DESIGNING A NEW INHERENTLY PARALLEL ALGORITHM FOR UNCONSTRAINED OPTIMIZATION

Ş. İlker Birbil[*] & Figen Öztoprak[†]

[*]Sabancı University & [†]İstanbul Technical University

ICOTA 9
Taipei
December, 2013

Why Consider Parallel Algorithms?

"We have reached the limit of what is possible with one or more traditional, serial CPUs. It is past time for the computing industry–and everyone who relies on it– to take the leap into parallel processing."

Bill Dally, Chief scientist at NVIDIA*, Forbes, April, 2010*

A typical picture of resource usage when solving a nonlinear programming (NLP) problem on a 8 core machine.

```
Linux 2.6.32-32-server (mango) 06/27/2011 _x86_64_ (8 CPU)
12:40:11 PM  CPU    %usr   %nice   %sys   %soft   %steal    %idle
12:40:12 PM  all   10.54    0.00   0.00    0.00     0.00    89.46
12:40:12 PM    0    0.00    0.00   0.00    0.00     0.00   100.00
12:40:12 PM    1    0.00    0.00   0.00    0.00     0.00   100.00
12:40:12 PM    2    0.00    0.00   0.00    0.00     0.00   100.00
12:40:12 PM    3    0.00    0.00   0.00    0.00     0.00   100.00
12:40:12 PM    4    0.00    0.00   0.00    0.00     0.00   100.00
12:40:12 PM    5    0.00    0.00   0.00    0.00     0.00   100.00
12:40:12 PM    6    0.00    0.00   0.00    0.00     0.00   100.00
12:40:12 PM    7  100.00    0.00   0.00    0.00     0.00     0.00
```

Parallel Algorithms for Optimization

In 1995, R.B. Schnabel[1] from Colorado State University wrote:

"... the consideration of parallelism has not led to the development of exciting new general purpose optimization methods for small to medium size unconstrained optimization problems. Instead, the capabilities of parallel computers have best been utilized by parallelizing existing sequential algorithms in ways that do not affect the algorithms at the optimization level. "

"... once one considers large-scale problems, there are many opportunities for the development of interesting new parallel optimization algorithms, including possibilities that superior sequential methods may be discovered through this process."

[1] "A view of the limitations, opportunities, and challenges in parallel nonlinear optimization," *Parallel Computing*, 21:875-905, 1995.

How to Obtain Parallel NLP Algorithms?

1. Parallelize the costly operations of an existing algorithm
   - function / derivative evaluation procedures
   - linear algebra operations
2. Design a new algorithm (or extension) for parallel implementation
   - data level (generally requires a special problem structure)
   - control level

A Framework

1. Overall workload consists of blocks of tasks executed in parallel
2. Some of the tasks are included to provide additional information
3. Interaction among the tasks leads to information exchange

Ideally: use as few parameters as possible, minimize synchronization (scalability)

Take a look at the following page for the PhD thesis

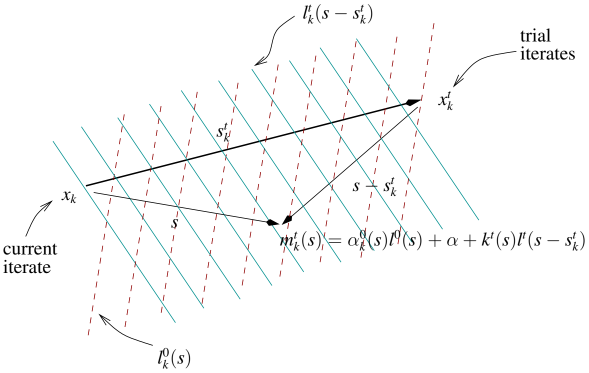
`http://people.sabanciuniv.edu/sibirbil/figen/`

and for a supplementary document about miserable failures, embarrassing ideas, bloopers and more.

AN ALGORITHM FOR UNCONSTRAINED NLP

$$\begin{array}{ll} \text{minimize} & f(x), \\ \text{subject to} & x \in \mathbb{R}^n \end{array}$$

- Use an extended model function m in a way that the computations can be done in independent subdomains
- m is based on the first order approximations at two reference points



The Model Function

$$m_k^t(s) = \alpha_k^0(s) l^0(s) + \alpha_k^t(s) l^t(s - s_k^t),$$

where

$$l_k^0(s) = f_k + g_k^T s, \qquad l_k^t(s - s_k^t) = f_k^t + (g_k^t)^T (s - s_k^t),$$

$$\alpha_k^0(s) = \frac{(s - s_k^t)^T(-s_k^t)}{(-s_k^t)^T(-s_k^t)}, \quad \text{and} \quad \alpha_k^t(s) = \frac{s^T s_k^t}{(s_k^t)^T s_k^t}.$$

We also require the following constraint to hold:

$$\|s\|^2 + \|s - s_k^t\|^2 \leq \|s_k^t\|^2.$$

Then, it is easy to prove that

$$\alpha_k^0(s) + \alpha_k^t(s) = 1, \quad \alpha_k^0(s) \geq 0, \alpha_k^t(s) \geq 0.$$

Simplifying the notation:

$$f_k := f(x_k), \quad g_k := \nabla f(x_k), \quad x_k^t := x_k + s_k^t,$$
$$f_k^t := f(x_k^t), \quad g_k^t := \nabla f(x_k^t), \quad y_k^t := g_k^t - g_k.$$

ALGORITHM

```
Algorithm 1: Outline
Input: $x_0$; $\rho \in (0,1)$; $k = 0$
while $x_k$ is not a stationary point do
    $t = 0$;
    Compute the first trial step $s_k^0$;
    while $f_k^t - f_k > \rho g_k^\intercal s_k^t$ do
        Compute the trial step $s_k^{t+1}$ by approximately solving
            $$\begin{array}{ll} \text{minimize} & m_k^t(s) = \alpha_k^0(s) l_k^0(s) + \alpha_k^t(s) l_k^t(s - s_k^t) \\ \text{subject to} & \|s\|^2 + \|s - s_k^t\|^2 \leq \|s_k^t\|^2. \end{array}$$
        t = t + 1;
    $x_{k+1} = x_k + s_k^t$;
```

HOW TO SOLVE THE SUBPROBLEM?

Two steps are crucial:

1. Tie the gradients of the model function and the original function.
2. Use regularization to control the sizes of the steps.

1. Relax the constraint, add it to the objective function and obtain

$$\min_{s \in \mathbb{R}^n} \{ m_k^t(s) + \sigma_1 (s - s_k^t)^\intercal (s - s_k^t) \}.$$

To make sure $\nabla m_k^t(0) = g_k$, set

$$\sigma_1 = \frac{1}{2\|s_k^t\|^2} (f_k^t - (g_k^t)^\intercal s_k^t - f_k).$$

How to Solve the Subproblem?

2. Add a regularization term to control the size of s and derive

$$\min_{s\in\mathbb{R}^n}\{\tfrac{1}{2}(f_k^t - (g_k^t)^\intercal s_k^t - f_k) + g_k^T s + s^\intercal \frac{1}{\|s_k^t\|^2}[\sigma I + s_k^t(y_k^t)^\intercal]s\},$$

where

$$\sigma = (\sigma_1 + \sigma_2)\|s_k^t\|^2 = \tfrac{1}{2}(f_k^t - (g_k^t)^\intercal s_k^t - f_k) + \sigma_2\|s_k^t\|^2.$$

In addition, guarantee for some $\eta \in (0,1)$ that

$$\|s_k^{t+1}\| \leq \eta\|s_k^t\|.$$

After some algebra, we obtain

$$\sigma_2 \geq \frac{1}{2\|s_k^t\|^2}\left(\|s_k^t\|\left(\|y_k^t\| + \frac{1}{\eta}\|g_k\|\right) + g_k^\intercal s_k^t - f_k^t + f_k\right).$$

How to Solve the Subproblem?

Analytical solution!

The optimal solution of the subproblem is given by

$$s_k^{t+1} = c_g(\sigma) g_k + c_y(\sigma) y_k^t + c_s(\sigma) s_k^t,$$

where

$$c_g(\sigma) = -\frac{\|s_k^t\|^2}{2\sigma}, \quad c_y(\sigma) = -\frac{\|s_k^t\|^2}{2\sigma\theta}[-((y_k^t)^\intercal s_k^t + 2\sigma)((s_k^t)^\intercal g_k) + \|s_k^t\|^2((y_k^t)^\intercal g_k)],$$

$$c_s(\sigma) = -\frac{\|s_k^t\|^2}{2\sigma\theta}[-((y_k^t)^\intercal s_k^t + 2\sigma)((y_k^t)^\intercal g_k) + \|y_k^t\|^2((s_k^t)^\intercal g_k)],$$

with

$$\theta = \left((y_k^t)^\intercal s_k^t + 2\sigma\right)^2 - \|s_k^t\|^2\|y_k^t\|^2,$$

and

$$\sigma = \tfrac{1}{2}\left(\|s_k^t\|\left(\|y_k^t\| + \frac{1}{\eta}\|g_k\|\right) - (y_k^t)^\intercal s_k^t\right).$$

PARALLEL ALGORITHM

```
Algorithm 2: Implementation
Input: $x_0$; $\rho \in (0,1)$; $k = 0$
while $x_k$ is not a stationary point do
    $t = 0$;
    Compute the first trial step $s_k^0$        (Parallel);
    $x_k^0 = x_k + s_k^0$        (Parallel);
    $\Delta = g_k^\intercal s_k^0$        (Parallel);
    for $t = 1, 2, \cdots$ do
        Compute $f_k^t$, $g_k^t$        (Parallel);
        if $f_k - f_k^t \geq -\rho\Delta$ then
            $x_{k+1} = x_k^t, f_{k+1} = f_k^t, g_{k+1} = g_k^t$;
            $k = k + 1$;
            break;
        Compute $v_i, i = 1, 2, \cdots, 6$        (Parallel, $O(n/p)$);
        Compute $c_g, c_s, c_y$        (Sequential, $O(1)$);
        $\Delta = c_g v_5 + c_y v_4 + c_s v_6$        (Sequential, $O(1)$);
        $s_k^{t+1} = c_g g_k + c_y y_k^t + c_s s_k^t$        (Parallel);
        $x_k^{t+1} = x_k^t + s_k^{t+1}$        (Parallel);
```

$$v_1 = (s_k^t)^\intercal y_k^t$$
$$v_2 = (s_k^t)^\intercal s_k^t$$
$$v_3 = (y_k^t)^\intercal y_k^t$$
$$v_4 = (y_k^t)^\intercal g_k$$
$$v_5 = g_k^\intercal g_k$$
$$v_6 = (s_k^t)^\intercal g_k$$

Convergence

Lemma

Suppose that the first trial step s_k^0 *satisfies* $m_0\|g_k\| \leq \|s_k^0\| \leq M_0\|g_k\|$ *and* $s_k^0 g_k \geq -\lambda_0\|g_k\|^2$ *for some* $m_0, M_0, \lambda_0 \in (0, \infty)$*. Then, at any iteration* k*, the optimal solution* s_k^{t+1} *becomes an acceptable step in finite number of inner iterations at a nonstationary point* x_k*.*

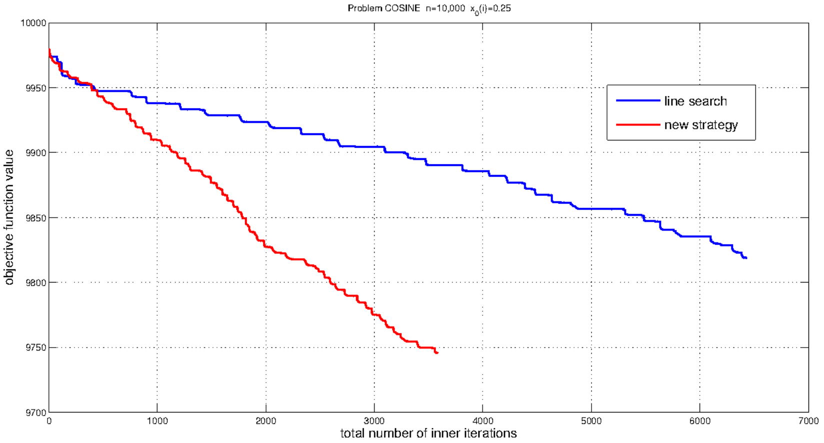
Theorem

Let $\{x_k\}$ *be the sequence of iterates generated by the proposed algorithm and* $\{s_k^0\}$ *satisfy the requirements in the lemma above. Then, any limit point of* $\{x_k\}$ *is a stationary point of the objective function* f*.*

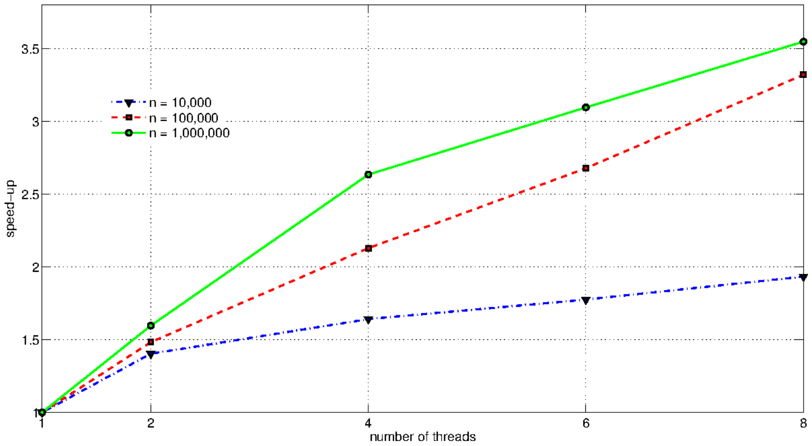
NUMERICAL EXPERIMENTS

- The specifications of the computing environment:
  - Programming language and libraries: C++ (gcc), Intel Threading Building Blocks (TBB)
  - Computer: 2 Quad-Core Intel Xeon CPUs @2.66GHz (8 cores)
  - Operating System: Ubuntu 10.04, 64-bit
- The problems are from the CUTEr collection

OBSERVATIONS

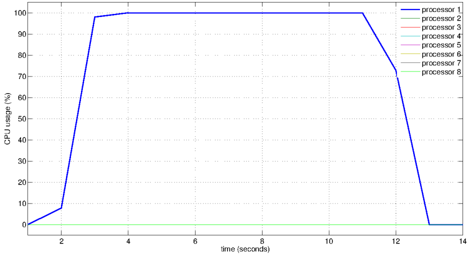


OBSERVATIONS

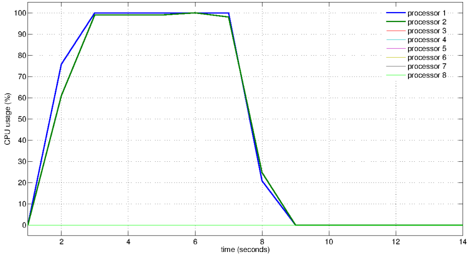


OBSERVATIONS

COSINE with $N = 1,000,000$



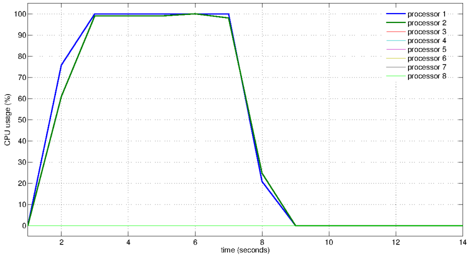
$p = 1$



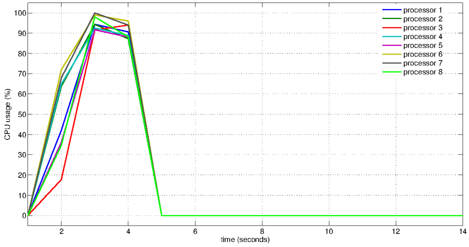
$p = 2$

OBSERVATIONS

COSINE with $N = 1,000,000$



$p = 2$



$p = 8$

CONCLUSION

- Parallel nonlinear programming algorithms better be designed from scratch
- One may invent a new sequential method while designing a parallel algorithm
- New unconstrained algorithm looks robust and scalable but further numerical experiments are required
- We already developed another version to solve bound constrained problems
- Numerous future research questions arise. To name a few nontechnical ones:
  - How about massively parallel environments?
  - Problem specific algorithms?
  - Recovering multiple local minima?