

Beyond Classical Search

- Chapter 3 covered problems that considered the whole search space and produced a sequence of actions leading to a goal.
- Chapter 4 covers techniques (some developed outside of AI) that don't try to cover the whole space and only the goal state, not the steps, are important.
- The techniques of Chapter 4 tend to use much less memory and are not guaranteed to find an optimal solution.

More Search Methods

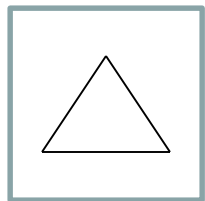
- Local Search
 - Hill Climbing
 - Simulated Annealing
 - Beam Search
 - Genetic Search
- Local Search in Continuous Spaces
- Searching with Nondeterministic Actions
- Online Search (agent is executing actions)

Local Search Algorithms and Optimization Problems

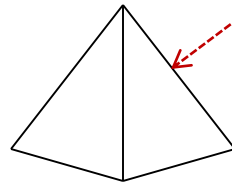
- **Complete state** formulation
 - For example, for the 8 queens problem, all 8 queens are on the board and need to be moved around to get to a goal state
- Equivalent to **optimization problems** often found in science and engineering
- Start somewhere and try to get to the solution from there
- **Local search** around the current state to decide where to go next

Pose Estimation Example

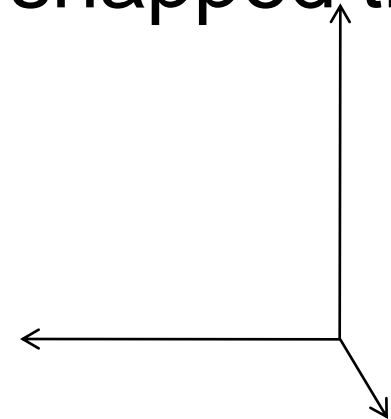
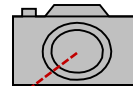
- Given a geometric model of a 3D object and a 2D image of the object.
- Determine the **position** and **orientation** of the object wrt the camera that snapped the image.



image



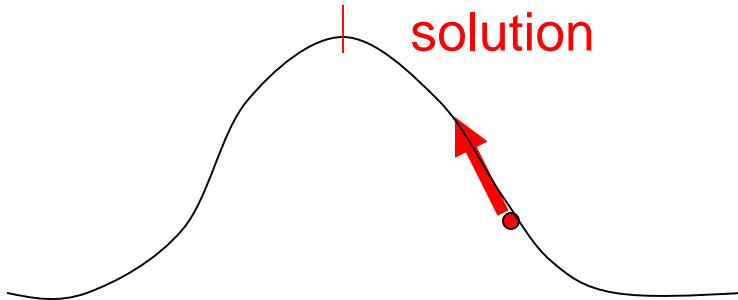
3D object



- State **$(x, y, z, \theta_x, \theta_y, \theta_z)$**

Hill Climbing

“Gradient ascent”



Note: solutions shown here as **max** not min.

Often used for numerical optimization problems.

How does it work?

In continuous space, the gradient tells you the direction in which to move uphill.

AI Hill Climbing

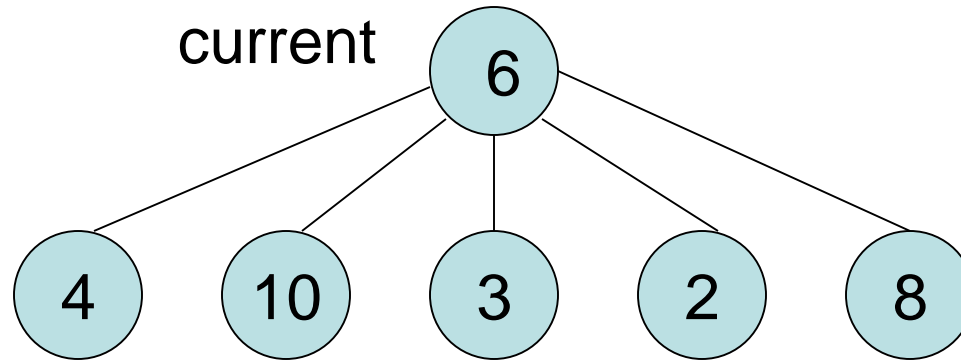
Steepest-Ascent Hill Climbing

- $current \leftarrow$ start node
- loop do
 - $neighbor \leftarrow$ a highest-valued successor of $current$
 - if $neighbor.Value \leq current.Value$ then return $current.State$
 - $current \leftarrow neighbor$
- end loop

At each step, the current node is replaced by the best (highest-valued) neighbor.

This is sometimes called **greedy local search**.

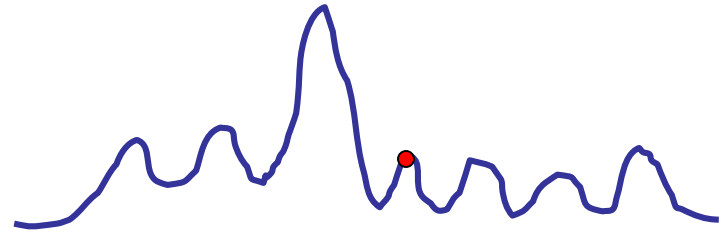
Hill Climbing Search



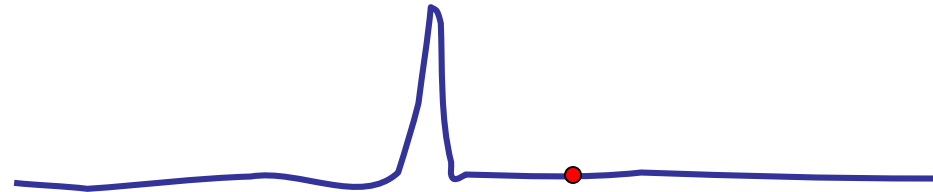
What if current had a value of 12?

Hill Climbing Problems

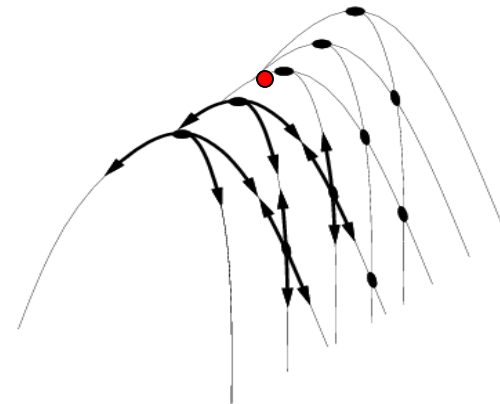
Local maxima



Plateaus



Diagonal ridges



What is it sensitive to?
Does it have any advantages?

Solving the Problems

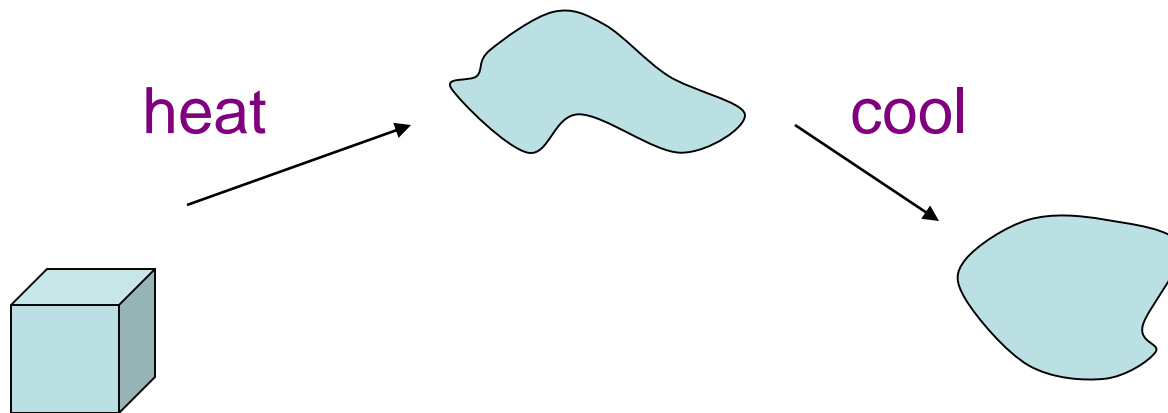
- **Allow backtracking** (What happens to complexity?)
- **Stochastic hill climbing**: choose at random from uphill moves, using steepness for a probability
- **Random restarts**: “If at first you don’t succeed, try, try again.”
- **Several moves** in each of several directions, then test
- **Jump** to a different part of the search space

Simulated Annealing

- Variant of hill climbing (so up is good)
- Tries to **explore** enough of the search space **early on**, so that the final solution is less sensitive to the start state
- May make some **downhill moves** before finding a good way to move uphill.

Simulated Annealing

- Comes from the physical process of annealing in which **substances** are raised to high energy levels (**melted**) and then **cooled** to solid state.



- The probability of moving to a higher energy state, instead of lower is

$$p = e^{(-\Delta E/kT)}$$

where ΔE is the positive change in energy level, T is the temperature, and k is Boltzmann's constant.

Simulated Annealing

- At the beginning, the temperature is high.
- As the temperature becomes lower
 - kT becomes lower
 - $\Delta E/kT$ gets bigger
 - $(-\Delta E/kT)$ gets smaller
 - $e^{(-\Delta E/kT)}$ gets smaller
- As the process continues, the probability of a downhill move gets smaller and smaller.

For Simulated Annealing

- ΔE represents the change in the value of the objective function.
- Since the physical relationships no longer apply, drop k . So $p = e^{(-\Delta E/T)}$
- We need an **annealing schedule**, which is a sequence of values of T : T_0, T_1, T_2, \dots

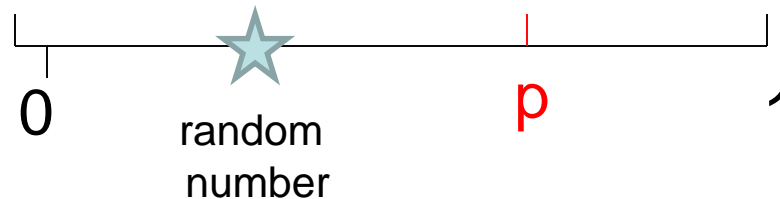
Simulated Annealing Algorithm

- $current \leftarrow$ start node;
- for each T on the schedule /* need a schedule */
 - $next \leftarrow$ randomly selected successor of $current$
 - evaluate next; if it's a goal, return it
 - $\Delta E \leftarrow next.Value - current.Value$ /* already negated */
 - if $\Delta E > 0$
 - then $current \leftarrow next$ /* better than current */
 - else $current \leftarrow next$ with probability $e^{(\Delta E/T)}$

How would you do this probabilistic selection?

Probabilistic Selection

- Select *next* with probability p



- Generate a random number
- If it's $\leq p$, select *next*

Simulated Annealing Properties

- At a fixed “temperature” T , state occupation probability reaches the Boltzmann distribution

$$p(x) = \alpha e^{-(E(x)/kT)}$$

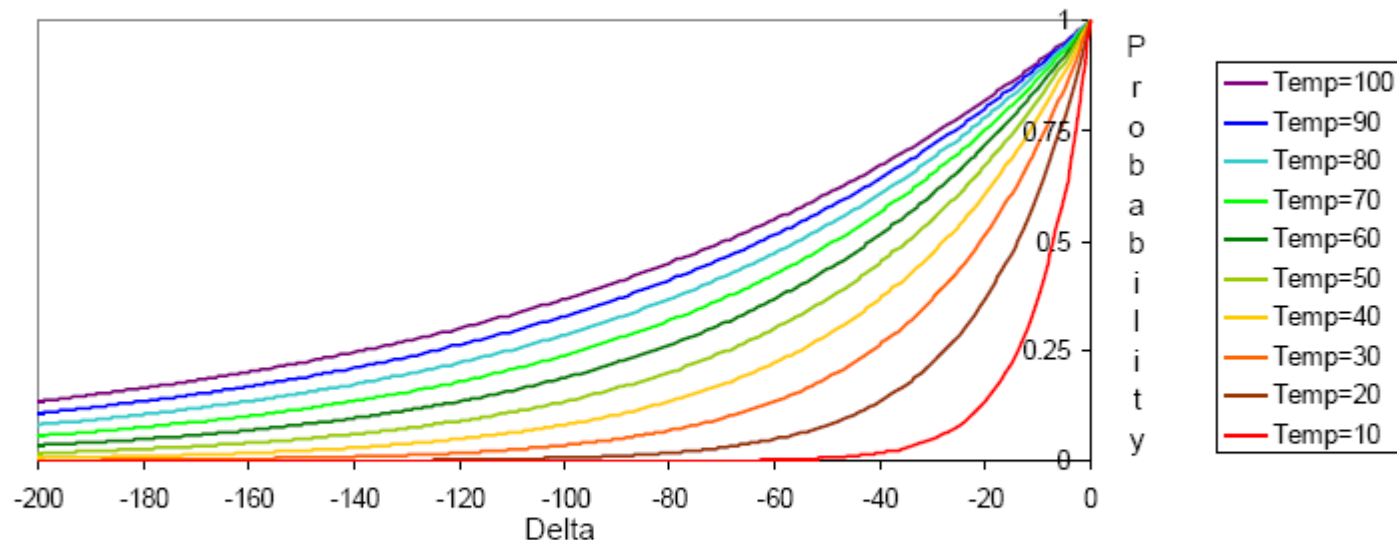
- If T is decreased slowly enough (very slowly), the procedure will reach the best state.
- Slowly enough has proven too slow for some researchers who have developed alternate schedules.

Simulated Annealing Schedules

- Acceptance criterion and cooling schedule

if ($\text{delta} \geq 0$) accept

else if ($\text{random} < e^{\text{delta} / \text{Temp}}$) accept, else reject /* $0 \leq \text{random} \leq 1$ */



Initially temperature is very high (most bad moves accepted)

Temp slowly goes to 0, with multiple moves attempted at each temperature

Final runs with temp=0 (always reject bad moves) greedily “quench” the system

Simulated Annealing Applications

- Basic Problems
 - Traveling salesman
 - Graph partitioning
 - Matching problems
 - Graph coloring
 - Scheduling
- Engineering
 - VLSI design
 - Placement
 - Routing
 - Array logic minimization
 - Layout
 - Facilities layout
 - Image processing
 - Code design in information theory

Local Beam Search

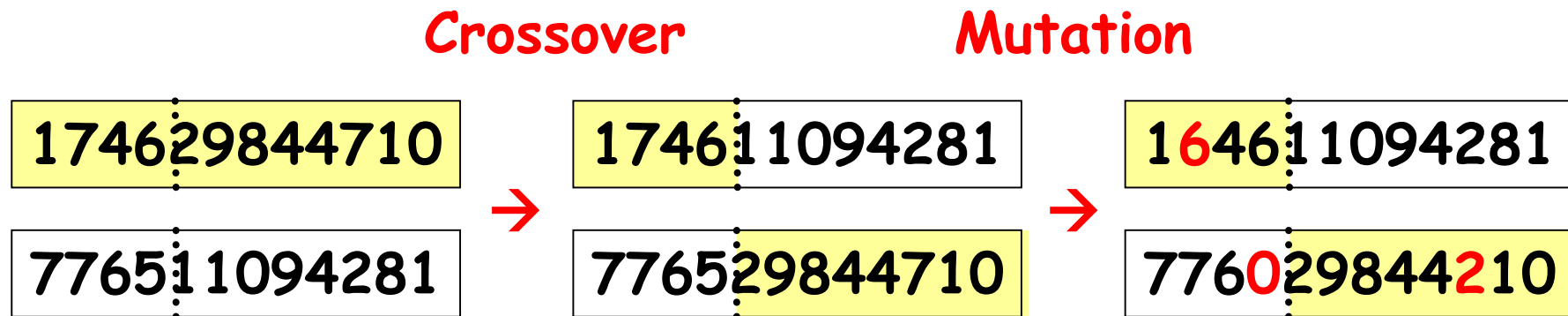
- Keeps more previous states in memory
 - Simulated annealing just kept one previous state in memory.
 - This search **keeps k states in memory.**
 - randomly generate **k** initial states
 - if any state is a goal, terminate
 - else, generate all successors and select best **k**
 - repeat

Local Beam Search



Genetic Algorithms

- Start with random population of states
 - Representation serialized (ie. strings of characters or bits)
 - States are ranked with “fitness function”
- Produce new generation
 - Select random pair(s) using probability:
 - probability \sim fitness
 - Randomly choose “crossover point”
 - Offspring mix halves
 - Randomly mutate bits



Genetic Algorithm

- Given: population **P** and fitness-function **f**
- repeat
 - **newP** \leftarrow empty set
 - for $i = 1$ to **size(P)**
 - $x \leftarrow \text{RandomSelection}(\mathbf{P}, \mathbf{f})$
 - $y \leftarrow \text{RandomSelection}(\mathbf{P}, \mathbf{f})$
 - $child \leftarrow \text{Reproduce}(x, y)$
 - if (small random probability) then $child \leftarrow \text{Mutate}(child)$
 - add $child$ to **newP**
 - **P** \leftarrow **newP**
- until some individual is fit enough or enough time has elapsed
- return the best individual in **P** according to **f**

Using Genetic Algorithms

- 2 important aspects to using them
 - 1. How to encode your real-life problem
 - 2. Choice of fitness function
- Research Example
 - I have N variables V_1, V_2, \dots, V_N
 - I want to produce a single number from them that best satisfies my fitness function F
 - I tried linear combinations, but that didn't work
 - A guy named Stan I met at a workshop in Italy told me to try Genetic Programming

Genetic Programming

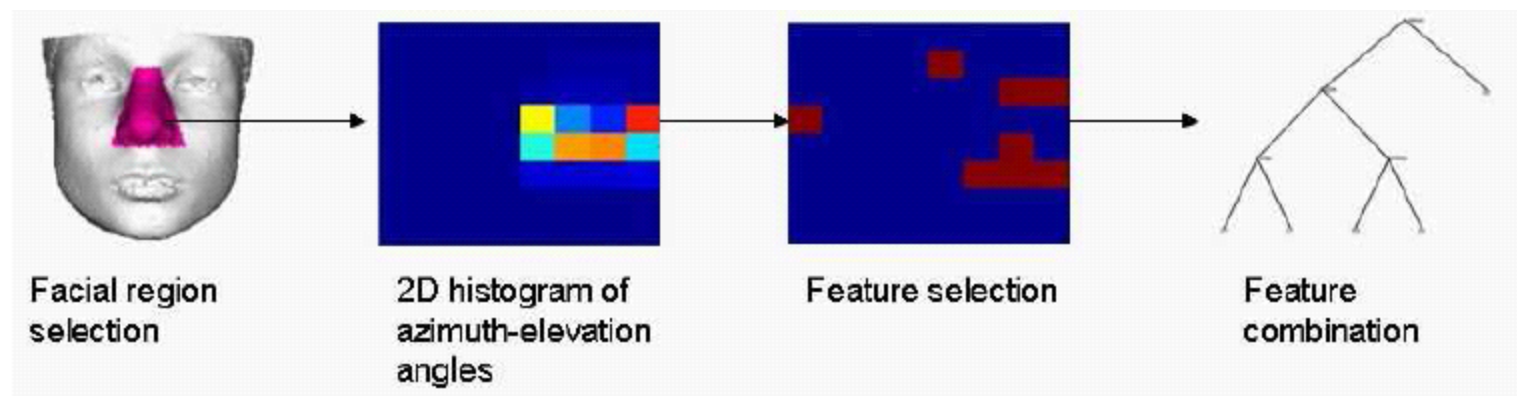
- Like genetic algorithm, but instead of finding the best character string, we want to find the **best arithmetic expression tree**
- The leaves will be the variables and the non-terminals will be arithmetic operators
- It uses the same ideas of crossover and mutation to produce the arithmetic expression **tree that maximizes the fitness function.**

Example: Classification and Quantification of Facial Abnormalities

- Input is 3D meshes of faces
- Disease is **22q11.2 Deletion Syndrome**.
- Multiple different facial abnormalities
- We'd like to assign severity scores to the different abnormalities, so need a single number to represent our analysis of a portion of the face.

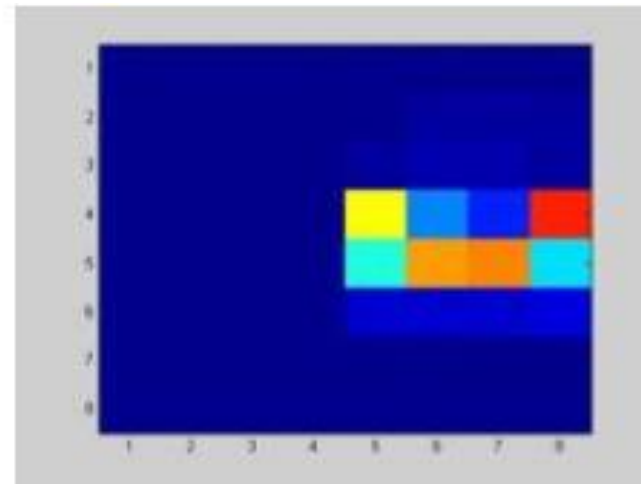
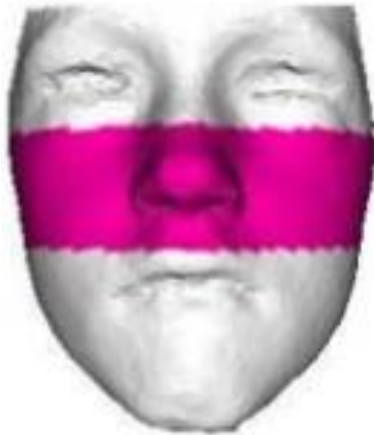
Learning 3D Shape Quantification

- Analyze 22q11.2DS and 9 associated facial features
- **Goal:** **quantify** different shape variations in different facial abnormalities



Learning 3D Shape Quantification - 2D Histogram Azimuth Elevation

- Using azimuth and elevation angles of surface normal vectors of points in selected region

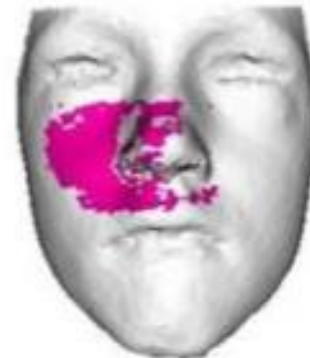
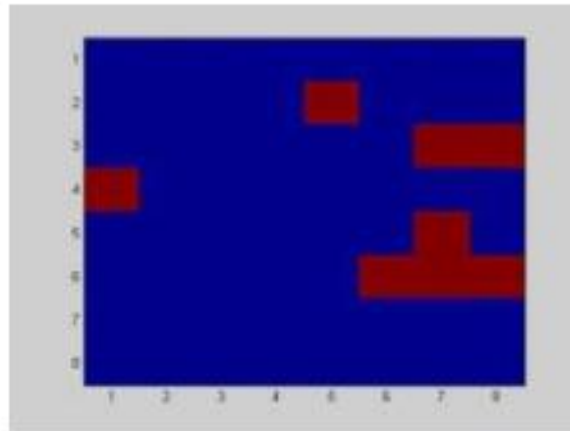


azimuth

elevation

Learning 3D Shape Quantification - Feature Selection

- Determine most discriminative bins
- Use **Adaboost** learning
- Obtain positional information of important region on face

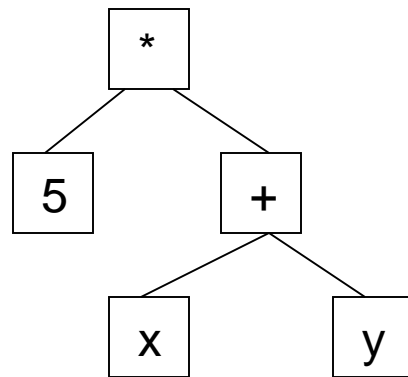


Learning 3D Shape Quantification - Feature Combination

- Use **Genetic Programming** (GP) to evolve mathematical expression
- Start with random population
 - Individuals are evaluated with fitness measure
 - Best individuals reproduce to form new population

Learning 3D Shape Quantification - Genetic Programming

- Individual:
 - Tree structure
 - Terminals e.g variables eg. 3, 5, x, y, ...
 - Function set e.g +, -, *, ...
 - Fitness measure e.g sum of square ...



$$5*(x+y)$$

Learning 3D Shape Quantification - Feature Combination

- 22q11.2DS dataset
 - Assessed by craniofacial experts
 - Groundtruth is union of expert scores
- **Goal:** classify individual according to given facial abnormality

Learning 3D Shape Quantification - Feature Combination

- Individual

- **Terminal:** selected histogram bins

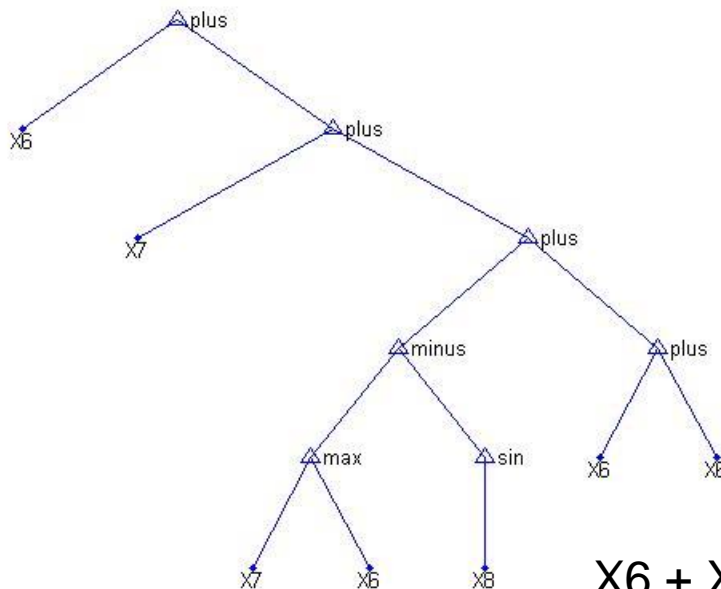
- **Function set:** +, -, *, min, max, sqrt, log, 2x, 5x, 10x

- **Fitness measure:** F1-measure

$$F(\text{prec}, \text{rec}) = \frac{2 \times (\text{prec} \times \text{rec})}{\text{prec} + \text{rec}}$$

$$\text{precision} = \text{TP} / (\text{TP} + \text{FP})$$

$$\text{recall} = \text{TP} / \text{all positives}$$



$$X6 + X7 + (\max(X7, X6) - \sin(X8)) + (X6 + X6)$$

Learning 3D Shape Quantification - Experiment 1

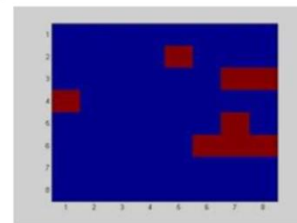
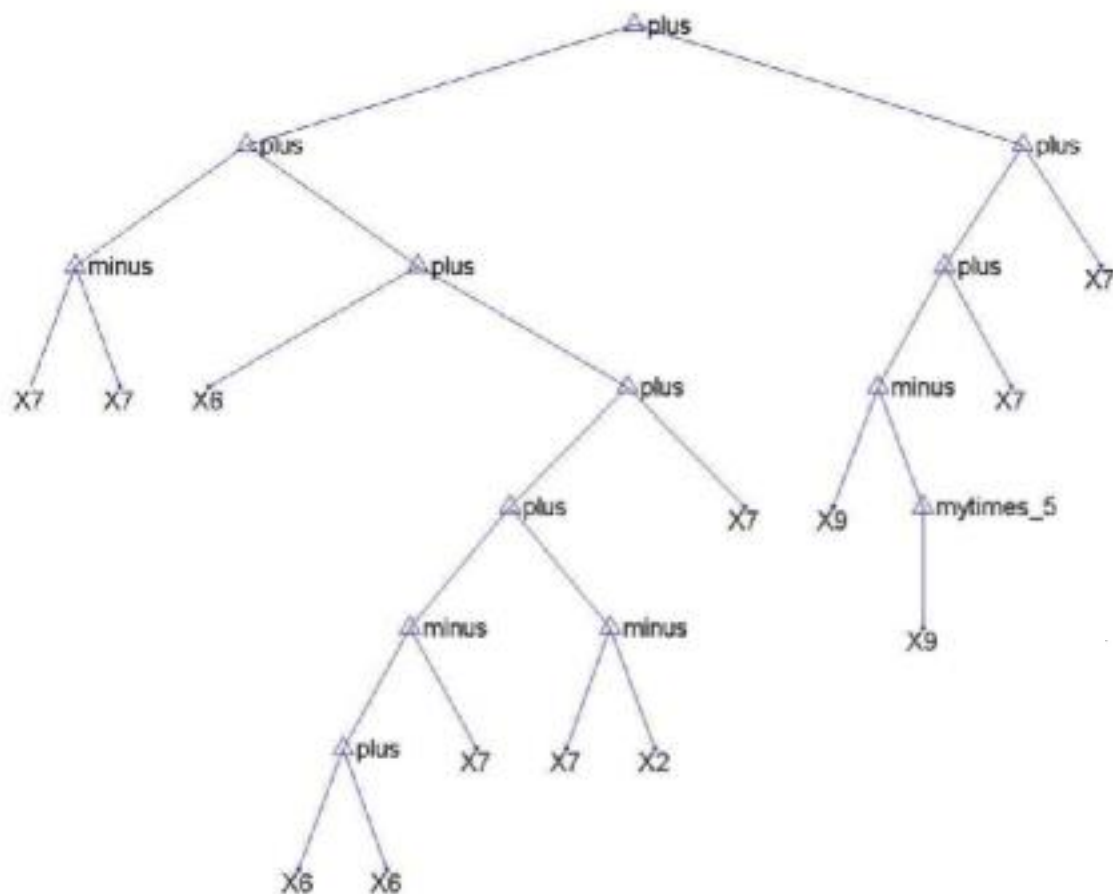
- **Objective:** investigate function sets
 - Combo1 = $\{+, -, *, \min, \max\}$
 - Combo2 = $\{+, -, *, \min, \max, \text{sqrt}, \log_2, \log_{10}\}$
 - Combo3 = $\{+, -, *, \min, \max, 2x, 5x, 10x, 20x, 50x, 100x\}$
 - Combo4 = $\{+, -, *, \min, \max, \text{sqrt}, \log_2, \log_{10}, 2x, 5x, 10x, 20x, 50x, 100x\}$

Learning 3D Shape Quantification - Experiment 1

- Best F-measure out of 10 runs

Facial anomaly	Combo1	Combo2	Combo3	Combo4
Midface Hypoplasia	0.8393	0.8364	0.8527	0.80
Tubular Nose	0.8571	0.875	0.8667	0.8813
Bulbous Nasal Tip	0.8545	0.8099	0.8103	0.7544
Prominent Nasal Root	0.8667	0.8430	0.8571	0.8335
Small Nasal Alae	0.8846	0.8454	0.8454	0.8571
Retrusive Chin	0.7952	0.8000	0.7342	0.7586
Open Mouth	0.9444	0.9714	0.9189	0.9189
Small Mouth	0.6849	0.7568	0.6829	0.7750
Downturned mouth	0.8000	0.7797	0.8000	0.8000

Tree structure for quantifying midface hypoplasia



X_i are the selected histogram bins from an azimuth-elevation histogram of the surface normals of the face.

Learning 3D Shape

Quantification - Experiment 2

- **Objective:** compare local facial shape descriptors

Facial abnormality	Region Histogram	Selected Bins	GP
Midface hypoplasia	0.697	0.721	0.853
Tubular nose	0.701	0.776	0.881
Bulbous nasal tip	0.617	0.641	0.855
Prominent nasal root	0.704	0.748	0.867
Small nasal alae	0.733	0.801	0.885
Retrusive chin	0.658	0.713	0.800
Open mouth	0.875	0.889	0.971
Small mouth	0.694	0.725	0.775
Downturned mouth	0.506	0.613	0.800

Learning 3D Shape Quantification - Experiment 3

- **Objective:** predict 22q11.2DS

Method	F-measure
Quantification vector with SVM	0.709
Quantification vector with Adaboost	0.721
Quantification vector with GP	0.821
Global saliency map	0.764
Selected bins of global saliency map	0.9
Global 2D histogram	0.79
Selected bins of global 2D histogram	0.9
Selected bins of global saliency map with GP	0.96
Selected bins of global 2D histogram with GP	0.92
Expert's median	0.68

Local Search in Continuous Spaces

- Given a continuous state space
 $S = \{(x_1, x_2, \dots, x_N) \mid x_i \in \mathbb{R}\}$
- Given a continuous objective function
 $f(x_1, x_2, \dots, x_N)$
- The **gradient** of the objective function is a vector $\nabla f = (\partial f / \partial x_1, \partial f / \partial x_2, \dots, \partial f / \partial x_N)$
- The gradient gives the magnitude and direction of the steepest slope at a point.

Local Search in Continuous Spaces

- To find a maximum, the basic idea is to set $\nabla f = 0$
- Then updating of the current state becomes $x \leftarrow x + \alpha \nabla f(x)$
where α is a small constant.
- Theory behind this is taught in numerical methods classes.
- Your book suggests the Newton-Raphson method. Luckily there are packages.....

Computer Vision Pose Estimation Example

I have a 3D model of an object and an image of that object.

I want to find the pose: the position and orientation of the camera.



Figure 18: The pose computed from six point correspondence using the algorithm described in Section 5.2.



Figure 19: The pose computed from an ellipse-circle correspondence using the algorithm described in Section 5.6.

and

$$R = \begin{pmatrix} s^2 + l^2 - m^2 - n^2 & 2(lm - sn) & 2(ln + sm) \\ 2(lm + sn) & s^2 - l^2 + m^2 - n^2 & 2(mn - sl) \\ 2(ln - sm) & 2(mn + sl) & s^2 - l^2 - m^2 + n^2 \end{pmatrix}. \quad (52)$$

Powell's method [19] in the seven-dimensional space of the pose solution (four quaternion parameters and the translation t) is used, along with the constraint that the sum of the squares of the quaternion parameters must equal 1, as seen in equation (51). Figure 21 shows an initial pose estimate for a single-object image as



Figure 20: The pose computed from six point correspondence and one ellipse-circle correspondence using the generalized methodology.

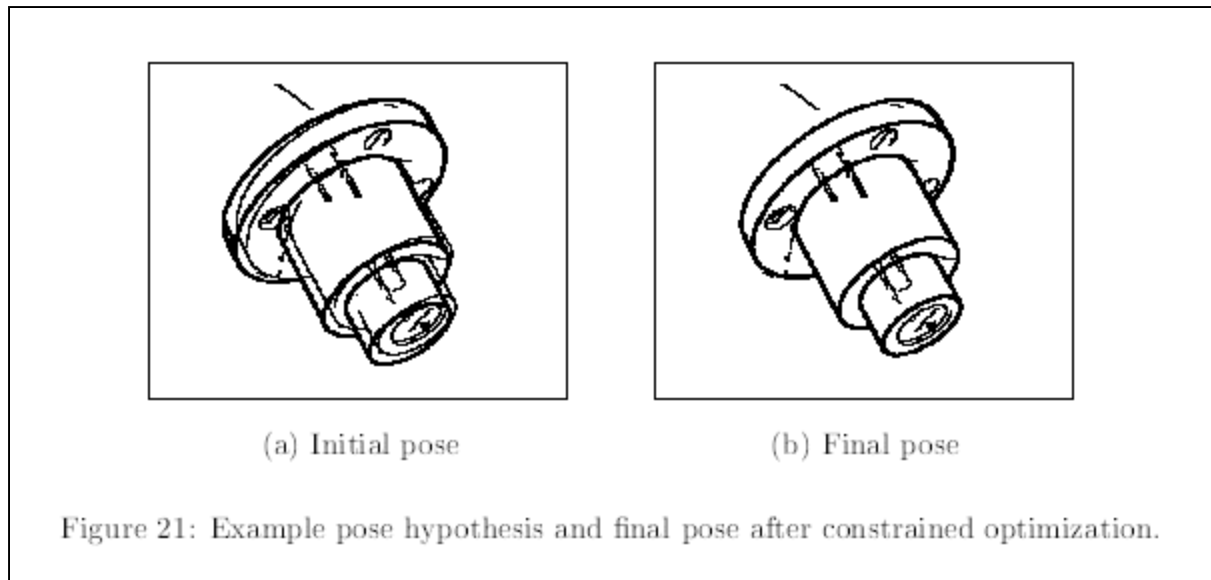
pose from 6 point correspondences

pose from ellipse-circle correspondence

pose from both 6 points and ellipse-circle correspondences

Computer Vision Pose Estimation Example

Initial pose from points/ellipses and final pose after optimization.



- The optimization was searching a 6D space: $(x, y, z, \theta_x, \theta_y, \theta_z)$
- The fitness function was how well the projection of the 3D object lined up with the edges on the image.

Searching with Nondeterministic Actions

Please read it in the text.

- Vacuum World (actions = {left, right, suck})

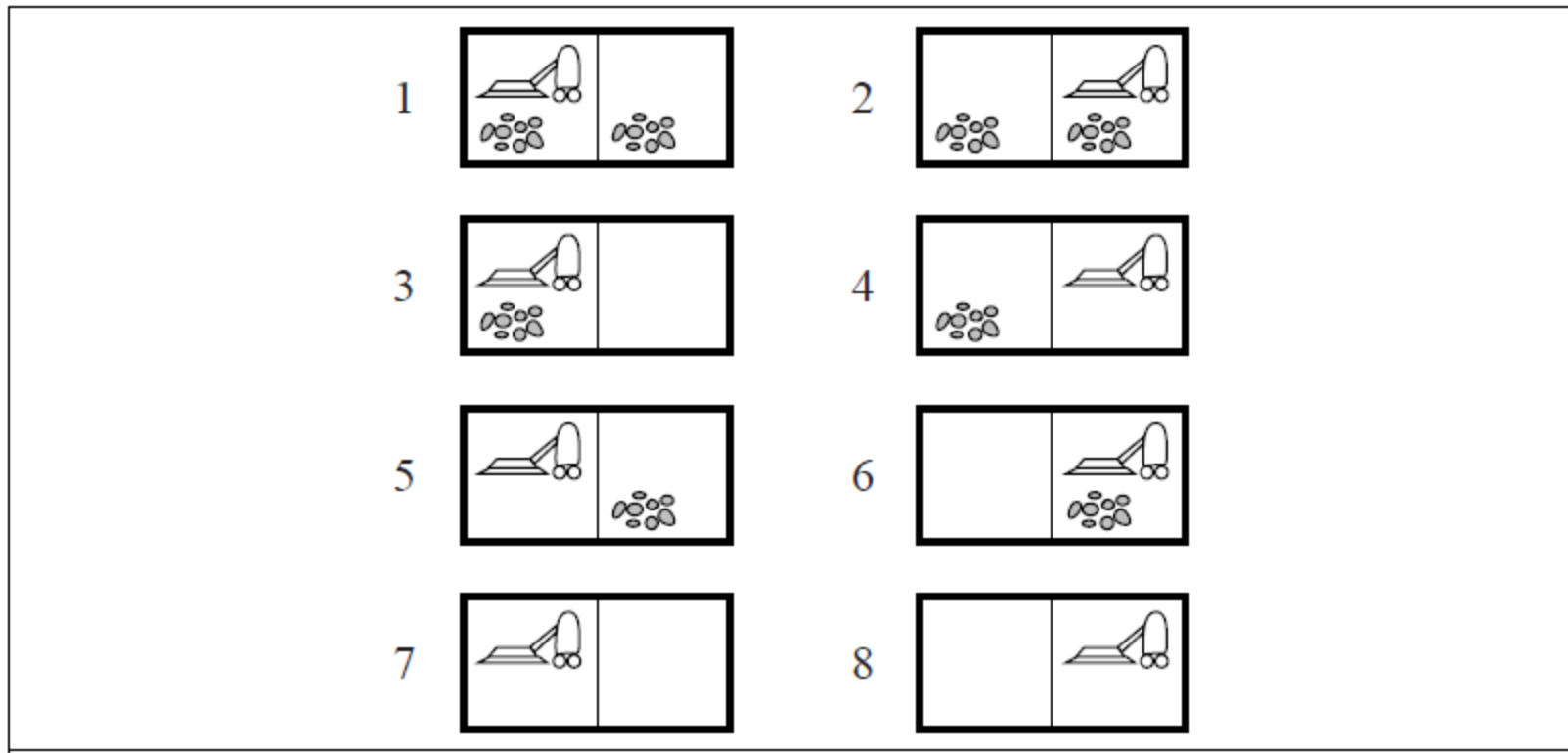


Figure 4.9 The eight possible states of the vacuum world; states 7 and 8 are goal states.

Searching with Partial Observations

- The agent does not always know its state!
- Instead, it maintains a **belief state: a set of possible states it might be in.**
- **Example:** a robot can be used to build a map of a hostile environment. It will have sensors that allow it to “see” the world.

Online Search Problems

- **Active agent**
 - executes actions
 - acquires percepts from sensors
 - deterministic and fully observable
 - has to perform an action to know the outcome
- **Examples**
 - Web search
 - Autonomous vehicle