A Dominant Gene Genetic Algorithm for a Substitution Cipher in Cryptography

Derrick Erickson and Michael Hausman

University of Colorado at Colorado Springs

CS 591

Substitution Cipher

1. Remove all but the letters in the original text (NO formatting, spaces, punctuation)
2. Create a character mapping for each letter

Cipher/Key	
Original letters:	`a,b,c,d,e,f,g,h,i,j,k,l,m,…,z`
Encrypted letters:	`q,z,y,m,h,j,b,x,o,a,f,i,p,…,t`

Example	
Original Message:	`iamasubstitutioncipher`
Encrypted Message:	`oqpqgvzglolvlodkyonxhw`

Overview of Genetic Algorithms

- Based on Darwin's Theory of Evolution
  - Take several solutions and use them to make "better" solutions over time
- Steps to a Genetic Algorithm

1. Start with a set of solutions (1st Generation)
2. Take original "parent" solutions and combine them with each other to create a new set of "child" solutions (Mating)
3. Somehow measure the solutions (Cost Function) and only keep the "better" half of the solutions
   - Some may be from "parent" set, others are from "child" set
4. Introduce some random changes in case solutions are "stuck" or are all the same (Mutation)
5. Repeat starting with Step 2

Cost Function (Initialization Table)

- Made Custom Gram Table Program
- Two Cost Function Tables
- Find top N grams from Bible
  - All Unigrams
  - Top N bigrams
  - Top N trigrams
  - Top N four-grams
- Scores proportional to occurrence
- Scores proportional to gram size
  - Score * 2 for bigram
  - Score * 3 for trigram
  - Score * 4 for four-gram

Table 1: Top 100 gram table

Gram	Score
e	1
t	0.686449195
o	0.648787116
…	…
th	200
he	185.2286915
an	88.3373248
…	…
the	300
and	112.6185938
you	73.82056059
…	…
will	400
nthe	389.8032989
…	…

Cost Function (Run Table)

- Also from Custom Gram Table Program
- Find top 10 grams for each letter
  - Top 10 bigrams with an a, b, etc
  - Top 10 trigrams with an a, b, etc
  - Top 10 four-grams with an a, b, etc
- Scores proportional to occurrence
- Scores proportional to gram size
  - Score * 2 for bigram
  - Score * 3 for trigram
  - Score * 4 for four-gram

Table 2: Top 10 gram per letter table

Gram	Score
an	88.3373248
ea	53.84633505
ha	49.59365553
at	44.90440108
…	…
but	14.07204436
bec	10.39076377
heb	9.59472339
llb	9.51349478
…	…
nthe	389.8032989
dthe	373.8908
thel	368.8100909
…	…

Initialization of First Generation

- The first set of solutions
- 5 ways to create a solution:
  1. Unigrams 10%
  2. Bi-grams 10%
  3. Tri-grams 10%
  4. Four-grams 10%
  5. Random 60%
- The solutions are built by ranking the unigrams, bi-grams, etc from the cipher text and matching them with the unigrams, bi-grams, etc in the initialization table
  - If top trigram is "xqz" then that represents "the"

Mating Selection

- Mating finds new solutions
  - Similar to solutions in current generation
  - Potentially closer to "real" solution
- Select solutions by the total "cost"
  - Look at all of the bigrams, trigrams, etc. in the cipher text and add the score of all the grams found in the run table
  - Higher scores represent a better solution
- Randomly mate two chromosomes from the top half of each generation (Elitism)
- Both parents and both children inserted into new generation
  - Keeps the best solution
  - "Children" should be better or just as good as "parents"

Genetic Algorithm Mating

1. Add up the number of occurrences of each gram in the run table for each letter

2. Find Dominant Genes

Parent 1:	q	e	t	v	p	r	…
Gene Cost:	11	12	3	24	15	4	

Select the upper 1/3

Dom. Genes: v, p, etc

Parent 2:	z	v	y	g	d	i	…
Gene Cost:	25	15	5	10	8	14	

Select the upper 2/3

Dom. Genes: z, v, i, g, etc

3. Place Dominant Genes Based on First Parent

Child 1:	*	*	*	v	p	*	…
Child 2:	z	v	*	g	*	i	…

4. Fill in Blanks from Second Parent

Child 1:	z	v	*	v	p	i	…
Child 2:	z	v	*	g	p	i	…

5. Fill in any Remaining Blanks from First Parent

Child 1:	z	v	t	v	p	i	…
Child 2:	z	v	y	g	p	i	…

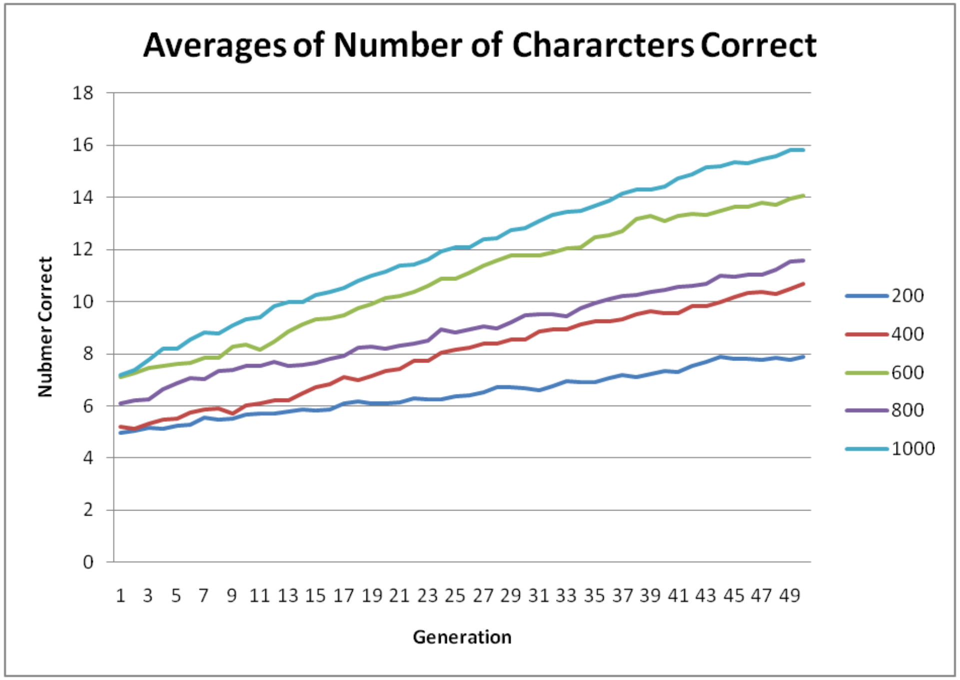
Mutation Selection

- Modify solutions
  - Keeps a generation from having the same solution
  - Potentially opens up new solutions not found through mating
- Mutate everything but the top solution
- The Mutation Randomly swaps two letters
  - Original: a b c d e f … z
  - Mutated: a b f d e c … z
  - Swap Positions: 0 0 1 0 0 1 … 0
- If the solution has a higher score than before the mutation it is kept!
- Otherwise a second mutation is applied

Results

Table 3: Number of Letters Correct

Cipher Text Letters	Average	Min	Max
200	7.9	3	16
400	10.68	5	19
600	14.06	5	19
800	11.58	5	20
1000	15.82	12	20

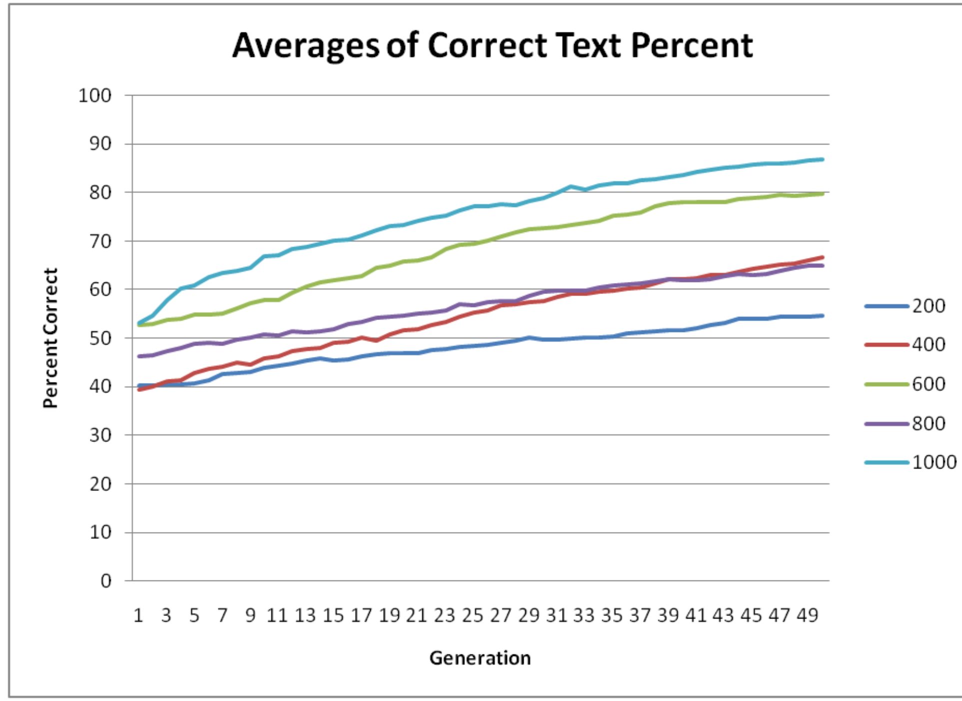


- Values are number of correct letters in key
- Over 50 iterations of 100 solutions over 50 generations
- In general, the more cipher text available, the better the results

Results Continued

Table 4: Percentage of cipher text correct

Cipher Text Letters	Average %	Min %	Max %
200	54.67	35	86
400	66.7	34.25	92.5
600	79.76	44.33	93.66
800	64.9	34.12	95.37
1000	86.77	60.4	94.3



- Percentage of text correct is not equivalent number of letters correct in key
- 15.82 letters correct in key is 86.77% of the output text on average
- Some letters appear more often
    - Better to get some common letters (e, t, h, a) than many uncommon ones (q, x, w, z)

Conclusion

- Dominant Gene Algorithm
  - Keeps best letters
  - Uses gram statistics to determine "better" solutions
  - Gets a high percentage of cipher text correct
- Works by
  - Using cost function on the gene level
  - Using dominant genes in mating
  - Improving recessive genes in mutation

Questions?

More Results