Solving a Substitution Cipher

Locally informed proposals in Simulated Annealing

Alexander Wei

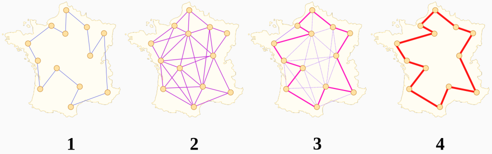
May 10, 2021

Combinatorial Optimization

Set of all solutions

$$\Omega = \{(a_1, a_2, \ldots, a_n)\}$$

Example: traveling salesman problem

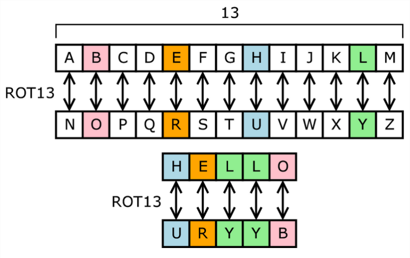


Find a solution ω that optimizes some function,

$$f(\omega) = \min\{f(\Omega)\}\,.$$

Substitution Cipher

Define a cipher C. For example we can "rotate" each letter by 13 places:



If $\Omega =$ all possible (English) substitution ciphers,

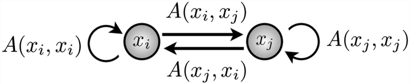
$$|\Omega| = 26!$$

Solution techniques

- Brute forcing all 26! possible keys

The exact algorithms like branch and bound, simplex method, brute force etc methodology is very inefficient for solving combinatorial problem because of their prohibitive complexity (time and memory requirement). The Evolutionary Computation algorithms are employed in an attempt to find an adequate solution to the problem. (Garg)

Evolutionary algorithms



Cipher Key Annealing

Begin with a best guess,

$$G : (a, b, c, \ldots, z) \mapsto (x_1, \ldots, x_{26})$$

Applying $G_0 = (a, b, c, d) \mapsto (h, e, l, o)$ to

$$"abccd"$$

Cipher Key Annealing

Begin with a best guess,

$$G : (a, b, c, \ldots, z) \mapsto (x_1, \ldots, x_{26})$$

Applying $G_0 = (a, b, c, d) \mapsto (h, e, l, o)$ to

$$"abccd" \mapsto "hello"$$

We got lucky here.

Cipher Key Annealing

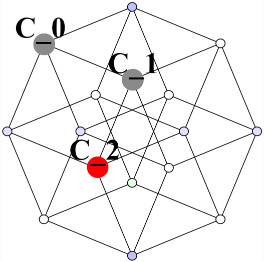
Begin with a best guess,

$$G : (a, b, c, \ldots, z) \mapsto (x_1, \ldots, x_{26})$$

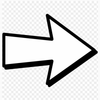
Applying $G_0 = (a, b, c, d) \mapsto (h, e, l, o)$ to

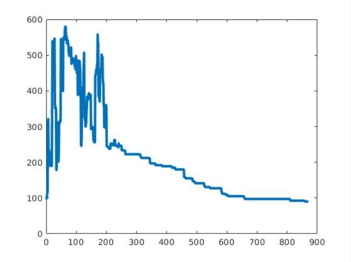
$$"abccd" \mapsto "hello"$$

We got lucky here. In most cases we need a way to evaluate a cipher-key's fitness.



increasing strictness





Going from key to key: how do we judge?

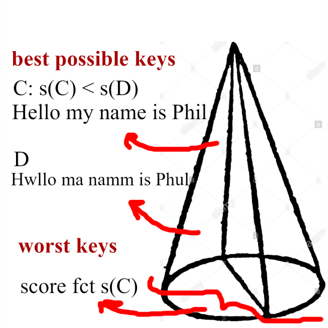
We will discuss how to propose new keys in a later slide.
For now consider Key Scoring.

"Hello, my name is Phil" is an English phrase.
What about "Hwllo, ma namm is Phul?"

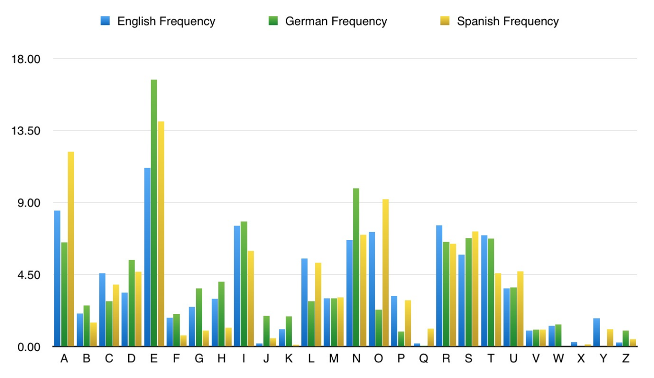
We need to make objective judgements on any text.

$$C(\text{"}djff\ldots\text{"}) = \text{Hello my name is Phil}$$

$$D(\text{"}djff\ldots\text{"}) = \text{Hwllo ma namw us Phul}$$



Key scoring via Frequency Analysis



For a score function s, (lower is better),

$$s(\text{"}hello\text{"}) < s(\text{"}xyzlo\text{"})$$

Efficient scoring

For a single-letter frequency score s_1,

$$s_1(\text{"}helo\text{"}) = s_1(\text{"}ehlo\text{"}) = s_1(\text{"}ehol\text{"}) = \cdots = s_1(S_4 \times \text{"}helo\text{"})$$

s_1 is vulnerable to a bottleneck:

$$C_1 \to \cdots \to C_k \to C_{k+1}$$

$$\text{"}xkgo\text{"}_{C_1} \to \cdots \text{"}ehlo\text{"}_{C_k} \to \{\text{"}helo\text{"}, \text{"}heol\text{"}, \ldots\}_{C_{k+1}}$$

We will benefit from a more efficient scoring method.

Forming more natural cipher guesses

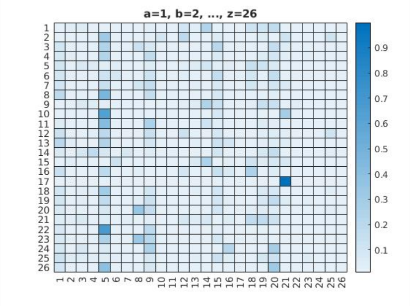
Interested in entire phrases, not just sets of letters

- We know that *e* is the most common letter in the language
- But then should *eeeee* be a fairly common phrase?

The letter-letter (sound-sound) flow of natural language induces a canonical score: is it readable?

Considering digram frequencies takes care of both troublespots we've seen:

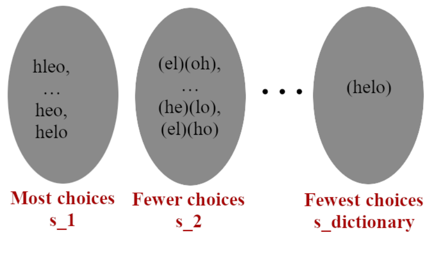
- $s_2(\text{"}helo\text{"}) < s_2(\text{"}eeee\text{"})$
- and the bottleneck $s_2(\text{"}helo\text{"}) < s_2(\text{"}ehlo\text{"})$



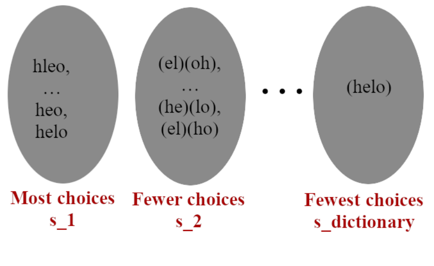
Digram decomposition

Decomposition into digrams (first order transition comparison)

$$\text{Hello my name is Phil} \rightarrow (h,e),(e,l),(l,o),\ldots,(h,i),(i,l)$$



Digram decomposition



Theorem

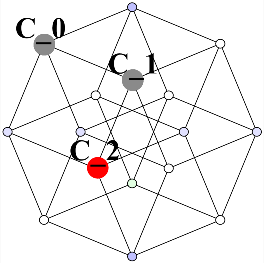
For natural language, analysis of higher order transition frequencies provides a more direct attack than lower ones. In particular,

$$\begin{aligned} &\#s_1\text{–deciphers observing single letter freq} \\ > &\#s_2\text{–deciphers observing digram freq} \\ > &\#s_{dict}\text{–deciphers observing word-dictionary frequencies}. \end{aligned}$$

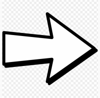
Putting it together

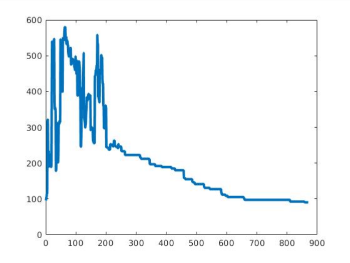
So to recap, annealing on cipherkeys can be done effectively by

- scoring digrams (consecutive letter pairs)
- moving from a cipher key to a (hopefully better) cipher key



increasing strictness





Cipher key proposals

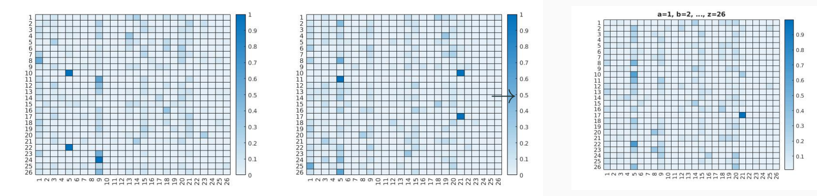
Let C be a proposed cipher key,

$$C = \{a, b, \ldots, z\} \to \{\sigma(a), \sigma(b), \ldots, \sigma(z)\}$$

Choose a pair of letters at random, ie $(a \to \sigma(a),\ b \to \sigma(b))$ and swap them, generating $C' = T(C)$

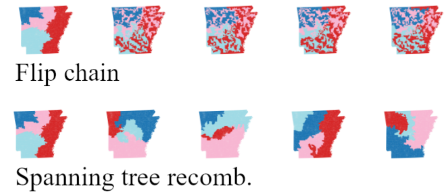
$$C' = \{a, b, \ldots, z\} \to \{\sigma^*(a), \sigma^*(b), \ldots, \sigma(z)\}$$

We compare the frequency scores for C, C'



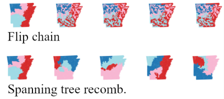
Do we update one letter assignment at a time, or can we do better?

Recall districting



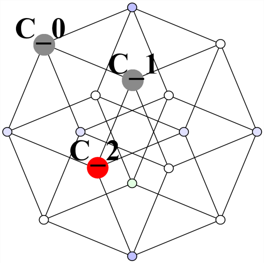
"ReCom samples preferentially from fairly compact districting plans [and] the tendency of the spanning tree process will be to produce districts without skinny necks or tentacles." (Deford, Duchin, et al.)

Recall districting

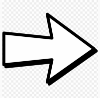


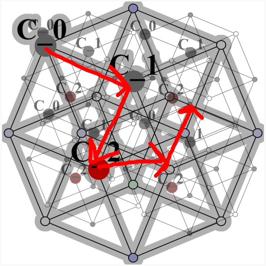
Similarly, define a new move over the key-flip. We can call it the "key-shuffle,"

$$C' = \{a, b, \ldots, z\} \to \{\sigma^*(a), \sigma^*(b), \ldots, \sigma^*(z)\}$$



fast mix





Metropolis Hastings Walk

Accept a new cipherkey proposal with probability

$$\lambda^{\tanh(\Delta \text{score})+1}, \quad 0 < \lambda \leq 1$$

where

$$\text{score} = \sum_{i \in I} \frac{(x_i - \mu_i)^2}{\mu_i} \sim \chi^2$$

for the frequencies $I = \{f(aa), f(ab), f(ac), \ldots, f(zz)\}$ and standard English μ_i.

Metropolis Hastings Walk

Accept a new cipherkey proposal with probability

$$\lambda^{\tanh(\Delta\text{score})+1}, \quad 0 < \lambda \leq 1$$

where

$$\text{score} = \sum_{i \in I} \frac{(x_i - \mu_i)^2}{\mu_i} \sim \chi^2$$

for the frequencies $I = \{f(aa), f(ab), f(ac), \ldots, f(zz)\}$ and standard English μ_i.

However:

> *Although [. . .] easy to implement, the new state y is proposed "blindly" (i.e. using no information about π) and this can lead to bad mixing and slow convergence. (Zanella)*

Metropolis Hastings Walk

Accept a new cipherkey proposal with probability

$$\lambda^{\tanh(\Delta\text{score})+1}, \quad 0 < \lambda \leq 1$$

where

$$\text{score} = \sum_{i \in I} \frac{(x_i - \mu_i)^2}{\mu_i} \sim \chi^2$$

for the frequencies $I = \{f(aa), f(ab), f(ac), \ldots, f(zz)\}$ and standard English μ_i.

However:

> *Although [...] easy to implement, the new state y is proposed "blindly" (i.e. using no information about π) and this can lead to bad mixing and slow convergence. (Zanella)*

$$C' = \{a, b, \ldots, z\} \rightarrow \{\sigma^*(a), \sigma^*(b), \ldots, \sigma^*(z)\}$$

Locally informed proposal for a biased MH walk

A simple way to circumvent the problem described above is to map discrete spaces to continuous ones and then apply informed schemes in the latter [... For an] alternative approach [...] informed proposals are obtained by introducing auxiliary variables and performing Gibbs Sampling in the augmented space. (Zanella)

Locally informed proposal for a biased MH walk

> *A simple way to circumvent the problem described above is to map discrete spaces to continuous ones and then apply informed schemes in the latter [. . . For an] alternative approach [. . .] informed proposals are obtained by introducing auxiliary variables and performing Gibbs Sampling in the augmented space. (Zanella)*

What we will do, instead, is bias the walker toward improved cipher keys.

- Restrict the number of letter-assignments to some $1 \leq k \leq 26$.
- Update the k worst letter-assignments in C to propose a new cipher key C'.

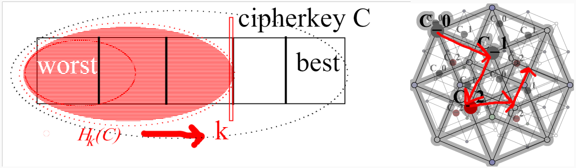
$$C' = \{a, b, \ldots, z\} \to \{\sigma^*(a), \sigma^*(b), \ldots, \sigma^*(z)\},$$

where

$$\mathbb{P}(\sigma^*(i) \neq i) \propto \mathrm{Keyrank}(i)$$

Proposal by Keyrank

- Restrict the number of re-assignments to some $1 \leq k \leq 26$.



- Update the k worst letter-assignments in C,

$$a(C, C') = \lambda^{\tanh(\Delta \text{score})+1}, \quad 0 < \lambda \leq 1$$

$$P(C, C') = a(C, C')\, \pi^*(C')\, \mathbb{1}(C' \in H_k(C))$$

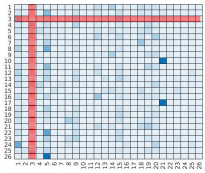
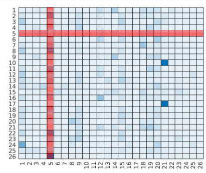
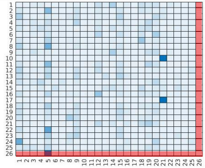
$$\doteq a(C, C')\, \pi^*(C')$$

where π^* is weighted by letter assignments scores (Grathwohl et al).

Computing Keyrank

$$P(C, C') = a(C, C')\, \pi^*(C')\, \mathbb{1}(C' \in H_k(C))$$
$$\doteq a(C, C')\, \pi^*(C')$$

Make a locally informed move given the ranking of letter assignments,



...


and compute $\pi^*(C')$ per letter assignments most likely to improve.

We will do this by comparison of row-column sums,

$$\sum \sqrt{\chi^2} \rightsquigarrow \pi^*(C')$$

(Transitions into and out of "a," transitions into and out of "b," ... transitions into and out of "z")

Demo

We will be informing key proposals on the basis of *A Treatise of Human Nature* by David Hume (available on Project Gutenberg).

> *Nothing is more usual and more natural for those, who pretend to discover anything new to the world in philosophy and the sciences, than to insinuate the praises of their own systems, by decrying all those, which have been advanced before them. ...*

Can we decipher something like

> *rcpkpkicgsbwhigbcqibilhiupkirwweuwggibulgiilumpwgiswxzctipkiuwdgplupiwj*
> *iucuecupczspchilzepkimnlmfiduieczulbleufdpuolgczxbmfisldu ...*
> *dbpcsiwjpkioblzpklufiizuiepwpgilpuribbczxu?*

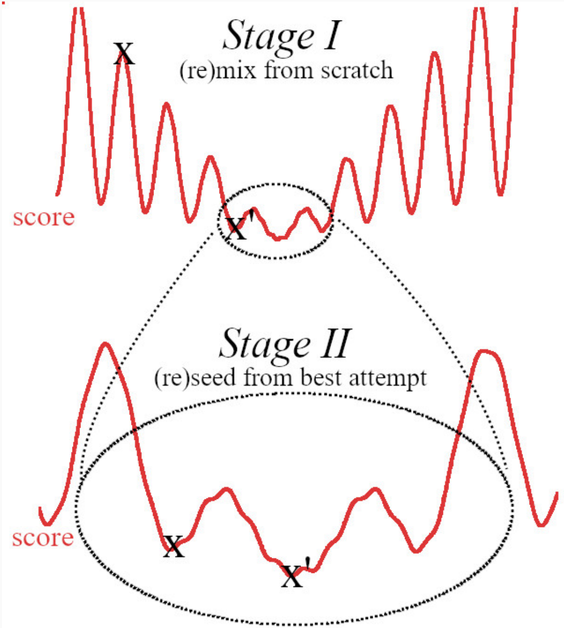
(spoiler)

Did we do it?

We should get

> *with their clover like leaves the wood sorrels are easy to recognize the sour taste of the leaves is distinctive and they may be used in salads but sparingly because of the oxalic acid content the genus name comes from the greek oxyssour this species is a cosmopolitan weed perhaps originally native to north america . . .*

Annealing stages



Github

Check out the code at
`github.com/alexander-wei/MCMC-Annealing-2021`.

```
properFREQ(properFREQ < .01) = .01;

% annealing with respect to the
% best pair (<Cipher>, score)

% stage I initialize best cipher tracking
if stage == 1
    G = cypher();
    bestC = G;
    bestS = freqAnal(S_,G);
end

while 1
% hard coded stopping points for demo purpose
if stage == 1 && bestS < 60, return; end

% schedule selection
if stage == 1
    % Stage I Schedule -- < transposition shuffle mixing time
    ITS_ = [170, repmat([1],1,200), 500]';
    LAM_ = [1, slowcoolings(1:200), .01]';
elseif stage == 2
    % Stage II reseed:
    G = bestC;

    % Stage II Schedule
    ITS_ = [500, 2000]'; %1000 -> 10000
    LAM_ = [bumpcooling(1:500),.01]';
end
```

Artwork

Thanks to the following for artwork used in these slides:

https://en.wikipedia.org/wiki/Travelling_salesman_problem#
/media/File:Aco_TSP.svg
https://en.wikipedia.org/wiki/ROT13#/media/File:
ROT13_table_with_example.svg
https://en.wikipedia.org/wiki/Hypercube_graph#/media/File:
Hypercubestar.svg
https://www.alamy.com/
the-plane-crosses-the-scraped-cone-through-the-vertex-the-cross
html
https://commons.wikimedia.org/wiki/File:
Letter-frequency_West-Germanic.png
https://mggg.org/uploads/ReCom.pdf
https://www.pinterest.com/pin/739223726331638373/
https://www.pnas.org/content/111/49/17408

Works Cited

Deford, Daryl, Duchin, Moon, et al. "Recombination: A family of Markov chains for redistricting." Mar. 27, 2020. MGGG Redistricting Lab. ¡mggg.org/uploads/ReCom.pdf¿.

Garg, Poonam. "Evolutionary Computation Algorithms for Cryptanalysis: A Study." IJCSIS International Journal of Computer Science and Information Security, Vol. 7, No. 1, 2010. Arxiv. ¡arxiv.org/pdf/1006.5745¿.

Grathwohl, Will, et al. "Oops I Took A Gradient: Scalable Sampling for Discrete Distributions." 2021. Arxiv. ¡arxiv.org/pdf/2102.04509¿.

Ireland, Michael, et al. "Monte-Carlo Imaging for Optical Interferometry." 2007. Arxiv. ¡arxiv.org/pdf/2007.00716¿.

Zanella, Giacomo. "Informed proposals for local MCMC in discrete spaces." Nov. 21, 2017. Arxiv. ¡arxiv.org/pdf/1711.07424.pdf¿.