Constraint Satisfaction Problems

Local Methods

Generate and Test

Given Problem Space (S, G),

Repeat
    s <- generate(S)
Until (test(solution) or tired(self))

Why is this method interesting?

What limits its usefulness?

can view much of problem solving
as reformulating a problem
until generate and test works
Constraint Satisfaction Problems

Weak Methods

Hill Climbing

beyond Problem Space (S, G), the method requires:

a metric $m$
the higher (lower),
the closer is the state to G
operations \{o\}
each operation maps state to state

Method: steepest ascent hill climbing

1. $s \leftarrow$ generate(S)
2. $m \leftarrow m(s)$
3. Repeat
   $s' \leftarrow s$
   $m' \leftarrow m$
   $\{s\} \leftarrow \{o\}(s')$ (apply set of operations)
   $s \leftarrow$ state that maximizes $m$ over $\{s\}$
   $m \leftarrow$ maximum value of $m$ over $\{s\}$
   Until $m < m'$
4. Report (check) $s'$ as solution in G
Hill Climbing
Example

Find 9 coins that have total value equal to $1.00.

Problem Space
State Space S
<p, n, d, q, h>
such that p+5n+10d+25q+50h=100
Goal G
p+n+d+q+h=9

Method
operations
making change
op0: h <-> 2q
op1: q <-> 2d, n
op2: 2q <-> 5d
op3: d <-> 2n
op4: n <-> 5p

Metric (minimize)
| p + n + d + q - 9 |
Hill Climbing (example)

\[
\begin{array}{ccc}
\langle 0,0,0,4,0 \rangle & 5 \\
\text{op1} & \text{op2} & \text{op0} \\
\langle 0,1,2,3,0 \rangle & 3 & \langle 0,0,5,2,0 \rangle & 2 & \langle 0,0,0,2,1 \rangle & 6 \\
\text{op1} \\
\langle 0,1,7,1,0 \rangle & 0 \\
\langle 0,0,10,0,0 \rangle & 1 \\
\text{op2} & \text{op3} \\
\langle 0,0,5,2,0 \rangle & 2 & \langle 0,2,9,0,0 \rangle & 2
\end{array}
\]
Hill Climbing

Issues

- metric
  - multipeaked
  - sharp peak
  - plateau

- operations
do not cover state space
  (can't get everywhere)

Approaches

- multiple searches from different initial states
- informed choice of initial state
- alter operation size (abstraction)
- simulated annealing
Constraint Satisfaction Problems

Simulated Annealing

chance of accepting a new state
   with lower evaluation

according to "heat" or "noise" of process

let noise be higher earlier in process
   decreases as process proceeds

parameters for

   initial heat
   decrease function (sigmoidal or exponential)
   acceptance threshold

Has proven useful... Why?
Conflict-based Search

conflict

given a complete state,

a constraint that is violated

method

1. start with random, complete state

2 while (conflicts exist)

   select a variable
       (at random, or maximum conflicts)
   select a new value
       (at random, or minimum conflicts)
3-SAT

local algorithms

GSAT
flip variable that minimizes number of unsatisfied clauses

WSAT
flip variable from unsatisfied clause that minimizes number unsatisfied

TABU
refuse to flip any variable that has been flipped within the last t flips

BSAT
flip variable minimizing number of true clauses that become false

all with probability $p$, flip any variable
all start $\text{maxtries}$ times,
search to $\text{maxflips}$ depth

accept lesser evaluations, limit search time
Optimization Problems

given

variable domains (state space)
set of constraints
measure to be optimized

find

state meeting constraints with
best value of measure

hill climbing and simulated annealing
are natural methods to use

Biologically-Inspired Methods

genetic algorithms

ant colony optimization

(particle) swarm optimization
Biologically-Inspired Methods

genetic algorithms

ant colony optimization

(particle) swarm optimization

rather than considering one possible solution at a time, consider a group of candidates

population-based methods

group of candidate solutions

a next group of candidate solutions is created as a function of current group

move from “generation” to “generation”
Evolution

processes

recombination

<parents>

mutation

<genotype>

selection

development

<phenotype>

evaluation
Genetic Algorithms

example problem

8-Queens

place 8 queens on chess board
so they do not attack each other

genotype

location of queen in each column
permutation as no two can be in same row
Genetic Algorithms

genetic operations

recombination

single-point crossover

select random location

head of one, tail of other

146 | 82375
14614782 / 63582375
635 | 14782

uniform crossover

randomly select value from two genotypes

14682375
13582785
63514782

application specific

14614782 => 14653782 (replace repeated)
Genetic Algorithms

genetic operations

mutation

with mutation probability (a parameter)
modify a given location
a “small” change to genotype

if binary genotype
flip bit

if integer genotype
pick another value at random

if permutation genotype
swap with another location

14653782 ==> 14853762
Genetic Algorithms

fitness function

evaluation of phenotypes

problem specific

every example

for 8 queens-

\[ 28 - \frac{\text{#of pairwise mutual attacks}}{28} \]

range --- 0.0 all on diagonal
1.0 no attacks
Genetic Algorithms

parent selection

based on evaluated of phenotypes

fitness proportionate selection

chance of selecting individual i to be a parent is equal to

\[
\frac{f(i)}{\sum_{x} f(x)}
\]

all x

rank selection

sort population given a range of values (low, high)
make a linear scale between lowest and highest individual
i individual as rank value low+i*(high-low)/(N-1)

evens out early selection, keeps selection active
Genetic Programming

genotype represents a program as possible solution to construct possible solution often represented as a tree

human competitive problem solving

\[
\left(2.2 - \left(\frac{X}{11}\right)\right) + \left(7 \times \cos(Y)\right)
\]
Optimization Problems

Genetic Algorithms

a form of hill climbing

instead of a single current solution,
maintain set as current population

add recombination/mutation operations
that combine genotypes of two individuals

parental selection based on value of metric
"fitness function"

parameters
mutation rate, reproduction rate

Method:
1. current-pop <- generate(S)
2. e-pop <- evaluate-fitness(current-pop)
3. until [done(e-pop)]
   parents <- select-parents(e-pop)
   current-pop <- apply-genetics(parents)
   e-pop <- evaluate-fitness(current-pop)
4. report solution
Ant Colony Optimization

analogy is from ant colony foraging..

initially all ants go off at random
searching for food
as ants return, they lay down a pheromone trail
whose strength is directly
related to success of search

as an ant searches, it’s direction is determined as
a function of pheromone strength..
more pheromone in a given direction,
more likely to follow

still random exploration...
but with exploitation of recent success
Ant Colony Optimization

ant behavior is instance of

stigmergy

ability to change the environment to alter behavior

termites building nests...

as the nest reaches a certain structure,

termites change behavior in response to the

structure they have built

swarm-based robotics---

complex tasks completed by reflex agents

using techniques based on stigmergy
ACO Algorithm

1. Initialize selection probabilities to equal values (depending on number of choices)

2. for some number of generations
   for each “ant”
   construct a solution, making choices based on current selection probabilities
   evaluate the resultant solution

3. Update best solution found

4. Update selection probabilities based on solutions, their choices and their evaluations
   the better the solution, more increase in probabilities
   reduce by a decay factor
   (details depend on problem)
ACO Algorithm

example

Traveling Salesman Problem
visit all cities traveling shortest distance

choices .. go(i,j), travel from i to j

start ants at arbitrary cities, choose next city
based on current selection probabilities

maintain selection weights

for all tours T, update selection weights due to T

\[ \frac{c}{\text{length}(T)} \]

subtract a decay constant from all weights
normalize weights to determine probabilities

performs as well as genetic algorithm approaches
ACO Applications

network routing...

adaptive routing

responsive to changing situations

research

multi-objective, dynamic problem contexts

exploration, exploitation, decay

parallel/distributed/networked implementations
Particle Swarm Optimization

derived from ideas explored with
  flocking behaviors

http://www.red3d.com/cwr/boids/index.html

Boids

Suppose are looking for something.. not only interacting with each other

need a heuristic measure.... distance to goal
  of goodness of position

want agents to move toward best in group

  each agent (particle) has a velocity

  update velocity
v = v + c1 * rand * (vector to its best)
  + c2 * rand * (vector to global best)

  update position
  add velocity v to current position