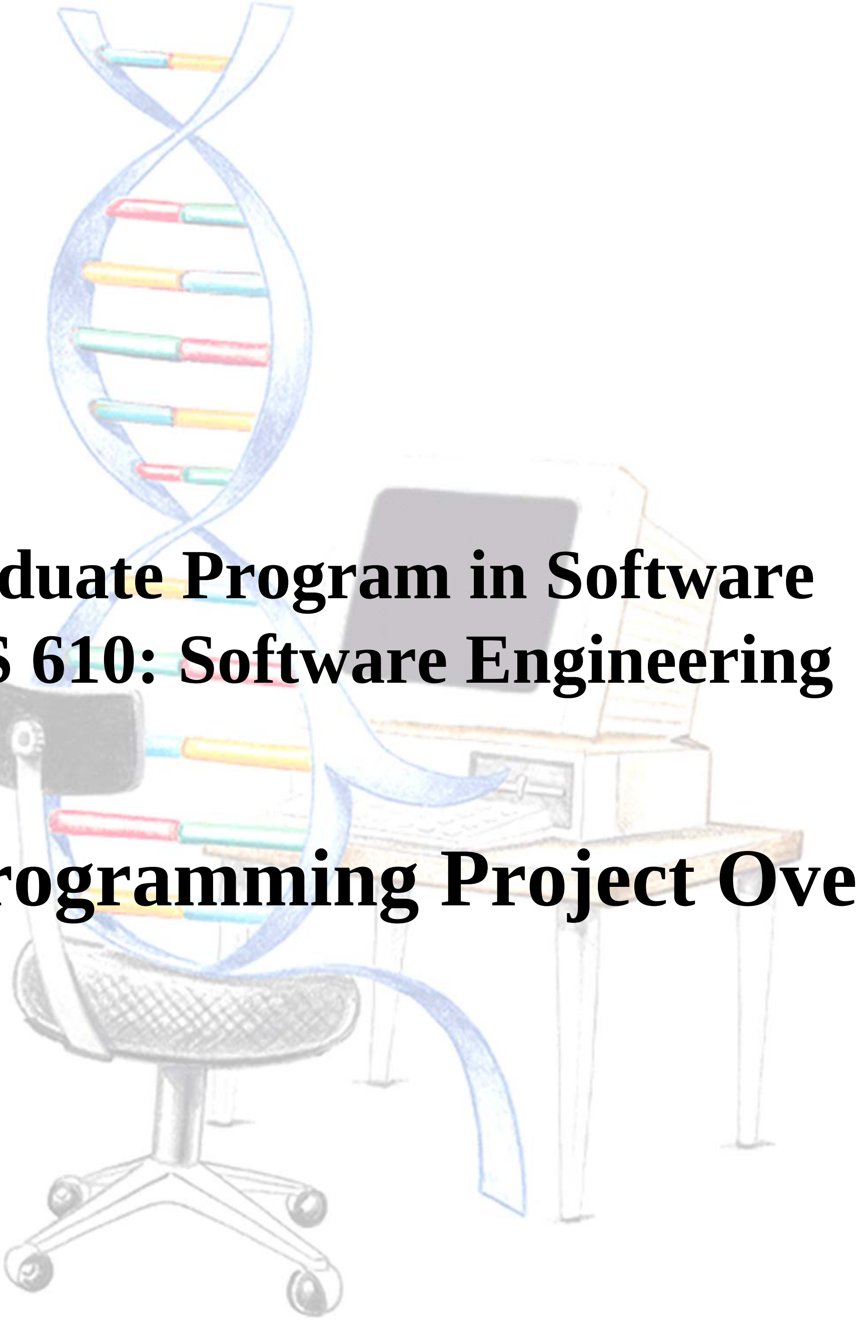
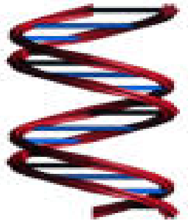


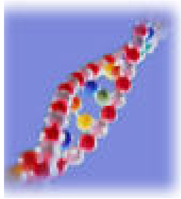
Graduate Program in Software
CSIS 610: Software Engineering

Genetic Programming Project Overview

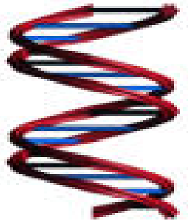
GP Introduction



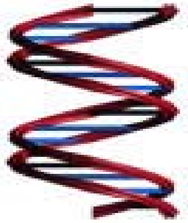
- Inspired by Darwin's theory of evolution
  - Expecting individuals or solutions in a new generation to be better than the ones in old generations
  - Solution will be found by the evolutionary process →
    - **Best** individual (solution/program) survives
  - A fitness function is used to evaluate each individual (solution)
    - Evaluation is done against training data (experiences)



Genetic Programming (GP) Introduction

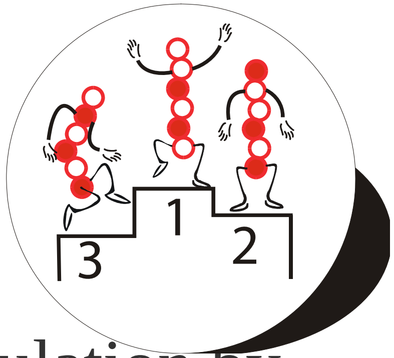


- Genetic programming is a system that computer programs can automatically evolve based on the ***experiences*** the system ***learned***
  - Experiences = training data provided to the system
  - Individuals = Programs
  - System or environment = your GP
- Genetic programming = automatic learning of computer program
- GP systems use a learning algorithm (genetic algorithm) that is similar to the DNA evolution
  - *Crossover* and *mutation* are primary transformation operators
  - Sequences of DNA act like instructions in computer programs, mediating the manufacture of proteins
  - In a GP system, computer programs are treated as sequences of DNA, producing (desired) outputs / data



General GP Steps

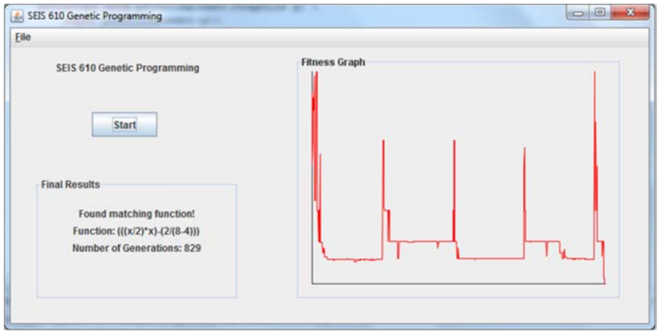
1) Randomly initialize the population of solutions (individuals/programs)
   - 20-30, sometimes 50-100 (your choice!!)



2) Evaluate the fitness of each solution

3) If termination condition not met, generate a new population by
   1) ***Select*** two parent solutions from a population according to their fitness
      - ❖ The better fitness, the bigger chance to be selected
   2) ***Crossover*** the parents to produce a new solution
   3) ***Mutate*** new solutions based on mutation probability, typically very small

4) Go to Step 2
   - You can go back to step 1 for reset after …

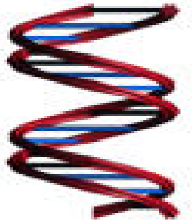


Found matching function!

Function: (((x/2)*x)-(2/(8-4)))

Number of Generations: 829

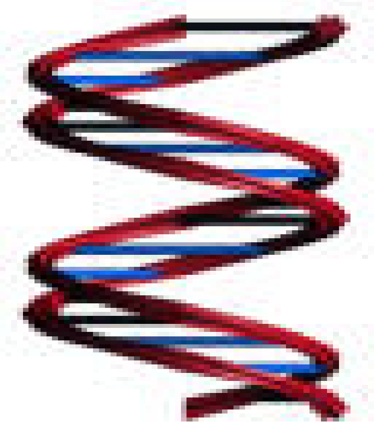
GP Steps, Summary



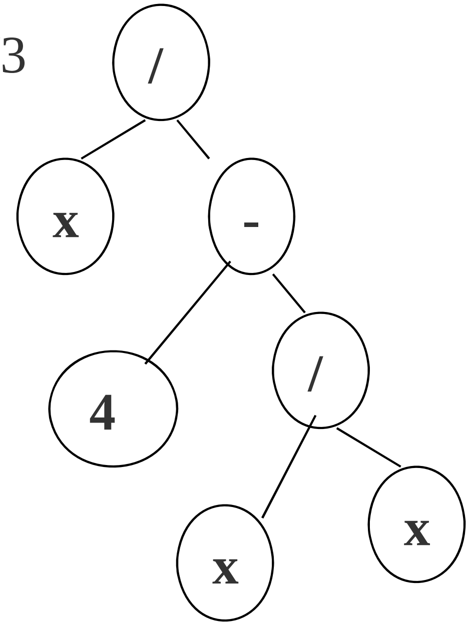
- Each generation produces a population of possible solutions based on the population from the previous generation
  - ***Natural Selection***: Solutions with a good fitness value survive to the next generation with high probability while solutions with a low fitness value perish with a high probability
  - ***Reproduction***: Two solutions chosen via natural selection mate (via ***crossover***) to produce new solutions
  - ***Mutation***: Solutions can undergo spontaneous changes (with small probability) to help prevent all solutions in the population from becoming local (to increase diversity)
    - e.g. Crossover (a, b) (a, c) may produce the same children (a, c) (a, b)
- A GP searches a "potentially" vast solution space for an optimal or ***"near optimal"*** solution

Data Representation??

- OK, let’s just do it!!
- OK, let’s begin programming / coding / construction!!
- But……
- How does each individual look like???
  - Data representation……

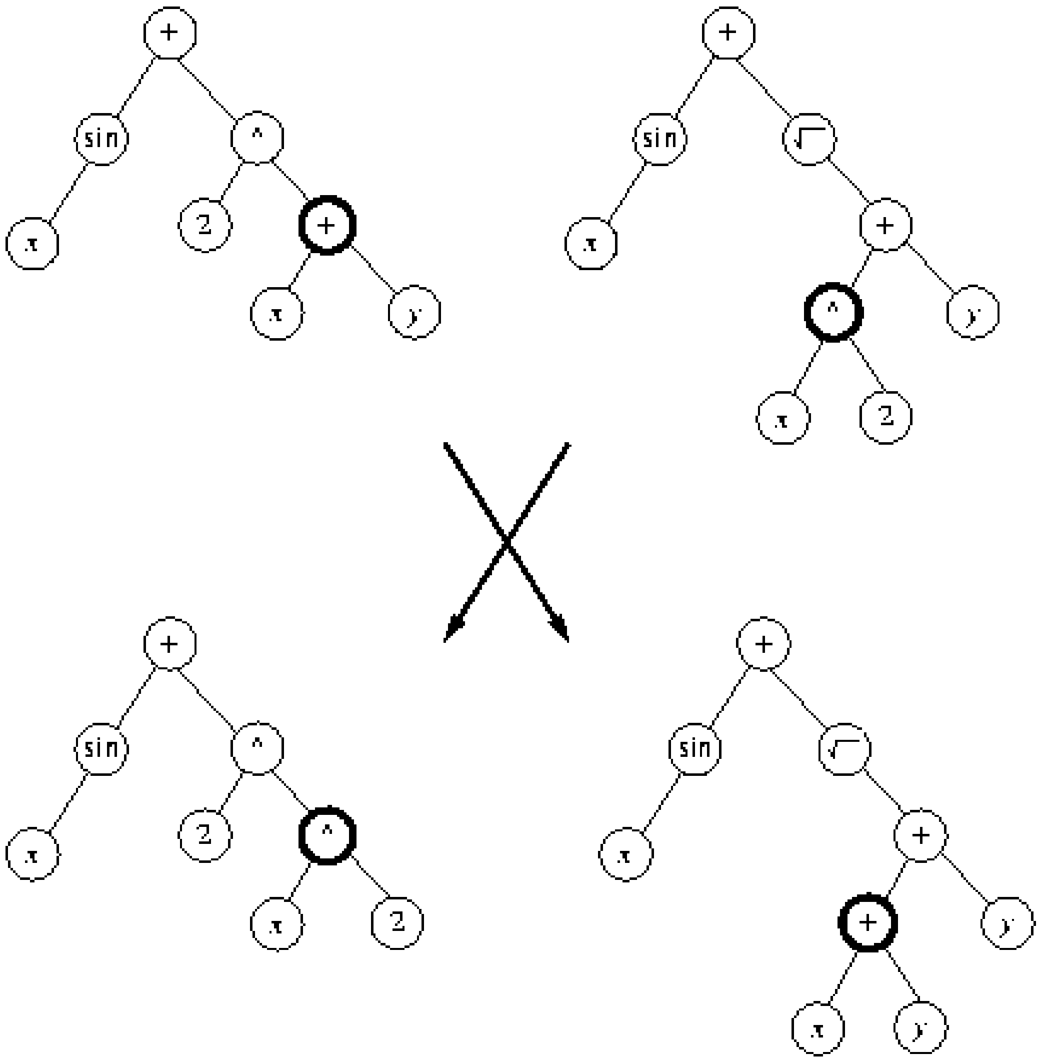


$f_0(x)=x/3$

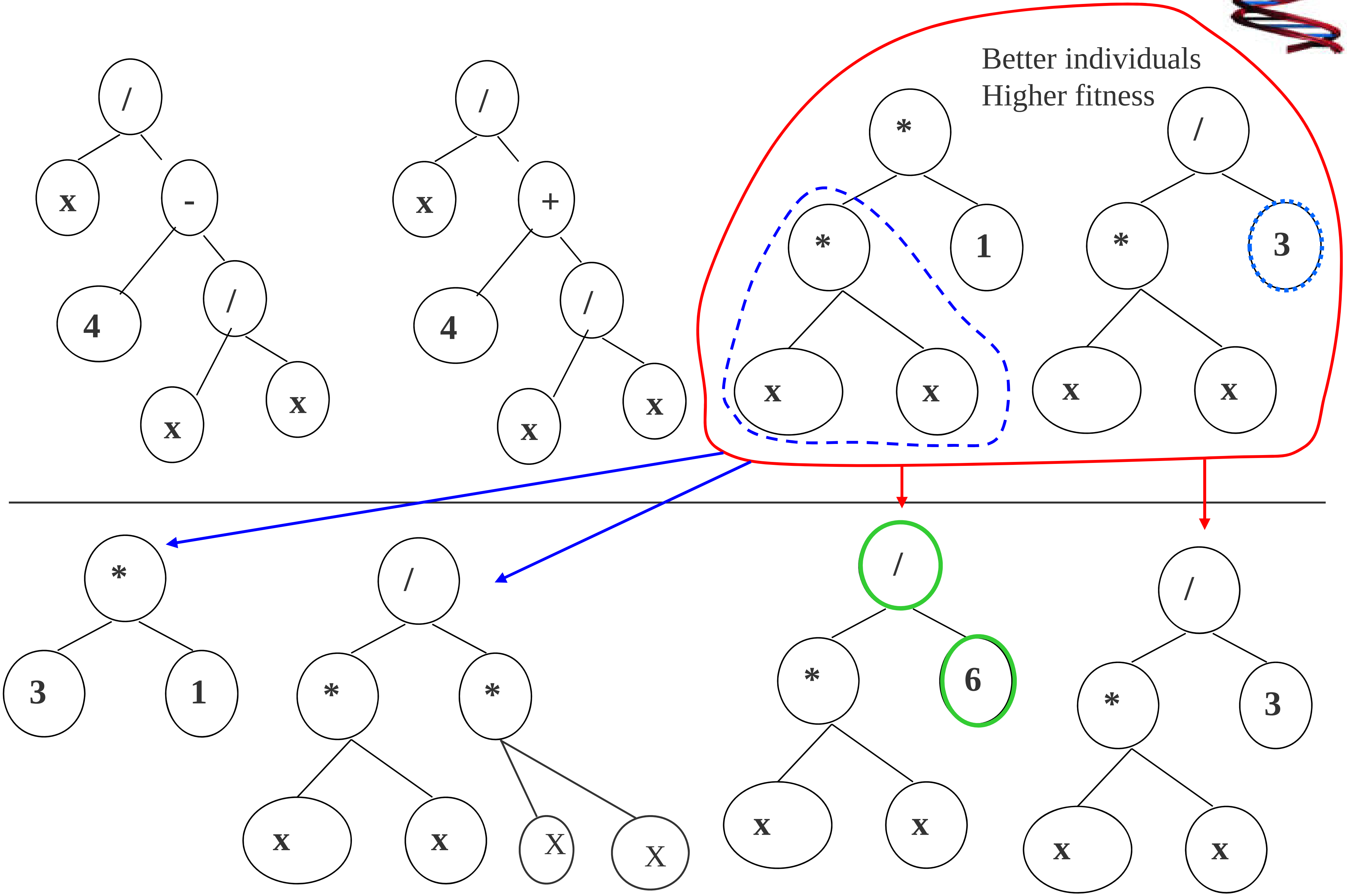


One GP *Crossover* Example

- Each individual (solution / program) is represented as a ***program tree***



Generations


Better individuals
Higher fitness

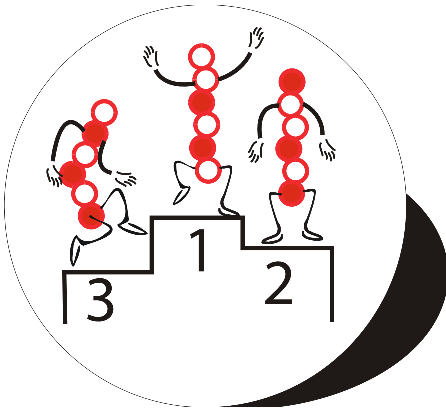
Fitness and Selection

- Suppose we need a function of $(x^2-1) / 2$

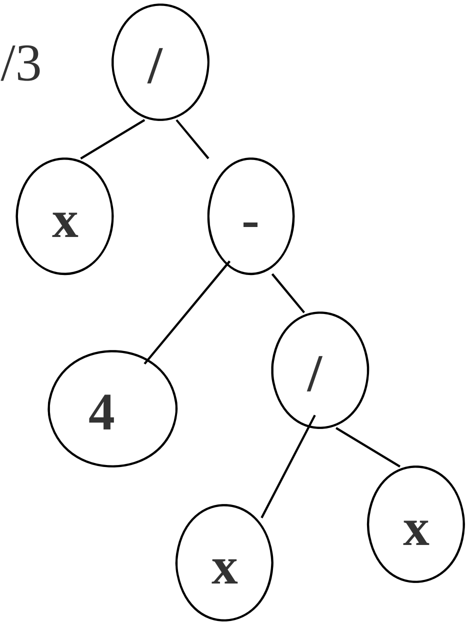
Higher fitness

Training examples

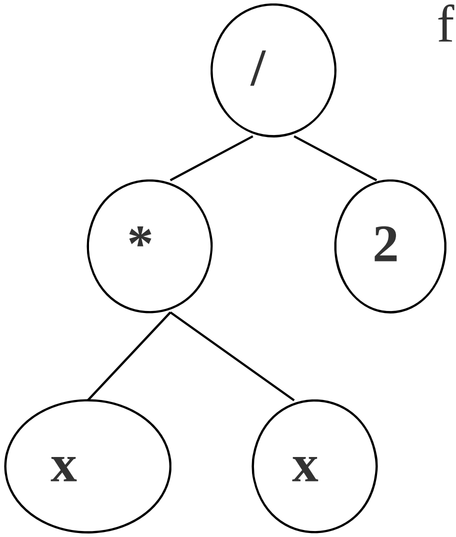
x	$y = (x^2-1) / 2$		$y = f_0$		$y = f_3$
0	-0.5		0		0
1	0		0.333		0.5
2	1.5		0.666		2
3	4		1		4.5



$f_0(x)=x/3$

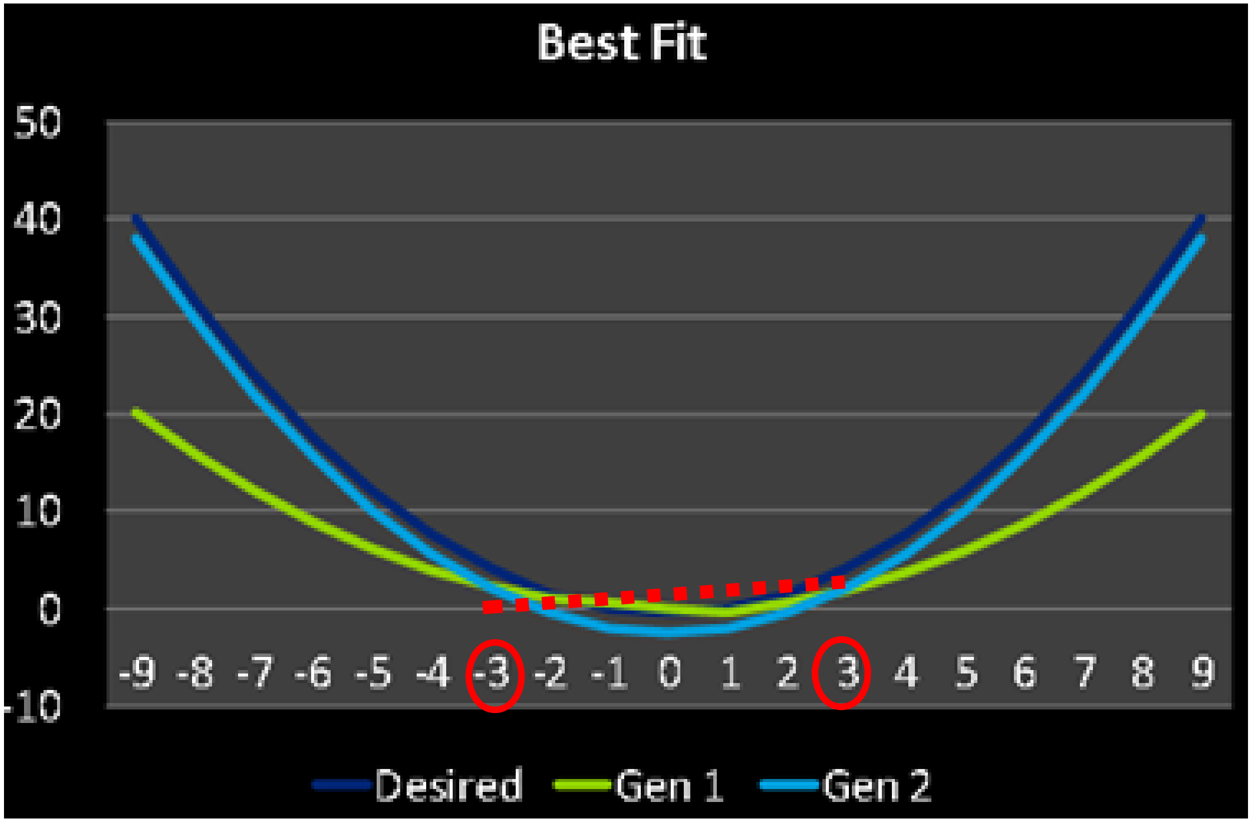


$f_3(x)=x^2/2$



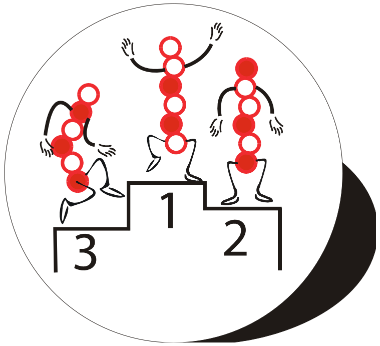
Training Data Selection

- Use large range of training inputs

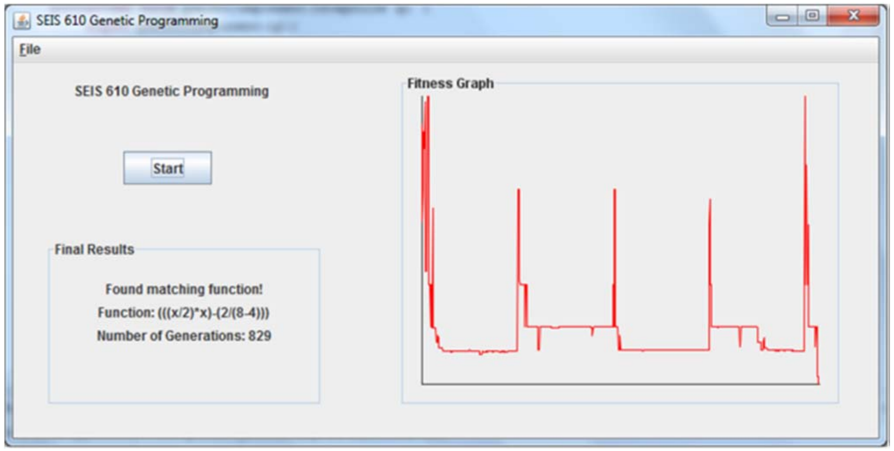


Primary GP Operators

- Selection
  - ***The proportionate selection***: An individual i with fitness value f_i is allocated f^* / f_i offspring



- Crossover
- Mutation
- Fitness evaluation



Found matching function!

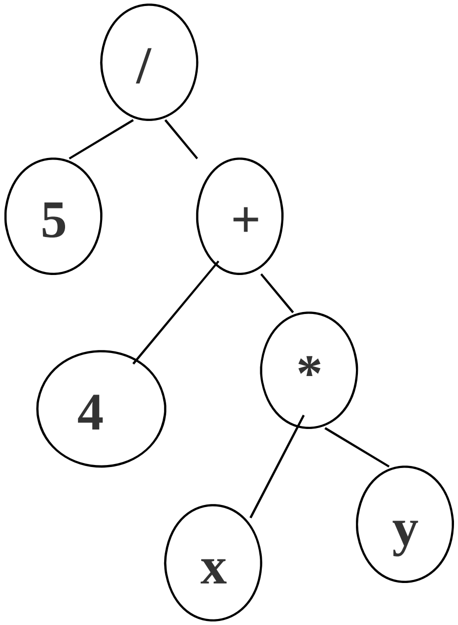
Function: (((x/2)*x)-(2/(8-4)))

Number of Generations: 829

Do we have enough information to build our GP?

Additional Information Needed for A GP

- ***Terminal set*:**
  - The Terminal Set is composed of the *inputs* to the GP program
  - *e.g. 0, 1, 2, …. , 9, x, y, z….*
- ***Function set*:**
  - The function set is composed of the statements, *operators*, and functions available to the GP system
  - *e.g.* +, - *, /
  - *SQRT(), ……*
- Like A, T, G, C in genes

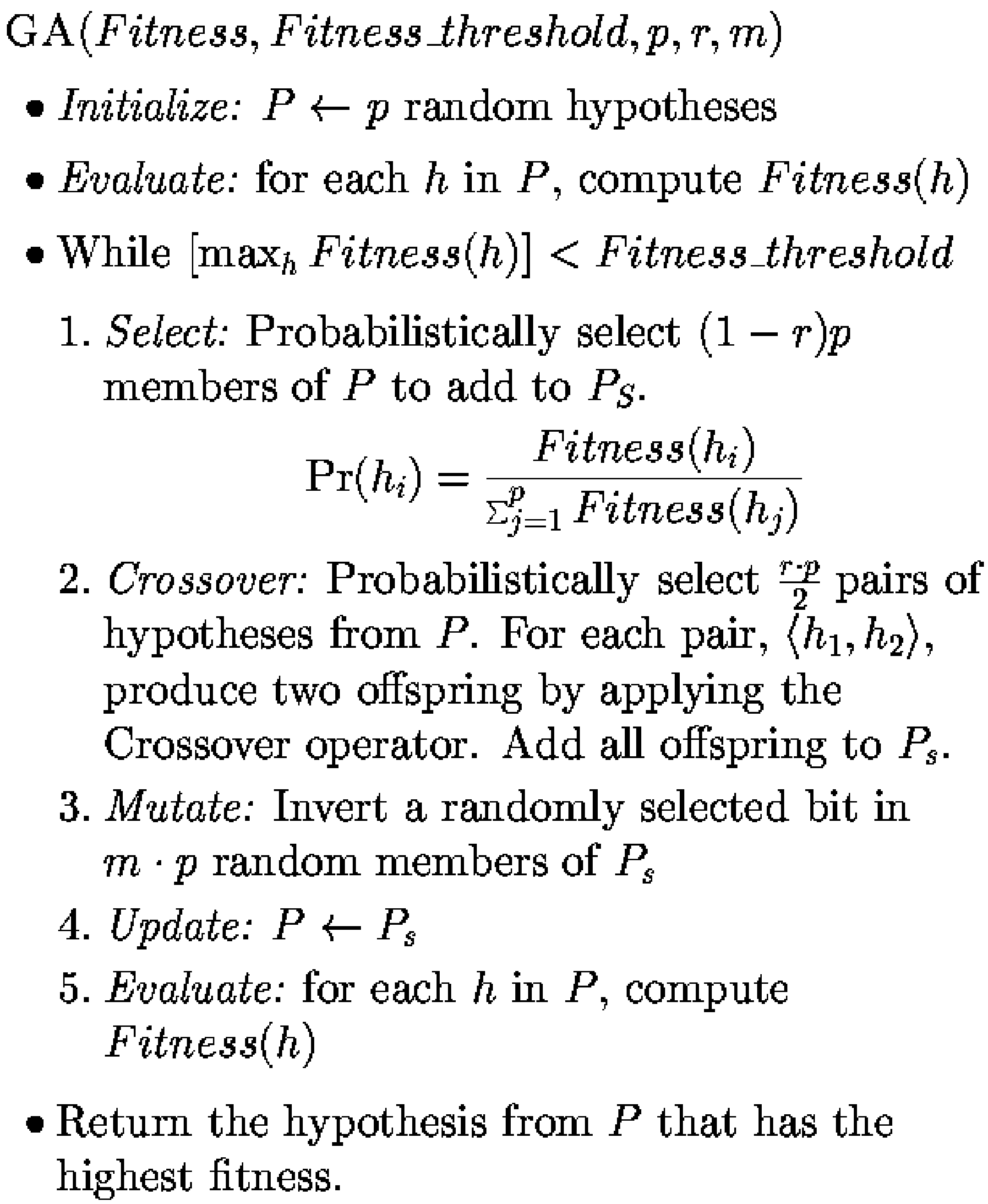


- ***15-minute limitation***

GP Implementation

- Terminal set: *x* and constants
- Function set: +, -,* and /
- Crossover: Single point crossover (to create two new trees according to two parent trees)
- Mutation: Change functions or terminals
- Fitness Function: Compare produced outputs with expected outputs
  - With training data
- Abstract Data Structures / algorithms: binary tree, traversal, stack

Detailed Genetic Algorithm



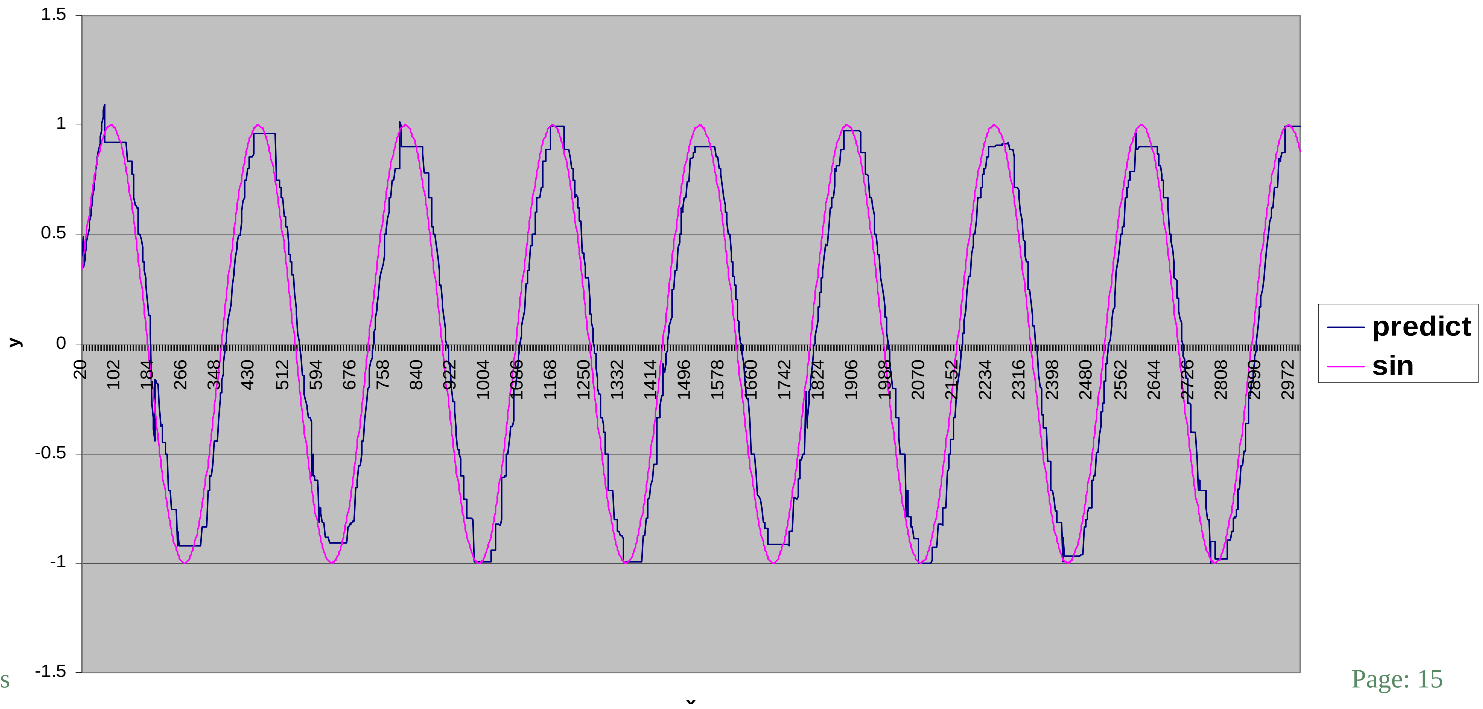
```
$GA(Fitness, Fitness\_threshold, p, r, m)$

• Initialize: $P \leftarrow p$ random hypotheses
• Evaluate: for each $h$ in $P$, compute $Fitness(h)$
• While $[\max_h Fitness(h)] < Fitness\_threshold$
    1. Select: Probabilistically select $(1-r)p$ members of $P$ to add to $P_S$.
           $\Pr(h_i) = \frac{Fitness(h_i)}{\Sigma_{j=1}^{p} Fitness(h_j)}$
    2. Crossover: Probabilistically select $\frac{r \cdot p}{2}$ pairs of hypotheses from $P$. For each pair, $\langle h_1, h_2 \rangle$, produce two offspring by applying the Crossover operator. Add all offspring to $P_s$.
    3. Mutate: Invert a randomly selected bit in $m \cdot p$ random members of $P_s$
    4. Update: $P \leftarrow P_s$
    5. Evaluate: for each $h$ in $P$, compute $Fitness(h)$
• Return the hypothesis from $P$ that has the highest fitness.
```

What Rule(s) Are Embedded in The Data?

Find the rule of data:

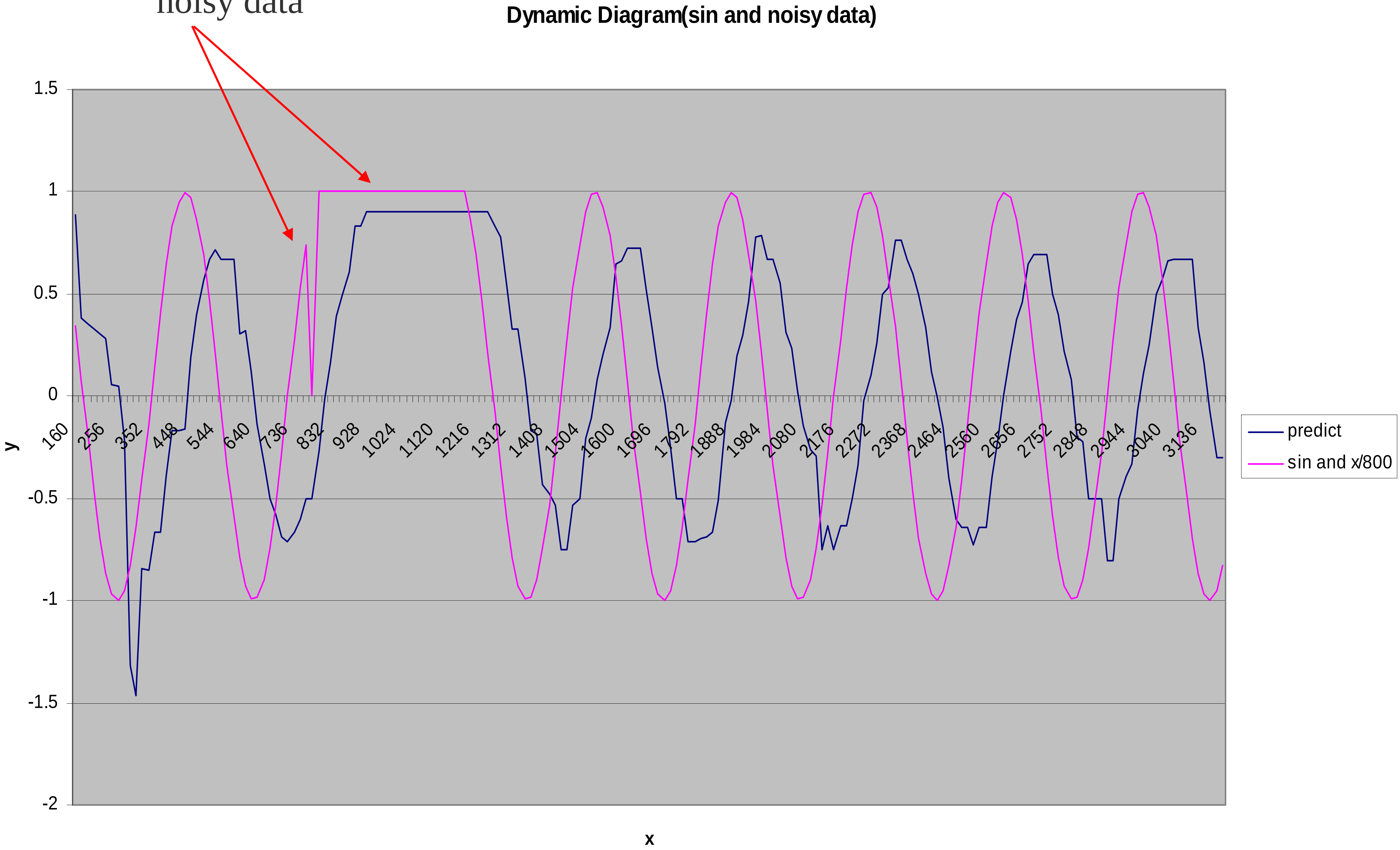
x:	1,	42,	87,	140,	191,	237,	281,	351,	432
y:	0.02,	0.67,	0.99,	0.64,	-0.19,	-0.84,	-0.98,	-0.15,	0.95

If the above data is the only thing your customer knows, then you need to implement a **BIG** switch statement

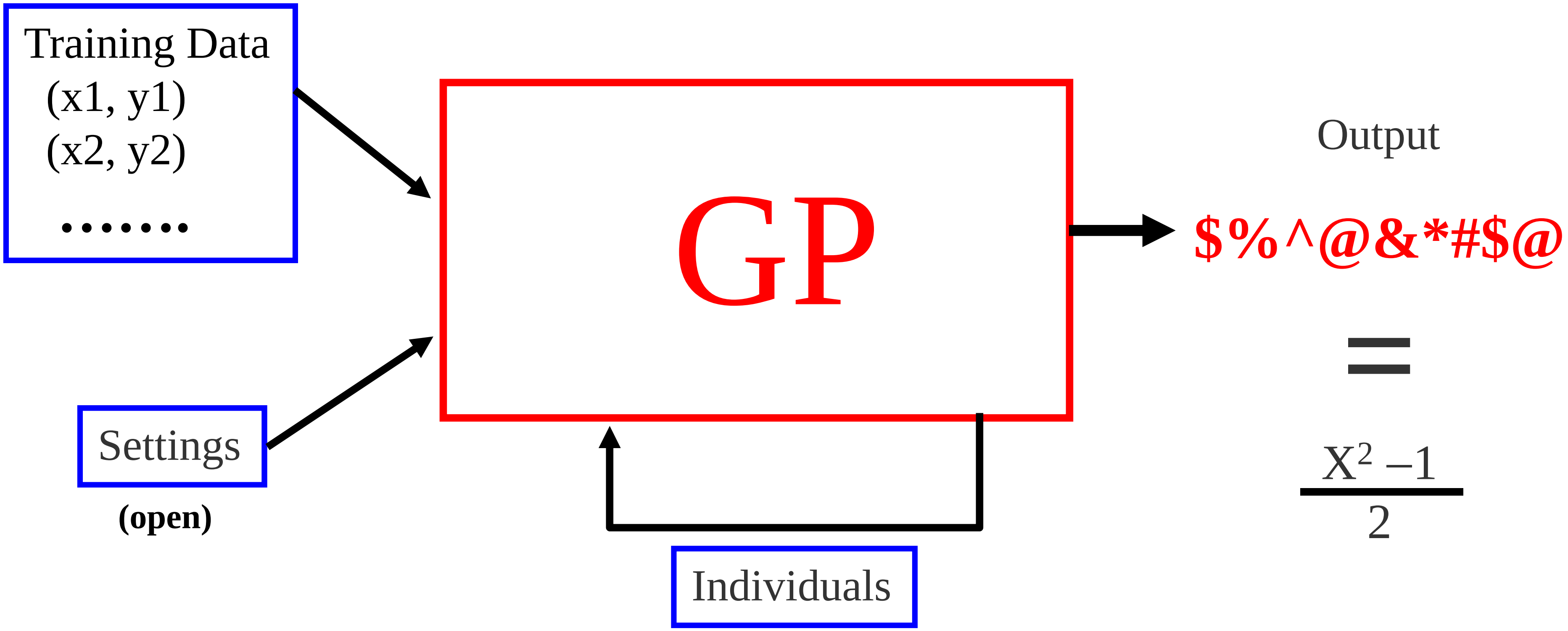


Predicted sin(x) Curve With Noisy Data

noisy data



Project Summary



Training Data
(x1, y1)
(x2, y2)
…….

Settings

(open)

GP

Individuals

Output

$%^@&*#$@

$$= \frac{X^2 - 1}{2}$$

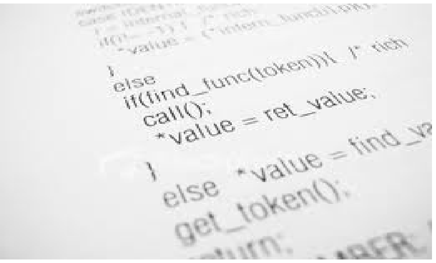
http://www.algebrahelp.com/calculators/equation/simplifying/

For example, $%^@&*#$@ =
(x * x + 20 -6 / 4/ 9 * 114) / (2 * x + 5 – 6 / 2 – x)

Using Utilities or Building Your Own

- Goal– produce a small software project
  - In a team of ***X*** people
  - How do you break down to teams??

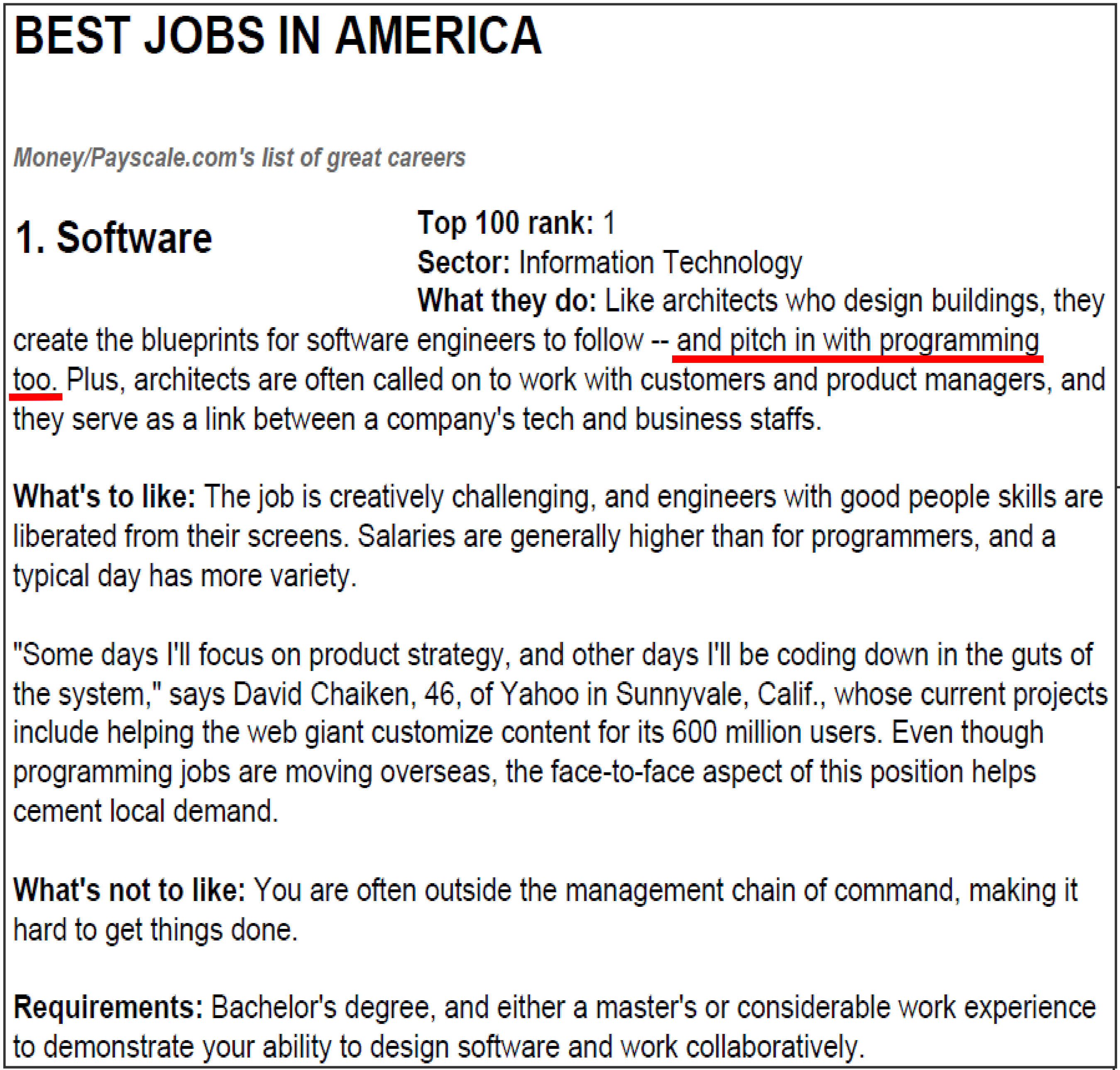
- Using utilities or building your own code?
  - Flexibility, Learning curve



```
else
if(find_func(token)){ /* rich
call();
*value = ret_value;
}
else *value = find_va
get_token();
```

  - Extra credits for building your own code
  - Extra credits for excellent ***quantitative*** analysis of **utilities**
    - i.e. Structural & component complexity, logical complexity, performance monitor
    - Not just simple statements without **quantitative** measurements / evidences
      - "I like it because it just works."
      - "I like it because I tossed a coin."
      - "I like it because it is easy to understand."

Programming or Not Programming



BEST JOBS IN AMERICA

Money/Payscale.com's list of great careers

1. Software

Top 100 rank: 1
Sector: Information Technology

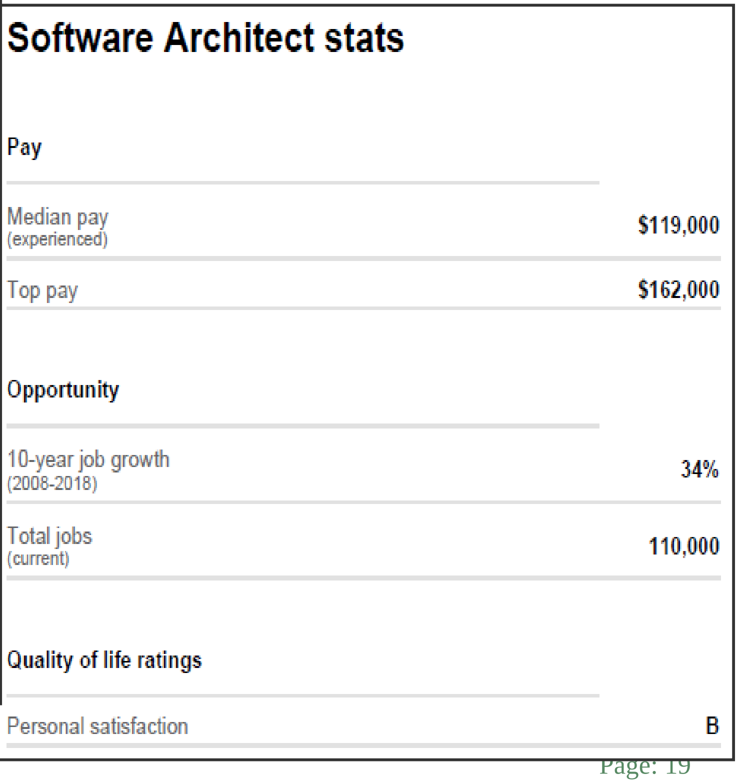
What they do: Like architects who design buildings, they create the blueprints for software engineers to follow -- and pitch in with programming too. Plus, architects are often called on to work with customers and product managers, and they serve as a link between a company's tech and business staffs.

What's to like: The job is creatively challenging, and engineers with good people skills are liberated from their screens. Salaries are generally higher than for programmers, and a typical day has more variety.

"Some days I'll focus on product strategy, and other days I'll be coding down in the guts of the system," says David Chaiken, 46, of Yahoo in Sunnyvale, Calif., whose current projects include helping the web giant customize content for its 600 million users. Even though programming jobs are moving overseas, the face-to-face aspect of this position helps cement local demand.

What's not to like: You are often outside the management chain of command, making it hard to get things done.

Requirements: Bachelor's degree, and either a master's or considerable work experience to demonstrate your ability to design software and work collaboratively.



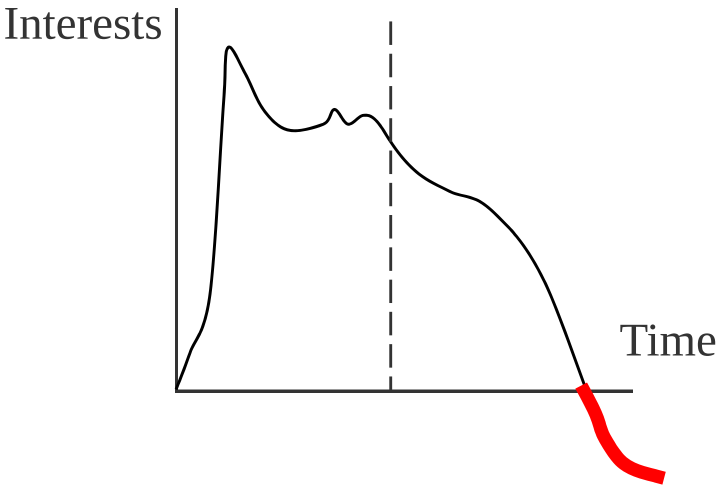
Software Architect stats

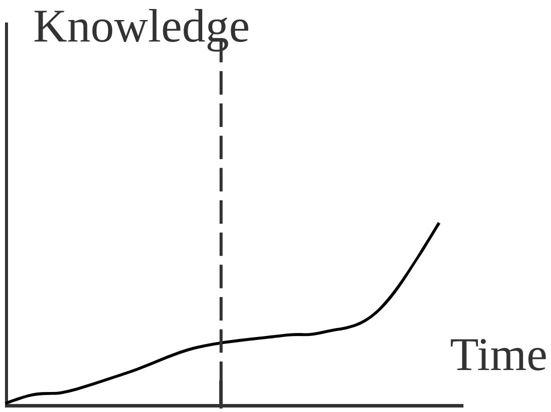
Pay	
Median pay (experienced)	$119,000
Top pay	$162,000
Opportunity	
10-year job growth (2008-2018)	34%
Total jobs (current)	110,000
Quality of life ratings	
Personal satisfaction	B

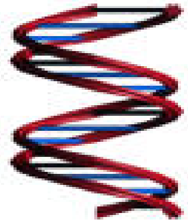
Project

- We wish we started earlier
- We all have full-time jobs➔ hard to find common time to meet or communicate
- We have to spend time with family (or I almost become single)
- No common background / programming language knowledge
- There are some unexpected problems

- It’s all the instructor’s (customer’s) fault







Web Sites for Drawing Functions

- Function Grapher:

  http://rechneronline.de/function-graphs/

- Infix, Post-fix, Pre-fix Parser:

  http://www.unf.edu/~rzucker/cot3100dir/parser.html

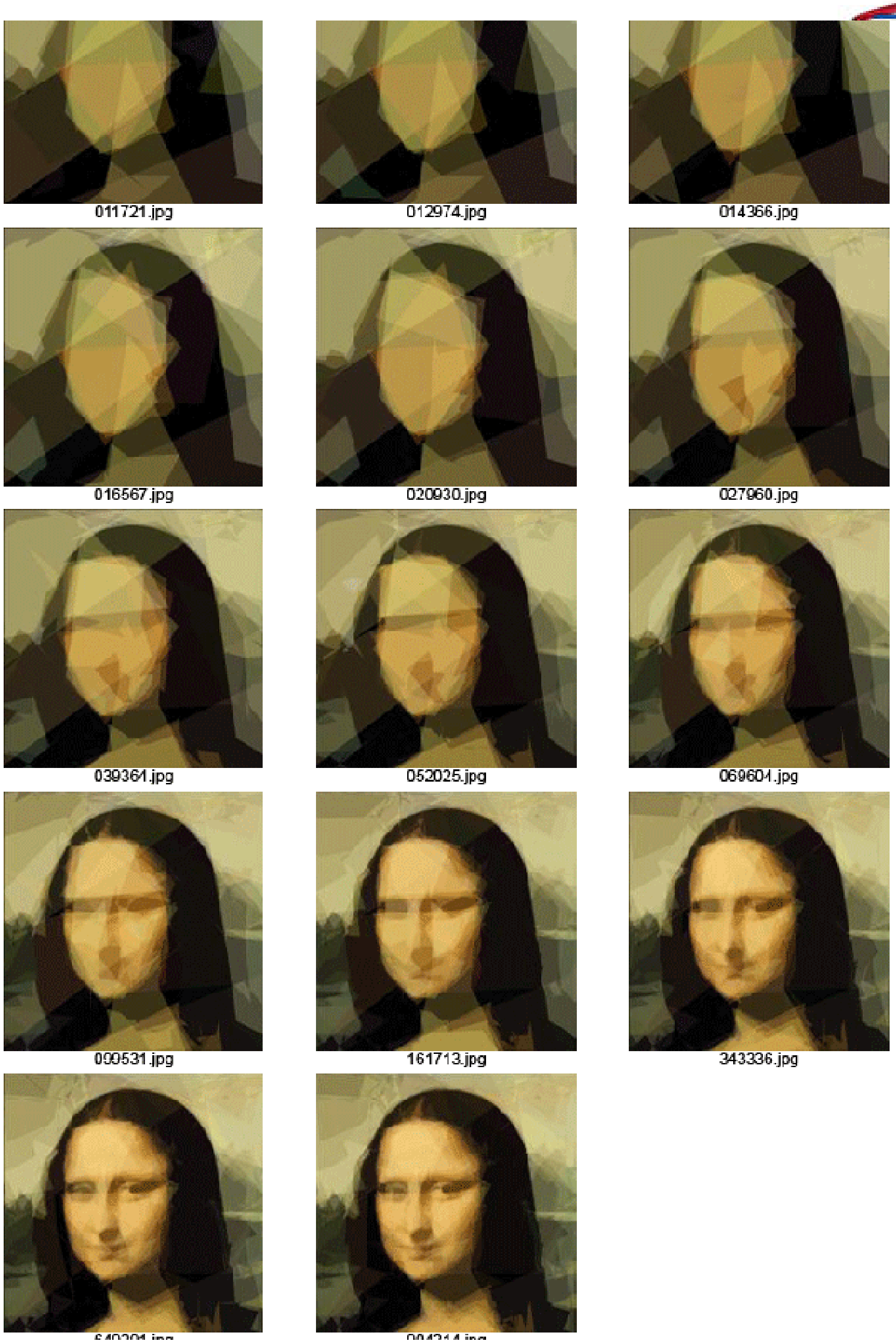
- http://www.algebrahelp.com/calculators/equation/simplifying/

- http://rogeralsing.com/2008/12/07/genetic-programming-evolution-of-mona-lisa/

Evolution of Mona Lisa

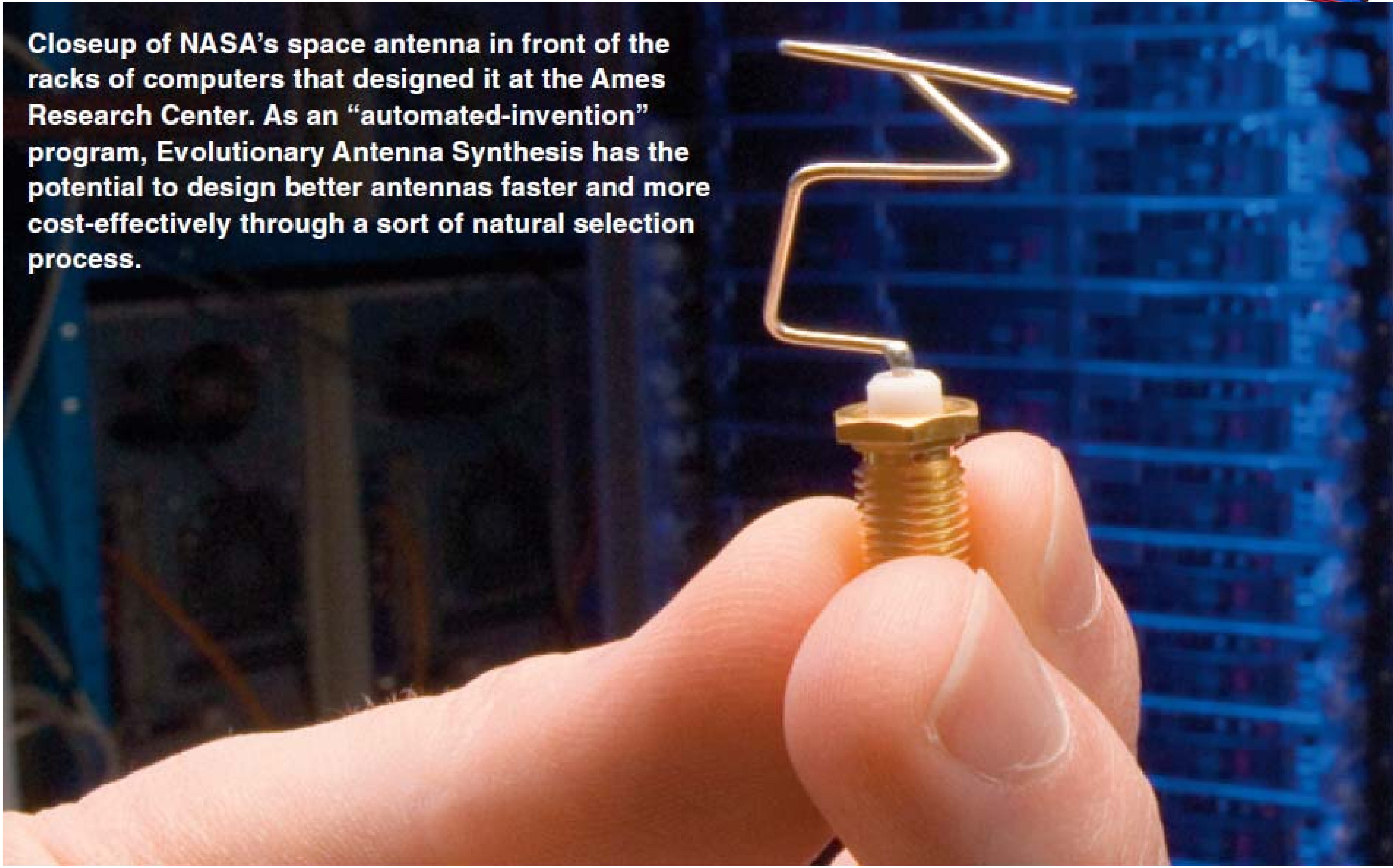
- Each individual has random combination of polygons of different colors at different locations

- If a new painting looks more like the source image than a previous painting did, then the new painting survives



http://rogeralsing.com/2008/12/07/genetic-programming-evolution-of-mona-lisa/

Another Application of GP



Closeup of NASA's space antenna in front of the racks of computers that designed it at the Ames Research Center. As an "automated-invention" program, Evolutionary Antenna Synthesis has the potential to design better antennas faster and more cost-effectively through a sort of natural selection process.

THE FUTURIST July-August 2009 www.wfs.org