problem is sufficiently feasible, the multipliers are updated and the tolerances for feasibility and stationarity tightened. Otherwise the penalty parameter is reduced and the tolerances for feasibility and stationarity are reset. Overall convergence occurs when both sufficient stationarity of the augmented Lagrangian and sufficient feasibility of the original problem are obtained.

There are two major drawbacks of LANCELOT's augmented Lagrangian approach, which in turn serve as motivations for interior point methods which are discussed in the next section. The first disadvantage is the way linear constraints are handled. After they are incorporated into the function Φ they appear quadratically. Arguably, this is not a disadvantage for general nonlinear constraints, but in the case of linear constraints the structural advantages of linearity are lost. The second issue is the ability to deal with degeneracy, manifested either as redundant active bounds or non-unique Lagrange multipliers. In fact, circuit tuning problems based on static timing analysis are highly degenerate. For such problems LANCELOT has difficulty determining the correct active set. On the other hand, since interior point methods approach a solution from the interior of the feasible region, it is reasonable to assume that they are less prone to difficulties posed by identification of properties at the boundary. Although this advantage of interior point methods seems logical, there has not been enough experience with large, difficult nonlinear problems to make a definitive conclusion.

1.3 Interior point methods

It is convenient to consider a particular case of (1), namely

$$\begin{array}{ll} \min\limits_{x \in \mathbf{R}^n} & f(x) \\ \text{s.t. positivity bounds} & x \geq 0. \end{array} \tag{6}$$

Consider a different kind of penalty function that places a *barrier* around the feasible region and so assumes a feasible start point. In particular, consider the *logarithmic barrier function*

$$\phi(x,\mu) \stackrel{\text{def}}{=} f(x) - \mu e^T log(x) = f(x) - \mu \sum_{i=1}^{n} \log[x_i], \tag{7}$$

where $\mu > 0$ is called the *barrier parameter*. Once again the intuition is useful. As the boundary is approached and x approaches zero, the log term approaches negative infinity. Thus the minimization of ϕ prevents us from coming too close to the boundary. In most cases the solution lies at the boundary so it is not surprising that, as before, it is desirable for μ to approach zero (see [18] for the theory and [38] for a good overview).

Consider applying Newton's method to find a stationary point of ϕ. The quadratic model for the barrier term m expanded about x^k can be written as

$$m^k(x^k + d) = \mu_k \left(-e^T log(x^k) - e^T {X_k}^{-1} d + \tfrac{1}{2} d^T {X_k}^{-2} d \right), \tag{8}$$

where X_k indicates the diagonal matrix generated by the vector x^k with $[X_k]_{i,i} = x_i^k$. Then differentiating the merit function with respect to d gives, to first-order

$$\nabla_x f(x^{k+1}) - \mu_k X_{k+1}^{-1} e. \tag{9}$$

Define dual variables z by

$$XZ = \mu. \tag{10}$$

As long as $x \geq 0$ (primal feasibility) then (9) becomes the usual dual feasibility equation and (10) is just a perturbed form of the usual complementarity equation. These equations are the basis for so-called *primal-dual interior point methods* in which X^{-2} is replaced by ZX^{-1}/μ in (8). A detailed explanation is contained in [7, 23], where some excellent results are reported for general quadratic programming problems, including non-convex cases.

1.4 Filter methods

Penalty functions, including barrier methods, are one way to balance the two usually conflicting aims of staying feasible and optimizing the objective function. Recently, an approach that uses ideas from multicriteria optimization and avoids the often troublesome choice of penalty parameters has been proposed. A point is acceptable for the next iterate as long as it not dominated by a previous point, where a point is dominated if it is worse both from the point of view of the objective function *and* feasibility. Initial numerical experiments suggest that this *filter algorithm* can be implemented as an effective sequential quadratic method, but it is too soon to be sure. The idea of using a filter as an alternative to penalty functions was introduced in [21] and forms the basis of the package filterSQP. For a recent reference, including convergence results, see [20].

1.5 Linear algebra and other packages

The brief review above has paid scant attention to linear algebra aspects. This is not due to their lack of importance, but rather because a terse description is not too useful — the *details* are important. Chapters 4 and 5 of [12] include both details and references for much of the required linear algebra.

Besides those already mentioned, some additional pointers on software are included below. An excellent resource is `http://www-fp.mcs.anl.gov/otc/Guide/SoftwareGuide/`. Some additional specific packages include MINOS [26], a venerable active-set method that is still useful for problems where the reduced space is relatively low in dimension. It uses simplex-like methods (in the linear programming sense), with the nonlinear constraints linearized and the difference between this linearization and the true constraint incorporated into an augmented Lagrangian function. Another more modern sequential quadratic programming package is SNOPT [22]. An interior point approach for nonlinear optimization is LOQO [33].

2. LOOKING TO THE FUTURE

Because of the disadvantages of LANCELOT already outlined, we are currently planning a replacement for our optimization kernel. What we intend to have is a trust-region method based upon a combination of the methods described above. Linear constraints will be handled directly in the trust-region subproblem. General nonlinear inequalities and equalities will be handled as in LANCELOT. Simple bounds will be handled as in primal-dual interior point methods. One particular important detail is that, as in [7], the trust-region norm will be scaled to reflect the Hessian of the merit function. This scaling has significant advantages in handling the inevitable ill-conditioning of the full Hessian as the simple bounds approach tightness while the barrier parameter approaches zero.

There are several avenues of important ongoing research in

nonlinear optimization. Automatic differentiation [24] holds promise with the advent of efficient software to accumulate derivatives directly from computer code that provides function values (see `http://www.mcs.anl.gov/Projects/autodiff`). To solve practical engineering problems, non-asymptotic behavior, handling of noise and convergence in the presence of inaccurate function and gradient values [1, 8] will gain importance. The ability to handle large, noisy and poorly scaled or ill-conditioned problems will improve. Investigations of parallel implementations and the search for suitable preconditioners will continue. Attention will be paid to such difficult topics as nonlinear mixed integer and global optimization problems, and classes of problems in which matrix factorization is impossible.

3. APPLICATION TO CIRCUIT TUNING

It has already been mentioned that the optimization must be carefully integrated with and customized to the application. Almost all the theory in optimization is based upon asymptotic results. However, in circuit tuning, due to the high cost of function evaluation, and the inherent numerical noisiness of function and gradient data, the optimizer terminates before the asymptotics are seen, except on very small problems. Nevertheless, it is still important that algorithms with good asymptotic properties be employed. Therefore, most efficient applications imitate Newton's method, provided at least in theory their components have smooth second derivatives.

In the rest of this paper, the application of the nonlinear optimization methods described in the previous sections to circuit tuning will be described. Circuit tuning implies optimal sizing of transistors and wires. Automated sizing of circuits leads not only to better circuit performance, but also enhanced designer productivity. Two approaches to circuit tuning have been implemented: *dynamic* tuning based on a time-domain simulation of the underlying circuit, and *static* tuning based on transistor-level static timing analysis. The dynamic tuning implementation is called JiffyTune [4, 5] and uses SPECS [37, 17] for fast simulation and time-domain gradient computation, and LANCELOT for the nonlinear optimization. The static tuner is called EinsTuner [6, 34] and consists of the same two software components, SPECS and LANCELOT, and EinsTLT [31] for transistor-level timing.

In both tuners, the continuous nonlinear optimization software used has been heavily tailored to the application. Further, attention has been paid to the formulation of the problem, and efficient function and gradient computation in order to render the tuning of large circuits feasible. Most nonlinear optimization packages are benchmarked by their ability to solve analytic problems. In circuit sizing problems, however, function and gradient values are not available in analytic form; rather, they are computed by a simulator or timer. This difference has three important implications.

First, function and gradient evaluation dominates the run time, so we take all reasonable steps available to us to reduce the number of iterations. Therefore, the optimizer is tuned to aggressively take large steps whenever possible. In the early stages, when the optimizer is far from the solution, almost all modern optimization algorithms would not solve the inner problem accurately. Nevertheless, that is what we do in circuit tuning because it decreases the overall number of simulations required. The simulations are more expensive than the kernel solves, prompting us to make this tradeoff. Further, *two-step updating* [16] is employed to accelerate convergence. In two-step updating, variables are divided into two categories. The first is variables that appear in the merit function in a known functional form and whose contribution to the merit function is easy to compute. The second is variables that appear nonlinearly and typically require a simulation run to recompute the merit function when they change. Internal slack variables, for example, fall into the first category, while transistor widths are in the second category. The basic idea is that after the step proposed by the nonlinear optimizer, a *second step* is taken in the first category of variables to obtain further decrease in the merit function. Another way of looking at is is that we trust our model in the first set of variables infinitely, and allow them to wander outside the trust-region. With a small computational overhead, two-step updating reduces the total number of iterations and therefore the total number of expensive simulations required before convergence is obtained.

Second, function and gradient data from a numerical simulator are inherently noisy. Optimization in the presence of numerical noise is a little understood and difficult problem. In our implementations, noise considerations permeate several aspects such as choice of initializations, choice of stopping criteria and choice of tolerances. Once the step size falls below the noise threshold, optimization quality suffers and the tuning can quickly spiral to a halt; therefore many measures are applied to keep the step size from becoming prematurely small. Even the choice of an initial Hessian or trust-region radius is key. In an analytic problem, with a few additional iterations, a poor choice irons itself out. On a circuit tuning problem, however, if a poor initial choice causes the step size to be reduced below the noise level, recovery may be jeopardized! Many, but not all, of LANCELOT's tolerances that are related to machine precision have been revised upwards by several orders of magnitude to account for the noise level in simulation and gradient computation. However, tolerances for solving the inner quadratic minimization are untouched since once the quadratic model is formed, it does not depend on simulation values.

Third, the functions, such as delays, rise/fall times or power measurements, are only valid over a certain range of variables, unlike analytic functions. For example, if the optimizer makes a driver very small, but its fanout loads large, the output may not switch in a reasonable time! Unfortunately, tuning the optimizer to be aggressive often leads to such failures. Hence *failure recovery* is an essential ingredient of these tuners. Thus, the simulator detects "electrical failure" and signals the nonlinear optimizer to cut back on its step size. Such simplistic failure recovery works well in practice and allows the optimizer to be aggressive.

In addition to these customizations, the following two sections describe special techniques and algorithms employed in dynamic and static tuning, respectively.

4. DYNAMIC TUNING

Fig. 1 shows a high-level view of JiffyTune's software components [4, 5].

4.1 Formulation

JiffyTune formulates the circuit tuning problem in one of two ways. The first is a regular optimization problem:

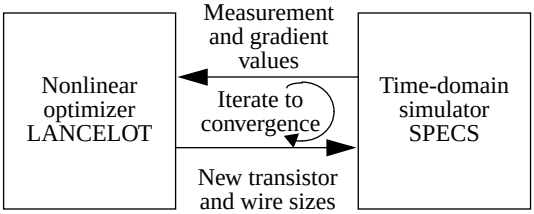


Figure 1: High-level view of JiffyTune.

$$\begin{aligned} \min \quad & m^o(x) \\ \text{s.t. } m_i^g(x) &\geq t_i^g, i = 1, 2, \cdots, G \\ \text{s.t. } m_i^l(x) &\leq t_i^l, i = 1, 2, \cdots, L \\ \text{s.t. } m_i^e(x) &= t_i^e, i = 1, 2, \cdots, E \\ \text{s.t. } x_i^l &\leq x_i \leq x_i^u, \end{aligned} \tag{11}$$

and the second is a minimax problem:

$$\begin{aligned} \min \quad & z \\ \text{s.t. } z &\geq m_i^m(x), i = 1, 2, \cdots, M \\ \text{s.t. } m_i^g(x) &\geq t_i^g, i = 1, 2, \cdots, G \\ \text{s.t. } m_i^l(x) &\leq t_1^l, i = 1, 2, \cdots, L \\ \text{s.t. } m_i^e(x) &= t_i^e, i = 1, 2, \cdots, E \\ \text{s.t. } x_i^l &\leq x_i \leq x_i^u, \end{aligned} \tag{12}$$

where the $m(x)$ are circuit measurements like delays, rise/fall times, area, power or noise measurements, and x are the independently tunable transistor and wire widths. Note that generally all measurements are assumed to depend on all transistor widths. In the case of minimax optimization, z is an auxiliary variable typically introduced to deal with the situation where we seek to minimize the worst of an enumerated set of measurements, usually path delays. In minimax mode, it is easy to prove from the optimality conditions that the Lagrange multipliers of the minimax constraints must sum to unity at the solution. So each of these multipliers is initialized to $1/M$ where M is the number of minimax constraints. Since z appears linearly in (12) and therefore quadratically in the merit function, this variable is amenable to two-step updating.

The designer provides a schematic and all the information necessary to run a SPICE-like simulation, including input patterns. Further, the user provides a well-thought-out statement of the optimization problem; tacit constraints will be exploited by the optimizer and the result may not be practically useful! Tunable parameters include transistor and wire sizes. These parameters can be "grouped" or "ratioed" to preserve layout or noise-related restrictions. The specification of the set of tunable parameters, their upper and lower bounds, the set of measurements, the objective function and constraints in terms of these measurements is all done via a graphical user interface in the Cadence schematic environment. When the tuning is completed, the final wire and transistor sizes, and the values of the measurements are graphically back-annotated onto the schematic.

When the optimization begins, the simulation is performed by SPECS and all measurement values, and gradients with respect to all tunable transistors are returned. Since "grouping" and "ratioing" between transistor widths are allowed, the gradients are chain-ruled and combined to obtain gradients with respect to the *independently* tunable parameters. These are fed to LANCELOT which comes up with a new set of parameter values for evaluation. This iterative procedure is repeated until convergence is obtained. Circuits with up to 20,000 transistors have been optimized by this method in an overnight run on a desktop engineering workstation.

4.2 Adjoint Lagrangian

Time-domain incremental gradient computation is a bottleneck in dynamic tuning. By using the adjoint method, gradients with respect to all tunable parameters are computed by means of a single adjoint analysis. However, each new measurement requires a new adjoint analysis. Instead, JiffyTune employs the adjoint Lagrangian method [14, 5] to obtain the gradient of LANCELOT's merit function with respect to all the tunable parameters. To do so, the adjoint circuit is appropriately excited at all measurement ports simultaneously. The value of each excitation depends on the contribution of the corresponding measurement to the merit function. Therefore tight coupling between LANCELOT and SPECS is required since the excitations depend on such optimizer quantities as the Lagrange multipliers, constraint values, scale factors and penalty parameter. Of course, the use of a scalar merit function by the optimizer, such as an augmented Lagrangian merit function, is a pre-requisite for the adjoint Lagrangian method.

Fig. 2 shows the growth of CPU time as a function of the number of measurements and parameters. In the direct method (Fig. 2a), the growth is small in the number of measurements, but significantly larger in the number of parameters, as indicated by the arrow. In the adjoint method (Fig. 2b), this dependence is reversed. Traditionally, programs heuristically switch between these two methods depending on the number of measurements and parameters (Fig. 2c). In the adjoint Lagrangian method, however, the growth of CPU time with respect to both the number of measurements and number of parameters is minimal (Fig. 2d).

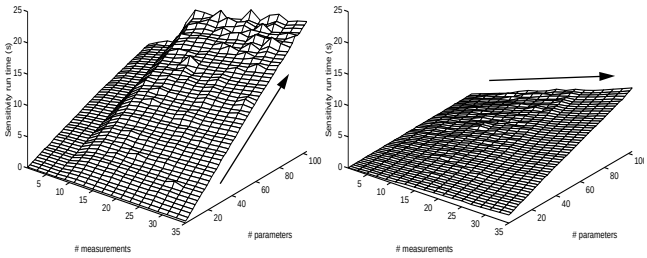
As a result of the adjoint Lagrangian method, the gradients of individual nonlinear elements are not available; rather, only the gradient of the scalar merit function is computed. So secant or quasi-Newton methods are required on the entire merit function to produce Hessian approximations. However, the special form in which slack variables appear in the merit function can be exploited to compute their contributions to the Hessian matrix. In JiffyTune, specialized symmetric rank one (SR1) formulas [10] are used so as to take advantage of this knowledge of certain elements of the Hessian [14, 5].

4.3 Noise measurements

A noise constraint in the time-domain can be expressed by a semi-infinite constraint of the form

$$v(x,t) \geq NM_H \quad \forall t \in [t_1, t_2], \tag{13}$$

where $v(x,t)$ is the noisy waveform, t_1 to t_2 is the time period of interest and NM_H is the high noise margin (see Fig. 3). A similar constraint can be written for a signal that is supposed to be at a logical zero in the absence of noise. From the point of view of any optimizer, semi-infinite constraints are difficult to handle directly. Fortunately, the semi-infinite constraint can be converted to an integral equal-



(a) Direct method (b) Adjoint method

(c) Heuristic choice of method (d) Adjoint Lagrangian formulation

Figure 2: CPU time of gradient computation.

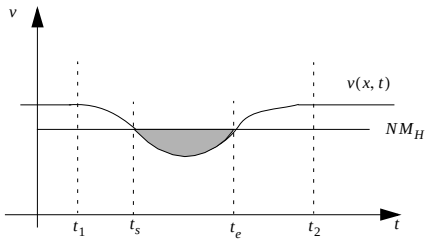


Figure 3: A noise violation.

ity constraint in the time-domain [13, 36] of the form

$$c(x) = \int_{t_1}^{t_2} \max\{NM_H - v(x,t), 0\} dt = 0 \qquad (14)$$

or

$$c(x) = \int_{t_s}^{t_e} \{NM_H - v(x,t)\} dt = 0. \qquad (15)$$

This is particularly appropriate since the integral form is easy to evaluate during nominal simulation and also lends itself well to adjoint gradient computations, where each measurement is required to be expressed as a convolution integral. By this simple remapping of the semi-infinite constraint, any number of time-domain noise constraints can be incorporated during circuit optimization.

5. STATIC TUNING

The advantages of dynamic tuning include accuracy, and the ability to handle noise and power considerations easily. The disadvantages are that input patterns are required, critical paths must be identified by the user and a carefully posed optimization problem is essential. Static tuning, on the other hand, is based on static timing analysis and overcomes these problems. All paths through the circuit are implicitly considered and no input patterns are necessary. A static tuner is therefore easier to use.

This section describes the application of nonlinear optimization in EinsTuner [6, 34], a static tuner that has been

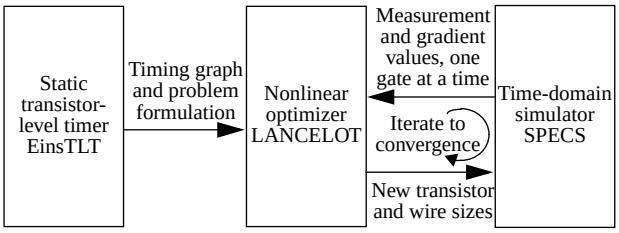


Figure 4: High-level view of EinsTuner.

applied to optimize high-performance macros on several different microprocessor chips. A high-level view of EinsTuner is shown in Fig. 4. Improvements of 15% or more have been realized on even previously hand-tuned circuits, at constant area and constant input loading.

5.1 Formulation

All static timers build a timing graph. Consider an edge in the timing graph from timing point 1 to timing point 2. The following assumes that separate timing points are created for falling and rising signals. For every such edge in the timing graph, two constrains are introduced [32, 2, 3]:

$$\begin{aligned} AT_2 &\geq AT_1 + d_{12}(w, s_1, c_{out}) \\ s_2 &\geq s_{12}(w, s_1, c_{out}), \end{aligned} \qquad (16)$$

where AT represents arrival time, s represents slew (rise/fall time), w is the vector of transistor widths in the block on which the delay/slew depend and c_{out} is the vector of fanout capacitances which in turn depends on downstream transistor widths. Note that the AT and s quantities are treated as variables of the problem, which is what saves us from exponential path enumeration. The quantities d_{ij} and s_{ij} are the computed delay and slew of the arc of the timing graph, respectively, which are nonlinear functions of the variables of the problem, and become LANCELOT's nonlinear element functions.

Three basic formulations are allowed in EinsTuner. In the first, the critical path delay is minimized subject to an area constraint. In this case, an auxiliary z variable is introduced as in the case of minimax optimization in JiffyTune, and z is constrained to be larger than the arrival time at all of the primary outputs. In the second mode, area is minimized subject to arrival time requirements on each of the primary outputs. In the third mode, a weighted sum of area and critical path delay is minimized.

In addition to arrival time, slew and area constraints, additional constraints are required to make the results of the tuning of practical use. As in the case of JiffyTune, "grouping" and "ratioing" between transistor widths are supported. Further, an upper bound on each rising or falling slew is allowed. Simple bounds on transistor widths are accommodated. The β ratio (ratio of P-strength to N-strength) of each gate is constrained to stay within certain bounds for noise considerations and to prevent a large mismatch between rising and falling slews. Finally, input loading constraints are posed so that the tuner does not grow the input stages to the detriment of the delay of the circuits driving these input stages.

5.2 Pruning

Formulating the arrival time and slew constraints as in (16) has far-reaching implications. Every node in the graph has

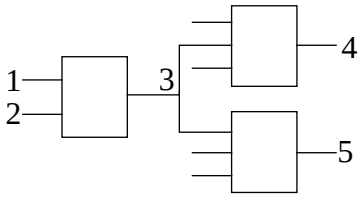


Figure 5: Illustration of pruning.

four variables, a rising and falling arrival time variable and a rising and falling slew variable. Thus, for even medium-sized circuits, the number of variables can be quite large. EinsTuner can presently solve optimization problems with about 20,000 transistors and it is not uncommon to face optimization problems with 50,000 variables and 50,000 constraints! The formulation inherently assumes that the largest incoming slew at each timing point is propagated downstream, a continuous choice, rather than the more popular but discontinuous propagation of the slew corresponding to the last arriving signal. At the solution, the arrival times along the critical path will be correct, but the values off the critical path will take one of several equally correct values. The off-critical arrival times in the conventional static timing analysis sense must be computed by running the timer after the optimization has converged. All off-critical arrival time and slew constraints have zero Lagrange multipliers at the solution and thus they lead to degeneracies or redundancies.

Consequently, the formulation is rife with degeneracies and redundancies. When these undesirable numerical properties are added to the size of the problem and the noisiness of the inherent function and gradient values, solving the problem becomes a challenge! The *pruning* [35] described in this section and *group partial separability* of the next section are crucial elements in the strategy to overcome these difficulties.

Pruning attempts to eliminate arrival time variables from the statement of the problem by listing all sub-paths that pass through a timing point but terminate on unpruned timing points. Referring to Fig. 5, the arrival time constraints before and after pruning AT_3 are:

$$\begin{array}{rcl} AT_3 & \geq & AT_1 + d_{13} \\ AT_3 & \geq & AT_2 + d_{23} \\ AT_4 & \geq & AT_3 + d_{34} \\ AT_5 & \geq & AT_3 + d_{35}, \end{array} \tag{17}$$

and

$$\begin{array}{rcl} AT_4 & \geq & AT_1 + d_{13} + d_{34} \\ AT_4 & \geq & AT_2 + d_{23} + d_{34} \\ AT_5 & \geq & AT_1 + d_{13} + d_{35} \\ AT_5 & \geq & AT_2 + d_{23} + d_{35}. \end{array} \tag{18}$$

By such simple topological manipulation carried out using an elementary graph theoretic approach, on average 90% of the arrival time variables and 40% of the arrival time constraints can be eliminated across a wide variety of circuits. This pruning not only reduces the size of the problem, but perhaps more importantly, reduces the degeneracies, making the problem much easier to solve. The repetition of the nonlinear elements d_{ij} is not a problem since they are computed only once per iteration and their Hessian approximations and gradients are processed only once per iteration.

5.3 Group partial separability

The delays, slews and β ratios are the nonlinear element functions of the optimization problem. Luckily, each of these depends on just a small subset of the total number of variables. For example, in a 2-input NAND gate, the delay from one input to the output depends on four transistor widths, one input slew and a few downstream transistor widths that constitute the load. So the nonlinear elements for this 2-input NAND gate depend on a small handful of variables even if the problem at hand has a total of 50,000 variables! This structure is exploited via *group partial separability* in LANCELOT. The gradients of each nonlinear element are computed just once in their respective smaller subspaces and the updating of the approximate Hessians exploits these subspaces. Then a quadratic model of the merit function is composed efficiently from these individual components. Indeed if every nonlinear element depended on every variable, we would not be able to solve such large problems.

5.4 Special circuits

While the formulation in (16) works for simple combinational circuits, care must be taken in problem formulation for sequential, dynamic and special kinds of circuits. In sequential circuits, tests such as setup and hold tests must be translated into corresponding constraints, and the arrival times that they reference must be specially handled so that they are not pruned. Similarly, dynamic circuits have special timing requirements that must be expressed explicitly to the optimizer. In a pass-gate MUX with one-hot select lines, for example, a falling select line will not cause the output to switch. Thus, in a single-input-switching model of state analysis [31], there will be no timing arc from the select line falling to the output. The tuner would therefore be emboldened to size transistors so that the arrival time and slew of the falling select line ends up being unacceptably large. The tacit assumption that a select line that is high must fall before any other rises must be expressed to the optimizer to obtain meaningful results.

5.5 Physical design

A circuit optimization package is only as good as the design methodology into which it fits. Tuning each transistor independently may produce the best optimization result, but then each cell will have a unique layout, leading to a huge physical design burden. One way in which we alleviate this problem is to exploit the grouping allowed by the tuners so that all instances of a given cell are tuned together; thus the hierarchy of the design is preserved and the number of unique layouts is manageable. Another method that we commonly apply is to use automated physical design in conjunction with a parameterized library and semi-custom design flow [30]. Each library cell has two parameters to control the width of all the NFETs and all the PFETs, respectively. The parameters are tuned as continuous variables. The library offers a rich selection of β ratios, taper ratios and low threshold voltage devices. When the optimization is completed, the parameterized cell layouts are automatically generated. The parameterization helps reduce the data volume, while preserving high design quality; thus, circuits that are close to custom circuits in quality are produced in turnaround times that are close to those of synthesized flows.

6. CONCLUSIONS

The fields of nonlinear optimization, transistor-level static timing analysis, fast time-domain simulation and gradient computation have seen tremendous recent progress. It is our belief that continuous nonlinear optimization has many as yet untapped applications in VLSI design.

The advances have made possible transistor-level dynamic and static optimization. In both cases, the optimizer has to be carefully customized to the application and the formulation of the problem must be carried out in such a manner that the optimizer can solve the large, noisy and often poorly conditioned problems that result. In this paper, JiffyTune and EinsTuner were presented as a pair of circuit tuning tools that can obtain optimal sizing results. They have both been used on several generations of high-performance PowerPC and S/390 microprocessor circuits. In addition to productivity increases, they have led to 15% or more improvement in circuit performance.

7. ACKNOWLEDGMENTS

The authors would like to thank the entire JiffyTune and EinsTuner teams for their invaluable contributions to this work.

8. REFERENCES

[1] R. G. Carter. On the global convergence of trust-region methods using inexact gradient information. *SIAM Journal on Numerical Analysis*, 28(1):251–265, 1991.

[2] C.-P. Chen, C. C. N. Chu, and D. F. Wong. Fast and exact simultaneous gate and wire sizing by Lagrangian Relaxation. *IEEE International Conference on Computer-Aided Design*, pages 617–624, November 1998.

[3] C.-P. Chen, C. N. Chu, and D. F. Wong. Fast and exact simultaneous gate and wire sizing by Lagrangian Relaxation. *IEEE Transactions on Computer-Aided Design of ICs and Systems*, 18(7), July 1999.

[4] A. R. Conn, P. K. Coulman, R. A. Haring, G. L. Morrill, and C. Visweswariah. Optimization of custom MOS circuits by transistor sizing. *IEEE International Conference on Computer-Aided Design*, pages 174–180, November 1996.

[5] A. R. Conn, P. K. Coulman, R. A. Haring, G. L. Morrill, C. Visweswariah, and C. W. Wu. JiffyTune: circuit optimization using time-domain sensitivities. *IEEE Transactions on Computer-Aided Design of ICs and Systems*, 17(12):1292–1309, December 1998.

[6] A. R. Conn, I. M. Elfadel, W. W. Molzen, Jr., P. R. O'Brien, P. N. Strenski, C. Visweswariah, and C. B. Whan. Gradient-based optimization of custom circuits using a static-timing formulation. *Proc. 1999 Design Automation Conference*, pages 452–459, June 1999.

[7] A. R. Conn, N. I. M. Gould, D. Orban, and Ph. L. Toint. A primal-dual trust-region algorithm for minimizing a non-convex function subject to bound and linear equality constraints. *Mathematical Programming, Series B*, 87(2):215–249, 2000.

[8] A. R. Conn, N. I. M. Gould, A. Sartenaer, and Ph. L. Toint. Global convergence of a class of trust region algorithms for optimization using inexact projections on convex constraints. *SIAM Journal on Optimization*, 3(1):164–221, 1993.

[9] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. Global convergence of a class of trust region algorithms for optimization with simple bounds. *SIAM Journal on Numerical Analysis*, 25:433–460, 1988. See also same journal, 26:764–767, 1989.

[10] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. Convergence of quasi-Newton matrices generated by the symmetric rank one update. *Mathematical Programming*, 50(2):177–196, 1991.

[11] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. LANCELOT*: a Fortran package for large-scale nonlinear optimization (Release A)*, volume 17 of *Springer Series in Computational Mathematics*. Springer Verlag, Heidelberg, Berlin, New York, 1992.

[12] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. *Trust-Region Methods*, volume 1 of *MPS/SIAM Series on Optimization*. SIAM and MPS, Philadelphia, USA, 2000.

[13] A. R. Conn, R. A. Haring, and C. Visweswariah. Noise considerations in circuit optimization. *IEEE International Conference on Computer-Aided Design*, pages 220–227, November 1998.

[14] A. R. Conn, R. A. Haring, C. Visweswariah, and C. W. Wu. Circuit optimization via adjoint Lagrangians. *IEEE International Conference on Computer-Aided Design*, pages 281–288, November 1997.

[15] A. R. Conn, K. Scheinberg, and Ph. L. Toint. Recent progress in unconstrained nonlinear optimization without derivatives. *Mathematical Programming, Series B*, 79(3):397–414, 1997.

[16] A. R. Conn, L. N. Vicente, and C. Visweswariah. Two-step algorithms for nonlinear optimization with structured applications. *SIAM Journal on Optimization*, 9(4):924–947, September 1999.

[17] P. Feldmann, T. V. Nguyen, S. W. Director, and R. A. Rohrer. Sensitivity computation in piecewise approximate circuit simulation. *IEEE Transactions on Computer-Aided Design of ICs and Systems*, 10(2):171–183, February 1991.

[18] A. V. Fiacco and G. P. McCormick. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. J. Wiley and Sons, New York, 1968. Reprinted as *Classics in Applied Mathematics 4*, SIAM, 1990.

[19] R. Fletcher. *Practical Methods of Optimization: Unconstrained Optimization*. J. Wiley and Sons, New York, 1980.

[20] R. Fletcher, N. I. M. Gould, S. Leyffer, and Ph. L. Toint. Global convergence of trust-region SQP-filter algorithms for nonlinear programming. Technical Report 99/03, Department of Mathematics, FUNDP, Namur, Belgium, 1999.

[21] R. Fletcher and S. Leyffer. Nonlinear programming without a penalty function. Technical Report Technical Report NA/171, Department of Mathematics, University of Dundee, Dundee, UK, 1997.

[22] P. E. Gill, W. Murray, and M. A. Saunders. SNOPT 5.3 USER'S GUIDE. Technical Report NA97-5, Department of Mathematics, University of California, San Diego, USA, 1997.

[23] N. I. M. Gould and Ph. L. Toint. Modern methods for non-convex quadratic programming. Technical Report RAL-TR-2001-009, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 2001.

[24] A. Griewank. On automatic differentiation. In M. Iri and K. Tanabe, editors, *Mathematical Programming: recent developments and applications*, pages 83–108. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1989.

[25] A. Griewank and Ph. L. Toint. On the unconstrained optimization of partially separable functions. In M. J. D. Powell, editor, *Nonlinear Optimization 1981*, pages 301–312. Academic Press, London and New York, 1982.

[26] B. A. Murtagh and M. A. Saunders. MINOS 5.4 USER'S GUIDE (revised). Technical Report SOL 83-20R, Department of Operations Research, Stanford University, Stanford, CA 94305, USA, 1993. Revised 1995.

[27] S. G. Nash and A. Sofer. *Linear and nonlinear programming*. Prentice-Hall, Englewood Cliffs, NJ, 1996.

[28] J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

[29] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Verlag, Heidelberg, Berlin, New York, 1999.

[30] G. A. Northrop and P.-F. Lu. A semi-custom design flow in high-performance microprocessor design. *Proc. 2001 Design Automation Conference*, June 2001. To be published.

[31] V. B. Rao, J. P. Soreff, T. B. Brodnax, and R. E. Mains. EinsTLT: transistor level timing with EinsTimer. *Proc. TAU*, December 1999.

[32] A. Srinivasan, K. Chaudhary, and E. S. Kuh. RITUAL: A performance driven placement algorithm for small cell ICs. *IEEE International Conference on Computer-Aided Design*, pages 48–51, November 1991.

[33] R. J. Vanderbei. LOQO user's manual — version 3.10. *Optimization Methods and Software*, 12:485–514, 1999.

[34] C. Visweswariah. Formal static optimization of high-performance digital circuits. *Proc. TAU*, page 51, December 2000.

[35] C. Visweswariah and A. R. Conn. Formulation of static circuit optimization with reduced size, degeneracy and redundancy by timing graph manipulation. *IEEE International Conference on Computer-Aided Design*, pages 244–251, November 1999.

[36] C. Visweswariah, R. A. Haring, and A. R. Conn. Noise considerations in circuit optimization. *IEEE Transactions on Computer-Aided Design of ICs and Systems*, 19(6):679–690, June 2000.

[37] C. Visweswariah and R. A. Rohrer. Piecewise approximate circuit simulation. *IEEE Transactions on Computer-Aided Design of ICs and Systems*, 10(7):861–870, July 1991.

[38] M. H. Wright. Interior methods for constrained optimization. In A. Iserles, editor, *Acta Numerica*, volume 1, pages 341–407. Cambridge Univesity Press, New York, 1992.