Testing fine-grained parallelism for the ADMM on a factor-graph

Ning Hao AmirReza Oghbaee Mohammad Rostami Nate Derbinsky José Bento

1. Problem

There is a shortage of not-problem-specific general-purpose optimization tools that can automatically exploit GPU parallelism.

2. Focus

The Alternating Direction Method of Multipliers (ADMM).

- deals with non-smooth functions;
- better bounds for global convergence rates and variants of the ADMM than for other first order methods such as Nesterov's and gradient descent [Lessard et al. 14; França and Bento 15];
- convergence guarantees for convex problems;
- breaks problems into a series of parallel computations.

3. Questions

Can we use the ADMM to automatically exploit parallelism in general optimization without having to write problem specific parallel code?

Do we get good speedup on a GPU?

Is there an advantage on using a GPU vs other parallel frameworks?

Our numerical experiments with three different domains say

YES

4. "Typical" ADMM

$$\begin{aligned}&\text{minimize } f(w_1) + g(w_2)\\ &\text{subject to } Aw_1 + Bw_2 = c\end{aligned}$$

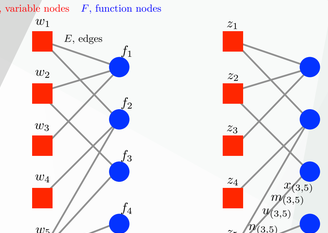
while !stopping criteria **do**
$x \leftarrow \arg\min_s f(s) + \frac{\rho}{2}\|As + Bz - c + u\|^2$
$z \leftarrow \arg\min_r g(r) + \frac{\rho}{2}\|Ax + Br - c + u\|^2$
$u \leftarrow u + Ax + Bz - c$
end while

It does not expose parallelism immediately although for specific problem it does become visible.

5. ADMM on factor-graph

$$\min_w \; f_1(w_1, w_2, w_3) + f_2(w_1, w_4, w_5) + f_3(w_2, w_5) + f_4(w_5)$$

V, variable nodes F, function nodes



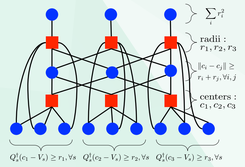
while !stopping criteria **do**
  for $a \in F$ **do** ▷ x-update
    $x_{(a,\partial a)} \leftarrow \mathbf{Prox}_{f_a,\rho_{(a,b)}}(n_{(a,b)})$
  end for
  for $(a,b) \in E$ **do** ▷ m-update
    $m_{(a,b)} \leftarrow x_{(a,b)} + u_{(a,b)}$
  end for
  for $b \in V$ **do** ▷ z-update
    $z_b \leftarrow \sum_{a\in\partial b} \rho_{(a,b)} m_{(a,b)} \Big/ \sum_{a\in\partial b} \rho_{(a,b)}$
  end for
  for $(a,b) \in E$ **do** ▷ u-update
    $u_{(a,b)} \leftarrow u_{(a,b)} + \alpha_{(a,b)}(x_{(a,b)} - z_b)$
  end for
  for $(a,b) \in E$ **do** ▷ n-update
    $n_{(a,b)} \leftarrow z_b - u_{(a,b)}$
  end for
end while

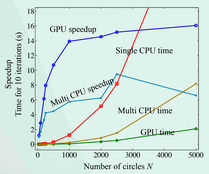
6. Testing fine-grained par.

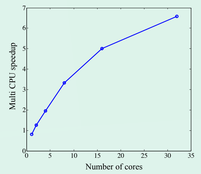
GPU = TeslaK40; CPU = 2.8GHz AMD;
Multi CPU-cores = up to 32 ×2.8GHz AMD.
Software: https://github.com/parADMM/

❖ Combinatorial optimization: What is the best way to pack N circles inside a triangle to maximize covered area?

$$\begin{aligned}&\text{minimize } \sum_{i=1}^{N} r_i^2\\ &\text{subject to}\\ &\|c_i - c_j\| \geq r_i + r_j \;\forall i,j\\ &Q_s^\dagger(c_i - V_s) \geq r_i \;\forall s,i\end{aligned}$$

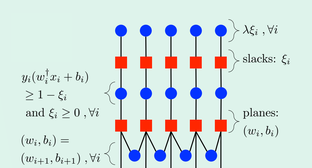


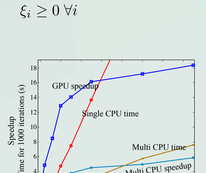


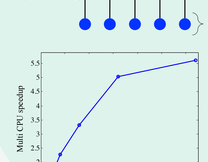


❖ Machine learning: What is the best separating hyper plane for N data points labeled +1 or -1?

$$\begin{aligned}&\text{minimize} \sum_{i=1}^{N} \frac{1}{2N}\|w_i\|^2 + \lambda\xi_i\\ &\text{subject to}\\ &w_i = w_{i+1}, b_i = b_{i+1} \;\forall i\\ &y_i(w_i^\dagger x_i + b_i) \geq 1 - \xi_i \;\forall i\\ &\xi_i \geq 0 \;\forall i\end{aligned}$$







❖ Optimal control: What is the best sequence of K inputs that filters perturbations?

$$\begin{aligned}&\text{minimize} \sum_{t=0}^{K} q^\dagger(t)Q_t q(t)\\ &\quad + u^\dagger(t)R_t u(t)\\ &\text{subject to}\\ &q(t+1) - q(t) = Aq(t) + Bu(t) \;\forall t\\ &q(0) = q_0\end{aligned}$$