Global & Stochastic Optimization on metacomputing platforms [a]

G. Zakeri

Argonne National Labs./NWU

Feb 99

[a] metaNEOS project

Features of well suited applications

It is desirable that the applications be

- coarse grained
- need little communication and synchronization
- be able to cope with a dynamic pool of resources (e.g. an iterative algorithm should be "fault tolerant" to suit these platforms.)

There are a number of opimization applications that meet the above, including global, stochastic and structured optimization applications.

Global optimization in protein folding (distance geometry)

Protein atoms conform to a structure which is evinced to minimize a potential function which basically amounts to a function of the sum of the pairwise distances of the atoms involved.

An example of such function is:

$$f(x) = \sum_{i,j \in \mathcal{S}} P_{i,j}(x_i - x_j)$$

where

$$P_{i,j}(x) = \min\{\frac{||x||^2 - l_{i,j}^2}{l_{i,j}^2}, 0\} + \max\{\frac{||x||^2 - u_{i,j}^2}{l_{i,j}^2}, 0\}$$

In distance geometry problems, we are interested in finding the global optimizer of the above function f.

Smoothing solution technique

Moré and Wu developed a smoothing technique for solving distance geometry problems.

- start with a (highly) smoothed function (f_λ) which is obtained using a Gaussian transform with a “high enough” value of λ. This initial f_λ is expected to have only a few minimizers,
- they then find the minimizers of the smoothed function,
- then coarsen the approximation f_λ (by decreasing λ) and reoptimize starting from the previous minima.

Moré and Wu have also developed a package DGSOL that implements the above method.

DGSOL on Condor

- Original parallel code was written in MPI.
- We found out that we could not run an MPI application on Condor's standard universe due to the dynamic nature of the pool of resources.
- We changed the code to a task farm model so that each machine could simply call the executable itself.
- The largest job we've ran so far consisted of 12000 initial protein configurations as starting points and running DGSOL on each.

The verification problem in stochastic programming

In stochastic optimization, we are aiming to find

$$z^* = \min_{x \in X} Ef(x, \tilde{\xi})$$

with

$$x^* = \text{argmin }_{x \in X} f(x, \tilde{\xi}).$$

Often however, it is not possible to do this due to the very large or infinite distributions involved, so we solve an approximating problem:

$$z_n^* = \min \frac{1}{n} \sum_{i=1}^{n} f(x, \tilde{\xi}^i)$$

We now have to find out how good an approximation of z^* is z_n^*.

Mak, Morton and Wood Upper Bound

Suppose that we have an approximate solution $\hat{x} \in X$ for a stochastic program. We can estimate $Ef(\hat{x}, \tilde{\xi})$ via the standard sample mean estimator:

$$\overline{U}_n = \frac{1}{n} \sum_{i=1}^{n} f(\hat{x}, \tilde{\xi}^i)$$

($\tilde{\xi}^i$ are iid from $\tilde{\xi}$.) then

- $\overline{U}_n$ is an unbiased estimator
- We have a CLT satisfied by $\overline{U}_n$

$$n^{1/2}[\overline{U}_n - Ef(\hat{x}, \tilde{\xi})] \to \mathcal{N}(0, \sigma_u^2) \text{ as } n_s \to \infty$$

we estimate $\sigma_u = \text{var } f(\hat{x}, \tilde{\xi}$ by the sample variance s_u.

Lower Bound

$$E[\min \frac{1}{n}\sum_{i=1}^{n} f(x,\tilde{\xi}^{i})] \leq z^{*}$$

Now let $\tilde{\xi}^{i,1},\cdots,\tilde{\xi}^{i,n}$ be iid batches of random vactors and let $z_n^{*\,i} := \min_{x\in X} \frac{1}{n}\sum_{i=1}^{n} f(x,\tilde{\xi}^{i,j})$. Now define

$$\overline{L}(n_s) = \frac{1}{n_s}\sum_{i=1}^{n_s} z_n^{*\,i}$$

then $\overline{L}(n_s)$ will satisfy the CLT

$$n_s^{1/2}[\overline{L}(n_s) - Ez_n^{*}] \rightarrow \mathcal{N}(0,\sigma_l^2)$$

where $\sigma_l^2 = \mathrm{var}\ z_n^{*}$.

verification algorithm

- Given a candidate solution $\hat{x}$, we generate n_s iid samples drawn from $\tilde{\xi}$, each of size n.
- In order to generate our estimate of $\overline{U}(n_s)$ we have to solve a number of math programs.
- In order to generate out estimate of $\overline{L}(n_s)$ we have to solve n_s stochastic programs.
- From the above probabilistic lower and upper bounds, we can draw a confidence interval around the candidate solution.

This application is coarse grained, requires little communication and in the absence of apriori knowledge of appropriate n_s is perfect for a master/worker paradigm.