Efficient Distributed Linear Programming with Limited Disclosure[⋆]

Yuan Hong, Jaideep Vaidya and Haibing Lu

MSIS Department and CIMIC, Rutgers University, NJ, USA
{yhong, jsvaidya, haibing}@cimic.rutgers.edu

Abstract. In today's networked world, resource providers and consumers are distributed globally and locally. However, with resource constraints, optimization is necessary to ensure the best possible usage of such scarce resources. *Distributed linear programming (DisLP)* problems allow collaborative agents to jointly maximize profits (or minimize costs) with a linear objective function while conforming to several shared as well as local linear constraints. Since each agent's share of the global constraints and the local constraints generally refer to its private limitations or capacities, serious privacy problems may arise if such information is revealed. While there have been some solutions proposed that allow secure computation of such problems, they typically rely on inefficient protocols with enormous communication cost. In this paper, we present a secure and extremely efficient protocol to solve DisLP problems where constraints are arbitrarily partitioned and no variable is shared between agents. In the entire protocol, each agent learns only a partial solution (about its variables), but learns nothing about the private input/output of other agents, assuming semi-honest behavior. We present a rigorous security proof and communication cost analysis for our protocol and experimentally validate the costs, demonstrating its robustness.

1 Introduction

Optimization is a fundamental problem found in all industries. As an essential subclass of optimization, linear programming models are widely applicable to solving numerous profit-maximizing or cost-minimizing problems in various fields such as transportation, commodities, airlines and communication.

For instance, in the packaged goods industry, delivery trucks are empty 25% of the time. Just four years ago, Land O'Lakes truckers spent much of their time shuttling empty trucks down slow-moving highways, wasting several million dollars annually. By using a web based collaborative logistics service (Nistevo.com), to merge loads from different companies (even competitors) bound to the same destination, huge savings were realized (freight costs were cut by 15%, for an annual savings of $\$2$ million[1]). This required sending all information to a central site. Such complete sharing of data may often be impossible for many corporations, and thus result in great loss of possible efficiencies. Since this is a transportation problem which can be modeled through linear programming, a *Distributed linear programming* (DisLP) solution that tightly limits

[⋆] This work is supported in part by the National Science Foundation under Grant No. CNS-0746943.

the information disclosure would make this possible without the release of proprietary information. Specifically, DisLP problems can facilitate collaborative agents to jointly maximize global profits (or minimize costs) while satisfying several (global or local) linear constraints. Since each agent's share of the global constraints and the local constraints generally refer to its private limitations or capacities and the optimal solution represents its decision, limited disclosure should prevent revealing such information in this distributed computing scenario.

While completely arbitrary partitioning of constraints and variables is possible, in many realistic DisLP problems, each company holds its own variables: the values for which together constitute the global optimum decision. Variables are generally not shared between companies because collaborators may have their own operations w.r.t. a maximized profit or minimized cost. Consider the following example:

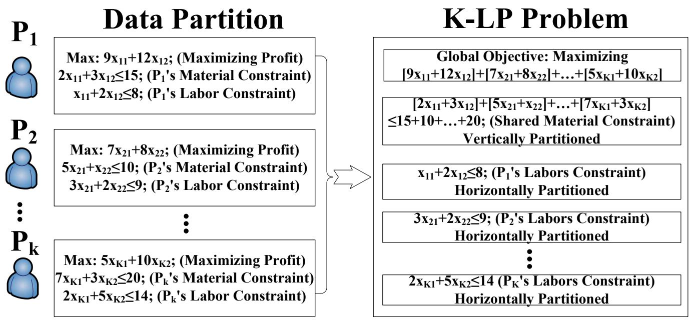

Data Partition
K-LP Problem
P_1
P_2
P_k
Max: $9x_{11}+12x_{12}$; (Maximizing Profit)
$2x_{11}+3x_{12}\leq 15$; (P_1's Material Constraint)
$x_{11}+2x_{12}\leq 8$; (P_1's Labor Constraint)
Max: $7x_{21}+8x_{22}$; (Maximizing Profit)
$5x_{21}+x_{22}\leq 10$; (P_2's Material Constraint)
$3x_{21}+2x_{22}\leq 9$; (P_2's Labor Constraint)
Max: $5x_{K1}+10x_{K2}$; (Maximizing Profit)
$7x_{K1}+3x_{K2}\leq 20$; (P_k's Material Constraint)
$2x_{K1}+5x_{K2}\leq 14$; (P_k's Labor Constraint)
Global Objective: Maximizing
$[9x_{11}+12x_{12}]+[7x_{21}+8x_{22}]+\ldots+[5x_{K1}+10x_{K2}]$
$[2x_{11}+3x_{12}]+[5x_{21}+x_{22}]+\ldots+[7x_{K1}+3x_{K2}]$
$\leq 15+10+\ldots+20$; (Shared Material Constraint)
Vertically Partitioned
$x_{11}+2x_{12}\leq 8$; (P_1's Labors Constraint)
Horizontally Partitioned
$3x_{21}+2x_{22}\leq 9$; (P_2's Labors Constraint)
Horizontally Partitioned
$2x_{K1}+5x_{K2}\leq 14$ (P_K's Labors Constraint)
Horizontally Partitioned

Fig. 1. Distributed LP Problem Formulation (Example 1)

Example 1. K Companies $P_1 \ldots P_K$ share some raw materials for production (maximizing profits): the amount of company P_i's $(i \in [1, K])$ product j to be manufactured are denoted as x_{ij}, thus P_i holds $x_i = \{\forall j, x_{ij}\}$.

In the collaborative production problem above, the constraints are arbitrarily partitioned. On one hand, $P_1 \ldots P_K$ should have some local constraints (i.e. each company's local labor constraint) that is only known to each company. On the other hand, there may be some global constraints (i.e. the total quantity of the shared raw materials). Figure 1 demonstrates a simple example of this. K companies jointly manufacture products (two for each company) using a shared material where P_1, P_2, ..., P_K have the amount 15, 10, ..., 20, respectively (The sum of the global profits can be increased by this collaboration since the combined resources are better utilized). They also have their local constraints, i.e. the total labor for producing each company's products are bounded with constraints $8, 9$ and 14 respectively. After solving this DisLP problem, each company should know the (global) optimal production amount for only its products but should not learn anything about the private constraints and solution of other companies. To simplify the notation, we formally define it as below:

Definition 1 (K-Agent LP Problem (K-LP)). *An LP problem is solved by K distributed agents where each agent P_i holds n_i variables x_i, share of the objective c_i,* *its local constraints $B_i x_i \bowtie_i b_i$, and the matrix/vector A_i/b_0^i in the global constraints $\sum_{i=1}^{K} A_i x_i \bowtie_0 b_0$ (as shown in Equation 1*[1,2,3]*)($i \in [1, K]$ and $\sum_{i=1}^{K} b_0^i = b_0$).*

$$
\begin{array}{ll}
\max & c_1^T x_1 + c_2^T x_2 + \cdots + c_K^T x_K \\
s.t. \begin{cases} x_1 \in \mathbb{R}^{n_1} \\ x_2 \in \mathbb{R}^{n_2} \\ \vdots \\ x_K \in \mathbb{R}^{n_K} \end{cases} : & \begin{pmatrix} A_1 & \dots & A_K \\ B_1 & & \\ & \ddots & \\ & & B_K \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_K \end{pmatrix} \begin{matrix} \bowtie_0 \\ \bowtie_1 \\ \vdots \\ \bowtie_K \end{matrix} \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_K \end{pmatrix}
\end{array}
\tag{1}
$$

Indeed, besides collaborative production, K-LP problems occur very frequently in reality, i.e. collaborative delivery of goods for different companies to save transportation cost, selling the goods in bundles for distributed agents to maximize the global profits, and determining profit-maximized travel packages for hotels, airlines and car rental companies.

We intend to introduce a secure and efficient distributed computing solution to the K-LP problem. Thus, our key contributions are: 1) to propose a privacy-preserving transformation for the K-LP problem; 2) to propose a secure protocol robust against honest-but-curious adversaries (semi-honest model: assuming that all the agents follow our protocol) that is fair to all agents, and 3) to experimentally validate the cost of the proposed protocol.

The rest of this paper is structured as follows. Section 2 reviews some related work. Section 3 introduces some preliminaries for our approach. Section 4 presents the transformation process (for security purpose) and shows that how to derive the optimal solution for each agent after solving the transformed problem. In Section 5, we present the secure protocol with security proof and computation cost analysis. Finally, we experimentally validate the protocol in Section 6 and conclude the paper in Section 7.

2 Literature Review

Optimization problems occur in all walks of real life. There is work in distributed optimization that aims to achieve a global objective using only local information. Distributed Constraint Satisfaction was formalized by Yokoo[2] to solve naturally distributed constraint satisfaction problems. These problems are divided between agents, who then have to communicate among themselves to solve them. ADOPT[3] is a backtracking based bound propagation mechanism. It operates completely decentralized, and asynchronously. The downside is that it may require a very large number of messages, thus producing big communication overheads.

However, in general, the work in distributed optimization has concentrated on reducing communication costs and has paid little or no attention to security constraints. Thus, some of the summaries may reveal significant information. In particular, the rigor of security proofs has not been applied much in this area. There is some work in secure optimization. Silaghi and Rajeshirke [4] show that a secure combinatorial problem

[1] $\bowtie$ denotes $\leq, =$ or $\geq$.
[2] Due to $\{\min : c^T x \equiv \max : -c^T x\}$, we model $\max : c^T x$.
[3] size: $\forall i \in [1, K]$, $\{A_i : m_0 \times n_i\}$, $\{B_i : m_i \times n_i\}$, $\{c_i : n_i\}$, $\{b_0 : m_0\}$ and $\{b_i : m_i\}$

solver must necessarily pick the result randomly among optimal solutions to be really secure. Silaghi and Mitra [5] propose arithmetic circuits for solving constraint optimization problems that are exponential in the number of variables for any constraint graph. A significantly more efficient optimization protocol specialized on generalized Vickrey auctions and based on dynamic programming is proposed by Suzuki and Yokoo [6]. However, much of this work is still based on generic solutions and not quite ready for practical use. Even so, some of this work can definitely be leveraged to advance the state of the art by building general transformations or privacy-preserving variants of well known methods.

Privacy-preserving linear programming problem has been introduced to solve the LP problem with limited information disclosure between two agents [7][8][9][10]. Nevertheless, several shortcomings can be discovered in their work. First, neither of them is applicable to solving multi-agent (more than two) distributed LP problems. Second, the secure protocols require enormous computation costs: even if the computational cost of Li et al.'s work [8] and Vaidya's work [7][9] includes a polynomial number of homomorphic encryptions, it still requires considerable time complexity for the total encryption. The efficiency should be greatly declined for large DisLP problems. Mangasarian [11]proposed a privacy-preserving formulation of a linear program over vertically partitioned constraint matrix while our approach is introduced to privately solve arbitrarily partitioned LP problems in this paper, and no formal security analysis is given in [11]. A secure third-party based protocol for LP was proposed by Du [10], however the LP problem is not addressed fully or formally and an optimal solution is not guaranteed. We will propose a secure and efficient DisLP approach to resolve the above limitations.

3 Preliminaries

In this section, we briefly review some definitions and properties related to LP problems.

3.1 Polyhedra

From the geometrical point of view, LP problems can be represented as polyhedra. We thus present some geometrical definitions for LP problems.

Definition 2 (Polyhedron of Linear Constraints). *A polyhedron $P \subseteq \mathbb{R}^n$ is the set of points that satisfy a finite number (m) of linear constraints $P = \{x \in \mathbb{R}^n : Ax \bowtie b\}$ where A is an $m \times n$ constraint matrix.*

Definition 3 (Convex Combination). *A point $x \in \mathbb{R}^n$ is a convex combination of a set $S \subseteq \mathbb{R}^n$ if x can be expressed as $x = \sum_i \lambda_i x^i$ for a finite subset $\{x^i\}$ of S and $\lambda > 0$ with $\sum_i \lambda_i = 1$.*

Definition 4 (Vertex). *A point $x^e \in P$ is a vertex of $P = \{x \in \mathbb{R}^n : Ax \bowtie b\}$ if it cannot be represented as a convex combination of two other points $x^i, x^j \in P$.*

Definition 5 (Ray in Polyhedron). *Given a non-empty polyhedron $P = \{x \in \mathbb{R}^n : Ax \bowtie b\}$, a vector $r \in \mathbb{R}^n, r \neq 0$ is a ray if $Ar \bowtie 0$.*

Definition 6 (Extreme Ray). *A ray r is an extreme ray of $P = \{x \in \mathbb{R}^n : Ax \bowtie b\}$ if there does not exist two distinct rays r^i and r^j of P such that $r = \frac{1}{2}(r^i + r^j)$.*

3.2 Dantzig-Wolfe Decomposition

Assume that we let x^i (size n vector) represent a vertex or extreme ray in the LP problem. Hence, every point inside the polyhedron can be represented by all the vertices and/or extreme rays using convexity combination (Minkowski's Representation Theorem [12]). Thus, a polyhedron P can be represented by another polyhedron $P' = \{\lambda \in \mathbb{R}^{|E|} : \sum_{i \in E} \delta_i \lambda_i = 1; \lambda \leq 0\}$ where

$$\delta_i = \begin{cases} 1 \text{ if } x^i \text{ is a vertex} \\ 0 \text{ if } x^i \text{ is an extreme ray} \end{cases} \tag{2}$$

Hence, the original LP problem (Equation 1) can be transformed to a master problem (Equation 3) using Dantzig-Wolfe Decomposition [12]. Assuming that x_i^j represents the extreme point or ray associated with λ_{ij}.

$$\begin{aligned} & max \quad \sum_j c_1^T x_1^j \lambda_{1j} + \cdots + \sum_j c_K^T x_K^j \lambda_{Kj} \\ & s.t. \begin{cases} \sum_j A_1 x_1^j \lambda_{1j} + \cdots + \sum_j A_K x_K^j \lambda_{Kj} \bowtie b_0 \\ \sum_j \delta_{1j} \lambda_{1j} \qquad\qquad\qquad\qquad = 1 \\ \qquad\qquad \ddots \\ \qquad\qquad\qquad \sum_j \delta_{Kj} \lambda_{Kj} = 1 \\ \lambda_1 \in \mathbb{R}^{|E_1|}, \ldots, \lambda_K \in \mathbb{R}^{|E_K|}, \delta_{ij} \in \{0, 1\}, i \in [1, K] \end{cases} \end{aligned} \tag{3}$$

As proven in [12], primal feasible points, optimal primal points, an unbounded rays, dual feasible points, optimal dual points and certificate of infeasibility in the *master problem* are equivalent to the *original problem*.

4 Revised Dantzig-Wolfe Decomposition

As shown in Equation 1, K-LP problem has a typical Block-angular structure, though the number of global constraints can be significantly larger than each agent's local constraints. Hence, we can solve the K-LP problem using Dantzig-Wolfe decomposition. In this section, we transform our K-LP problem to an anonymized (block-angular) format that preserves each agent's private input/output. We also show that the optimal solution for each agent's variables can be derived after solving the transformed problem.

4.1 K-LP Transformation

Du [13][10] and Vaidya [7] proposed a transformation approach for solving two-agent DisLP problems: transforming an $m \times n$ constraint matrix M (the objective vector c^T) to another $m \times n$ matrix $M' = MQ$ ($c'^T = c^T Q$) by post-multiplying an $n \times n$ matrix Q, solving the transformed problem and deriving the original solution. Meanwhile, Bednarz et al. [14] showed how to select transformation matrix Q. Following them, we let each agent P_i ($i \in [1, K]$) transform its local constraints B_i, its share of the global constraints A_i and its objective vector c_i using its own transformation matrix Q_i.

We let K agents transform A_i and B_i by Q_i individually for the following reason. Essentially, we extend a revised version of Dantzig-Wolfe decomposition to solve K-LP and ensures that the protocol is secure. Thus, an arbitrary agent should be chosen as the master problem solver whereas all agents (including the master problem solving agent) should solve the pricing problems. For transformed K-LP problem (Equation 4), we can let $\forall P_i$ ($i \in [1, K]$) send its transformed matrices/vector A_iQ_i, B_iQ_i, $c_i^TQ_i$ to another agent P_j ($j \in [1, K], j \neq i$) and let P_j solve P_i's transformed pricing problems. In this case, we can show that no private information can be learnt while solving the problems (The attack specified in [14] can be eliminated in our secure K-LP problem). Otherwise, if each agent solves its pricing problem, since each agent knows its transformation matrix, additional information might be disclosed from master solver to pricing problem solvers (this is further discussed in Section 5).

$$\max \quad \sum_{i=1}^{K} c_i^T Q_i y_i$$
$$s.t. \begin{cases} y_1 \in \mathbb{R}^{n_1} \\ y_2 \in \mathbb{R}^{n_2} \\ \vdots \\ y_K \in \mathbb{R}^{n_K} \end{cases} \begin{pmatrix} A_1Q_1 & \dots & A_KQ_K \\ B_1Q_1 & & \\ & \ddots & \\ & & B_KQ_K \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_K \end{pmatrix} \begin{matrix} \bowtie_0 \\ \bowtie_1 \\ \vdots \\ \bowtie_K \end{matrix} \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_K \end{pmatrix} \tag{4}$$

The K-LP problem can be transformed to another block-angular structured LP problem as shown in Equation 4. We can derive the original solution from the solution of the transformed K-LP problem using the following theorem.

Theorem 1. *Given the optimal solution of the transformed K-LP problem $y^* = (y_1^*, y_2^*, \dots, y_K^*)$, the solution $x^* = (Q_1y_1^*, Q_2y_2^*, \dots, Q_Ky_K^*)$ should be the optimal solution of the original K-LP problem.*

Proof. Suppose $x^* = (Q_1y_1^*, Q_2y_2^*, \dots, Q_Ky_K^*)$ is not the optimal solution of the original vertical LP problem. In this case, we have another vector $x' = (x_1', x_2', \dots, x_K')$ such that $c^Tx' > c^Tx^* \implies c_1^Tx_1' + \dots + c_K^Tx_K' > c_1^Tx_1^* + \dots + c_K^Tx_K^*$ where $Mx' \bowtie b$ and $x' \geq 0$. Let $y' = (y_1', \dots, y_K') = (Q_1^{-1}x_1', \dots, Q_K^{-1}x_K')$, thus we have $c_1^TQ_1y_1' + \dots + c_K^TQ_Ky_K' = c_1^TQ_1Q_1^{-1}x_1' + \dots + c_K^TQ_KQ_K^{-1}x_K' = c_1^Tx_1' + \dots + c_K^Tx_K'$.

Thus, $c_1^Tx_1' + \dots + c_K^Tx_K' = c_1^TQ_1y_1' + \dots + c_K^TQ_Ky_K' > c_1^Tx_1^* + \dots + c_K^Tx_K^* \implies c_1^TQ_1y_1' + \dots + c_K^TQ_Ky_K' > c_1^TQ_1Q_1^{-1}x_1^* + \dots + c_K^TQ_KQ_K^{-1}x_K^* \implies c_1^TQ_1y_1' + \dots + c_K^TQ_Ky_K' > c_1^TQ_1y_1^* + \dots + c_K^TQ_Ky_K^*$ (since $Q_1^{-1}x_1^* = y_1^*, \dots, Q_K^{-1}x_K^* = y_K^*$)

Hence, y' is a better solution than y^* which is a contradiction to that y^* is the optimal solution. Thus, Theorem 1 has been proven.

4.2 Righthand-side Value b Anonymization Algorithm

Besides protecting each party's share of the global constraint matrix A_i, B_i, solving the LP problems also requires the righthand side constants b in the constraints. Since b sometimes refers to the amount of limited resources (i.e. labors, materials) or some demands (i.e. the amount of one product should be no less than 10, $x_{ij} \geq 10$), they should not be revealed. We can anonymize b for each agent before transforming the constraint matrix and sending them to other agents.

Specifically, each agent P_i ($i \in [1, K]$) has two distinct constant vectors in the global and local constraints: b_0^i and b_i where $b_0 = \sum_{i=1}^{K} b_0^i$. Indeed, we can create artificial variables and equations to anonymize either b_0^i or b_i. For anonymizing b_0^i in the global constraints $\sum_{i=1}^{K} A_i x_i \bowtie_0 b_0^i$, each agent P_i can create a new artificial variable $s_{ij} = \eta_{ij}$ (fixed value) for the jth row ($j \in [1, m_0]$) of A_i. Hence, $m_0 \times n_i$ matrix A_i is expanded to a greater $m_0 \times (n_i + m_0)$ matrix as shown in Equation 5 ($A_i^1, \ldots, A_i^{m_0}$ denote the rows of matrix A_i).

$$A_i = \begin{pmatrix} A_i^1 \\ A_i^2 \\ \vdots \\ A_i^{m_0} \end{pmatrix} \implies A_i' = \begin{pmatrix} A_i^1 & s_{i1} & 0 & \ldots & 0 \\ A_i^2 & 0 & s_{i2} & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A_i^{m_0} & 0 & 0 & \ldots & s_{im_0} \end{pmatrix} \quad (5)$$

Algorithm 1: Righthand-side Value b Anonymization

```
Input  : $K$ honest-but-curious agents $P_1, \ldots, P_K$ where $P_i$ ($i \in [1, K]$) holds a set of variables $x_i$,
         $A_i(m_0 \times n_i$ matrix), $B_i(m_i \times n_i$ matrix), vectors $b_i$ (size $m_i$), $b_0^i$(size $m_0$), and $c_i$ (size $n_i$)
Output: anonymized $b' = \{b_0', b_1', \ldots, b_K'\}$ (size $m_0, m_1', \ldots, m_K'$) where $A_i, B_i, c_i$ are updated to
         $A_i'(m_0 \times n_i'$ matrix), $B_i'(m_i' \times n_i'$ matrix), $c_i'$ (size $n_i'$) ($i \in [1, K]$)

/* $A_i^j$ and $B_i^j$ denote the $jth$ row of $A_i$ and $B_i$ */
1  forall agent $P_i, i \in \{1, 2, \ldots, K\}$ do
2      generates a $m_0$-dimensional random vector $\eta_i$ ;
3      initializes $m_0$ new variables $s_i = \{s_{i1}, \ldots, s_{im_0}\}$ where $s_i = \eta_i$;
4      $(b_0^i)' \leftarrow b_0^i + \eta_i$;
5      for the $jth$ global constraint ($j \in [1, m_0]$) do
6          $A_i^j x_i \leftarrow A_i^j x_i + s_{ij}$ ;
7          $(b_0^i)_j \leftarrow (b_0^i)_j'$;
8      forall constraint $B_i^j x_i \bowtie_i^j b_i^j$ in $B_i x_i \bowtie_i b_i$ do
9          generates a linear equation using $\forall s_{ij} \in s_i$: $\sum_{\forall h} h_{ij} s_{ij} = \sum_{\forall j} h_{ij} \eta_{ij}$ where $h_{ij}$ is a random
           number;
10         $B_i^j x_i \bowtie_i^j b_i^j \leftarrow B_i^j x_i + \sum_{\forall j} h_{ij} s_{ij} \bowtie_i^j b_i^j + \sum_{\forall j} h_{ij} \eta_{ij}$;
11     generates $m_0$ linear independent equations: $\sum_{\forall j} r_{ij} s_{ij} = \sum_{\forall j} r_{ij} \eta_{ij}$ where random numbers $\forall r_{ij}$
       guarantee linear independence;
12     update them into local constraints: $B_i' x_i' \bowtie_i b_i' \leftarrow B_i x_i \bowtie_i b_i' \cup \sum_{\forall j} r_{ij} s_{ij} = \sum_{\forall j} r_{ij} \eta_{ij}$ ;
       /* permutate the variables and generate more artificial variables if
          necessary */
```

We thus have $(b_0^i)' \leftarrow b_0^i + \eta_i$ where $\eta_i = \{\forall j, \eta_{ij}\}$ (can be negative) is a random m_0-dimensional vector generated by agent P_i. Finally, each agent P_i creates additional m_0 linear independent local constraints $\sum_{\forall j} r_{ij} s_{ij} = \sum_{\forall j} r_{ij} \eta_{ij}$ using variables $\{\forall j, s_{ij}\}$ and associate them with constraints in $B_i x_i \bowtie_i b_i$ that ensure $s_i = \eta_i$ where $s_i = \{\forall j, s_{ij}\}$. Therefore, we have:

– the jth global constraint should be converted to $\sum_{i=1}^{K} A_i^j x_i + \sum_{i=1}^{K} s_{ij} \bowtie_0^j \sum_{i=1}^{K} (b_0^i)_j'$ where $(b_0^i)_j'$ represents the jth number in $(b_0^i)'$.
– additional local constraints ensure $\sum_{i=1}^{K} A_i x_i \bowtie_0 \sum_{i=1}^{K} b_0^i$ for a feasible K-LP problem since $\forall i, s_i = \eta_i$.

Besides b_0^i, we can anonymize b_i using a similar approach. P_i can use the same set of artificial variables s_i to anonymize b_i. By generating linear combination (not required to be linear independent) of the variables $s_i = \{\forall j, s_{ij}\}$, the left-hand side of the jth constraint in $B_i x_i \bowtie_i b_i$ can be updated: $B_i^j x_i \leftarrow B_i^j x_i + \sum_{\forall j} h_{ij} s_{ij}$ where h_{ij} is a random number. (the jth value in b_i is updated by $b_i^j \leftarrow b_i^j + \sum_{\forall j} h_{ij} \eta_{ij}$. If anonymizing b_i as above, adversaries may guess m_0 additional (linear independent) local constraints out of $m_i + m_0$ constraints from P_i's sub-polyhedron. The probability of guessing out m_0 linear independent constraints and calculating the values of the artificial variables is $\frac{m_0! m_i!}{(m_i+m_0)!}$ (if we standardize all the operational symbols $\bowtie_i$, guessing equations is choosing m_0 from $(m_i + m_0)$ constraints). However, the anonymization process should be prior to the matrix multiplication transformation, thus those m_0 equations include $n_i + m_0$ variables (coefficients of the non-artificial variables in these equations is transformed to non-zero). Hence, although the adversary knows m_0 linear independent equations, it is also impossible to figure out values η_i. Hence, b_i and b_0^i can be secure against adversaries. Algorithm 1 introduces the detailed steps of anonymizing b. Note: if any agent P_i requires higher privacy guarantee, P_i can generate more artificial variables for both b_0^i and b_i (A typical tradeoff between privacy and efficiency).

4.3 Revised Dantzig-Wolfe Decomposition

Dantzig-Wolfe decomposition was originally utilized to solve large-scale block-angular structured LP problems. However, for all the K-LP problems, we can appropriately partition the constraints into block-angular structure. Specifically, we can consider each agent's local constraints as the constraints of its pricing problems. By contrast, any constraint that is shared by at least two agents is regarded as the global constraint. Even if A_i may have more rows than B_i, the constraints are still block-angular partitioned.

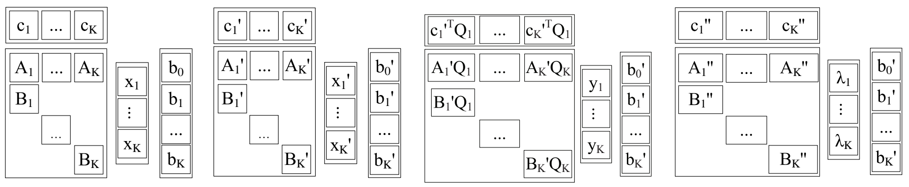
Furthermore, after locally anonymizing b and transforming the blocks, each agent still has its local constraints block $B_i' Q_i$ and the global constraints share $A_i' Q_i$. Hence, we can solve the transformed K-LP problem using Dantzig-Wolfe decomposition. We thus denote the entire process as Revised Dantzig-Wolfe Decomposition:

Definition 7 (Revised Dantzig-Wolfe Decomposition). *A secure and efficient approach to solving K-LP problems that includes the following stages: anonymizing b by each agent, transforming blocks by each agent and solving the transformed K-LP problem using Dantzig-Wolfe Decomposition.*

According to Equation 3, the Dantzig-Wolfe representation of the transformed K-LP problem is:

$$
\begin{aligned}
&max \quad \sum_j c_1'^T Q_1 y_1^j \lambda_{1j} + \cdots + \sum_j c_K'^T Q_K y_K^j \lambda_{Kj} \\
&s.t. \begin{cases}
\sum_{\forall j} A_1' Q_1 y_1^j \lambda_{1j} + \cdots + \sum_{\forall j} A_K' Q_K y_K^j \lambda_{Kj} \bowtie_0 b_0' \\
\sum_{\forall j} \delta_{1j} \lambda_{1j} \qquad\qquad\qquad\qquad = 1 \\
\qquad\qquad \ddots \\
\qquad\qquad\qquad \sum_{\forall j} \delta_{Kj} \lambda_{Kj} = 1 \\
\lambda_1 \in \mathbb{R}^{|E_1'|}, \ldots, \lambda_K \in \mathbb{R}^{|E_K'|}, \delta_{ij} \in \{0,1\}, i \in [1, K]
\end{cases}
\end{aligned}
\tag{6}
$$

where $\forall i \in [1, K], c_i \subseteq c_i', A_i \subseteq A_i', B_i \subseteq B_i'$ (c_i', A_i', B_i' are expanded from c_i, A_i, B_i for anonymizing b).



(a) Original K-LP (b) Anonymized b (c) Transformed Problem (d) DW (Equation 6)

Fig. 2. Revised Dantzig-Wolfe Decomposition for K-LP Problem

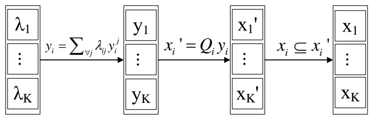


Fig. 3. Solution Transformation After Solving K-LP Problem

Figure 2 presents the three steps of Revised Dantzig-Wolfe Decomposition. Furthermore, after solving the problem, each agent P_i should obtain an optimal solution $\lambda_i = \{\forall j, \lambda_{ij}\}$. Figure 3 shows the process of deriving each agent's optimal solution for the original K-LP problem. Specifically, in step 1, the optimal solutions for each agent's transformed problem can be derived by computing the convexity combination of all vertices/extreme rays y_i^j: $y_i = \sum_{\forall j} \lambda_{ij} y_i^j$. In step 2 ($x_i' = Q_i y_i$)[4], the optimal solution of the original problem with anonymized b can be derived by left multiply Q_i for each agent (Theorem 1). In step 3, each agent can extract its individual optimal solution in the K-LP problem by excluding the artificial variables (for anonymizing b) from the optimal solution of x_i'.

The advantages of this sort of decomposition are: the pricing problems can be solved independently; the master problem solver does not need to get into the details on how the proposals are generated; if the subproblems have special structure (e.g., perhaps one is a transportation problem) then those specialized solution techniques can be used. This also makes it easier to preserve privacy if the large problem could be solved without knowing the precise solutions of the pricing problems. Particularly, we can let an arbitrary agent formulate and solve the transformed master problem (Equation 6). However, the efficiency and security is not good enough for large-scale problems since the number of vertices/extreme rays are $\frac{n_i'!}{m_i'!(n_i'-m_i')!}$ for each agent and all the variables should be sent to the master problem solver (assuming that the K-LP problem is standardized with slack variables before transformation). In Section 5, the K-agent Column

[4] Apparently, if $y_i = 0$ and we have $x_i' = Q_i y_i$, x_i' should be 0 and revealed to other agents. However, y_i includes some transformed variables that is originally the value-fixed but unknown artificial variables for anonymizing b. Hence, x_i' cannot be computed due to unknown Q_i and non-zero y_i (the situation when the optimal solution in y_i is 0, is not known to the holder other than P_i), and this possible privacy leakage can be resolved.

Generation Protocol can handle this problem and the detailed security proof and communication costs are also given there.

5 Secure Column Generation Protocol for K-LP Problems

While solving K-LP by revised Dantzig-Wolfe decomposition, it is fair to all K agents. Hence, we assume that an arbitrary agent can be the master problem solver. Each agent's subproblems can be solved by another agent while the problem is iteratively solving (the pricing problems and the solvers can be randomly permutated). To simplify the notation, we assume that P_1 solves the restricted master problems (RMP), P_i sends $A_i'Q_i$, $B_i'Q_i$, $c_i'^TQ_i$, $(b_0^i)'$ and b_i' to P_{i+1} that solves P_i's pricing problems (P_1 solves P_K's pricing problems). In this section, we present our K-agent column generation protocol with security proof and computation cost analysis.

5.1 Solving RMP by an Arbitrary Agent

As mentioned in Section 4, the full master problem in the revised Dantzig-wolfe decomposition includes $\sum_{i=1}^{K} \frac{n_i'!}{m_i'!(n_i'-m_i')!}$ variables. However, it is not necessary to involve all the vertices/extreme rays simply because a fairly small number of constraints in the master problem might result in many non-basis variables in the full master problem. Hence, restricted master problem (RMP) of the transformed K-LP problem is introduced to improve efficiency.

We let $[c_i] = (\forall j \in [1, \frac{n_i'!}{m_i'!(n_i'-m_i')!}], c_i'^TQ_iy_i^j)$ and $[A_i] = (\forall j \in [1, \frac{n_i'!}{m_i'!(n_i'-m_i')!}], A_i'Q_iy_i^j)$. For RMP, we denote the coefficients in the master problem restricted to $\mathbb{R}^{|\widehat{E_1}|}, \dots, \mathbb{R}^{|\widehat{E_K}|}$ as $\widehat{c_i}$, $\widehat{A_i}$, $\widehat{y_i}$, $\widehat{\delta}$ and $\widehat{\lambda}$. Specifically, some of the variables λ for all agents are initialized to non-basis 0. τ_i denotes the number of vertices in P_i's pricing problem that has been proposed to the master solver where $\forall i \in [1, K], \tau_i \leq \frac{n_i'!}{m_i'!(n_i'-m_i')!}$. Hence, we represent the RMP as below:

$$max \;\; \widehat{c_1}^T\widehat{\lambda_1} + \dots + \widehat{c_K}^T\widehat{\lambda_K}$$
$$s.t. \begin{cases} \widehat{A_1}\widehat{\lambda_1} + \dots + \widehat{A_K}\widehat{\lambda_K} \bowtie_0 b_0' \\ \sum_{j=1}^{\tau_1} \delta_{1j}\lambda_{1j} \;\;= 1 \\ \vdots \\ \sum_{j=1}^{\tau_K} \delta_{Kj}\lambda_{Kj} = 1 \\ \lambda_1 \in \mathbb{R}^{|\widehat{E_1}|}, \dots, \lambda_K \in \mathbb{R}^{|\widehat{E_K}|}, \delta_{ij} \in \{0,1\}, i \in [1, K] \end{cases} \tag{7}$$

Lemma 1. *Solving the RMP of a K-LP problem Reveals only:*

- *the revised DW representation of the K-LP problem;*
- *the optimal solution of the revised DW representation;*
- *the total payoff (optimal value) of each agent;*

Proof. RMP is a special case of the full master problem where some variables in $\forall i, \lambda_i$ are fixed to be non-basis (not sent to the RMP solver P_1). Hence, the worse case is that all the columns of the master problem are required to formulate the RMP. We thus discuss the privacy leakage in this case.

We look at the matrices/vectors that are acquired by P_1 from all other agents P_i where $\forall i \in [1, K]$. Specifically, $[c_i] = (\forall j \in [1, \frac{n'_i!}{m'_i!(n'_i-m'_i)!}], c_i'^T Q_i y_i^j)$ and $[A_i] = (\forall j \in [1, \frac{n'_i!}{m'_i!(n'_i-m'_i)!}], A'_i Q_i y_i^j)$ should be sent to P_1. At this time, $[c_i]$ is a vector with size $\frac{n'_i!}{m'_i!(n'_i-m'_i)!}$ and $[A_i]$ is an $m_0 \times \frac{n'_i!}{m'_i!(n'_i-m'_i)!}$ matrix. The jth value in $[c_i]$ is equal to $c_i'^T Q_i y_i^j$, and the jth column in matrix $[A_i]$ is equal to $A'_i Q_i y_i^j$.

Since P_1 does not know y_i^j and Q_i, it is impossible to calculate or estimate the (size n'_i) vector c'_i and sub-matrices A_i and B_i. Specifically, even if P_1 can construct $(m_0 + 1) \cdot \frac{n'_i!}{m'_i!(n'_i-m'_i)!}$ non-linear equations based on the elements from $[c_i]$ and $[A_i]$, the number of unknown variables in the equations (from c'_i, A'_i, Q_i[5] and $\forall j \in [1, \frac{n'_i!}{m'_i!(n'_i-m'_i)!}], y_i^j$) should be $n'_i + m_0 n'_i + n'_i + n'_i \cdot \frac{n'_i!}{m'_i!(n'_i-m'_i)!}$. Due to $n'_i >> m_0$ in linear programs, we have $n'_i + m_0 n'_i + n'_i + n'_i \cdot \frac{n'_i!}{m'_i!(n'_i-m'_i)!} >> (m_0 + 1) \cdot \frac{n'_i!}{m'_i!(n'_i-m'_i)!}$. Thus, those unknown variables in c'_i, A'_i, Q_i and $\forall j \in [1, \frac{n'_i!}{m'_i!(n'_i-m'_i)!}], y_i^j$ cannot be derived from the non-linear equations. As a result, P_1 learns nothing about A_i, c_i, b_0^i (anonymized) and $B_i x_i \bowtie_i b_i$ (since vertices/extreme rays $\forall j, y_i^j$ are unknown) from any agent P_i.

By contrast, while solving the problem, P_1 formulates and solves the RMPs. P_1 thus knows the primal and dual solution of the RMP. In addition, anonymizing b and transforming c_i, A_i and B_i does not change the total payoff (optimal value) of each agent, the payoffs of all values are revealed to P_1 as well (Vaidya's protocol [7] also reveals this payoff). Nevertheless, the private constraints and the optimal solution cannot be inferred based on this limited disclosure.

Hence, solving the RMPs is secure.

5.2 Solving Pricing Problems by Peer-agent

While solving the K-LP problem by the column generation algorithm(CGA), in every iteration, each agent's pricing problem might be formulated to test that whether any column of the master problem (vertex/extreme ray of the corresponding agent) should be proposed to the master problem solver or not. If any agent's pricing problem cannot propose column to the master solver in the previous iterations, no pricing problem is required for this agent anymore. As discussed in Section 4.1, we permutate the pricing problem owners and the pricing problem solvers where private information can be protected via transformation. We now introduce the details of solving pricing problems and analyze the potential privacy loss.

Assuming that an honest-but-curious agent $P_{i+1} (i \in [1, K])$ has received agent P_i's (if $i = K \implies i+1 = 1$) variables y_i, transformed matrices/vector $A'_i Q_i, B'_i Q_i, c_i'^T Q_i$

[5] As described in [14], Q_i should be a monomial matrix, thus Q_i has n'_i unknown variables located in $n_i'^2$ unknown positions.

and the anonymized vectors $b'_i, (b^i_0)'$ (as shown in Figure 2(c)). Agent P_{i+1} thus formulates and solves agent P_i's pricing problem.

In every iteration, after solving RMP (by P_1), P_1 sends the optimal dual solution $\{\pi, \mu_i\}$ to P_{i+1} ($\mu_i = \{\forall j, (\mu_i)_j\}$) if the RMP is feasible. The reduced cost d_{ij} of variable λ_{ij} for agent P_i can be derived as:

$$d_{ij} = (c'^T_i Q_i - \pi A'_i Q_i) y^j_i - \begin{cases} (\mu_i)_j & \text{if } y^j_i \text{ is a vertex} \\ \;\;0 & \text{if } y^j_i \text{ is an extreme ray} \end{cases} \tag{8}$$

Therefore, P_{i+1} formulates P_i's pricing problem as:

$$\begin{aligned} max \quad & (c'^T_i Q_i - \pi A'_i Q_i) y_i \\ s.t. & \begin{cases} B'_i Q_i y_i \bowtie b'_i \\ y_i \in \mathbb{R}^{n'_i} \end{cases} \end{aligned} \tag{9}$$

Lemma 2. *If P_{i+1} solves P_i's transformed pricing problems, P_{i+1} learns only:*

- *the feasibility of P_i' block sub-polyhedron $B_i x_i \bowtie_i b_i$;*
- *dual optimal values (π, μ_i) of the RMP for transformed K-LP;*

Proof. Since we can let another arbitrary peer-agent solve any agent's pricing problems (fairness property): assuming that P_{i+1} solves P_i's pricing problem ($i = K \implies i + 1 = 1$). Similarly, we first look at the matrices/vectors acquired by P_{i+1} from P_i: size n'_i vector $c'^T_i Q$, $m'_i \times n'_i$ matrix $B'_i Q_i$ and $m_0 \times n'_i$ matrix $A'_i Q_i$. The jth value in $c'^T_i Q_i$ is equal to $c'^T_i Q^j_i$ (Q^j_i denotes the jth column of Q_i), and the value of the kth row and the jth column in $A'_i Q_i$ (or $B'_i Q_i$) is equal to the scalar product of the kth row of A'_i (or B'_i) and Q^j_i.

Since P_{i+1} does not know Q_i, it is impossible to calculate or estimate the (size n'_i) vector c'_i and matrices A'_i (or A_i) and B'_i (or B_i). Specifically, even if P_{i+1} can construct $(m_0 + m'_i + 1)n'_i$ non-linear equations based on the elements from $c'^T_i Q_i$, $A'_i Q_i$ and $B'_i Q_i$, the number of unknown variables in the equations (from c'_i, A'_i, B_i and Q_i) should be $n'_i + m_0 n'_i + m'_i n'_i + n'_i$. Due to $n'_i >> 0$ in linear programs, we have $n'_i + m_0 n'_i + m'_i n'_i + n'_i >> (m_0 + m'_i + 1)n'_i$. Thus, those unknown variables in c'_i, A'_i, B_i and Q_i cannot be derived from the non-linear equations. [6]

Hence, P_{i+1} learns nothing about A_i, B_i, c_i, b^i_0 (anonymized) and b_i (anonymized) from P_i if P_{i+1} solves P_i's pricing problems.

By contrast, before solving the pricing problem, P_{i+1} should acquire the some dual optimal values of the RMP (only π and μ_i). P_{i+1} thus knows the dual optimal solution of the RMP related to the convexity combination represented global constraints

[6] Note: Bednarz et al. [14] proposed a possible attack on inferring Q with the known transformed and original objective vectors ($C^T Q$ and C^T) along with the known optimal solutions of the transformed problem and the original problem ($y*$ and $x* = Qy*$). However, this attack only applies to the special case of DisLP in Vaidya's work [7] where one party holds the objective function while the other party holds the constraints. In our protocol, P_i sends $C'^T_i Q_i$ to P_{i+1}, but C'^T_i is unknown to P_{i+1}, hence it is impossible to compute all the possibilities of Q_i by P_{i+1} in terms of Bednarz's approach. In addition, the original solution is not revealed as well. It is impossible to verify the exact Q_i by P_{i+1} following the approach in [14].

(π) and the constraints $\sum_{\forall j} \delta_{ij}\lambda_{ij} = 1$ (μ_i). However, P_{i+1} cannot learn the actual pricing problem since everything in the K-LP is transformed in the RMP. Furthermore, if the polyhedron $B_i' Q_i y_i \bowtie_i b_i'$ is infeasible, we have: polyhedron $B_i' x_i \bowtie_i b_i'$ is also infeasible (Theorem 2). Hence, the specific agent with the infeasible local constraints should be spotted (Actually, this should be revealed in any case). However, the private constraints and the meanings of the concrete variables cannot be inferred with this information. (For more rigorous privacy protection, we can randomly permutate the agents.)

Hence, solving the Pricing Problems by another arbitrary agent is secure.

Theorem 2. *The polyhedra $B_i x_i \bowtie_i b_i$ and $B_i Q_i y_i \bowtie_i b_i$ have the same feasibility where $i \in [1, K]$.*

Proof. We prove this equivalence in two facts:

First, suppose that the polyhedron $B_i x_i \bowtie_i b_i$ is feasible and one of its feasible solutions is x_i. Now, we have all the constraints (equalities or inequalities) in B_i that satisfy $B_i x_i \bowtie_i b_i$. Let $x_i = Q_i y_i$, hence $B_i Q_i y_i \bowtie_i b_i$ are all satisfied and the polyhedron $B_i Q_i y_i \bowtie_i b_i$ is feasible.

On the contrary, suppose that the polyhedron $B_i Q_i y_i \bowtie_i b_i$ is feasible and one of its feasible solutions is y_i. Now, we have all the constraints (equalities or inequalities) in $B_i Q_i$ that satisfy $B_i Q_i y_i \bowtie_i b_i$. Let $y_i = Q_i^{-1} x_i$, hence $B_i x_i \bowtie_i b_i$ are all satisfied and the polyhedron $B_i x_i \bowtie_i b_i$ is feasible.

Thus, Theorem 2 has been proven.

5.3 Secure K-agent Column Generation Algorithm (SCGA)

In the standard column generation algorithm [12], the RMP solver will ask the pricing problem solvers for proposals and choose a combination of proposals that maximizes global profits while meeting all the constraints in the RMP. Figure 4 demonstrates our secure K-agent column generation protocol where the steps represent:

1. $\forall i \in [1, k]$, P_i sends $A_i' Q_i$, $B_i' Q_i$, $(b_0^i)'$, b_i' and $c_i'^T Q_i$ to P_{i+1}.
2. P_1 solves a RMP problem.
3. P_1 distributes dual values (π, μ_i) to P_{i+1}.
4. P_{i+1} solves P_i's pricing problems.
5. P_{i+1} proposes P_i's column to P_1 if necessary.

Practically, the main drawback of this approach is in possible convergence problems. Normally, this method gets very good answers quickly, but it requires a lot of time to find the optimal solution. The subproblems may continue to generate proposals only slightly better than the ones before. Thus, we might have to stop with a near-optimal solution for efficiency reasons if necessary [12]. Specifically, if the RMP is feasible and the pricing problems are all feasible and bounded, P_1 can calculate a new upper bound (dual value) of the master problem $\hat{z} = z^* + \sum_{i=1}^{K} (z_i^* - \mu_i)$. If $\hat{z} < \bar{z}^*$, update the best known dual value $\bar{z}^* \leftarrow \hat{z}$. P_1 thus compute the optimal gap $d = \bar{z}^* - z^*$ and the relative optimal gap $d' = \frac{d}{1+|z^*|}$. If the gap is tolerable, we stop the protocol where the optimal solution of the current RMP is near-optimal. In case of near-optimal tolerance, all the optimal values of the pricing problems $\forall i \in [1, K], z_i^*$ should be sent to P_1 along with the proposed column. However, the protocol is still secure in semi-honest model.

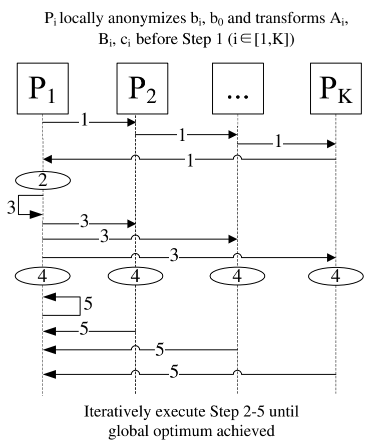

P_i locally anonymizes b_i, b_0 and transforms A_i, B_i, c_i before Step 1 (i∈[1,K])
P_1
P_2
...
P_K
1
2
3
4
5
Iteratively execute Step 2-5 until global optimum achieved

Fig. 4. Secure K-agent Column Generation Protocol

Theorem 3. *The K-agent Column Generation Protocol is secure in Semi-honest model.*

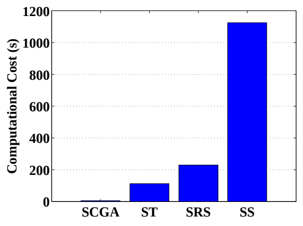
Proof. As proven in Lemma 1 and 2, solving RMPs and pricing problems is secure for all K honest-but-curious agents. Since our K-agent Column Generation Protocol the repeated steps of solving transformed RMPs and pricing problems, it is straightforward to show that the protocol is secure against semi-honest adversaries.

5.4 Communication Cost Analysis

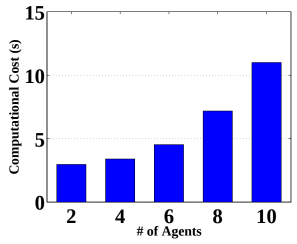
Our secure column generation protocol is mainly based on local transformation rather than cryptographic encryption that dominates the cost in current privacy-preserving DisLP techniques[10][8][7][9]. Hence, our approach significantly outperforms the above work on communication costs, especially in large-scale problems. Specifically, the size of the constraints matrix (all the constraints) should be $(m_0 + \sum_{i=1}^{K} m_i) \times \sum_{i=1}^{K} n_i$. After anonymizing b, the constraint matrix is enlarged to $(m_0 + \sum_{i=1}^{K} m'_i) \times \sum_{i=1}^{K} n'_i$. Each pair of matrices A'_i, B'_i is locally transformed. Besides solving the LP problem, only one-time $(m_0 + m'_i + 1)n'_i$ scalar product computation (transforming c', A'_i, B'_i) is required for each agent since anonymizing b does take ignorable computational cost (generating random numbers and equations). For large-scale block-angular structured problems, column generation algorithm has been proven to be more efficient than some standard methods (i.e. simplex or revised simplex algorithm)[15][12]. As discussed in Section 1, K-LP problem is a typical block-angular structured LP problem (distributed among K agents). Hence, the communication cost of our secure column generation algorithm is tiny and negligible.

6 Experiments

We implemented the secure column generation algorithm (SCGA) for solving K-LP problems. Specifically, we present two groups of results: 1. the performance comparison for all secure (two-agent) DisLP methods. 2. the performance of SCGA on varying number of agents where each agent has 15 variables. All the experiments were carried on an HP machine with Intel Core 2 Duo CPU 3GHz and 3G RAM.



(a) Secure (Two-agent) DisLP Methods



(b) Multi-agent SCGA

Fig. 5. Experimental Results (Near-optimal Tolerance Parameter=10^{-6})

To compare all secure DisLP methods, we generate 10 LP problems with 50 variables and 30×50 constraint matrix (not very dense) and run 4 algorithms for all 10 problems. Specifically, we assume that two agents collaboratively solve the LP problems where each agent holds 25 distinct variables. The number of local constraints for each agent and the number of global constraints are determined by the structure of 10 different 30×50 constraint matrix (we guarantee that every agent has at least one local constraints via the density of the constraint matrix). Before collaboratively solving the problem, each agent anonymizes the right-hand value and transforms the matrices/vector (the LP problems should be expanded a little bit). Figure 5(a) demonstrates the average runtime (10 LP problems) of SCGA, Secure Transformation (ST)[7], Secure Revised Simplex Method (SRS)[9] and Secure Simplex Method (SS) [8]. It is quite clear that the efficiency of SCGA significantly outperforms other algorithms in secure K-LP problems.

Furthermore, we run another group of experiments for validating the performance of SCGA on multiple agents. We generate different size of K-LP problems by assuming that each agent holds 15 variables and 5 local constraints. We let the number of global constraints be 10, thus the constraint matrix becomes $(5K + 10) \times 15K$. Hence, we run SCGA for different number of agents $K \in \{2, 4, 6, 8, 10\}$. The total computational cost (including anonymization, transformation and solving the problems) on varying K is shown in Figure 5(b). Thus, our SCGA exhibits great scalability for securely solving increasing scale of K-LP problems.

7 Conclusion and Future Work

DisLP problems allow collaborative agents to improve their global maximum profit (or save their global minimum cost). However, the private constraints (input) and solutions (output) of distributed agents might be revealed among them while solving the DisLP problem. In this paper, we have introduced an extremely efficient protocol to solve K-agent DisLP problems with limited disclosure. Our protocol is robust against semi-honest adversaries and is fair to all agents. In the future, we also plan to make the protocol resilient to malicious adversaries by making it incentive compatible.

References

1. E. Turban, R. K. Rainer, and R. E. Potter, "Introduction to information technology, chapter 2nd, information technologies: Concepts and management," *John Wiley and Sons, 3rd edition*.
2. M. Yokoo, E. H. Durfee, T. Ishida, and K. Kuwabar, "Distributed constraint satisfaction for formalizing distributed problem solving," in *In Proceedings of International Conference on Distributed Computing Systems*, pp. 614–621, 1992.
3. P. J. Modi, W.-M. Shen, M. Tambe, and M. Yokoo, "An asynchronous complete method for distributed constraint optimization," in *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, (New York, NY, USA), pp. 161–168, ACM Press, 2003.
4. M.-C. Silaghi and V. Rajeshirke, "The effect of policies for selecting the solution of a discsp on privacy loss," in *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 1396–1397, 2004.
5. M. C. Silaghi and D. Mitra, "Distributed constraint satisfaction and optimization with privacy enforcement," *iat*, vol. 00, pp. 531–535, 2004.
6. K. Suzuki and M. Yokoo, "Secure generalized vickrey auction using homomorphic encryption," *Lecture Notes in Computer Science*, vol. 2742.
7. J. Vaidya, "Privacy-preserving linear programming," in *SAC*, pp. 2002–2007, 2009.
8. J. Li and M. J. Atallah, "Secure and private collaborative linear programming," in *Collaborative Computing: Networking, Applications and Worksharing, 2006. CollaborateCom 2006. International Conference on*, pp. 1 –8, 2006.
9. J. Vaidya, "A secure revised simplex algorithm for privacy-preserving linear programming," in *AINA '09: Proceedings of the 23rd IEEE International Conference on Advanced Information Networking and Applications*, 2009.
10. W. Du, *A Study of Several Specific Secure Two-party Computation Problems*. PhD thesis, Purdue University, West Lafayette, Indiana, 2001.
11. O. L. Mangasarian, "Privacy-preserving linear programming," *Optimization Letters*, vol. 5, no. 1, pp. 28–34, 2010.
12. J. R. Tebboth, *A Computational Study of Dantzig-Wolfe Decomposition*. PhD thesis, University of Buckingham, 2001.
13. W. Du and M. J. Atallah, "Privacy-preserving cooperative scientific computations," in *In Proceedings of the 14th IEEE Computer Security Foundations Workshop*, pp. 273–282, 2001.
14. A. Bednarz, N. Bean, and M. Roughan, "Hiccups on the road to privacy-preserving linear programming," in *Proceedings of the 8th ACM workshop on Privacy in the electronic society*, WPES '09, (New York, NY, USA), pp. 117–120, ACM, 2009.
15. G. L. Nemhauser and L. A. Wolsey, *Integer and combinatorial optimization*. New York, NY, USA: Wiley-Interscience, 1988.