

# Efficient Serial and Parallel Coordinate Descent Methods for Huge-Scale Truss Topology Design

Peter Richtárik and Martin Takáč (June 30, 2011)

**Abstract** In this work we propose solving *huge-scale* instances of the truss topology design problem with coordinate descent methods. We develop four efficient codes: *serial* and *parallel* implementations of *randomized* and *greedy* rules for the selection of the variable (potential bar) to be updated in the next iteration. Both serial methods enjoy an  $O(n/k)$  iteration complexity guarantee, where  $n$  is the number of potential bars and  $k$  the iteration counter. Our parallel implementations, written in CUDA and running on a graphical processing unit (GPU), are capable of speedups of up to two orders of magnitude when compared to their serial counterparts. Numerical experiments were performed on instances with up to 30 million potential bars.

## 1 Introduction

In the process of designing mechanical structures such as railroad bridges, airplanes or buildings, one faces the problem of designing a truss—a system of elastic bars of varying volumes with endpoints at nodes, which are usually given by a 2D/3D grid, and are either fixed (and cannot move) or free—capable of withstanding a specified vector of forces applied to the nodes in an “optimal way”. After the forces are applied, the structure deforms (bars are stretched, free nodes move) to an equilibrium position, storing potential energy (compliance). Trusses of smaller compliance are more rigid. The goal of Truss Topology Design (TTD) is for a given grid structure of nodes and a vector of forces acting on them to construct a truss of at most a given total volume having *minimum compliance*.

In recent years methods of mathematical programming have been applied to solve various formulations of the TTD problem. For instance, a single load TTD can be cast as a linear program [5], robust TTD as a semidefinite program [1], TTD with

---

Peter Richtárik

School of Mathematics, University of Edinburgh, UK, e-mail: Peter.Richtarik@ed.ac.uk

Martin Takáč

School of Mathematics, University of Edinburgh, UK, e-mail: Takac.MT@gmail.com

integer variables as an integer linear semidefinite program [3]. We refer to [9] for a comprehensive survey.

Our work is motivated by the need to solve *huge-scale* TTD problems using methods with *provable iteration complexity guarantees*. These are problems on which second order methods fail due to memory limitations and for which even the evaluation of the gradient is time-consuming. We argue that coordinate descent methods (CDM) are well suited for this task; these are some of the reasons:

1. CDMs in general have *low per-iteration memory and work requirements*, which is an essential prerequisite for any method harboring hopes to solve a huge-scale problem.
2. This effect is *extreme* for TTD problems under formulation (1), which we propose in this paper, due to *inherent super-sparsity* of matrix  $A$  caused by each bar being related to two 2D forces. Each column of matrix  $A$  will have at most 4 nonzeros and one iteration of a (non-greedy) CDM will hence only need  $O(1)$  *memory* and will perform  $O(1)$  *work*, independently of the dimension of the problem.
3. CDMs can be easily *parallelized*, achieving *huge speedups* when compared to serial implementations.

For alternative heuristic approaches to huge-scale TTD problem see [2, 10].

## 2 Problem formulation

Consider an  $r \times c$  grid of nodes,  $m$  being free (although due to space limitations we only work with 2D trusses in this paper, the results are applicable to the 3D case as well). We allow, potentially, all pairs of nodes to be joined by a bar, except we require that no two potential bars intersect at more than one point. By  $n$  we denote the number of potential bars,  $w_i \geq 0$  is the weight of bar  $i$ , the total weight of all bars cannot exceed 1. The collection of 2D forces (load) acting at the free nodes is represented by  $d \in \mathbf{R}^{2m}$ ; the collection of node displacement associated with bar  $i$  after the load is applied is denoted by  $a_i \in \mathbf{R}^{2m}$ .

The compliance of a truss with weights  $w$  under the load  $d$  is equal to  $\frac{1}{2}d^T v$ , where  $v$  is any solution of the equilibrium system  $\sum_i w_i a_i a_i^T v = d$ . The problem of *minimizing compliance* subject to  $\sum_i w_i = 1$  can be equivalently written as the linear program  $\max_v \{d^T v : |a_i^T v| \leq 1, i = 1, \dots, n\}$ , the dual of which is equivalent to  $\min_x \{\|x\|_1 : Ax = d\}$ , where  $A = [a_1, \dots, a_n]$ . For more detail about the construction above we refer to Section 1.3.5 in [5]. In [7] a gradient method is described for solving all the above problems simultaneously. In this paper we will work with a penalized (and scaled) version of the last problem:

$$\min_{x \in \mathbf{R}^n} \{\|Ax - d\|_2^2 + \lambda \|x\|_1\}, \quad \lambda > 0. \quad (1)$$

Although this problem is unconstrained, our methods work also in the case of simple lower and upper bounds on the variables  $x$ .

### 3 Iteration complexity of the serial methods

Consider now the problem

$$F^* \stackrel{\text{def}}{=} \min_{x \in \mathbf{R}^n} \{F(x) \equiv f(x) + g_1(x^{(1)}) + \dots + g_n(x^{(n)})\}, \quad (2)$$

where  $f$  and  $\{g_i\}$  satisfy these assumptions: (A1)  $f : \mathbf{R}^n \rightarrow \mathbf{R}$  is convex and has coordinate-wise Lipschitz gradient, uniformly in  $x$ , with constants  $L_1, \dots, L_n > 0$ , i.e.,  $|\nabla_i f(x) - \nabla_i f(x + te_i)| \leq L_i |t|$  for all  $i$  and  $x, x + te_i \in \text{dom} F$ , (A2) functions  $g_i : \mathbf{R} \rightarrow \mathbf{R} \cup \{+\infty\}$  are convex and closed. Problem (1) is a special case of (2) with

$$f(x) = \|Ax - d\|_2^2, \quad g_i(x^{(i)}) = \lambda |x^{(i)}|, \quad L_i = 2\|a_i\|_2^2, \quad (3)$$

in which case Step 4 of Algorithm 1 can be computed in closed form and is known as *soft-thresholding*.

---

#### Algorithm 1 Serial Coordinate Descent

---

- 1: Choose initial point  $x_0 \in \mathbf{R}^n$
  - 2: **for**  $k = 0, 1, \dots$  **repeat**
  - 3:   Choose  $i \in \{1, 2, \dots, n\}$  *randomly or greedily*
  - 4:    $t_i^* = \arg \min_{t \in \mathbf{R}} \nabla_i f(x_k) t + \frac{L_i}{2} t^2 + g_i(x_k^{(i)} + t)$
  - 5:    $x_{k+1} := x_k + t_i^* e_i$
- 

The following result gives iteration complexity guarantees for randomized and greedy version of Algorithm 1. The greedy selection rule in part (ii) coincides with the one in [4]; the authors do not, however, provide any complexity bounds.

**Theorem 1.** *Let  $f$  and  $\{g_i\}$  satisfy assumptions (A1), (A2) and choose  $x_0 \in \text{dom} F$  and  $0 < \varepsilon < F(x_0) - F^*$ . Further let  $C = \max\{R_L^2(x_0), F(x_0) - F^*\}$ , where  $R_L(x_0) = \max_x \max_{x^* \in X^*} \{\|x - x^*\|_L : F(x) \leq F(x_0)\}$ ,  $\|x\|_L = (\sum_{i=1}^n L_i(x^{(i)})^2)^{\frac{1}{2}}$  and  $X^*$  is the set of minimizers of (2).*

- (i) **Random:** *Choose  $0 < \rho < 1$  and consider Algorithm 1, where in Step 3 each coordinate is chosen with probability  $\frac{1}{n}$ . Then after  $K \geq \frac{2nC}{\varepsilon}(1 + \log \frac{1}{\rho}) + 2 - \frac{2nC}{F(x_0) - F^*}$  iterations we have  $\mathbf{P}[F(x_K) - F(x^*) \leq \varepsilon] \geq 1 - \rho$ .*
- (ii) **Greedy:** *Let  $f$  and  $\{g_i\}$  be as in (3) and consider Algorithm 1, where in Step 3 coordinate  $i$  is chosen greedily as  $i = \arg \max_{j \in \{1, \dots, n\}} \alpha_k(j)$ , where*

$$\alpha_k(j) = \begin{cases} \frac{L_j}{2}(t_j^*)^2 + \lambda x_k^{(j)} (\text{sign}(x_k^{(j)}) - \text{sign}(x_k^{(j)} + t_j^*)), & \text{if } x_k^{(j)} + t_j^* \neq 0, \\ \lambda |x_k^{(j)}| - \nabla_j f(x_k) t_j^* - \frac{L_j}{2} t_j^{*2}, & \text{otherwise.} \end{cases}$$

*Then after  $K \geq \frac{2nC}{\varepsilon} - \frac{2nC}{F(x_0) - F^*}$  iterations we get  $F(x_K) - F(x^*) \leq \varepsilon$ .*

*Proof.* (Rough Sketch) Statement (i) is identical to part (i) of Theorem 5 in [8]. Part (ii) can be proved in an analogous way using the fact that  $\alpha_k(j) = F(x_k) - F(x_k + t_j^* e_j)$  and  $F(x_k + t_i^* e_i) = \min_{t, i} F(x_k + te_i) \leq \frac{1}{n} \sum_j F(x_k + t_j^* e_j)$ .  $\square$

## 4 Numerical experiments

In this final section we numerically compare our serial methods (SR = Serial Random, SG = Serial Greedy) described in Section 3 with their GPU-accelerated variants (PR = Parallel Random, PG = Parallel Greedy). All algorithms were run on TTD instances of the form (1) with  $\lambda = 10^{-4}$ . Due to space limitations we need to limit our exposition to a sketch of two experiments only.

**Implementation details.** Our serial codes (SR, SG) were written in C++, the parallel ones (PR, PG) using a CUDA C/C++ compiler from NVIDIA. All experiments were performed on a system with Intel Xeon CPU X5650@2.67GHz (we have only used one out of the 6 cores) and 48GB DDR2 PC3-1066 6.4GT/s memory. We have used NVIDIA Tesla C2050 GPU device with 448 cores, peak performance of 1.03Tflops for single precision and 3GB GDDR5 RAM. PG was implemented using CUSPARSE and CUBLAS libraries; all linear algebra and the greedy selection rule were implemented in parallel. In case of PR we choose independently for each thread a random coordinate and perform a single iteration of Algorithm 1. Race conditions were avoided using atomic operations.

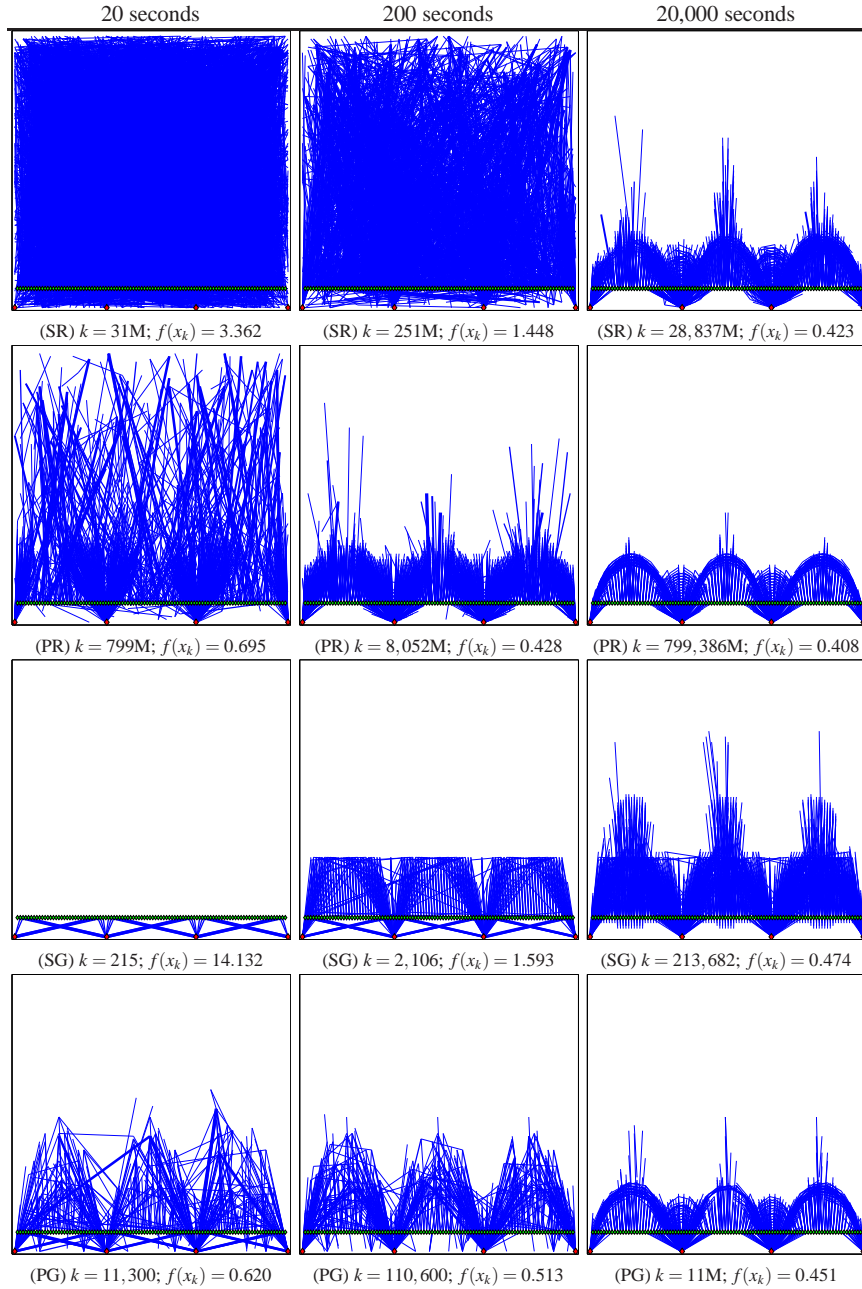
**Experiment 1.** In Table 1 the three `it/sec` columns illustrate how the number of iterations per second decreases with increasing problem size for methods SG, SR and NEST (Nesterov’s accelerated full-gradient method [6]).

$r \times c$	$2m$	$n$	$\ A\ _0 = 4n$	time (it) SeDuMi	it/sec SG	accel PG	it/sec SR	accel PR	it/sec NEST
20×20	800	48,934	195,736	61 (40)	5,101	0.6×	2.4M	28.3×	53.05
30×30	1,800	246,690	986,760	658 (10) NP	1,195	2.7×	2.2M	26.1×	9.01
40×40	3,200	779,074	3,116,296	8.6k (32) NP	380	8.0×	1.9M	24.9×	3.43
50×50	5,000	1,901,930	7,607,720	48.4k (4) NP	159	16.0×	1.5M	27.2×	1.41
80×80	12,800	12,454,678	49,818,712	X	26	42.3×	1.5M	34.0×	0.15
100×100	20,000	30,398,894	121,595,576	X	11	52.2×	1.2M	30.9×	0.05

**Table 1** Comparison of performance of SeDuMi, SG, PG, SR, PR and NEST on 6 TTD instances ( $A \in \mathbf{R}^{2m \times n}$ , NP=failure due to numerical problems, X=no iteration after 10 hours).

For instance, while  $n$  increases by a factor of 621 when going from the  $20 \times 20$  to the  $100 \times 100$  problem, the per-iteration speed of SR is merely halved (cf. with point 2 in the introduction). The `accel` columns indicate the acceleration achieved by parallelization. Note that the speedup increases with problem size for the greedy method (to  $52 \times$  for the largest problem) and stays virtually constant ( $\approx 30 \times$ ) for the randomized method. Iterations of any second order method (we have used SeDuMi (v1.21)) become prohibitively expensive as  $n$  increases. Indeed, for the  $80 \times 80$  problem SeDuMi is not able to perform a single iteration in 10 hours while PR does  $34 \times 1.5$  million iterations per second. Note that a similar, albeit much less pronounced, effect holds for NEST.

**Experiment 2.** In Figure 1 we run methods SR, PR, SG and PG on a  $100 \times 100$  “bridge” instance (with more than 30 million variables/potential bars) for 20, 200 and 20,000 seconds.



**Fig. 1** “Bridge” truss after 20s, 200s ( $\approx 3.34$  minutes) and 20,000s ( $\approx 5.56$  hours) of computation time for algorithms SR, PR, SG and PG (rows in this order). Number of iterations (“M” = millions) and objective value is shown under each plot.

There are 4 fixed nodes evenly spaced at the bottom (large red diamonds), and a unit downward force is applied at every node at height 7 (out of 100) above the “ground” (at the small green diamonds). We have used  $x_0 = 0$  for which  $f(x_0) = 49$ . Note that the parallel methods have managed to push the objective function down from 49 to 0.695 (PR) and 0.620 (PG) after 20 seconds already; despite the size of the problem. These trusses do not resemble a bridge yet, but after 5.56 hours the parallel methods do produce visibly bridge-like structures.

**Vanilla methods vs heuristics.** Note that while we have implemented our methods efficiently, further significant speedups are possible by introducing *heuristics*. For instance, note that in an optimal truss most of the potential bars will have zero weight. However, our methods, in the “vanilla” form appearing in this paper (this form enables us to prove rigorous iteration complexity bounds), will unnecessarily consider these zero weight bars in each iteration. Therefore, for instance in the case of SR and PR, decreasing the probability of selecting zero-weight bars will introduce a speedup (see *q-shrinking* strategy in [8]). We have not implemented this or any other heuristics in this work as our goal was to demonstrate that even vanilla methods are able to solve huge-scale problems. However, in solving any practical large TTD problem, we recommend introducing acceleration heuristics.

**Acknowledgements** The work of the first author was supported in part by EPSRC grant “Mathematics for vast digital resources” (EP/I017127/1); both authors were also supported in part by the Centre for Numerical Algorithms and Intelligent Software (funded by EPSRC grant EP/G036136/1 and the Scottish Funding Council).

## References

1. Ben-Tal, A., Nemirovski, A.: Robust truss topology design via semidefinite programming, *SIAM Journal on Optimization* **7**, 991–1016 (1997)
2. Gilbert M., Tyas A.: Layout optimization of large-scale pin-jointed frames, *Engineering Computations* **20** (8), 1044–1064 (2003)
3. Kočvara, M.: Truss topology design with integer variables made easy, *Opt. Online* (2010)
4. Li, Y., Osher, S.: Coordinate descent optimization for  $l_1$  minimization with application to compressed sensing; a greedy algorithm, *Inverse Problems and Imaging* **3**, 487–503 (2009)
5. Nemirovski, A., Ben-Tal, A.: *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications*. SIAM, Philadelphia, PA, USA (2001)
6. Nesterov, Yu.: Gradient methods for minimizing composite objective function. CORE Discussion Paper #2007/76 (2007)
7. Richtárik, P.: Simultaneously solving seven optimization problems in relative scale. *Optimization Online* (2009)
8. Richtárik, P., Takáč, M.: Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. Technical Report ERGO 11-011 (2011)
9. Rozvany, G., Bendsøe, M.P., Kirsch, U.: Layout optimization of structures. *Applied Mechanics Reviews*, **48** (2), 41–119 (1995)
10. Sokół, T.: Topology optimization of large-scale trusses using ground structure approach with selective subsets of active bars. Extended Abstract, *Computer Methods in Mechanics* (2011)