RN, Chapter
4.3 – 4.4; 7.6

Local and Stochastic Search

Some material based on D Lin, B Selman

Search Overview

- Introduction to Search
- Blind Search Techniques
- Heuristic Search Techniques
- Constraint Satisfaction Problems
- Local Search (Stochastic) Algorithms
  - Motivation
  - Hill Climbing
  - Issues
  - SAT … Phase Transition, GSAT, …
  - Simulated Annealing, Tabu, Genetic Algorithms
- Game Playing search

A Different Approach

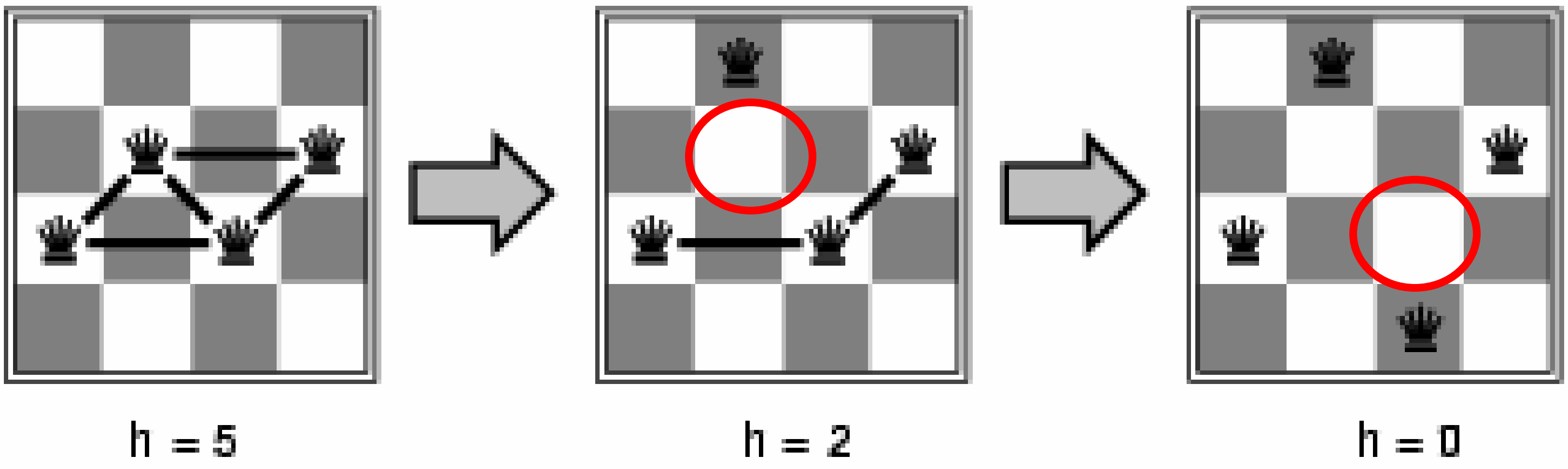
- So far: systematic exploration:
  - Explore full search space
    (possibly) using principled pruning (A*, ...)
- Best such algorithms (IDA*) can handle
  - 10^{100} states; ≈500 binary-valued variables
    (ballpark figures only!)
- but... some real-world problem have 10,000 to 100,000 variables; $10^{30,000}$ states
- We need a completely different approach:
  - Local Search Methods
  - Iterative Improvement Methods

Local Search Methods

- Applicable when seeking Goal State …& don't care how to get there
- E.g.,
  - *N-queens, map coloring, VLSI layout, planning, scheduling, TSP, time-tabling, …*
- Many (most?) real Operations Research problems are solved using local search!
  - E.g., schedule for Delta airlines, …

Example#1: 4 Queen

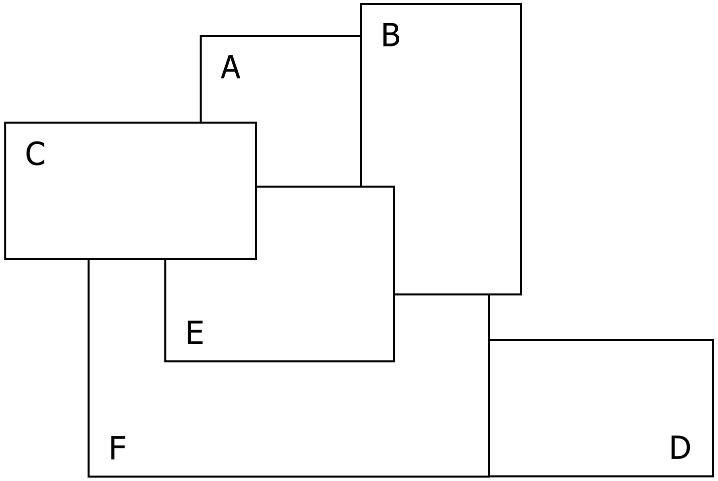
- **States:** 4 queens in 4 columns (256 states)
- **Operators:** move queen in column
- **Goal test:** no attacks
- **Evaluation:** h(n) = number of attacks



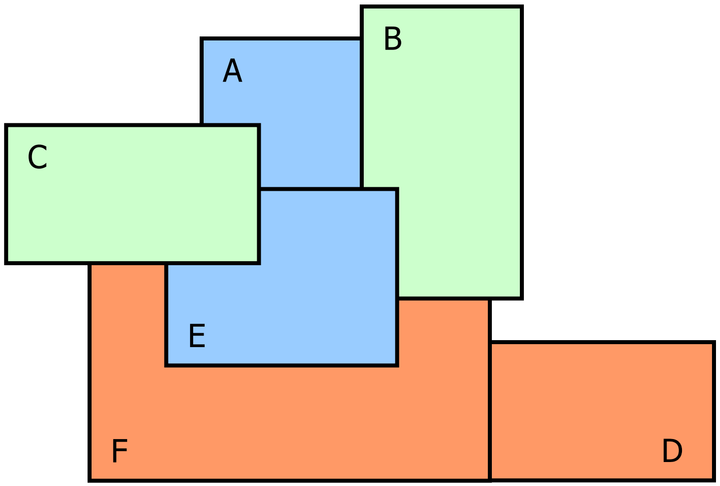
Example#2: Graph Coloring

1. Start with random coloring of nodes
2. Change color of one node
   to reduce #conflicts

Graph Coloring Example

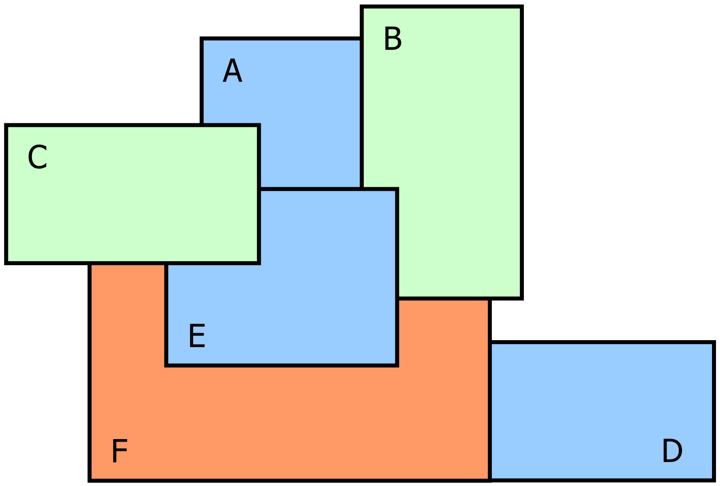


Graph Coloring Example



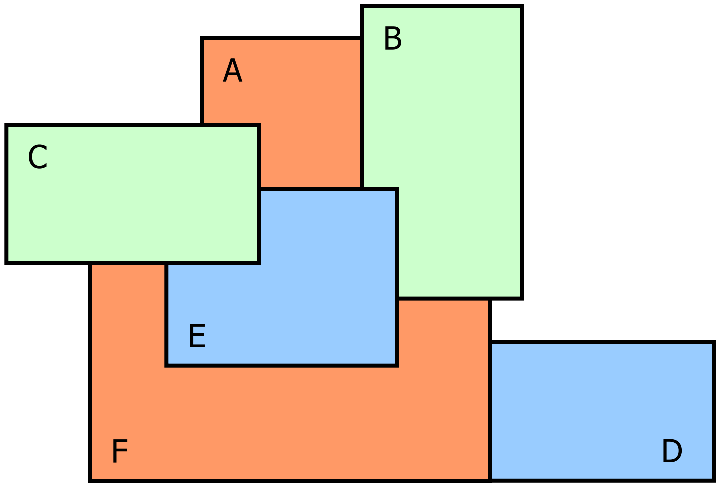
Iteration	A	B	C	D	E	F	# conflicts
1	b	g	g	r	b	r	2 {AE, DF}

Graph Coloring Example



Iteration	A	B	C	D	E	F	# conflicts
1	b	g	g	r	b	r	2 {AE, DF}
2	b	g	g	B	b	r	1 {AE}

Graph Coloring Example



Iteration	A	B	C	D	E	F	# conflicts
1	b	g	g	r	b	r	2 {AE, DF}
2	b	g	g	b	b	r	1 {AE}
3	R	g	g	b	b	r	0 {}

“Local Search”

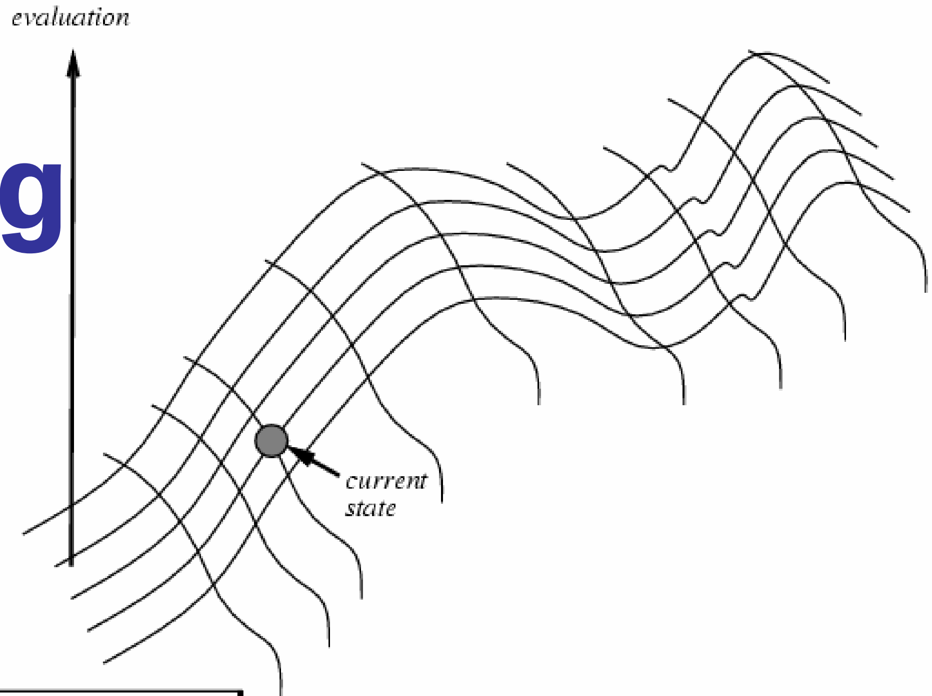
1. Select (random) initial state
   (initial guess at solution)
2. While GoalState not found (& more time)
   - Make *local modification* to improve current state

Requirements:

- Generate a random (probably-not-optimal) guess
- Evaluate quality of guess
- Move to other states
  (well-defined neighborhood function)

… and do these operations quickly…

Hill-Climbing




```
function HILL-CLIMBING(problem) returns a solution state
   inputs: problem, a problem
   static: current, a node
           next, a node

   current ← MAKE-NODE(INITIAL-STATE[problem])
   loop do
       next ← a highest-valued successor of current
       if VALUE[next] < VALUE[current] then return current
       current ← next
   end
```

If Continuous ....

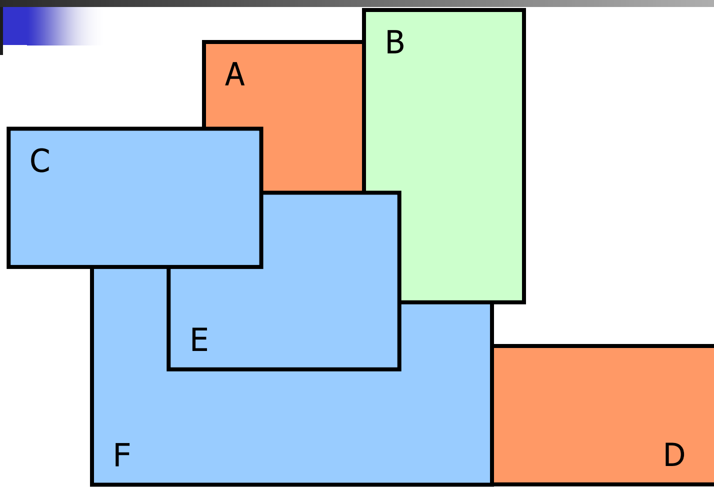
- Situation
  - State = $\langle v_1, \ldots, v_n \rangle \in \Re^n$
  - quality $h : \Re^n \mapsto \Re$

    $h(\text{state}) \in \Re$
- To find optimum:

Guess random initial state $\vec{v}^0 \in \Re^n$
While $\exists i \left.\frac{\partial h(X)}{\partial X_i}\right\vert_{X=\vec{v}} \neq 0$ do
For $i = 1..n$
$\vec{v}_i := \vec{v}_i - \eta \left.\frac{\partial h}{\partial X_i}\right\vert_{X=\vec{v}}$
Return $\vec{v}$

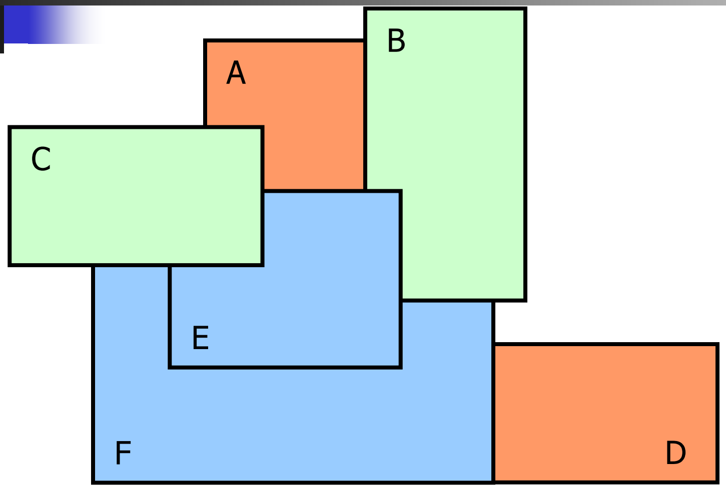
- May have other termination conditions
- If η too small: very slow
- If η too large: overshoot
- May have to approximate derivatives from samples

But …



Iteration	A	B	C	D	E	F	# conflicts
1	r	g	b	r	b	b	3 {CE, CF, EF}

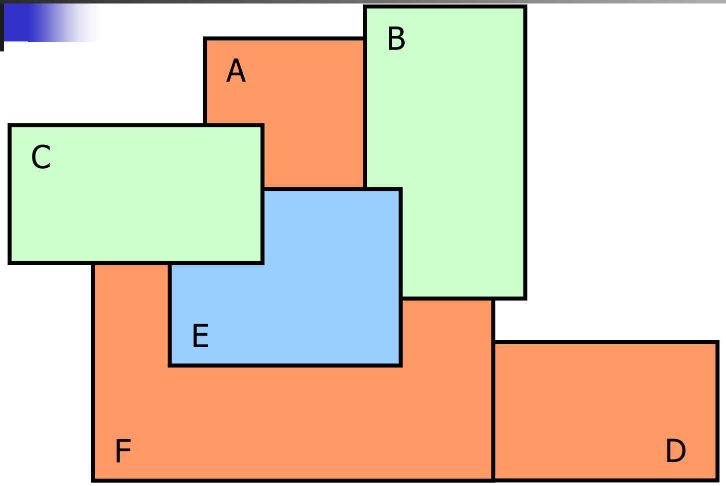
But …



- Pure "Hill Climbing" will not work!
- Need "Plateau Walk"

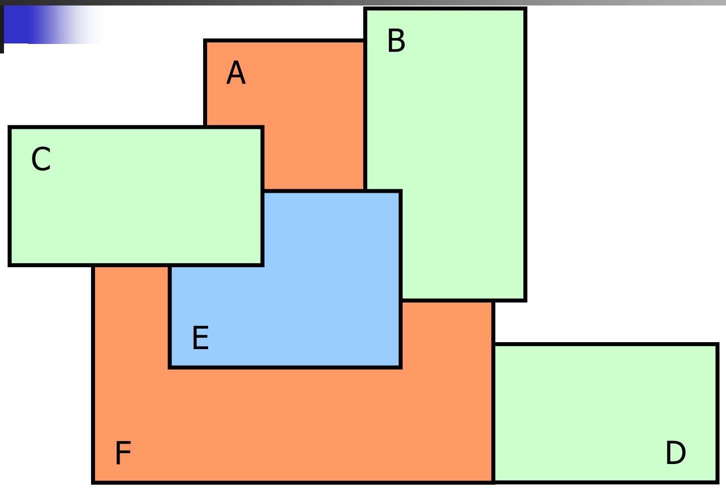
Iteration	A	B	C	D	E	F	# conflicts
1	r	g	b	r	b	b	3 {CE, CF, EF}
2	r	g	G	r	b	b	1 {EF}

But …



Iteration	A	B	C	D	E	F	# conflicts
1	r	g	b	r	b	b	3 {CE, CF, EF}
2	r	g	G	r	b	b	1 {EF}
3	r	g	g	r	b	R	1 {DF}

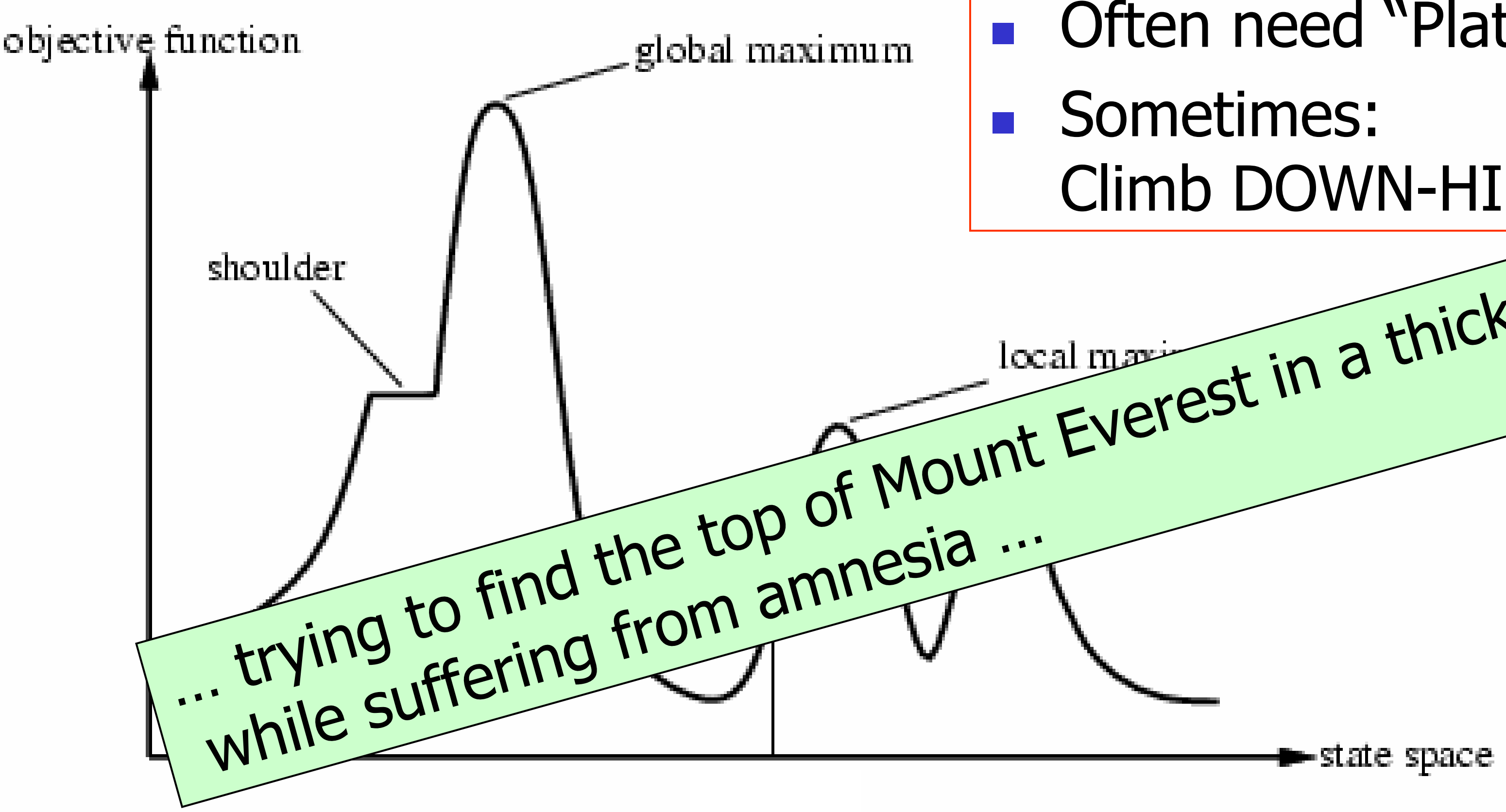
But …



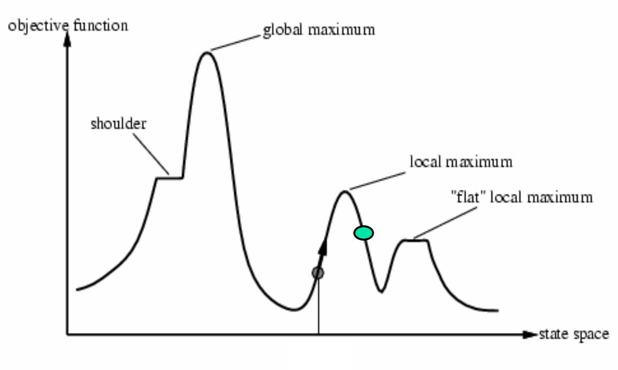
Iteration	A	B	C	D	E	F	# conflicts
1	r	g	b	r	b	b	3 {CE, CF, EF}
2	r	g	G	r	b	b	1 {EF}
3	r	g	g	r	b	R	1 {DF}
4	r	g	g	G	b	r	0 {}

Problems with Hill Climbing

- Pure “Hill Climbing” does not always work!
- Often need “Plateau Walk”
- Sometimes:
  Climb DOWN-HILL!



Problems with Hill Climbing



- Foothills / Local Optimal:
  No neighbor is better, but not at global optimum
  - Maze: may have to move AWAY from goal to find best solution
- Plateaus: All neighbors look the same.
  - 8-puzzle: perhaps no action will change # of tiles out of place
- Ridge: going up only in a narrow direction.
  - Suppose no change going South, or going East, but big win going SE
- Ignorance of the peak: Am I done?

Issues

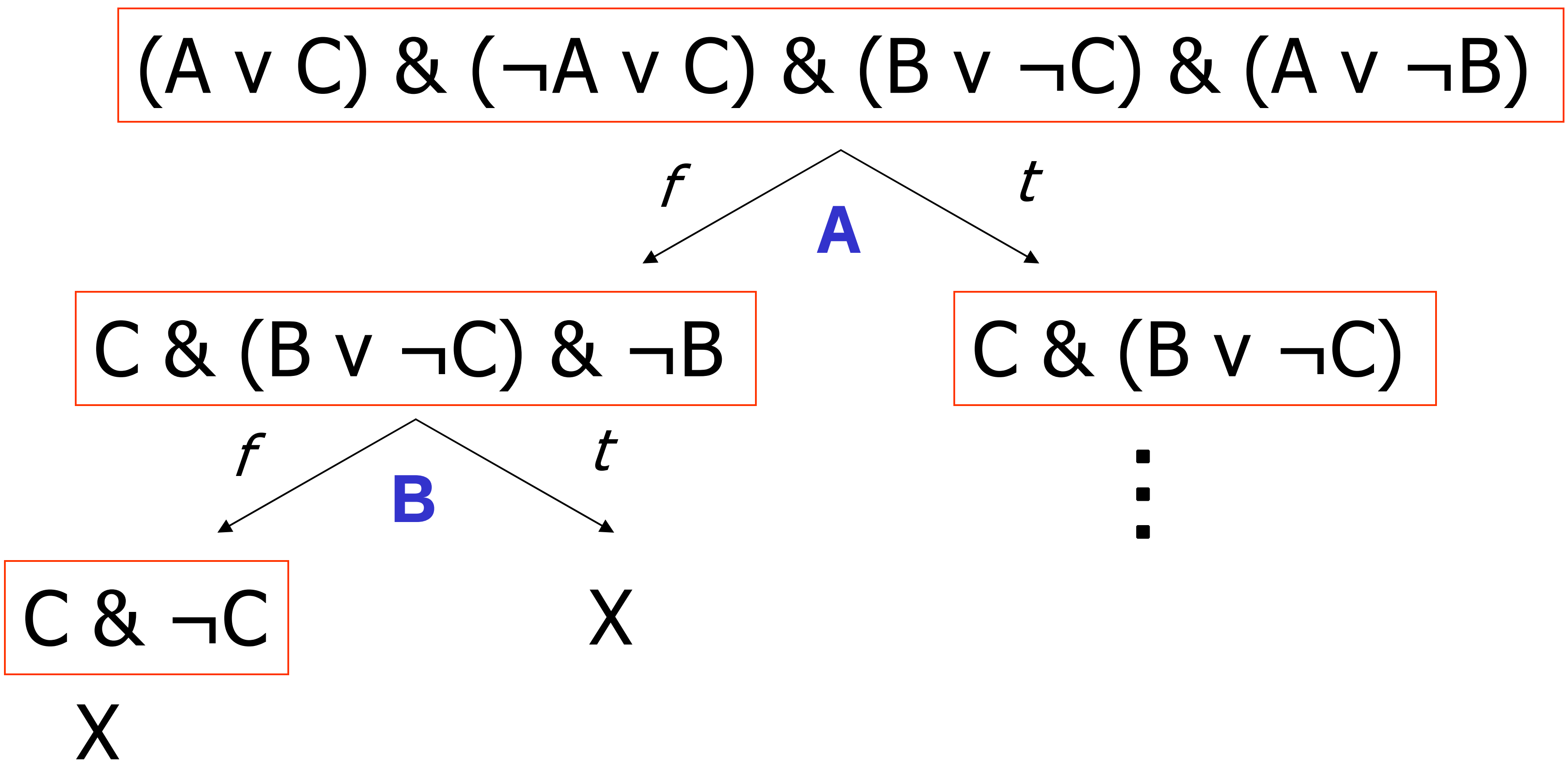
Goal is to find GLOBAL optimum.

1. How to avoid LOCAL optima?
2. How long to *plateau walk*?
3. When to stop?
4. Climb down hill? When?

Local Search Example: SAT

- Many real-world problems ≈ propositional logic
  (A v B v C) & (¬B v C v D) & (A v ¬C v D)
- Solved by finding truth assignment to (A, B, C, …) that satisfy formula
- Applications
  - planning and scheduling
  - circuit diagnosis and synthesis
  - deductive reasoning
  - software testing
  - …

Obvious Algorithm



Satisfiability Testing

Davis-Putnam Procedure (1960)

- Backtracking depth-first search (DFS) through space of truth assignments
(+ unit-propagation)
- *fastest* sound + complete method
  - … best-known systematic method *…*
- … but …
∃ classes of formulae where it scales badly…

Greedy Local Search

- Why not just HILL-CLIMB??
- Given
  - formula: φ = (A v C) & (¬A v C) & (B v ¬C)
  - assignment: σ = {–a, –b, +c }

  Score(φ, σ) = #clauses unsatisfied … = 0
- Just flip variable that helps most!

A	B	C	(A v C)	& (¬A v C)	& (B v ¬C)	Score
0	0	0	x	+	+	1
0	0	+	+	+	x	1
0	+	+	+	+	+	0

Greedy Local Search: GSAT

1. Guess random truth assignment
2. Flip value assigned to the variable that yields the greatest # of satisfied clauses.
   (Note: Flip even if no improvement)
3. Repeat until all clauses satisfied, or have performed "enough" flips
4. If no sat-assign found,
   repeat entire process,
   starting from a new initial random assgmt

A	B	C	(A v C) & (¬A v C) & (B v ¬C)			Score
0	0	0	x	+	+	1
0	0	+	+	+	x	1
0	+	+	+	+	+	0

Does GSAT Work?

- First intuition:
  GSAT will get stuck in local minima,
  with a few unsatisfied clauses.
- Very bad…
  "almost satisfying assignments" are worthless
  (Eg, plan with one "magic" step is useless)
  ...ie, NOT optimization problem
- Surprise: GSAT often found global minimum!
  Ie, satisfying assignment!
  10,000+ variables; 1,000,000+ constraints!
- No good theoretical explanation yet…

GSAT vs. DP on Hard Random Instances

form. vars	GSAT m.flips	GSAT retries	GSAT time	Davis-Putnam choices	Davis-Putnam depth	Davis-Putnam time
50	250	6	0.5 *sec*	77	11	1 *sec*
70	350	11	1 *sec*	42	15	15 *sec*
100	500	42	6 *sec*	10^3	19	3 *min*
120	600	82	14 *sec*	10^5	22	18 *min*
140	700	53	14 *sec*	10^6	27	5 *hrs*
150	1500	100	45 *sec*	—	—	—
200	2000	248	3 *min*	—	—	—
300	6000	232	12 *min*	—	—	—
500	10000	996	2 *hrs*	10^{30}	> 100	10^{19} *yrs*

Notes: Define “Hard” later
Only “satisfiable” formulae
(else GSAT does not terminate)

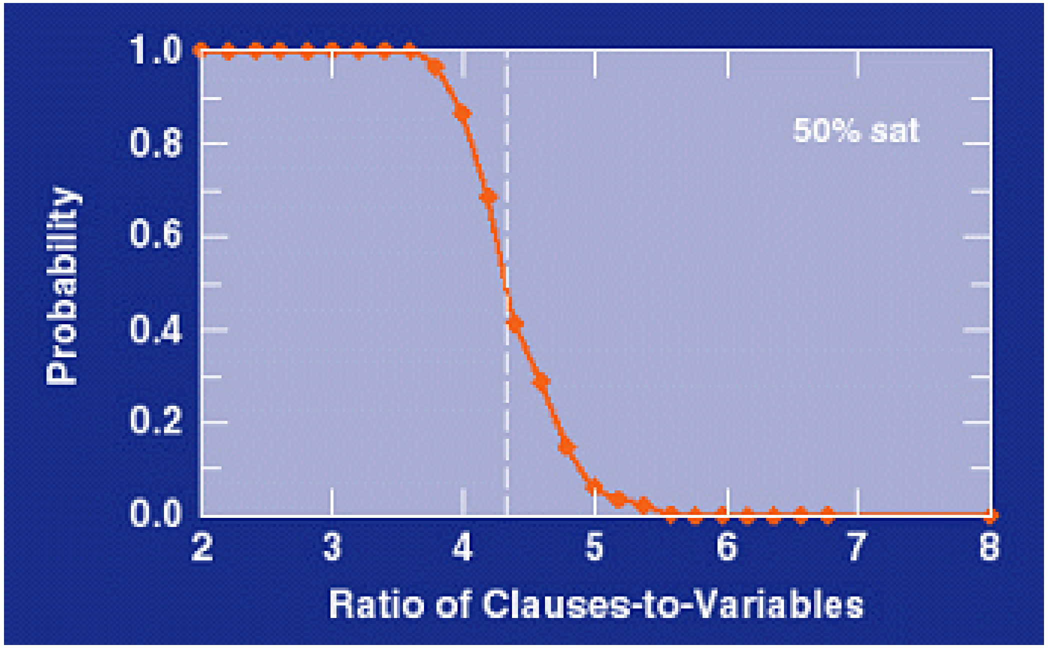
Systematic vs. Stochastic

- Systematic search:
  - DP systematically checks all possible assignments
  - Can determine if the formula is unsatisfiable
- Stochastic search:
  - Once we find it, we're done!
  - Guided random search approach
  - Can't determine unsatisfiability

What Makes a SAT Problem Hard?

- Randomly generate formula φ with
  - *n* variables; *m* clauses with *k* variables each
  - #possible_clauses = $\binom{n}{k} \times 2^k$
- Will φ be satisfied??
  - If n << m: ??
  - If n >> m: ??

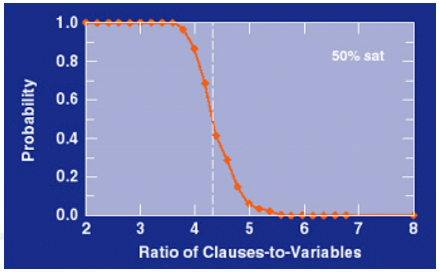
Phase Transition



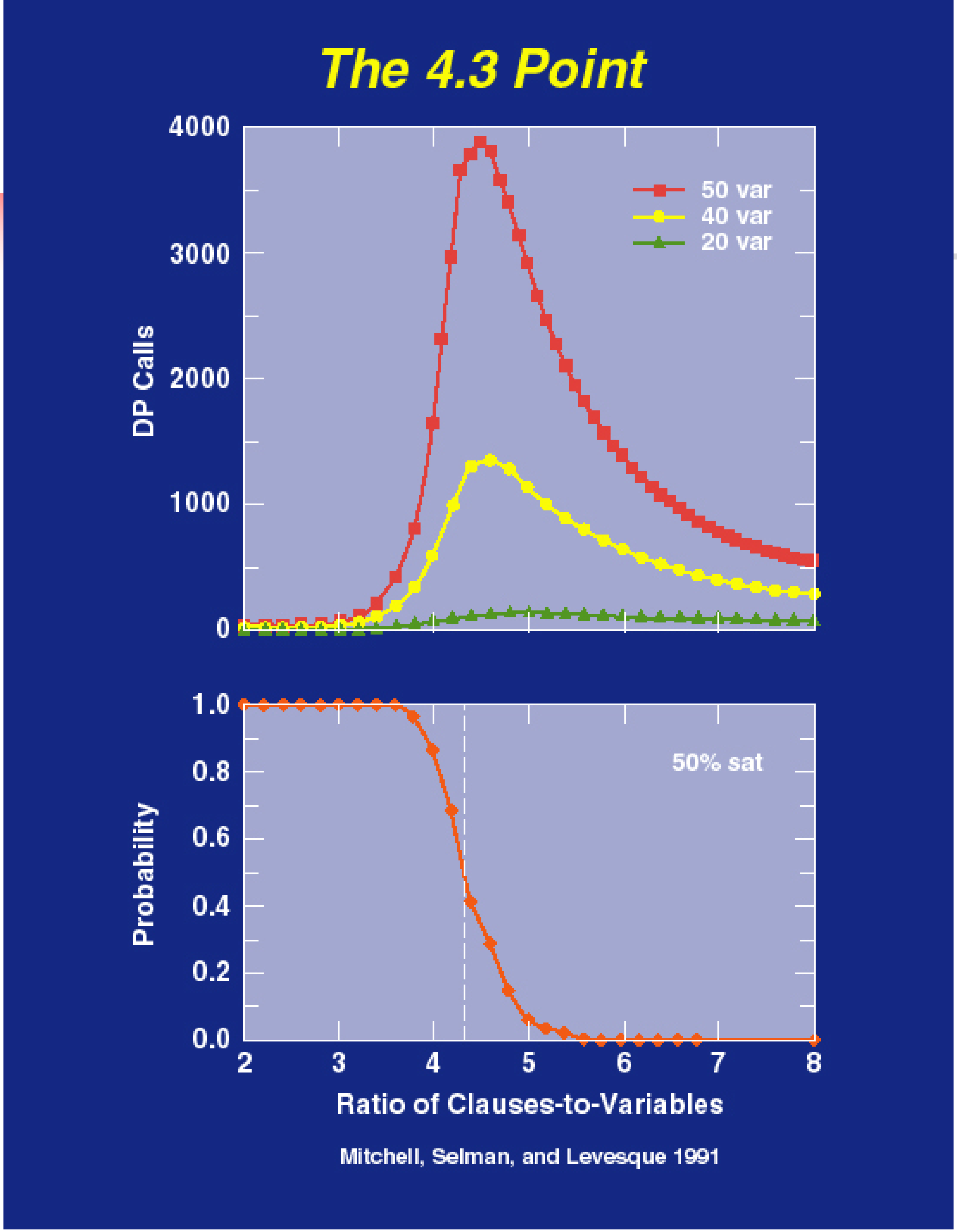
For 3-SAT

- $m / n < 4.2$, under constrained $\Rightarrow$ **nearly all formulae sat.**
- $m / n > 4.3$, over constrained $\Rightarrow$ **nearly all formulae unsat.**
- $m/ n \sim 4.26$, critically constrained $\Rightarrow$ need to search

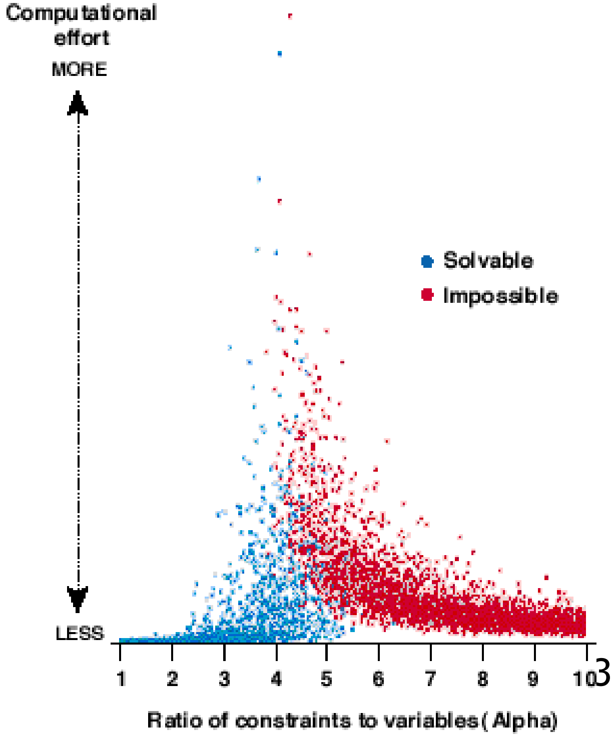
Phase Transition



- Under-constrained problems are easy: just guess an assignment
- Over-constrained problems are easy: just say "unsatisfiable"
  (… often easy to verify using Davis-Putnam)
- At $m/n \approx 4.26$, $\exists$ ***phase transition*** between these two different types of easy problems.
  - This transition sharpens as n increases.
- For large n, hard problems are extremely rare (in some sense)



- Hard problems are at Phase Transition!!



Improvements to Basic Local Search

- Issues:
  - How to move more quickly to successively better plateaus?
  - Avoid “getting stuck” / **local minima**?
- Idea: Introduce uphill moves (“noise”) to escape from plateaus/local minima
- Noise strategies:
  1. Simulated Annealing
     - Kirkpatrick et al. 1982; Metropolis et al. 1953
  2. Mixed Random Walk
     - Selman and Kautz 1993

Simulated Annealing

Pick a random variable

If flip improves assignment: do it.

Else flip with probability $p = e^{-\delta/T}$ (go the wrong way)

- δ = #of additional clauses becoming unsatisfied
- T = "temperature"
  - Higher temperature = greater chance of wrong-way move
  - Slowly decrease T from high temperature to near 0
- Q: What is p as T tends to infinity?

  ... as T tends to 0?

  For $\delta = 0$?

Simulated Annealing Algorithm

current, *next*: nodes/states

T: "temperature" controlling prob. of downward steps

schedule: mapping from time to "temperature"

h: heuristic evaluation function

current $\leftarrow$ initial state
for $t \leftarrow 1..\infty$ do
    $T \leftarrow$ *schedule*$[t]$
    if $T = 0$ then return *current*
    next $\leftarrow$ randomly selected successor of *current*
    $\Delta E \leftarrow h(\textit{next}) - h(\textit{current})$
    if $\Delta E > 0$ then *current* $\leftarrow$ *next*
    else *current* $\leftarrow$ *next* only with probability $e^{\Delta E/T}$

Notes on SA

- Noise model based on statistical mechanics
  - Introduced as analogue to physical process of growing crystals
  - *Kirkpatrick et al. 1982*; *Metropolis et al. 1953*
- Convergence:
  1. W/ exponential schedule, will converge to global optimum
  2. No more-precise convergence rate
     (Recent work on rapidly mixing Markov chains)
- Key aspect: **upwards / sideways** moves
  - Expensive, but (if have enough time) can be best
- Hundreds of papers/ year;
  - Many applications: VLSI layout, factory scheduling, ...

Pure WalkSat

PureWalkSat(formula)

Guess initial assignment

While (unsatisfied) do

Select unsatisfied clause $c = \pm X_i \vee \pm X_j \vee \pm X_k$

Select variable *v* in unsatisfied clause *c*

Flip *v*

Example:

Eg: $(A \vee B)$ & $(\neg A \vee C)$ & $(\neg B \vee \neg D)$ & $\ldots$

A	B	C	D	$\ldots$
0	0	0	+	$\ldots$

Clause $(A \vee B)$ not satisfied.
so flip either A or B... say A

A	B	C	D	$\ldots$
+	0	0	+	$\ldots$

$(A \vee B)$ now satisfied.
...but $(\neg A \vee C)$ is now NOT satisfied!

Mixing Random Walk with Greedy Local Search

```
MixedWalkSat_p( formula )
    Guess initial assignment
    While unsatisfied do
      W/ prob p, walk
          (flip var in an unsatisfied clause)
      W/ prob 1 − p, greedy
          (flip var producing fewest unsatisfied clauses)
```

- Usual issues:
  - Termination conditions
  - Multiple restarts
- Determine value of *p empirically*
  … finding best setting for problem class

Finding the best value of *p*

- Let
  - *Q[p, c]* be *quality* of using WalkSat[p] on problem c

Q[p, c] = Time to return answer, or
= 1 if WalkSat[p] returns (correct) answer within 5mins and 0 otherwise, or
= … perhaps some combination of both …

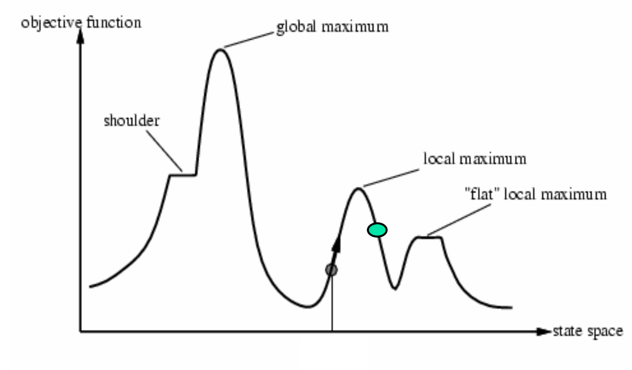
- For each $p \in \{ 0.1, 0.2, \ldots, 0.9 \}$:
  - $QQ[p] = \sum_{c \in S} Q[p, c]$
- Set $p^* = \text{argmax}_p QQ[p]$

Experimental Results: Hard Random 3CNF

	GSAT				Simul. Ann.	
	basic		walk			
vars	time	eff.	time	eff.	time	eff.
100	.4	.12	.2	1.0	.6	.88
200	22	.01	4	.97	21	.86
400	122	.02	7	.95	75	.93
600	1471	.01	35	1.0	427	.3
800	*	*	286	.95	*	*
1000	*	*	1095	.85	*	*
2000	*	*	3255	.95	*	*

- Time in seconds (SGI Challenge)
- Effectiveness: prob. that random initial assignment leads to a solution
- Complete methods, such as DP, up to 400 variables
- Mixed Walk … better than Simulated Annealing
  - better than Basic GSAT
  - better than Davis-Putnam

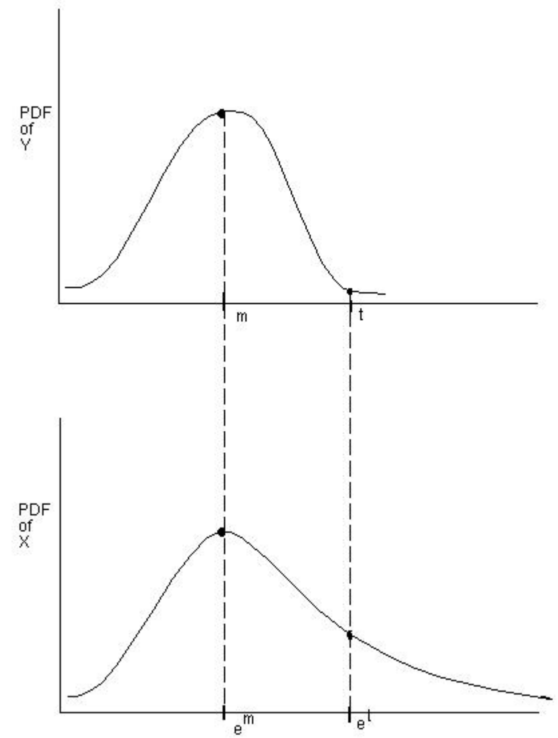
Overcoming Local Optima and Plateaus



- ✓ Simulated annealing
- ✓ Mixed-in random walk
- Random restarts
- Tabu search
- Genetic alg/programming
- ...

Random Restarts

- Restart at new random state
 after pre-defined # of local steps.
- Useful with "Heavy Tail" distribution
- Done by GSAT



Tabu Search

- Avoid returning quickly to same state
- Implementation:
  - Keep fixed length queue (tabu list)
  - Add most recent step to queue; drop oldest step
  - Never make step that's on current tabu list
- Example:
  - without tabu:
  - with tabu (length 4):
- Tabu very powerful;
  - competitive w/ simulated annealing or random walk (depending on the domain)

v1
v2
v4
~~v2~~
v10
v11
v1
~~v10~~
v3
...

Genetic Algorithms

- Class of probabilistic optimization algorithms
  - A genetic algorithm maintains a population of candidate solutions for the problem at hand, and makes it evolve by iteratively applying a set of stochastic operators
- Inspired by the biological evolution process
- Uses concepts of “Natural Selection” and “Genetic Inheritance” (Darwin 1859)
- [John Holland, 1975]

Examples: Recipe

To find optimal quantity of three major ingredients (sugar, wine, sesame oil)

- Use an alphabet of 1-9 denoting ounces
- Solutions might be
  - 1-1-1
  - 2-1-4
  - 3-3-1
  - …

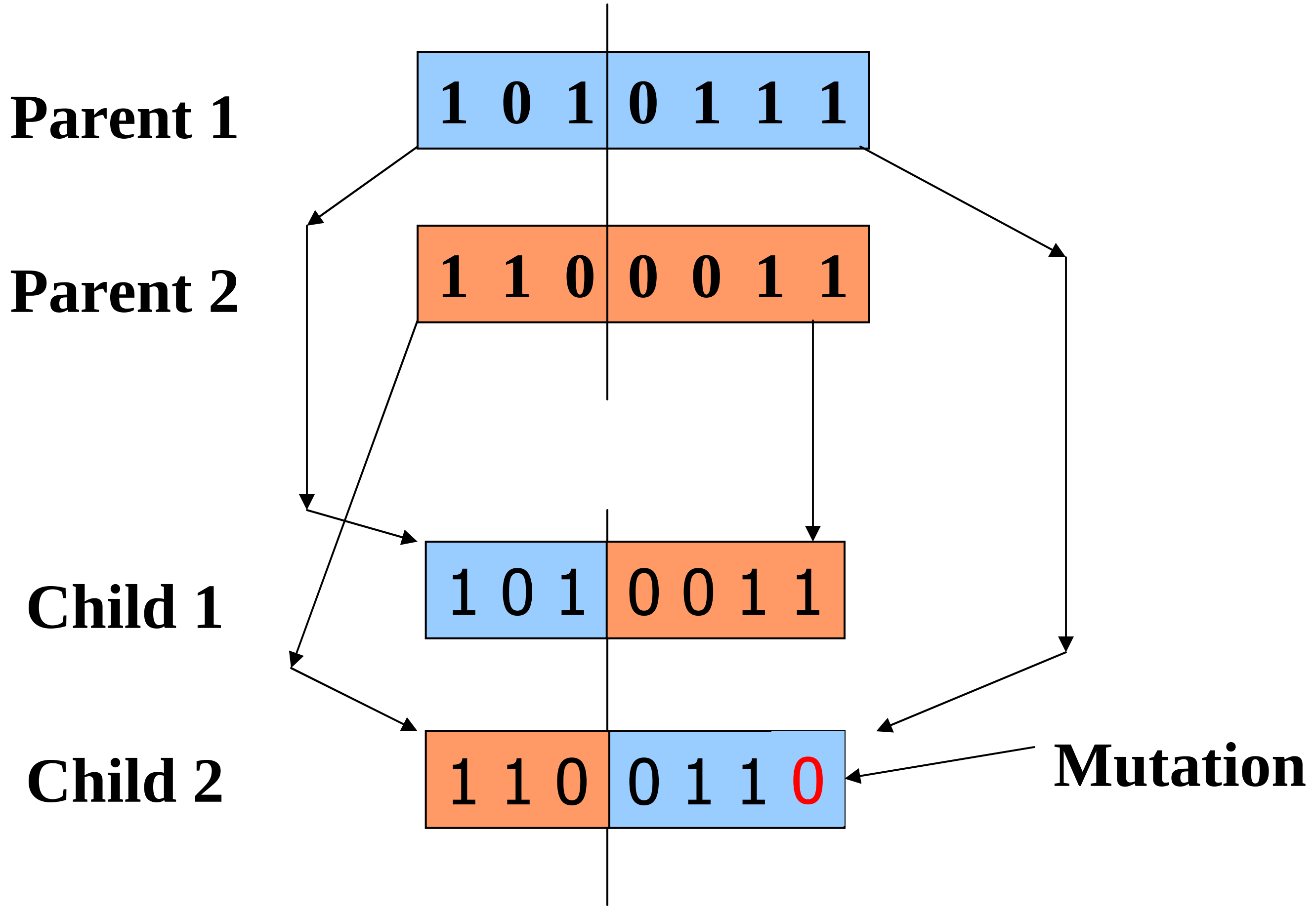
Standard Genetic Algorithm

- Randomly generate an initial population
- For i=1..N
  - Select parents
    and "reproduce" the next generation
  - Evaluate fitness of the new generation
  - Replace some of the old generation
    with the new generation

Stochastic Operators

- **Cross-over**
  - decomposes two distinct solutions
  - then randomly mixes their parts to form novel solutions
- **Mutation**
  - randomly perturbs a candidate solution

Genetic Algorithm Operators Mutation and Crossover



Examples

- Mutation:

  In recipe example, 1-2-3 may be changed to

  - 1-3-3 or
  - 3-2-3
- Parameters to adjust
  - How often?
  - How many digits change?
  - How big?

More examples:

- Crossover

  In recipe example:

  - Parents 1-3-3 & 3-2-3
    Crossover point after the first digit
  - Generate two offspring: 3-3-3 and 1-2-3

Can have one or two point crossover

Local Search Summary

- Surprisingly efficient search technique
- Wide range of applications
- Formal properties elusive
- Intuitive explanation:
  - Search spaces are too large for systematic search anyway. . .
- Area will most likely continue to thrive