The Traveling Salesman Problem, Genetic Algorithms, and Ant Colony Optimization

Adam Byerly
adam@rokrol.com
(309) 369-9257

The Traveling Salesman Problem

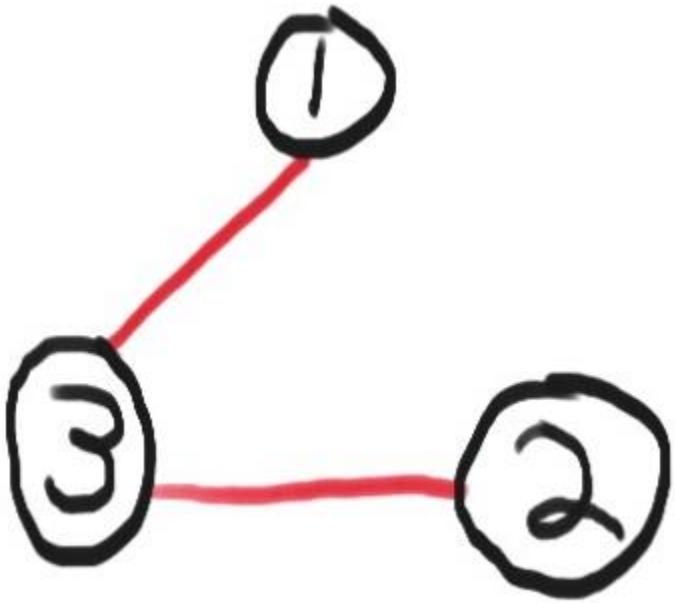
The TSP is worth studying because it is a good analog for a broad group of real world problems including planning, logistics, microchip manufacturing, and DNA sequencing [1].

"...one of the most famous and important combinatorial-optimization problems..."

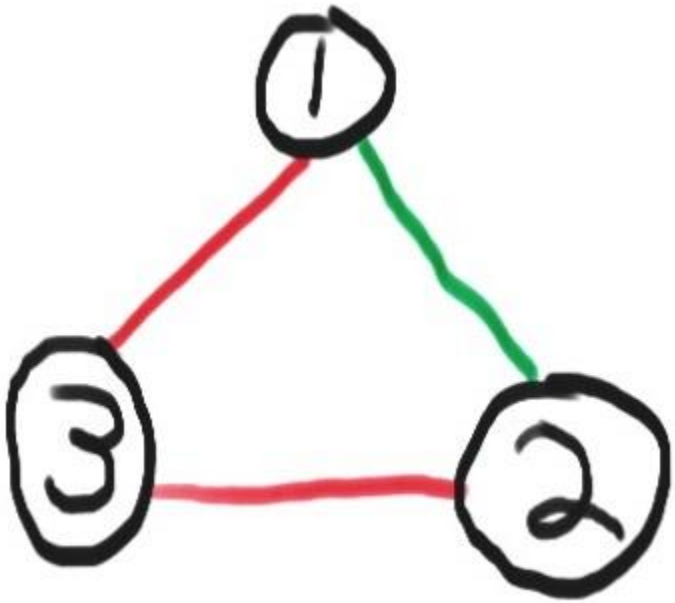
[1] Applegate, David L. The Traveling Salesman Problem: A Computational Study. Princeton: Princeton UP, 2006.

The Traveling Salesman Problem

The Traveling Salesman Problem

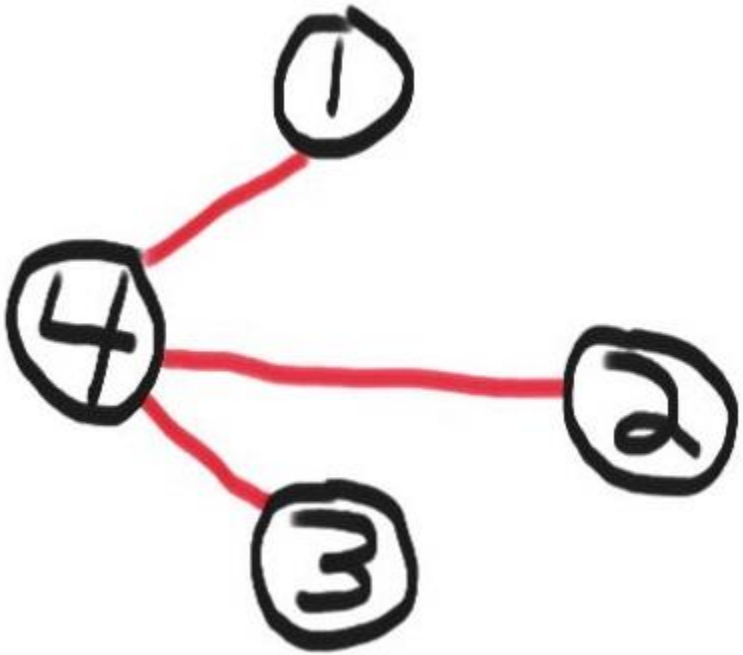


The Traveling Salesman Problem

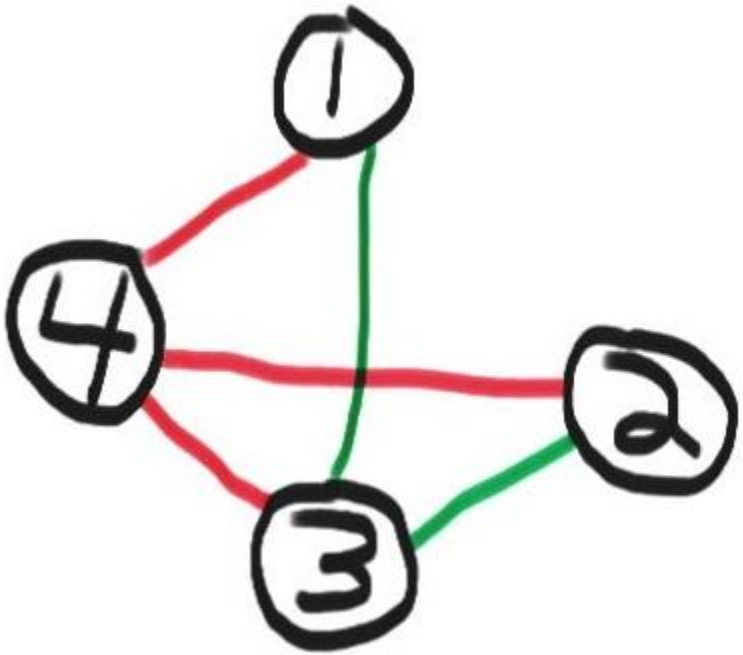


The Traveling Salesman Problem

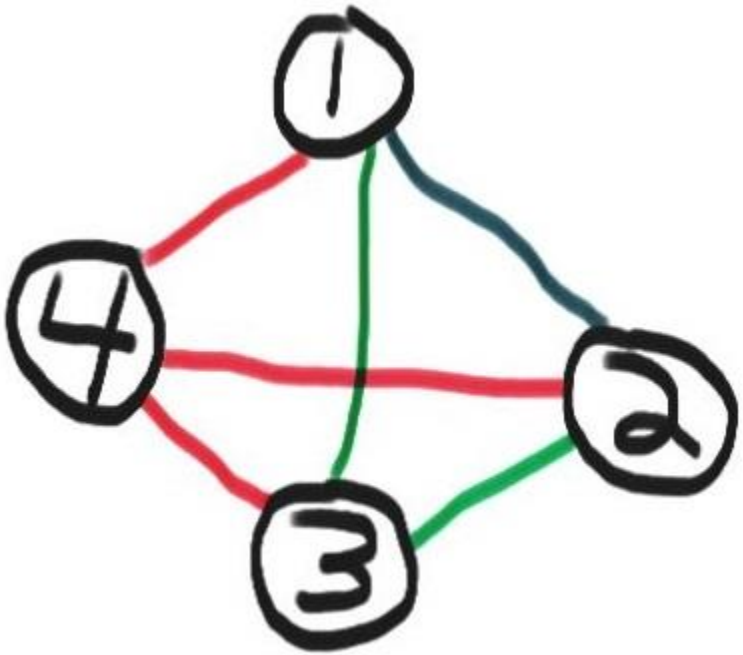
The Traveling Salesman Problem



The Traveling Salesman Problem

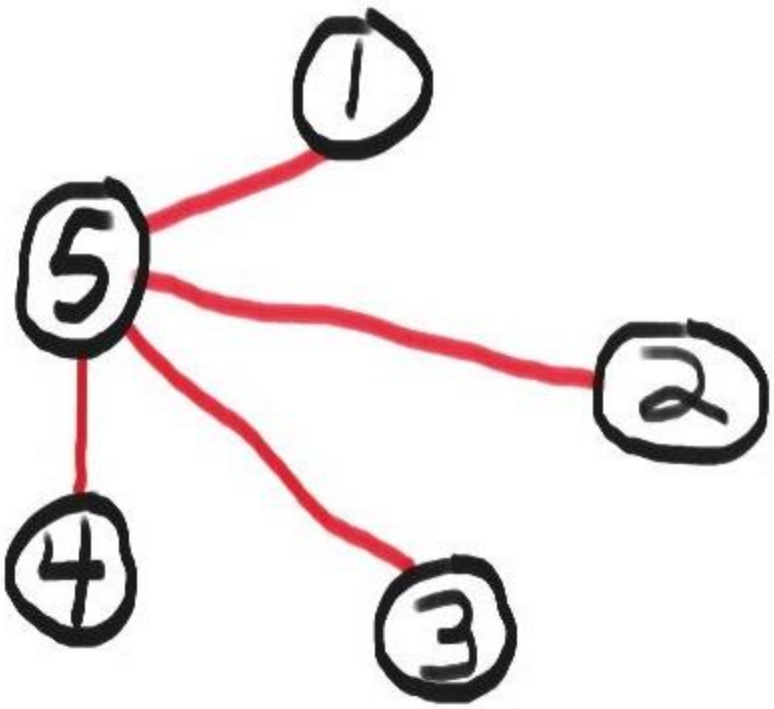


The Traveling Salesman Problem

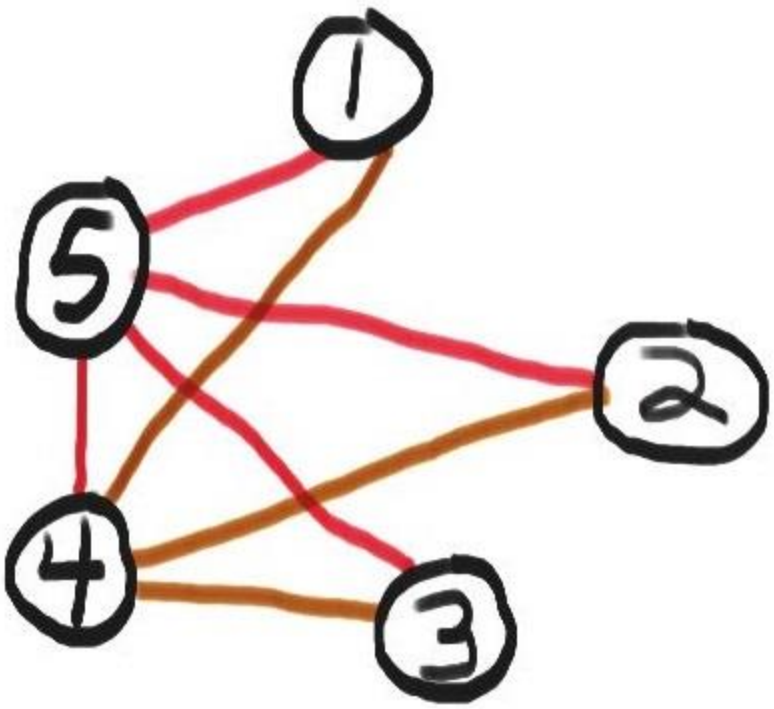


The Traveling Salesman Problem

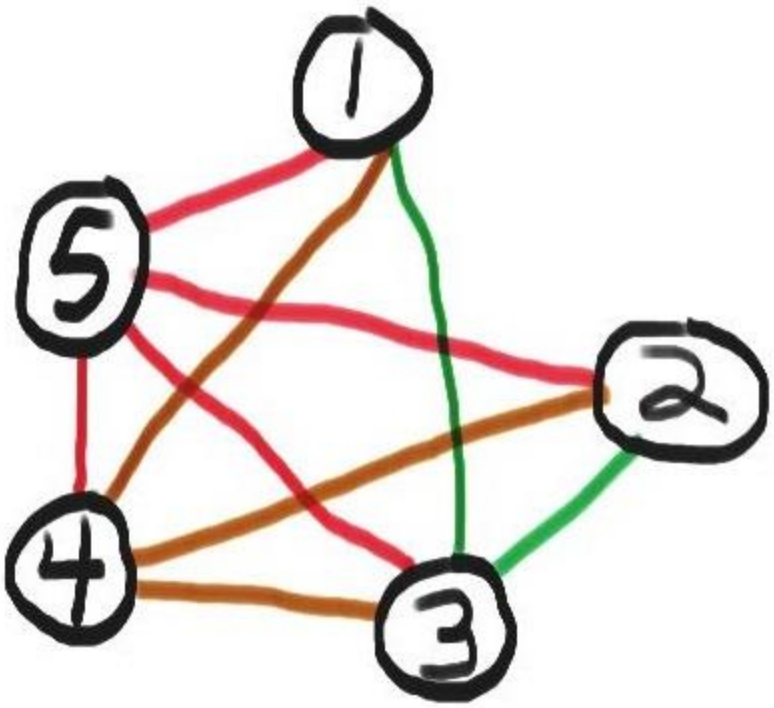
The Traveling Salesman Problem



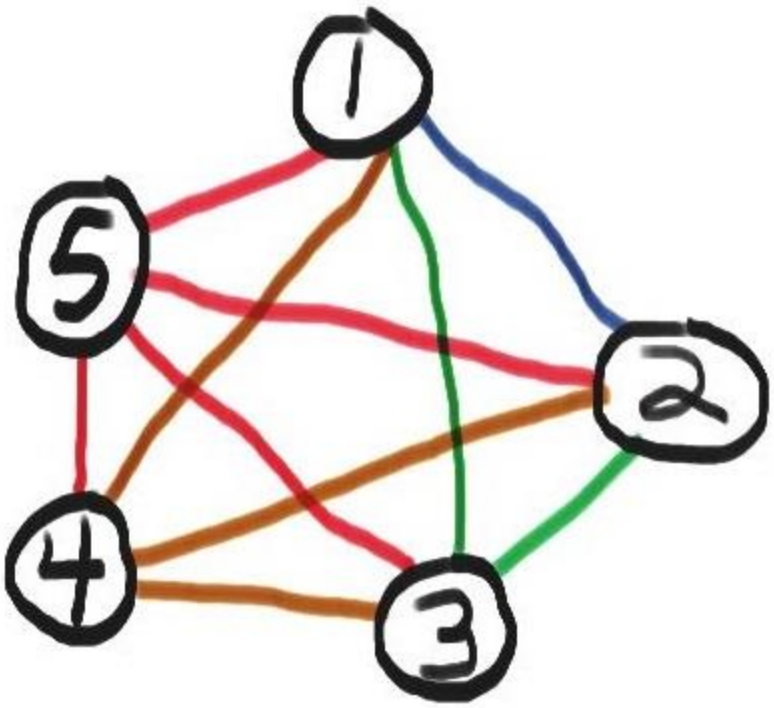
The Traveling Salesman Problem



The Traveling Salesman Problem



The Traveling Salesman Problem



The Traveling Salesman Problem

The number of edges grows fast!

The Traveling Salesman Problem

The number of edges grows fast!

Where n is the number of cities:

$$\frac{n(n-1)}{2} = \frac{n^2 - n}{2}$$

$$O(n^2)$$

The Traveling Salesman Problem

But that's edges.

The number of possible tours the salesman can take is worse!

The Traveling Salesman Problem

But that's edges.

The number of possible tours the salesman can take is worse!

Where n is the number of cities:

Symmetric: $\frac{(n-1)!}{2}$

Asymmetric: $(n-1)!$

$O(n!)$

The Traveling Salesman Problem

- # of milliseconds since the big bang $= 4.35 \times 10^{20}$

The Traveling Salesman Problem

- # of milliseconds since the big bang $= 4.35 \times 10^{20}$
- # of atoms in the universe $= 4 \times 10^{80}$

The Traveling Salesman Problem

- # of milliseconds since the big bang $= 4.35 \times 10^{20}$
- # of atoms in the universe $= 4 \times 10^{80}$

- Cities in U.S. with 500 or more residents $= 13{,}509$

The Traveling Salesman Problem

- # of milliseconds since the big bang $= 4.35 \text{ x } 10^{20}$
- # of atoms in the universe $= 4 \text{ x } 10^{80}$

- Cities in U.S. with 500 or more residents $= 13{,}509$
- Possible TSP solutions $= \sim 1 \text{ x } 10^{50{,}000}$

The Traveling Salesman Problem

- # of milliseconds since the big bang $= 4.35 \times 10^{20}$
- # of atoms in the universe $= 4 \times 10^{80}$

- Cities in U.S. with 500 or more residents $= 13{,}509$
- Possible TSP solutions $= \sim 1 \times 10^{50{,}000}$

n	n!	n	n!
1	1	13	6227020800
2	2	14	87178291200
3	6	15	1307674368000
4	24	16	20922789888000
5	120	17	355687428096000
6	720	18	6402373705728000
7	5040	19	121645100408832000
8	40320	20	2432902008176640000
9	362880	25	$1.551121004 \times 10^{25}$
10	3628800	50	$3.041409320 \times 10^{64}$
11	39916800	70	$1.197857167 \times 10^{100}$

The Traveling Salesman Problem

It's Really Hard!

- The optimization problem is NP-Hard
- The decision version is NP-Complete
- The complexity class is the same:
  - For symmetric and asymmetric versions
  - Whether or not the salesman returns to the starting city

The Traveling Salesman Problem

- There are branch and bound methods that for certain instances of the TSP, we can get a deterministic answer before the end of the universe.
- But in general, we cannot solve arbitrary, non-trivial instances of the TSP.

The Traveling Salesman Problem

- There are branch and bound methods that for certain instances of the TSP, we can get a deterministic answer before the end of the universe.
- But in general, we cannot solve arbitrary, non-trivial instances of the TSP.

  (Unless it turns out that P = NP …the world will get weird in that case…)

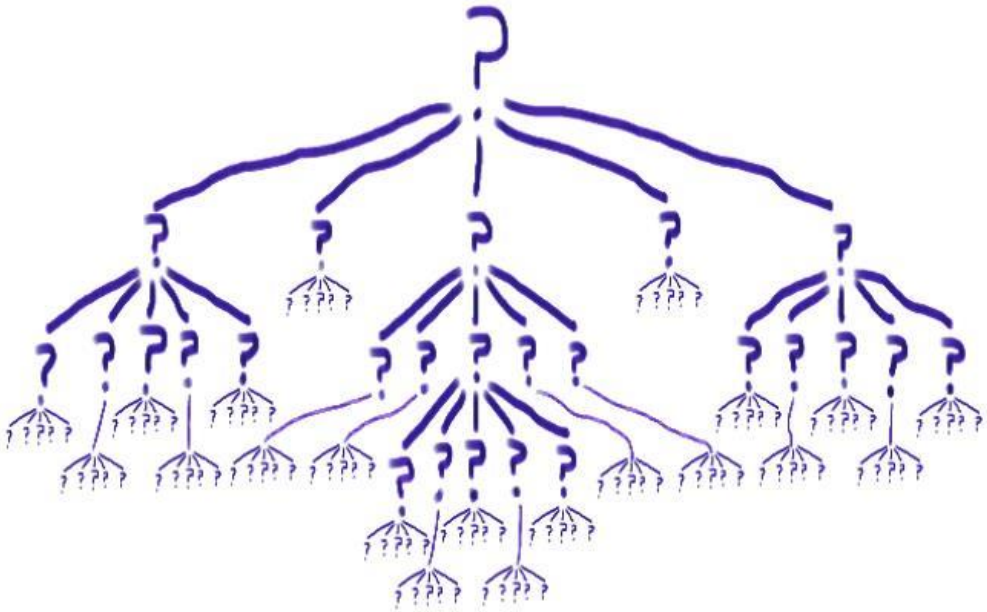
The Traveling Salesman Problem

- There are branch and bound methods that for certain instances of the TSP, we can get a deterministic answer before the end of the universe.
- But in general, we cannot solve arbitrary, non-trivial instances of the TSP.

  (Unless it turns out that P = NP …the world will get weird in that case…)

- We call this property of a problem “intractability”.

The Traveling Salesman Problem



The Traveling Salesman Problem

This is why we need AI!

The Traveling Salesman Problem

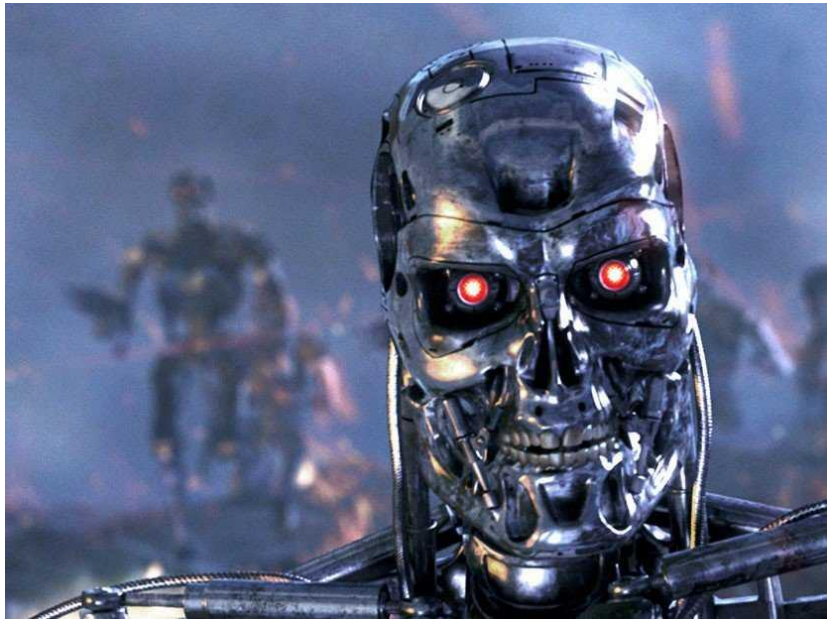
This is why we need AI!

To search a **subset** of, rather than the entire, solution space. In an “intelligent” fashion.

The Traveling Salesman Problem

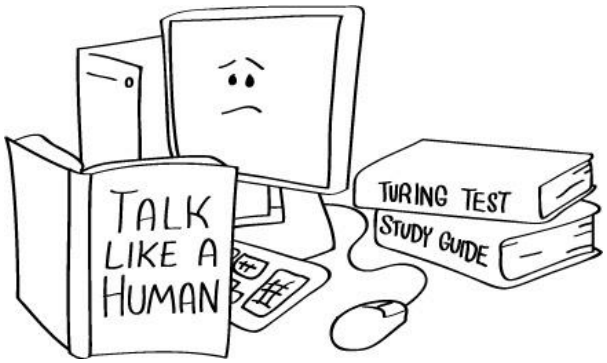
This is why we need AI!

To search a **subset** of, rather than the entire, solution space. In an “intelligent” fashion.



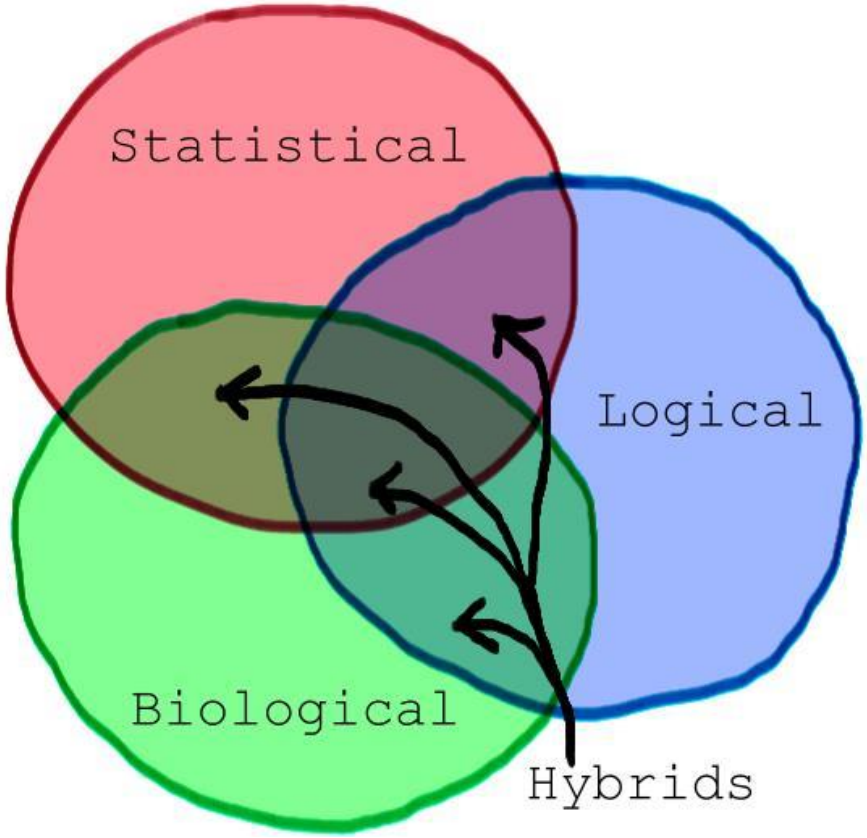
Artificial Intelligence

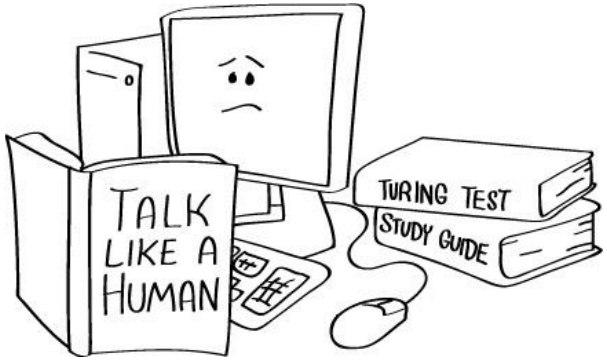
AI, very informally, is reaching “near optimal” or “good enough” solutions in a “reasonable” amount of time.



Artificial Intelligence

AI Families

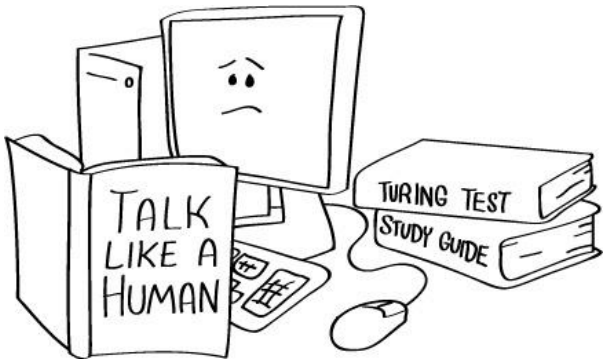




Artificial Intelligence

AI Families

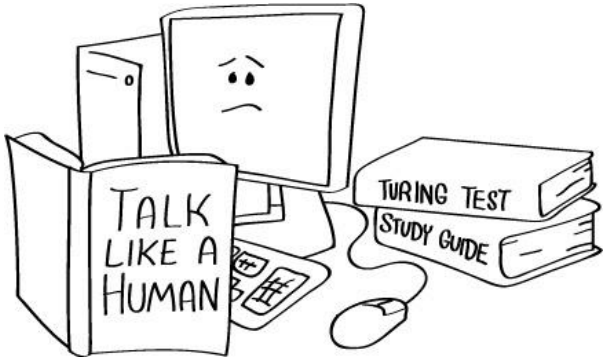
- Logical – Expert Systems
- Statistical – Regression, Clustering, Bayesian Methods, Markov Chains
- Biologically Inspired – Artificial Neural Networks, Genetic Algorithms



Artificial Intelligence

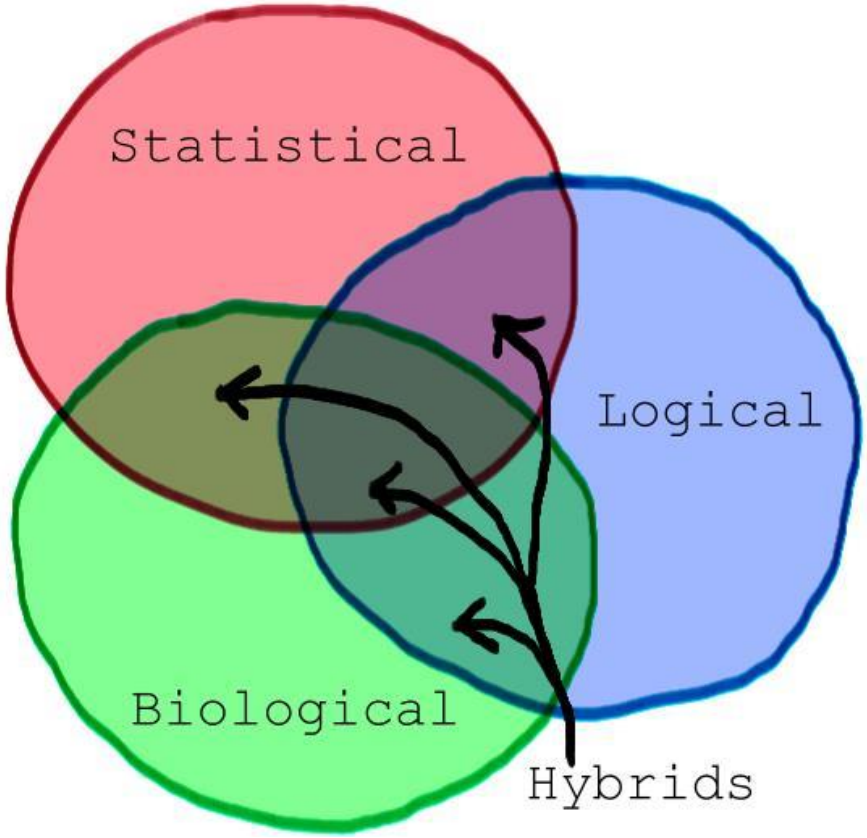
Hybrids

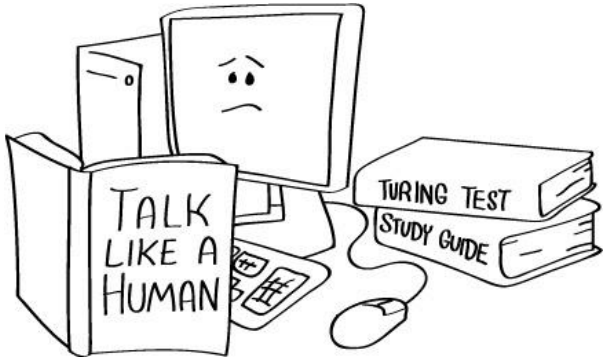
- Fuzzy Logic – part logical, part statistical
- Ant Colony and Swarm algorithms – mostly biologically inspired, but involves statistics and heuristics



Artificial Intelligence

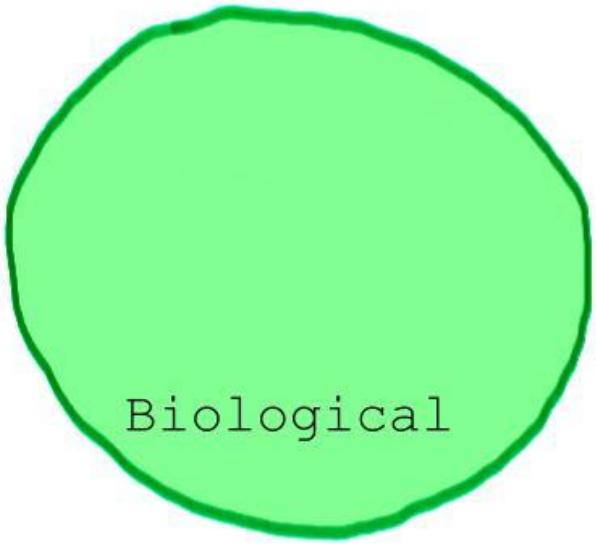
AI Families

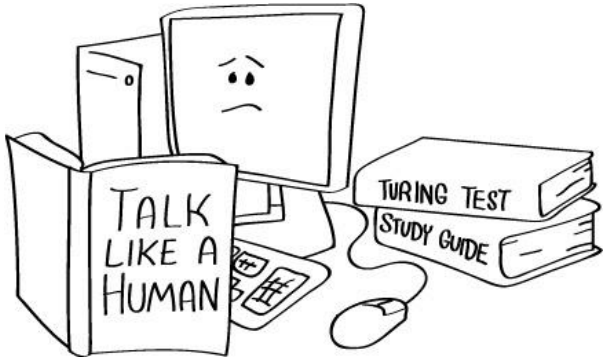




Artificial Intelligence

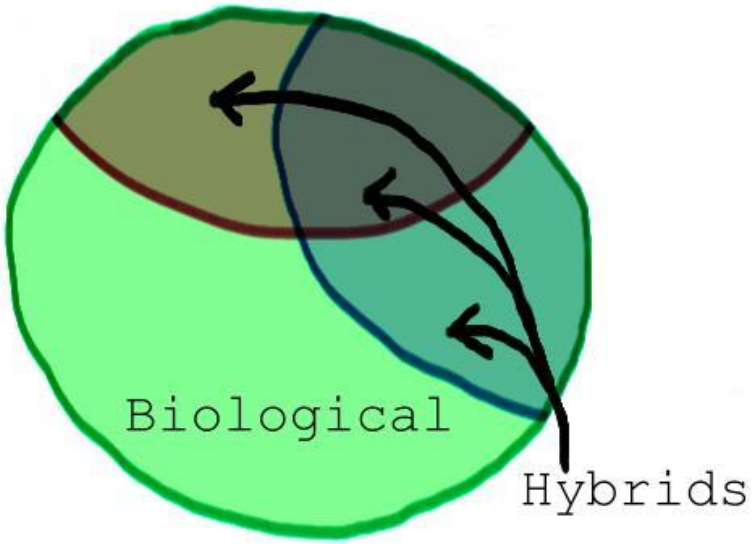
AI Families

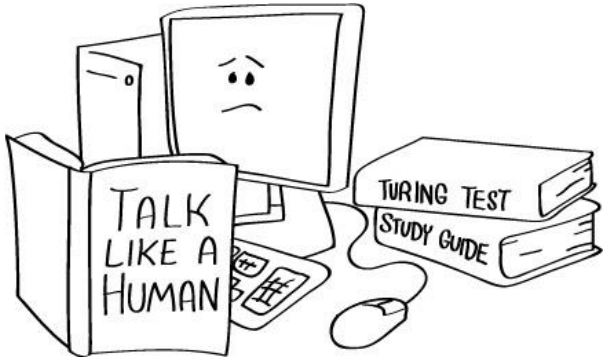




Artificial Intelligence

AI Families





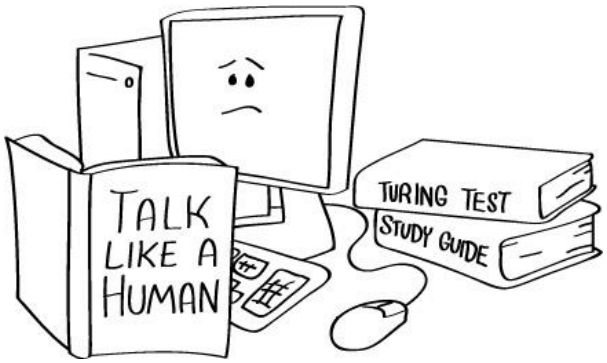
Artificial Intelligence

John Henry Holland

Adaptation in Natural and Artificial Systems (1975, MIT Press)

Introduced the world to Genetic Algorithms

(Tho, others had done some primitive work, including Alan Turing, who in 1950 contemplated a “learning machine” based on evolutionary principles.)

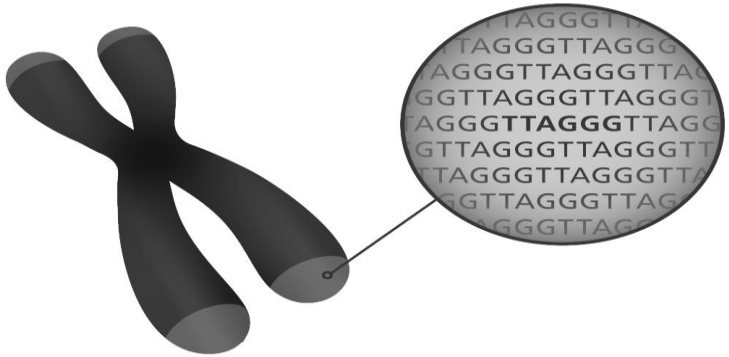


Genetic Algorithms

John Henry Holland

Fun Facts:

- Received the first PhD in Computer Science awarded by Univ. of Michigan in 1959.
- Named a MacArthur Fellow in 1992
- Was PhD advisor to Edgar Codd



Genetic Algorithms

Put a little too simply, GAs involve using

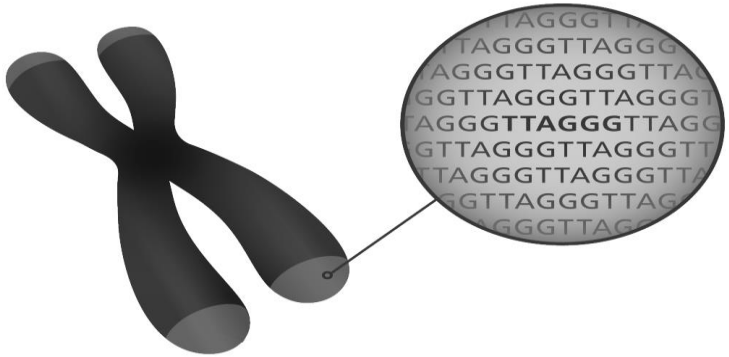
a random number generator to get

“better”

answers to

“hard problems”

“quickly”.



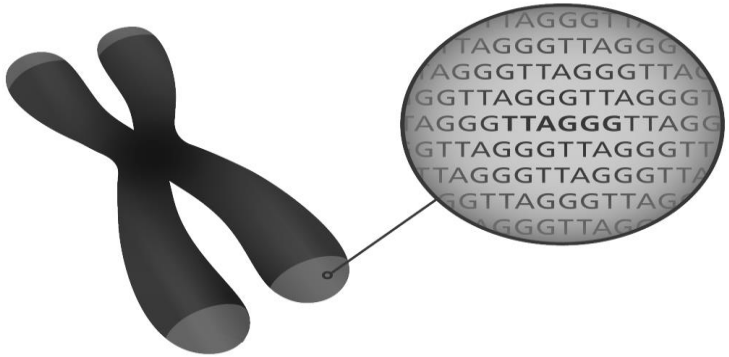
Genetic Algorithms

Best suited for finding

"near optimal"

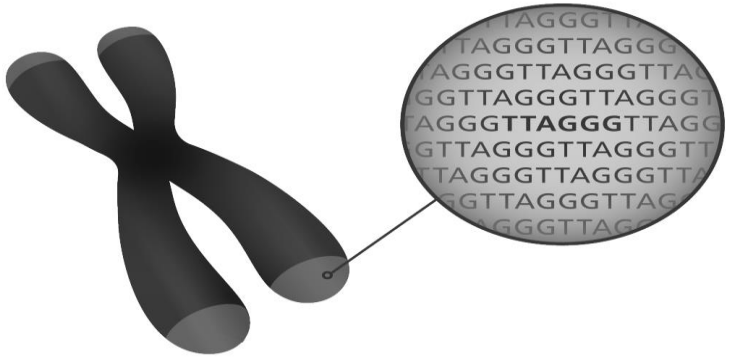
solutions to problems that have so many possible solutions that determining the *true optimal* would take

"too long" (like the TSP).



Genetic Algorithms

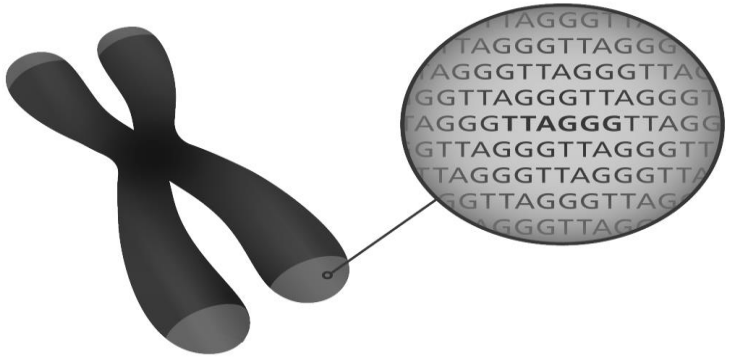
The main components of a GA are:



Genetic Algorithms

The main components of a GA are:

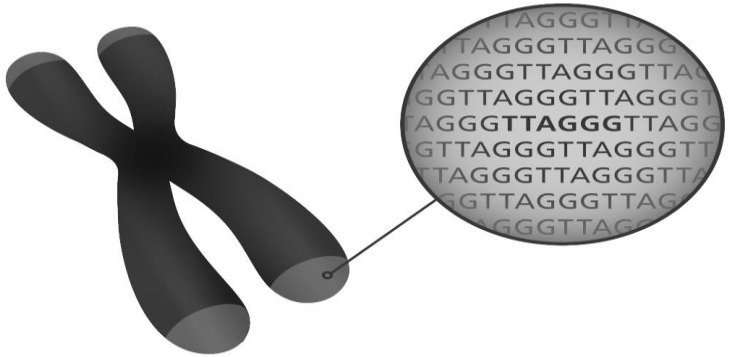
- A goal function



Genetic Algorithms

The main components of a GA are:

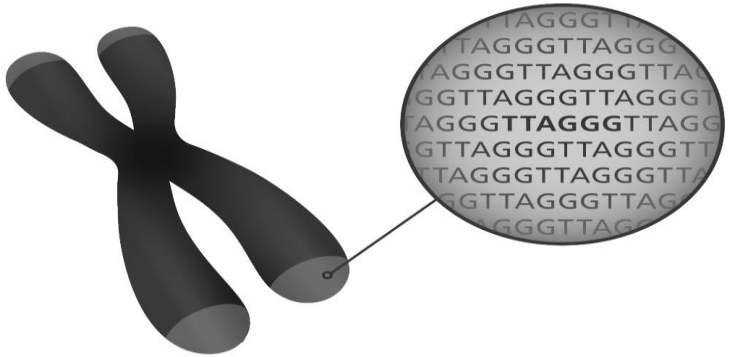
- A goal function
- A “population” of “individuals” or “chromosomes”, implying an individual with a single haploid chromosome. (As opposed to humans and other mammals that have multiple diploid chromosomes.)



Genetic Algorithms

The main components of a GA are:

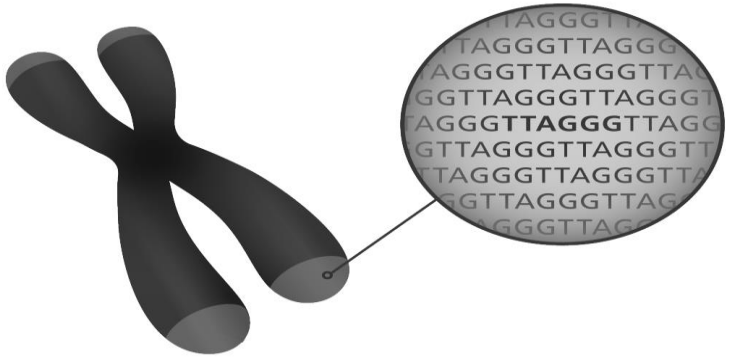
- A goal function
- A “population” of “individuals” or “chromosomes”, implying an individual with a single haploid chromosome. (As opposed to humans and other mammals that have multiple diploid chromosomes.)
- Successive Generations, during each of which we apply:



Genetic Algorithms

The main components of a GA are:

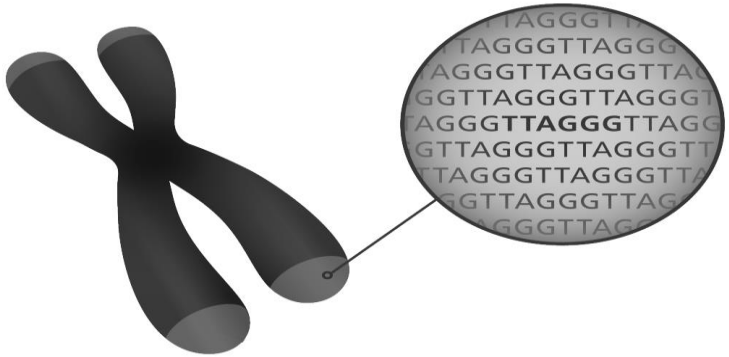
- A goal function
- A “population” of “individuals” or “chromosomes”, implying an individual with a single haploid chromosome. (As opposed to humans and other mammals that have multiple diploid chromosomes.)
- Successive Generations, during each of which we apply:
    - Selection



Genetic Algorithms

The main components of a GA are:

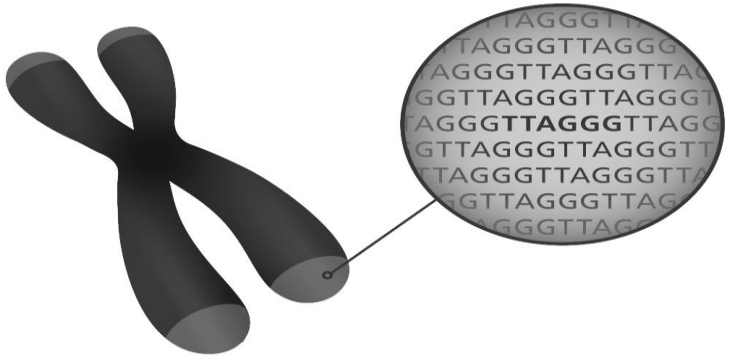
- A goal function
- A “population” of “individuals” or “chromosomes”, implying an individual with a single haploid chromosome. (As opposed to humans and other mammals that have multiple diploid chromosomes.)
- Successive Generations, during each of which we apply:
    - Selection
    - Crossover



Genetic Algorithms

The main components of a GA are:

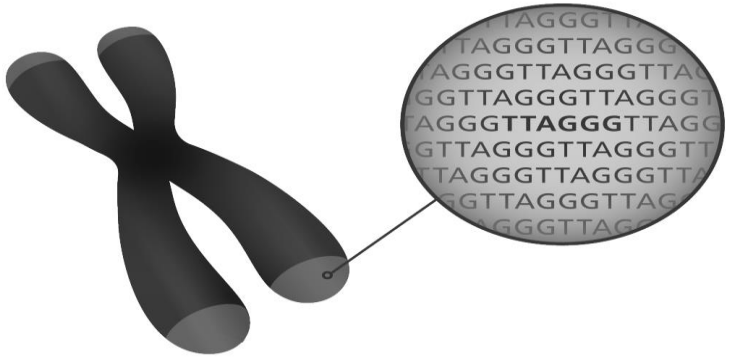
- A goal function
- A “population” of “individuals” or “chromosomes”, implying an individual with a single haploid chromosome. (As opposed to humans and other mammals that have multiple diploid chromosomes.)
- Successive Generations, during each of which we apply:
    - Selection
    - Crossover
    - Mutation



Genetic Algorithms

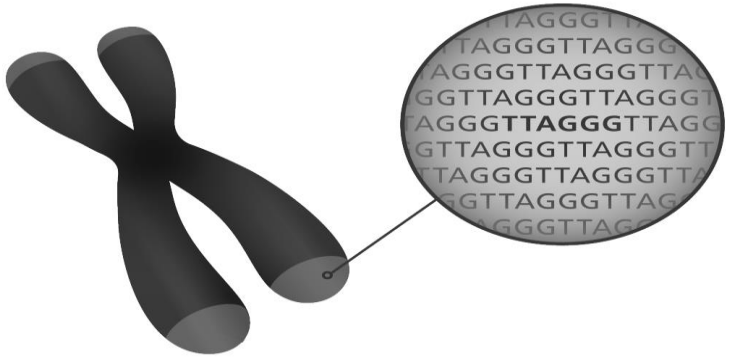
The main components of a GA are:

- A goal function
- A “population” of “individuals” or “chromosomes”. With GAs, we use a single haploid chromosome to represent the entire individual. (As opposed to humans and other mammals that have multiple diploid chromosomes.)
- Successive Generations, during each of which we apply:
    - Selection
    - Crossover
    - Mutation
- A stopping condition



Genetic Algorithms

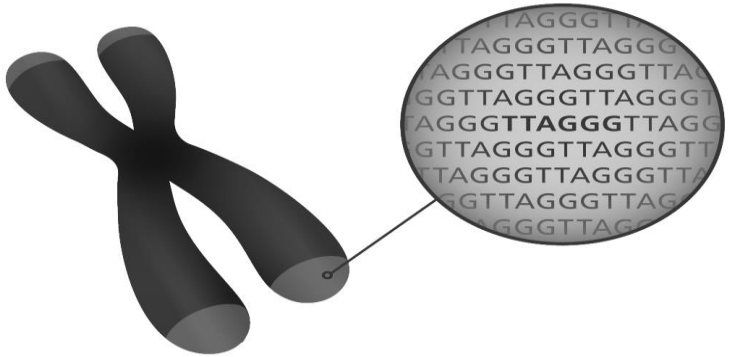
Common selection methods include:



Genetic Algorithms

Common selection methods include:

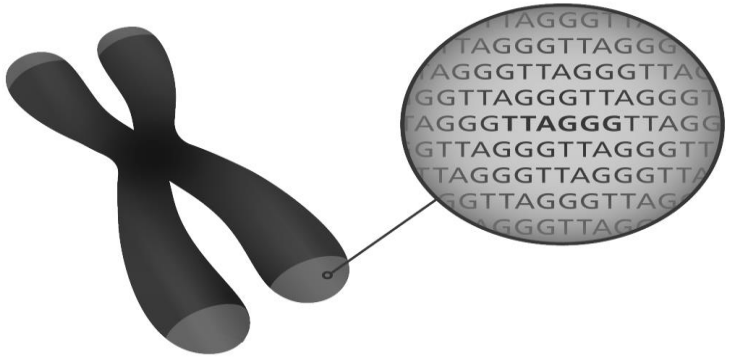
- Elitism



Genetic Algorithms

Common selection methods include:

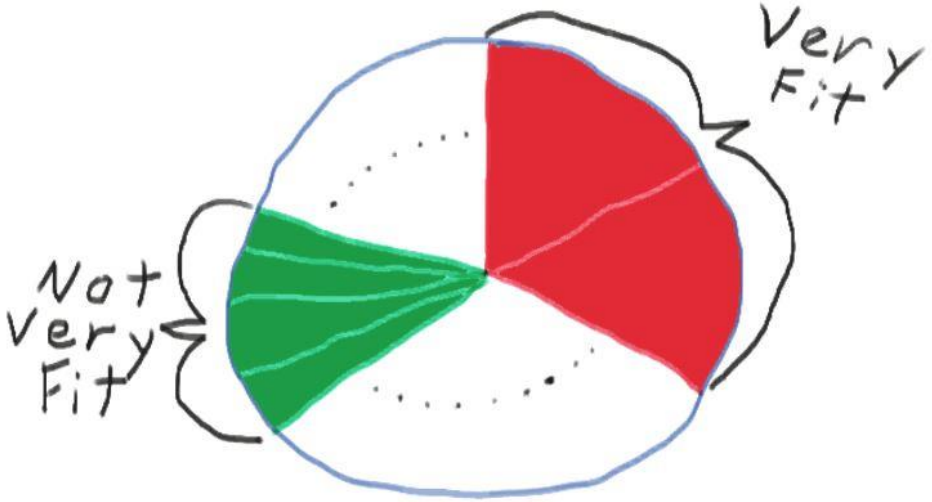
- Elitism
- (An unfair) Roulette Wheel

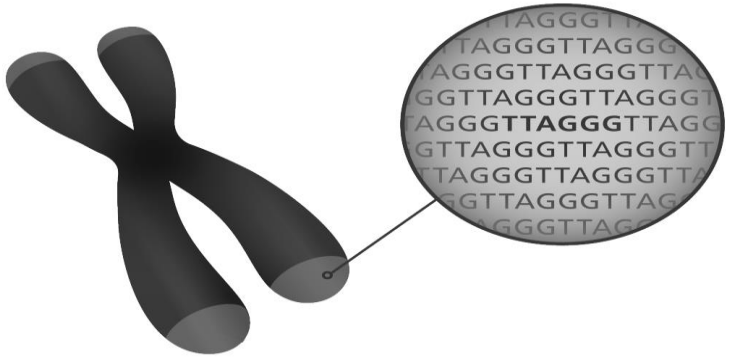


Genetic Algorithms

Common selection methods include:

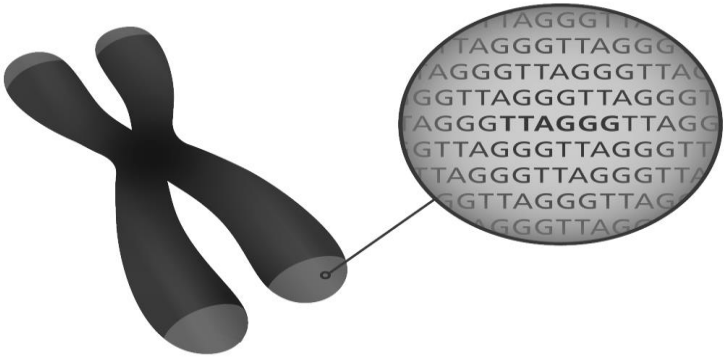
- Elitism
- (An unfair) Roulette Wheel





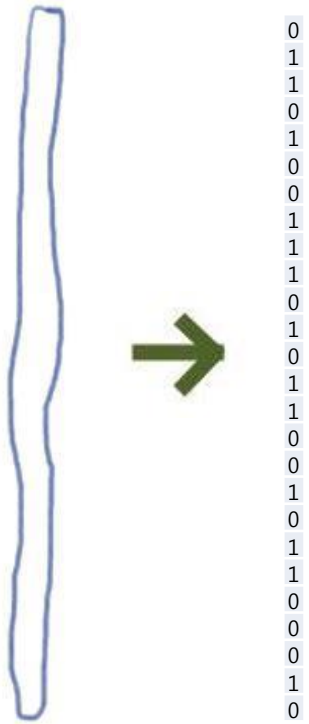
Genetic Algorithms

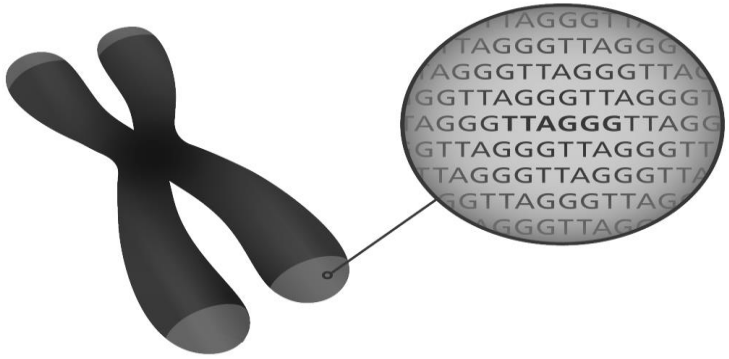
Traditional Chromosome Encoding



Genetic Algorithms

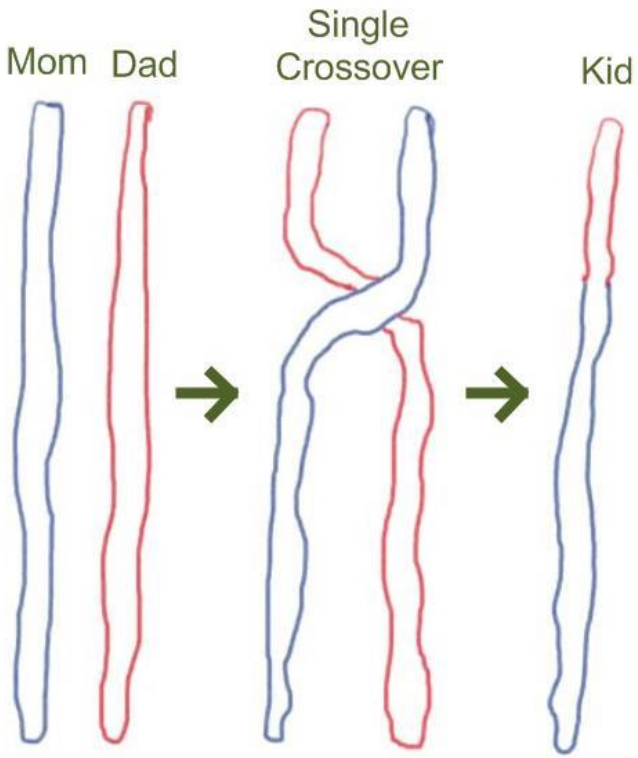
Traditional Chromosome Encoding

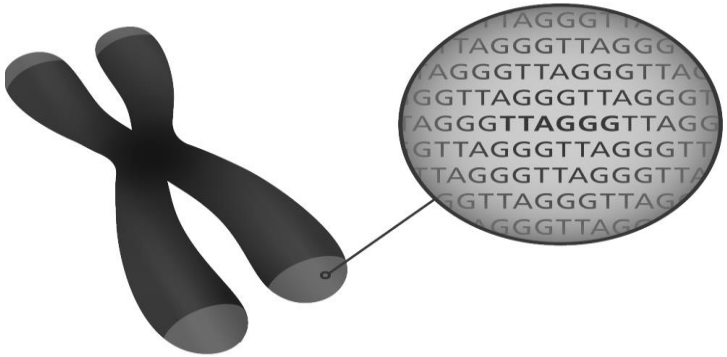




Genetic Algorithms

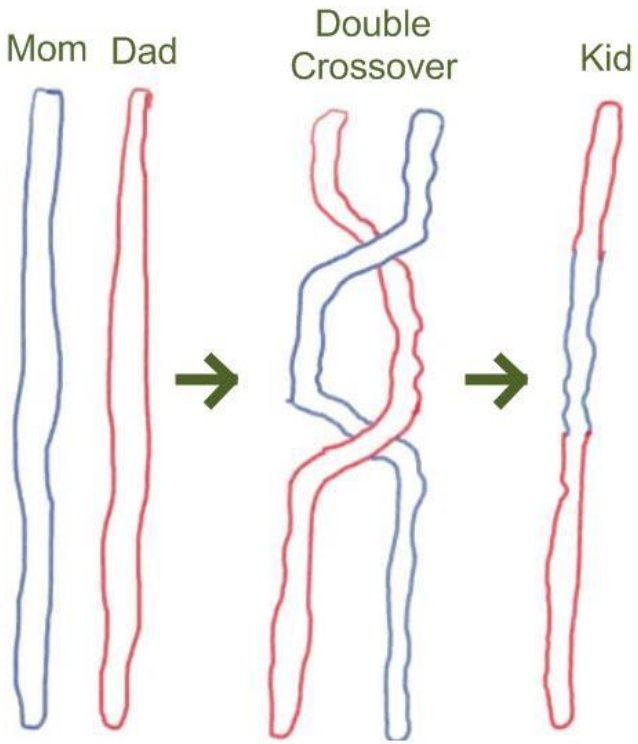
Traditional Single Point Crossover

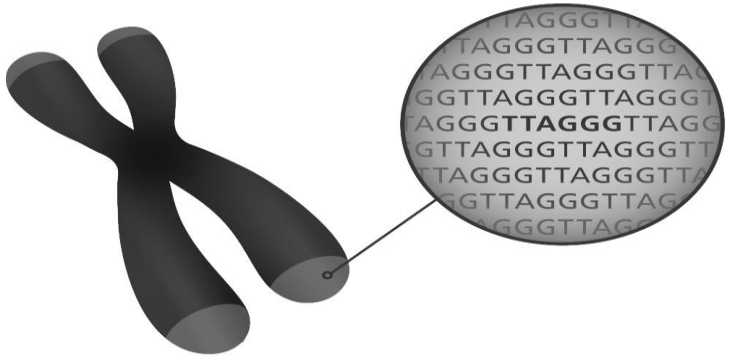




Genetic Algorithms

Traditional Double Point Crossover





Genetic Algorithms

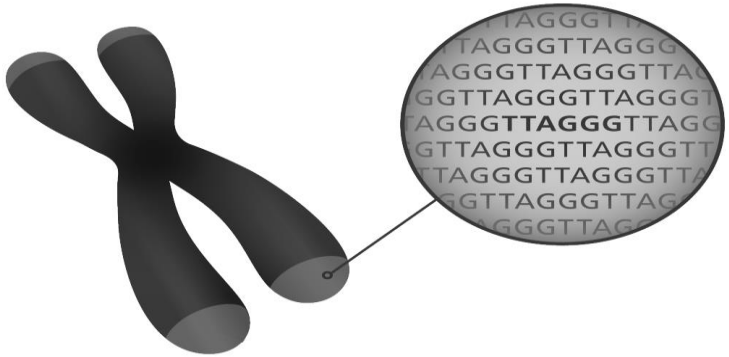
Mutation

- In traditional GA solution encoding, we have a long list of bits that represent a solution.
- So mutation is simple. Every once in a great while (the mutation rate), we just flip a bit.
- This has the effect of introducing new alleles that may not have been present yet in any individual.

al·lele
/əˈlēl/

noun GENETICS

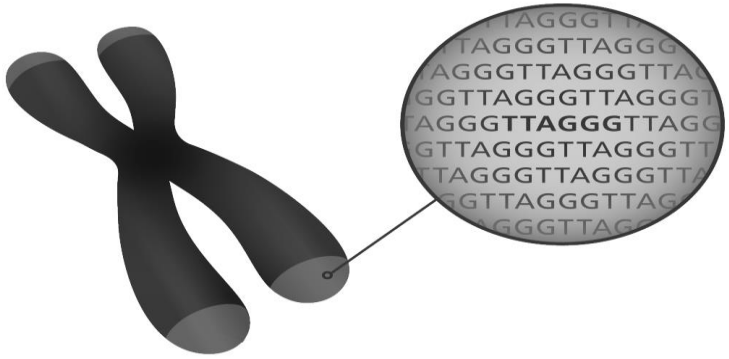
one of two or more alternative forms of a gene that arise by mutation and are found at the same place on a chromosome.



Genetic Algorithms

Traditional Single Crossover – Applied To Ordered Lists

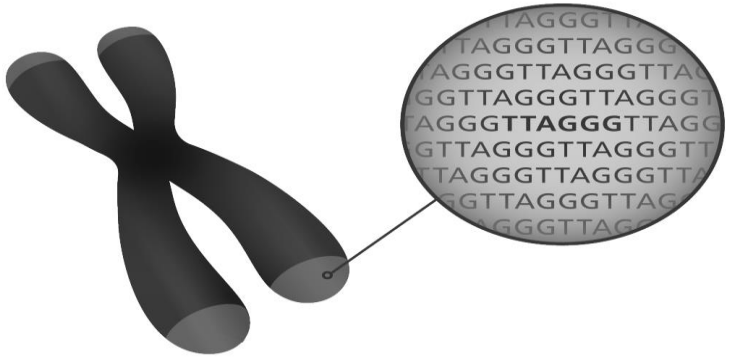
Mom	Dad
D	E
F	D
B	G
E	H
C	J
G	B
A	A
I	I
J	F
H	C



Genetic Algorithms

Traditional Single Crossover – Applied To Ordered Lists

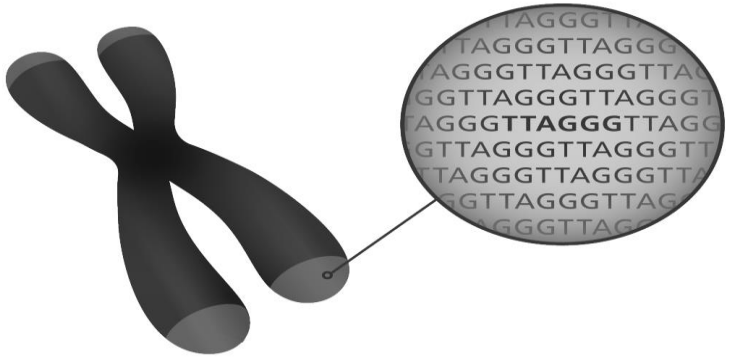
Mom	Dad	Kid	
D	E	D	
F	D	**F**	
B	G	**B**	**E ?**
E	H	H	**G ?**
C	J	J	
G	B	**B**	
A	A	A	
I	I	I	
J	F	**F**	
H	C	C	



Genetic Algorithms

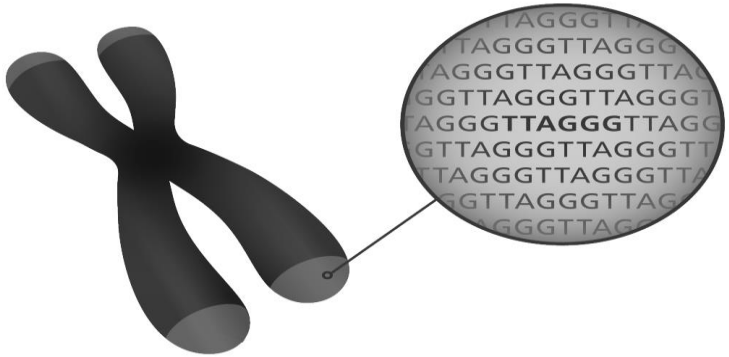
Traditional Mutation – Applied To Ordered Lists

D
F
B ← ??
E
C
~~G~~ **B**
A
I
J
H



Genetic Algorithms

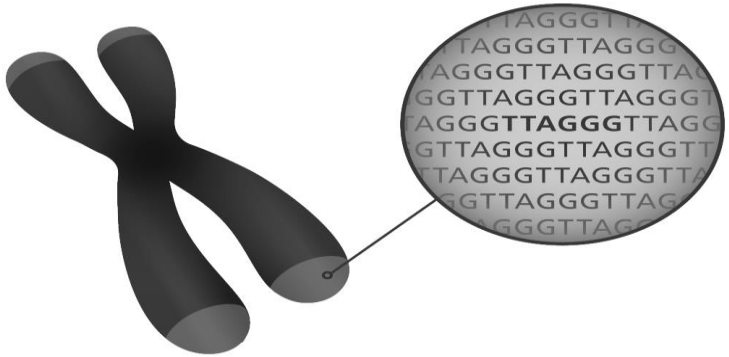
Crossover for the TSP – The Edge Recombination Operator (ERO)



Genetic Algorithms

Crossover for the TSP – The Edge Recombination Operator (ERO)

- First published in 1989
- Experimentally shown superior to the other methods for preserving the ordered list constraint of a TSP solution in 1999.

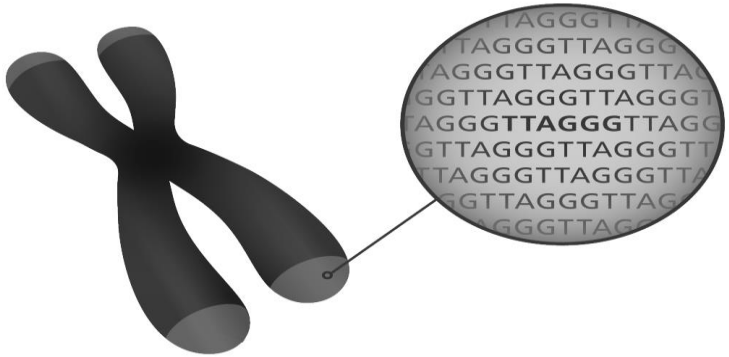


Genetic Algorithms

Crossover for the TSP – The Edge Recombination Operator (ERO)

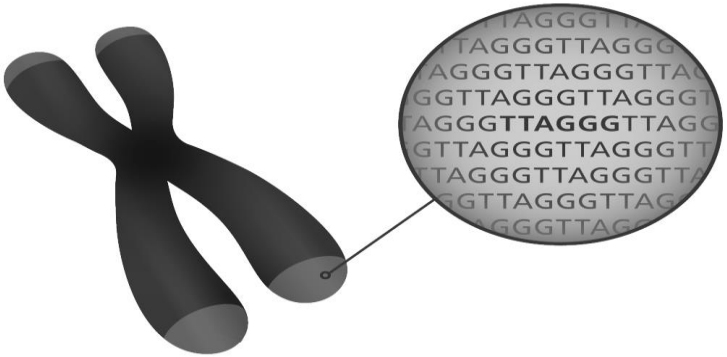
Unfortunately it has a complexity class O(n^2), whereas traditional crossover had a constant O(1) complexity class.

(In both cases I'm discounting the overhead of copying data from the complexity calculation).



Genetic Algorithms

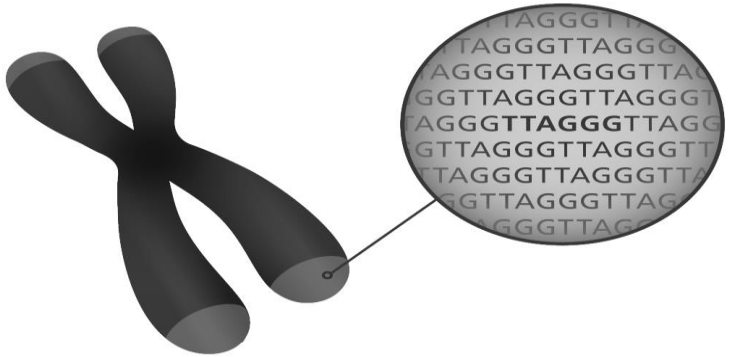
So are GAs a good way to solve the TSP?



Genetic Algorithms

So are GAs a good way to solve the TSP?

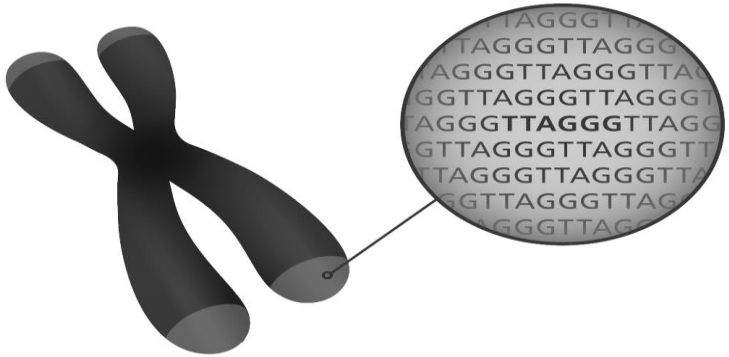
- Encoding solutions is simple and space efficient



Genetic Algorithms

So are GAs a good way to solve the TSP?

- Encoding solutions is simple and space efficient
- Evaluating fitness, selection, and mutation are as computationally efficient as in any “traditional” GA



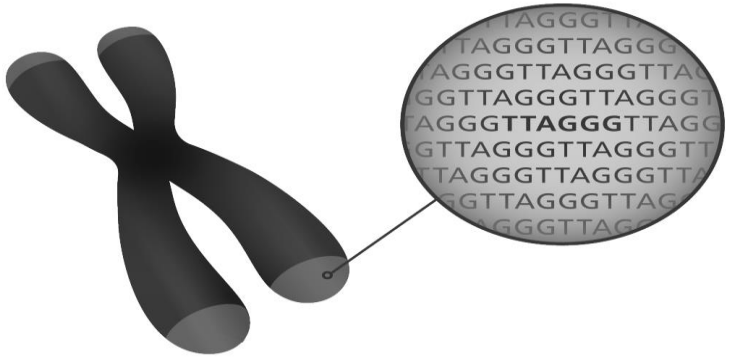
Genetic Algorithms

So are GAs a good way to solve the TSP?

- Encoding solutions is simple and space efficient
- Evaluating fitness, selection, and mutation are as computationally efficient as in any “traditional” GA

  --but--

- Crossover is relatively inefficient



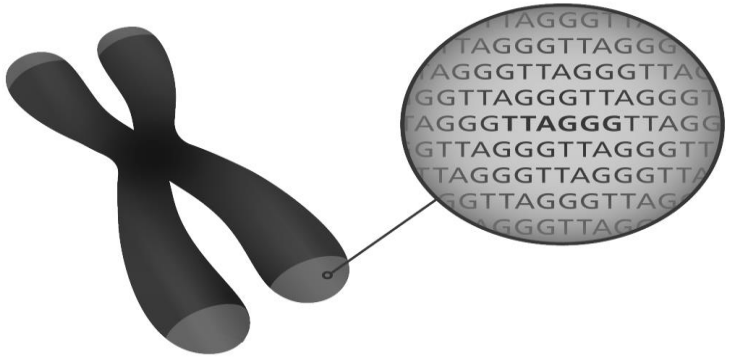
Genetic Algorithms

So are GAs a good way to solve the TSP?

- Encoding solutions is simple and space efficient
- Evaluating fitness, selection, and mutation are as computationally efficient as in any “traditional” GA

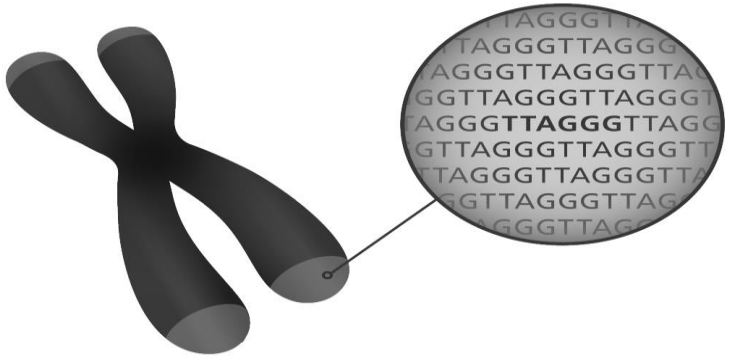
  --but--

- Crossover is relatively inefficient
- In general, maintaining the constraints of a viable solution to the TSP requires somewhat of a forced-fit onto the way GAs model natural selection.



Genetic Algorithms

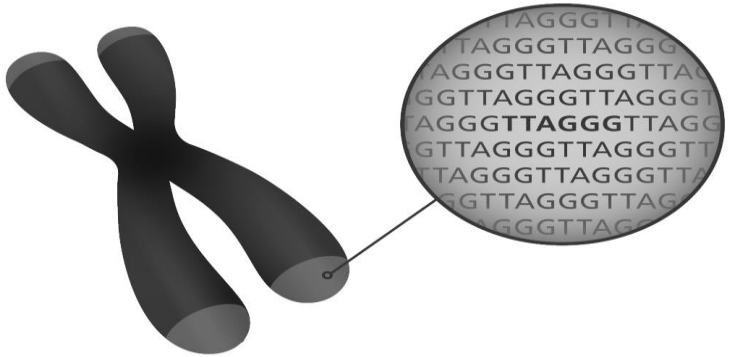
"Wow! Modelling natural processes is neat!"



Genetic Algorithms

"Wow! Modelling natural processes is neat!"

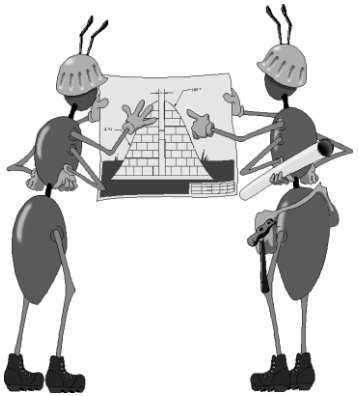
"Is there a natural process that is by its very nature more like the TSP that we could model?"



Ant Colony Optimization

Marco Dorigo

In his PhD thesis, in 1991, Marco Dorigo introduced the world to the first ant colony optimization algorithm, which he called “Ant System”.

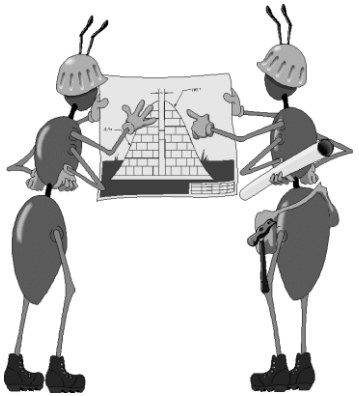


Ant Colony Optimization

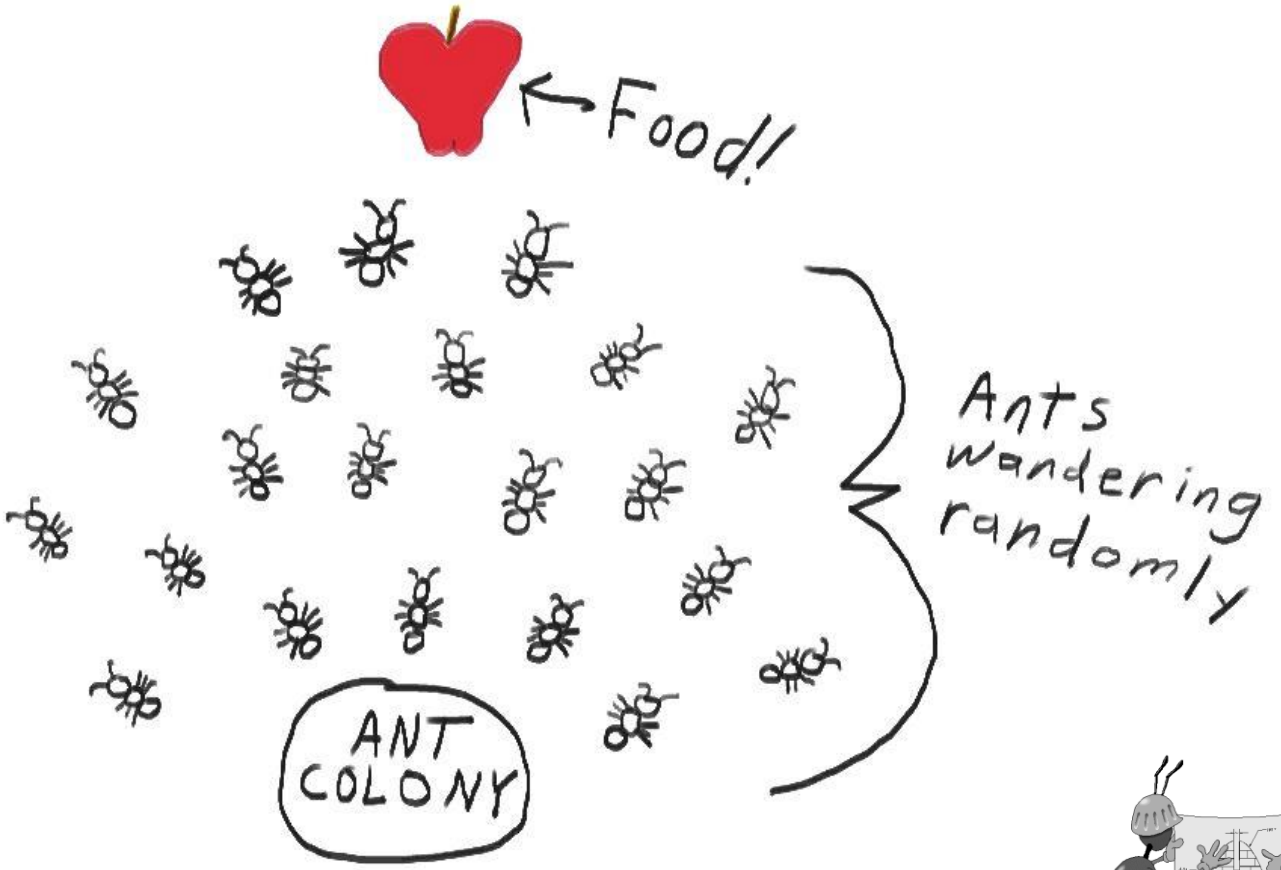
Marco Dorigo

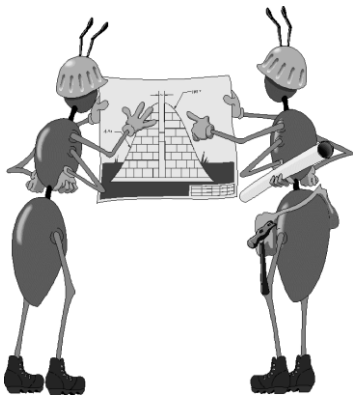
In his PhD thesis, in 1991, Marco Dorigo introduced the world to the first ant colony optimization algorithm, which he called “Ant System”.

From the very first paper on ACOs, they were being used to solve TSP instances.



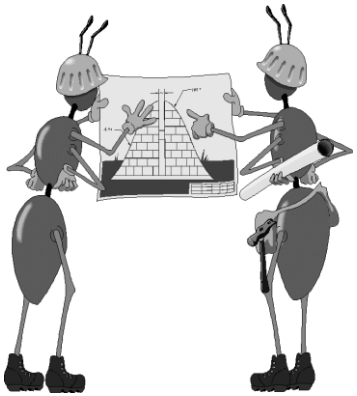
Ant Colony Optimization





Ant Colony Optimization

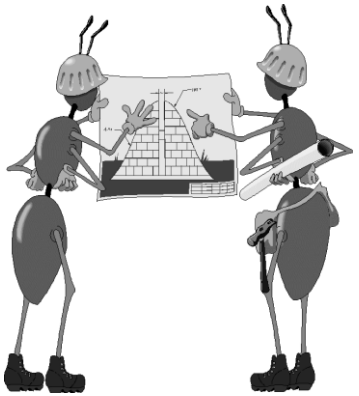
Similarities between GAs and ACOs



Ant Colony Optimization

Similarities between GAs and ACOs

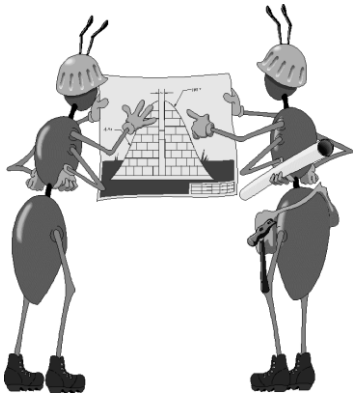
- Both use a population of individuals



Ant Colony Optimization

Similarities between GAs and ACOs

- Both use a population of individuals
- Both run over a series of generations (GA terminology) or iterations (ACO terminology)

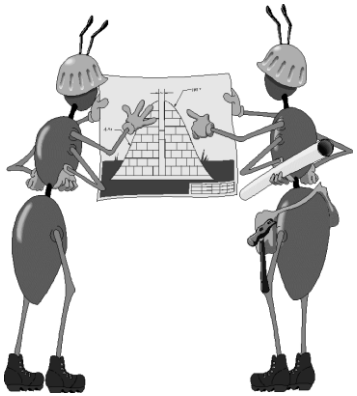


Ant Colony Optimization

Similarities between GAs and ACOs

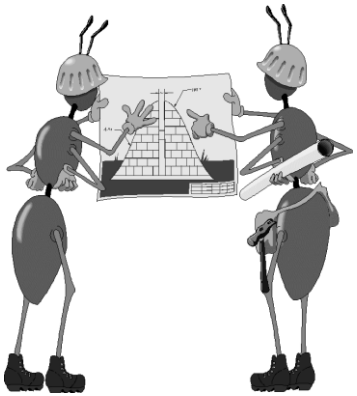
- Both use a population of individuals
- Both run over a series of generations (GA terminology) or iterations (ACO terminology)

In both bases, those numbers (size of population and number of generations/iterations) tend to be smaller for ACOs than GAs in order to achieve similar results.



Ant Colony Optimization

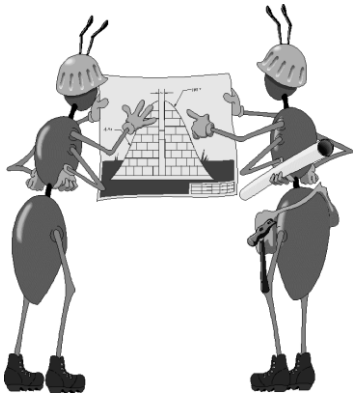
Differences between GAs and ACOs



Ant Colony Optimization

Differences between GAs and ACOs

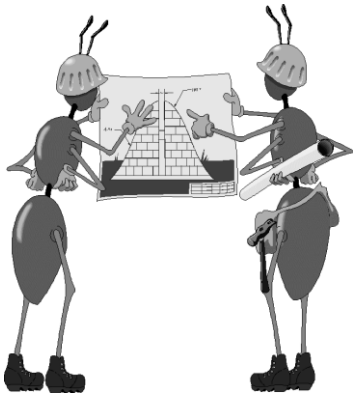
- The individuals (ants) in ACOs don't change thru the iterations …they are agents, not encoded solutions.



Ant Colony Optimization

Differences between GAs and ACOs

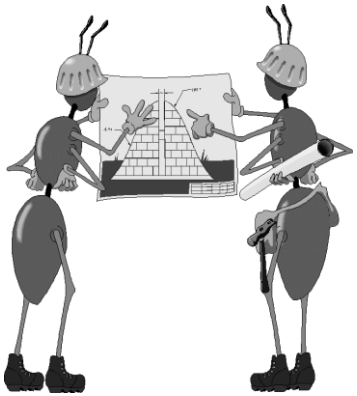
- The individuals (ants) in ACOs don't change thru the iterations …they are agents, not encoded solutions.
- The individuals are co-operating rather than competing



Ant Colony Optimization

Differences between GAs and ACOs

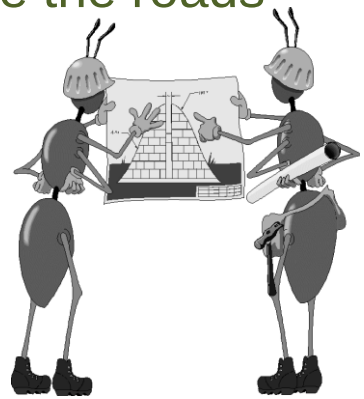
- The individuals (ants) in ACOs don't change thru the iterations …they are agents, not encoded solutions.
- The individuals are co-operating rather than competing
- The "intelligence" isn't encoded in the genes of the individuals but rather laid on the graph as pheromones



Ant Colony Optimization

Differences between GAs and ACOs

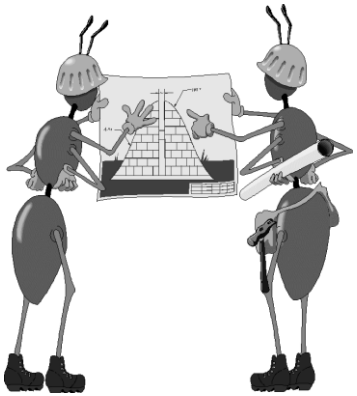
- The individuals (ants) in ACOs don't change thru the iterations …they are agents, not encoded solutions.
- The individuals are co-operating rather than competing
- The "intelligence" isn't encoded in the genes of the individuals but rather laid on the graph as pheromones
- ACOs are by their nature graph-based. Paths, edges and vertices are inherent. (The TSP is a graph problem where paths are routes, edges are the roads between cities and vertices are the cities.)



Ant Colony Optimization

ACOs At A High Level

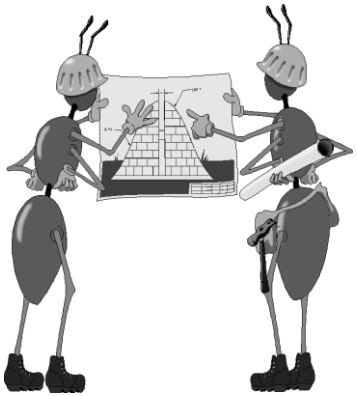
- We start with a small number of ants (typically 10-20).



Ant Colony Optimization

ACOs At A High Level

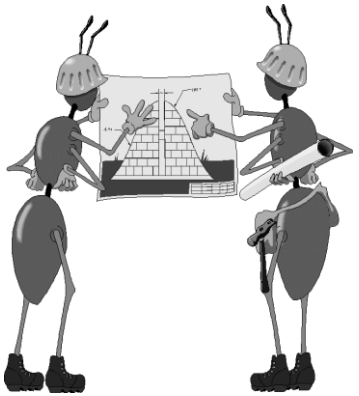
- We start with a small number of ants (typically 10-20).
- Step by step (city by city) they each probabilistically choose what edge to take (what city to go to next).



Ant Colony Optimization

ACOs At A High Level

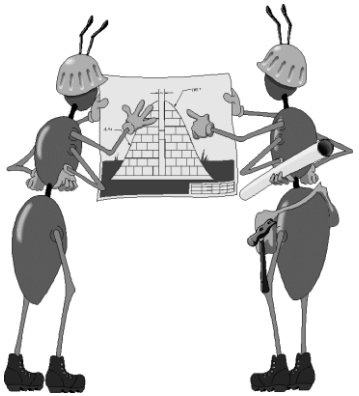
- We start with a small number of ants (typically 10-20).
- Step by step (city by city) they each probabilistically choose what edge to take (what city to go to next).
- After they've all created a tour, we:



Ant Colony Optimization

ACOs At A High Level

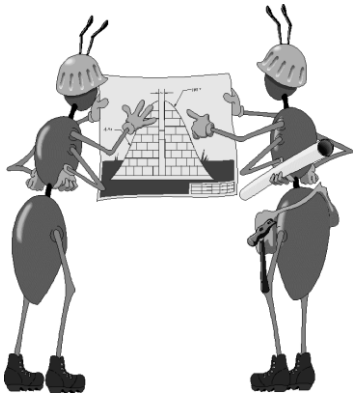
- We start with a small number of ants (typically 10-20).
- Step by step (city by city) they each probabilistically choose what edge to take (what city to go to next).
- After they've all created a tour, we:
    - Lay down pheromones along the edges each ant took.



Ant Colony Optimization

ACOs At A High Level

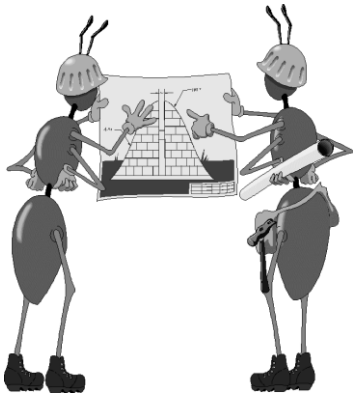
- We start with a small number of ants (typically 10-20).
- Step by step (city by city) they each probabilistically choose what edge to take (what city to go to next).
- After they've all created a tour, we:
  - Lay down pheromones along the edges each ant took.
  - Evaporate the pheromones.



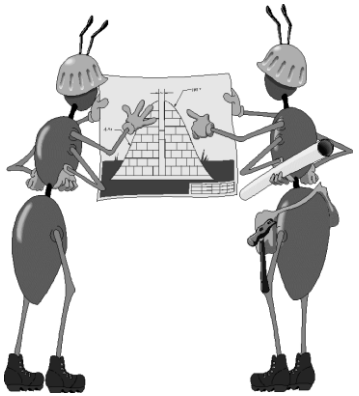
Ant Colony Optimization

ACOs At A High Level

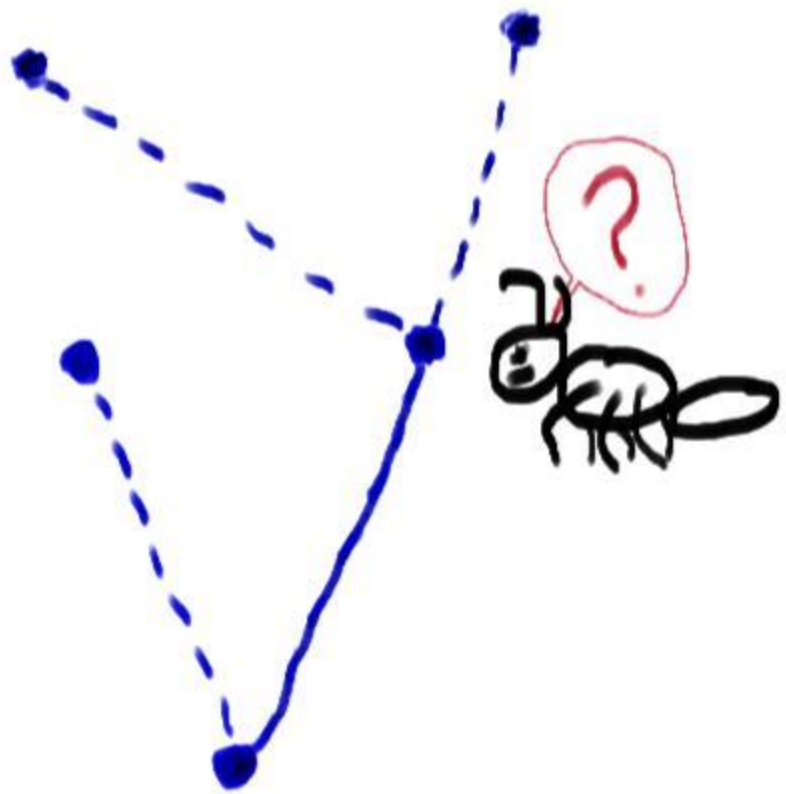
- We start with a small number of ants (typically 10-20).
- Step by step (city by city) they each probabilistically choose what edge to take (what city to go to next).
- After they've all created a tour, we:
  - Lay down pheromones along the edges each ant took.
  - Evaporate the pheromones.
- We repeat this process until we reach a stopping condition (often simply a number of iterations).

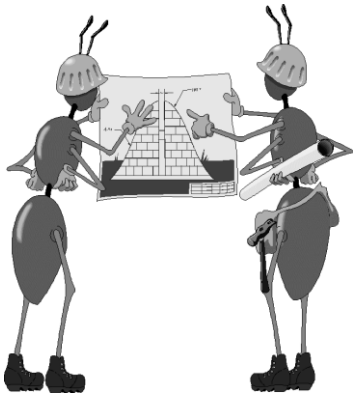


Ant Colony Optimization

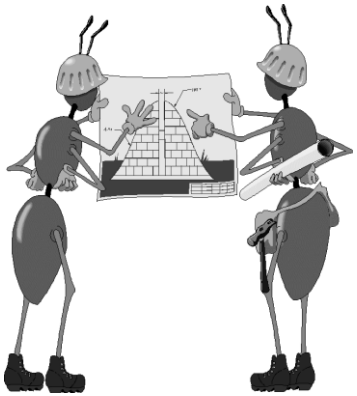


Ant Colony Optimization





Ant Colony Optimization

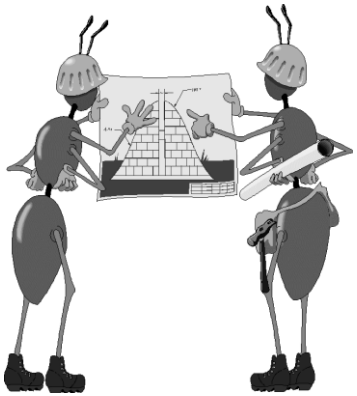


Ant Colony Optimization

Probability of choosing a remaining edge

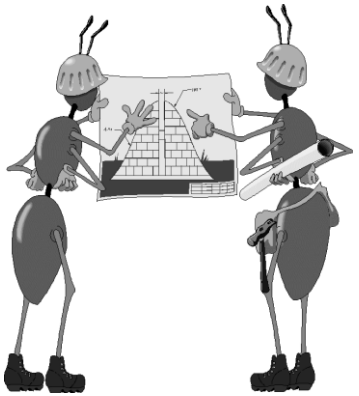
$$p_{xy}^{k} = \frac{(\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}{\sum_{y \in \text{allowed}_y}(\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}$$

(Equation 1)



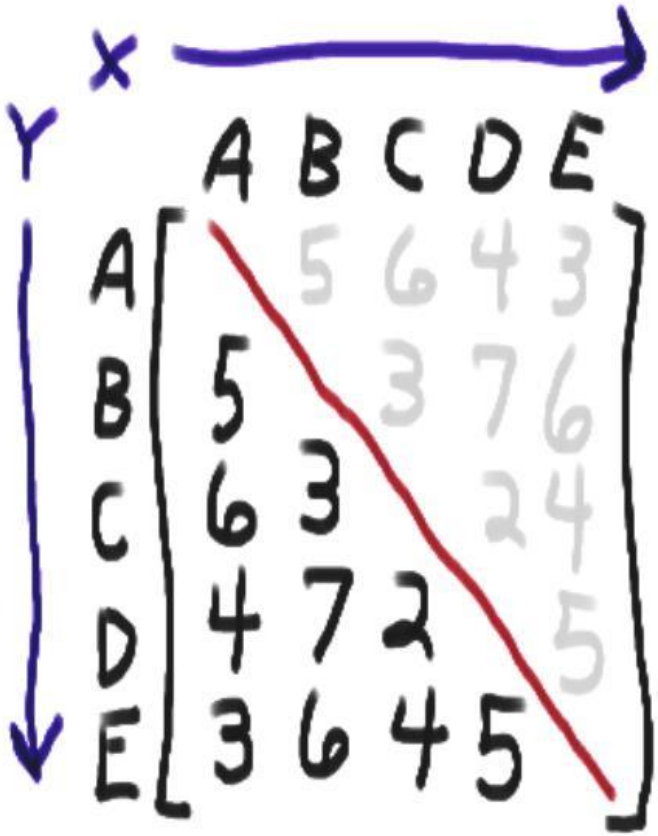
Ant Colony Optimization

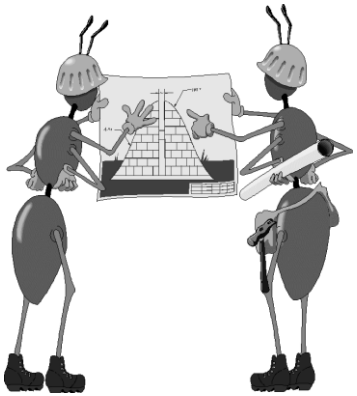
What would τ look like?



Ant Colony Optimization

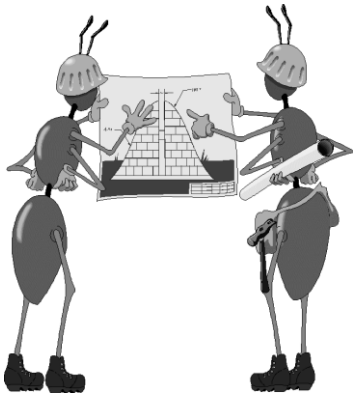
What would τ look like?





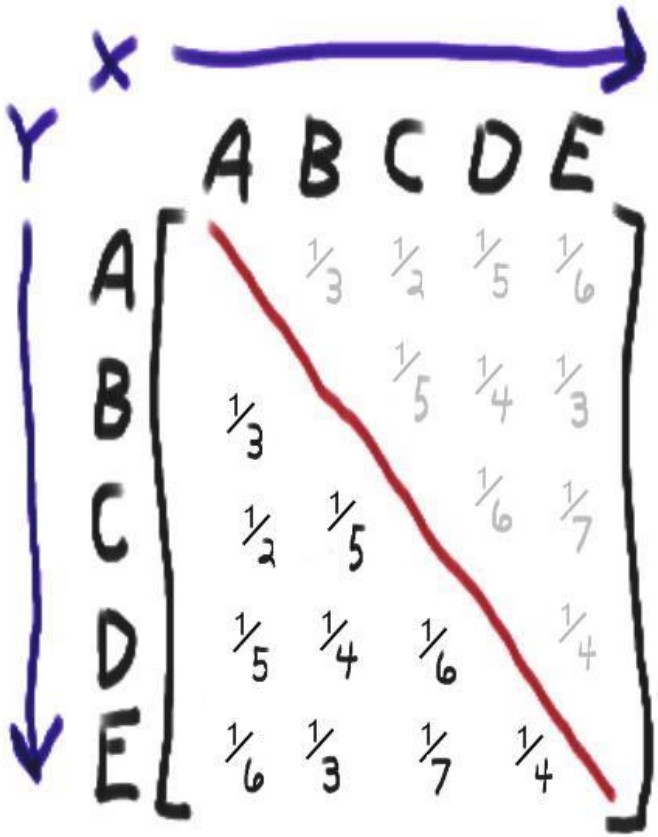
Ant Colony Optimization

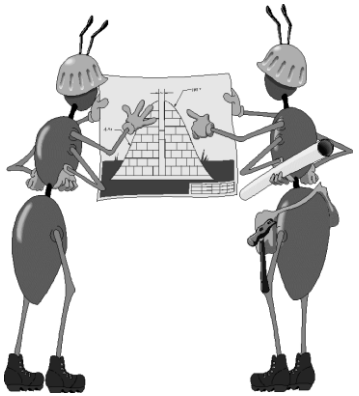
What would η look like?



Ant Colony Optimization

What would η look like?



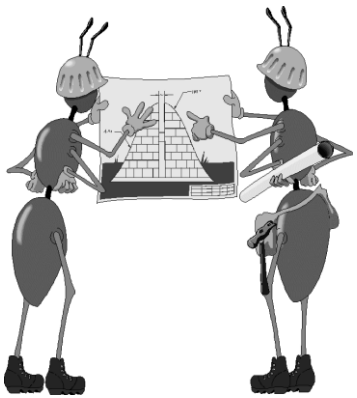


Ant Colony Optimization

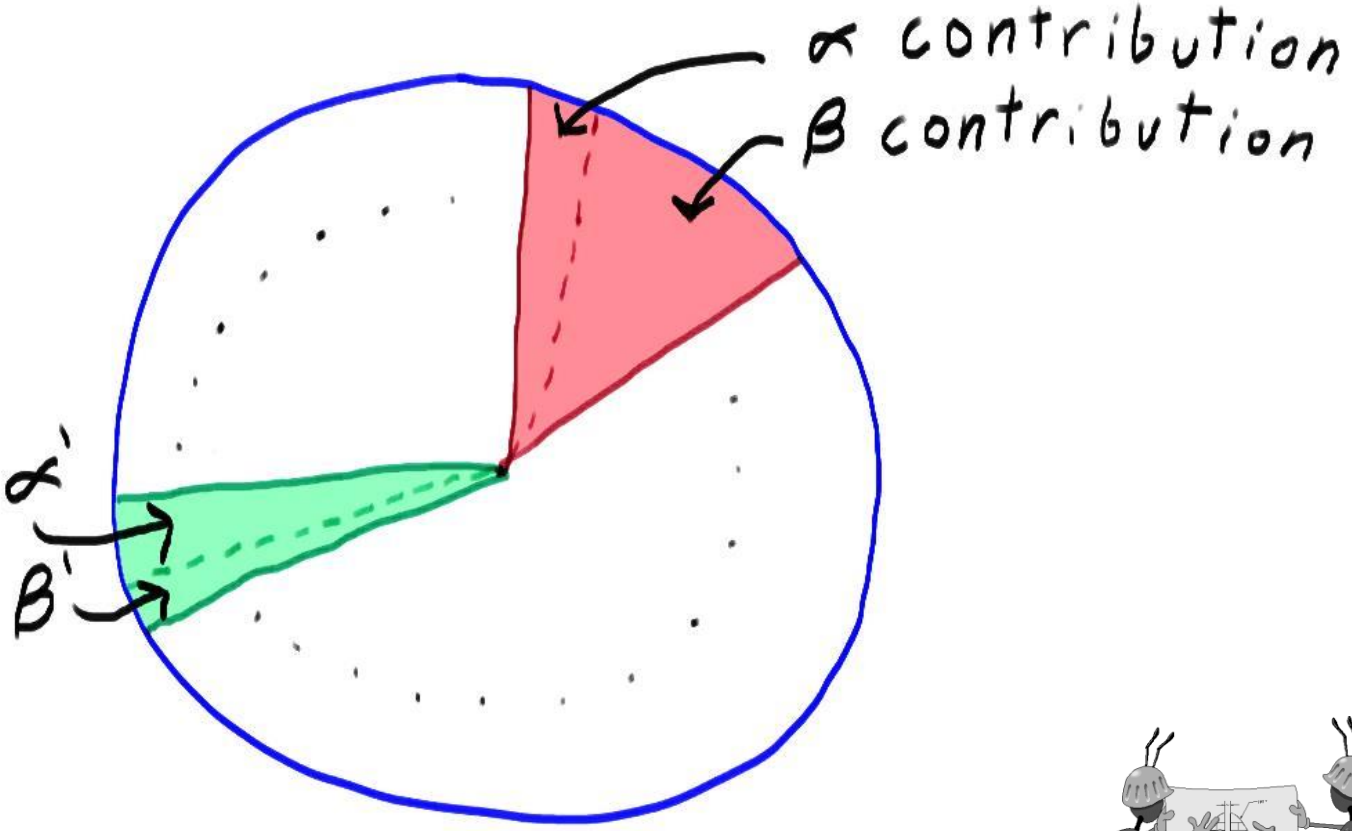
Probability of choosing a remaining edge

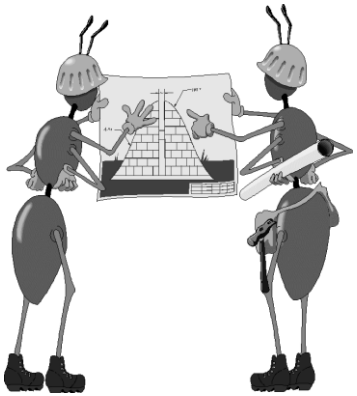
$$p_{xy}^{k} = \frac{\left[\tau_{xy}^{k}\right]^{\alpha}\left[\eta_{xy}^{k}\right]^{\beta}}{\text{The sum of the same calculation for ALL values in the same column.}}$$

(Equation 1)



Ant Colony Optimization





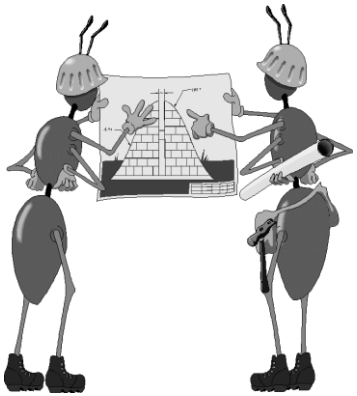
Ant Colony Optimization

Updating the pheromones along the edges

$$\tau_{xy} = \tau_{xy} + \sum_{k} \Delta\tau_{xy}^{k}$$

where $\Delta\tau_{xy}^{k} = \begin{cases} 1/L_k\,, & \text{if ant } k \text{ uses path segment } xy \\ 0, & \text{otherwise} \end{cases}$

(Equation 2)

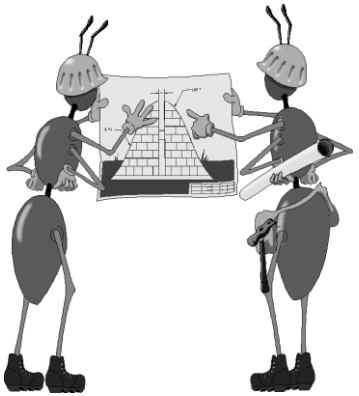


Ant Colony Optimization

Evaporating the pheromones

$$\tau_{xy} = \tau_{xy}(1 - \rho)$$

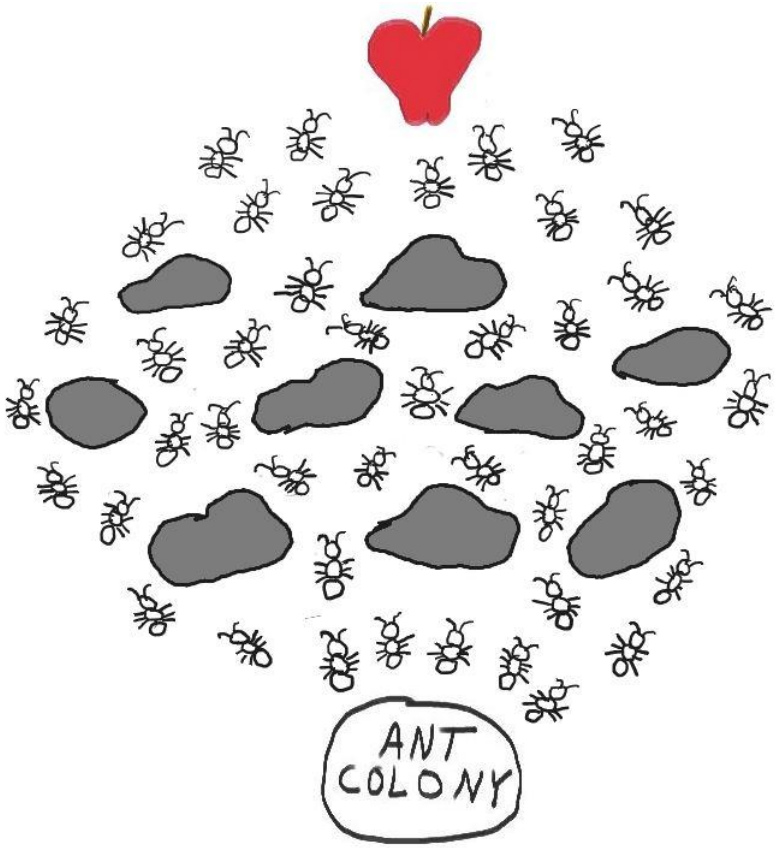
(Equation 3)

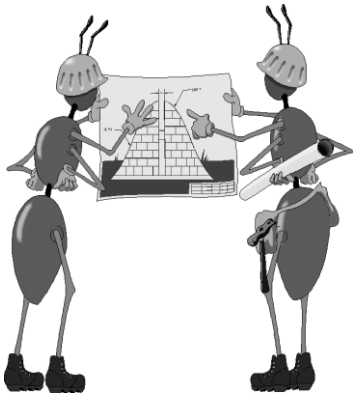


Ant Colony Optimization

Obstacles

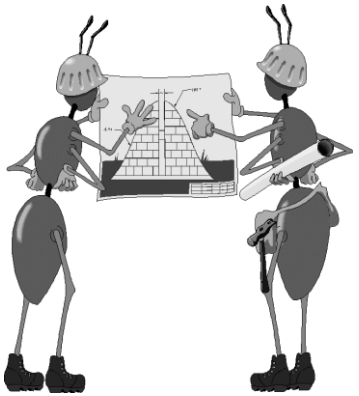
- What makes all of this non-trivial (i.e. not a straight line) and what allows for it to naturally model the TSP





Ant Colony Optimization

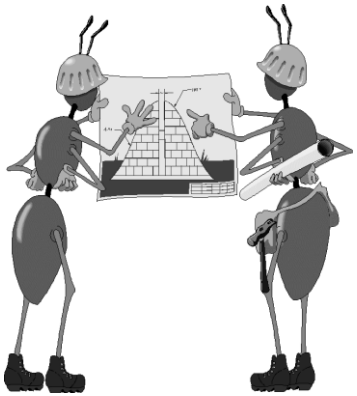
Extensions to ACO



Ant Colony Optimization

Extensions to ACO

- Ant Colony System (ACS) – introduced q0, which is a percentage (often 10%) chance, at any step, to bypass the probabilistic equation 1 for choosing the next city, and instead go directly to the city closest to the current one.

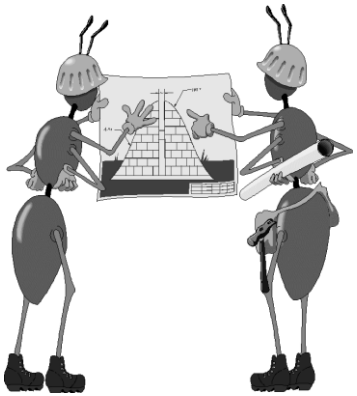


Ant Colony Optimization

Extensions to ACO

- Ant Colony System (ACS) – introduced q0, which is a percentage (often 10%) chance, at any step, to bypass the probabilistic equation 1 for choosing the next city, and instead go directly to the city closest to the current one.
- Max-Min Ant System (MMAS) – introduced maximum and minimum values for the pheromone trails along all edges.

(both introduced in 1996)

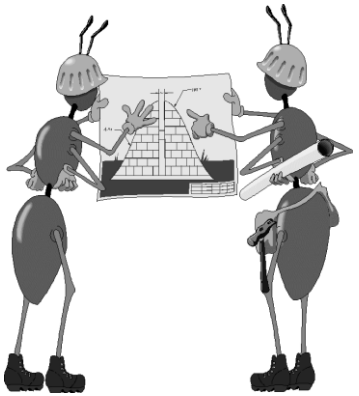


Ant Colony Optimization

Interestingly…

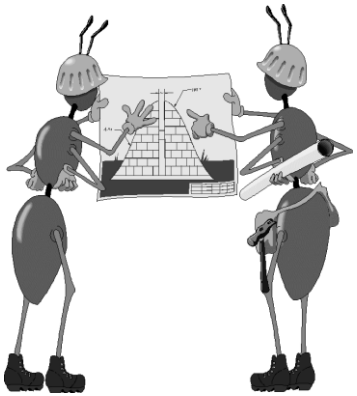
Theoretically, but not practically at this point, in 2002 it was proven that certain ACO algorithms (including the ACS and MMAS extensions) will eventually converge to the true optimal value.

Unfortunately, there is no way to predict how long that will take (yet)…



Ant Colony Optimization

Questions?



Thanks!

Adam Byerly
adam@rokrol.com
(309) 369-9257