`lsgs`: Large-scale Gate Sizing MATLAB Toolbox

Version 0.29

Users' Guide

Siddharth Joshi Stephen Boyd

Electrical Engineering Department
Stanford University

sidj@stanford.edu boyd@stanford.edu

March, 2008

The large-scale gate sizing (`lsgs`) toolbox provides an implementation of the algorithm described in the paper "An efficient method for large-scale gate sizing", S. Joshi and S. Boyd [JB]. We will use the notation described in [JB]. The algorithm solves the gate sizing optimization problem

$$\begin{array}{ll} \text{minimize} & a^T x \\ \text{subject to} & u_i \geq (g_i + \sum_{j \in \mathrm{FO}(i)} F_{ij} x_j)/x_i, \quad i = 1, \ldots, n \\ & t_i - t_j - d_i^{\min} \geq u_i, \quad j \in \mathrm{FI}(i), \quad i = 1, \ldots, n \\ & t_j = T, \quad j \in \mathrm{PO} \\ & x_i \geq 1, \quad i = 1, \ldots, n, \end{array} \tag{1}$$

where the variables are $x \in \mathbf{R}^n$, $u \in \mathbf{R}^n$, and $t \in \mathbf{R}^n$; and the problem parameters are $a \in \mathbf{R}^n$, $g \in \mathbf{R}^n$, $d^{\min} \in \mathbf{R}^n$, $F \in \mathbf{R}^{n \times n}$, and $T \in \mathbf{R}$. The sets $\mathrm{FI}(i)$, the *fan-in* of a gate i, and $\mathrm{FO}(i)$, the *fan-out* of gate i, are

$$\mathrm{FI}(i) = \{j \mid F_{ji} \neq 0\}, \qquad \mathrm{FO}(i) = \{j \mid F_{ij} \neq 0\}, \qquad i = 1, \ldots, n.$$

The set of *primary output gates* PO is $\mathrm{PO} = \{j \mid \mathrm{FO}(j) = \emptyset\}$.

The problem (1) is a geometric program, and can be transformed to a convex optimization problem. Using standard convex optimization or geometric programming solvers the optimization problem (1) can be solved (globally) optimally for problems with n up to 10^3 in a reasonable time. The toolbox provides an implementation of the customized method described in [JB], which exploits the particular structure of the problem (1). The customized method is far faster and can scale up to problem with $n = 10^6$ or more. For a complete description of the problem (1) and the customized method see [JB].

Setup. To set up the toolbox, run (in MATLAB) the `setup` command from the directory `$BASE = $DD/lsgs-<ver>`, where `$DD` is the directory the toolbox has been extracted to, and `<ver>` is the version number (0.29 in this case). It compiles the C files in `$BASE/src/mex` and puts the MEX files in `$BASE/lib`. Then it adds the directories `$BASE/src` and `$BASE/lib` to the MATLAB path.

Once the MEX files have been built, you can test `lsgs` by running the scripts `sizeiscas85c17.m` and `size10k.m`, which are in the `$BASE/examples` directory.

Usage. The simplest usage is

```
lsgs(a, g, F, dmin)
```

which does no gate sizing, but prints a summary of the circuit to the output. The summary includes the number of gates, interconnections, primary outputs, and primary inputs, and the circuit delay when all gates have minimum size, and the minimum possible circuit delay. These circuit delays are useful in deciding on an appropriate timing specification for the circuit (which should be between these two).

The simplest usage to size a circuit is

```
x = lsgs(a, g, F, dmin, T)
```

which returns the optimal gates sizes as the column vector `x`.

The usage with all the input and output arguments is

```
[x, t, d, cumpcgiters, area, areasoft] = lsgs(a, g, F, dmin, ...
                                        T, x0, quiet, MAXCUMPCGITERS)
```

Input arguments. The arguments `a`, `g`, `F`, and `dmin` are mandatory; the others are optional. The inputs `a`, `dmin`, `g`, corresponding to a, $d^{\min}$, and g, respectively, are column vectors of size n. The input `F`, corresponding to F, is a strictly upper triangular sparse matrix of size $n \times n$. All the entries in `a`, `g`, `F`, and `dmin` should be nonnegative.

There are two ways to give the timing specification T. The simplest is to simple pass in `T`. The alternative is to pass in `x0`, a column vector of size n that gives the initial gate sizes. In this case, `lsgs` first computes the delay of the circuit with the given gate sizes, and uses this value for T. Thus

```
x = lsgs(a, g, F, dmin, [], x0)
```

computes the gate sizes that minimize the area, and has the same circuit delay as the original gate sizes. If both `T` and `x0` are specified, `x0` is ignored.

The Boolean argument `quiet` suppresses printing of data giving algorithm progress during gate sizing. The default value of `quiet` is true, expect in the case when `lsgs` does no gate sizing but only outputs the circuit summary.

The argument `MAXCUMPCGITERS` specifies the maximum number of cumulative PCG iterations to be carried out. Its default value is 500.

Output arguments. The arguments `x`, `t`, and `d`, correspond to the values of the size x, the timing assignment t, and the gate delays $u + d^{\min}$, respectively, when the algorithm is terminated. The arguments `cumpcgiters`, `area`, and `areasoft` are column vectors with size equal to number of iterations of the algorithm, with the ith entry containing the cumulative PCG iterations, the objective value of the problem (1), and the area achieved by the smooth approximation, respectively, at the ith iteration of the algorithm.

Examples. In the directory `$BASE/examples`, the file `sizeiscas85c17.m` define the data for the ISCAS85 c17 circuit, and invokes the `lsgs` function to size it. A larger circuit, consisting of 10000 gates, is defined in `ckt10k.mat`. The file `size10k.m` uses `lsgs` to size the 10000 gate circuit for five timing specifications. (This script should run in a minute or two.)

Note. The toolbox was tested with MATLAB Version 7.5.0.338 (R2007b) installed on GNU/Linux platform, on both 32-bit and 64-bit architectures.

Copyright information

References

[JB] S. Joshi and S. Boyd. An efficient method for large-scale gate sizing. *To appear in IEEE Transactions on Circuits and Systems I: Regular Papers.* Available from: http://www.stanford.edu/~boyd/gatesizing.html.