A Genetic Algorithm for Privacy Preserving Combinatorial Optimization

Jun Sakuma
Tokyo Institute of Technology
4259 Nagatsuta Midori-ku
Yokohama Japan
jun@fe.dis.titech.ac.jp

Shigenobu Kobayashi
Tokyo Institute of Technology
4259 Nagatsuta Midori-ku
Yokohama Japan
kobayasi@dis.titech.ac.jp

ABSTRACT

We propose a protocol for a local search and a genetic algorithm for the distributed traveling salesman problem (TSP). In the distributed TSP, information regarding the cost function such as traveling costs between cities and cities to be visited are separately possessed by distributed parties and both are kept private each other. We propose a protocol that securely solves the distributed TSP by means of a combination of genetic algorithms and a cryptographic technique, called the secure multiparty computation. The computation time required for the privacy preserving optimization is practical at some level even when the city-size is more than a thousand.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Algorithms

Keywords

Privacy, TSP, GA, Privacy-Preserving Optimization

1. INTRODUCTION

The delivery route decision, the production scheduling and the procurement planning are fundamental problems, which are optimized in order to improve the correspondence speed to the customer or to shorten the cycle time in the supply chain management (SCM) [1]. When the SCM is developed between two or more enterprises, information that relates to the stock, the production schedule, the demand forecast is required to be shared between enterprises.

EDI (Electronic Data Interchange), the standardized data exchange format over the network, is often used to support the convenient ant prompt information sharing. Information sharing enhances the availability of SCM largely. However, in exchange for the benefit, there also exists a risk of leaking confidential information and the shared information must be managed under prudent control[2]. In the process of the delivery route decision or the production schedule decision, the combinatorial optimization plays a key role. In this paper, we focus on the distributed combinatorial optimization problem where information that relates to the cost function to be optimized is private and distributed among two or more parties. For intuitive understanding of the combinatorial optimization including private information, we show a simple scenario of the Traveling Salesman Problem (TSP).

Scenario: Let there be two shipping companies E_A, E_B and a client E_C. E_C tries to request the delivery of freights to point $F_1, ..., F_n$ to one of the two shipping companies. In order to choose a shipping company that offers a better solution, client E_C tries to compare delivery costs for optimized routes offered by E_A and E_B. However, E_C cannot reveal the delivery points to both E_A and E_B before contracting. Delivery costs between any two cities are different in E_A and E_B. Also, the costs are confidential each other and they cannot be revealed to other party. How can E_C make decision without revealing their confidential information?

For the optimization of the TSP, delivery costs between any two points and delivery points are required to be shared. However, as shown in this scenario, if they are confidential, this problem cannot be solved straightforwardly. To resolve such a conflicted situation, technologies called *Secure Multiparty Computation* (SMC) [3] are utilized. In SMC, parties with private information mask them with some cryptography or random values in the form that necessary computations can be processed. Then, masked information is exchanged and necessary computations are processed. Results are shared by canceling masks after the computation.

A few studies have been made on *Privacy-Preserving Optimization* (PPO). A depth first search for distributed constraint optimization problems has been proposed in [4]. A protocol for solving the generalized Vickley auction based on dynamic programming has also been proposed in [5]. These algorithms are designed based on deterministic algorithms that guarantee to reach the optimum in polynomial time.

Although meta-heuristics such as the Local Search (LS) or the Genetic Algorithm (GA) do not guarantee to be completed in polynomial time, they are considered to be appropriate for a general solver of PPO with respect to following reasons. Primarily, meta-heuristics are generally designed to be independent to problems, given some neighborhood structure of the problem. PPO algorithms described pre-

GECCO'07, July 7–11, 2007, London, England, United Kingdom.

viously solve some specific problem efficiently, however, a new protocol must be invented to solve other new problem. Meta-heuristics can be applied to various optimization problems with some modification of the neighborhood operator for the problem. Therefore, meta-heuristics are expected to work as a general PPO engine.

Secondly, the information regarding objective functions that meta-heuristics require is often limited to the rank. For example, most of GAs generally choose individuals to be survived based on the rank. The rank is computed from a series of paired comparisons such as whether $f(x_1) > f(x_2)$, where x_1, x_2 are individuals and $f(\cdot)$ is an objective function. Therefore, even when cost values $f(x_1), f(x_2)$ are not provided, the optimization process thereof works properly. This design simplicity is considered to be suitable for privacy preservation.

Considering above, we propose a LS and a GA that securely solves the distributed combinatorial optimization problem whose objective function is represented as the scalar product. As shown later, the distributed combinatorial optimization problem with this representation includes many of typical ones, such as the TSP, the Quadratic Assignment Problem (QAP), the Knapsack Problem and so on.

The rest of this paper is organized as follows. Section 2 defines the distributed TSP and its privacy. Section 3 proposes a novel protocol for comparing two scalar products privately and the security proof of the protocol is shown. In section 4, a protocol for solving distributed TSP is designed based on a scalar product comparison protocol. Section 5 shows experimental results. Section 6 presents our concluding remarks.

2. PROBLEM DEFINITION

2.1 Scalar Product Representation for Combinatorial Optimization

We show a few example of combinatorial optimization whose objective functions are represented as the scalar product.

TSP: Let $G = (V, E)$ be an undirected graph and $\mathcal{F}$ be the set of all Hamiltonian cycles (referred to as tours) in G. Let the number of city be $|V| = n$. For each edge $e_{i,j} \in E$, a cost connecting node i and node j, $y_{i,j}$, is prescribed. Then the TSP is to find a tour such that the sum of the cost of included edges is as small as possible. The permutation representation or the edge representation are used to describe tours in general. However, for the convenience of the protocol description, we introduce indicator variables.

Let $x = (x_{1,2}, ..., x_{1,n}, x_{2,3}, ..., x_{2,n}, ..., x_{n-1,n})$ be a tour vector where $x_{i,j}$ are indicator variables such that

$$x_{i,j} = \begin{cases} 1 & e_{i,j} \text{ is included in the tour,} \\ 0 & \text{otherwise.} \end{cases}$$

The cost can be written as a vector $y = (y_{1,2}, ..., y_{n-1,n})$ similarly. The number of elements of tour vector x and cost vector y are $d = n(n-1)/2$. For simplicity, we describe the i-th element of x and y as x_i and y_i, respectively. With this representation, the objective function of TSP is written in the form of the scalar product $f(x) = \sum_{i=1}^{d} x_i y_i = x \cdot y$.

Knapsack problem: The objective function and its constraint of the knapsack problem are also represented as the scalar product. Let there be d items. For each item, a value

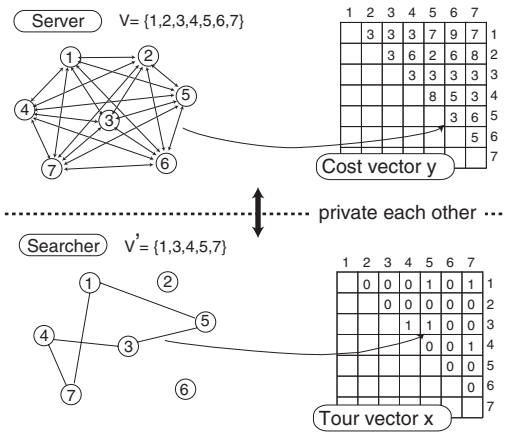

Figure 1: Distributed TSP

y_i and a weight z_i are prescribed. Then the knapsack problem is to find a combination of items such that the sum of the value of chosen items is as large as possible while the sum of the weight does not exceed the prescribed volume of the knapsack, v. Let $x = (x_1, ..., x_n)$ be a combination of item where x_i is an indicator variable such that

$$x_{i,j} = \begin{cases} 1 & i\text{-th item is included in the combination,} \\ 0 & \text{otherwise.} \end{cases}$$

Then, the objective function is written as a scalar product $f(x) = \sum_{i=1}^{d} x_i y_i = x \cdot y$. The constraint is also written as a scalar product, $\sum_{i=1}^{d} x_i z_i = x \cdot z \leq v$.

The objective function of the QAP and the VRP are also represented with the scalar product similarly. In latter section, we focus only on the TSP, however, please notice that following discussions are generally the case with problems described with the scalar product representation.

2.2 Distributed TSP

The distributed TSP and its privacy are defined. When $V' \subseteq V$ is arbitrarily chosen such that V' includes $n' \leq n$ cities, a TSP that routes n' cities is defined. If either y or V' is not private, there exists no privacy concern because conventional methods solve the problem by gathering y and V' in one party. If both of y and V' are private, it cannot be solved straightforwardly. We introduce a data partitioning model which defines how y and V' are distributed.

The simplest distributed TSP, which is referred to as $(1,1)$-TSP, is described(figure 1). In $(1,1)$-TSP, one party holds a private V' (referred to as *searcher*) and the other party holds a private y (referred to as *server*) separately. In this problem, the searcher searches for a local optimal tour that routes all cities in V' while the server works to help the computation of the searcher without revealing y. The scenario in introduction corresponds to $(1,1)$-TSP where two companies privately possess different cost vectors and the client tries to compare the evaluated cost of optimal routes.

When y is partitioned among k parties like $(y_1, ..., y_k)$, $\sum_i y_i = y$, this model is referred to as $(k,1)$-TSP. When V' is partitioned among k parties like $(V'_1, ..., V'_k)$, $V' = \cup_{i=1}^{k} V'_i)$, this model is referred to as $(1,k)$-TSP. Apparently, $(1,1) - TSP$ is a special case of them. Our protocol mainly focuses

on $(1,1)-$TSP to avoid complicated formulations. Although it is generalized to $(k,1)$-TSP with a simple extension, $(1,k)$-TSP is not considered in this paper.

To define the privacy in the distributed TSP, we introduce a Trusted Third Party (TTP). A TTP is an entity that facilitates interactions between two parties who both trust the TTP. If there exists a TTP, the searcher can send V' and the server can send y to the TTP. Then the TTP can find a local optimum by some conventional method. With a TTP, privacy preserving optimization of $(1,1)-TSP$ is ideally stated as follows:

Statement 1. Let there be a server and a searcher. The server holds a private cost vector y and the searcher holds a private city set $V' \subseteq V$. In the end of the protocol, the searcher learns a local optimal tour x^* but nothing else. The server learns nothing.

This idea works perfectly, however, building a TTP is often quite difficult mainly in terms of cost. Needless to say, a protocol that works only between a searcher and a server in the standard network environment (e.g. TCP/IP network) is preferred. Our target in this paper is to propose a protocol for the PPO that works without using any TTP.

2.3 Basic Idea

Many of GAs choose individuals to be survived by the rank of individual in fitness. If the objective function is represented as the scalar product, the secure rank computation is equivalent to a problem called *private Scalar Product Comparison* (SPC). Let x_1, x_2 be a solution and y be a cost vector. Although the cost vector is normally real-valued, it is can be treated as an integer vector without loss of generality. Here, let the domain of cost vector be $Z_m^d (= [0, ..., m]^d)$. Let inequations $x_2 \cdot y - x_1 \cdot y > 0$ be I_+ , $x_2 \cdot y - x_1 \cdot y = 0$ be I_0 and $x_2 \cdot y - x_1 \cdot y < 0$ be I_-. Then, the private SPC is formally stated as follows:

Statement 2. (private SPC) Let there be two parties, a searcher and a server. The searcher has two private vectors x_1, x_2 and the server has a private vector y. In the end of the protocol, Both share one of a correct inequation in $\{I_-, I_0, I_+\}$ and learn nothing else.

Assuming there exists a protocol to solve the private SPC, a LS and a GA that preserve searcher's and server's privacy are designed. Let a neighborhood of a tour x be $N(x)$. Then, the algorithm of LS is described as follows:

[Local search]

1. (generation) $x \in_r N(x_0), N(x_0) \leftarrow N(x_0) \setminus \{x\}$
2. (selection) If $x \cdot y < x_0 \cdot y$, then $x_0 \leftarrow x$
3. (termination) If $N(x_0) = \emptyset$ or satisfies some terminate conditions, $x^* \leftarrow x_0$ and output x^*. Else, go to step 1.

where $\in_r$ denotes an uniform random selection of an element from a set. Suppose that the searcher posesses private x, x_0 and the searcher posesses private y. Nevertheless, step 1 and 3 can be executed solely by the searcher for any V' and $N(\cdot)$. In other words, there is no privacy concern in these steps. Step 2 requires a coordination between the server and the searcher. Apparently, this step can be privately executed if there exists a protocol to solve the private SPC.

Next, we discuss the privacy preservation in GAs in the same distributed setting. GAs are conceptually described as follows:

[Genetic algorithm]

1. (selection for reproduction) Select a pair of tour P as parents from population X
2. (crossover) Generate a set of child C by crossover using P
3. (selection for survival) Update X using $P \cup C$
4. (termination) If some terminate conditions satisfied, output x^*. Else, go to step 1.

If selection for reproduction is random, the searcher can solely execute step 1 and 4 as well as LS. Step 2 is executed solely if the crossovers is designed independent on the cost function such as uniform and one-point crossover. Obviously, at step 3, the searcher needs coordination of the server. Well-known methods of selection for survival, such as Steady-State(replace the worst individual in the population for a child), CHC(replace all individual for the best N_{pop} individual from the union of the population and the children), Elitist Recombination(replace parents for the best and the second best individuals in the union of the parents and the children), CCM(replace a parent for the best individual in the union of the parents and the children) are designed based on the rank of individuals[1]. So, the step 3, selection for survival, is also privately executed if there exist a protocol for the private SPC.

Considering above, we propose a protocol for solving the private SPC to design a GA and a LS with privacy preservation in next section.

3. SCALAR PRODUCT COMPARISON

3.1 Homomorphic Cryptosystem

To solve private SPC, we utilize a public-key cryptosystem with homomorphic property. A public-key cryptosystem is a triple (Gen, Enc, Dec) of probabilistic polynomial-time algorithm for key-generation, encryption and decryption, respectively. The key generation algorithm generates a valid pair (s_k, p_k) of private and public keys. Please notice that private key and public key are used only for decryption and encryption. $\mathcal{M}$ denotes the plaintext space. The encryption of a plain text $t \in \mathcal{M}$ is denoted as $Enc_{p_k}(t; r)$, where r is a random integer. The decryption of a cipher text is denoted as $t = Dec_{s_k}(c)$. Given a valid key pair (p_k, s_k), $Dec_{s_k}(Enc_{p_k}(t; r)) = t$ for any t and r is required.

A public key cryptosystem with additive homomorphic property satisfies the following identities:

$$\begin{aligned} Enc(t_1; r_1) \cdot Enc(t_2; r_2) &= Enc(t_1 + t_2; r_1 + r_2), \\ Enc(t_1; r_1)^{t_2} &= Enc(t_1 t_2; r_1), \end{aligned}$$

where $t_1, t_2 \in \mathcal{M}$ are plain texts and r_1, r_2 are random numbers. These properties enable the addition of any two encrypted integers and the multiplication of a encrypted integer by a integer. A public-key cryptosystem is *semantically secure* when a probabilistic polynomial-time adversary cannot distinguish between random encryptions of two elements chosen by herself. Paillier cryptosystem is known as one of semantically secure cryptosystems with homomorphic property[7]. We use Paillier cryptosystem in experiments.

[1]See [12], [13] and [6] for the detail of these selection methods

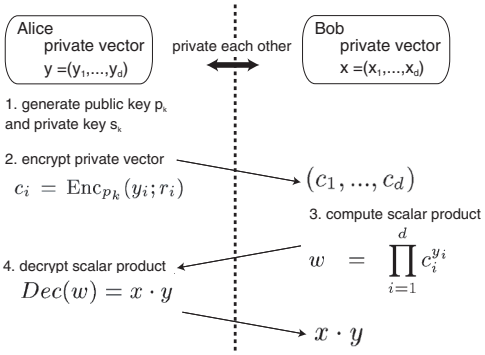

Figure 2: Private Scalar Product

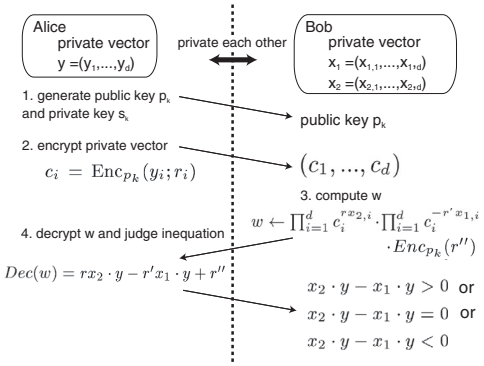

Figure 3: Private Scalar Product Comparison

3.2 Private Scalar Product

Based on this homomorphic public-key cryptosystem, Goethals et. al have proposed a protocol to compute scalar products of two distributed private vectors without revealing them each other[8]. This protocol is used as a building block of a protocol for the private SPC. For preserving the generality of the protocol, parties are described as Alice and Bob in this section. The private scalar product is stated as follows:

Statement 3. (Private scalar product) Let there be two parties, Alice and Bob. Alice has a private vector $x \in Z_\mu^d$ and Bob also has a private vector $y \in Z_\mu^d$. In the end of the protocol, Both Alice and Bob learn the scalar product $x \cdot y$ and nothing else.

Let Z_p be the message space for some large p. Set $\mu = \lfloor\sqrt{p/d}\rfloor$. Then, the protocol is described as follows:

[Private scalar product protocol]

- Private Input of Alice : $y \in Z_\mu^d$
- Private Input of Bob : $x \in Z_\mu^d$
- Output of Alice and Bob : $x \cdot y$

1. Alice: Generate a public and private key pair (p_k, s_k).
2. Alice For $i = 1, ..., d$, compute $c_i = \mathrm{Enc}_{p_k}(y_i)$ and send them to Bob.
3. Bob: Compute $w \leftarrow \prod_{i=1}^d c_i^{x_i}$ and send w to Alice.
4. Alice: Compute $Dec(w) = x \cdot y$ and send $x \cdot y$ to Bob.

At step 2, Alice sends the ciphertext of her private vector $(c_1, ..., c_d)$ to Bob. Because Bob does not possess the private key, he cannot learn Alice's vector from c. However, at step 3, he can compute the encrypted scalar product based on homomorphic property without knowing Alice's x as follows:

$$\begin{aligned} w &= \prod_{i=1}^d c_i^{y_i} = \prod_{i=1}^d Enc_{p_k}(x_i)^{y_i} \\ &= Enc_{p_k}(x_1 y_1) \cdots Enc_{p_k}(x_d y_d) \\ &= Enc_{p_k}(\sum_{i=1}^d x_i y_i) = Enc_{p_k}(x \cdot y). \end{aligned}$$

Then, at step 4, Alice correctly obtains $x \cdot y$ by decrypting w using her private key. Assuming Alice and Bob behave semi-honestly, it is proved that scalar product protocol is secure[8]. Semi-honest party is one who follows the protocol properly with the exception that it keeps a record of all its intermediate values. From accumulated records, semi-honest parties try to learn other party's privacy[11]. In following sections, we assume all party behave as semi-honestly.

3.3 Private Scalar Product Comparison

A protocol for the private SPC appears to be easily designed using private scalar product protocol. Instead of computing scalar product at step 3, the difference of two scalar products $x_2 \cdot y - x_1 \cdot y$ can be computed as follows:

$$\prod_{i=1}^d c_i^{x_{2,i}} \cdot \prod_{i=1}^d c_i^{-x_{1,i}} = Enc_{p_k}(x_2 \cdot y - x_1 \cdot y). \quad (1)$$

By sending this to Alice, Alice learns $x_1 \cdot y - x_2 \cdot y$. Although eq. 1 appears to successfully and privately compare two scalar products, it is not secure based on the statement 2 because not only the comparison result but also the value of $x_1 \cdot y - x_2 \cdot y$ is known to Alice. In case of the TSP, tour vectors are $x_1, x_2 \in \{0,1\}^d$. Therefore, so Bob's x_1 and x_2 are easily enumerated from the value of $x_1 \cdot y - x_2 \cdot y$ by Alice. In order to block Alice's enumeration, Bob can multiply some positive random value r_B to the difference of two scalar products,

$$\prod_{i=1}^d c_i^{r_B x_{2,i}} \cdot \prod_{i=1}^d c_i^{-r_B x_{1,i}} = Enc_{p_k}(r_B(x_2 \cdot y - x_1 \cdot y)).$$

Unfortunately, this is not also secure. Because r_B is one of a divider of $r_B(x_1 \cdot y - x_2 \cdot y)$, r_B is easily enumerated. Then, Alice can also enumerate the candidate of Bob's x_1 and x_2 for each r_B in polynomial time.

As shown, multiplying a random number does not contribute to hinder Alice's guess. Instead of the evaluation of $x_1 \cdot y - x_2 \cdot y$ or $r_B(x_1 \cdot y - x_2 \cdot y)$, we propose a protocol that evaluates $rx_2 \cdot y - r'x_1 \cdot y + r''$ with three random integers r, r', r'' as follows:

$$\prod_{i=1}^d c_i^{r x_{2,i}} \cdot \prod_{i=1}^d c_i^{-r' x_{1,i}} \cdot Enc_{p_k}(r'') = Enc_{p_k}(rx_2 \cdot y - r'x_1 \cdot y + r'').$$

The protocol is shown in figure 4. First, we explain why this protocol can correctly compare two scalar products. For simplicity, let $v_1 = x_1 \cdot y$, $v_2 = x_2 \cdot y$ and $M = dm^2$. Then

[Private scalar product comparison]

- Private Input of Alice : $y \in Z_m^d$
- Private Input of Bob : $x_1, x_2 \in Z_m^d$
- Output of Alice and Bob : An inequation $I \in \{I_-, I_0, I_+\}$

1. Alice: Generate a private and public key pair(p_k, s_k) and send p_k to Bob.
2. Alice: For $i = 1, ..., d$, Alice computes $c_i = \mathrm{Enc}_{p_k}(y_i)$. Send them to Bob.
3. Bob: Compute $w \leftarrow \prod_{i=1}^{d} c_i^{rx_{2,i}} \cdot \prod_{i=1}^{d} c_i^{-r'x_{1,i}} \cdot Enc_{p_k}(r'')$ and send w to Alice where r, r' and r'' are random integers satisfying
$$M^8 < r < \quad r' \quad < (1 + \frac{1}{M})r < \rho, 0 \leq r'' < M.$$
4. Alice: Compute $Dec_{s_k}(w) = rx_2 \cdot y - r'x_1 \cdot y + r''$. Then,
   - send $x_2 \cdot y - x_1 \cdot y > 0$ if $Dec_{s_k}(w) > M^7$
   - send $x_2 \cdot y - x_1 \cdot y = 0$ if $-M^8 < Dec_{s_k}(w) < M$
   - send $x_2 \cdot y - x_1 \cdot y < 0$ if $Dec_{s_k}(w) < -M^8$

Figure 4: Private scalar product comparison

[Privacy Preserving Local Search based on Neighborhood $N(x)$]

- Private Input of Server : cost vector $y \in Z_m^d$
- Private Input of Searcher : city subset $V' \subset V$
- Private Output of Searcher : local optimal tour x^*

1. Server: Generate a pair of a private and a public key (p_k, s_k) and send p_k to searcher.
2. Server: For $i = 1, ..., d$, compute $c_i = \mathrm{Enc}_{p_k}(y_i)$ and send them to searcher.
3. Searcher: Generate a random initial tour x_0 using V'
4. Searcher: $x \in_r N(x_0), N(x_0) \leftarrow N(x_0) \setminus \{x\}$
5. Searcher:
   (a) Compute $SPC(x_0, x, c)$ with probability 0.5. Otherwise, compute $SPC(x, x_0, c)$.
   (b) If the output of private SPC corresponds to $x \cdot y - x_0 \cdot y < 0$, then $x_0 \leftarrow x$
6. Searcher: If $N(x_0) = \emptyset$ or satisfies some terminate condition, $x^* \leftarrow x_0$ and output x^*. Else, go to step 4.

Figure 5: Privacy Preserving Local Search

$v_1, v_2 \in Z_M$ because $x_1, x_2, y \in Z_m^d$. Regarding $rv_2 - r'v_1 + r''$ learned by Alice at step 4, following properties are shown.

Lemma 1. Let v_1, v_2 be integers in Z_M. Let r, r', r'' be integers satisfying $LM < r, r < r' < (1 + \frac{1}{M})r, 0 \leq r'' < M$ and
$$S = v_2 r - v_1 r' + r''. \quad (2)$$
Assume that $L > M$. Then, for all $v_1, v_2 \in [0, ..., M]$,
$$v_2 - v_1 > 0 \iff S > L \quad (3)$$
$$v_2 - v_1 = 0 \iff -LM \leq S < M \quad (4)$$
$$v_2 - v_1 < 0 \iff S < -LM. \quad (5)$$

The Proof is shown in appendix. This lemma shows that the sign of $x_2 \cdot y - x_1 \cdot y$ is known from the value of $S = rv_2 - r'v_1 + r''$ when r, r', r'' satisfies conditions shown in this lemma. Intuitively, the sign of $v_2 - v_1$ is learned from $S = rv_2 - r'v_1 + r''$ because r and r' are very large, similar but slight different positive integers. Please notice that r' is bounded by $\rho = \lfloor p/2M \rfloor$ in the protocol such that S exists in $[\lceil -p/2 \rceil, ..., \lfloor p/2 \rfloor]$. This bound is required not to change the sign of S after the computation of modulo p used in the encryption/decryption.

From the value $S = v_2 r - v_1 r' + r''$ received from Bob, Alice may imply the value of two scalar products. Next lemma shows why Alice cannot imply anything.

Lemma 2. Assume that $L > M^7$. Given S, there exists some triple (r, r', r'') that satisfies $LM < r, r < r' < (1 + \frac{1}{M})r, 0 \leq r'' < M$ and $S = v_2 r - v_1 r' + r''$ for any v_1 and v_2.

The proof is also shown in appendix. According to this lemma, for any Alice's guess about v_1 and v_2, there exist some triple (r, r', r'') that satisfies $S = v_2 r - v_1 r' + r''$. Therefore, Alice cannot guess any information from S. Using these two lemmas, a theorem is shown for this protocol.

Theorem 1. Assuming Alice and Bob behave semi-honestly, private SPC protocol is secure in statement 2.

The sketch of proof is shown in appendix. Using this protocol for private SPC, we design a LS and a GA for solving TSP with preserving privacy in next section.

4. PRIVACY PRESERVING GENETIC ALGORITHM FOR TSP

4.1 Privacy Preserving Local Search

Using the private SPC protocol, Privacy Preserving Local Search (PPLS) is designed as shown in figure 5. Alice and Bob in the private SPC protocol correspond to the server and the searcher in PPLS, respectively.

Step 1 and 2 is the same with the private SPC protocol. At Step 3, an initial tour is generated randomly. At Step 4, a new tour is chosen as a candidate. Then at step 5, two cost values are compared by the protocol. $SPC(x_1, x_2, c)$ represents the execution of the private SPC protocol by taking y as server's input and x_1, x_2 as searcher's input. The reason why the order of the inputs of $SPC(\cdot, \cdot, c)$ is randomly changed is explained in next section in detail. Please notice that various neighborhood operators such as k-opt are available here because the generation of neighborhood can be executed independent to the server.

4.2 Privacy Preserving Genetic Algorithm

As discussed in section 2.3, a Privacy Preserving GA(PPGA) can be designed. [14] has reported that GAs using Edge Assembly Crossover (EAX) perform extremely well as against other solvers. [6] has reported that the combination of EAX and CCM improves the performance. Although various combination of crossovers and selection methods are available in PPGAs, we focus on a GA that adopts EAX[10] as a crossover and CCM as a selection method[6]. A GA using CCM is described as follows:

[CCM]

1. Generate N_{pop} solutions $X_t = \{x(1), ..., x(N_{pop})\}$ randomly. $t = 0$.
2. Shuffle the index of population X_t.
3. For each $i = 1, ..., N_{pop}$, do
   (a) Choose $x(i)$ and $x(i+1)$ as parents. Then, generate N_{child} children $X^c = \{x^c(1), ..., x^c(N_{child})\}$ by EAX

[**Privacy Preserving Genetic Algorithm (EAX/CCM)**]
(The inputs, outputs, step 1 and 2 are omitted because they are similar to PPLS)

3. Searcher: Generate N_{pop} initial tours $X = \{x(1), ..., x(N_{pop})\}$ using V' randomly. $t = 0$.
4. Searcher: Shuffle the index of population X_t
5. Searcher: For $i = 1, ..., N_{pop}$, do
   (a) Choose $x(i), x(i+1)$ as parents. Then, generate Offspring $X^c = \{x^c(1), ..., x^c(N_{child})\}$ by EAX where let $x(N_{pop})$ be $x(1)$.
      i. For $j = 1, ..., N_{child}$, do
         A. Compute $SPC(x(i), x(j)^c, c)$ with probability 0.5. Otherwise compute $SPC(x(j)^c, x(i), c)$.
         B. If output of private SPC corresponds to $x_j^c \cdot y - x_i \cdot y > 0$, $x_i \leftarrow x_j^c$

      where let $x_{N_{pop}+1}$ be x_0.
   (b) Replace $x(i)$ with the best tour in $X^c \cup \{x(i)\}$.
6. Searcher: If termination conditions are satisfied, output the best tour x^* in X_t. Else, $t \leftarrow t + 1$ and jump to step 4.

Figure 6: Privacy Preserving Genetic Algorithm.

4. If termination conditions are satisfied, output the best tour x^* in X_t. Else, $t \leftarrow t + 1$ and jump to step 2.

Because our focus of this paper is not on a GA for TSPs but on the privacy preservation technique for GAs, we do not mention the detail of EAX. See [10] for detail. Based on private SPC, a PPGA with EAX/CCM for TSP can be instantly designed as shown in figure 6.

Because inputs, outputs, step 1 and 2 are the same with PPLS, they are omitted here. At Step 3, the population is initialized. Step 4 corresponds to step 2 in CCM.

In EAX (step 5(a)), sub tours are merged using a 2-opt-like operator such that the cost of the generated individual is as small as possible. The private SPC protocol is used to perform this operation in privacy preserving manner in EAX (not shown in figure 6). Private SPC protocol is also iteratively utilized at step 5(a) i to find the best tour in $X_i^c \cup \{x_i\}$, which corresponds to step 3 in CCM.

4.3 Security of PPLS/GA

We verify what is leaked from the execution of the protocol. From the execution of the protocol, the searcher learns a sequence of comparison results such as $x_2 \cdot y > x_1 \cdot y$ and $x_2 \cdot y < x_1 \cdot y$. Although the value of cost vector y is not leaked from this, partial orders of each cost between two cities may be implied by the searcher. Therefore, the privacy of the server is not perfectly protected with this protocol and additional information is leaked.

Next, we explain what the server learns. Please recall that the server cannot guess anything from the value of w received as shown in lemma 2. Assume that outputs of private SPC protocol are only I_- or I_+. Then, the server learns only a random sequence of these two inequations because the order of inputs of private SPC is shuffled randomly at step 5(a)-i-A. If inequation I_0 is included in outputs of private SPC, the server learns it because I_0 is invariant to the order of the input. Therefore, when I_0 occurs, the server learns the frequency that the two tours with the same tour lengths are generated. This information is trivial and does not spoil the security of the protocol much.

As shown, the server and the searcher learns more from the execution of the protocol than described in statement 1. However, it is also shown that the server never learns searcher's private input V' and the searcher never learns server's private input y except for the order of each element.

4.4 Computational Analysis

Communication complexity: Communication between the server and the searcher occurs at step 1, 2 and 5(a)-i. In the distributed TSP, we can assume that the cost vector y is not changed. Therefore, the communication at step 1 and 2 occurs only once. The communication complexity of step 5(a)-i(private SPC) is $O(1)$. So, the communication complexity is not time consuming in this protocol very much.

Computational complexity: The bottleneck is in the exponentiation computed by the searcher (step 3 of private SPC) and this is repeated every time a new tour is generated. With naive implementation, step 3 of private SPC costs $O(d)(= O(n^2))$. To reduce this computation, we exploit the fact that the number of changed edges by EAX is much smaller than d. In most cases, EAX drops at most 50 edges and adds new 50 edges. Let x be a child generated by EAX and x_i and x_{i+1} be their parents. If $2k$ is the number of changed edges, in vector $rx - r'x_i$, k elements are changed from r to 0, the other k elements are changed from 0 to $-r'$ and the rest of elements are all 0. Because the number of changing edge are $2k$, the time complexity is saved as $O(k)$ by computing only in changing edges[2].

The computational complexity of the merge operation in EAX is also $O(n^2)$ with the naive implementation. This is saved as $O(n)$ if the searcher learns a list of k-nearest cities for each city. We can assume that these lists are given to the searcher because these lists are included in what the searcher learns as shown in the previous section.

5. EXPERIMENTS

5.1 Setting

The scalability of the PPLS/PPGA is studied by computational experiments. Five problem instances are chosen from TSPLIB where city size ranges from 195 to 1173. In the distributed TSP, the searcher can arbitrarily choose a subset of cities V' as the input. In our experiments, V' is set to V for all problems because the computation time becomes the largest when $V = V'$. PPLS is terminated when x_0 reaches to a local optimum. PPGA is terminated when the best cost in the population is not improved for twenty successive generations.

As a homomorphic cryptosystem, Paillier cryptosystem[7] with 512-bit and 1024-bit key is used . The server and the searcher program are implemented by J2SE ver. 1.4.1. Both programs are separately ran on Xeon2.8GHz(CPU), 1GB(RAM) Windows PCs on 100Mbps Ethernet.

In experiments, PPLS using 2-opt neighborhood and PPGA using EAX/CCM are compared. PPLS is repeated for 50, 100, 300 times with changing initial tours (depicted as 50-itr, 100-itr and 300-itr). In PPGA, the population size is set

[2]In 2-opt neighborhood, k is always two and the time complexity is constant.

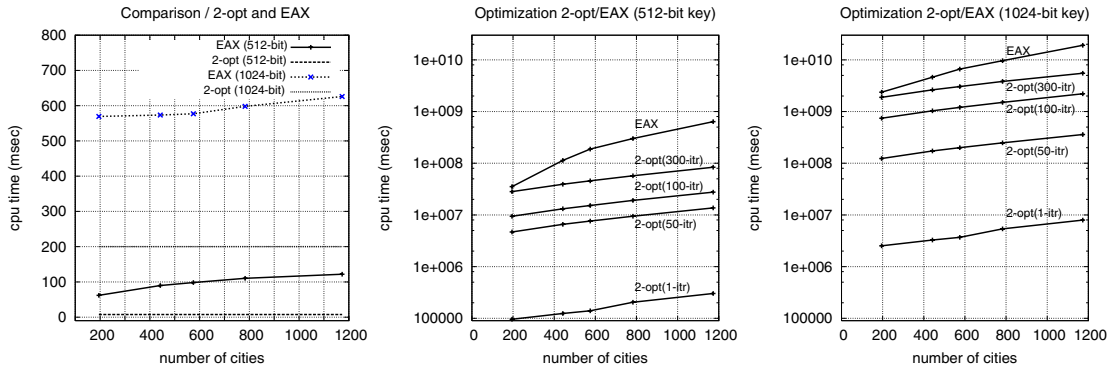

Figure 7: Computation time of PPLS/PPGA, left : Time required for generation and comparison (512-bit and 1024-bit key), center : Estimated time required for the completion of optimization (512-bit key), right : Estimated time required for the completion of optimization (1024-bit key)

to $N_{pop} = 100$ (each tour is optimized by PPLS using 2-opt) and number of child is set to $N_{child} = 30$.

Because both step 1 and 2 of PPLS/GA can be executed preliminary before choosing city subset V', we measured the execution time form step 3 to the termination of the protocol in both PPLS and PPGA. In addition, to verify the computation time per iteration, we measured searcher's computation time spent for the comparison of the cost of two different tours (corresponds to step 5(a)-i in PPLS/GS, step 3 and 4 in private SPC).

5.2 Results

Figure 7 (left) shows the computation time required for comparison per one iteration. As shown, the computation time of 2-opt is kept constant because the number of changed edge is always four. In EAX, the time increases because the number of changed edges slightly increases with respect to the number of cities. Figure 7 (center and right) shows the estimated time required for the optimization. With 512-bit key, PPLS(1-itr) spent 95 (sec) and 139 (sec) to search for local optimum of rat195 and rat575. EAX spent 9.8 (hour) and 52 (hour) for rat195 and rat575. With 1024-bit key, PPLS (1-itr) spent 42 (min) and 60 (min) and EAX spent 656 (hour) and 1837 (hour) for same problems.

Table 1 shows the error (=100× obtained best tour length / known best tour length) of PPLS/PPGA (average of 20 trials). Please notice that privacy preservation does not affect the quality of the obtained solution at all because the protocol does not change the behavior of LS/GA if the same random seed is used. Apparently, the quality of obtained solutions by PPGA is much better than those of PPLS.

If 10 % error is satisfactory, PPLS with 2-opt (1-itr) is reasonable because the computation ends within a few minutes for 512-bit key and a few hour for 1024-bit key. PPGA finds extremely good tours, however, the computation time for convergence is more than a day or a month, that is, parallel computation is essential in this setting. Please recall that we set $V' = V$ for all problems. Even when the number of city $|V|$ is very large, the computation time is kept small if the number of chosen city $|V'|$ is small because the computation time is dependent on the number of chosen cities.

Although the computation time is not yet sufficiently small in large-scale problem, it is confirmed that the protocol completes in practical time in privacy preserving setting when number of cities are not very large. Nevertheless, the PPGA protocol should be improved to be much more efficient.

	PPLS(2-opt)				PPGA(EAX)
	1-itr.	50-itr.	100-itr.	300-itr.	
rat195	13.3	9.16	8.73	8.65	0.0710
pcb442	16.4	8.88	8.88	8.88	0.0127
rat575	11.6	9.96	9.96	9.90	0.0391
rat783	12.2	10.8	10.8	10.3	0.0340
pcb1173	15.4	12.9	12.9	12.3	0.0280

Table 1: The error of tours obtained by PPLS (2-opt) and PPGA (EAX/CCM).

6. CONCLUSION

We propose a protocol for the privacy preserving distributed combinatorial optimization using a LS and a GA where the objective function is represented as the scalar product. As an example of distributed combinatorial optimization problems, we focus on the distributed TSP and design a privacy preserving LS that adopts 2-opt as neighborhood and a privacy preserving GA that adopts EAX as crossover and CCM as selection method based on a protocol that solves private scalar product comparison. The future work is to apply the PPGA to the distributed combinatorial optimization other than distributed TSP, such as the distributed QAP, VRP and Knapsack problem.

7. REFERENCES

[1] Philip W. Blasmeier and Wendell J. Voisin, Supply Chain Management : A Time-Based Strategy, Industrial Management, September-October 1996, pp. 24-27, (1996).

[2] Robert B. Handfield and Ernest L. Nichols, Introduction to Supply Chain Management, Prentice Hall, 1st edition (1998).

[3] Andrew C.-C. Yao, Protocols for secure computation, Proc. of IEEE Symposium on Foundations of Computer Science (FOCS), pp. 160-164 (1982).

[4] M.-C. Silaghi and D. Mitra, Distributed constraint satisfaction and optimization with privacy enforcement, Intelligent Agent Technology, pp, 531-535, (2004).

[5] Koutarou Suzuki and Makoto Yokoo, Secure Generalized Vickrey Auction using Homomorphic Encryption, Seventh International Financial Cryptography Conference (FC-03) (2003).

[6] Kokoro Ikeda, Shigenobu Kobayashi, Deterministic Multi-step Crossover Fusion: A Handy Crossover for GAs, PPSN 7, pp. 162-171, (2002).
[7] Pascal Paillier, Public-Key Cryptosystems Based on Composite Degree Residuosity Classes, EUROCRYPT 1999, pp. 223-238, (1999).
[8] Bart Goethals, Sven Laur, Helger Lipmaa and Taneli Mielika"inen, On Private Scalar Product Computation for Privacy-Preserving Data Mining, 7th Int'l Conf. in Information Security and Cryptology (ICISC), vol. 3506 of LNCS, pp. 104-120, (2004).
[9] S. Lin and B. Kernighan, Effective heuristic algorithm for the traveling salesman problem, Oper. Res., vol. 21, pp. 498-516, (1973).
[10] Yuichi Nagata and Shigenobu Kobaayshi, Edge Assembly Crossover : A High-power Genetic Algorithm for the Traveling Salesman Problem, 7th Int'l. Conf. on Genetic Algorithms, pp. 450-457, (1997).
[11] Goldreich, O., Foundations of Cryptography II: Basic Applications , Cambridge Univ Pr. (2004).
[12] D. E. Goldberg, Genetic Algorithm in Search, Optimization and Machine Learning, Addison Wesley, (1989).
[13] D. Thierens and D. E. Goldberg, Elitist Recombination : an integrated selection recombination GA, First IEEE Congress on Evolutionary Computation, pp. 508-512 (1994).
[14] H. Shimodaira, A Diversity Contral Oriented Genetic Algorithm (DCGA) : Development and Experimental Results, Proc. of GECCO 1999, pp. 603-611 (1999).

APPENDIX

A. PROOF OF LEMMA 1

Proof 1. We start from the sufficient condition of eq. 3.

$$\begin{aligned} S - r'' &= v_2 r - v_1 r' > v_2 r - v_1(1+\frac{1}{M})r \\ &= (v_2 - v_1)r - \frac{r}{M}v_1. \end{aligned}$$

From $v_2 - v_1 > 0$, $S - r''$ is minimum for any r when $v_2 = M, v_1 = M-1$, hence $S - r'' > \frac{r}{M} > L$. It follows that $v_2 - v_1 > 0 \Longrightarrow S > L$. The necessary condition is obvious because $L > M$.

Next, we prove eq. 4. Because $\frac{r'-r}{r} < \frac{1}{M}$, we obtain $0 < r' - r < \frac{r}{M} < L$. Then, from $v_1 = v_2, v_1 \neq 0$,

$$\begin{aligned} S - r'' = v_1(r - r') &\iff v_1(r-r') \leq S < v_1(r-r') + M \\ &\iff -LM \leq S < M. \end{aligned}$$

At last, we prove eq.5. For any v_1 and v_2, S is maximum when $r' = r + 1$. So from $v_2 - v_1 < 0$,

$$\begin{aligned} S - r'' &= v_2 r - v_1 r' < v_2 r - v_1(r+1) \\ &= -(v_1 - v_2)r - v_1 \leq -r - 1 < -LM. \end{aligned}$$

Next, we prove necessary condition. From $S < -LM$ and $r'' > 0$,

$$\begin{aligned} & S - r'' = v_2 r - v_1 r' < -LM \\ \iff & r(v_2 - v_1) < -LM + v_1 < 0 \iff v_2 - v_1 < 0 \end{aligned}$$

Then eq. 5 is proved.

B. PROOF OF LEMMA 2

Proof 2. First, we prove the case of $S > L$. From lemma 1, we obtain $1 \leq v_1 < v_2$. For all v_1, v_2, there exists some triple $(r, r' r'')$ satisfying the linear Diophantine equation $S = v_2 r - v_1 r' + r''$ if and only if $S - r''$ is divisible by $gcd(v_1, v_2)$. Because $1 \leq v_1 < v_2 \leq M$ and $0 \leq r'' < M$, there exists some positive integer β such that $S - r'' = \beta S'$ by choosing an appropriate r''. Because v_1 and v_2 is written as $v_2 = \beta X_2, v_1 = \beta X_1$, the Diophantine equation is equivalent to $S' = X_2 r - X_1 r'$, where $1 \leq X_1 < X_2 < M$, $S' > L/M$.

Let one solution of Diophantine equation $1 = X_2 r - X_1 r'$ be $r = r_0, r' = r'_0$. From them, solutions of $S' = X_2 r - X_1 r'$ are

$$r = S' r_0 + X_1 t, r' = S' r'_0 + X_2 t, (t = 0, \pm 1, \pm 2 ...)$$

From $r' < r < (1 + \frac{1}{M})r$, we obtain $\frac{r'-r}{r} < \frac{1}{M}$. We determine the condition of L that satisfies $\frac{r'-r}{r} < \frac{1}{M}$. $f(t) = \frac{r'-r}{r}$ and $f'(t) = \frac{df(t)}{dt}$ is represented as a function of t as follows.

$$f(t) = \frac{S'(r'_0 - r_0) + (X_2 - X_1)t}{S' r_0 + X_1 t} \quad (6)$$

$$f'(t) = \frac{S'\{(X_2 - X_1)r_0 - (r'_0 - r_0)X_1\}}{(S' r_0 + X_1 t)^2} \quad (7)$$

When $t_0 = \frac{-(X_2 - X_1)}{S'(r'_0 - r_0)}$, then $f(t_0) = 0$. With this t_0, we determine the condition that a positive integer t exists that satisfies $\frac{r'-r}{r} < \frac{1}{M}$.

By the nature of the linear fractional function, the form of function $f(t)$ is classifiable into two cases: $(X_2 - X_1)r_0 - (r'_0 - r_0)X_1 > 0$ or $(X_2 - X_1)r_0 - (r'_0 - r_0)X_1 < 0$.

We start from the case of $(X_2 - X_1)r_0 - (r'_0 - r_0)X_1 > 0$. In this case, $f(t)$ is monotonically increasing and convex upward around $t = t_0$. Consequently, if $f'(t_0) < \frac{1}{M}$, there exists at least one integer t that satisfies $f(t) < \frac{1}{M}$ within the range $[t_0, t_0 + 1)$. By substituting t_0 into eq. 7 and rearranging it, we obtain

$$f'(t_0) = \frac{(X_2 - X_1)^2}{S'\{(X_2 - X_1)r_0 - (r'_0 - r_0)X_1\}} < \frac{M^2}{L/M} = \frac{M^3}{L}.$$

Then, it is proved that there exists an integer t satisfying $\frac{r'-r}{r} < \frac{1}{M}$ if $L > M^4$.

Next we prove the second case, $(X_2 - X_1)r_0 - (r'_0 - r_0)X_1 < 0$. This function is monotonically decreasing and convex upward around $t = t_0$. Consequently, there exists at least one integer t within the range of $(t_0 - 1, t_0]$ if $f'(t_0) > -\frac{1}{M}$.

By substituting $t_0 = \frac{-(X_2 - X_1)}{S'(r'_0 - r_0)}$ into eq. 7 and rearranging it similarly, we obtain

$$f'(t_0) = \frac{(X_2 - X_1)^2}{S'\{(X_2 - X_1)r_0 - (r'_0 - r_0)X_1\}} > -\frac{M^2}{L/M} = -\frac{M}{L}$$

Therefore, it is also proved that there exists at least one integer t satisfying $\frac{r'-r}{r} < \frac{1}{M}$ if $L > M^4$.

The explanation above shows that this lemma is proved when $S > L$. Proofs with the case $S < -LM$ are shown similarly; the proof shows that there exists at least one integer t satisfying $\frac{r'-r}{r} < \frac{1}{M}$ if $L > M^7$. When $-LM \leq S < M$, the proof is readily apparent.

C. PROOF OF THEOREM 1

The proof is based on a formal definition on secure multiparty computation[11]. Here, we show a brief sketch of the proof of Theorem 1.

A computation is secure if the view (all received information) of each party during the execution of the protocol can be simulated by some polynomial machine only from the input and the output of the party.

In private SPC protocol, server's private input is the the secret key s_k and the output is the sign of $v_2 - v_1$. Server's view is w received at step 3 and $dec_{s_k}(w) = rv_2 - r'v_1 + r''$. The simulator for server generates some $\tilde{v}_1, \tilde{v}_2$ with satisfying the sign of $v_2 - v_1$ and generates $\tilde{r}, \tilde{r}', \tilde{r}''$ with satisfying conditions shown in lemma 2 randomly. Then the server computes the simulated view $dec_{s_k}(\tilde{w}) = \tilde{r}\tilde{v}_2 - \tilde{r}'\tilde{v}_1 + \tilde{r}''$.

The distinguisher checks whether there exists some (r, r', r'') that satisfies $dec_{s_k}(\tilde{w}) = rv_2 - r'v_1 + r''$. If it does not exist, the distinguisher knows the difference between $dec_{s_k}(w)$ and $dec_{s_k}(\tilde{w})$. However, lemma 2 shows that there exists some triple (r, r', r'') that satisfies eq. 2 for any v_1, v_2. Therefore, the distinguisher cannot distinguish the difference between server's view from protocol execution and the simulation view.

Simulation for a node is omitted here because server's privacy is apparently preserved because of the cryptosystem is semantically secure. Thereby, the theorem is proved.