CSE 590
Data Science Fundamentals

Optimization Methods

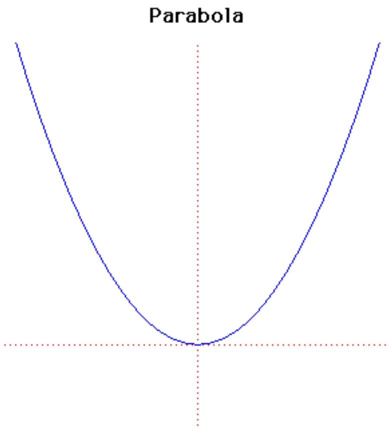
Klaus Mueller

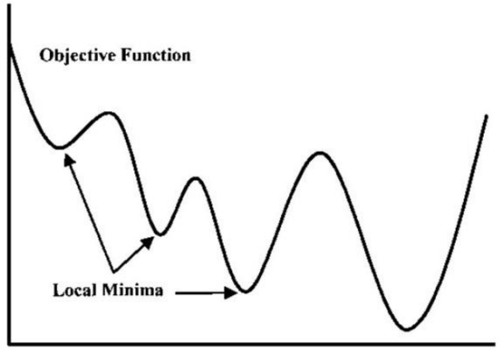
Computer Science Department
Stony Brook University and SUNY Korea

Lecture	Topic	Projects
1	Intro, schedule, and logistics	
2	Data Science components and tasks	
3	Data types	Project #1 out
4	Introduction to R, statistics foundations	
5	Introduction to D3, visual analytics	
6	Data preparation and reduction	
7	Data preparation and reduction	Project #1 due
8	Similarity and distances	Project #2 out
9	Similarity and distances	
10	Cluster analysis	
11	Cluster analysis	
12	Pattern mining	Project #2 due
13	Pattern mining	
14	Outlier analysis	
15	Outlier analysis	Final Project proposal due
16	Classifiers	
17	Midterm	
18	Classifiers	
19	Optimization and model fitting	
20	Optimization and model fitting	
21	Causal modeling	
22	Streaming data	Final Project preliminary report due
23	Text data	
24	Time series data	
25	Graph data	
26	Scalability and data engineering	
27	Data journalism	
	Final project presentation	Final Project slides and final report due

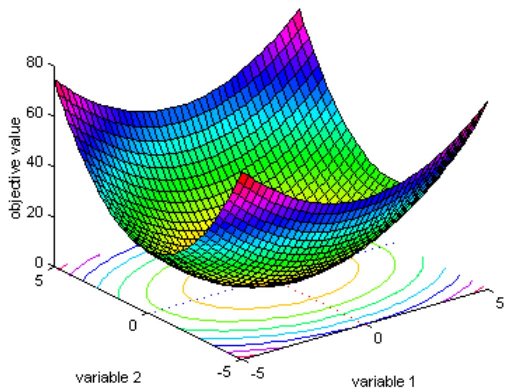
The Objective Function

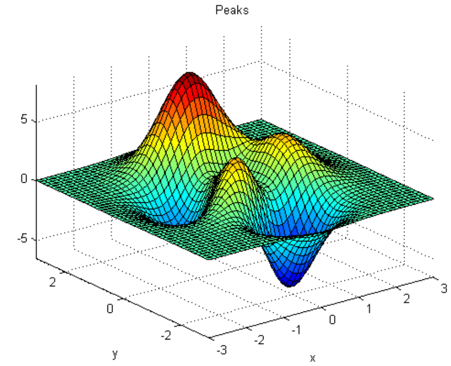
Minimize or maximize some criterion



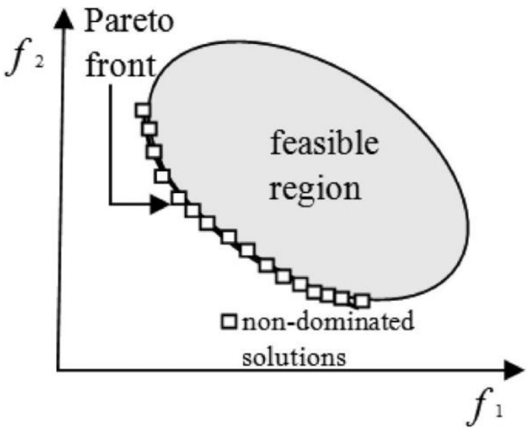


single objective, 1D



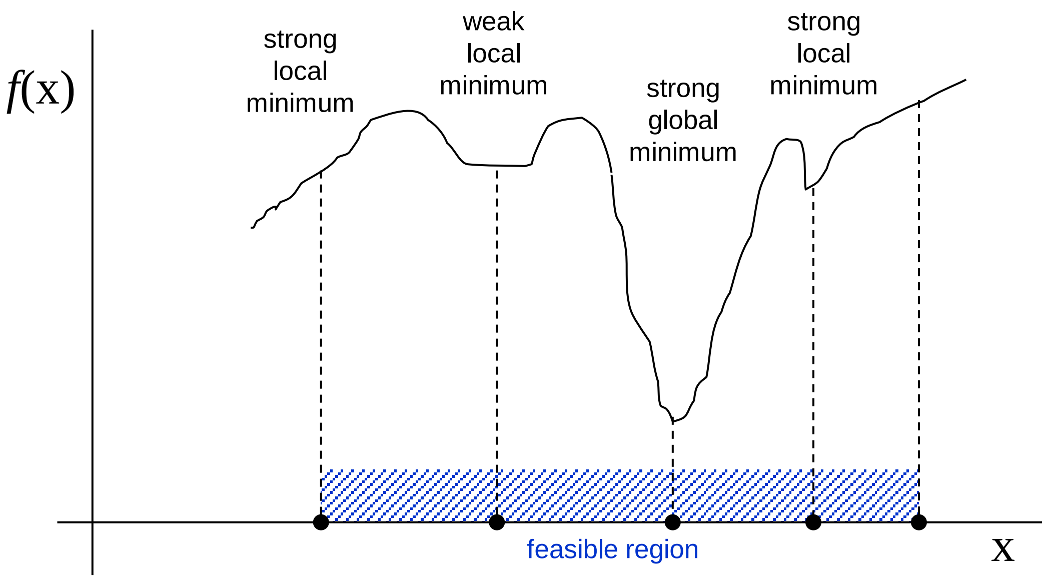


single objective, ND



multi-objective optimization

Types of Minima

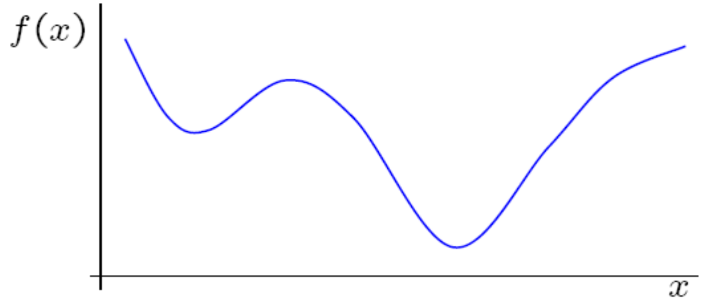


- which of the minima is found depends on the starting point
- such minima often occur in real applications

Unconstrained Univariate Optimization

Assume we can start close to the global minimum

$$\min_x f(x)$$



How to determine the minimum?

- search methods (Dichotomous, Fibonacci, Golden-Section)
- approximation methods
  - polynomial interpolation
  - Newton method
- combination of both

Search Methods

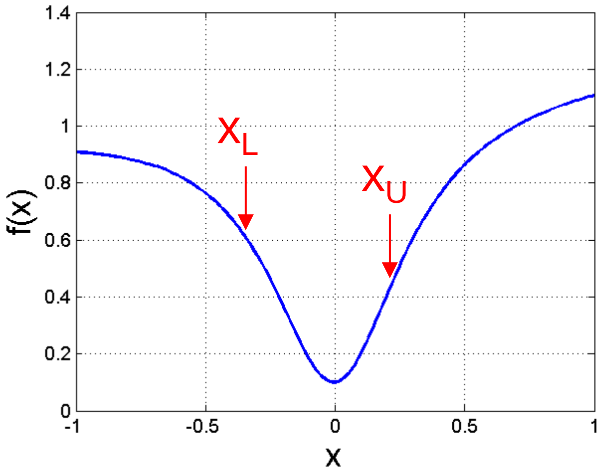
Start with the interval ("bracket") [x_L, x_U] such that the minimum x^* lies inside.

Evaluate $f(x)$ at two point inside the bracket.

Reduce the bracket.

Repeat the process

Can be applied to any function and differentiability is not essential.



Dichotomous

Newton's Method

Fit a quadratic approximation to $f(x)$ using both gradient and curvature information at x.

- Expand $f(x)$ locally using a Taylor series.

$$f(x + \delta x) = f(x) + f'(x)\delta x + \frac{1}{2} f''(x)\delta x^2 + o(\delta x^2)$$

- Find the δx which minimizes this local quadratic approximation.

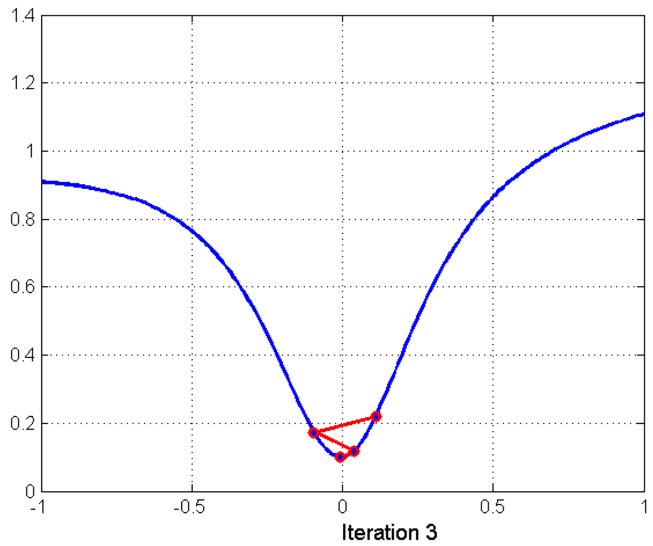
$$\delta x = -\frac{f'(x)}{f''(x)}$$

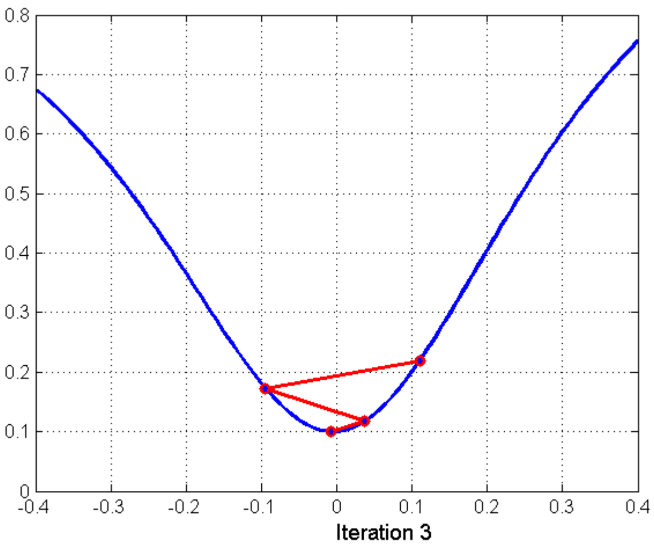
x_n

- Update x.

$$x_{n+1} = x_n - \delta x = x_n - \frac{f'(x)}{f''(x)}$$

Newton's Method

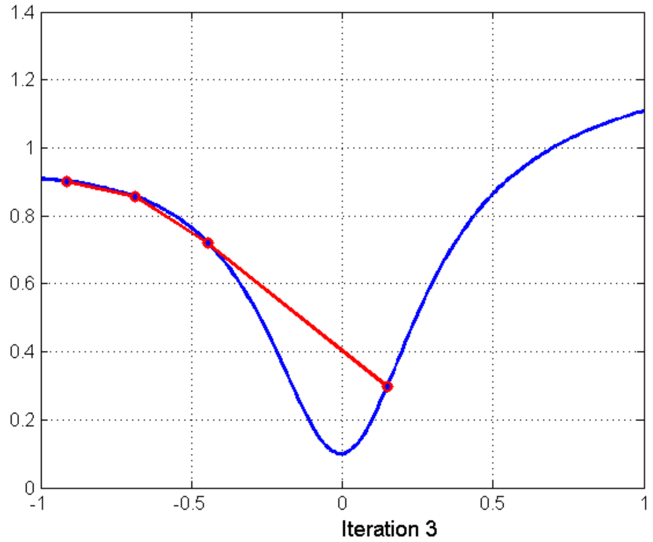


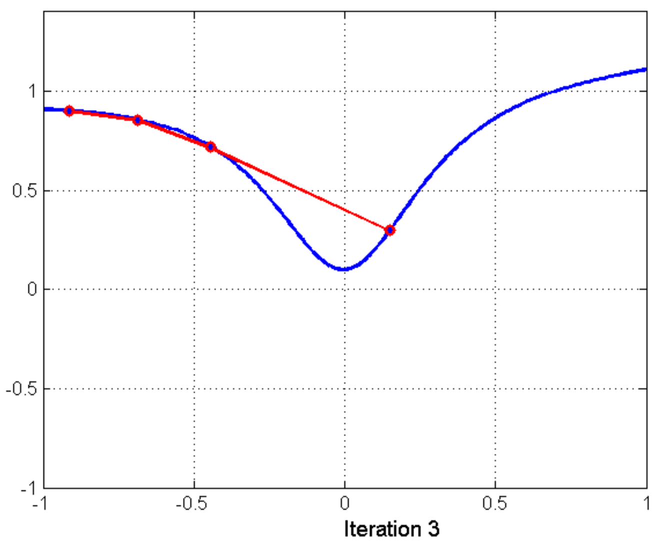


Newton's Method

Global convergence of Newton's method is poor.

Often fails if the starting point is too far from the minimum



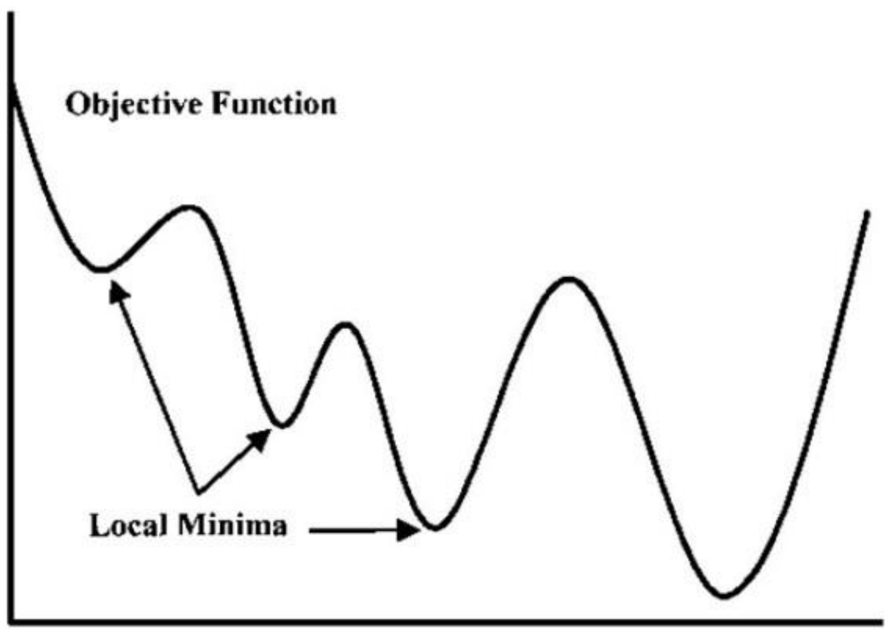


in practice, must be used with a globalization strategy which reduces the step length until function decrease is assured

Finding the Global Minimum

Can be challenging

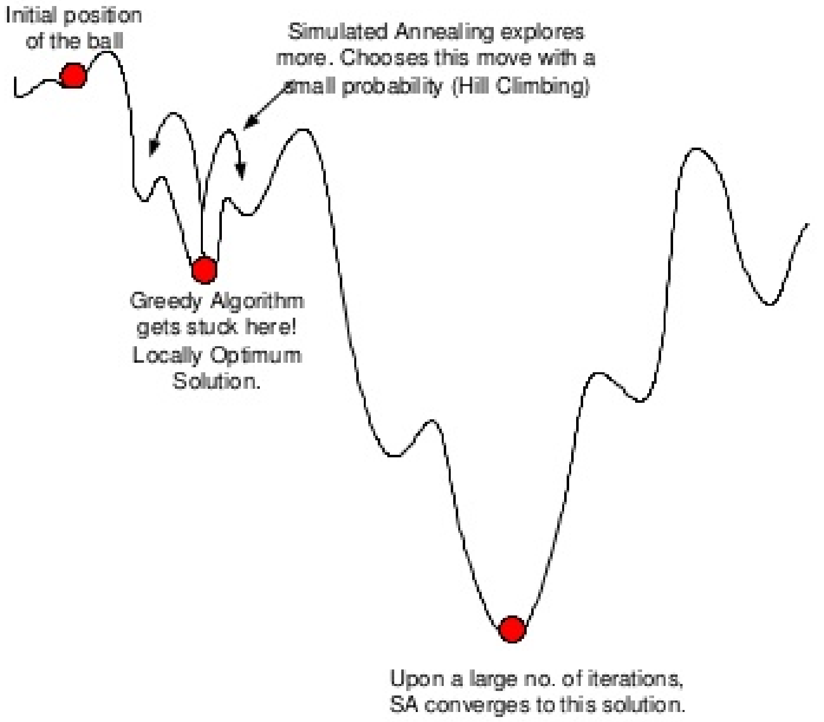
- may get stuck in a local minimum
- can happen in almost any algorithm
- how to deal with it?



Simulated Annealing

Finds the global minimum of a function by jumping to different sites

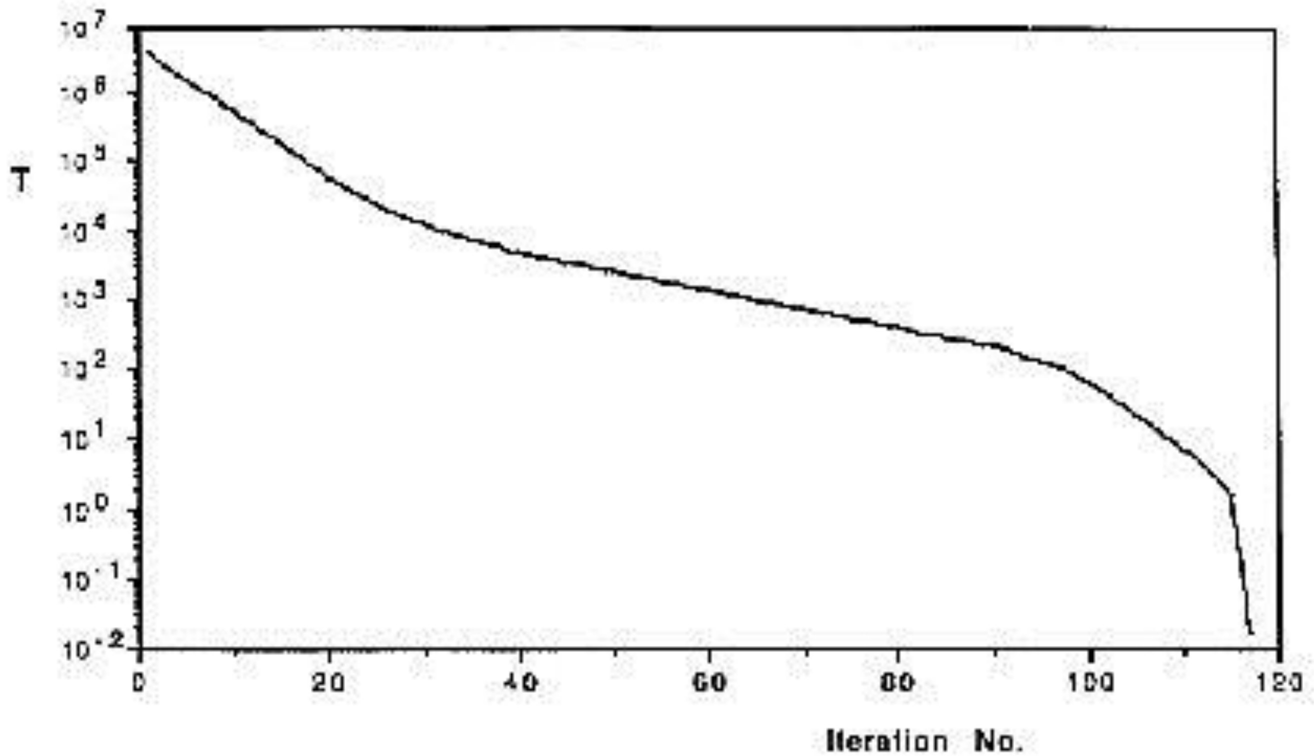
- extent of jumps depend on the time ort process, the cooling temperature



Cooling Function

Comes from annealing in metallurgy

- a technique involving heating and controlled cooling of a material to arrange the atoms in optimal patterns to reduce defects



Algorithm

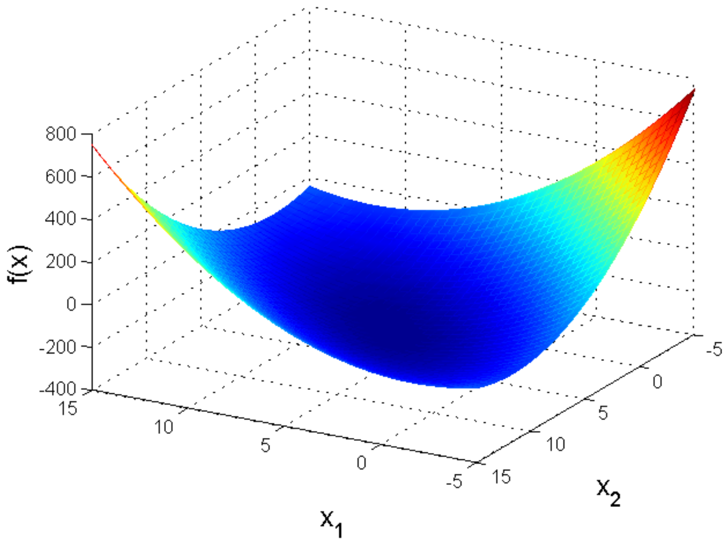
```
Algorithm SIMULATED-ANNEALING
Begin
        temp = INIT-TEMP;
        place = INIT-PLACEMENT;
        while (temp > FINAL-TEMP) do
                while (inner_loop_criterion = FALSE) do
                        new_place = PERTURB(place);
                        ΔC = COST(new_place)  - COST(place);
                        if (ΔC < 0) then
                                place = new_place;
                        else if (RANDOM(0,1) > e^(-(ΔC/temp))) then
                                place = new_place;
                temp = SCHEDULE(temp);
End.
```

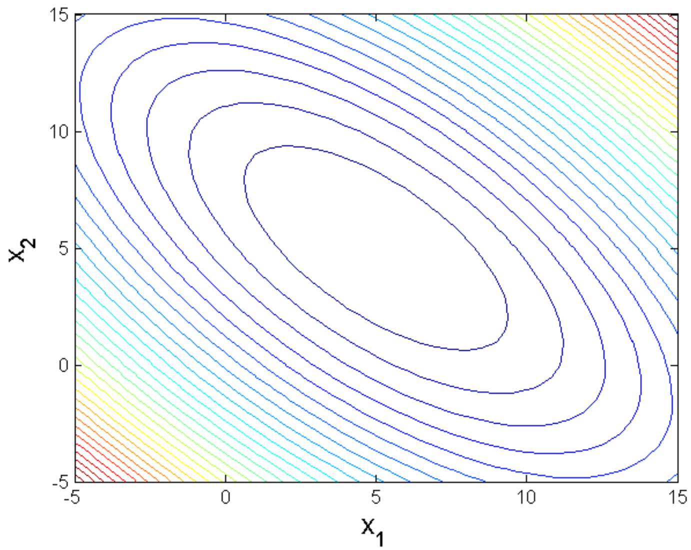
slide by Premchand Akella

Extension to multivariate (N) dimensions

How big N can be?

- problem sizes can vary from a handful of parameters to many thousands





Gradient Descent

Assume we have a large linear system of equations

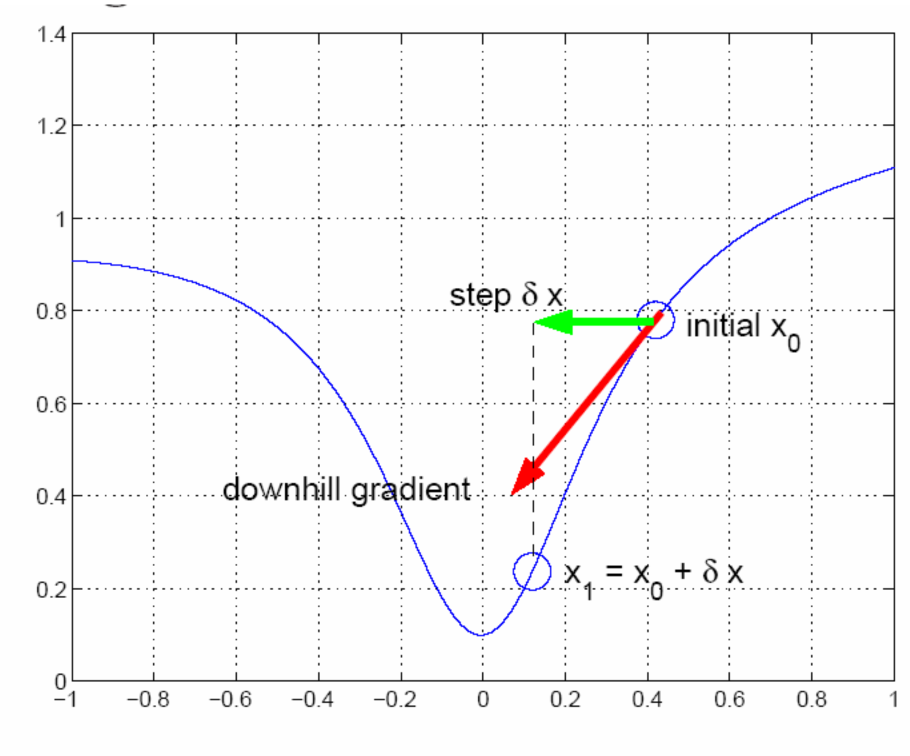
$$a_{11}x_1 + a_{12}x_2 + \ldots a_{1M}x_M = b_1$$
$$a_{21}x_1 + a_{22}x_2 + \ldots a_{2M}x_M = b_2$$
$$\ldots\ldots$$
$$a_{N1}x_1 + a_{N2}x_2 + \ldots a_{NM}x_M = b_N$$

- write in matrix form as Ax=b
- reality is that the equations are inconsistent due to noise
- so simple matrix inversion will not work well

Gradient Descent – Concept

Given a starting location, x_0, examine df/dx

- move into the *downhill* direction
- generate a new estimate, $x_1 = x_0 + \delta x$

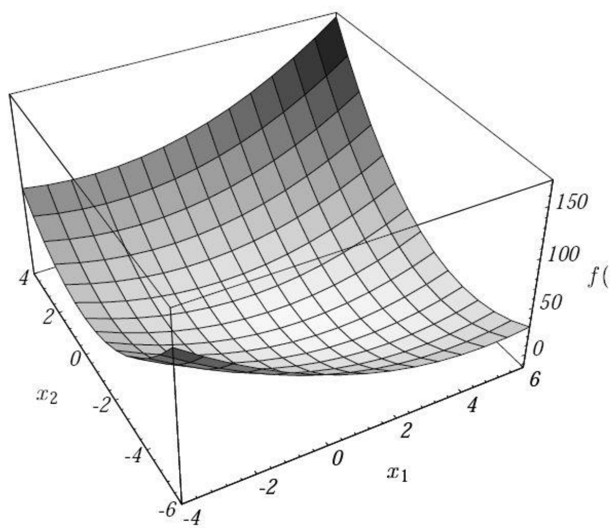


Gradient Descent – N > 1

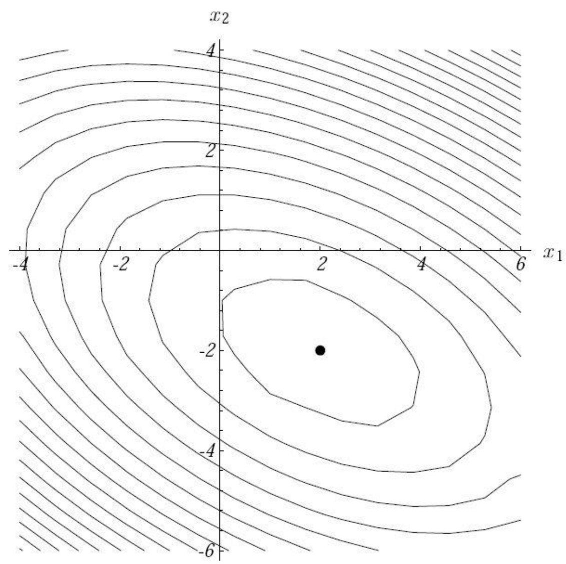
Quadratic form of a vector:

$$f(x) = \frac{1}{2} x^T A x - b^T x + c$$

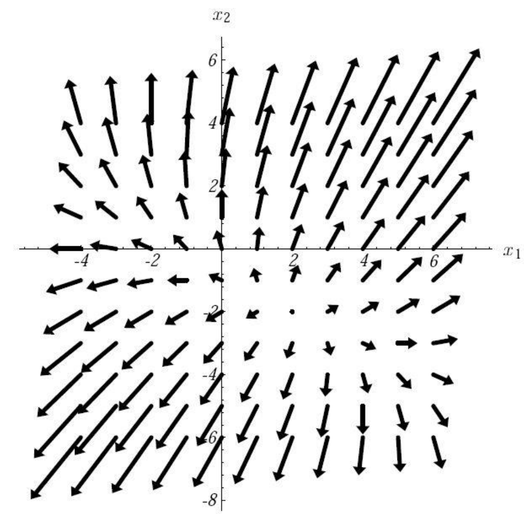
- this equation is minimized when *A·x=b*
- this occurs when f'(x)=0
- thus, minimizing the quadratic form will solve the matrix problem



Graph plot



Contour plot



Gradient plot

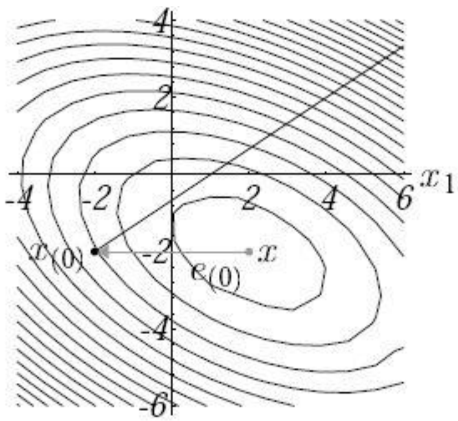
Steepest Descent

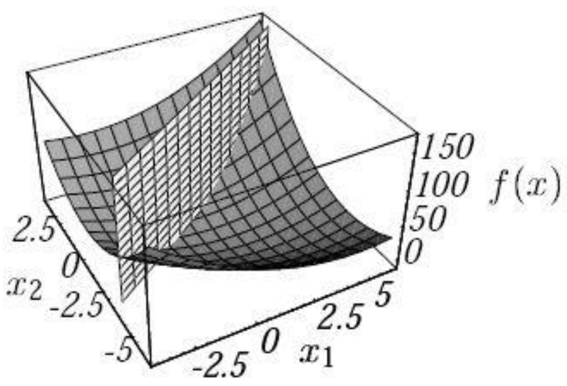
Start at an arbitrary point and slide down to the bottom of the parabola

- in practice this will be a hyper-parabola since *x, b* are high-dimensional
- choose the direction in which *f* decreases most quickly

$$-f'(x_{(i)}) = b - Ax_{(i)}$$

where $x_{(i)}$ is the current (predicted) solution



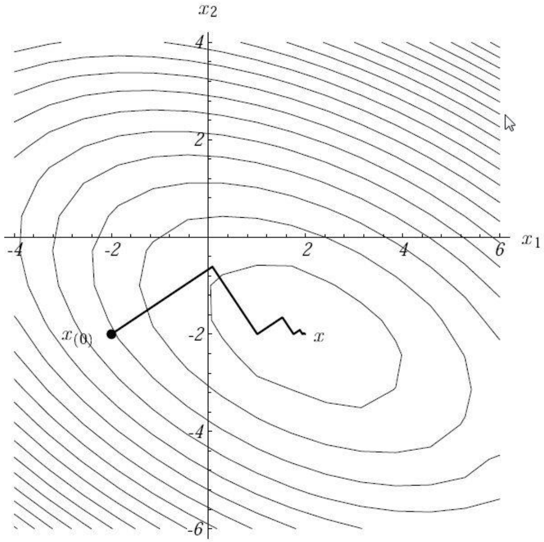


Figures from J. Shewchuk, UC Berkeley

Steepest Descent

Start at some initial guess $x_{(0)}$

- this will likely not find the solution
- need to follow $f'(x_{(0)})$ some ways and then change directions
- question is *where* do we change directions



Some basics:

- error: how far are we away from the solution – unknown

$$e_{(i)} = x_{(i)} - x$$

- residual: how far are we away from the correct value of *b* – computable

$$\mathrm{r}_{(i)} = b - Ax_{(i)}$$

$$\mathrm{r}_{(i)} = Ae_{(i)}$$

A transforms *e* into the space of *b*

$$\mathrm{r}_{(i)} = -f\,'(x_{(i)})$$

Steepest Descent

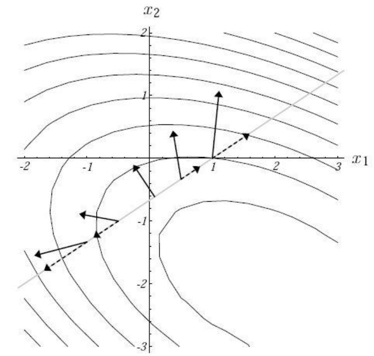
Finding the right place to turn directions is called *line search*

$$x_{(1)} = x_{(0)} + \alpha r_{(0)}$$

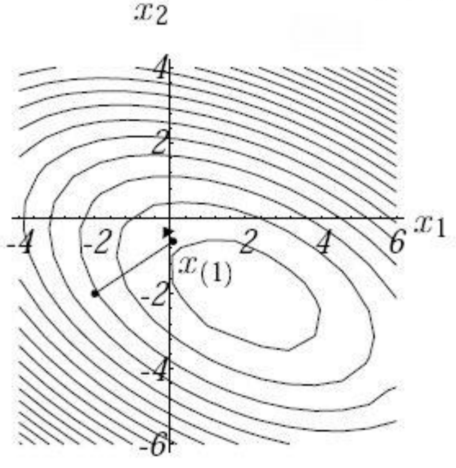
To find α we can use the following requirements:

- the new direction of *r* must be orthogonal to the previous:

$$r_{(1)}^{T} r_{(0)} = 0$$



- the residual at $x_{(1)}$ $\quad f'(x_{(1)}) = -r_{(1)}$



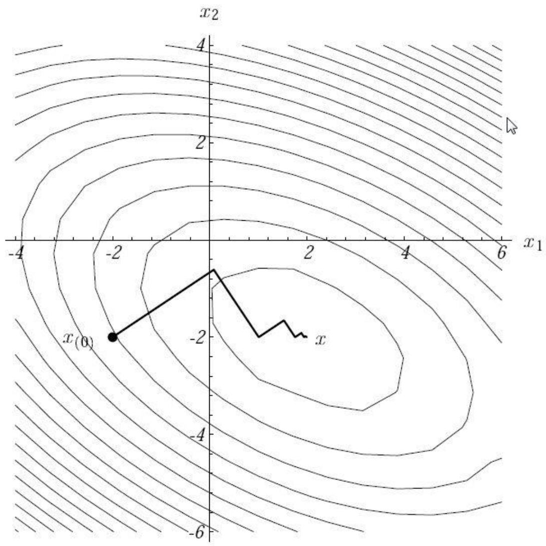
- after some math: $\alpha = \dfrac{r_{(0)}^{T} r_{(0)}}{r_{(0)}^{T} A r_{(0)}}$

Steepest Descent: Summary

$$r_{(i)} = b - Ax_{(i)}$$

$$\alpha = \frac{r_{(i)}{}^T r_{(i)}}{r_{(i)}{}^T A r_{(i)}}$$

$$x_{(i+1)} = x_{(i)} + \alpha r_{(i)}$$



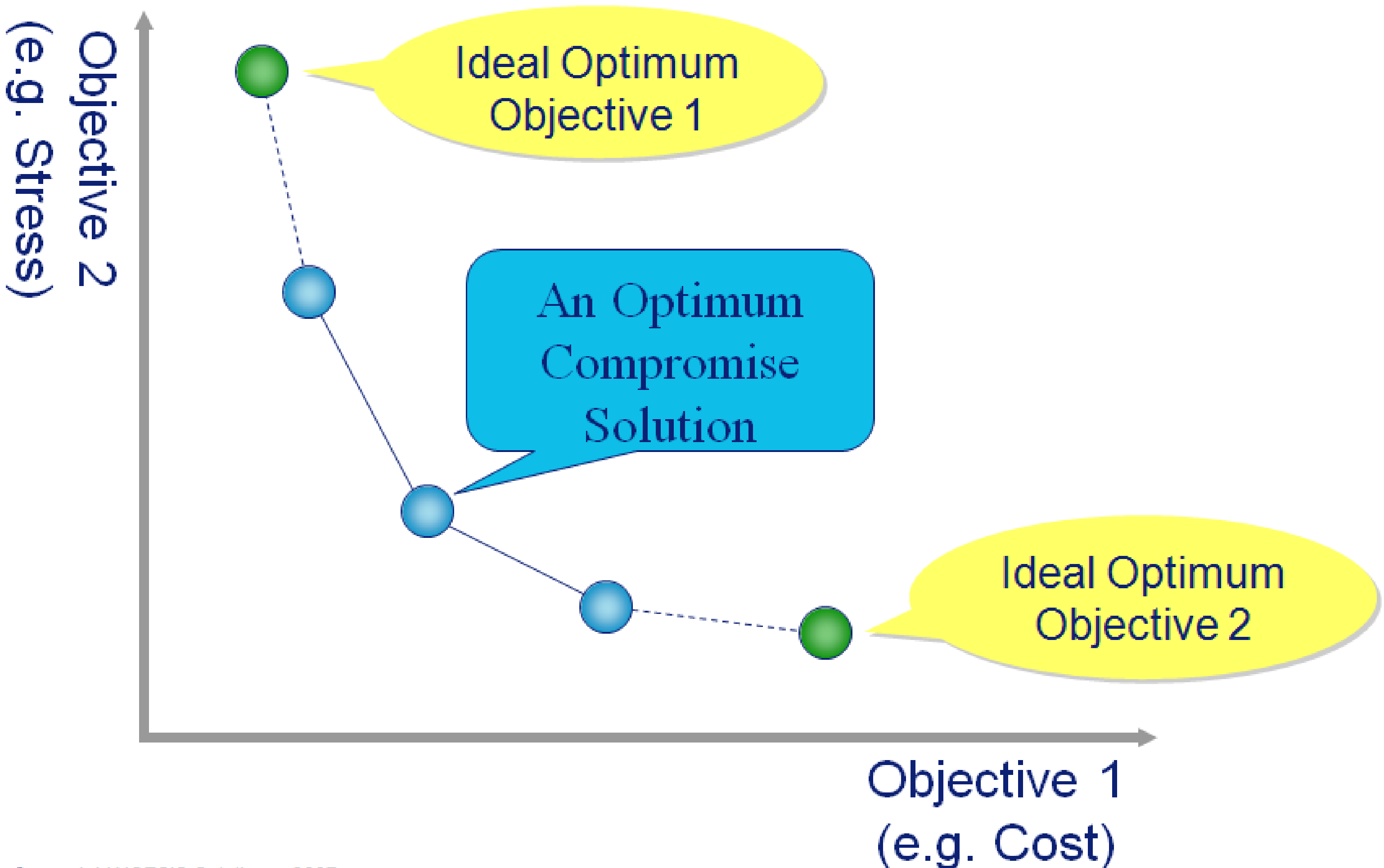
Shortcoming:

- sub-optimal since some directions might be taken more than once
- this can be fixed by the method of Conjugant Gradients

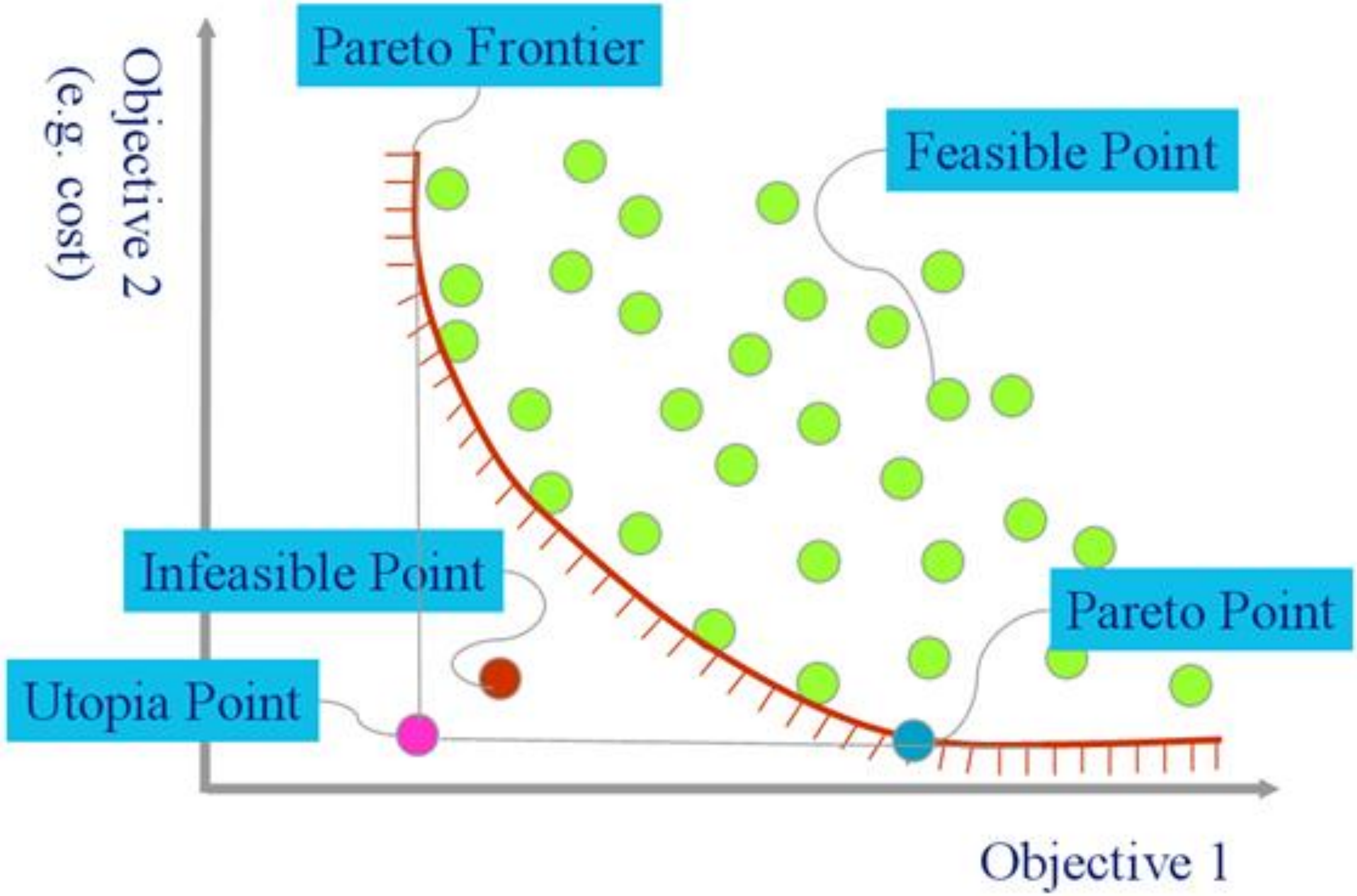
Multi-Objective Optimization

portions courtesy of http://www.noesissolutions.com/
and University of Ottawa

Multi-Objective Optimization



Multi-Objective Optimization



Multi-Objective Optimization

Multi-objective optimization (MOO) is the optimization of conflicting objectives

Conceptual Example

Suppose you need to fly on a long trip:
Should you choose the cheapest ticket (more connections) or shortest flying time (more expensive)?

It is impossible to put a value on time, so these two objectives can't be linked.

Also, the relative importance will vary.

- there may be a business emergency you need to go fix quickly.
- or, maybe you are on a very tight budget.

Pareto-Optimal Solutions

A MOO problem with constraints will have many solutions in the feasible region.

Even though we may not be able to assign numerical relative importance to the multiple objectives, we can still classify some possible solutions as better than others.

We will see this in the following example.

Pareto-Optimal Solutions Example

Suppose in our airplane-trip example we find the following tickets:

Ticket	Travel Time (hrs)	Ticket Price ($)
A	10	1700
B	9	2000
C	8	1800
D	7.5	2300
E	6	2200

Comparison of Solutions

If we compare tickets A and B, we can't say that either is superior without knowing the relative importance of Travel Time vs. Price.

However, comparing tickets B and C shows that C is better than B in both objectives, so we can say that C *"dominates"* B.

So, as long as C is a feasible option, there is no reason we would choose B.

Ticket	Travel Time (hrs)	Ticket Price ($)
A	10	1700
B	9	2000
C	8	1800
D	7.5	2300
E	6	2200

Comparison of Solutions

If we finish the comparisons, we also see that D is dominated by E.

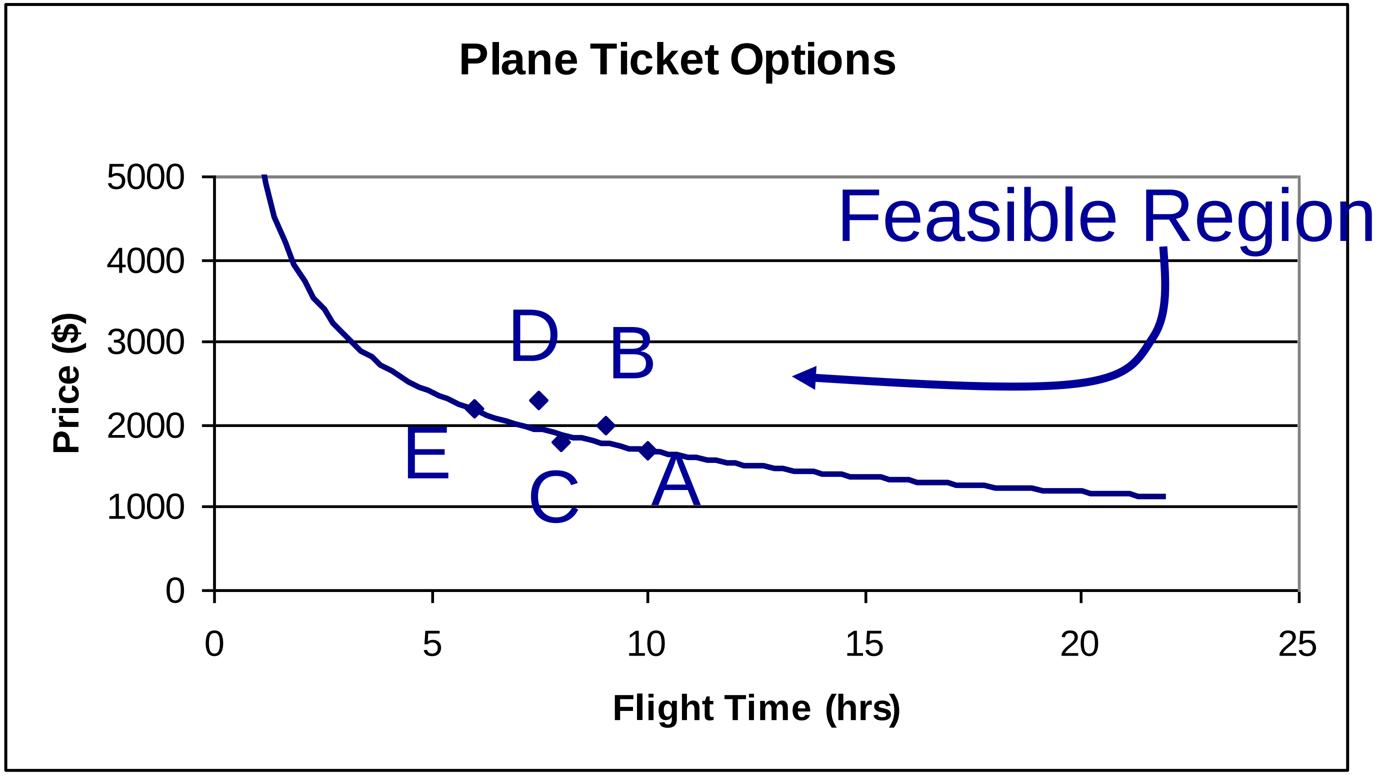
The rest of the options (A, C, E) have a trade-off associated with Time vs. Price, so none is clearly superior to the others.

We call this the *"non-dominated"* set of solutions because none of the solutions are dominated.

Ticket	Travel Time (hrs)	Ticket Price ($)
A	10	1700
B	9	2000
C	8	1800
D	7.5	2300
E	6	2200

Graph of Solutions

Usually, solutions of this type form a typical shape, shown in the chart below:



Types of Solutions

Solutions that lie along the line are non-dominated solutions

- those that lie inside the line are dominated
- there is always another solution on the line that has at least one objective that is better.

Pareto-optimal Solutions

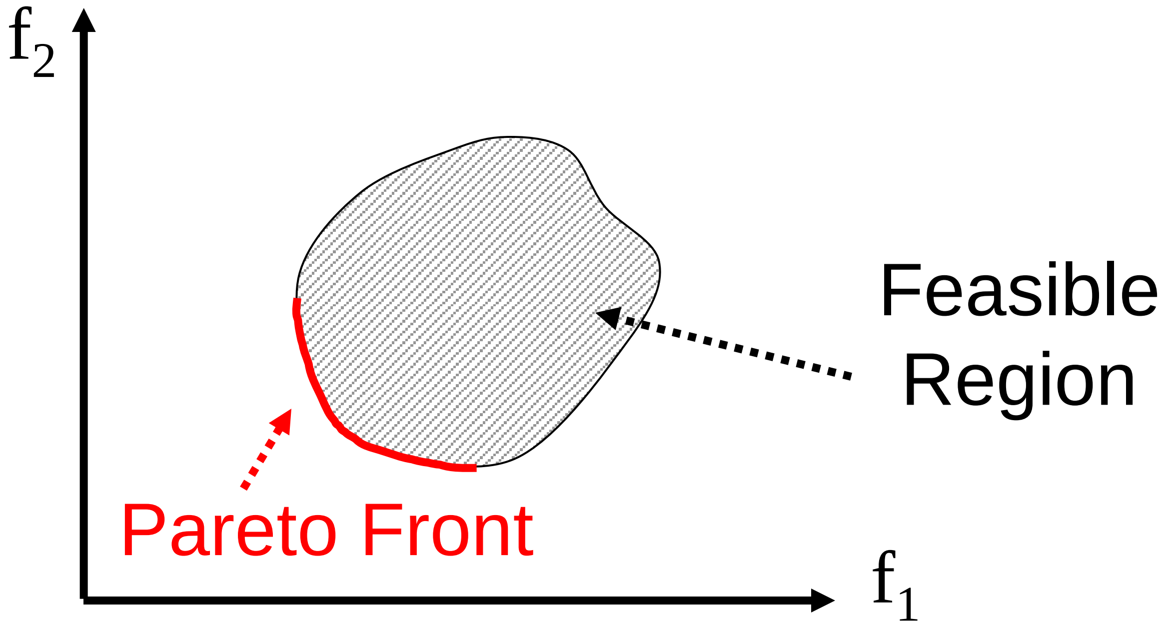
The line is called the *Pareto front* and solutions on it are called *Pareto-optimal*.

All Pareto-optimal solutions are non-dominated.

Thus, it is important in MOO to find the solutions as close as possible to the Pareto front and as far along it as possible.

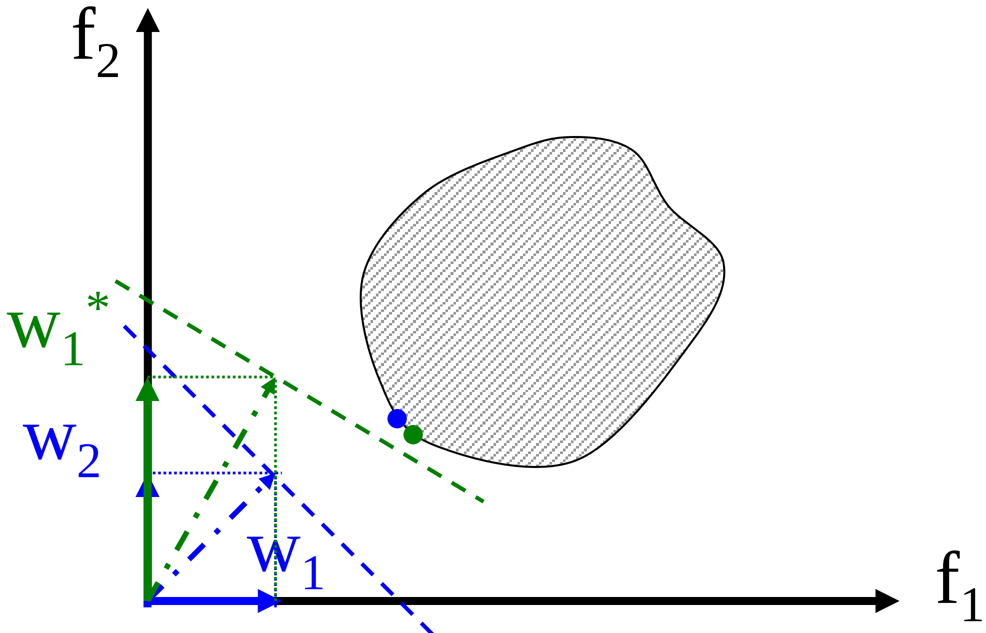
Graphical Example

For the following feasible region with objectives f_1 and f_2 where both f_1 and f_2 are minimized:



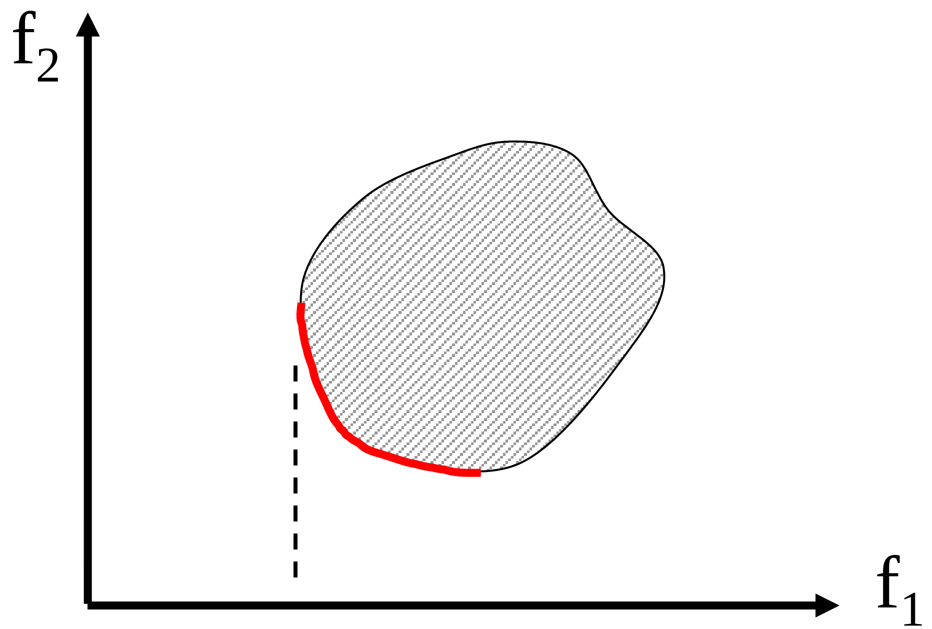
Finding the Pareto Front

One way to imagine finding points on the Pareto front is by using a combination of numerical weights for the two objectives:



Finding the Pareto Front

If this is done for a 90° span of lines, all the points on the Pareto front will be found.



Practicality of this Procedure

Actually, this is not the procedure that is used in practice, but it is a good illustration of the concept.

This procedure would require finding all possible points in the feasible region and then using many combinations of weights.

For more than two objectives, the complexities and the number of combinations make this impractical.

Realistic Procedures

There are different methods used in practice

- one is to use a genetic algorithm to enumerate points along the Pareto front over several iterations
- use some method to rank the quality of the trade-offs based on the particular application being modeled

Shall discuss one particular genetic algorithm

- ant colony optimization (ACO)

Ant Colony Optimization

- studies artificial systems that take inspiration from the behavior of real ant colonies
- used to solve discrete optimization problems

Ant Colony Optimization

portions courtesy of University of Central Florida

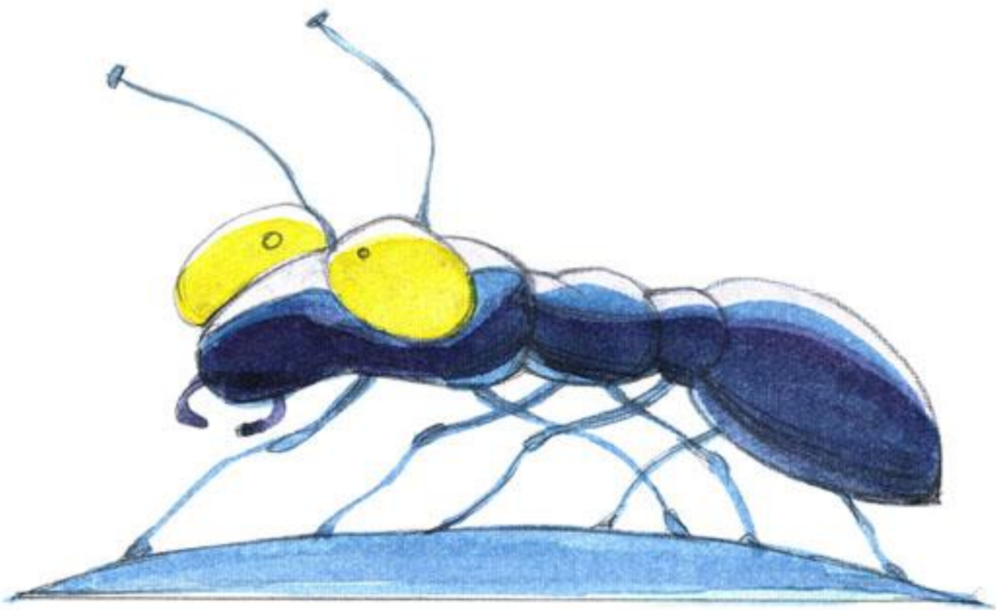
Almost blind.

Incapable of achieving complex tasks alone.

Rely on the phenomena of swarm intelligence for survival.

Capable of establishing shortest-route paths from their colony to feeding sources and back.

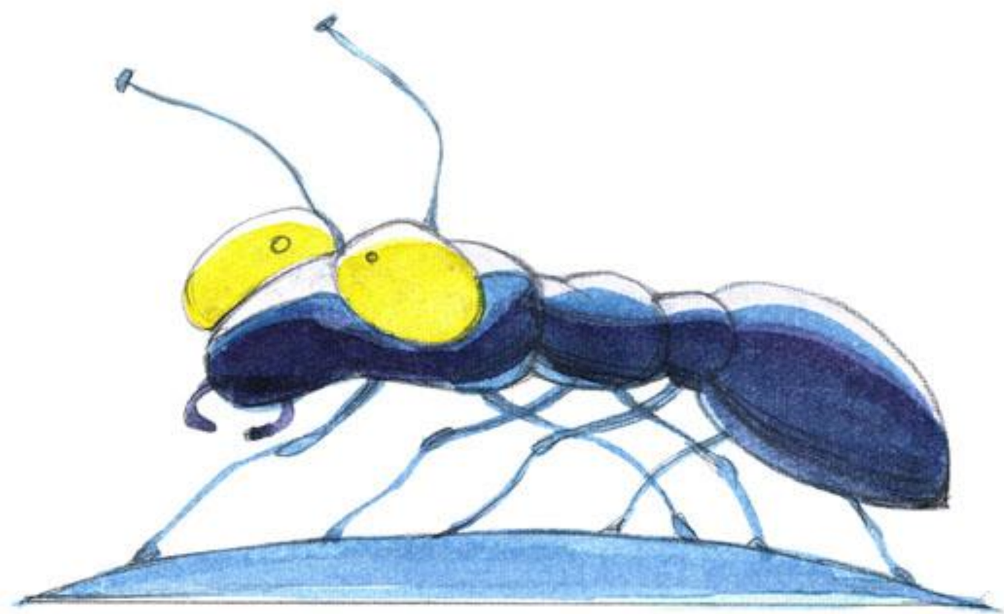
Use stigmergic communication via pheromone trails.



Follow existing pheromone trails with high probability.

What emerges is a form of *autocatalytic* behavior: the more ants follow a trail, the more attractive that trail becomes for being followed.

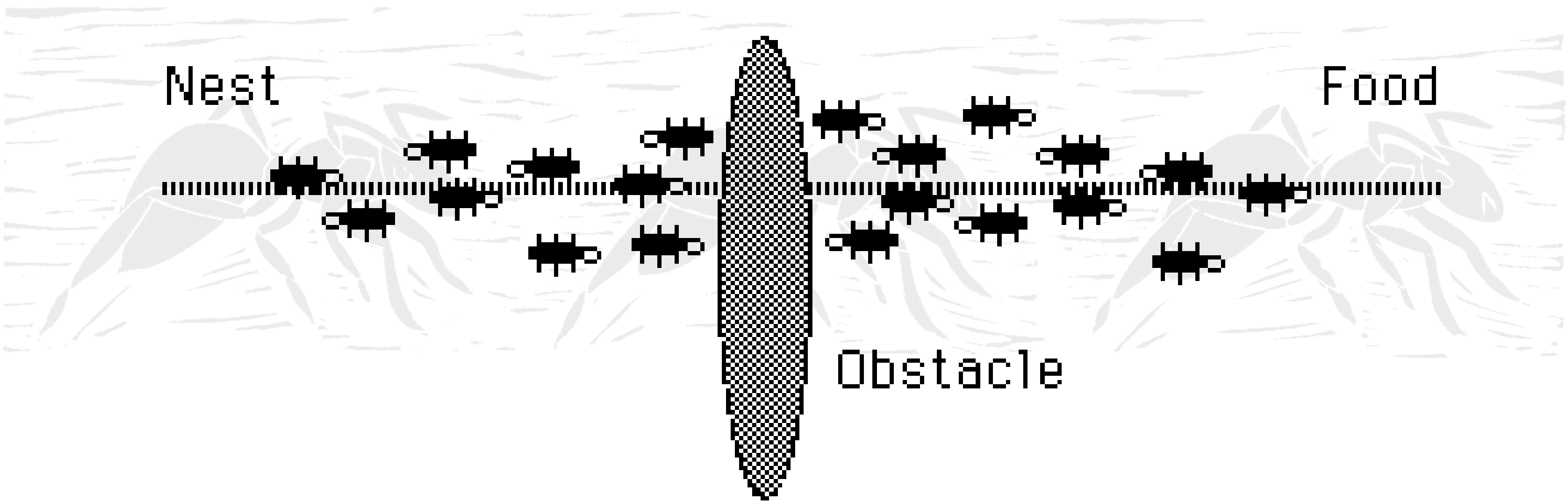
The process is thus characterized by a positive feedback loop, where the probability of a discrete path choice increases with the number of times the same path was chosen before.



Naturally Observed Ant Behavior

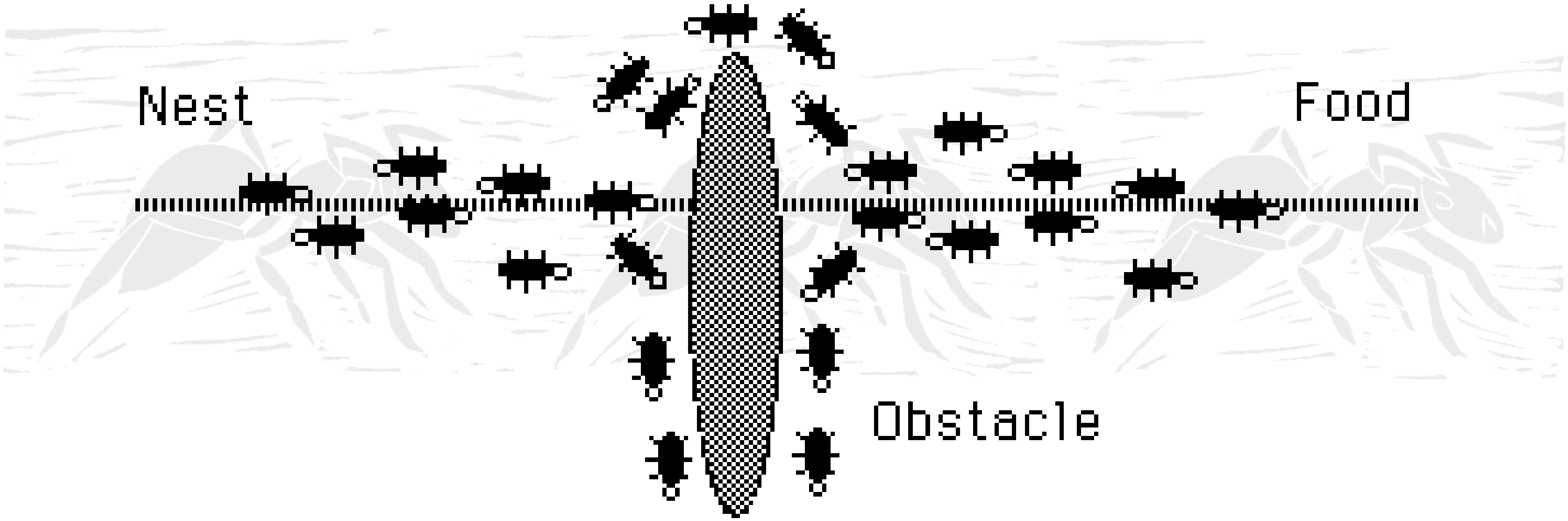
All is well in the world of the ant.

Naturally Observed Ant Behavior



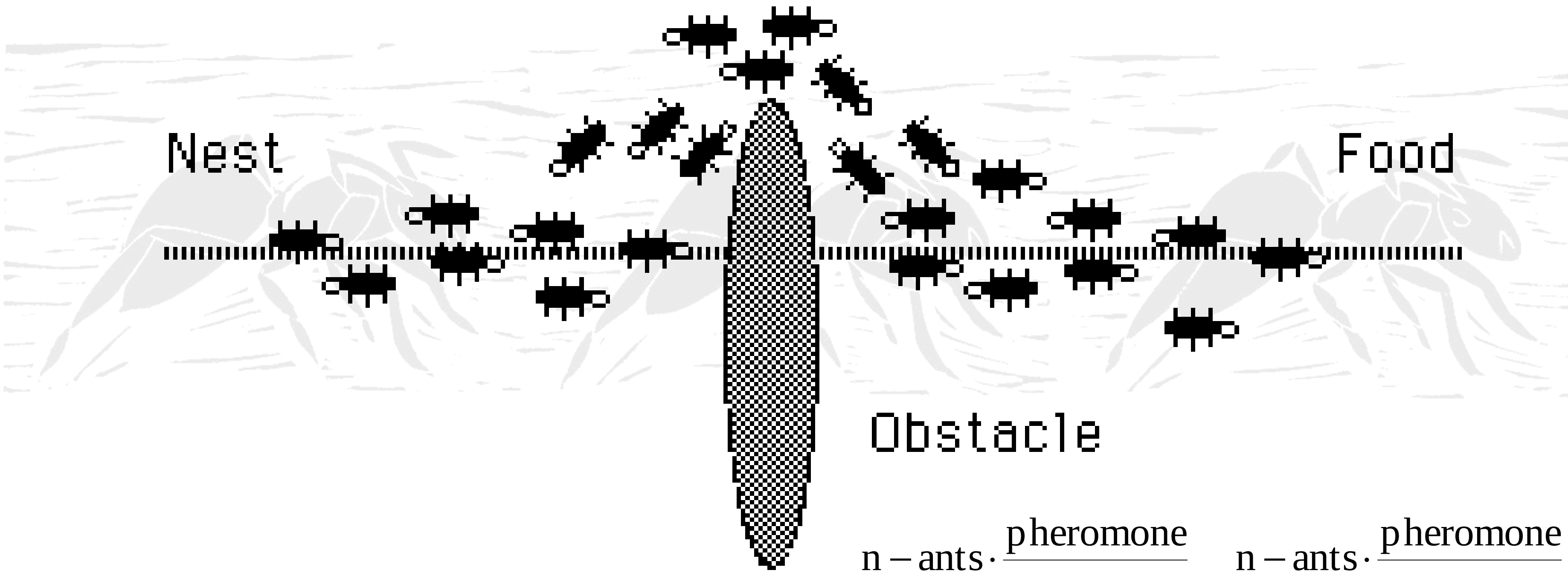
Oh no! An obstacle has blocked our path!

Naturally Observed Ant Behavior



Where do we go? Everybody, flip a coin.

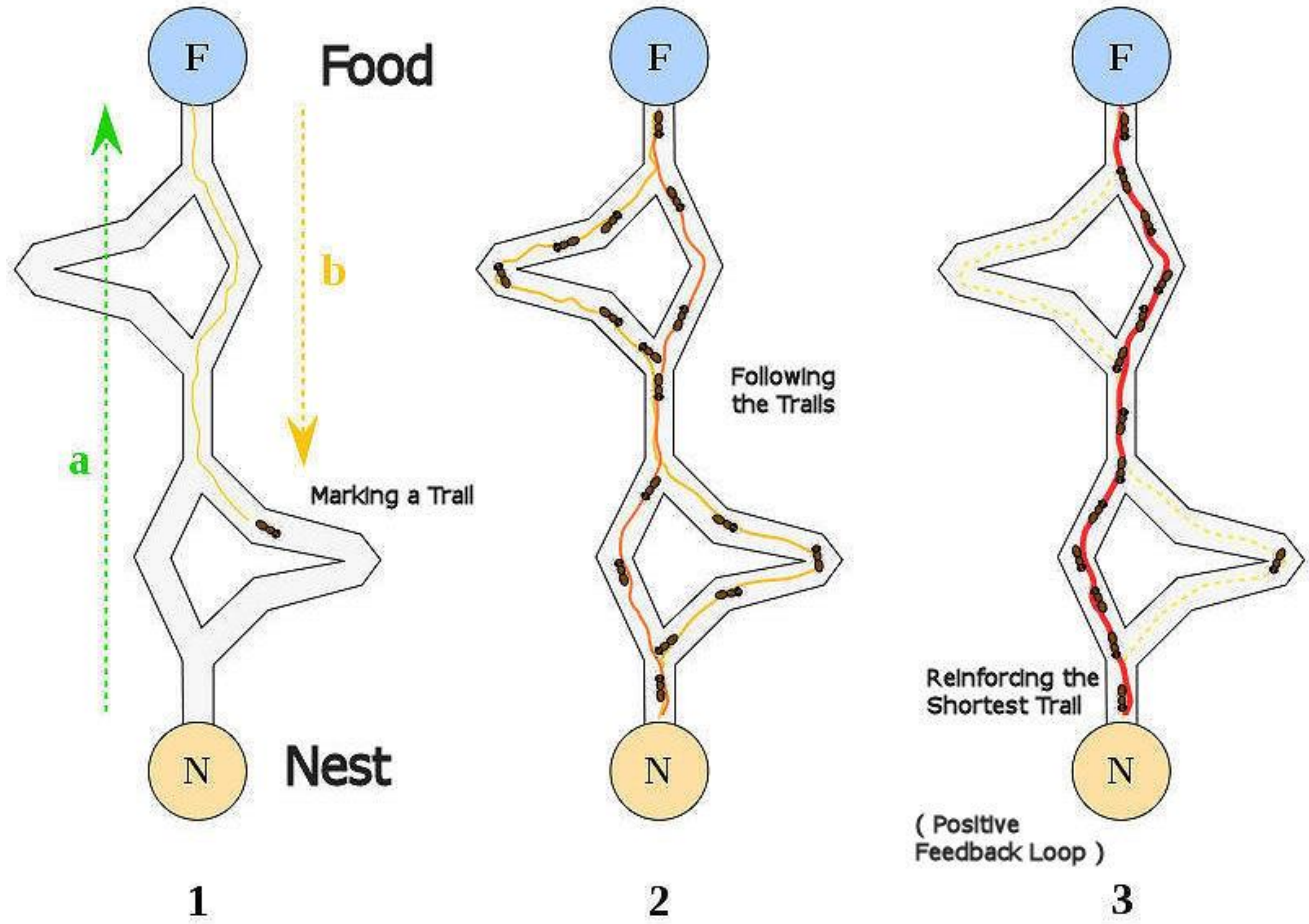
Naturally Observed Ant Behavior



Shorter path reinforced.

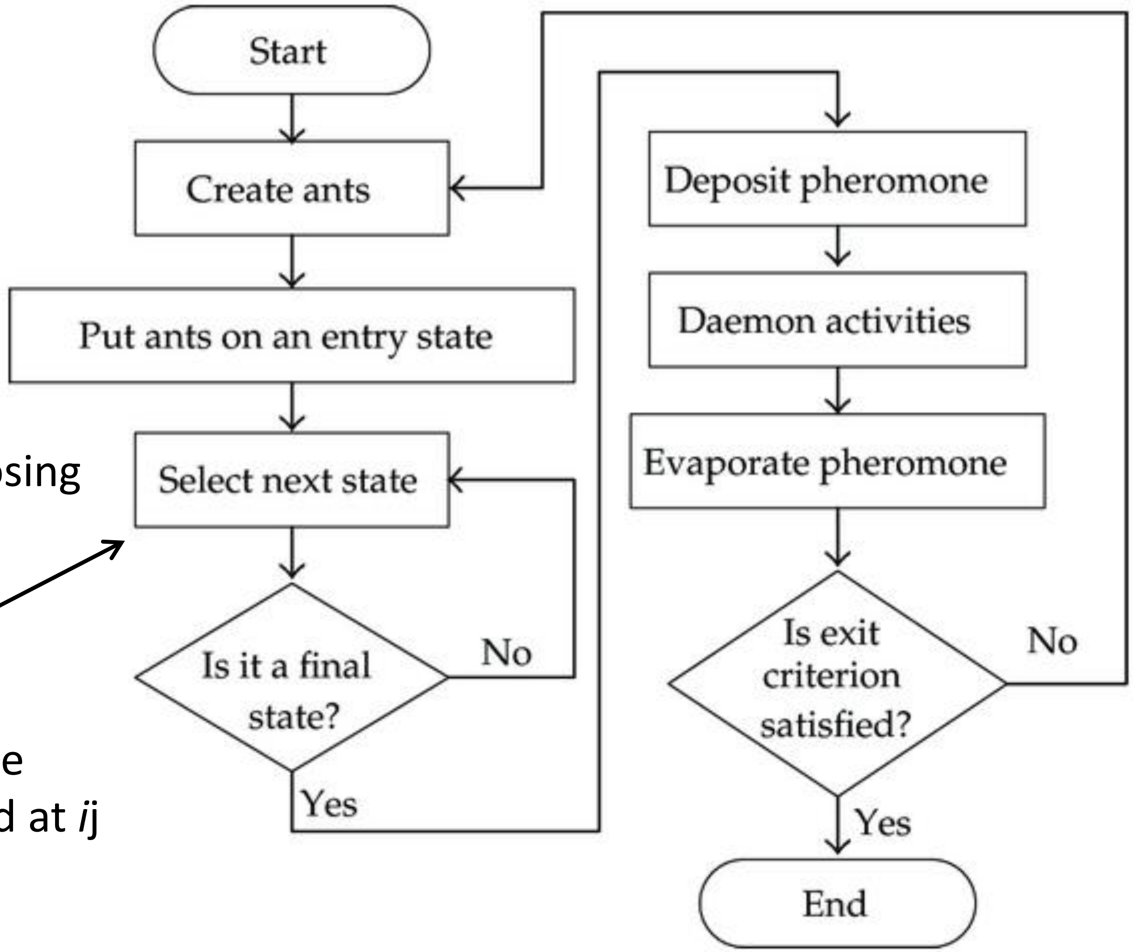
$$\frac{\text{n} - \text{ants} \cdot \dfrac{\text{pheromone}}{\text{x - distance}}}{\dfrac{\text{distance}_{\text{longerpath}}}{\text{time}}} < \frac{\text{n} - \text{ants} \cdot \dfrac{\text{pheromone}}{\text{x - distance}}}{\dfrac{\text{distance}_{\text{shorterpath}}}{\text{time}}}$$

Ant Colony Optimization



http://en.wikipedia.org/wiki/Ant_colony_optimization

ACO Algorithm



Probability of choosing value *j* for state *i* is

$$P_{i,j} = \frac{\tau_{i,j}}{\sum_{q=0}^{R_i} \tau_{i,q}}$$

τ_{ij} is the pheromone currently deposited at *ij*

Evaporate (using ρ) and update pheromone for value *j* in state *i* given average score s_{ij} for all ants that chose that state and value

$$\tau_{i,j}^{(m+1)} = (1-\rho)\tau_{i,j}^{(m)} + \bar{s}_{i,j}$$

Particle Swarm Optimization (PSO)

Related to ACO

~ Basic Idea: Social Behavior ~

- An individual gains knowledge from other members in the swarm (population)