J. Math. Computer Sci., 21 (2020), 335–343

Online: ISSN 2008-949X

Journal of Mathematics and Computer Science

Journal Homepage: www.isr-publications.com/jmcs

Efficient and verifiable outsourcing computation of large-scale nonlinear programming

Nedal M. Mohammed[a,*], Ali N. AL-Seadi[b], Santosh S. Lomte[a], Poonam M. Rokade[a], Ahmed A. Hamoud[c]

[a]Department of Computer Science, Dr. Babasaheb Ambedkar Marathwada University, Aurangabad, India.

[b]Department of Computer Science, Thi-Qar University, Thi-Qar, Iraq.

[c]Department of Mathematics, Dr. Babasaheb Ambedkar Marathwada University, Aurangabad, India.

Abstract

Nonlinear programming (NLP) problems arise in various fields, such as transport, financial engineering, logistics, urban planning, supply chain management, and power system control. Solving large-scale NLPs are usually so computationally expensive for resource-constrained users within a feasible time. The cost-effective solution is computation outsourcing, but this raises security concerns such as the input and output privacy of the customers, and cheating behaviors of the cloud since NLP problems always carry sensitive information. In this paper, we develop a practical secure and verifiable schema for solving outsourcing large-scale (NLP) with the GRG method. Also, we apply approximate KKT conditions for verifying the optimality of the result returned by the GRG algorithm. We implement the proposed schema on the customer side laptop and using AWS compute domain elastic compute cloud (EC2) for the cloud side.

Keywords: Cloud computing, secure computation outsourcing, verifiable computing, security and privacy nonlinear programming problems.

2020 MSC: 68Q99, 90C30.

1. Introduction

The powerful advantages of cloud computing is called outsourcing, where the customers with limited computing resource and storage devices can outsource the sophisticated computation workloads into powerful service providers. Despite the tremendous benefits, there are many challenges and security concerns because the cloud server and customer are not in the same trusted domain, to combat these security concerns [9, 14, 15, 23] first applying encryption techniques to customer's sensitive information before outsourcing to the cloud but still, there is a challenge how makes the task of computation over encrypted data [6–8, 10, 14, 18, 21]. Focusing optimization tasks we notice that the optimization computations problems (linear and nonlinear programming (LP & NLP)) frequently appear in various fields (optimal decision making, large-scale data analytics, etc) such as (financial engineering, support vector machines, scheduling routes in intelligent transportation systems, etc), but solving these large-scale LP &

*Corresponding author
Email address: dr.nedal.mohammed@gmail.com (Nedal M. Mohammed)

doi: 10.22436/jmcs.021.04.06

Received: 2020-01-09 *Revised:* 2020-02-04 *Accepted:* 2020-04-08

NLP problems is usually computationally so expensive. The customer with limited computing resource and storage devices are facing the challenge of solving large-scale LP& NLP problems [16], the alternative option to address this challenge is outsourcing to cloud computing. This work, focuses on secure outsourcing large-scale NLP. The nonlinear programming is a general optimization problem, it has many applications in both industry and academic communities. The more researchers and specialists have devoted considerable attention to this field of real-world problems [2, 17]. We are proposing a secure, efficient and verifiable scheme to offload large-scale (NLP) computation to the cloud side apply approximate KKT conditions for verifying the optimality of the result.

This paper is organized as follows. Section 2 shows some preliminaries. Section 3, describes our proposed system model to solve the large scale NLP problem. Section 4 presents protocol design of secure nonlinear programming. In Section 5, we provide experimental result analysis for proposed schema. At last the work conclusion is presented in Section 6.

2. Preliminaries

2.1. Nonlinear programming

In general, Nonlinear Program (NLP) is mathematical optimization expressed in the following standard form [3, 20]:

$$\begin{aligned} \text{Minimize} \quad & f(X) \\ \text{Subject to} \quad & g_i(x) \leqslant b_i, \quad i = 1, 2, \ldots, m, \\ & h_j(x) = d_j, \quad i = 1, 2, \ldots, l, \end{aligned} \tag{2.1}$$

where $x = (x_1, x_2, \ldots, x_n)^T$ is an n-dimension vector of the optimization variables, a general nonlinear function $f(x)$: $R^n \longrightarrow R$ is an objective function, general nonlinear functions $g_i(x) : R^n \longrightarrow R$ (for $i \in [1, m]$) are inequality constraints, general nonlinear functions $h_j(x) : R^n \longrightarrow R$ (for $j \in [1, l]$) are equality constraints, $b_1, b_2, \ldots, b_m$ and $d_1, d_2, \ldots, d_l$ are the bounds for the constraints.

2.2. Computational indistinguishability

Definition 2.1. Let I be a countable set. A probability ensemble indexed by I is a collection of random variables $\{Xi\}_{i \in I}$. I is either nature numbers N or an efficiently computable subset $\{0, 1\}^n$, and with this definition we can formally show what computational indistinguishability means for two ensembles [4].

Definition 2.2. Two probability ensembles $X = \{X_n\}_{n \in N}$ and $Y = \{Y_n\}_{n \in N}$ are computationally indistinguishable, denoted by $X \equiv Y$ if for every probabilistic polynomial-time distinguisher D there exists a negligible function neg(.) such that

$$|P_r[D(X_n) = 1] - P_r[D(Y_n) = 1]| \leqslant neg(.), \tag{2.2}$$

where the notation $D(X_n)$ means that x is chosen according to distribution X_n and $D(x)$ is run. $D(x)$ will output 1 if the distinguisher D determines that x is not from the ensemble X_n otherwise 0. The definition of computational indistinguishability between two ensembles can also be extended to two matrices which consist of multiple samples of ensembles.

Definition 2.3. Let $X \in R^{m \times n}$ be a matrix with its i^{th} row $\forall\ i \in [1, m]$ or j^{th} column $\forall\ j \in [1, n]$ be a probability ensemble. Matrices X and $Y \in R^{m \times n}$ are computationally indistinguishable denoted by $X \equiv Y$ if for any probabilistic polynomial time distinguisher D there exists a negligible function neg(.) such that

$$|P_r[D(X) = 1] - P_r[D(Y) = 1]| \leqslant neg(.),$$

where the notation D(X) means xi;j is chosen from the matrix X for $\forall\ i \in [1, m]$, $\forall\ j \in [1, n]$ and $D(x_{i,j})$ is run. Distinguisher D will output 1 when it determines $x_{i,j}$ is not chosen from matrix X and 0 otherwise.

Definition 2.4. A randomized algorithm satisfies computational indistinguishability if and only if for any two databases D and $\acute{D}$, for every probabilistic polynomial-time adversary machine M, there exists a negligible function neg(.) such that: $|Pr[MA(D)] - Pr[MA(D')]| \leqslant neg(.)$, where the notation $MA(D)$ (similarly for $MA(D')$) means that adversary machines have access to the database and try to extract private information from the data [11]. Definition 2.1 measures the information leakage level of the encryption scheme that encrypts the original NLP problem. If computational indistinguishability is achieved, the cloud server cannot learn anything significant about the original NLP problem.

2.3. KKT Conditions

Karush-Kuhn-Tucker (KKT) optimality conditions can be used to check the optimality of the result returned by algorithms of solving optimization problems [22]. For the NLP problem described in Eq. (2.1), the KKT conditions are defined as follows:

$$\nabla f(x^*) + \sum_{i=1}^{l} u_i \nabla g_i(x^*) = 0, \tag{2.3}$$

$$g_i(x^*) \leqslant 0, \ \forall i, \tag{2.4}$$

$$u_i g_i(x^*) = 0, \ \forall i, \tag{2.5}$$

$$u_i \geqslant 0, \forall i, \tag{2.6}$$

where u_i is the Lagrange multiplier for the i th constraint. The solution x^* that satisfies Eqs. (2.3)-(2.6) is called a KKT point. A norm of the left side in Eq. (2.3) is the KKT error; x^* is a feasible point in Eq. (2.4); each constraint in Eq. (2.5) is called the complementary slackness equation; Eq. (2.6) means that the Lagrange multipliers are nonnegative. Eq. (2.5) indicates that if the KKT point x^* makes a constraint inactive, the corresponding Lagrange multiplier must be zero. On the contrary, if x^* makes constraint active, as Eq. (2.6) shows, the Lagrange multiplier can be any nonnegative values. Hence, Eq. (2.3) implies that the negative of the gradient vector $\nabla f(x^*)$ of the objective function in Eq. (2.2) should be a positive linear combination of the gradient vectors of the active constraints.

Definition 2.5. A point x, which is feasible for problem P, is said to be a modified ε-KKT point for a given $\varepsilon > 0$, if there exists $x^* \in R^n$ such that $\|x - x^*\| \leqslant \sqrt{\varepsilon}$ and scalars $u_i \geqslant 0$ for $i = 1, \ldots, l$ such that

1. $\|\nabla f(x^*) + \sum_{i=1}^{l} u_i \nabla g_i(x^*)\| \leqslant \sqrt{\varepsilon}$;
2. $\sum_{i=1}^{l} u_i g_i(x) \geqslant -\varepsilon$.

Actually, x^* in above definition is not strict to be feasible. What calls for special attention is that the first condition is defined for x^* while the second condition must be true for the point x [1].

3. System model

We now give the general framework of our scheme, where the client with resource-constrained has a computationally extensive NLP problem to be optimized and due to limited computing resources the client cannot do this computation to solve the problem locally. He offloads the computation task to the cloud with ample computational resource, but cannot be trusted with the sensitive information. Then to avoid security problems our proposed scheme consists of these phases.

Problem generation: This phase consists of two parts: preprocessing and request. In preprocessing part, the client reformulates the Ψ original problem it generate secret key k by Key-generation algorithm to protect the privacy of the original NLP Ψ problem, and in request part, the client sends an outsourcing request to the cloud. The request includes the reformulated problem Ψ_k.

Computation outsourcing: When the cloud receives the client's request, it uses **GRG** algorithm to solve encrypted NLP Ψ_k problem. After GRG algorithm terminates, the cloud send its output back to the client.

Verification: On receiving the result from the cloud, the client executes the verification algorithm to check the honesty of the cloud it verify result correctness, and then decid to reject or accept it. If the returned result is proved to be correct, the client will decrypt it and treat it as the optimal solution see Fig. 1.

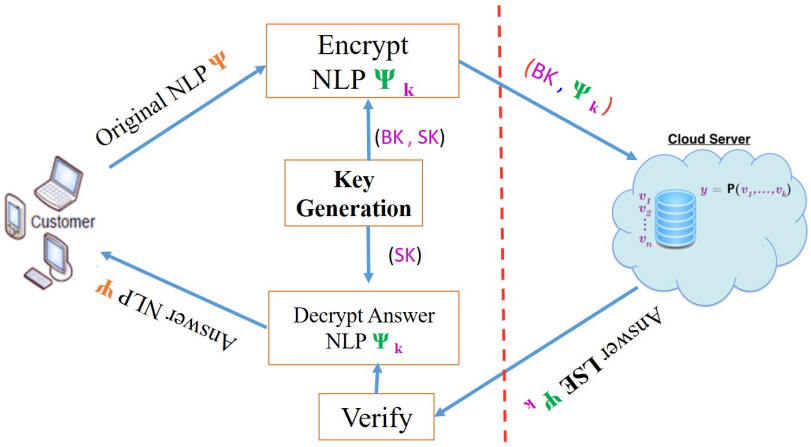

Figure 1: System model of secure outsourcing of NLP problem.

4. Protocol design of secure nonlinear programming problem

To achieve goals of our work, we design the following sub-algorithms.

4.1. Key generation algorithm

In key generation algorithm, secret keys are randomly generated by Key generation algorithm at the client side to to encrypt the original NLP problem. These keys are vector $r \in R^{n\times 1}$ can be used to protect the variable vector $(x_1; x_2; x_3; \ldots; x_n)^T$, $Q, W \in R^{m\times m}$ key matrices use to hid the coefficient matrix $A \in R^{m\times n}$, which contain sensitive information.

Algorithm 4.1 (Key generation).

1. **Input:** input size n;
2. **Output:** random vector r; two random matrixes P, W; random permutation matrixes Q, S;
3. set $M = 1, 2, 3, \Delta\Delta\Delta, n$;
4. for $j = 1$ to n
5. select i randomly from $i \in (1, j)$;
6. swap $M(i)$ and $M(j)$;
7. end for
8. for $i = 1$ to n
9. for $j = 1$ to n
10. φ outputs the i^{th} element from M with $\varphi(i)$;
11. δ outputs value with $\sigma\varphi(i), j$;
12. set $N(i, j) = \sigma\varphi(i), j$;
13. end for;

14. end for;
15. calculate $b' = MNA$;
16. calculate $A' = MNb$;
17. generate vector r as
18. for $i = 1$ to n
19. if $r[i] > (-N)$ and $r[i] < N$
20. $r[i] = r$;
21. else
22. $r[i] = 0$;
23. return (vector r; matrix Q; matrix W; permutation matrix P).

4.2. Computation outsourcing

We now describe the outsourced running of reduced gradient method algorithm for solving constrained NLP problem in the cloud. In reduced gradient method. The main input argument is the starting point z_0 and returns the result z^* the optimized solution.

4.3. Generalized reduced gradient method for solving NLPs

GRG method is robust and efficient in solving large-scale NLP practically. The procedure is going as follows:

First transfer inequality constraints as $\hat{h}_j(z) \leqslant \hat{b}_j, j = 1, \ldots, l$ to equality constraints,

$$\begin{aligned} \hat{h}_j(z) + s_j \leqslant \hat{b}_j = 0, \quad j = 1, \ldots, l \\ s_j \geqslant 0, \quad j = 1, \ldots, l. \end{aligned}$$

Thus we can rewrite NLP in the following general form:

$$\begin{aligned} \text{Minimize} \quad & f(z, s) \\ \text{subject to} \quad & \hat{c}_i(z, s) = 0, \ i = 1, \ldots, m, \ldots, m + l, \\ & \hat{a}_k \leqslant z_k \leqslant \hat{u}_k, \ k = 1, \ldots, n, \\ & 0 \leqslant s_p < \infty, \ p = 1, \ldots, l, \end{aligned}$$

where $\hat{c}_i(z, s) = 0$ are the constraints. For simplicity of notation, we can use $y = (z, s)$ to represent the variable vector, and $v \leqslant y \leqslant w$ to denote the range of the variables [19]. The optimization of problem is only dependent on the non-basic variables Z_N, since basic variable vector Z_B can be uniquely determined from Z_N. A simple modification of the gradient decent method will provide a feasible improving direction d to optimize the objective function. A feasible improving direction d at the point y must follow [3, 4]:

$$Ed = 0, \tag{4.1}$$

$$\bigtriangledown f(z)^T d < 0, \tag{4.2}$$

where $\bigtriangledown f(z)^T$ is the gradient vector of objective function $f(z)$ at point z. Eq. (4.1) means that if a feasible point y moves along the direction d, the feasibility of the constraints will not be damaged. Eq. (4.2) indicates that moving along d will make the objective function $f(z)$ approach the optimal point. The reduced gradient method as the following will find such moving direction d that satisfies Eq. (4.1). The gradient vector corresponding to z_N (also called as reduced gradient) can be found by the following expression:

$$r^T = \bigtriangledown_N f(z)^T - \bigtriangledown_B f(z)^T E_B^{-1} E_N,$$

where $\bigtriangledown_N f(z)^T$ is the gradient vector of $\bigtriangledown f(z)^T$ that corresponds to z_N and $\bigtriangledown_B f(z)^T$ is the gradient vector corresponding to z_B. From above reduced gradient, we can construct the feasible moving direction d_N that will move $z_N + \lambda d_N$ in the feasible working space, where d_N can be determined as the following:

$$d_{Ni} = \begin{cases} -r_i, & r_i \leqslant 0, \\ -z_{Ni} r_i, & r_i > 0, \end{cases} \tag{4.3}$$

where d_{Ni} is the ith element of d_N, r_i is the ith element of r^T, and z_{Ni} is the ith element of z_N. The Eq. (4.3) provides the rules for finding improving feasible direction for non-basic variables z_N. Once the improving feasible direction for z_N is determined, we can get the corresponding moving direction d_B for z_B by expanding Eq. (4.1):

$$E_N d_N + E_B d_B = 0, \quad d_B = -E_B^{-1} E_N d_N. \tag{4.4}$$

Eq. (4.4) shows that dB can be uniquely calculated from d_N, and the moving direction is composed that $d = [d_B, d_N]$. It can be proved that $d = [d_B, d_N]$ satisfies Eq. (4.1) and the Eq. (4.2), indicating both feasibility and improvability will be achieved for d. The reduced gradient method partition the variables into basic and non-basic variable vector as $z = (z_B, z_N)$, and the corresponding Jacobi matrix of $\hat{c}(z)$ can also be grouped into:

$$\frac{\partial \hat{c}}{\partial z} = \left(\frac{\partial \hat{c}}{\partial z_B}, \frac{\partial \hat{c}}{\partial z_N}\right)$$

and a non-degeneracy assumption is made here that for any point y, $\frac{\partial \hat{c}}{\partial y_B} \in \mathbb{R}^{(m+l)\times(m+l)}$ is non-singular.

For the case of nonlinear constraints, the reduced gradient r^T with respect to y_N is expressed as:

$$r^T = \bigtriangledown_N f(z)^T - \bigtriangledown_B f(z)^T \left(\frac{\partial \hat{c}}{\partial z_B}\right)^{-1} \frac{\partial \hat{c}}{\partial z_N}.$$

Now we specify the direction d_N as follows:

$$d_{Ni} = \begin{cases} 0, & z_{Ni} r_i > 0 \text{ and } z_{Ni} = v_i, \\ 0, & z_{Ni} r_i < 0 \text{ and } z_{Ni} = w_i, \\ -z_{Ni} r_i, & \text{otherwise}, \end{cases} \tag{4.5}$$

where v_i is the lower bound of the variable z_i and w_i is the upper bound of the variable z_i. However, the difference with the linear form is that z_N moves a straight line along d_N, the nonlinear form of the constraints requires z_N move nonlinearly to continuously walk in the feasible space formed by the constraints. To address this, we can first move z_N along the direction defined by Eq. (4.5), then a correction procedure is employed making z_N return to working space to satisfy the feasibility of the constraints. Once a tentative move along d_N is made, the following iterative method can be used for the correction. Supposing z_k is the current feasible point, we first move the non-basic variable vector $z_{N(k+1)} = z_{Nk} + \lambda d_{Nk}$, to return point $z = (z_{Bk}, z_{N(k+1)})$ near z_k back to the constraint space, we can solve the following equation:

$$\hat{c}(z_{Bk}, z_{N(k+1)}) = 0$$

for z_{Bk} where $z_{N(k+1)}$ is fixed.

$$z_{Bk_{j+1}} = z_{Bk_j} - \left(\frac{\partial \hat{c}(z_{Bk_j}, z_{N(k+1)})}{\partial z_{Bk}}\right)^{-1} \hat{c}(z_{Bk_j}, z_{N(k+1)}),$$

when this iterative process produces a feasible point $z_{B(k+1)}$, we have to check if following conditions are satisfied:

$$f(z_{B(k+1)}, z_{N(k+1)}) < f(z_{Bk}, z_{Nk}), \quad v \leqslant (z_{B(k+1)}, z_{N(k+1)}) \leqslant w. \tag{4.6}$$

If Eq. (4.6) holds true, it indicates that the new point is feasible and improvable. Then we set $z_{k+1} = (z_{B(k+1)}, z_{N(k+1)})$ as a new approaching point, otherwise we will decrease the step length λ when we first make the tentative move for z_{Nk} and repeat the above iterative process [12].

4.4. Verification algorithm

Once the client receives the result from untrusted cloud, he verifies the result by invoking the verification algorithm, he check the correctness of the result and see whether the cloud has faithfully executed the computation of GRG algorithm. An verification algorithm with approximate KKT conditions to inspect the solution returned by GRG algorithm first we test whether the result is not a maximal solution. Because a maximal solution can also pass the verification in next step for the necessary of KKT conditions, second we mainly check whether the returned solution is a ε-KKT point, further conditions are necessary to establish the optimality of a point [13, 22].

ε-KKT modified point of the KKT optimality conditions is defined to overcome the difficulty [5]. The extent of violation of KKT conditions at points close to the KKT point may not reduce smoothly, thus making the KKT conditions hard to directly evaluate the optimality of an optimization algorithm. A minimizer solution might not meet the KKT conditions, but it can always be approximated by a sequence of approximate KKT points [1].

Algorithm 4.2 (Verification algorithm)**.**

1. Input: r, δ, ε;
2. randomly generate a $\triangle x, \triangle x \in [-\delta, \delta]$;
3. if $F(gbest + \triangle x) < F(gbest)$, then
4. return f;
5. end if;
6. if gbest satisfies all constraints in Eq. (2.3), then
7. calculate conditions in Section 2.3;
8. if gbest is a ε-KKT point, then
9. return true;
10. else
11. return f;
12. end if;
13. end for;
14. else
15. return f;
16. end if %(outputs: verification result.)

5. Experimental result analysis

The experimental results are the average of multiple trials. We design numerical experiments to evaluate the efficiency of the mechanism. We implemented it using Matlab (2013a), with the system configuration is "CPU Intel® Core™i3(CPUs)˜1.8GHZ4GB Ram" on a laptop and Amazon Elastic Compute Cloud (EC2) cluster. To measure the efficiency of our proposed mechanism, we define

1. $t_{original}$: required time to solve the NLP Ψ problem (customer side);
2. $t_{customer}$: full customer time to perform computation outsourcing;
3. t_{cloud} : required time for the cloud to solve Ψ_k problem in seconds;
4. asymmetric speedup: the performance gain represents savings of the computing resources for the customer to outsource the Ψ_k problem to the cloud;
5. efficiency: it is calculated as Efficiency$= \frac{t_{original}}{t_{cloud}}$.

Table 1: Performance of the proposed scheme for secure computation of NLP.

Problem Size		Original Problem	Encrypted Problem		Asymmetric Speedup	Cloud Efficiency
#	Dimension	$t_{original}$	t_{cloud}	t_{client}	$\frac{t_{original}}{t_{client}}$	$\frac{t_{original}}{t_{cloud}}$
1	$n = 2000$	28.01	19.03	0.69	40.59	1.47
2	$n = 4000$	181.98	111.99	5.06	35.96	1.62
3	$n = 8000$	1235.87	698.01	37.14	33.28	1.77
4	$n = 12000$	2776.84	1368.21	76.31	36.39	2.02
5	$n = 16000$	8246.01	3719.33	164.98	49.98	2.22

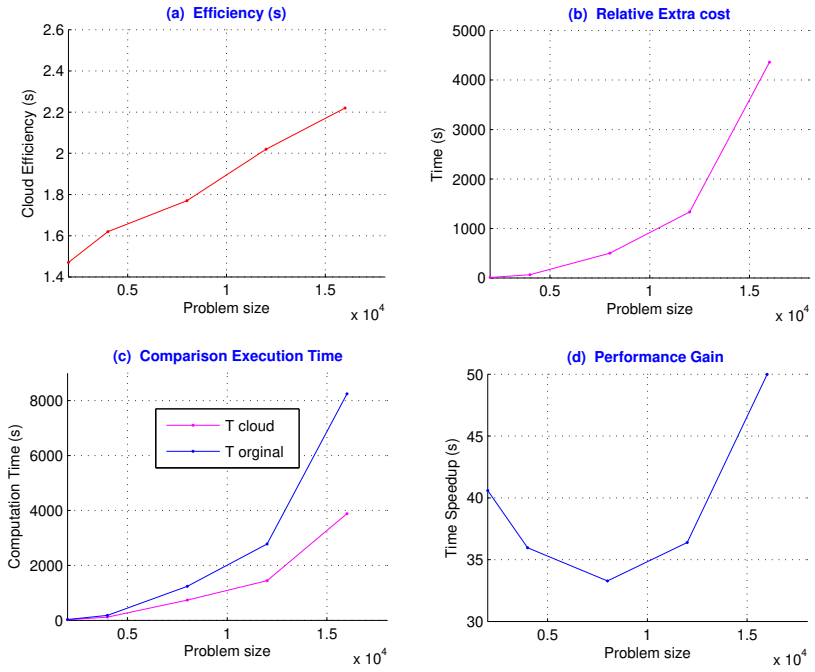

Figure 2: Performance of secure computation outsourcing of large scale NLP.

Fig. 2 (a) shows analyzed computing time at both the customer and cloud. We used efficiency and performance-gain to measure the performance of the proposed mechanism. Ideally, the efficiency of the algorithm should be close to one. If the efficiency is nearby one, it indicates that the execution time of original problem and the encrypted problem is almost same. We can see in Fig. 2 (b) and Table 1, that the efficiency parameter remains close to one, which means the outsourcing paradigm adds minimum overhead on the cloud for executing an encrypted problem. The second parameter is performance gain for the customer. Theoretically, the performance gain is expected to be greater than one as the results in Fig. 2 (d), otherwise it is meaningless for the customer to outsource the NLP problem. The extra time is defined as the amount of time for extra work done by the customer and cloud server in outsourcing paradigm as compared to direct method, the results in Fig. 2 (c).

6. Conclusion

In this paper, we study the secure computation outsourcing of large-scale (NLP) problem by (GRG) algorithm and its verification. We formulate the problem and its security model. Also we propose a verification scheme using (KKT) conditions with the (ε-KKT) point. We implement schema on the customer side laptop and using AWS compute domain elastic compute cloud (EC2) for the cloud side.

References

[1] R. Andreani, G. Haeser, J. M. Martinez, *On sequential optimality conditions for smooth constrained optimization*, Optimization, **60** (2011), 627–641. 2.5, 4.4

[2] E. B. Bajalinov, *Linear-fractional programming theory, methods, applications and software*, Springer Science & Business Media, Berlin, (2013). 1

[3] M. S. Bazaraa, H. D. Sherali, C. M. Shetty, *Nonlinear programming: theory and algorithms*, John Wiley & Sons, New York, (2013). 2.1, 4.3

[4] D. P. Bertsekas, *Nonlinear programming*, Athena scientific, Belmont, (1999). 2.1, 4.3

[5] J. Dutta, K. Deb, R. Tulshyan, R. Arora, *Approximate KKT points and a proximity measure for termination*, J. Glob. Optim., **56** (2013), 1463–1499. 4.4

[6] C. Gentry, *Computing arbitrary functions of encrypted data*, Comm. ACM, **53** (2010), 97–105. 1

[7] A. A. Hamoud, K. H. Hussain, N. M. Mohammed, K. P. Ghadle, *Solving Fredholm integro-differential equations by using numerical techniques*, Nonlinear Funct. Anal. Appl., **24** (2019), 533–542.

[8] A. A. Hamoud, N. M. Mohammed, K. P. Ghadle, *A study of some effective techniques for solving Volterra-Fredholm integral equations*, Dyn. Contin. Discrete Impuls. Syst. Ser. A Math. Anal., **26** (2019), 389–406. 1

[9] C. Hu, A. Alhothaily, A. Alrawais, X. Cheng, C. Sturtivant, H. Liu, *A secure and verifiable outsourcing scheme for matrix inverse computation*, IEEE INFOCOM'17 (Atlanta, GA, U.S.A.), **4** (2017), 2304–2312. 1

[10] K. H. Hussain, A. A. Hamoud, N. M. Mohammed, *Some new uniqueness results for fractional integro-differential equations*, Nonlinear Funct. Anal. Appl., **24** (2019), 827–836. 1

[11] J. Katz, Y. Lindell, *Introduction to modern cryptography*, CRC press, Boca Raton, (2015). 2.4

[12] A. Li, W. Du, Q. Li, *Privacy-preserving outsourcing of large-scale nonlinear programming to the cloud*, Int. Conf. Security Privacy Comm. Syst. (Springer), **2018** (2018), 569–587. 4.3

[13] D. G. Luenberger, Y. Ye, *Linear and nonlinear programming*, Springer, New York, (2008). 4.4

[14] N. M. Mohammed, S. S. Lomte, *Recent advances on secure computations outsourcing in cloud computing*, Asian J. Math. Comput. Res., **24** (2017), 192–205. 1

[15] N. M. Mohammed, S. S. Lomte, *Secure computations outsourcing of mathematical optimization and linear algebra tasks: Survey*, National Conference on Recent Innovation in Computer Science & Electronics, **2019** (2019), 6 pages. 1

[16] N. M. Mohammed, S. S. Lomte, *Secure and efficient outsourcing of large scale linear fractional programming*, Adv. Intel. Syst. Comput., **2020** (2020), 277–286. 1

[17] N. M. Mohammed, S. S. Lomte, *Verifiable secure computation of linear fractional programming using certificate validation*, Int. J. Power Electron. Drive Syst., **11** (2020), 284–290. 1

[18] N. M. Mohammed, L. Sultan, S. S. Lomte, *Privacy preserving outsourcing algorithm for two-point linear boundary value problems*, Indonesian J. Ele. Eng. Comput. Sci., **16** (2019), 1065–1069. 1

[19] J. B. Rosen, *The gradient projection method for nonlinear programming. I: Linear constraints*, J. Soc. Indust. Appl. Math., **8** (1960), 181–217. 4.3

[20] J. B. Rosen, *The gradient projection method for nonlinear programming. II. Nonlinear constraints*, J. Soc. Indust. Appl. Math., **9** (1961), 514–532. 2.1

[21] W. Shen, B. Yin, X. H. Cao, Y. Cheng, X. S. Shen, *A distributed secure outsourcing scheme for solving linear algebraic equations in ad hoc clouds*, IEEE Trans. Cloud Comput., **4** (2017), 415–430. 1

[22] R. Tulshyan, R. Arora, K. Deb, J. Dutta, *Investigating EA solutions for approximate KKT conditions in smooth problems*, Proceedings of Annual Conference on Genetic and Evolutionary Computation, **2010** (2010), 689–696. 2.3, 4.4

[23] K. Zhou, J. Ren, *CASO: Cost-aware secure outsourcing of general computational problems*, IEEE Tran. Services Comput., **2018** (2018), 13 pages. 1