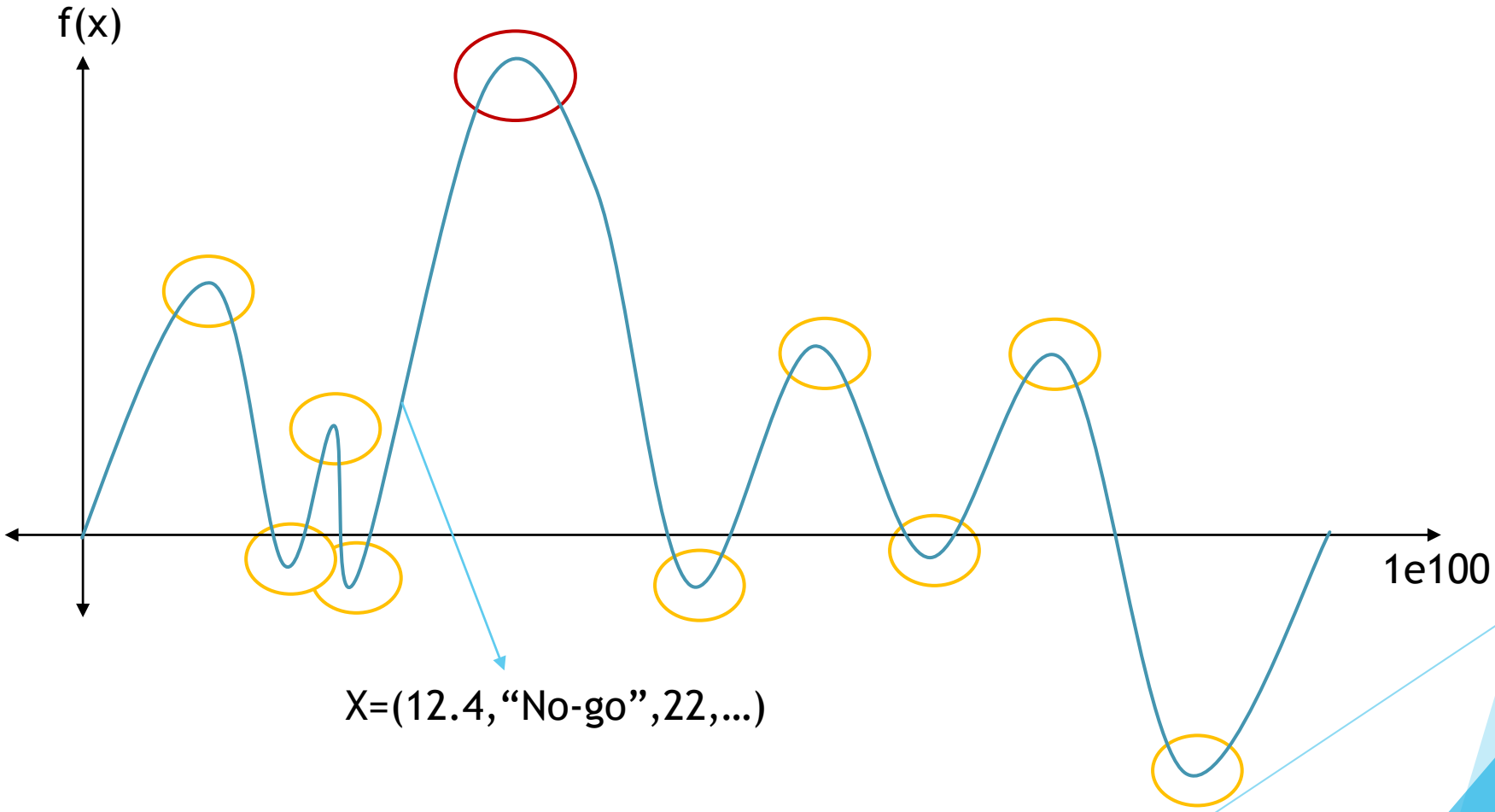
Graph Guided Genetic Algorithms

Kyle Sweeney

Understanding Genetic Algorithms
Part 1: the Problem



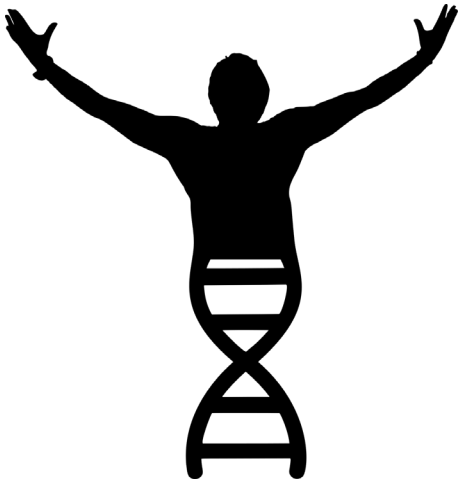
Understanding Genetic Algorithms
Part 2: Just Copy Nature

- Solution == DNA
  - e.g (12.4,“No-go”,22,…)
- Fitness function
  - A method for determining how “good” a solution is
  - Can be a score, where higher is better
- Breeding
  - Combine DNA in different ways
  - E.g (12.4, “No-go”,22,…) + (-3, “Go”, “9) == 2^X possible combinations
- Survival of the Fittest

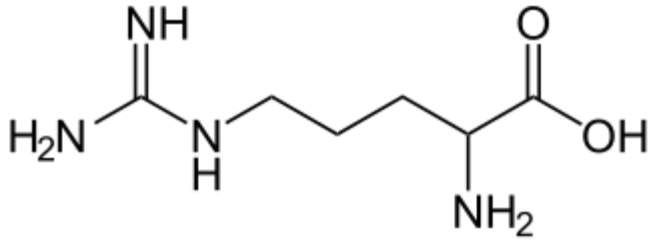
Application: Genetic Engineering

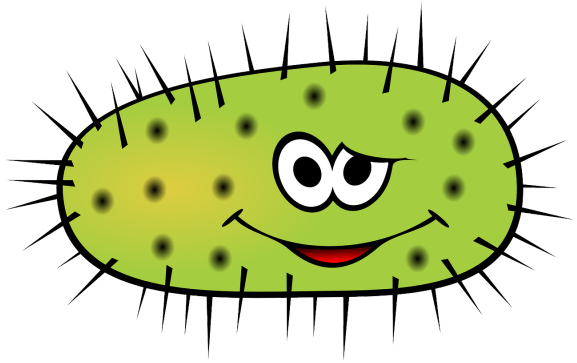
ACGT – building blocks of DNA	“Nucleotide”
Amino Acids – building blocks of proteins	20 in all: (Tryptophan, Valine, Glutamine….)
Group of 3 nucleotides == 1 Amino Acid	4^3 = 64 combinations There are redundancies + start and stop codons

Humans and Bacteria are Different



CGT: 0.123
CGC: 0.334
CGA: 0.462
CGG: 0.111





CGT: 0.121
CGC: 0.235
CGA: 0.461
CGG: 0.213

Solution: Genetic Algorithms to solve Genetic Engineering Problems

- "DNA": the specific Codon encodings which generate the same Protein
- Fitness Function: $\sum |\,\mathrm{MinMax}(\mathrm{human}) - \mathrm{MinMax}(\mathrm{SolutionInBacteria})|$
- Breeding:
  - Zip Children: for each position, alternate between taking from parents
  - Skip Children: Zip Children where zip_num > 2
  - Random Children: randomly choose from parents
  - Half and Half: first half one parent, second half the other
  - ….
- Take top 10 each generation

How Graphs Made things Different

- *Graph Based Evolutionary Algorithms* by Bryden K.M. et al
- Take a graph and place a potential solution on each vertex
  - The only mating partners for that vertex are its neighbors
  - Choose from potential mates who to mate with
- Only replace parent if child is better than parent

Pseudocode

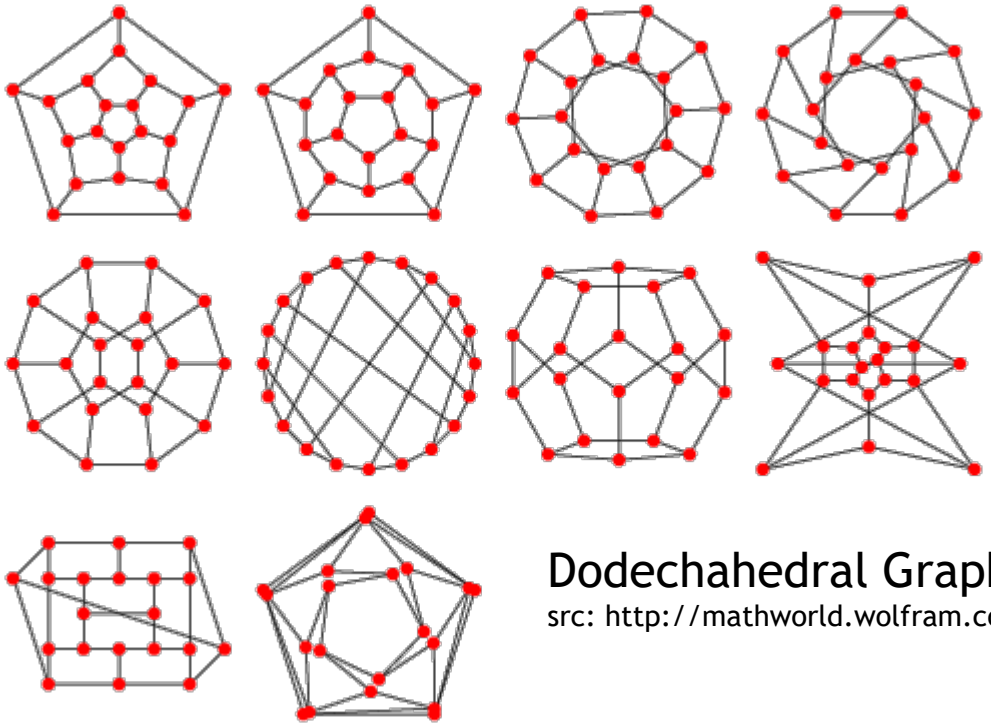
```
1   graph = new graph([ list of random permutations of start ])
2   for i in range 50:
3      for v in graph.nodes():
4         children = []
5         for n in graph.neighbors(v):
6            children += breed(n,v,10)
7         sort(children)
8         if children[0].score < v.score:
9            graph.replace(v,children[0])
10  return sort(graph.nodes())[0]
```

complexity: $O(V^2B)$ where $O(B)$ is time complexity of Breeding algorithm, in this case $O(N)$ where N is length of solutions.

Graphs – variable nodes

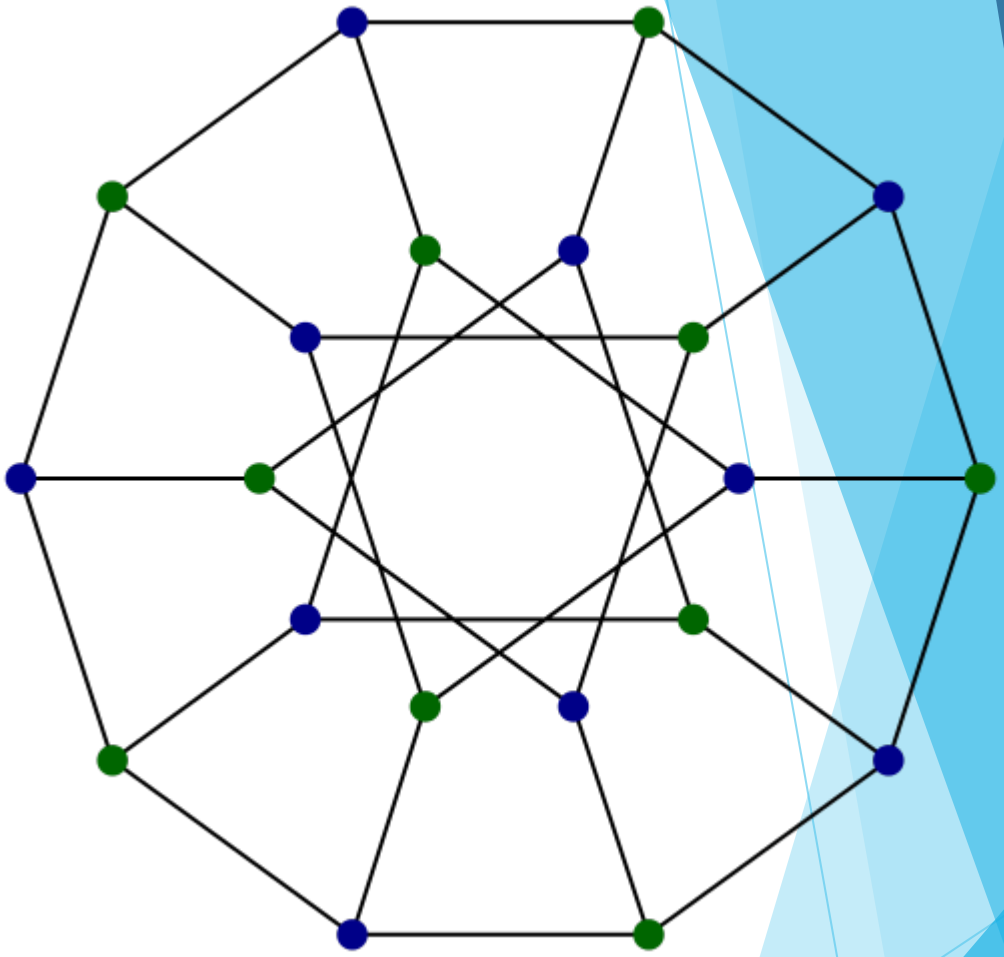
- Complete Graph
  - Every node is connected to each other
- 2D-Grid
  - Each node is stored in a 2D grid with avg 4 neighbors
- Caveman Graph
  - K connected Q cliques in a ring
- Windmill Graph
  - Q cliques with all nodes connected to a central node
- Erdos-Renyi aka GNP
  - For each possible edge between N nodes has a probability P of existing

Established Node Sizes



Dodechahedral Graph

src: http://mathworld.wolfram.com



Desargues Graph

https://upload.wikimedia.org/wikipedia/commons/thumb/2/2e/DesarguesGraph.svg/516px-DesarguesGraph.svg.png

Implementation Details

- Software Libraries
  - Python3 targeted
  - Graphs generated and manipulated via NetworkX
  - Pypy3 used to execute the program
- Graph Manipulation Technique
  - Single threaded, going from each node one after another
- Data Collection
  - 4 Specimens being compared against an e.coli strain
    - caenorhabditis elegans, Mus musculus, Homo sapien, Saccharomyces cerevisiae
  - 10 runs averaged in score and time elapsed

Results – 20 nodes - Variable

Fully Connected				
	Time		*Score*	
Genes	mean	stdev	mean	stdev
5	1106.064027	84.105426	19126.919	1283.620401
4	1025.578735	02.427339	24928.944	1606.458714
3	1020.771278	02.082315	21137.479	2176.747175
2	1041.69969	00.767798	23789.236	1482.189131

2D-Grid				
	Time		*Score*	
Genes	mean	stdev	mean	stdev
5	112.497632	0.603611	25786.073	2065.608277
4	108.761492	0.569377	28664.414	2154.667222
3	107.421994	0.119317	26082.867	2162.381269
2	110.420743	0.108324	26082.867	1815.084957

Windmill Graph - 4,5				
	Time		*Score*	
Genes	mean	stdev	mean	stdev
5	223.842559	1.001441	23436.453	2572.074629
4	215.994595	0.466179	28287.147	1813.922553
3	216.386268	1.20397	24182.217	1931.304768
2	224.740323	1.584885	27756.876	2479.184106

Caveman graph - 4,5				
	Time		*Score*	
Genes	mean	stdev	mean	stdev
5	223.976335	2.085542	25257.561	2122.087532
4	217.215015	0.891218	30374.471	1695.429033
3	220.781013	0.234819	26713.461	1856.30301
2	223.300958	0.234594	28866.525	1547.000103

Erdos-Renyi - 20,0.5				
	Time		*Score*	
Genes	mean	stdev	mean	stdev
5	525.188143	34.916041	19755.386	1891.456603
4	510.376475	34.632428	26097.224	1983.388064
3	525.785302	57.888515	21797.674	1586.296764
2	602.725836	68.400198	25190.05	1644.452654

Results – 20 Nodes - Steady

Desargues Graph				
	Time		*Score*	
Genes	mean	stdev	mean	stdev
5	169.624232	0.762033	22348.231	1677.688622
4	177.160388	0.500322	27590.251	2675.609935
3	164.078059	0.545794	24807.326	2002.273596
2	167.43908	0.325993	27069.238	2175.457151

Dodecahedral graph				
	Time		*Score*	
Genes	mean	stdev	mean	stdev
5	169.379436	00.768239	23218.362	1363.691871
4	163.400114	00.178623	26018.677	1663.768732
3	163.627567	00.509778	24207.807	1362.717518
2	176.147642	11.073559	26481.906	1667.529337

Results – 40 Nodes - Variable

Fully Connected - 40				
	Time		*Score*	
Genes	mean	stdev	mean	stdev
5	4407.949818	180.660595	17747.837	1073.674007
4	4166.038358	030.571147	22628.39	1405.021405
3	4203.88681	108.770135	19826.178	2073.063743
2	4295.030674	046.220033	20571.038	1319.340475

2D-Grid - 40				
	Time		*Score*	
Genes	mean	stdev	mean	stdev
5	444.243351	24.377109	21019.449	2326.198972
4	391.372798	33.135986	23924.596	1550.27023
3	373.068643	00.282972	23104.234	1864.526685
2	373.068643	00.593986	24259.694	1947.400487

Windmill Graph - 4,10				
	Time		*Score*	
Genes	mean	stdev	mean	stdev
5	0981.594376	6.547543	19369.346	2047.492682
4	0961.863631	2.460456	24557.848	2047.492682
3	0977.074179	0.66965	21314.179	2339.926058
2	1003.081456	4.429312	23416.692	2092.55662

Caveman graph - 4,10				
	Time		*Score*	
Genes	mean	stdev	mean	stdev
5	1164.003078	122.147287	20447.382	1357.424586
4	0997.866954	000.791199	25169.657	0849.044711
3	0984.32995	001.111364	21371.185	1739.21804
2	1010.327206	000.632495	24271.767	1496.809353

Erdos-Renyi - 40,0.1				
	Time		*Score*	
Genes	mean	stdev	mean	stdev
5	454.887323	48.710071	20182.799	1593.168119
4	421.002154	31.075691	25366.729	2271.80358
3	418.624505	44.980772	21653.777	1181.243829
2	432.977019	33.668121	24098.38	2721.854925

Results – Random Mate

Fully Connected - 40				
	Time		*Score*	
Genes	mean	stdev	mean	stdev
5	112.914445	0.692567	25491.094	2011.471445
4	109.614343	0.530169	27581.154	1917.922058
3	109.074456	0.139277	25906.26	2120.395358
2	111.604002	0.19365	28398.73	1873.491964

2D-Grid - 40				
	Time		*Score*	
Genes	mean	stdev	mean	stdev
5	117.153094	0.237125	25994.986	1969.151356
4	112.069839	0.209047	27425.349	1155.710915
3	113.049799	0.12099	26199.627	1720.679499
2	116.833984	0.141174	29231.478	2798.136781

Windmill Graph - 4,5				
	Time		*Score*	
Genes	mean	stdev	mean	stdev
5	105.765696	0.399097	25199.198	2081.407954
4	104.575735	0.573102	27805.829	1347.823895
3	105.645179	0.501267	25258.195	1135.440949
2	108.486528	0.629473	29777.532	2081.029823

Caveman graph - 4,10				
	Time		*Score*	
Genes	mean	stdev	mean	stdev
5	114.436177	0.55969	25539.219	1705.047569
4	111.138927	1.669333	29753.6	2519.051726
3	108.949419	0.258024	26960.741	1125.010804
2	112.650083	0.096759	28374.057	1614.289676

Erdos-Renyi - 40,0.1				
	Time		*Score*	
Genes	mean	stdev	mean	stdev
5	110.97712	2.371762	27572.692	1414.054499
4	107.17137	2.95958	28022.321	1545.729126
3	109.823711	2.074018	26623.335	2014.636918
2	112.247242	2.680797	28052.558	2186.395163

Generalized Results

- Diminishing Returns Relationship between Time and Quality of Score
- Rather large Variability in solutions
- Graphs overall impact runtime by limiting number of possible breeding pairs
- Limiting to 1 mate, graphs have no impact

Future Direction

- Multi-thread the process by taking a vertex-program approach
- Create a separate thread/process for each vertex, combine results at end of each generation round