Comment on “Efficient Secure Outsourcing of Large-Scale Sparse Linear Systems of Equations”

Zhengjun Cao and Olivier Markowitch

Abstract—The scheme [IEEE Trans. Big Data, 4 (1), 2018, 26-39] is flawed because the sparsity of coefficient matrix is neglected. In the discussed scenario, we find it is unnecessary to outsource such a problem because the client can solve it locally. Even if the matrix is not sparse, the proposed paradigm which requires a great number of interactions, is rarely adopted because of the possibly incurred massive communication cost.

Index Terms—Cloud computing, sparse linear equations, linear least squares problem, conjugate gradient method.

1 INTRODUCTION

Recently, Salinas et al. [1] have presented a scheme for secure outsourcing of large-scale sparse linear equations \(A\mathbf{x} = \mathbf{b}\), where \(A \in \mathbb{R}^{m \times n} (m \geq n)\) is a full rank coefficient matrix with \(M\) non-zero elements (\(n \leq M \leq mn\)), \(\mathbf{x} \in \mathbb{R}^{n \times 1}\) is the solution vector, and \(\mathbf{b} \in \mathbb{R}^{m \times 1}\) is the constant vector. Note that this is an overdetermined system if \(m > n\). In nature, they did consider the least squares problem \(A'\mathbf{x} = \mathbf{b}'\), where \(A' = A \mathbf{A}\), \(\mathbf{b}' = A \mathbf{b}\). The scheme makes use of the general conjugate gradient method to solve the equivalent quadratic program, \(\min f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top A'\mathbf{x} - (\mathbf{b}'^\top \mathbf{x})\), in the client-server scenario.

In this note, we show that the scheme is flawed because it neglected the sparsity of coefficient matrix. Besides it also neglects the fact that the general conjugate gradient method needs to execute almost \(n\) iterations at worst.

2 REVIEW OF THE SCHEME

The scheme involves two entities, the client, and the server. In order to ensure the privacy of input and output, the client needs to use some masking transformations (see Table 1).

3 ANALYSIS

The scheme is a revision of Ref. [2], which assumes that \(A\) is a general matrix. In view of that large sparse matrices often appear in scientific or engineering applications when solving partial differential equations, the revised scheme specifies that \(A\) must be sparse. But we find it did not take into the sparsity account.

• It is no longer necessary to outsource the computations because the client has the capability to finish the computations locally.

In fact, at each iteration we have

\[
\alpha_i = \frac{r_i^\top r_i}{p_i^\top A p_i}, \quad r_{i+1} = r_i + \alpha_i A' p_i,
\]

and \(x_{i+1} = x_i + \alpha_i p_i\),

\[
\beta_i+1 = r_{i+1}^\top r_{i+1} / r_i^\top r_i, \quad p_{i+1} = -r_{i+1} + \beta_{i+1} p_i.
\]

Eq. (1) involves only the sparse matrix-vector multiplication, instead of the general matrix-vector multiplication because \(A' = A A\) and \(A\) is sparse (which generally means \(A'\) is sparse, too). The calculation takes only \(\mathcal{O}(n)\) scalar multiplications, not \(\mathcal{O}(n^2)\). In view of that the general conjugate gradient method requires \(k \leq n\) iterations, the total complexity for the client to finish the whole task locally, is \(\mathcal{O}(n^2)\) at worst.

In the outsourcing scheme, at each iteration the client needs to compute

\[
\sum_{(\gamma,\theta) \in Z'} (p_k^\top \tilde{u}_\gamma) (v_\theta^\top p_k)
\]

which takes just \(\mathcal{O}(n)\) scalar multiplications. The total complexity for the client to finish the whole task interactively, is \(\mathcal{O}(n^2)\), too.

Note that an outsourcing scheme should greatly mitigate the client’s computational burden. But the scheme neglected the premise, and did not truly alleviate the client’s computational burden.

• The scheme is inapplicable even if \(A\) is not sparse. In this case, the client needs to compute

\[
M = Z_0^\top A + A' Z_1 + Z_0^\top Z_1
\]

where \(Z_0 = \sum_{(\gamma,\theta) \in Z} \tilde{u}_\gamma v_\theta^\top\). It involves the general matrix-vector multiplications. Apart from the general matrix-matrix multiplication \(G = \tilde{A}_1 \hat{A}_1\), finished by the server, in the later iterations the outsourced calculations

\[
f_k = \hat{A}' \hat{p}_{k2}, f_k = \hat{p}_{k1}^\top \hat{A}' \hat{p}_{k2}
\]

can be done by the client himself, because only matrix-vector multiplications are needed. Taking into account the communication cost for interactions (\(n\) times at worst),
The Salinas et al.'s scheme for outsourcing large-scale sparse linear equations

<table>
<thead>
<tr>
<th>Client [Input: A, b]</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Formalization] Pick $\tilde{Z}_0, \tilde{Z}_1 \in \mathbb{R}^{m \times n}$, and permutations $P_0, T_0, Q_0$, compute $\tilde{A}_0 = Q_0^T (A + \tilde{Z}_0) P_0^T$, $\tilde{A}_1 = Q_0^T (A + \tilde{Z}_1) T_0^T$. $\triangleleft \tilde{A}_0, \tilde{A}_1 \triangleright G = \tilde{A}_0^T \tilde{A}_1$</td>
<td></td>
</tr>
<tr>
<td>Compute $b' = A^T b$, $A' = P_0^T G T_0^T - M$, where $M = Z_0^T A + A^T \tilde{Z}<em>1 + T_0^T \tilde{Z}<em>1$ (To avoid matrix-matrix multiplications) replace $Z_0, \tilde{Z}<em>1$ with $\sum</em>{(\gamma, \delta) \in Z} u</em>{\gamma} v</em>{\delta}^T$, $\sum_{(\gamma, \delta) \in Z} u_{\delta} v_{\gamma}^T$.</td>
<td></td>
</tr>
<tr>
<td>[Initialization] Generate $P, T, \tilde{Z} \in \mathbb{R}^{n \times n}$ by the same method as that of composing $\tilde{Z}_0$. Pick $x_0 \in \mathbb{R}^{n \times 1}$, compute $\hat{A} = P(A' + \tilde{Z}) T$, $x_0 = T^T x_0$. $\triangleright \hat{A}, x_0 \triangleright h_0 = \hat{A} x_0$</td>
<td></td>
</tr>
<tr>
<td>Compute $r_0 = P^T h_0 - \sum_{(\gamma, \delta) \in Z'} u_{\gamma} (v_{\delta} x_0) - b' = A' x_0 - b'$.</td>
<td></td>
</tr>
<tr>
<td>Check $\sqrt{r_0^T r_0} \leq \nu |b'|<em>2$, where $\nu$ is a tolerance value. If not, set $p_0 = -r_0$, $p</em>{01} = P p_0$, $p_{02} = T^T p_0$. $\triangleright e_0, \hat{p}<em>{01}, \hat{p}</em>{02} \triangleright \ldots$</td>
<td></td>
</tr>
<tr>
<td>[Iterations] $\ldots$ Compute $\alpha_k = \frac{-r_k^T r_k}{e_k - \sum_{(\gamma, \delta) \in Z'} (p_{01} u_{\gamma} (v_{\delta} p_{01}))}$ $r_{k+1} = r_k + \alpha_k (P^T f_k - \sum_{(\gamma, \delta) \in Z'} u_{\delta} (v_{\gamma} p_{k2})).$ $\triangleright e_k, \hat{p}<em>{k1}, \hat{p}</em>{k2} \triangleright t_k = \hat{A}^T \hat{p}_{k2}$</td>
<td></td>
</tr>
<tr>
<td>If $\sqrt{r_{k+1}^T r_{k+1}} \leq \nu |b'|<em>2$, compute $x</em>{k+1} = x_k + \alpha_k p_k$.</td>
<td></td>
</tr>
</tbody>
</table>

which is determined in terms of data reading, data writing, data transfer rate, etc., and cannot be estimated theoretically (the general complexity is only determined in terms of bit operations), and the computational errors created in the main iterations, it seems better for the client to execute the main iterations by himself. The frequently interactive collaboration paradigm is rarely adopted in outsourcing schemes. We think, the amount of interactions between the client and the cloud server should be restricted to $O(\log n)$, where $n$ is the size of the outsourced problem. Otherwise, the tremendous interactions would be unbearable for a practical communication system.

By the way, it seems improper to take the comparison (see Table 3, Ref. [1]) with some flawed schemes [3], [4], [5]. In fact, the constructed sequence was not convergent [3]. The original problem was wrongly converted into a generally unsolvable problem [4]. The proposed scheme is somewhat artificial because it is unnecessary for the client to outsource the problem in the discussed scenario [5].

4 Conclusion
We show that the Salinas et al.’s outsourcing scheme is inapplicable because in the discussed scenario the client can solve the original problem locally. We would like to stress that the general conjugate gradient method could take almost $n$ iterations at worst. In this case, the communication cost for massive interactions between the client and the server could be unbearable.

Acknowledgment
We thank the National Natural Science Foundation of China (61411146001), the Brussels Region Innoviris and the SeCloud Project.

References