Pattern Recognition Letters 5 (1987) 119–128
North-Holland

February 1987

Representation of patterns of symbols by equations with applications to puzzle solving

Haim SHVAYTSER and Shmuel PELEG
Department of Computer Science, The Hebrew University of Jerusalem, 91904 Jerusalem, Israel

Received September 1985
Revised 25 January 1986

Abstract: An algorithm is presented for obtaining a representation of patterns in strings of symbols. The patterns are represented by equations among bit values in a binary encoding of the symbols. An application is described where the symbols are English letters and the strings are words. The patterns enable the solution of cryptograms, and the construction of crossword puzzles.

Key words: Relational patterns, puzzle solving.

1. Introduction

This paper considers the detection of repeated patterns within a string of symbols, where the patterns are not known in advance. If a pattern of symbols repeats itself throughout the string, many properties of the entire string can be learned by considering the pattern and the way in which it repeats itself. In some cases, several of such patterns can completely describe the string. However, the detection of such patterns is usually difficult, because they may take complicated forms and repeat themselves irregularly.

In this work we consider a simplified version of the above problem. The string is assumed to be divided in some 'natural' way into substrings of equal length; only patterns that appear in all substrings are considered. It seems at first that the solution of the simplified problem is of no special interest, since patterns are not usually found after an arbitrary partitioning of the string. Our results show, however, that very complex patterns always exist for any partition of the string. These patterns cannot always be described as simple *arrangements* of symbols, but rather as *relations* that exist among the symbols. We refer to these patterns as *relational patterns* and will give their exact definition in Section 2.

As an example, let the symbols be the English alphabet and the string be an English text. An arbitrary partition of the text into substrings of five letters each will not enable the recognition of patterns among words, but will enable the detection of the following relation among letters: unless the letter 'q' is last in a substring, it will be followed by the letter 'u'.

We will show that the problem of determining relational patterns is equivalent to determining constraints among events, for which a theoretical framework was given in [1]. Using some results from [1] will enable the development of an efficient algorithm for the recognition of these patterns. We will also discuss the application of the algorithm to some NP-complete problems such as crossword puzzle construction and substitution cipher problems.

This research has been supported by a grant from the Israel National Council for Research and Development.

2. A formal definition of relational patterns

Let $\sum = \{\alpha_1, \ldots, \alpha_m\}$ be a finite set of symbols, and $S \in \sum^*$ be a string of these symbols. To simplify notation and analysis the symbols are assumed to be encoded into bits (strings of zeros and ones). As will be seen later, different encodings do not change the existence of relational patterns, but may change their complexity. Therefore, it is possible to assume without loss of generality that the string S is a string of bits: $S \in \{0,1\}^*$.

We consider a 'natural' partition of S into n-tuples of bits

$$S = S_1 S_2 \cdots S_k, \quad S_i \in \{0,1\}^n, \ i = 1, \ldots, k.$$

We denote by X_j^i the j-th bit of S_i, i.e.,

$$S_i = X_1^i X_2^i \cdots X_n^i.$$

In their most general form, the relational patterns can be expressed by a functional f.

Definition. A relational pattern is a functional $f(X_1, \ldots, X_n)$ whose value is 0 for all substrings S_i, i.e.,

$$f(X_1^i, \ldots, X_n^i) = 0, \quad i = 1, \ldots, k.$$

Example 1. Let $\sum = \{a, b, c\}$. Consider a string $S \in \sum^*$ partitioned into the following substrings of two characters each:

cb cc da.

Using the encoding

$$a - 00, \quad b - 01, \quad c - 10, \quad d - 11,$$

the binary representations of the substrings are:

1001 1010 1100.

The values of the X_j^i for the three substrings are given in the following table:

	X_1	X_2	X_3	X_4
cb	1	0	0	1
cc	1	0	1	0
da	1	1	0	0

In this example, relational patterns are

$$P_1: \ X_1 - 1,$$

$$P_2: \ X_2 + X_3 + X_4 - 1$$

(since $P_1 \equiv P_2 \equiv 0$ in all three substrings). Notice that the above patterns completely characterize the original substrings, since by regarding them as equations with unknowns that can have only bivalent (zero or one) values, the following system of equations is obtained:

$$X_1 = 1, \qquad X_2 + X_3 + X_4 = 1.$$

with the solutions:

	X_1	X_2	X_3	X_4
solution 1	1	0	0	1
solution 2	1	0	1	0
solution 3	1	1	0	0

These three solutions correspond exactly to the original three substrings.

3. Relational patterns and constraints among events

If the bits in the representation of the substrings are taken as events that can either happen ($X_i = 1$) or not happen ($X_i = 0$), and the substrings are taken as permitted relations among the events, the definition of relational patterns coincides with the definition of constraints among events as given in [1]. Therefore, it is possible to apply some of the theoretical results obtained in [1] for our case. Let V be the following set of variables:

$$V = \{1, X_1, \ldots, X_n, X_1 X_2, \ldots, X_1 \cdots X_n\}$$

i.e., V includes $1, X_1, \ldots, X_n$, and all their possible products. Consider linear combinations of the variables $y_1, \ldots, y_k \in V$:

$$a_1 \cdot y_1 + \cdots + a_k \cdot y_k \tag{1}$$

where $a_1, \ldots, a_k$ are real numbers. These linear combinations are actually multilinear forms in the variables $1, X_1, \ldots, X_n$. Since each substring $S_i = X_1^i X_2^i \cdots X_n^i$ determines the values of all variables in V (the variable X_j in V is being assigned the valued X_j^i of S_i), it also determines the values of all the multilinear forms of (1). Given a set of substrings, two differently represented multilinear forms may have identical values under all substrings. For example, $X_2 + X_3$ and $1 - X_4$ have identical values for all substrings of Example 1. We will consider much multilinear forms as identical (under the given set of substrings).

The following are direct consequences of the equivalence between constraints among events and relational patterns. The proofs (for constraints) appear in [1].

1. The set of linear combinations of variables from a subset $V^+ \subset V$ is a finite dimensional vector space. (Notice that the variables in V^+ are not necessarily linearly independent because differently represented multilinear forms may still be identical.)

2. Every relational pattern can be expressed as a linear combination of variables from V.

3. The set of all relational patterns that can be expressed as a linear combination of variables from a subset $V^+ \subset V$ is a finite dimensional vector space. (Notice that all relational patterns are identified with '0' in the vector space described in 1. above).

4. If the variables of the subset $V^+ \subset V$ are $y_1, \ldots, y_k$, where $y_1, \ldots, y_d$ are linearly independent in the vector space described in 1. above, and $y_{d+1}, \ldots, y_k$ depend on $y_1, \ldots, y_d$, then $y_{d+1}, \ldots, y_k$ can be expressed as linear combinations of $y_1, \ldots, y_d$, i.e.

$$y_i = L_i(y_1, \ldots, y_d), \quad i = d+1, \ldots, k.$$

In this case,

$$y_i - L_i(y_1, \ldots, y_d), \quad i = d+1, \ldots, k,$$

are relational patterns, and they form a basis to the vector space of all relational patterns described in 3, above.

We define the *complexity of a relational pattern* as the degree of its multilinear form.

5. For any string S and a partition into substrings of length n there always exist relational patterns of complexity n.

4. An algorithm for determining relational patterns

Although relational patterns of complexity n always exist, they may be too complicated for practical problems. If, for example, relational patterns are to be determined for English words of five letters each, and each letter is represented by six bits, the relational patterns of complexity n are described by equations of degree 30. However, much simpler relational patterns cam be found for most practical problems. In this section we present as algorithm for determining all the linearly independent relational patterns that can be expressed as linear combinations of variables from a subset $V^+ \subset V$ (as defined in Section 3). Since the complexity of these relational patterns cannot exceed the maximal degree of a monomial in V^+, choosing the elements of V^+ with bounded degree guarantees the detection of low complexity relational patterns (if such patterns exist).

Let $V^+ = \{y_1, \ldots, y_k\}$. The relational patterns are determined from the co-occurrence matrix, which is defined as follows:

Definition. The co-occurrence matrix $R = (R_{ij})$ is the $k \times k$ matrix in which R_{ij} is the number of substrings of which $y_i = 1$ and $y_j = 1$.

Example 2. In Example 1 let V^+ be $\{y_1, y_2, y_3, y_4, y_5\}$, where

$$y_1 = 1, \quad y_2 = X_1, \quad y_3 = X_2, \quad y_4 = X_3, \quad y_5 = X_4.$$

The co-occurrence matrix is

$$R = \begin{pmatrix} 3 & 3 & 1 & 1 & 1 \\ 3 & 3 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

where $R_{11} = 3$ since in the three substrings $y_1 = 1$; $R_{23} = 1$ since only in a single substring $y_2 = X_1 = 1$ and $y_3 = X_2 = 1$, etc.

The co-occurrence matrix can be used to determine relational patterns because of some useful properties:

(a) R is symmetric since by definition $R_{ij} \equiv R_{ji}$.

(b) If $y_1, \ldots, y_k$ are independent, R is positive definite.

Proof. Consider the expression

$$P_i = (a_1 \cdot y_1^i + \cdots + a_k \cdot y_k^i)^2$$

where $a_1, \ldots, a_k$ are arbitrary constants, and y_j^i is the value of $y_j \in V^+$ in the substring S_i. Clearly,

$$P_i \geq 0, \quad \forall i.$$

Therefore,

$$\sum_i P_i \geq 0$$

and

$$\begin{aligned} 0 \leq \sum_i P_i &= \sum_i (a_1 \cdot y_1^i + \cdots + a_k \cdot y_k^i)^2 \\ &= \sum_i \sum_{s,t} a_s a_t y_s^i y_t^i = \sum_{s,t} a_s a_t \sum_i y_s^i y_t^i \\ &= \sum_{s,t} a_s a_t R_{st}. \end{aligned}$$

If $y_1, \ldots, y_k$ are independent, there must exist a substring S_i such that

$$a_1 \cdot y_1^i + \cdots + a_k \cdot y_k^i \neq 0,$$

since otherwise

$$a_1 \cdot y_1^i + \cdots + a_k \cdot y_k^i = 0, \quad \forall i,$$

so that

$$a_1 \cdot y_1 + \cdots + a_k \cdot y_k = 0$$

as a vector space identity in contradiction to their independence. Therefore,

$$\sum_{s,t} a_s a_t R_{st} = \sum_i (a_1 \cdot y_1^i + \cdots + a_k \cdot y_k^i)^2 > 0. \quad \square$$

(c) If $y_1, \ldots, y_k$ are dependent, R is singular.

Proof. Since $y_1, \ldots, y_k$ are dependent, there exist coefficients $a_1, \ldots, a_k$ not identically zero such that

$$a_1 \cdot y_1 + \cdots + a_k \cdot y_k = 0,$$

i.e.,

$$a_1 \cdot y_1^i + \cdots + a_k \cdot y_k^i = 0, \quad \forall i.$$

Multiplying the above equation by y_j^i and summing over i we get

$$a_1 \cdot R_{1j} + \cdots + a_k \cdot R_{kj} = 0, \quad \forall j. \quad \square \tag{2}$$

(d) If $y_1, \ldots, y_{k-1}$ are independent, and $y_1, \ldots, y_k$ are dependent, then

$$y_k = a_1 \cdot y_1 + \cdots + a_{k-1} \cdot y_{k-1}$$

where $a_1, \ldots, a_{k-1}$ can be obtained from the system of equations

$$R \cdot \begin{pmatrix} a_1 \\ \vdots \\ a_{k-1} \end{pmatrix} = \begin{pmatrix} R_{1,k} \\ \vdots \\ R_{k-1,k} \end{pmatrix}$$

where R is the co-occurrence matrix for the variables $y_1, \ldots, y_{k-1}$. (In linear estimation theory, this system of equations is called the Yule–Walker equations [2].)

Proof. Because $y_1, \ldots, y_{k-1}$ are independent, there always exist coefficients a_i in (2) such that $a_k = -1$. $\square$

The algorithm is based on the Choleski method for decomposition of symmetric positive definite matrices [3]. For a matrix $A = (a_{ij})$ which is symmetric and positive definite, there exists a lower triangular matrix $L = (l_{ij})$ such that

$$A = LL^{\mathrm{T}}.$$

The lower triangular matrix L can be obtained by the following Choleski decomposition algorithm which is listed here in reversed order.

```
for $i = 1, 2, \ldots, n$,
{
    for $j = 1, \ldots, i-1$,  $l_{ij} = \dfrac{a_{ij} - \sum_{t=1}^{j-1} l_{it} l_{jt}}{l_{jj}}$,
    $l_{ii} = \sqrt{a_{ii} - \sum_{t=1}^{i-1} l_{it}^2}$
}
```

Whenever A is positive definite, $l_{ii} > 0$ $\forall i$. If, however, A is singular, there exists i such that $l_{ii} = 0$. Because of the properties of the co-occurrence matrix, the Choleski decomposition algorithm can be used together with property (4) of the relational patterns (see Section 3) to detect relational patterns. The resulting algorithm will generate a basis for the set of *all* relational patterns that can be expressed as linear combinations of the variables

$$\{y_1, \ldots, y_k\} = V^+ \subset V.$$

Input: The co-occurrence values of $y_1, \ldots, y_k$.
Output: A list of relational patterns.
Method: the algorithm builds a subset $I \subset V^+$ of independent variables and generates the lower triangular matrix Z for which $ZZ^{\mathrm{T}} = R$, where R is

the co-occurrence matrix of the variables in I. It outputs the relational patterns which are expressed as dependencies of variables from $V^+ - I$. Because of Property 4 of Section 3, these relational patterns form a basis to the set of all relational patterns that are linear combinations of variables from V^+.

The algorithm: In the algorithm let $|I|$ denote the number of elements in I. We assume without loss of generality that the variables in I are $y_1, \dots, y_{|I|}$. u and u_j are auxiliary variables, and $R(X, Y)$ denotes the co-occurrence value of the two variables $X, Y \in V^+$.

```
Initially $I = \emptyset$.
While $V^+$ is not empty
{
Remove an element $Y$ from $V^+$.
    For $j = 1, \dots, |I|$, set $u_j = \frac{R(Y, y_j) - \sum_{t=1}^{j-1} u_t z_{jt}}{z_{jj}}$,
    $u = \sqrt{R(Y, Y) - \sum_{t=1}^{|I|} u_t^2}$.
If $u \neq 0$, add the row $u_1, u_2, \dots, u_{|I|}, u$ to $Z$, i.e.,
    $z_{|I|+1, j} \leftarrow u_j, \quad j = 1, \dots, |I|, \qquad z_{|I|+1, |I|+1} \leftarrow u$
and add $Y$ to $I$.
If $u = 0$, output the relational pattern
    $a_1 y_1 + \cdots + a_{|I|} y_{|I|} - Y$
where $a_1, \dots, a_{|I|}$ can be determined by forward and backward substitutions from
    $ZZ^T \begin{pmatrix} a_1 \\ \vdots \\ a_{|I|} \end{pmatrix} = \begin{pmatrix} R(y_1, Y) \\ \vdots \\ R(y_{|I|}, Y) \end{pmatrix}$
}
```

Example 3. In this example we use the co-occurrence matrix of Example 2. The indices of u_j and z_{ij} are taken as the indices of the corresponding variables y_j.

$$I = \emptyset.$$

Iteration 1:

$$Y \leftarrow y_1, \quad u = \sqrt{3} \neq 0, \quad z_{11} = \sqrt{3}, \quad I = \{y_1\}.$$

Iteration 2:

$$Y \leftarrow y_2, \quad u_1 = \sqrt{3}, \quad u = 0.$$

Solve:

$$\sqrt{3} \cdot \sqrt{3} \cdot a_1 = 1 \quad \Rightarrow \quad a_1 = 1.$$

Output: The relational pattern $y_2 - y_1$ (i.e. $X_1 - 1$).

Iteration 3:

$$Y \leftarrow y_3,$$

$$u_1 = \frac{1}{\sqrt{3}}, \quad u = \sqrt{\frac{2}{3}} \neq 0,$$

$$z_{31} = \frac{1}{\sqrt{3}}, \quad z_{33} = \sqrt{\frac{2}{3}},$$

$$I = \{y_1, y_3\}$$

Iteration 4:

$$Y \leftarrow y_4,$$

$$u_1 = \frac{1}{\sqrt{3}}, \quad u_3 = -\frac{1}{\sqrt{6}}, \quad u = \frac{1}{\sqrt{2}} \neq 0,$$

$$z_{41} = \frac{1}{\sqrt{3}}, \quad z_{43} = -\frac{1}{\sqrt{6}}, \quad z_{44} = \frac{1}{\sqrt{2}},$$

$$I = \{y_1, y_3, y_4\}.$$

Iteration 5:

$$Y \leftarrow y_5,$$

$$u_1 = \frac{1}{\sqrt{3}}, \quad u_3 = -\frac{1}{\sqrt{6}}, \quad u_4 = -\frac{1}{\sqrt{2}}, \quad u = 0.$$

Solve:

$$\begin{vmatrix} \sqrt{3} & & \\ \frac{1}{\sqrt{3}} & \sqrt{\frac{2}{3}} & \\ \frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{6}} & \frac{1}{\sqrt{2}} \end{vmatrix} \cdot \begin{vmatrix} \sqrt{3} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\ & \sqrt{\frac{2}{3}} & -\frac{1}{\sqrt{6}} \\ & & \frac{1}{\sqrt{2}} \end{vmatrix} \cdot \begin{vmatrix} a_1 \\ a_3 \\ a_4 \end{vmatrix} = \begin{vmatrix} 1 \\ 0 \\ 0 \end{vmatrix}$$

to obtain

$$a_1 = 1, \quad a_3 = -1, \quad a_4 = -1.$$

Output: The relational pattern $y_5 - y_1 + y_3 + y_4$ (i.e. $X_2 + X_3 + X_4 - 1$).

The two relational patterns generated by the algorithm are P_1 and P_2 initially mentioned in Example 1 (Section 2).

5. Some applications of relational patterns

In this section we show how relational patterns in English words can be used to solve simple cryptograms and to construct crossword puzzles.

5.1. Complexity considerations

The problems we discuss belong to the class of search problems whose corresponding decision problems are NP-complete, and therefore are NP-hard [4]. This means that an efficient algorithm cannot be found for the general case (assuming $P \neq NP$). Our method will show how to reduce the solution of these problems to the solution of a system of equations in bivalent unknowns. Although solving such a system is also NP-hard [4] (in fact, even a solution of a single linear equation in bivalent unknowns is NP-complete, because it can be reduced to the knapsack problem), efficient heuristic methods are known to yield good results when the equations involved are of low degree. If simple relational patterns exist, they can be used to generate a system of equations of low degree for the solution of these problem. These equations will approximate the search problems in the following sense: each solution of the search problem will also be a solution of the system of equations (i.e. the solution set of the search problem is a subset of the solution set of the system of equations). In many practical cases, simple relational patterns yield a system of equations whose solutions are exactly the same as the solution set of the search problem. In our experiments, the bivalent equations were solved using the algorithm suggested in [5] for linear equations, and the ideas of additive penalties [6] to extend the algorithm to the nonlinear case. The technical details of these algorithms are omitted.

5.2. Crossword puzzle construction

The following crossword puzzle construction problem was proved to be NP-complete by Lewis and Papadimitriou (see [4]): Given a finite set $W \subset \sum^*$ of words, and an $n \times n$ matrix of black/white squares, is it possible to fill the white squares with symbols from $\sum$ such that any maximal (greater than 1) horizontal or vertical contiguous sequence of white squares is a word in W? (i.e. a crossword puzzle according to the usual rules).

We are interested in the corresponding search problem: Given a set of words $W \subset \sum^*$ and a crossword puzzle matrix, find *all* crossword puzzles that can be constructed from the words in W and the crossword puzzle matrix. Using relational patterns, the problem can be approximated by a system of equations in bivalent unknowns. Let $|\sum| = m$. Each symbol $\alpha \in \sum$ can be encoded using $\lceil \log m \rceil$ bits. Let the white squares of the crossword puzzle matrix be $c_1, \ldots, c_r$. The unknowns to be determined are the $r \cdot \lceil \log m \rceil$ bits in the representation of all symbols to be inserted into the white squares. Since every maximal horizontal or vertical contiguous sequence of white squares must be a word in W, the relational patterns in these words imply relational patterns among the unknown bits which can be taken as equations.

Example 4. Consider a 2×2 crossword puzzle matrix where all squares are white, and the set of words as in Example 1. There are eight unknowns $x_1, \ldots, x_8$, two for each square:

x_1x_2	x_3x_4
x_5x_6	x_7x_8

The two relational patterns that were found earlier for this example are:

$$X_1 - 1, \qquad X_2 + X_3 + X_4 - 1,$$

which imply the following system of equations:
For the upper horizontal word:

$$x_1 = 1, \qquad x_2 + x_3 + x_4 = 1.$$

For the lower horizontal word:

$$x_5 = 1, \qquad x_6 + x_7 + x_8 = 1.$$

For the left vertical word:

$$x_1 = 1, \qquad x_2 + x_5 + x_6 = 1.$$

For the right vertical word:

$$x_3 = 1, \qquad x_4 + x_7 + x_8 = 1.$$

There are two solutions to these equations:

$$x_1 = 1, \quad x_2 = 0, \quad x_3 = 1, \quad x_4 = 0,$$
$$x_5 = 1, \quad x_6 = 0, \quad x_7 = 0, \quad x_8 = 1$$

and

$$x_1 = 1, \quad x_2 = 0, \quad x_3 = 1, \quad x_4 = 0,$$
$$x_5 = 1, \quad x_6 = 0, \quad x_7 = 1, \quad x_8 = 0,$$

which are the two crossword puzzles:

c	*c*
c	*b*

c	*c*
c	*c*

Further experimental results are described in Section 6.

5.3. *Substitution ciphers*

In popular crossword puzzle magazines, puzzles such as 'cryptoquizzes' and 'cryptograms' are very popular [7]. In these puzzles, a sentence or a list of related words are put into a simple code in which another letter of the alphabet is substituted for the original letter. A solution to these puzzles is a letter mapping which assigns a plaintext letter to each ciphertext letter, thus creating a sentence which makes sense in English.

As an encryption scheme, the substitution cipher has almost no practical use, since the mapping can be found easily by using relative frequencies of n-tuples of letters in English. Shannon, in [8], showed that less than thirty letters are almost always enough to uniquely determine the mapping, when this mapping is a permutation. Computer programs for solving substitution ciphers usually require a much larger ciphertext, and use approximations to obtain probabilities of large n-tuples of English text. See for example the use of relaxation algorithms in [9]. Another disadvantage of algorithms based on probabilities is that they produce a unique result which is best according to some measure. When the encrypted text is short, there may be several solutions that make sense, and the 'true' solution may be missed. In order to avoid these difficulties we suggest a non-probabilistic model of the substitution cipher problem. It will be shown that the problem in our model is NP-complete, and a solution algorithm based on relational patterns will be suggested. In Section 6, the suggested method will be applied for 'real problems', using probabilities in a new way.

The following is the non-probabilistic version of the substitution cipher problem. The problem is first presented as a decision problem to show its NP-completeness, although we are interested in its corresponding search problem.

Instance: Two alphabets Σ_1 and Σ_2, and two sets of words $W_1 \subset \Sigma_1^*$ and $W_2 \subset \Sigma_2^*$, where each word is of length less than a fixed number k. W_1 describes the plaintext and W_2 describes the ciphertext.

Question: Can a function $f\colon \Sigma_2 \to \Sigma_1$ be found such that each ciphertext word $w_2 \in W_2$ is mapped into a plaintext word $w_1 \in W_1$?

We consider two versions of this problem, where the function f is either an isomorphism (one-to-one) or homomorphism. In the case of isomorphism, the function f is (or can be extended to) a permutation; when f is a homomorphism, an extension to a permutation is usually impossible, and the encryption process can be viewed as nondeterministic (f has no inverse).

Example 5. The substrings of Example 1 are used as the plaintext words, i.e.,

$$\Sigma_1 = \{a, b, c\}, \qquad W_1 = \{cb, cc, da\}.$$

Given the encrypted word XX:

$$\Sigma_2 = \{X\}, \qquad W_2 = \{XX\}$$

and the only solution for both isomorphism and homomorphism is

$$f(X) = c.$$

Given the encrypted word XY:

$$\Sigma_2 = \{X, Y\}, \qquad W_2 = \{XY\}.$$

There are two solutions for isomorphism:

$$f_1(X) = c, \quad f_1(Y) = b$$

and

$$f_2(X) = d, \quad f_2(Y) = a.$$

In the homomorphism case there is an additional solution:

$f_3(X)=c, \quad f_3(Y)=c.$

Theorem. *Both versions of the substitution cipher are NP-complete.*

Proof. Membership in NP is simple since a non-deterministic machine can 'guess' a function f (isomorphism or homomorphism) and then check polynomially whether each word is mapped as required. To prove completeness, a transformation to the subgraph isomorphism/homomorphism problem is shown. (These problems are both NP-complete [4].)

Given two graphs $G=(V_1,E_1)$ and $H=(V_2,E_2)$ the subgraph isomorphism/homomorphism problem is whether there exist a subgraph of G which is isomorphic/homomorphic to H. The reduction to the substitution cipher is as follows:

$$\Sigma_1=V_1, \quad W_1=E_1, \qquad \Sigma_2=V_2, \quad W_2=E_2,$$

i.e. for each edge (v_1,v_2) in G, a two letter word v_1v_2 is in W_1, and the same for the edges of H and the words in W_2. Clearly, a function f which is isomorphism/homomorphism exists for the substitution cipher if and only if the subgraph isomorphism/homomorphism problem has a solution. □

5.4. *Solving substitution ciphers using relational patterns*

The set of plaintext words W_1 can be searched for relational patterns where each subset of words having the same length is treated separately. As before, these relational patterns are relations among bits in the representation of the symbols in Σ_1. (Let the number of bits needed for each symbol be r.) Therefore, the functions

$$f: \Sigma_2 \to \Sigma_1$$

which are solutions to be substitution cipher problem, can also be represented as

$$f: \Sigma_2 \to \{0,1\}^r$$

where $\{0,1\}^r$ are strings of r bits. Treating the bits in the representation of $f(\alpha_i)$, $\alpha_i \in \Sigma_2$ as unknowns, the relational patterns imply a set of equations for each word $w_2 \in W_2$.

Example 6. For the same cases as Example 5, if

$$\Sigma_2=\{X\}, \qquad W_2=\{XX\},$$

the unknowns are x_1,x_2, the binary representation of $f(X)$. The decoded representation of the word XX is therefore $x_1x_2x_1x_2$. The equations induced by the relational patterns of Example 1 are:

$$x_1=1, \quad x_2+x_1+x_2=1,$$

with the unique solution

$$x_1=1, \quad x_2=0,$$

i.e.,

$$f(X)=c.$$

For

$$\Sigma_2=\{X,Y\}, \qquad W_2=\{XY\},$$

the unknowns are x_1,x_2,x_3,x_4 where

$$f(X)=x_1x_2, \quad f(Y)=x_3x_4.$$

The decoded representation of XY is therefore $x_1x_2x_3x_4$ and the equations are

$$x_1=1, \qquad x_2+x_3+x_4=1$$

with the three solutions:

$$x_1=1, \quad x_2=1, \quad x_3=0, \quad x_4=0;$$
$$x_1=1, \quad x_2=0, \quad x_3=1, \quad x_4=0;$$
$$x_1=1, \quad x_2=0, \quad x_3=0, \quad x_4=1;$$

i.e.,

$$f_1(X)=d, \quad f_1(Y)=a;$$
$$f_2(X)=c, \quad f_2(Y)=c;$$
$$f_3(X)=c, \quad f_3(Y)=b;$$

Further experimental results are described in Section 6.

6. Experimental results

The non-probabilistic version of the substitution cipher described in Section 5 requires a list of all words in the language. This requirement is unrealistic, since in practice almost any combination of letters may appear as a rare word (perhaps a spelling mistake). The method suggested here for the use of probabilities in handling substitution ciphers is a simple technique for transforming the probabilistic problem into the discrete problem that was discussed in Section 5. As certain words appear in English much more frequently than others, we can choose as W_1 the set of the most frequent plaintext words and as W_2 the set of the most frequent ciphertext words. If W_1 is long enough, and the ciphertext is also long enough, by choosing W_2 as a small set of words we get a high probability that the decoded text of W_2 will be included in W_1.

In our experiments, the set W_1 was chosen as the set of the 200 most frequent words in English, taken from [10]. It was found that about twenty ciphertext words were enough to almost uniquely determine the homomorphic transformations, so that the condition for the success of this method is that the ciphertext is long enough to have its most frequent twenty words among the 200 most frequent words in English.

The relational patterns that were found among the 200 most frequent English words support our assumption about the existence of simple relational patterns. Although patterns of complexity 1 (linear) were very rare, patterns of complexity 2 were enough to completely determine the text. Using the encoding:

a - 00000	n - 01101
b - 00001	o - 01110
c - 00010	p - 01111
d - 00011	q - 10000
e - 00100	r - 10001
f - 00101	s - 10010
g - 00110	t - 10011
h - 00111	u - 10100
i - 01000	v - 10101
j - 01001	w - 10110
k - 01010	x - 10111
l - 01011	y - 11000
m - 01100	z - 11001

Examples of relational patterns for two letter words (with the bit representation $x_1, \ldots, x_{10}$) are:

$$x_6 + x_8 = 1,$$

$$2x_1 - x_1x_3 - x_1x_4 - x_1x_6 - x_1x_7 = 0.$$

An example of a relational pattern for the three letter words (with the bit representation $x_1, \ldots, x_{15}$) is:

$$\begin{aligned} &4x_1 + 2x_3 + 2x_5 - x_6 - x_8 - x_9 + 3x_{10} + 2x_{11} + 2x_{13} \\ &\quad + 2x_{15} + 1.5\,x_1x_2 - 2x_1x_3 - 1x_1x_4 - 2x_1x_5 \\ &\quad + 3x_1x_6 + 2x_1x_7 + 2x_1x_8 - 2x_1x_9 - 1.5\,x_1x_{11} \\ &\quad + 0.5\,x_1x_{12} - 2x_1x_{13} + 0.5\,x_1x_{14} - 3x_1x_{15} \\ &\quad + 2x_2x_3 - 2x_2x_4 + 3x_2x_6 + 2x_2x_8 - 2x_2x_{11} \\ &\quad - 2x_2x_{13} - 2x_2x_{15} - 2x_3x_5 - 1x_3x_8 + x_3x_9 \\ &\quad - x_3x_{10} = 4. \end{aligned}$$

In general, twenty words gave few solutions for homomorphism, and usually a unique solution for isomorphism. The following is an example.

The ciphertext: A ARE AS AT FOR HE HIS IN IS IT OF ON THAT THE THEY TO WAS WITH YOU

Solutions:

	f_1	f_2	f_3	f_4	f_5	f_6
A	a	a	a	a	a	a
E	e	e	e	e	e	e
F	f	f	f	f	f	f
H	h	h	h	h	h	h
I	i	i	i	i	i	i
N	n	f	f	f	n	n
O	o	o	o	o	o	o
R	r	r	r	r	r	r
S	s	s	s	s	s	s
T	t	t	t	t	t	t
U	u	u	w	t	w	t
W	w	w	w	w	w	w
Y	y	y	n	n	n	n

The resulting plaintext for each of the solutions is:

f_1: a are as at for he his in is it of on that the they to was with you.

f_2: a are as at for he his if is it of of that the they to was with you.

f_3: a are as at for he his if is it of of that the then to was with now.

f_4: a are as at for he his if is it of of that the then to was with not.

f_5: a are as at for he his in is it of on that the then to was with now.

f_6: a are as at for he his in is it of on that the then to was with not.

Notice that only f_1 is a solution for the isomorphism case, while all other solutions are legitimate solutions for the homomorphism case since the plaintext words are all among the 200 most frequent English words.

Crosswords puzzles were also created from the 200 most frequent English words. Among the ten 3×3 crossword puzzles found were:

t	*o*	*o*
o	*w*	*n*
o	*n*	*e*

w	*h*	*o*
h	*o*	*w*
o	*w*	*n*

w	*a*	*s*
a	*r*	*e*
s	*e*	*t*

4×4 crossword puzzles with black squares in the middle were also searched for among the 200 most frequent words. Some of the 64 solutions found are listed below.

t	*h*	*i*	*s*
h	■	*s*	*o*
i	*s*	■	*m*
s	*o*	*m*	*e*

s	*a*	*i*	*d*
a	■	*s*	*o*
i	*s*	■	*e*
d	*o*	*e*	*s*

a	*w*	*a*	*y*
w	■	*t*	*o*
a	*n*	■	*u*
y	*o*	*u*	*r*

r	*e*	*a*	*d*
e	■	*t*	*o*
a	*n*	■	*w*
d	*o*	*w*	*n*

7. Concluding remarks

This paper has presented an approach to constructing crossword puzzles and breaking substitution ciphers. The novelty in this approach is that simple constraints can be discovered among letters in the form of relational pattern. In general, NP-hard problems (such as those above) are solved by using search algorithms with exponential time complexity. Sometimes, however, instances of NP-hard problems arising from particular applications (as in these cases) satisfy special constraints that affect their complexity. Here, faster backtracking algorithms can be devised to take advantage of these constraints. A construction of algorithms sophisticated enough to exploit the special problem constraints requires skill and intuition. Using the relational patterns approach, special constraints can be extracted automatically, using an algorithm which does not require any insight of the problem. There is a secondary advantage in this approach. After finding constraints as relational patterns, the solution can be obtained by solving a system of equations in bivalent variables, a problem which has efficient heuristic solutions.

References

[1] Shvaytser, H. (1985). Constraints among events. TR-1490, Center for Automation Research, University of Maryland.

[2] Papoulis, A. (1984). *Probability, Random Variables, and Stochastic Processes.* Second edition, McGraw-Hill, New York.

[3] Wendroff, B. (1967). *Theoretical Numerical Analysis.* Academic Press, New York.

[4] Garey, M.R. and D.S. Johnson (1979). *Computers and intractability. A guide to the theory of NP-completeness.* Freeman, San Francisco.

[5] Hammer, P.L. and S. Rudeanu (1968). *Boolean Methods in Operations Research.* Springer, Berlin.

[6] Hansen, P. (1979). Methods of nonlinear 0-1 programming. *Ann. Math.* 5, 53-70.

[7] *Dell Pocket Crossword Puzzles* (1985). Dell Publishing, New York.

[8] Shannon, C.E. (1949). Communication theory for secrecy systems. *Bell Syst. Techn. J.* 28, 656-715.

[9] Peleg, S. and A. Rosenfeld (1979). Breaking substitution ciphers using a relaxation algorithm, *Comm. ACM* 22, (November 1979), 598-605.

[10] Carrol, J.B., P. Davis and B. Richman (1971). *Word Frequency Book.* American Heritage Publishing.