Foundations of Artificial Intelligence

CS472/3 — Fall 1999
Lecture #7

Bart Selman

Note: no separate lecture notes for
lect. #6; finished notes from #5.

Slide CS472–1

Today's Lecture

Local Search, Cont.
GSAT, Simulated Annealing,
Genetic Algorithms

Readings: R&N, Chapter 4 & Sect. 20.8.

Slide CS472–2

Local Search Methods

Approach quite general: any NP-complete problem / CSP.

Examples: planning, scheduling, TSP, graph coloring, time-tabling, etc., etc.

Slide CS472–3

Key idea (suprisingly simply):

```
a) Select (random) initial state
     (initial guess at solution)

b) Make local modification to try
        to improve current state.

c) repeat b) till goal state found
        (or out of time).
```

Slide CS472–4

Example

A wide variety of key CS problems can be translated into a propositional logical formalization

e.g., $(A \lor B \lor C) \land (\neg B \lor C \lor D) \land (A \lor \neg C \lor D)$

and solved by finding a truth assignment to the propositional variables (A, B, C,...) that makes it true, i.e., a *model*.

If a formula has a model, we say that it is "satisfiable".

Special kind of CSP.

Greedy Local Search — GSAT

Begin with a random truth assignment (assume CNF).
Flip the value assigned to the variable that
yields the *greatest number of satisfied clauses.*
(Note: Flip even if there is no improvement.)
Repeat until a model is found, or have performed
a specified maximum number of flips.
If a model is still not found, repeat the entire process,
starting from a different initial random assignment.

form.	GSAT			Davis-Putnam		
vars	m.flips	retries	time	choices	depth	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*

Slide CS472–7

Improvements to Basic Local Search

Issue: How to move more quickly to successively lower plateaus?

Avoid getting "stuck" / **local minima**.

Idea: introduce uphill moves ("noise") to escape from long plateaus (or true local minima)

Noise strategies:

a) Simulated Annealing

Kirkpatrick *et al.* 1982; Metropolis *et al. 1953*

b) Mixed Random Walk

Selman and Kautz 1993

Slide CS472–8

Simulated Annealing

•Noise model based on statistical mechanics.

•Pick a random variable

If flip improves assignment: do it.

else flip with probability $p = e^{-\delta/T}$ ("upward").

δ number of additional clauses becoming *unsatisfied*

T = "temperature"

Higher temperature = greater likelihood of upward moves.

Slowly decrease T from high temperature to near zero.

What is p for $T \rightarrow \inf$? For $T \rightarrow 0$? For $\delta = 0$?

Sim. Annealing introduced as analogue to a **physical process**

Way to grow crystals.

Kirkpatrick *et al.* 1982; Metropolis *et al. 1953*

Physical analogy remains elusive.

Can prove that with exponential schedule will converge to global optimum.

Difficult to be more precise about convergence rate.

See recent work on rapidly mixing Markov chains.

Key aspect: **upwards moves / sideways moves.**

Expensive, but if you have time can be best.

(hundreds of papers per year / many applications)

Random Walk

Random walk SAT algorithm:

I Pick random truth assignment.

II Repeat until all clauses satisfied:

Flip variable from any unsatisfied clause.

•Solves 2-SAT (2 variables per clause) in O(n^2) flips.
(Papadimitriou 1992) Why limited value?

•Does not work at all for hard k-SAT (k $\geq$ 3).

Slide CS472–11

Mixing Random Walk with Greedy Local Search

•With probability p, **walk**,
i.e., pick a variable in some
unsatisfied clause and flip it;
with probability $(1 - p)$ make a **greedy** flip,
i.e., one that makes greatest decrease in number of
unsatisfied clauses.

•Value for parameter p determined empirically,
by finding best setting for a problem class.

Slide CS472–12

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	*	*

Slide CS472–13

- 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*
  Simul. Ann. better than
  Basic GSAT better than
  Backtracking (Davis-Putnam).

Slide CS472–14

Annealing / mixed random walk introduced to deal with:

Local minima (maxima)
Plateaus

True local minima are rare in high dim. space.
Term often relaxed to include plateaus.

Slide CS472–15

1. **local minimum (maximum):** rate of change is positive (negative) in all directions even though the global minimum (maximum) hasn't been reached. (maze problem — may have to move AWAY from goal to find the (best) solution)
2. **plateaus:** all of the local steps look the same (in 8-puzzle, maybe none of the available moves changes the number of tiles out of place)

Slide CS472–16

Approaches

1. Simulated annealing (done)
2. Mixed-in random walk (done)
3. Random restarts
4. Tabu search
5. Genetic algorithms / programming

1. **random restarts:** simply restart at a new random state after a pre-defined number of local steps.
2. **tabu:** prevent 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 currently on the tabu list.

```
without tabu:
  flip var 1, var 2, var 4, var 2, var 10, var 11,
                            var 1, var 10, var 3 ...

with tabu (length 5) --- possible (why?) sequence:
  flip var 1, var 2, var 4, var 10, var 11,
                            var 1, var 3...
```

Tabu quite powerful; competitive with simul. ann.
random walk (depends on domain).

Slide CS472–19

Genetic algorithm

Approach mimics *evolution*.
See Section 20.8 R&N.
Often presented using rich new vocabulary:
fitness function, population, invividuals, "genes", crossover, mutations, etc.
Still, can be viewed quite directly in terms of standard **local search**.
Note: in some sense, natural evolution performs a local search to improve species: "survival of the fittest".
"local search in the gene pool"
What is the evaluation function?

Slide CS472–20

What are some of the special aspect of the evolutionary search process?

(Friedberg 1958; Holland 1975; Koza 1992)

Slide CS472–21

- High degree of parallelism
- New individuals ("next state / neighboring states"):
  - derived from "parents" ("crossover operation")
  - genetic mutations
- Selection next generation:
  - Survival of the fittest. (gives greediness)

Slide CS472–22

General Idea

Maintain a population of strings (states / individuals / candidate solutions)

Generate a sequence of generations:

- From the current generation, select pairs of strings (based on fitness) to generate new strings, using **crossover**.
- Introduce some noise through random **mutations**.

Average and maximum fitness (i.e. value to be optimized) increases over time.

Slide CS472–23

Example

Graph coloring.

	#1	#2	#3	#4	#5	fitness
1	G	B	G	R	B	
2	R	G	G	R	B	
3	R	R	R	G	B	
4	G	G	R	R	B	

generation 1

Slide CS472–24

Example Crossover

```
              ``parents''                   ``childern''
  #1   #2   #3   #4   #5        #1   #2   #3   #4   #5      fitness
1  G    B    G    R    B         G    B    R    G    B

3  R    R    R    G    B         R    R    G    R    B

example mutation:
   R    R    G    R    B         G    R    G    R    B
```

Slide CS472–25

Children + mutations = new generation.

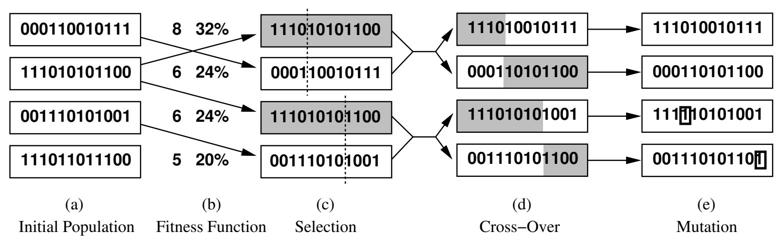
Note: in parents pairing, preference given to "fitter" parents.

Leads to gradual increase of average fitness.

Always??

Slide CS472–26

Example R&N page 621.



Slide CS472–27

Remarks

In practice, several 100 to 1000's of strings.

Value crossover difficult to determine (so far).

What if you have only mutations?

What makes this work for graph coloring?

Could this work for SAT testing (GSAT etc.)?

Slide CS472–28

Remarks, cont.

Locality crucial for crossover.
In general: highly sensitive to representation.
Genetic programming: underlying repr. are actual programs!
Given enough compute time, it's the best search algorithm in **certain** domains.
Often combined with standard greedy local search.

Perspective

Jury still out on Genetic Algorithms in general, but nature suggests it can be a highly powerful mechanism for evolving highly complex systems. (e.g., humans)
Formal properties far from understood.

Local Search — Summary

Surprisingly efficient search method.

Wide range of applications.

any type of optimization / search task

Formal properties elusive.

Intuitive explanation:

Search spaces (e.g., 10^{1000}) are often too

too large for systematic search anyway . . .

Area will most likely continue to thrive.

Often best evailable with lack of global info.

To some extent how evolution built us . . ..