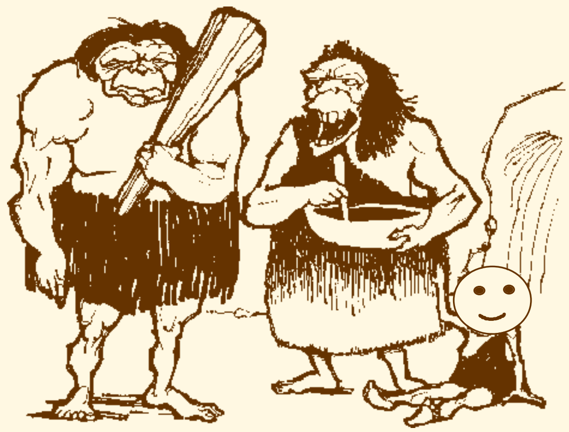
Adaptation



Adaptive agent

An agent that can respond to its environment.

Four primary ways of adapting:

- **reaction** - a direct, predetermined response to a particular event or an environmental signal.
- **reasoning** - ability to make inferences.
- **learning** - change that occurs during the lifetime of an agent.
- **evolution** - change in a population that occurs over successive generations of agents.

Adapting by reaction

A direct, predetermined response to a particular event or environmental signal.

Typically expressed in the form

*"**WHEN** event, **IF** condition(s), **THEN** action"*

Examples:

- ❑ thermostats
- ❑ robotic sensors that can detect the presence of a nearby wall and activate a device for avoiding it
- ❑ washing machines and vacuum cleaners that use fuzzy logic

Adapting by reasoning

A reactive response that uses inference rules.

A more advanced form of reactive adaptation using a set of rules to infer.

Typically *chains* of rules in the form—

*"**WHEN** event, **IF** condition(s), **THEN** action"*

Typical kinds of techniques:

- ❑ patient diagnosis
- ❑ bulletin board or web foraging agents
- ❑ data mining

Adapting by learning

Change that occurs during the lifetime of an agent.

Typical kinds of techniques:

- ❑ credit assignment
- ❑ Bayesian (or probabilistic) rules
- ❑ neural networks
- ❑ classifier rules
- ❑ exploration of internal representations
- ❑ Q learning

Adapting by evolution

Change that occurs over successive generations of agents.

Typical kinds of strategies:

- ❑ Natural & artificial selection (*i.e.,* "survival of the fittest")
- ❑ Darwinian *vs.* Lamarckian evolution (*e.g.,* genotype and phenotype)
- ❑ Differentiation into ecosystem roles and community formation
- ❑ Competition (*e.g.,* increasing returns)
- ❑ Cooperation (*e.g.,* multiagent composition)
- ❑ Coevolution & arms races
- ❑ Cultural transmission (*e.g.,* Richard Dawkins' "memes", the Baldwin effect)

Four primary ways of adapting

- ❑ Reaction (minimum requirement)
- ❑ Reasoning
- ❑ Learning
- ❑ Evolution

} Any or all in combination (*e.g.* learning Bayesian inference webs using memes and genes...)

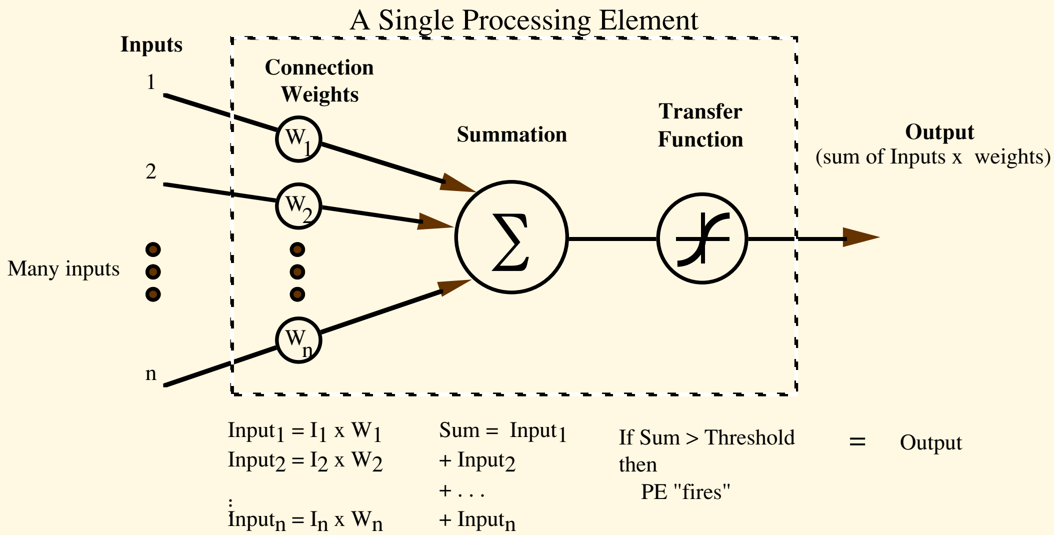
Some common adaptive learning and evolution methods

- ❑ Neural networks
- ❑ Genetic algorithms
- ❑ Genetic programming
- ❑ Ant colony optimization

(all in insufficient detail…)

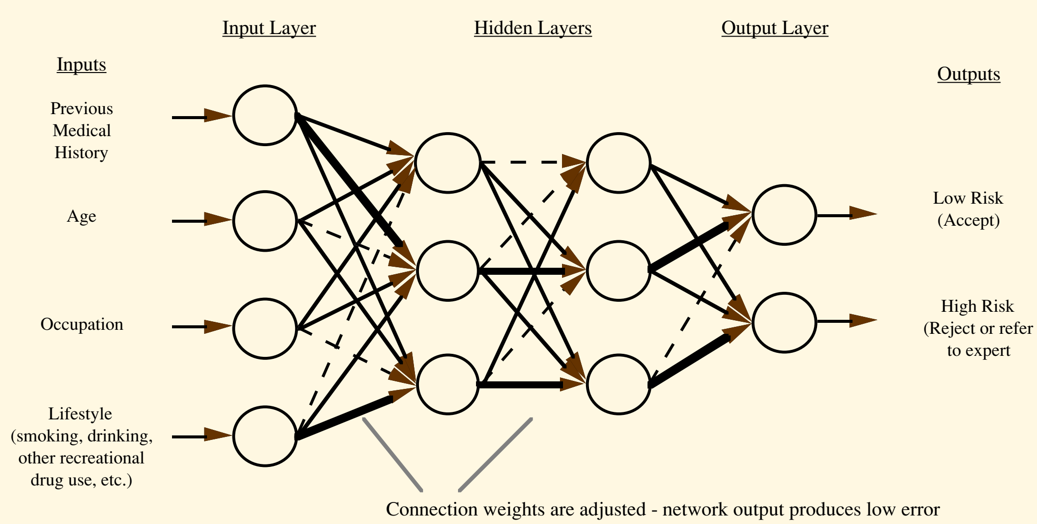
Neural networks

A network algorithm that identifies patterns using neuron-based techniques.



Neural networks

Insurance company example



Neural networks

Some of the challenges:

- ❑ Steep learning curve
- ❑ No real development methodology
- ❑ Demanding preprocessing requirements
- ❑ Integration issues
- ❑ Interpretability of results

Genetic algorithms

Algorithms based on biogenetics and the principles of Darwinian evolution

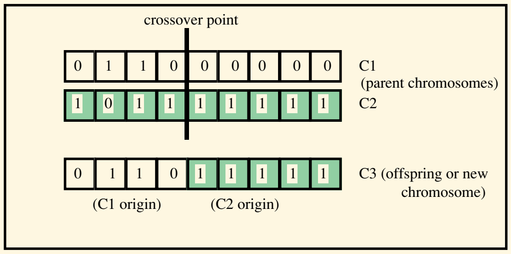
GAs are parallel stochastic search algorithms that

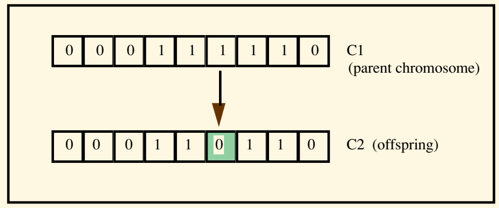
- ❑ represent possible solutions as strings (chromosomes) comprised of input values (genes).
- ❑ create consecutive generations of solutions using operators inspired by real genetics—mainly crossover and mutation—in order to create new variation in the population.
- ❑ evaluate each solution (individual) based on fitness criteria determined by the specific problem being solved (the environment).
- ❑ use the fitness information to favor the best individuals (strings) when creating new generations.

Genetic algorithms

Two of the most common operations:

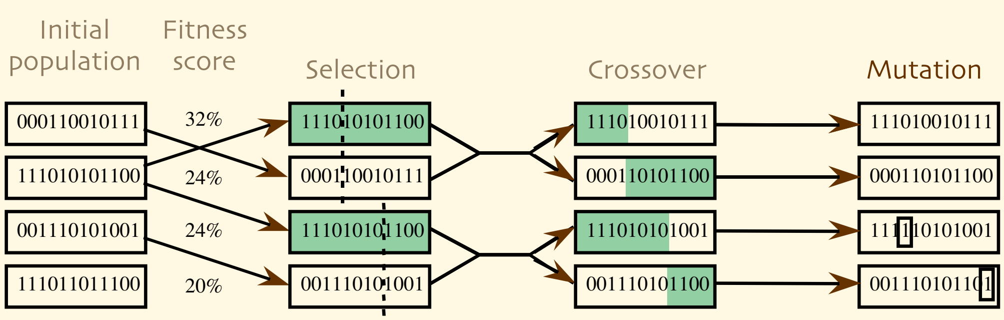
Crossover and mutation.





Others include operations such as inversion, deletion, and duplication.

Genetic algorithms



Example using steps in

genetic algorithm cycle

Genetic algorithms

Some of the challenges:

- Problem solving is concurrent (multiagent)
- Can result in counterintuitive solutions
- Unplanned results can be scary
- Is processing-intensive

Other evolutionary algorithms

- Evolution strategies (ES)
- Evolutionary programming (EP)
- GANNs & cellular encoding
- Evolvable hardware
- Genetic programming

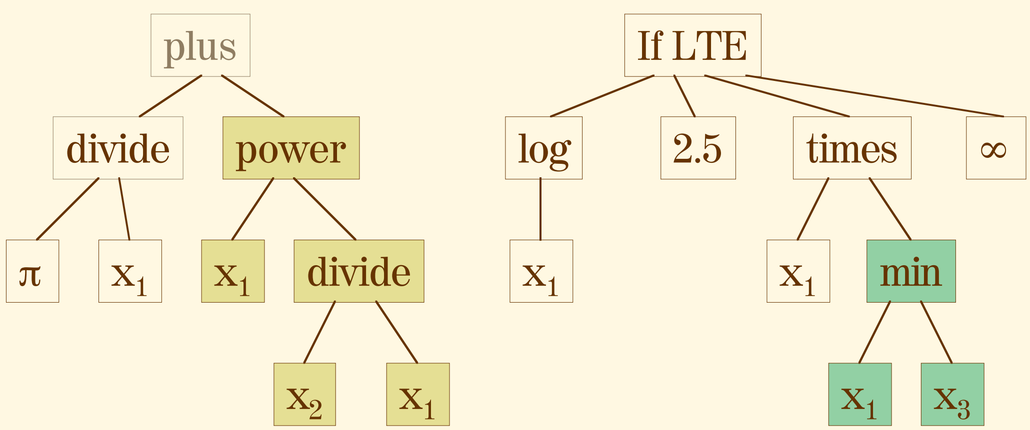
Genetic programming

Algorithm production based on the principles of genetic algorithms.

GP is a population-based evolutionary algorithm in which

- the desired solution is a function or algorithm itself, not just a set of numerical values.
- a *non-brittle* language is used to represent individual prospective solutions in the population.
- crossover, mutation, and the other search operators occur between programs, not just swapping values but exchanging arbitrary snippets of code.

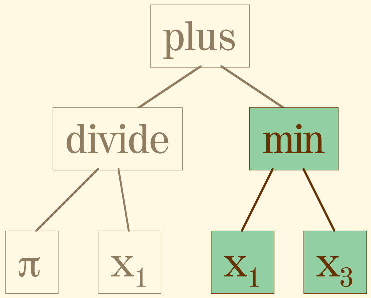
Genetic programming

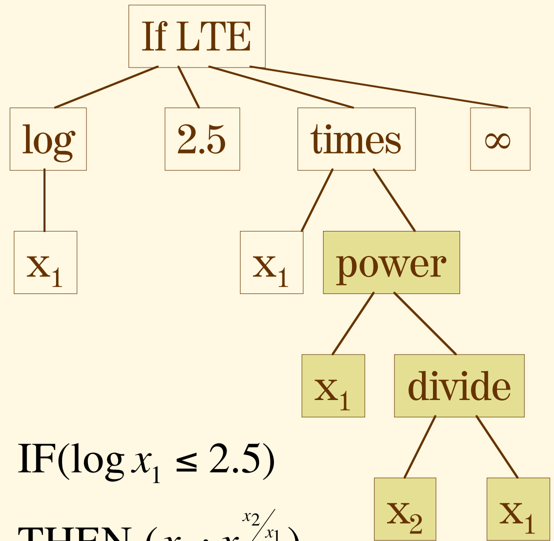


$$\frac{\pi}{x_1} + x_1^{x_2/x_1}$$

$$\text{IF}(\log x_1 \leq 2.5)$$
$$\text{THEN } (x_1 \cdot \min(x_1, x_3))$$
$$\text{ELSE } (\infty)$$

Genetic programming





$$\frac{\pi}{x_1} + \min(x_1, x_3)$$

$\mathrm{IF}(\log x_1 \leq 2.5)$

$\mathrm{THEN}\ (x_1 \cdot x_1^{x_2/x_1})$

$\mathrm{ELSE}\ (\infty)$

Genetic programming

Familiar formula:

$$ax^2 + bx + c = 0 \qquad x = \frac{-b \pm \sqrt{(b^2 - 4ac)}}{2a}$$

Given: `T = {A, B, C}` (3 coefficients) `F = {+, -, *, %, SQRT}` (function set)

Generated code after 30 generations:

```
(LIST2 (% (- (% B (+ A A)) (% (SQRT (% (* (* (- (SQRT B)
          (SQRT B)) (* (+ (- C A) A) (% A B))) (* (SQRT
          (% B B)) C(* -3))) (% (SQRT A) (SQRT (SQRT (SQRT
          A))) (% (SQRT -1) (* C -3))) (- B B)))) (* A
          A)) (* (- (% (% (- (SQRT (% (* (* (-
          (SQRT (% 3 -4)) (SQRT B)) (* (SQRT -1) (% A
          B))) (* (SQRT (% B B)) (* (- -2 C) (- B B))))
          (* A A))) (% (SQRT (SQRT (* (- B B) B))) (* C
          -3))) (- C A)) (SQRT -1)) (% (SQRT A) (% (SQRT
          (% (* (* (- (SQRT B) (SQRT B)) (* (SQRT -1) (%
          A B))) (* (SQRT (% B B)) (* (- -2 C) (- B
          B)))) (* A A))) (* C -3)))))
       (% (SQRT (+ (+ B -5) (+ (- (* (- (+ B -5) 1) (%
          (+ 1 B) 4)) (+ (* C A) (- (+ (% (- B B) 4) -5)
          1))) (% (+ 1 (+ 1 B)) 4)))) A)).
```

Genetic programming

Some of the challenges:

- Problem solving is concurrent (for multiagents)
- Can create novel, counterintuitive solutions
- Can result in uninterpretable structures—*even more easily than other evolutionary algorithms*
- Uninterpretable algorithms can be *very* scary
- Is *extraordinarily* processing-intensive

Swarm Intelligence

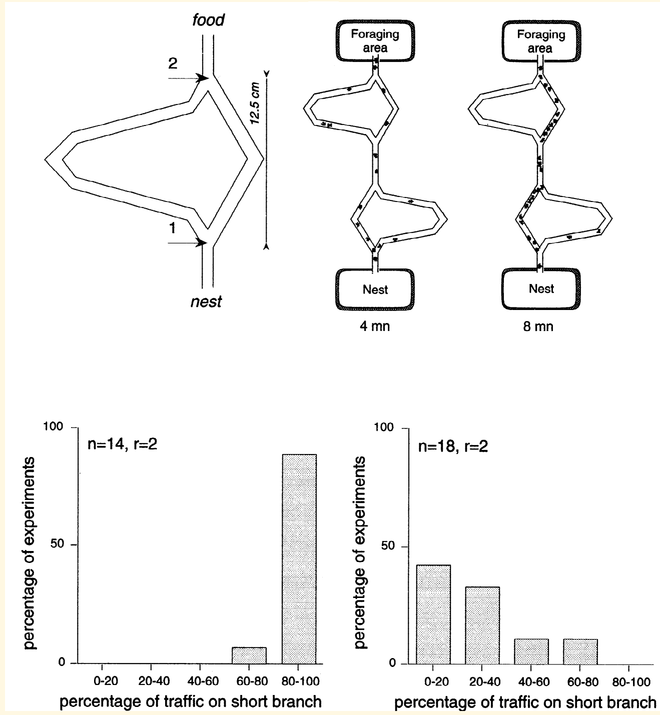
Technique of using the models of social insect behavior to design complex sysems

This social insect analogy for problem solving has become a hot topic in the last few years.

- It is inspired by the mechanisms that generate collective behavior in insects.
- Individual behavior modifies the environment, which in turn modifies the behavior of other individuals
- It emphasizes distributedness, indirect interaction among agents, flexibility, and robustness

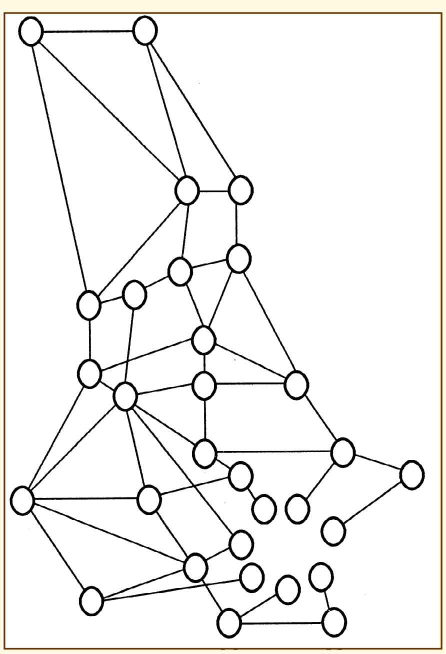
Swarm intelligence methods

Experimental setup and drawings of the short brach selection of the *Linepithema humile* ant



Swarm intelligence methods

Network connection application

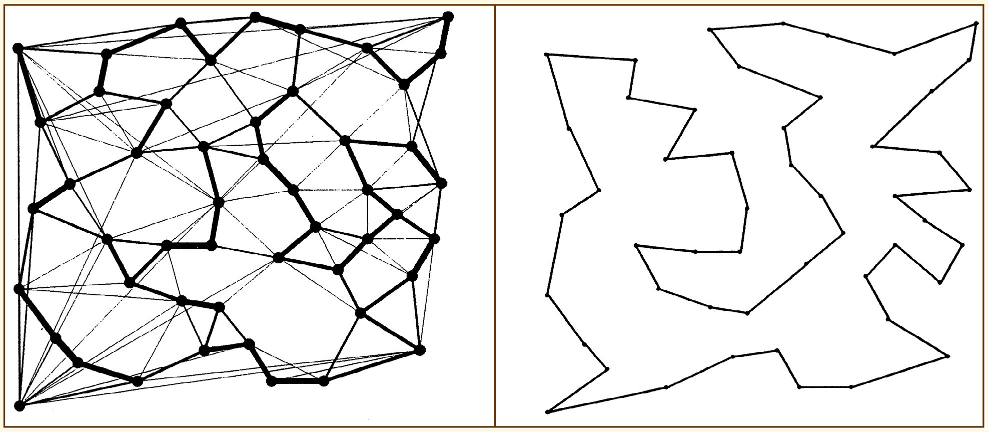


	Average call failures	Std. Dev.
Shortest path	12.53	2.04
Mobile agents	9.24	0.80
Improved mobile agents	4.41	0.85
ABC without noise	2.72	1.24
ABC with noise	2.56	1.05

Interconnection structure of the British Telecom SDH network

Swarm intelligence methods

Traveling salesman application

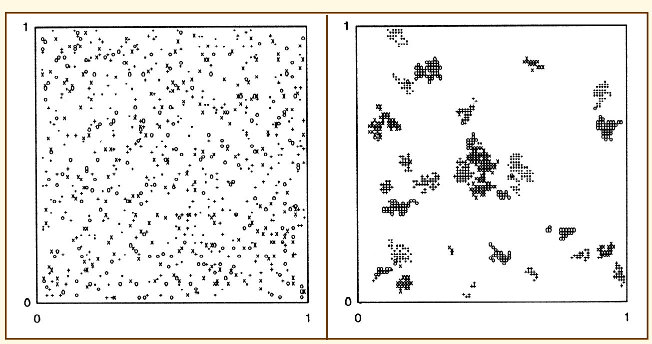


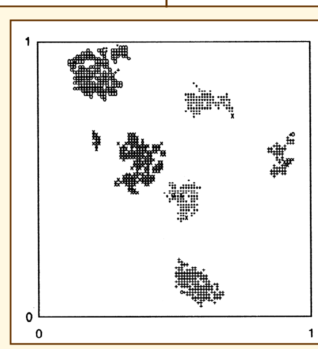
Pheromone concetration   Best solution found by ants

An example of a trail configuration found in a 50-city problem

Swarm intelligence methods

Sorting application

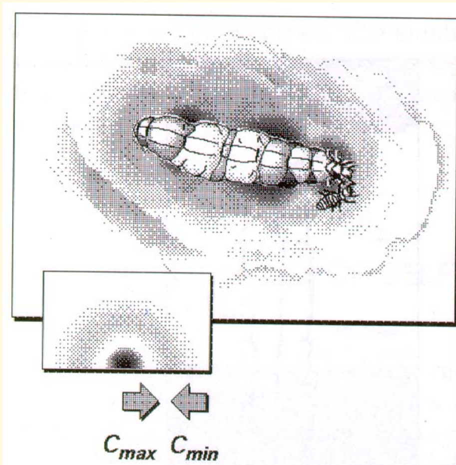


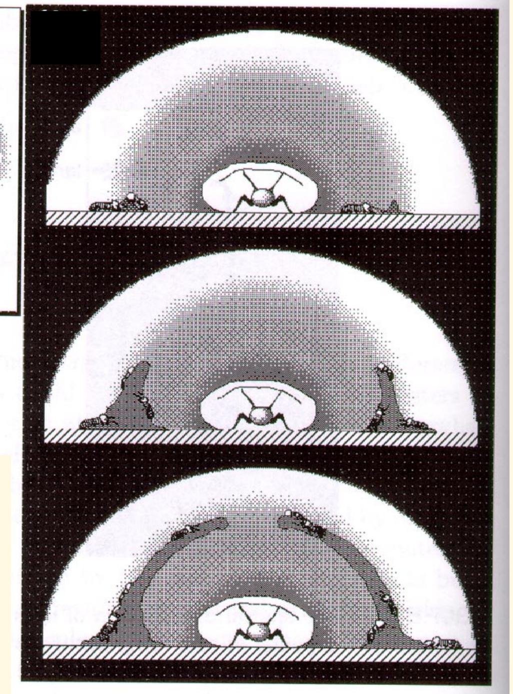


An example of sorting seven kinds of items (800 items total)

Swarm intelligence methods

Pheromonal template application





When the pheromones of an Acantholepsis custodiens queen are detected, workers respond with template-based behavior.

Swarm intelligence methods

Some of the challenges:

- ❑ Problem solving is concurrent and massively parallel
- ❑ Poor at finding novel, counterintuitive solutions
- ❑ Must hand-design internal ant dynamics and population size
- ❑ Mapping from ant algorithm to emergent dynamics is poorly understood

Bonabeau, Eric, Marco Dorigo and Guy Theraulaz (1999) *Swarm Intelligence: From Natural to Artificial Systems*, Oxford University Press, New York.