Solving Cryptograms with the Constrained Cyrpto-EM Algorithm

Andrew Gelfand

December 6, 2009

Final Project CS-271

1 Problem Description

A cryptogram is a type of word puzzle containing a sentence that has been encrypted using an arbitrarily transposed version of the standard alphabet. The goal of a cryptogram solver is to learn the mapping between the transposed alphabet and the standard alphabet, known as a cipher, and then use the cipher to decode the encrypted text. In most cryptograms, the cipher is assumed to be a permutation of the standard alphabet; meaning that each letter of the standard alphabet is encoded (or replaced) by a single, different letter in the alphabet. So, for example, the letter 'a' can be used to encode the letter 'z', but cannot also be used to encode the letter 'q' in the same cryptogram. As an example of the encryption process, consider the following cipher:

$standard \rightarrow encoding$	$standard \rightarrow encoding$	$standard \rightarrow encoding$
$a \rightarrow l$	$j \rightarrow a$	$s \rightarrow k$
$b \rightarrow i$	$k \rightarrow p$	$t \rightarrow o$
$c \rightarrow v$	$l \rightarrow r$	$u \rightarrow m$
$d \rightarrow x$	$m \rightarrow g$	$v \rightarrow b$
$e \rightarrow u$	$n \rightarrow e$	$w \rightarrow s$
$f \rightarrow j$	$o \rightarrow c$	$x \rightarrow q$
$g \rightarrow w$	$p \rightarrow h$	$y \rightarrow f$
$h \rightarrow y$	$q \rightarrow z$	$z \rightarrow t$
$i \rightarrow d$	$r \rightarrow n$	

Using this cipher, the standard sentence

I walked the dog

would be encrypted as

D slrpux oyu xcw

The challenge, given this encrypted text, is two-fold: 1) We must learn the cipher that best translates from the encrypted words to decoded, English words; and 2) We must actually decode the encrypted text.

Cryptograms are a popular form of word puzzle (there is even a society, called the American Cryptogram Association (ACA), devoted entirely to the hobby of creating and solving mono-alphabetic substitution ciphers!). Given their popularity, it should come as no surprise that many methods for solving cryptograms have been proposed. While early strategies were largely manual, over the years several researchers have presented computational approaches. An early algorithm by Peleg and Rosenfeld formulated the task of discovering the cipher as a graph (vertex) labeling problem, where the graph's vertices represented the encrypted letters in the cryptogram and the edges represented different trigrams [9]. To solve the labeling problem they used an iterative 'relaxation' algorithm in which the labels of each vertex were estimated using the label probabilities of neighboring vertices. 15 years later, Hart proposed a heuristic tree search method that leveraged both word frequency information and word patterns to effectively prune the search space [7]. Around the same time, Spillman et al. proposed a genetic algorithm for

solving simple permutation ciphers [8]. To the author's knowledge, this is the first time a stochastic search method was used to solve a simple substitution cryptogram. More recent ideas include a robust pattern matching or 'dictionary attack' method that also addresses the handling of non-dictionary words (e.g. names) and an evolutionary method that introduces letter-wise and word-wise constraints imposed by the cryptogram [10, 11].

This report presents a set of algorithms for solving cryptograms using the Expectation Maximization (EM) framework [2]. In a manner similar to [6], different constraints derived from the word puzzle are imposed on the posteriors of the latent variables. This has the effect of restricting the values taken by the latent variables to those that are consistent with the constraints of the cryptogram itself. Decoding the word puzzle is formulated as a weighted constraint satisfaction problem (CSP) and consistent solutions are found using a depth-first branch and bound algorithm.

The structure of this paper is as follows. Section 2 formulates solving a cryptogram as a hidden data problem. Section 3 provides background on EM and its application to the cryptogram problem. Section 4 provides a description of the constrained EM algorithm. Section 5 contains experimental results from the constrained EM cryptogram solver and Section 6 contains some concluding remarks.

2 Problem Formulation

The challenge of solving a cryptogram is formulated as a hidden data problem in the following manner. We observe an encoded word sequence $C = c_1, ..., c_n$ consisting of n independent and identically distributed words that were generated using a decoded word sequence $W = w_1, ..., w_n$ that is hidden from us. The dependencies between encoded and decoded words is modeled using an $R \times S$ matrix Θ (referred to herein as the cipher). Each element of the cipher describes the belief that an encoded letter l_c maps to a decoded letter l_w. - i.e. the $(r,s)^{th}$ entry $\theta_{rs} = p(l_c = r | l_w = s)$ describes the probability that the r^{th} letter in the encoded alphabet maps to the s^{th} letter in the decoded alphabet. Under this setup, the challenge of decoding the cryptogram C involves learning the most likely cipher Θ and then using the cipher to properly decode C. The issue of learning the most likely cipher will be addressed in this section. Discussion of the decoding challenge will be delayed until the experimental results section.

Out of all possible values of Θ we seek the value which is most probable given the observed encoded word sequence,

$$\begin{aligned} \hat{\Theta} &= \arg\max_{\Theta} p(C|\Theta) = \arg\max_{\Theta} \sum_{W \in \mathbf{W}} p(C, W|\Theta) \\ & s.t. \quad \sum_{s \in S} \theta_{rs} = 1 \text{ for } r \in R \end{aligned} \tag{1}$$

where the equality constraints ensure that each row of Θ is a valid discrete distribution.

To make the objective in (1) operational, we make a few simplifying assumptions. For a given $W \in \mathbf{W}$ (our dictionary), the objective can be written as $p(C, W|\Theta) = p(C|W, \Theta) \cdot p(W|\Theta)$. Assuming that the probability of word i is independent of surrounding encoded words and deciphered words, $p(C, W|\Theta)$ can be approximated as

$$p(C, W|\Theta) \approx \prod_{i=1}^{n} p(c_i|w_i, \Theta) \cdot p(w_i|\Theta) = \prod_{i=1}^{n} p(c_i|w_i, \Theta) \cdot p(w_i) \tag{2}$$

At this point, the likelihood (2) is expressed in terms of entire words (i.e. c_i's and w_i's), while the parameter Θ describes a letter-wise mapping. To connect Θ to the likelihood expression, we assume that the letters in a word occur independently of the neighboring letters in that word. This allows us to approximate the probability of the i^{th} word as

$$p(c_i|w_i, \Theta) \approx \prod_{j=1}^{len_i} p(c_{i_j}|w_{i_j}, \Theta) = \prod_{j=1}^{len_i} \theta_{c_{i_j} w_{i_j}} \tag{3}$$

where w_{i_j} is the j^{th} letter in the i^{th} decoded word and len_i is the number of letters in the i^{th} word. Combining (2)

with (3) yields the following likelihood expression

$$p(C, W|\Theta) \approx \prod_{i=1}^{n} \left\{ \prod_{j=1}^{len_i} \theta_{c_{i_j} w_{i_j}} \right\} \cdot p(w_i) \tag{4}$$

Several assumptions in this development turn out to be problematic when learning Θ for a particular C. In particular, independence across words is troubling because it does not enforce consistent assignments across words. For example, in the two-word cryptogram 'zla tel', the current expression does not prevent a different letter from being mapped to the 'l' in word 1 and 2. In addition, the letter-wise independence assumption implies that mappings within a word need not be consistent. So, in the encrypted word 'zlvz', the letter 'a' could map to the first 'z' while a different letter 'q' could map to the second 'z'. These issues are addressed in Section 4, where the distributions considered are restricted to those that are letter-wise and word-wise consistent.

3 Maximizing Θ using EM

As described in the previous section, our goal is to find the maximum likelihood estimate of Θ

$$\begin{aligned} \hat{\Theta} &= \arg\max_{\Theta} \ln p(C|\Theta) = \arg\max_{\Theta} \sum_{i=1}^{n} \ln p(c_i|\Theta) = \arg\max_{\Theta} \sum_{i=1}^{n} \ln \sum_{w_i} p(c_i, w_i|\Theta) \\ s.t. &\quad \sum_{s \in S} \theta_{rs} = 1 \text{ for } r \in R \end{aligned}$$

The Expectation-Maximization (EM) algorithm is an appropriate procedure for solving such a problem (see e.g. [2, 3, 4]). It maximizes Θ indirectly using an alternative lower bound $\mathcal{F}(\Theta)$

$$\begin{aligned} \mathcal{L}(\Theta) &= \sum_{i=1}^{n} \ln \sum_{w_i} p(c_i, w_i|\Theta) \\ &= \sum_{i=1}^{n} \ln \sum_{w_i} q_i(w_i) \frac{p(c_i, w_i|\Theta)}{q_i(w_i)} \\ &\geq \sum_{i=1}^{n} \sum_{w_i} q_i(w_i) \ln \frac{p(c_i, w_i|\Theta)}{q_i(w_i)} \equiv \mathcal{F}(q_1(w_1), ..., q_n(w_n), \Theta) \end{aligned}$$

where $q_i(w_i)$ is a non-negative function that sums to one over w for each c_i. Standard EM works iteratively on $\mathcal{F}(\Theta)$ via two steps:

$$\begin{aligned} \text{E} - \text{Step}: &\quad q_i^{(t)} \leftarrow \arg\max_{q_i} \mathcal{F}(q_1^{(t-1)}(w_1), ..., q_n^{(t-1)}(w_n), \Theta^{(t-1)}) \\ \text{M} - \text{Step}: &\quad \Theta^{(t)} \leftarrow \arg\max_{\Theta} \mathcal{F}(q_1^{(t)}(w_1), ..., q_n^{(t)}(w_n), \Theta^{(t-1)}) \end{aligned}$$

The lower bound $\mathcal{F}(\Theta)$ can be re-written as

$$\begin{aligned} \mathcal{F}(\Theta) &= \sum_{i=1}^{n} \sum_{w_i} q_i(w_i) \ln \frac{p(c_i, w_i|\Theta)}{q_i(w_i)} \\ &= \sum_{i=1}^{n} \sum_{w_i} q_i(w_i) \ln \frac{p(w_i|c_i, \Theta) p(c_i|\Theta)}{q_i(w_i)} \\ &= \sum_{i=1}^{n} \sum_{w_i} q_i(w_i) \ln p(c_i|\Theta) + \sum_{i=1}^{n} \sum_{w_i} q_i(w_i) \ln \frac{p(w_i|c_i, \Theta)}{q_i(w_i)} \\ &= \sum_{i=1}^{n} \ln p(c_i|\Theta) - \sum_{i=1}^{n} \sum_{w_i} q_i(w_i) \ln \frac{q_i(w_i)}{p(w_i|c_i, \Theta)} \end{aligned}$$

which means that maximizing $\mathcal{F}(\Theta)$ is equivalent to minimizing the KL Divergence between the approximating function $q_i(w_i)$ and the hidden posterior $p(w_i|c_i,\Theta)$. This distance is clearly minimized when $q_i(w_i) = p(w_i|c_i,\Theta)$ for all i.

The M-Step, involves finding the maximal value of Θ in $\mathcal{F}(q_1^{(t)}(w_1),...,q_n^{(t)}(w_n),\Theta^{(t-1)})$. This maximization is simplified by re-writing $\mathcal{F}$ as

$$\begin{aligned}
\mathcal{F}(q_1(w_1),...,q_n(w_n),\Theta) &= \sum_{i=1}^{n}\sum_{w_i} q_i(w_i)\ln\frac{p(c_i,w_i|\Theta)}{q_i(w_i)} \\
&= \sum_{i=1}^{n}\left\{\sum_{w_i} q_i(w_i)\ln p(c_i,w_i|\Theta) - \sum_{w_i} q_i(w_i)\ln q_i(w_i)\right\} \\
&= \sum_{i=1}^{n}\left\{\sum_{w_i} q_i(w_i)\ln p(c_i|w_i,\Theta) + \sum_{w_i} q_i(w_i)\ln p(w_i) - \sum_{w_i} q_i(w_i)\ln q_i(w_i)\right\} \qquad (5)
\end{aligned}$$

and noticing that the second and third terms in (5) do not depend on Θ. Thus, the M-step can be written as

$$\begin{aligned}
\Theta^{(t)} &= \arg\max_{\Theta}\sum_{i=1}^{n}\sum_{w_i} q_i^{(t)}(w_i)\ln p(c_i|w_i,\Theta) \qquad (6)\\
&= \arg\max_{\Theta}\sum_{i=1}^{n}\sum_{w_i} p(w_i|c_i,\Theta^{(t-1)})\ln\left\{\prod_{j=1}^{len_i}\theta_{c_{i_j}w_{i_j}}\right\} \qquad (7)\\
&= \arg\max_{\Theta}\sum_{i=1}^{n}\sum_{w_i} p(w_i|c_i,\Theta^{(t-1)})\left\{\sum_{j=1}^{len_i}\ln\theta_{c_{i_j}w_{i_j}}\right\} \qquad (8)\\
&s.t.\quad \sum_{s\in S}\theta_{rs} = 1 \text{ for } r\in R \qquad (9)
\end{aligned}$$

The objective in (6) is a concave function of the matrix Θ, subject to the normalization constraints in (9). We can cast the optimization problem in terms of a new function $f(\theta_{11},\theta_{12},...,\theta_{1|S|},\theta_{21},...,\theta_{|R||S|})$ of the elements within Θ, where $|R|$ and $|S|$ represent the number of rows and columns in Θ, respectively. The normalization constraints can be enforced by introducing Lagrange multipliers and forming a new functional (see e.g. [1])

$$L(\theta_{11},\theta_{12},...,\theta_{|R||S|},\lambda_1,...,\lambda_{|R|}) = f(\theta_{11},\theta_{12},...,\theta_{|R||S|}) + \lambda_1 g_1 + \cdots + \lambda_{|R|}g_{|R|} \qquad (10)$$

where $g_r = \theta_{r1},\theta_{r2},...,\theta_{r|S|} - 1 = 0$ ensure that each row of the cipher sums to 1 and λ_r is the Lagrange multiplier associated with the r^{th} equality constraint. To find the maxima of $L(\cdot)$, we take the derivative with respect to each of the θ_{rs}'s and set them equal to zero

$$\frac{\partial f}{\partial\theta_{rs}} + \lambda_1\frac{\partial g_1}{\partial\theta_{rs}} + \cdots + \lambda_{|R|}\frac{\partial g_{|R|}}{\partial\theta_{rs}} = 0 \qquad (11)$$

Keeping in mind that $g_r = 0$ for each row, we now have a system with $|R|\times|S| + |R|$ equations and $|R|\times|S| + |R|$ unknowns. The partial $\frac{\partial f}{\partial\theta_{rs}}$ is

$$\begin{aligned}
\frac{\partial f}{\partial\theta_{rs}} &= \frac{\partial}{\partial\theta_{rs}}\sum_{i=1}^{n}\sum_{w_i} p(w_i|c_i,\Theta^{(t-1)})\left\{\sum_{j=1}^{len_i}\ln\theta_{c_{i_j}w_{i_j}}\right\} \\
&= \sum_{i=1}^{n}\sum_{w_i} p(w_i|c_i,\Theta^{(t-1)})\left\{\sum_{j=1}^{len_i}\frac{\partial}{\partial\theta_{rs}}\ln\theta_{c_{i_j}w_{i_j}}\right\} \qquad (12)
\end{aligned}$$

To simplify (12), we note that the partial $\frac{\partial}{\partial\theta_{rs}}\ln\theta_{c_{i_j}w_{i_j}}$ is:

- 0 if letter $c_{i_j}\neq r$ or letter $w_{i_j}\neq s$ (i.e. if we are considering variables other than θ_{rs} in Θ); and

- $1/\theta_{rs}$ if letter $c_{i_j} = r$ and letter $w_{i_j} = s$

Using an indicator function

$$\mathrm{I}\left[c_{i_j} = r, w_{i_j} = s\right] \quad = \quad \begin{cases} 1 & \text{if index of j}^{\text{th}} \text{ letter of } c_i = r \text{ and index of j}^{\text{th}} \text{ letter of } w_i = s \\ 0 & \text{otherwise} \end{cases}$$

equation (12) can be rewritten as

$$\frac{\partial f}{\partial \theta_{rs}} \quad = \quad \sum_{i=1}^{n} \sum_{w_i} \left\{ \sum_{j=1}^{len_i} \frac{1}{\theta_{rs}} \cdot \mathrm{I}\left[c_{i_j} = r, w_{i_j} = s\right] \right\} p(w_i | c_i, \Theta^{(t-1)}) \tag{13}$$

In a similar fashion, the partial $\frac{\partial g_t}{\partial \theta_{rs}}$ is non-zero only if $t = r$. In such cases, the partial is

$$\frac{\partial g_r}{\partial \theta_{rs}} \quad = \quad \frac{\partial}{\partial \theta_{rs}} \left(p_{r1} + p_{r2} + \cdots + p_{r|S|} - 1\right) = 1$$

Substituting this and (13) into (11) gives:

$$\sum_{i=1}^{n} \sum_{w_i} \left\{ \sum_{j=1}^{len_i} \frac{1}{\theta_{rs}} \cdot \mathrm{I}\left[c_{i_j} = r, w_{i_j} = s\right] \right\} p(w_i | c_i, \Theta^{(t-1)}) + \lambda_r \quad = \quad 0$$

which we must solve for θ_{rs}. Since the constant λ_r exists solely to enforce the constraint $\sum_{s \in S} \theta_{rs} = 1$, we can set λ_r to a constant of our choice and normalize the θ_{rs}'s after updating (i.e. $\theta_{rs} \leftarrow \theta_{rs} / \sum_{s \in S} \theta_{rs}$). Setting $\lambda_r = -1$ and solving for θ_{rs} gives the following update rule

$$\theta_{rs}^{(t)} \quad \propto \quad \sum_{i=1}^{n} \sum_{w_i} \left\{ \sum_{j=1}^{len_i} \mathrm{I}\left[c_{i_j} = r, w_{i_j} = s\right] \right\} p(w_i | c_i, \Theta^{(t-1)}) \tag{14}$$

In other words, the probability that the r^{th} letter in the encoded alphabet maps to the s^{th} letter in the decoded alphabet is updated by an amount proportional to the occurrence of that mapping in our dictionary of words, weighted by the likelihood of each word under the current parametrization. The general crypto-EM algorithm is given in Algorithm 1 below.

Algorithm 1 Crypto-EM

Initialization:

- Compute prior over dictionary words $p(w_i)$ for $w_i \in \mathrm{W}$
- Initialize $\Theta^{(0)}$ with a uniform distribution (i.e. $\theta_{rs} = 1/|S|$)
- Set $t = 0$

Loop:

- $t \leftarrow t + 1$
- Update $\theta_{rs}^{(t)}$ using equation 14
- Normalize $\mathbf{\Theta}^{(t)}$ by computing $\theta_{rs}^{(t)} \leftarrow \theta_{rs}^{(t)} / \sum_{s \in S} \theta_{rs}^{(t)}$ for each element in Θ

Until converged$(\mathbf{\Theta}^{(t)}, \mathbf{\Theta}^{(t-1)})$ or $t >$ max iterations

4 Constrained EM

Up to this point, we have not addressed how to compute $p(w_i|c_i, \Theta^{(t-1)})$. Typically, this would be computed using Bayes rule as

$$\begin{aligned} p(w_i|c_i, \Theta^{(t-1)}) &= \frac{p(c_i|w_i, \Theta^{(t-1)}) \cdot p(w_i|\Theta^{(t-1)})}{p(c_i|\Theta^{(t-1)})} \\ &= \frac{p(c_i|w_i, \Theta^{(t-1)}) \cdot p(w_i)}{\sum_{w_j} p(c_i|w_j, \Theta^{(t-1)}) \cdot p(w_j)} \end{aligned}$$

where the summation is over all w_j in our dictionary. However, as was alluded to in Section 2, the independence assumptions implicit in the expression for $p(c_i|w_i, \Theta)$ can lead to the learning of an cipher that is inconsistent with the constraints imposed by the cryptogram. To address this shortcoming, we impose constraints on the posteriors and, as in [5, 6], we express our desired constraints as the requirement that $p(w_i|c_i, \Theta) \in Q(C)$. In particular, we consider two different sets Q; one that enforces letter-wise constraints within encrypted words and a second that also enforces constraints across words. To better describe how $Q(C)$ is identified, we formulate a constraint satisfaction problem (CSP) over the letters in C. For a thorough review of CSPs (and constraint processing in general) see [12].

Let $X = x_1, x_2 ... x_m$ be the sequence of m encrypted letters in the n-word cryptogram $C = c_1, c_2, ..., c_n$. Each x_i has domain $D_i = \{a, b, ..., z\}$ as it can be assigned to any decoded letter in the alphabet. Let R be a relation on some subset of letters $S \subseteq X$ and let T be a constraint defined on R. If $S_k = \{x_{k_1}, ..., x_{k_r}\}$ then R_k is the Cartesian product $D_{k_1} \times \cdots \times D_{k_r}$ and T_k is some subset of R_k. For example, consider the following text:

Z	H	B	Z		O	B	C	E		C	B	O
x_1	x_2	x_3	x_4		x_5	x_6	x_7	x_8		x_9	x_{10}	x_{11}

$$\begin{aligned} T_1 : R_1 = R_{\{1,2,3,4\}} &= \{\{a,l,g,a\}, \{a,l,m,a\}, \{a,q,u,q\}, ...\} \\ T_2 : R_2 = R_{\{5,6,7,8\}} &= \{\{a,i,m,s\}, \{a,i,r,s\}, \{a,p,e,x\}, ...\} \\ T_3 : R_3 = R_{\{9,10,11\}} &= \{\{a,g,e\}, \{a,i,d\}, \{a,i,l\}, ...\} \\ T_4 : R_4 = R_{\{3,6,10\}} &= \{\{a,a,a\}, \{b,b,b\}, \{c,c,c\}, ...\} \\ T_5 : R_5 = R_{\{5,11\}} &= \{\{a,a\}, \{b,b\}, \{c,c\}, ...\} \\ T_6 : R_6 = R_{\{7,9\}} &= \{\{a,a\}, \{b,b\}, \{c,c\}, ...\} \\ T_7 : R_7 = R_{\{1,2,3,5,7,8\}} &= \text{all different} \end{aligned}$$

The first relation R_1 is over the letters in the first word 'ZHBZ'. The domain of R_1 includes all 26^4 permutations of the letters a,b,...,z. The constraint T_1 enforces that: 1) Assignments to the letters $x_1x_2x_3x_4$ result in a valid English word; 2) The first and fourth letters are the same - i.e. $x_1 = x_4$; and 3) Letters 2 and 3 are different from letters 1 and 4 - i.e. $x_1 \neq x_2 \neq x_3$. Similar constraints are imposed on the second and third words via T_2 and T_3, respectively. The fourth constraint T_4 imposes a restriction across the first, second and third word. In particular, is enforces the fact that in any valid assignment $x_3 = x_6 = x_{10}$.

The goal in a CSP is to find a consistent solution - i.e. an assignment that satisfies all of the constraints of the problem. The primal and dual constraint graphs for the sample CSP are given in Figure 1. In the primal graph, nodes represent variables (letters) and the arcs connect variables included in a constraint. The absence of an arc means there is no direct constraint. The dual graph represents each constraints scope by a node and connects nodes (constraints) with overlapping variables.

Using the previous CSP formulation, we can define the restricted distribution space $Q(C)$ to include only the posteriors of words that are not inconsistent with the constraints in the CSP. In the experiments that follow, we consider two types of consistency: 1) Node Consistency - $Q_{NC}(C)$; and 2) Arc Consistency - $Q_{AC}(C)$. In the node consistent case, $p(w_i|c_i, \Theta) \in Q_{NC}(C)$ is restricted to the distribution over English words $w_i \in$ W that are node consistent with the constraints of the CSP of cryptogram C. Essentially, this means reducing the set of dicitonary words used to compute $p(c_i|\Theta)$ from a summation over the full dictionary to a summation over those words that

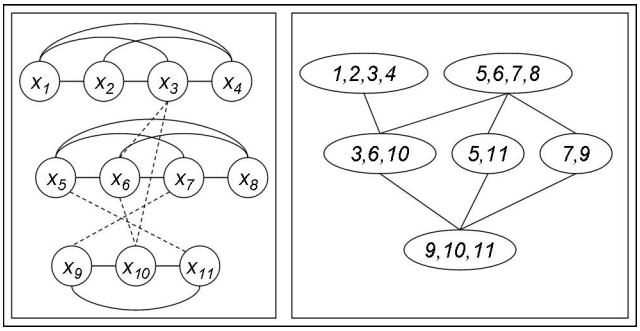

Figure 1: The Primal (left) and Dual (right) constraint graphs for sample CSP. For clarity, the 'all different' constraint has been omitted.

have a consistent letter-wise pattern with c_i, the i^{th} encrypted word. In the arc-consistent case, $Q_{AC}(C)$ is further restricted so that the domain of each letter contains contain only words that are pairwise consistent among the constraints of the CSP. This means removing from consideration words in the domain of c_i for which no assignment to the other words in the cryptogram can be found.

5 Results

Evaluation of the constrained crypto-EM algorithm was performed on randomly generated puzzles. The random puzzles consisted of 4,5,6,7 and 9 words and were generated using the Eppstein dictionary (see [13]). To eliminate the possibility of selecting words with only a few feasible mappings, candidate puzzle words were selected from 3 to 6 letter words, which account for more than half of the dictionary entries. A set of 100 puzzles of length 4,5,6,7 and 9 were generated and three different variants of the Crypto-EM algorithm were used to solve each puzzle - namely, the unconstrained crypto-EM algorithm, the node-consistent constrained algorithm (i.e. $p(w_i|c_i,\Theta) \in Q_{NC}(C)$) and the arc-consistent constrained algorithm (i.e. $p(w_i|c_i,\Theta) \in Q_{AC}(C)$).

After learning the most likely cipher, the puzzles were decoded using a Depth-First Branch and Bound (DFBB) procedure. Decoding is formulated as a search problem by constructing a tree of depth n (# words in cryptogram) where each branch includes feasible encrypted-word-to-English-word assignments. For example, consider a simple two-word cryptogram 'zhbz cbr'. The first level in the tree would correspond to all feasible assignments of English words to 'zhbz', while the second level would consist of assignments to 'cbr' consistent with the assignment at level 1. In other words, if 'that' was assigned to 'zhbz' at level 1, the domain of assignments to 'cbr' would consist of only three-letter words with middle letter 'a'. At each level of the tree a score (likelihood) is computed for the assignment of w_i to c_i as $score_{w_i \to c_i} = \prod_{j=1}^{len_i} \hat{\theta}_{c_{ij} w_{ij}} \cdot p(w_i)$. Bounding is used to prevent exploration of branches with maximum likelihood less than the best known solution. On each puzzle, the DFBB procedure was used to determine the k most likely assignments, where k was set to 15.

The performance of the algorithms was evaluated using two different metrics. Since the DFBB procedure returned the k best solutions, the first metric used was the Mean Reciprocal Rank (MRR). The reciprocal rank is the multiplicative inverse of the rank of the correct answer returned by the decoding procedure. So, for example if the solution to a puzzle was 'that car' and the algorithm returned the following solutions (ranked from most likely to least likely): 1) 'that cab'; 2) 'that can'; 3) 'that car'... the RR would be $\frac{1}{3}$, since the correct solution was returned as third most likely. The mean RR is found by averaging across the RR's of all puzzles (queries). The second metric used was the percentage of correctly assigned letters in the first solution returned by DFBB. Using the previous example, the '% correct' would be $\frac{4}{5}$ since there are 5 unique letters in the puzzle 'that car' and the first solution 'that cab' only incorrectly assigned one letter - namely, 'b' to 'r'.

Results for each of the algorithms are summarized in the following table and in Figures 3, 4 and 5. Several interesting points can be taken from this analysis. In general, and as would have been expected, the arc-consistent algorithm out-performed the node consistent algorithm which in turn out-performed the unconstrained algorithm. While the improvement is slight, this result seems to support the idea that there is benefit to learning the cipher under constraints and relaxing both inter-word and intra-word independence assumptions. It also interesting to note that all three algorithms improve steadily as the number of words in the puzzle increases. While increasing puzzle

	Arc Consistent				
# Words	Avg. MRR	Var. MRR	Avg. % Correct	Var. % Correct	Uniqe Chars
4	0.32203	0.13591	0.60728	0.11781	13.13750
5	0.49358	0.17941	0.79628	0.07120	14.91765
6	0.53454	0.16575	0.89077	0.02498	16.28571
7	0.55973	0.15175	0.92816	0.00615	17.09000
9	0.80154	0.09659	0.97376	0.00205	18.60563
	Node Consistent				
# Words	Avg. MRR	Var. MRR	Avg. % Correct	Var. % Correct	Uniqe Chars
4	0.32243	0.13776	0.61289	0.12392	13.13750
5	0.46096	0.18960	0.78522	0.06760	14.91765
6	0.53825	0.17019	0.88957	0.02369	16.28571
7	0.55562	0.15334	0.91669	0.01074	17.09000
9	0.78562	0.10178	0.96970	0.00296	18.39216
	No Consistency				
# Words	Avg. MRR	Var. MRR	Avg. % Correct	Var. % Correct	Uniqe Chars
4	0.32731	0.13721	0.58981	0.12937	13.13750
5	0.46759	0.18449	0.78283	0.07387	14.91765
6	0.53089	0.16861	0.88291	0.02715	16.28571
7	0.56022	0.15021	0.91664	0.01101	17.09000
9	0.79945	0.09782	0.97132	0.00308	18.57377

Figure 2: Summary of performance of the 3 crypto-EM algorithms. The means and variances of MRR and % Correct are given for several puzzle lengths.

length leads to an increase in the number of unique characters in the puzzle, the information gained by increasing the probability of observing repeated letters far outweighs the cost of mapping additional letters. This also suggests that the smaller the puzzle, the less benefit there is in enforcing arc-consistency during learning. Along these lines as shown in Figure 4, it is interesting to note the drastic decrease in MRR as puzzle complexity increases (puzzle size is computed as the log of the total number of nodes in the search problem faced by the DFBB procedure, which is computed as the Cartesian product of the domain of each $c_i \in C$).

6 Conclusion

In this report, the problem of solving simple substitution ciphers was formulated as a hidden data problem and an EM-based algorithm was presented to find the maximum likelihood estimate of the substitution cipher. Since some of the independence assumptions in the original formulation were detrimental to learning 'correct' ciphers, the EM algorithm was augmented to include constraints on the posteriors used in updating. By formulating an auxiliary CSP on the letters in the cryptogram, the posteriors were restricted to a constraint set that that was either node-consistent or arc-consistent with the CSP formulation.

Overall the Crypto-EM algorithm(s) performed quite well, returning the correctly solved puzzle on average within the 4th most likely response for three-word puzzles. Performance steadily improved as puzzle length increased and puzzle size (difficulty) decreased. In addition, the analysis in Section 5 demonstrated that constraining the E-step of EM yielded a modest improvement in performance. Further experimentation and analysis is needed to better evaluate the benefit of imposing node and arc consistency constraints on this problem. Ultimately, the idea of constraining the EM algorithm to find more meaningful (useful) solutions is very interesting and has potential application to other types of problems in AI, such as solving weighted CSPs.

References

[1] Boyd, S. & Vandenberghe, L. Convex Optimization. Cambridge University Press. 2004.

[2] Dempster, A.P., Laird, N.M. & Rubin, D.B. "Maximum Likelihood from Incomplete Data via the EM Algorithm," Journal of the Royal Statistical Society, Series B, 1997. Vol. 29. No. 1. pp:1-38.

[3] Bilmes, J. "A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models," International Computer Science Institute, Technical Report: TR-97-021, April 1998, Berkeley, CA.

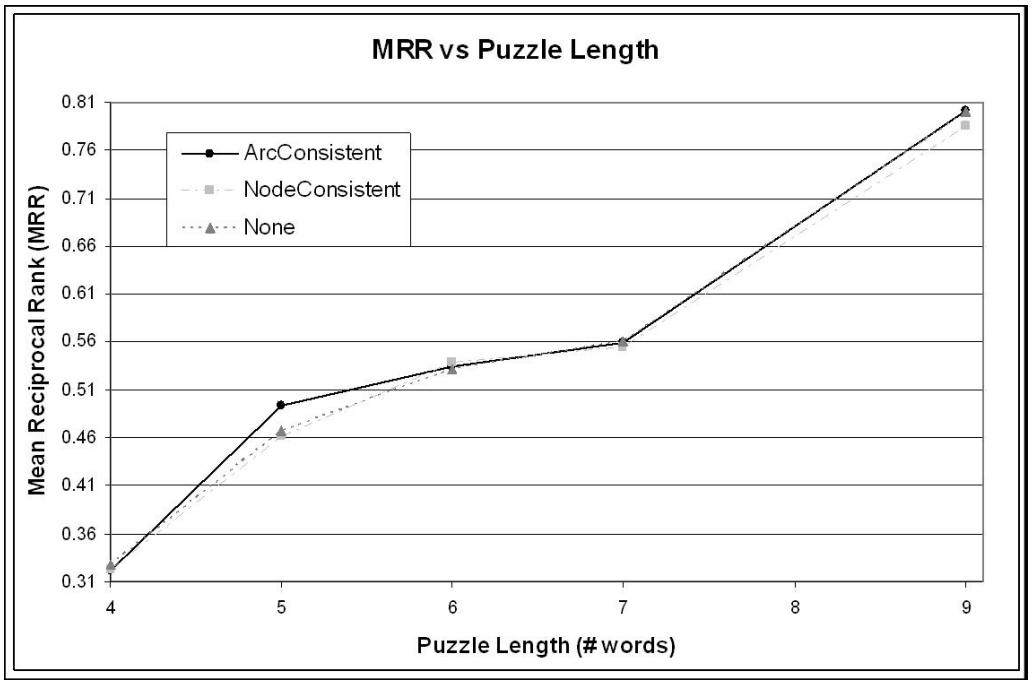

Figure 3: MRR versus Puzzle Length

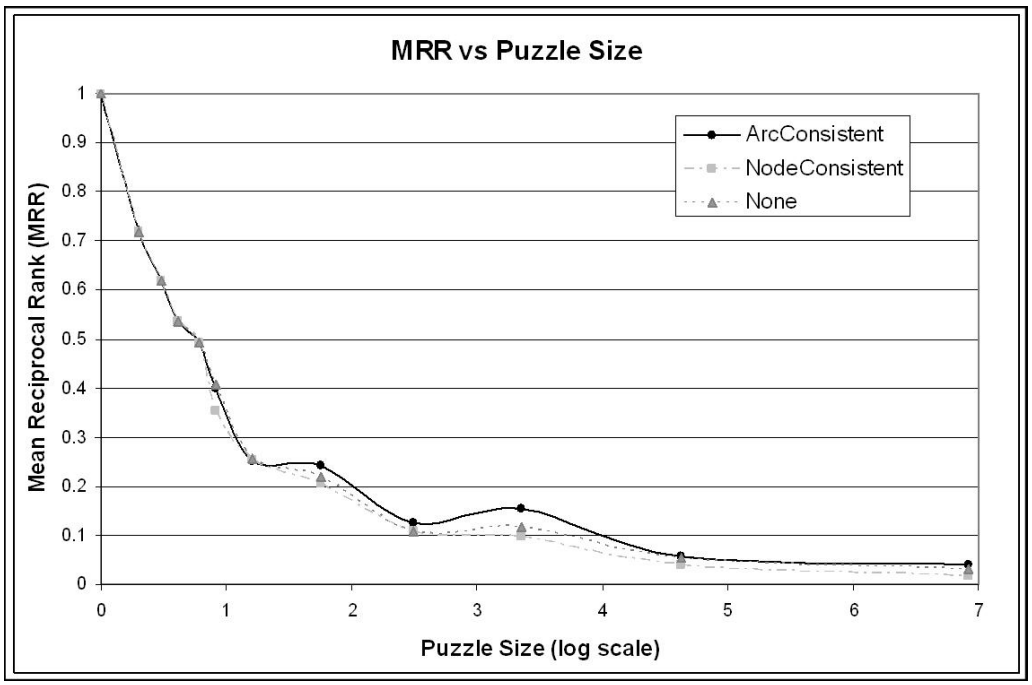

Figure 4: MRR versus Puzzle Size (domain size of unconstrained puzzle)

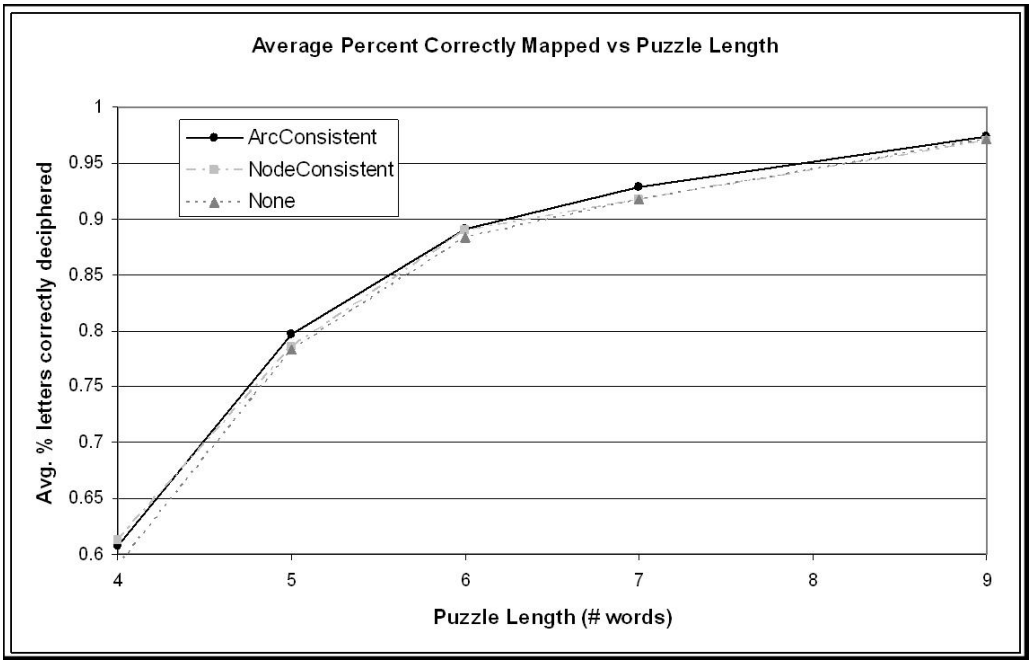

Figure 5: Average % Correctly Mapped versus Puzzle Length

[4] Beal, M.J. "Variational Algorithms for Approximate Bayesian Inference," Ph.D. Thesis, The Gatsby Computational Neuroscience Unit, University College London, 2003.

[5] Brown, P.F. et al. "The mathematic of statistical machine translation: Parameter estimation," Computational Linguistics, 19(2):263–311, 1994.

[6] Graca, J.V., Ganchev, K. & Taskar, B. "Expectation Maximization and Posterior Constraints," Advances in Neural Information Processing Systems (NIPS) 20, MIT Press, 2008.

[7] Hart, G.W., "To Decode Short Cryptograms," Communictions of the ACM, v.37 n.9, p.102-108, Sept. 1994.

[8] Spillman et al., "Use of a genetic algorithm in the cryptanalysis of simple substitution ciphers," Cryptologia, vol. 17 , Issue 1 (January 1993), pp:31-44.

[9] Peleg, S. & Rosenfeld, A. "Breaking substitution ciphers using a relaxation algorithm," Communications of the ACM, v.22 n.11, p.598-605, Nov. 1979.

[10] Olson, E. "Robust Dictionary Attack of Short Simple Substitution Ciphers," Cryptologia, v.31 n.4, p.332-342, October 2007.

[11] Oranchak, D. "Evolutionary algorithm for decryption of monoalphabetic homophonic substitution ciphers encoded as constraint satisfaction problems," Proc. of 10th Conf. on Genetic and Evolutionary Computation, July 12-16, 2008, Atlanta, GA.

[12] Decther, R. Constraint Processing. Morgan Kauffman, San Francisco, CA 2003.

[13] http://www.ics.uci.edu/~eppstein/cryptogram