How to Sign Paper Contracts? Conjectures & Evidence Related to Equitable & Efficient Collaborative Task Scheduling

Eric Brier[1], David Naccache[2], Li-yao Xia[2]

[1] Ingenico
1, rue Claude Chappe, BP 346, F-07503 Guilherand-Granges, France
eric.brier@ingenico.com
[2] École normale supérieure, Département d'informatique
45, rue d'Ulm, F-75230, Paris Cedex 05, France.
{david.naccache,li-yao.xia}@ens.fr

Abstract. This paper explores ways of performing commutative tasks by N parties. Tasks are defined as *commutative* if the order at which parties perform tasks can be freely changed without affecting the final result. It is easy to see that arbitrary N-party commutative tasks cannot be completed in less than $N-1$ basic time units.

We conjecture that arbitrary N-party commutative tasks cannot be performed in $N-1$ time units by exchanging less than $4N-6$ messages and provide computational evidence in favor this conjecture. We also explore the most equitable commutative task protocols.

1 Introduction

This paper explores ways of performing commutative tasks by N parties denoted $\mathcal{A}_0, \ldots, \mathcal{A}_{N-1}$. Tasks are defined as *commutative* if the order at which parties perform them can be freely changed without affecting the final result.

A typical example, used throughout this work, is the material signature of a contract by N parties. As the contract signing protocol $\mathfrak{P}$ ends each party obtains a printed contract bearing the N signatures of all other parties. Empty contracts can be printed by all parties. Each contract must transit through all parties to eventually bear all the required signatures.

This problem is not only of theoretical interest. Cryptography conceals the meaning of information but not its existence. In many cases network monitoring allows to infer useful information from the message flow. This attack is called *traffic analysis*. A well-known way to defeat traffic analysis consists in continuously padding the communication channel with dummy packets to simulate constant bandwidth occupation.

[1] states that *"...it is very hard to hide information about the size or timing of messages. The known solutions require Alice to send a continuous stream of messages at the maximum bandwidth she will ever use...This might be acceptable for military applications, but it is not for most civilian applications..."*

And [2] mentions that: *"... In practice this problem has been known for a very long time, and countermeasures are routinely used in modern link encryptors, by making sure that they always send information between sender and receiver, inserting dummy information if necessary [3]. By doing so, they seek to obscure the difference between actual communication and non-communication. Unfortunately, the approach taken by link encryptors to "keep the channel full" is infeasible on the Internet, due to the requirement that the communication infrastructure serves the needs of multiple parties..."*

It is hence useful to look for *economical* ways in which parties can exchange information without revealing their activity. Here envelopes represent constant-size encrypted data containers[3]. We show how to exchange $4N-6$ containers between N parties in a way that ascertains that $\forall i \neq j$, party $\mathcal{A}_i$ can send a message to $\mathcal{A}_j$ in $N-1$ elementary time units, provided that the container's capacity has not been exceeded.

We study protocols according the following three natural criteria:

Communication. Envelopes containing partially signed contracts must circulate between all the parties. We assume that if party $\mathcal{A}$ wants to send several contracts to party $\mathcal{B}$, $\mathcal{A}$ can put all those contracts in *one* envelope and pay a fixed postage fee for the envelope (hereafter \$1), *i.e.*, the cost of sending several contracts in an envelope is independent of the number of contracts sent. The following day, $\mathcal{B}$ will receive the contracts and sign them[4]. Therefore, the total postage fees paid by $\mathcal{A}$ at a given day is proportional to the number of envelopes sent by $\mathcal{A}$, which is also the number of parties to which $\mathcal{A}$ sent at least one contract. A first natural goal consists in *minimizing the postage fees* $\texttt{Cost}(\mathfrak{P}, N)$. We prove that $\min_{\mathfrak{P}} \texttt{Cost}(\mathfrak{P}, N) = 2N-2$.

[3] $\mathcal{A}_i$ gets a container, decrypts it and examines its contents: $\mathcal{A}_i$ extracts any messages sent to him and erases these messages from the container. $\mathcal{A}_i$ potentially inserts into the container new messages for other parties and re-encrypts the container for the next receiving party *without changing the container's size*.

[4] Assuming that the received contracts do not already bear $\mathcal{B}$'s signature.

Latency. It is easy to see that the contract signing task cannot be completed in less than $N-1$ days. We call protocols that run in $N-1$ days *fast protocols*. If N days are allowed, reaching the $\$(2N-2)$ cost's lower bound is simple (*e.g.* protocol $\mathfrak{P}_{\text{seq}}$ in section 2). Hence, we will focus our attention on the costs of fast protocols. We show how to construct fast protocols that cost $\$(4N-6)$ and conjecture that this cost is optimal:

Conjecture 1. For all N the cheapest fast protocol costs $\$(4N-6)$.

We checked this conjecture by exploiting problem symmetries and by using backtracking for $N \leq 8$.

Equitableness. It is interesting to find protocols in which postage costs are distributed between parties *as evenly as possible*.

We observed that for $6 \leq N \leq 8$ there exist fast protocols in which $N-6$ parties pay \$4 and 6 parties pay \$3.

We do not know how to construct such optimally equitable protocols otherwise than by computerized search. We call such protocols *equitable*.

Although current evidence that equitable protocols exist for all N is very limited, heuristics (*cf.* to section 10) suggest that all fast protocols are inherently inequitable in the following sense:

Conjecture 2. In every fast N-party protocol, the most burdened party must pay $\$\Omega(N)$.

Convention: In *"xxxx-protocol"* the xxxx will stand for any combination of the letters F,C,E,M meaning: fast, cheap, equitable and minimal.

Convention: The notation $\mathcal{A}_i \overset{k}{\rightsquigarrow} \mathcal{A}_j$ will mean *"$\mathcal{A}_i$ signs k contracts and sends them to $\mathcal{A}_j$"*.

2 Straightforward Non-Fast Protocols

A trivial sequential protocol is the following:

The Sequential Protocol $\mathfrak{P}_{\text{seq}}$	
Day	**Event**
0	$\mathcal{A}_0$ prints N empty contracts.
$i = 0, \ldots, N-2$	$\mathcal{A}_i \overset{N}{\rightsquigarrow} \mathcal{A}_{i+1}$
$N-1$	For $j = 0, \ldots, N-2$: $\mathcal{A}_N - 1 \overset{1}{\rightsquigarrow} \mathcal{A}_j$.

Note that:

- $\mathfrak{P}_{\text{seq}}$ is not fast because $\mathfrak{P}_{\text{seq}}$ validates the contracts on day N, assuming that indexing days starts from 0.
- $\mathfrak{P}_{\text{seq}}$ is cost-optimal, *i.e.* $\texttt{Cost}(\mathfrak{P}_{\text{seq}}, N) = 2N - 2$.
- $\mathfrak{P}_{\text{seq}}$ is inequitable because $\mathcal{A}_{N-1}$ pays $\$(N-1)$ while all other parties pay \$1.

3 Graphic Representation

A protocol is entirely defined by the path followed by each contract, *i.e.* the sequence of $\mathcal{A}_i$s that the contracts transit through each day (one line in Fig. 1).

For such a matrix to reflect a valid protocol, each $\mathcal{A}_i$ must appear at least once in each line and once in the last column.

We will use a very convenient graphic representation to illustrate protocols (*e.g.* Fig. 1). The graph of a protocol for N parties and D days is a bi-dimensional graph with $N \times (D+1)$ vertices.

Vertex (d, i) represents $\mathcal{A}_i$ on day d.

An edge is drawn between (d, i) and $(d+1, j)$ if $\mathcal{A}_i$ sends an envelope to $\mathcal{A}_j$ on day d. Edges may be labeled with the number of contracts in the corresponding envelope.

Note that such graphs may not uniquely characterize a protocol (see Fig. 2).

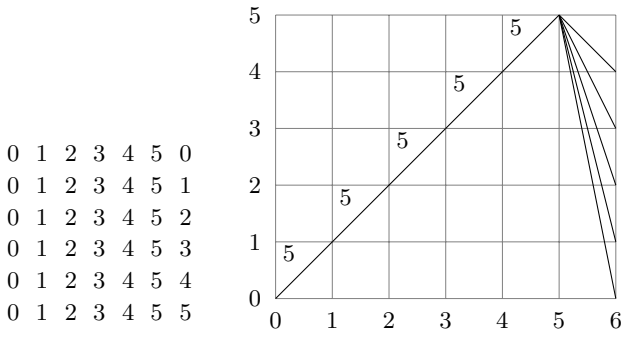


Fig. 1. The matrix and the graph of $\mathfrak{P}_{\text{seq}}$

0	1	2	0		0	1	2	0		1	0	2	0		
1	0	2	1		0	1	2	1		0	1	2	1		
0	1	2	2		1	0	2	2		0	1	2	2		

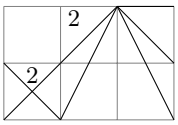


Fig. 2. A graph may correspond to several different protocols

4 Fast Protocols

It is easy to see that it takes at least $N-1$ days to complete the contract signing process and that there is a very simple solution for doing so:

The Circular Protocol $\mathfrak{P}_{\text{cir}}$	
Day	**Event**
0	Each party prints one empty contract.
$i = 0, \ldots, (N-1)$	For $j = 0, \ldots, N-1$: $\mathcal{A}_j \overset{1}{\rightsquigarrow} \mathcal{A}_{j+1 \bmod N}$.

- $\mathfrak{P}_{\text{cir}}$ is fast because $\mathfrak{P}_{\text{cir}}$ validates the contracts on day $N-1$, assuming that indexing days starts from 0.
- $\mathfrak{P}_{\text{cir}}$ is far from being cost-optimal, *i.e.* $\texttt{Cost}(\mathfrak{P}_{\text{cir}}, N) = N(N-1)$.
- $\mathfrak{P}_{\text{cir}}$ is equitable because each party pays $\$(N-1)$.

$\mathfrak{P}_{\text{cir}}$ outperforms $\mathfrak{P}_{\text{seq}}$ by one day but this (small) improvement comes at the rather high price of a quadratic increase in postage costs.

1	2	3	4	0
2	3	4	0	1
3	4	0	1	2
4	0	1	2	3
0	1	2	3	4

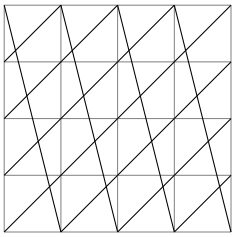


Fig. 3. The matrix and the graph of $\mathfrak{P}_{\text{cir}}$

It is hence natural to ask if linear-cost fast protocols exist and, more generally, find out what the cost $\texttt{CFP}(N)$ of the cheapest fast protocol is.

5 A Linear Protocol

The following protocol was designed following the *intuition* that to reduce costs, contracts must follow very similar routes. The obstruction to this is that each contract must also carefully avoid one participant, namely the party at which this contract's route will end. We hence design two parallel routes with one contract jumping from one route to the other, at each step.

The Linear Protocol $\mathfrak{P}_{\text{lin}}$	
Day	**Event**
0	$\mathcal{A}_0$ prints $N-1$ empty contracts. $\mathcal{A}_1$ prints one empty contract.
$i = 0, \ldots, N-3$	$\triangleright$ $\mathcal{A}_i$ has $N-i-1$ contracts; $\mathcal{A}_i \overset{1}{\rightsquigarrow} \mathcal{A}_{i+2}$; $\mathcal{A}_i \overset{N-i-2}{\rightsquigarrow} \mathcal{A}_{i+1}$; $\triangleright$ $\mathcal{A}_{i+1}$ has $i+1$ contracts; $\mathcal{A}_{i+1} \overset{i+1}{\rightsquigarrow} \mathcal{A}_{i+2}$.
$N-2$	$\mathcal{A}_{N-2} \overset{1}{\rightsquigarrow} \mathcal{A}_{N-1}$ For $j = 0, \ldots, N-2$: $\mathcal{A}_{N-1} \overset{1}{\rightsquigarrow} \mathcal{A}_j$[a]

[a] Each $\mathcal{A}_j$ gets from $\mathcal{A}_{N-1}$ the contract unsigned by $\mathcal{A}_j$

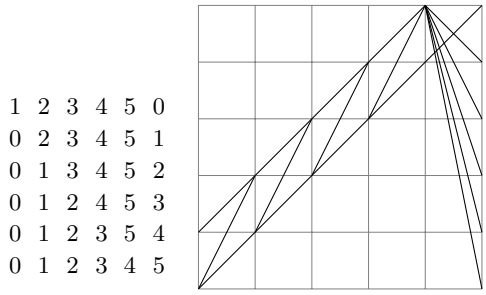


Fig. 4. The matrix and the graph of $\mathfrak{P}_{\text{lin}}$

$\texttt{Cost}(\mathfrak{P}_{\text{lin}}, N) = 4N-6$. The cost vector of $\mathfrak{P}_{\text{lin}}$ (fees paid by $\{\mathcal{A}_0, \ldots, \mathcal{A}_{N-1}\}$) is:

$$(2, \underbrace{3, 3, \ldots, 3, 3}_{N-3 \text{ times}}, 2, N-1)$$

As mentioned previously, we conjecture $\$(4N-6)$ to be optimal *i.e.* $\texttt{CFP}(N) = 4N-6$. We thus call $\$(4N-6)$ protocols *"cheap protocols"*.

6 Counting Protocols

We denote by

$\mathbb{S}_N$: the set $\{0, \ldots, N-1\}$

$\mathfrak{S}_N$: the set of $N!$ permutations of $\mathbb{S}_N$

6.1 Observations

Label each contract by the index of the party that will eventually own this contract; the sequence of parties that each contract n goes through in $N-1$ days must be a permutation of the set $\{\mathcal{A}_0, \ldots, \mathcal{A}_{N-1}\}$. As such we can identify a fast protocol with an ordered set of N permutations[5], in which the n-th permutation ends with n.

No matter what the fast protocol is, on day $N-1$ there will always be N envelopes sent, one to each party.

6.2 Number of Protocols

We have tried to enumerate fast protocols and look for some pattern in their structure.

As pointed-out *supra*, a fast protocol can be bijectively mapped to an ordered set of N permutations of $\mathbb{S}_N$ denoted $\mathfrak{P} = (\mathfrak{P}_0, \ldots, \mathfrak{P}_{N-1})$ where $\mathfrak{P}_n(N-1) = n$.

Using $\mathfrak{P} = \mathfrak{P}_{\text{lin}}$ in Fig. 4 as an example, the n-th line $\mathfrak{P}_n$ is the cycle $\gamma(n, \ldots, N) = (0, \ldots, n-2, n-1, n+1, n+2, \ldots, N, n)$.

For $n = 0, \ldots, N-1$, consider the n-th line without its last coordinate : $(\mathfrak{P}_n(0), \ldots, \mathfrak{P}_n(N-2))$ is a permutation of $\mathbb{S}_N \setminus \{n\} \simeq \mathbb{S}_{N-1}$.

The last coordinate that was removed must be equal to the line index. Consequently, fast protocols can be *bijectively* mapped onto sets of N permutations of $\mathbb{S}_{N-1}$. There are therefore $((N-1)!)^N$ fast protocols. Using that identification, we denote the set of fast protocols by $\mathfrak{S}_{N-1}^N$.

Using symmetry. There is a lot of symmetry in this problem, that we exploited to examine a (somewhat) lesser number of protocols.

The *relabeling* of $\mathfrak{P}$ by a permutation $\sigma \in \mathfrak{S}_N$ is the protocol obtained by renaming each party $\mathcal{A}_n$ as $\mathcal{A}_{\sigma(n)}$:

[5] of $\mathbb{S}_N$.

$$\sigma(\mathfrak{P}) = (\sigma \circ \mathfrak{P}_{\sigma^{-1}(0)}, \sigma \circ \mathfrak{P}_{\sigma^{-1}(1)}, \ldots, \sigma \circ \mathfrak{P}_{\sigma^{-1}(N-1)})$$

Notice that the change of index is such that $(\sigma(\mathfrak{P}))_n(n) = n$.

Protocol isomorphism. Two protocols $\mathfrak{P}, \mathfrak{P}'$ are *isomorphic* if $\mathfrak{P}$ can be transformed into $\mathfrak{P}'$ by relabeling. We denote this relation by $\mathfrak{P} \equiv \mathfrak{P}'$.

$$\mathfrak{P} \equiv \mathfrak{P}' \overset{\text{def}}{\iff} \exists \sigma\, \mathfrak{P}' = \sigma(\mathfrak{P})$$

The number of isomorphism classes $\texttt{NFMP}(N) = \left|\mathfrak{S}_{N-1}^N / \mathfrak{S}_N\right|$ (number of fast protocols) as a function of N is currently unknown for $N > 6$.

A naïve algorithm for deciding if $\mathfrak{P} \equiv \mathfrak{P}'$ requires $O(N^2 \cdot N!)$ time. We will now show that the protocol isomorphism decisional problem[6] can be solved in $O(N^3)$ time.

An interesting relabeling is $\sigma_{\mathrm{Id}} = \mathfrak{P}_n^{-1}$ for some $n \in \mathbb{S}_N$. σ_{Id} satisfies:

$$(\sigma_{\mathrm{Id}}(\mathfrak{P}))_{N-1} = \mathrm{Id}$$

And this equality holds if and only if $\sigma_{\mathrm{Id}} = \mathfrak{P}_n^{-1}$ for some n.

In the lexicographical order on permutations $\pi = (\pi(0), \ldots, \pi(N-1))$ seen as words of length N, Id is the smallest of all permutations.

Hence, when looking at protocols, which are ordered sets of N permutations $(\mathfrak{Q}_n)_{n=0,\ldots,N-1}$ as the concatenation $(\mathfrak{Q}_{N-1}, \ldots, \mathfrak{Q}_0)$ (this is a relation on $N \times N$ matrices used as words of length N^2), we notice that that the set $I_{\mathfrak{P}} = \{\mathfrak{P}_n^{-1}(\mathfrak{P}) \,|\, n = 0, \ldots, N-1\}$ contains the lexicographically smallest protocols which are isomorphic to $\mathfrak{P}$: it is exactly the set of protocols isomorphic to $\mathfrak{P}$ such that the last line of their matrix is Id.

Note that $I_{\mathfrak{P}}$ does not always have cardinality N, *e.g.* in $\mathfrak{P}_{\mathrm{cir}}$ illustrated in Fig. 3, $I_{\mathfrak{P}_{\mathrm{cir}}} = \{\mathfrak{P}_{\mathrm{cir}}\}$ is a singleton.

By examining only the N permutations that constitute $\mathfrak{P}$, it is possible to determine in $O(N^3)$ time the smallest protocol in the isomorphism class of $\mathfrak{P}$ (*e.g.* Fig. 5)

It is then a matter of checking equality between those single minimal representatives to decide if two protocols are isomorphic.

All in all, this process claims $O(N^3)$ time.

[6] *i.e.* Given $\mathfrak{P}, \mathfrak{P}' \in \mathfrak{S}_{N-1}^N$, decide if $\mathfrak{P} \equiv \mathfrak{P}'$.

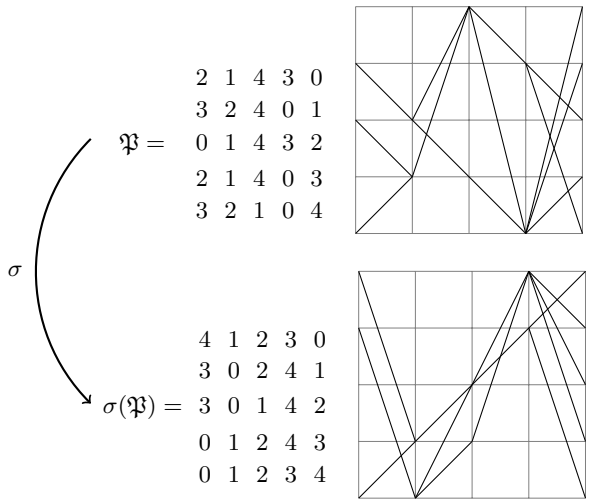


Fig. 5. Two isomorphic protocols $\mathfrak{P}$ (above) and $\mathfrak{P}' = \sigma(\mathfrak{P})$ (below), where $\sigma = \mathfrak{P}_0^{-1} = (4\,1\,0\,3\,2)$. $\mathfrak{P}'$ is the lexicographically smallest protocol isomorphic to $\mathfrak{P}$.

Backtracking. We have designed a backtracking algorithm to enumerate all mutually non-isomorphic fast protocols.

Only one representative of each isomorphism class is visited. To suit the above comparison algorithm, we chose to visit the minimal representatives. One property of such lexicographically smallest representatives is that the last line of their corresponding matrix is Id.

The number of protocols with this latter property is $((N-1)!)^{N-1}$; compared with the original $((N-1)!)^N$, this saves the effort of one iterative layer over a set of permutations.

Furthermore, the sets $I_{\mathfrak{P}}$ define a partition of that last protocol set, $I_{\mathfrak{P}}$ has size at most N, and only one protocol per set will be visited.

Hence a lower bound on the number of different visited protocols (fast and minimal protocols) is:

$$\texttt{NFMP}(N) \stackrel{\text{def}}{=} \left|\mathfrak{S}_{N-1}^N/\mathfrak{S}_N\right| \geq \frac{((N-1)!)^{N-1}}{N}$$

This is also a rough estimate of the actual cardinality of $\mathfrak{S}_{N-1}^N/\mathfrak{S}_N$, assuming that for most protocols $|I_{\mathfrak{P}}| \sim N$.

For $N = 5$, we found that there are $\texttt{NFMP}(5) = 66,360$ different protocols up to isomorphism, which is pretty close to $\frac{(4!)^4}{5} = 66,355.2$

For $N = 6$ we get $\texttt{NFMP}(6) = 4,147,236,820 \simeq \frac{(5!)^5}{6} = 4,147,200,000$.

The backtracking consists in incrementally completing a partial protocol in every possible way while keeping track of a lower bound on the cost, and pruning as soon as an upper limit is reached (*e.g.* when the lower bound exceeds $4N - 6$)

When a complete protocol is obtained, we check if it is lexicographically minimal in its isomorphism class, in which case it can be processed or stored for further examination.

To prune even more possibilities, we can further exploit the fact that the protocols we are looking for need to be the lexicographically smallest. For example, instead of checking minimality once the protocol has been completed, it is possible to relabel the partial protocol to see that any completion of it will not be minimal. Unfortunately, in our attempt to code this trick the resulting overhead outweighed the pruning. We assume this is due to the small values of N that we could examine, and that this modification yields a faster algorithm for larger N values.

By exhaustively examining all protocols whose last line is Id, we could enumerate all fast protocols for $N \leq 6$ (Table 2).

And using backtracking as described above, we enumerated all cheap protocols for $N \leq 8$ (Table 1) while checking[7] that protocols cheaper than $\$(4N - 6)$ do not exist.

N	NCMP(N)	NCEMP(N)	ATICMP(N)	TICEMP(N)
2	1	1	0	0
3	2	2	0	0
4	9	9	0.020136	0.020136
5	61	61	0.037728	0.011069
6	663	5	0.057825	0
7	8,529	12	0.077496	0.005786
8	134,772	27	0.094730	0.008475

NCMP = number of cheap minimal protocols
NCEMP = number of cheap and equitable minimal protocols
ATICMP = average Theil index of cheap minimal protocols
TICEMP = Theil index of cheap and equitable minimal protocols

Table 1. Minimal protocols represent their isomorphism class

Table 2 also provides the number of $\$c$ protocols for $4N - 6 \leq c \leq N(N - 1)$.

[7] our Ocaml code is available from `http://www.eleves.ens.fr/home/xia/posting`

NFMP(N) ↘	$N=4$	$N=5$	$N=6$	$N=7$	$N=8$
cost = \$10	9				
cost = \$11	10				
cost = \$12	104				
cost = \$14		61			
cost = \$15		416			
cost = \$16		1,918			
cost = \$17		6,300			
cost = \$18		15,221	663		
cost = \$19		21,180	8,206		
cost = \$20		21,264	69,138		
cost = \$21			433,554		
cost = \$22			2,269,917	8,529	
cost = \$23			9,945,474	186,484	
cost = \$24			36,922,032	2,331,501	
cost = \$25			114,376,002	22,592,196	
cost = \$26			298,714,009	181,221,263	134,772
cost = \$27			628,381,792	1,263,557,229	4,745,712
cost = \$28			1,019,946,014	7,833,563,489	86,813,703
cost = \$29			1,213,515,356	43,633,739,654	unknown
cost = \$30			822,654,663	unknown	unknown
cost > \$30				unknown	unknown
TOTAL	123	66,360	4,147,236,820	unknown	unknown

Table 2. Number of protocols per N and per cost (number of isomorphism classes). Note that there are no \$13 protocols.

7 Equitableness

In $\mathfrak{P}_{\text{lin}}$, all parties but one pay a fixed fee, and one party pays a fee that increases with N. This is not an equitable protocol. We hence looked for the most equitable cheap protocol.

We measure equitableness using the Theil index:

$$T_N(\mathfrak{P}) = \frac{1}{N}\sum_{n=1}^{N}\frac{m_n}{\widetilde{m}}\log(\frac{m_n}{\widetilde{m}})$$

where m_n is the fee paid by $\mathcal{A}_n$ and

$$\widetilde{m} = \frac{1}{N}\sum_{n=1}^{N} m_n$$

is the average fee.

A smaller $T_N(\mathfrak{P})$ value expresses a more equitable protocol.

For $N = 7$, the average Theil index computed over all minimal representatives of protocol isomorphism classes is $\simeq 0.077$ whereas the minimum index is $\simeq 0.0058$, reached by the 12 FCEM-protocols given in the appendix. We also illustrate in Fig. 6 one of the 27 FCEM-protocols for $N = 8$, found by automated search.

1	5	6	4	7	2	3	0
4	5	6	3	7	2	0	1
0	1	6	3	4	5	7	2
1	5	6	4	7	2	0	3
1	5	6	3	7	2	0	4
0	1	6	4	7	2	3	5
0	1	2	3	4	5	7	6
0	1	2	3	4	5	6	7

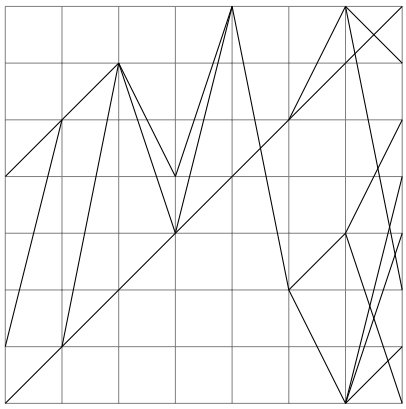


Fig. 6. Equitable protocol, $N = 8$ of cost vector $(4, 3, 3, 4, 3, 3, 3, 3)$ (example).

7.1 Symbol Insertion Experiments

It is natural to wonder if FCE-protocols can be constructed from smaller ones. To get a hint, we took all 27 eight-party FCEM-protocols $\mathfrak{P}_1, \ldots, \mathfrak{P}_{27}$ and performed the following exploration:

for $i = 1 \to 27$ **do**
  for $\ell = 1 \to 8$ **do**
    $M \leftarrow$ the matrix of $\mathfrak{P}_i$ where line ℓ was suppressed.
    $M' \leftarrow M$ where all occurences of ℓ were suppressed.
    Check if the protocol corresponding to M' is an FCE-protocol.
  end for
end for

Indeed, the above algorithm detected 168 different ways to build (non necessarily minimal) eight-party FCE-protocols by inserting new symbols into 7 seven-party FCEM-protocols. The process is illustrated in Fig. 7.

The experiment was repeated *mutatis mutandis* by eliminating all possible combinations of two rows (and their corresponding pairs of symbols). There were 136 ways to obtain eight-party FCE-protocols using symbol

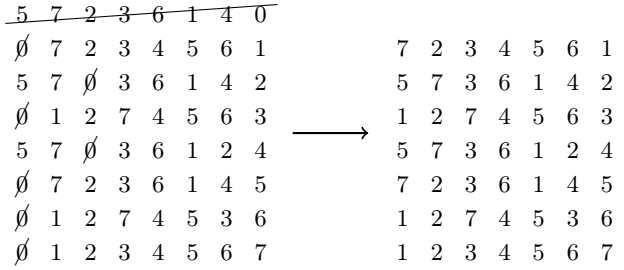


Fig. 7. Symbol Deletion Experiment (example). The deleted symbol is 0.

insertions into six-party FCEM-protocols. Only 2 protocols out of the 5 equitable six-party protocols enabled these insertions, and 17 out of the 27 eight-party FCEM-protocols could be reached that way.

Results are available on line[8].

We doubt that this process would allow to infer a general process for constructing $(N+1)$-party FCE-protocols by extending N-party FCE-protocols for the following reason: for $N = 6, 7, 8$ all FCE-protocols have 4 active parties on day $N-2$. Never 2 or 3, nor 5.

The exhaustive list of matrices for $N = 6, 7, 8$ hints that we cannot do better than 4 parties on day $N-2$. If there was an algorithm allowing to build FCE-protocols from smaller ones, this algorithm would have to add active parties on day $N-2$, and it would be unexpected for it not to work for $6, 7$ or 8 parties.

We regard this as evidence that the algorithmic construction of FCEM-protocols is a non-trivial problem.

This approach can be used to find a way of generating cheap protocols rather than equitable protocols. Indeed, we have discovered a pattern, though without using this approach, as explained in section 9.1.

8 Lower Bounds

8.1 General Case

With no conditions on the protocol's duration D we show that

$$\min_{\mathfrak{P}} \mathtt{Cost}(\mathfrak{P}, N) = 2N - 2$$

[8] `http://www.eleves.ens.fr/home/xia/posting`

as achieved by $\mathfrak{P}_{\text{seq}}$.

It should be noted that in general having some party hold a contract for several consecutive days without sending it away is an allowed "move", which is of course free of charge.

Only with the now unassumed constraint of contract validation in $N-1$ days it becomes necessary to have all contracts circulating in envelopes every day.

The same applies as well to the fact that a contract can transit through one same party multiple times.

Proof. The proof is done by induction on the number of parties N.

When $N = 1$, it is clear that $\texttt{Cost}(\mathfrak{P}, 1) = 0$.

Assume that for every N-party protocol $\mathfrak{P}'$, $\texttt{Cost}(\mathfrak{P}') \geq 2N - 2$. Let us prove that for every $(N+1)$-party protocol, $\texttt{Cost}(\mathfrak{P}) \geq 2N$.

Let $\mathfrak{P}$ be a $\$c$ $(N+1)$-party protocol.

By conveniently removing one party from $\mathfrak{P}$, we will create an N-party protocol $\mathfrak{P}'$ that costs at most $\$(c-2)$.

$$c - 2 \geq \texttt{Cost}(\mathfrak{P}')$$

Then, using the inductive hypothesis for $\mathfrak{P}'$,

$$\texttt{Cost}(\mathfrak{P}') \geq 2N - 2$$

will conclude the inductive step.

At least one party is to print an empty copy of the contract, which will be sent using one envelope. Without loss of generality, we can assume that $\mathcal{A}_N$ is one of those who print contracts, that is the party we will want to remove from this protocol.

A first protocol transformation: Instead of having $\mathcal{A}_N$ print contracts (on day 0) and send them to $\mathcal{A}_{\alpha_0}, \mathcal{A}_{\alpha_1}, \ldots$ (not necessarily on day 0), we will have $\mathcal{A}_{\alpha_0}, \mathcal{A}_{\alpha_1}, \ldots$ print these contracts. Because $\mathcal{A}_N$ is assumed to print at least one contract, at least one less envelope will be used (Fig. 8).

We now have to consider the points in time at which $\mathcal{A}_N$ receives some contracts.

This must happen at least once, as every party must receive a final copy of the contract at some point.

A second protocol transformation: The following transformation removes another envelope from the process.

- If $\mathcal{A}_N$ receives only one envelope containing only the contract that $\mathcal{A}_N$ is to own, then we can just remove this envelope from the protocol.
- Otherwise, $\mathcal{A}_N$ receives some contracts which are to be signed by $\mathcal{A}_N$. Then these contracts need to be rerouted away (not necessarily on the same day), excluding the contract that is ultimately bound to reach $\mathcal{A}_N$ that we will just remove.
  Denote by $\mathcal{A}_{\beta_0}, \mathcal{A}_{\beta_1}, \ldots$ the parties those contracts will be sent to next. There must be at least one of them, $\mathcal{A}_{\beta_0}$. Since $\mathcal{A}_N$ is to be removed, we can change the destination of the contracts to $\mathcal{A}_{\beta_0}$ instead of $\mathcal{A}_N$, and one less envelope will be used as $\mathcal{A}_{\beta_0}$ does not need to send a contract to himself (Fig. 9).

With the above two transformations, we can obtain an N-party protocol instead of an $N+1$ one, while removing at least one envelope by each transformation. Therefore the resulting protocol costs at most $\$(c-2)$.

We can conclude that for all N-party protocols, $\texttt{Cost}(\mathfrak{P}) \geq 2N-2$.

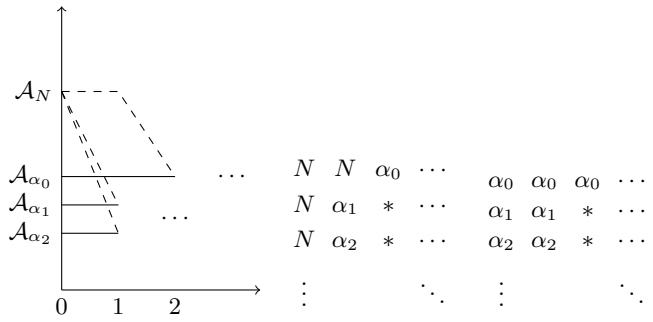


Fig. 8. Transformation on day 0 (first transformation)

8.2 Fast Protocols

Although still unsatisfactory, a lower bound $\texttt{CFP}(N) \geq 3N - 5 + \log_2(N)$ can be proven.

We first prove that $\texttt{CFP}(N) \geq 3N - 4$.

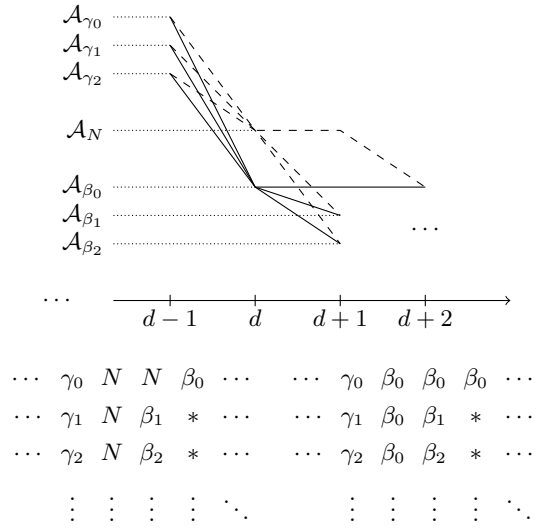


Fig. 9. Transformation when $\mathcal{A}_N$ receives envelopes (second transformation)

Proof. Assume that $\texttt{CFP}(N) \leq 3N-5$ for some N.

Since $\texttt{CFP}(2) = 2$, we can assume that $N \geq 3$.

Let $\mathfrak{P}$ be a $\$\texttt{CFP}(N)$ protocol, *i.e.* a protocol using less than $3N-5$ envelopes.

We know that on day $N-2$, exactly N envelopes are sent. Hence between days 0 and $N-3$, strictly less than $2(N-2)$ envelopes would be sent.

On at least one day $\leq N-3$, only one envelope is sent, therefore all contracts go through one same party, and on the last day the contract that this party receives would have gone through it twice, which is impossible.

The N contracts must follow N different paths between days 0 and $N-2$, as the final destination of each contract is the only party it hasn't gone through during days 0 to $N-2$. Moreover, we can bound the number of different available paths when using $3N-4+q$ envelopes by 2^{q+1}.

Proof. We say that party $\mathcal{A}_n$ is *active* on day d in protocol $\mathfrak{P}$ if $\mathcal{A}_n$ has at least one contract on day d. *i.e.* $\exists k$ such that $\mathfrak{P}_k(d) = n$.

For every active party on each day between days 1 and $N-3$, choose one envelope among those sent, we call those chosen envelopes *default* envelopes. Also choose only one default envelope on day 0.

The number of default envelopes is equal to the cumulated number of active parties in days 1 to $N-3$, plus one on day 0. That is at least $2N-5$

as a consequence of the previous proof. There are also N envelopes sent on day $N-2$.

Therefore there are at most $q+1$ non-default envelopes between days 0 and $N-2$.

We associate any path between days 0 and $N-2$ with the set L of non-default envelopes that it contains[9]. This defines an injection into the set of subsets of non-default envelopes, whose size is at most 2^{q+1}.

The reverse procedure to recover a path Δ from its associate set L consists in the following, starting on day 0:

- If no envelope sent on day 0 is in L, then path Δ starts with the default envelope.
- Otherwise there should be a unique such envelope, and this is the first envelope in the path.

The reason why we chose only one default envelope on day 0 is that we do not know yet where the path begins from. This default envelope allows to set a default starting party at the same time.

Once the first envelope in Δ is found, $\Delta(0)$ and $\Delta(1)$ are known.

We carry on by induction. On each day $d = 1, \ldots, N-2$, assume that $\Delta(d)$ is known, there is at most one envelope in L which was sent on day d. If there is none, then $\Delta(d+1)$ is the recipient of the default envelope sent by $\Delta(d)$.

A conflict in this procedure, where there are several envelopes in L among those sent on day d, means that L is not associated with any path.

This procedure shows that there are at most 2^{q+1} paths.

Since there must be at least N paths, $q \geq \log_2(N) - 1$.

In conclusion,

$$\texttt{CFP}(N) \geq 3N - 5 + \log_2(N)$$

9 Leads

Looking at the proof of the previous lower bound, it is natural to wonder whether we can improve on the lower bound of 2 envelopes per day.

This is however the best we can do as illustrated in Fig. 11.

[9] The reader is referred to Fig. 10 for a clarifying example.
An edge $(d, \alpha) - (d+1, \beta)$ means that $\mathcal{A}_\alpha$ sends an envelope to $\mathcal{A}_\beta$ on day d.
The path $(1, 5, 6, 4, 7, 2, 3)$ is associated to the set $\{(0,1) - (1,5); (2,6) - (3,4); (5,2) - (6,3)\}$ (the final destination of the corresponding contract is $\mathcal{A}_0$).
Note that all available paths in this graph are not necessarily taken by a contract, *e.g.* $(4, 5, 6, 4, 7, 2, 3)$ is associated to $\{(0,4) - (1,5); (5,2) - (6,3)\}$.

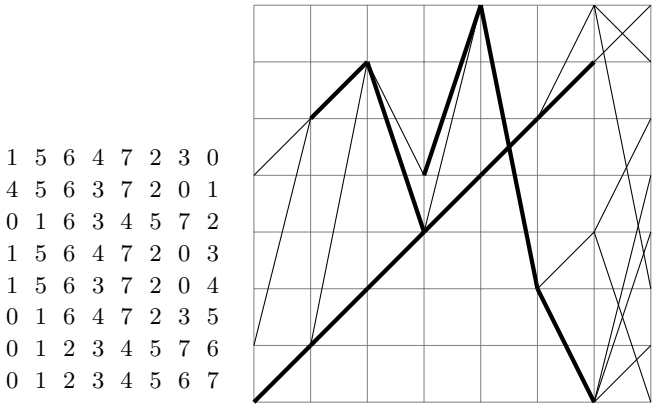


Fig. 10. Protocol from Fig. 6 where edges representing default envelopes are drawn in thick lines.

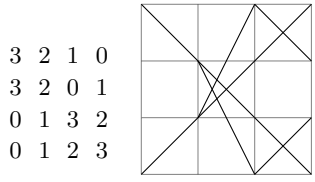


Fig. 11. On day 1, only two envelopes are sent.

This example can be generalized to all $N \geq 4$.

Other examples were found where only two envelopes were sent on a day other than $N - 3$.

9.1 A wider class of protocols

Given a cheap protocol $\mathfrak{P}$ for $N \geq 2$, such that on day 0 only two parties print empty contract copies, we can build an $(N + 1)$-party protocol verifying the same property.

This construction can produce the $\mathfrak{P}_{\text{lin}}$ protocol and many more cheap protocols that do not comply with the above property, starting from smaller-size protocols.

Extend every path in protocol $\mathfrak{P}$ by appending N at the beginning (N paths are defined that way).

Choose $n \in \mathbb{S}_N$, consider the path Δ ending at n, draw a new path that begins at n, ends at N, and follows Δ in between.

This method hints that the number of cheap protocols grows at least a fast as a factorial. Example: Fig. 12.

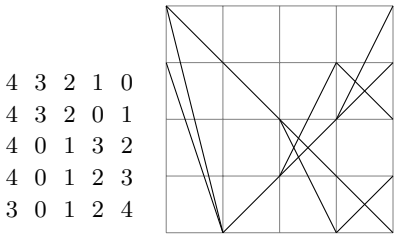


Fig. 12. Extension of the protocol of Fig. 11

10 Open Questions & Further Research

Besides proving (or refuting) conjectures 1 and 2, we encourage readers to research the following open problems:

Algorithmic construction of FCE-protocols: If equitable protocols exist for all N, design an efficient strategy for constructing FCE-protocols. Here *"efficient"* means constructing a protocol in $O(N^c)$ for some fixed c.

It seems there cannot be more than four "active" parties on day $N-2$, whatever the protocol is. If that is true then, and because there must be

N envelopes sent on this last day, one of the parties is going to pay at least $\$N/4$. The best case is $\$N/4$ for every four parties.

Assume on the contrary that equitable protocols exist for all N, or even a weaker form of equitableness where the individual cost is bounded by a constant C. On day $N-2$, N envelopes must be sent. Since every party pays less than $\$C$, there are at least N/C active parties on day $N-2$, N/C^2 on day $N-3$ and so on. There would be a tree-like structure at the end of the corresponding graph, which means a lot of active parties – whereas the idea behind the current minimal cost protocols is quite the opposite. When N is large enough, this takes a lot of envelopes to set up, in fact we believe that it takes too many, and that there won't be enough on the first days. But we are unsure and maybe such a tree would actually be possible.

Being able to look for equitable protocols for $N = 9$ would be a first step. Finding out if equitable protocols still exist for $N = 12$ or 13, would provide very strong evidence (in favor or) against the existence of equitable protocols for all N.

Finding a protocol matching (or best matching) a given cost vector: Given a cost-vector:

$$c = \{c_0, c_1, \ldots, c_{N-2}, 4N - 6 - \sum_{i=0}^{N-2} c_i\}$$

identify the protocol $\mathfrak{P}_s$ that deviates as little as possible from c.

What are the possible cost vectors? Let $\mathbb{P}_N = \{\mathfrak{P}_1, \ldots, \mathfrak{P}_{\mathtt{NCP}(N)}\}$ be all N-party FC-protocols. Let s_i denote the cost vector of $\mathfrak{P}_i \in \mathbb{P}_N$ with elements sorted by increasing order[10]. How many different s_is are there? What can be said about their frequencies?

Non-constant postage fees: We assumed that the cost of an envelope is independent of the number of contracts sent. What happens for a general cost function $f(k)$, for instance $f(k) = ak + b$ or $f(k) = a\lceil k/b \rceil$?

Continuous flow communication: This paper dealt with a latency of $N-1$ days. What happens if $N-1$ shifted protocols are started simultaneously so that $N-1$ protocols are always run in parallel? Here the

[10] *i.e.* renumbering the parties by increasing workload.

most equitable setting would be $N-6$ parties paying $\$(4N-10)$ and 6 parties paying $\$(4N-9)$ but is this achievable?[11]. If so, how regular can the spending rate of each party be? (*i.e.* avoid sudden "spending bursts").

If Conjecture 2 is true: Does relaxing the fast protocol requirement enable equitableness?

Bandwidth *vs.* **latency:** In this paper we considered latency rather than bandwidth. A trivial protocol for thwarting traffic analysis consists in rotating a single container circularly[12]. This is cheaper than opening a channel between every pair of parties (the problem addressed in this paper) but comes at the cost a reduced bandwidth. Compare the two approaches.

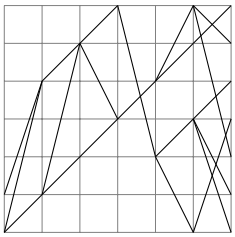
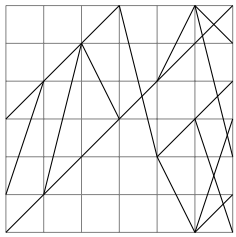
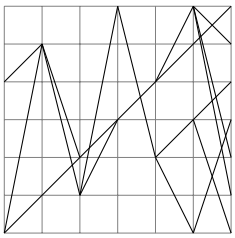
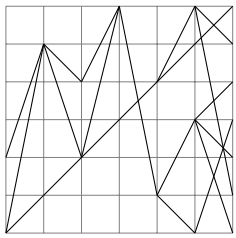
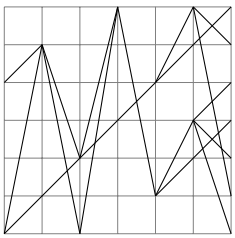
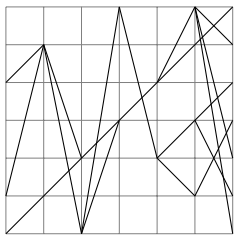
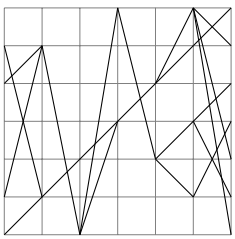
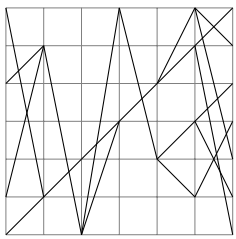
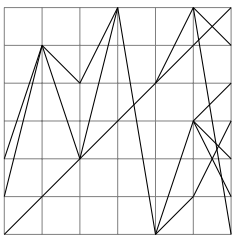
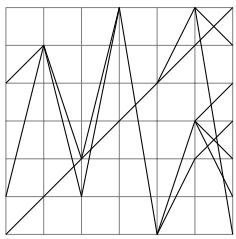
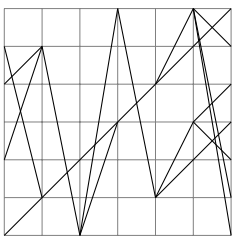
References

1. N. Ferguson and B. Schneier, *Practical Cryptography*, John Wiley & Sons, 2003.
2. K. McCurley, *Language Modeling and Encryption on Packet Switched Networks*, Advances in Cryptology - Eurocrypt 2006, Lecture Notes in Computer Science Volume 4004, 2006, pp. 359-372.
3. V. Voydoc and S. Kent, *Security mechanisms in high-level network protocols*. ACM Computing Surveys, pp. 135–171, 1983.

[11] *e.g.* if we launch 7 shifted instances of Fig. 6 we get a very uneven split of cost where $\mathcal{A}_0$ and $\mathcal{A}_3$ pay \$4 every day (*i.e.* a total of \$28 each) whereas the other 6 parties pay \$3 every day (*i.e.* a total of \$21 each).

[12] In other words, on day i do $\mathcal{A}_i \overset{1}{\rightsquigarrow} \mathcal{A}_{i+1 \bmod N}$.

11 Appendix: FCEM-Protocols for $N = 7$

1 4 5 6 2 3 0 0 4 5 6 2 3 1 0 1 5 3 4 6 2 1 4 5 6 2 0 3 0 1 5 6 2 3 4 0 1 2 3 4 6 5 0 1 2 3 4 5 6	
1 4 5 6 2 3 0 3 4 5 6 2 0 1 0 1 5 3 4 6 2 1 4 5 6 2 0 3 0 1 5 6 2 3 4 0 1 2 3 4 6 5 0 1 2 3 4 5 6	
4 5 1 6 2 3 0 0 5 2 3 4 6 1 0 5 1 3 4 6 2 4 5 1 6 2 0 3 0 5 1 6 2 3 4 0 1 2 3 4 6 5 0 1 2 3 4 5 6	
2 5 4 6 1 3 0 0 5 2 3 4 6 1 0 5 4 6 1 3 2 2 5 4 6 1 0 3 0 5 2 6 1 3 4 0 1 2 3 4 6 5 0 1 2 3 4 5 6	
4 5 2 6 1 3 0 0 5 2 3 4 6 1 4 5 0 6 1 3 2 4 5 0 6 1 2 3 0 5 2 6 1 3 4 0 1 2 3 4 6 5 0 1 2 3 4 5 6	
1 5 2 3 4 6 0 4 5 0 6 2 3 1 1 5 0 3 4 6 2 4 5 0 6 2 1 3 1 5 0 6 2 3 4 0 1 2 3 4 6 5 0 1 2 3 4 5 6	
5 1 2 3 4 6 0 4 5 0 6 2 3 1 1 5 0 3 4 6 2 4 5 0 6 2 1 3 1 5 0 6 2 3 4 0 1 2 3 4 6 5 0 1 2 3 4 5 6	
6 1 2 3 4 5 0 4 5 0 6 2 3 1 1 5 0 3 4 6 2 4 5 0 6 2 1 3 1 5 0 6 2 3 4 0 1 2 3 4 6 5 0 1 2 3 4 5 6	
1 5 2 3 4 6 0 2 5 4 6 0 3 1 1 5 4 6 0 3 2 2 5 4 6 0 1 3 1 5 2 6 0 3 4 0 1 2 3 4 6 5 0 1 2 3 4 5 6	
1 5 2 3 4 6 0 4 5 2 6 0 3 1 4 5 1 6 0 3 2 4 5 1 6 0 2 3 1 5 2 6 0 3 4 0 1 2 3 4 6 5 0 1 2 3 4 5 6	
5 1 2 3 4 6 0 2 5 0 3 4 6 1 4 5 0 6 1 3 2 4 5 0 6 1 2 3 2 5 0 6 1 3 4 0 1 2 3 4 6 5 0 1 2 3 4 5 6	
6 1 2 3 4 5 0 2 5 0 3 4 6 1 4 5 0 6 1 3 2 4 5 0 6 1 2 3 2 5 0 6 1 3 4 0 1 2 3 4 6 5 0 1 2 3 4 5 6