Methods for finding optimal configurations

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square

Announcements

• Homework assignment 3 is out
  – Due on Thursday next week !!!!
  – Programming and experiments
  – Simulated annealing + Genetic algorithm
  – Competition

Course web page:
http://www.cs.pitt.edu/~milos/courses/cs1571/
Search for the optimal configuration

**Optimal configuration search:**
- Configurations are described in terms of variables and their values
- Each configuration has a quality measure
- **Goal:** find the configuration with the best value

If the space of configurations we search among is
- **Discrete or finite**
  - then it is a combinatorial optimization problem
- **Continuous**
  - then it is a parametric optimization problem

---

**Example: Traveling salesman problem**

**Problem:**
- A graph with distances
- A tour – a path that visits every city once and returns to the start e.g. ABCDEF

- **Goal:** find the shortest tour
Example: N queens

- Originally a CSP problem
- But it is also possible to formulate the problem as an optimal configuration search problem:
  - **Constraints are mapped to the objective cost function that** counts the number of violated constraints

```
# of violations = 3
# of violations = 0
```

Iterative optimization methods

- Searching systematically for the best configuration with the DFS may not be the best solution
- Worst case running time:
  - Exponential in the number of variables
- Solutions to **large ‘optimal’ configuration** problems are often found more effectively in practice using **iterative optimization methods**

- **Examples of Methods:**
  - Hill climbing
  - Simulated Annealing
  - Genetic algorithms
Iterative optimization methods

**Basic Properties:**

- **Search** the space of “complete” configurations
- **Take advantage of local moves**
  - Operators make “local” changes to “complete” configurations
- **Keep track of just one state** (the current state)
  - no memory of past states
  - !!! No search tree is necessary !!!

Example: N-queens

- “Local” operators for generating the next state:
  - Select a variable (a queen)
  - Reallocate its position
Example: Traveling salesman problem

“Local” operator for generating the next state:
• divide the existing tour into two parts,
• reconnect the two parts in the opposite order

Example:

ABCDEF

A

B

C

D

E

F

Example: Traveling salesman problem

“Local” operator for generating the next state:
• divide the existing tour into two parts,
• reconnect the two parts in the opposite order

Example:

ABCDEF

A

B

C

D

E

F

ABCDFE
Example: Traveling salesman problem

“Local” operator:
– generates the next configuration (state)

Searching the configuration space

Search algorithms
• keep only one configuration (the current configuration)

Problem:
• How to decide about which operator to apply?
Example: Traveling salesman problem

“Local” operator:
- generates the next configuration (state)
  by rearranging the existing tour

<table>
<thead>
<tr>
<th>ABCDEF</th>
<th>ABCD</th>
<th>EF</th>
<th>ABCDFE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABCD</td>
<td>EF</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Search algorithms

Strategies to choose the configuration (state) to be visited next:
- Hill climbing
- Simulated annealing

- Extensions to multiple current states:
  - Genetic algorithms
  - Beam search

- Note: Maximization is inverse of the minimization
  \[
  \min f(X) \leftrightarrow \max \left[ - f(X) \right]
  \]
Hill climbing

- What configurations are considered next?
- What move the hill climbing makes?

Hill climbing

- Look at the local neighborhood and choose the one with the best value

- What can go wrong?
Hill climbing

• Hill climbing can get trapped in the local optimum

value

states

No more local improvement

Better

Hill climbing

• Hill climbing can get clueless on plateaus

value

states

plateau
Hill climbing

- How to remedy the problem of local optima?

No more local improvement

Hill climbing

- Multiple restarts of the hill climbing algorithms from different initial states

A new starting state may lead to the globally optimal solution
Hill climbing and n-queens

- The quality of a configuration is given by the number of constraints violated
- **Then: Hill climbing** reduces the number of constraints
- **Min-conflict strategy (heuristic):**
  - Choose randomly a variable with conflicts
  - Choose its value such that it violates the fewest constraints

Success !! But not always!!! The local optima problem!!!

Simulated annealing

- An alternative solution to the local optima problem
- Permits “bad” moves to states with a lower value hence lets us escape states that lead to a local optima
- **Gradually decreases** the frequency of such moves and their size (parameter controlling it – **temperature**)

![Diagram of value vs states showing simulated annealing process](image)
Simulated annealing algorithm

- Based on a random walk in the configuration space

**Basic iteration step:**
- Choose uniformly at random one of the local neighbors of the current state as a candidate state
- if the candidate state is better than the current state then
  accept the candidate and make it the current state;
- else
  calculate the probability $p(\text{ACCEPT})$ of accepting it using $p(\text{ACCEPT})$ choose randomly whether to accept or reject the candidate

---

Simulated annealing algorithm

**The probability $p(\text{ACCEPT})$ of the candidate state:**
- The probability of accepting a state with a better objective function value is 1
- The probability of accepting a candidate with a lower objective function value is $< 1$ and equal:
- Let $E$ denotes the objective function value (also called energy).

\[
p(\text{Accept } \text{NEXT}) = e^{\Delta E / T} \quad \text{where} \quad \Delta E = E_{\text{NEXT}} - E_{\text{CURRENT}} \quad T > 0
\]
- The probability is:
  - Proportional to the energy difference
Simulated annealing algorithm

Possible moves

Current configuration

Energy $E = 167$

Energy $E = 145$

Energy $E = 180$

Energy $E = 191$

$\Delta E = E_{\text{NEXT}} - E_{\text{CURRENT}}$

$= 145 - 167 = -22$

$p(\text{Accept}) = e^{\Delta E / T} = e^{-22 / T}$

Sometimes accept!
Simulated annealing algorithm

The probability of accepting a state with a lower value is

$$p(\text{Accept}) = e^{\Delta E / T}$$

where $\Delta E = E_{\text{NEXT}} - E_{\text{CURRENT}}$

The probability p(accept) is:

- **Modulated through a temperature parameter $T$:**
  - for $T \to \infty$ ?
  - for $T \to 0$ ?

- **Cooling schedule:**
  - Schedule of changes of a parameter $T$ over iteration steps

$\Delta E = E_{\text{NEXT}} - E_{\text{CURRENT}} = 180 - 167 > 0$

$p(\text{Accept}) = 1$

**Always accept!**
Simulated annealing algorithm

The probability of accepting a state with a lower value is

\[ p(\text{Accept}) = e^{\frac{\Delta E}{T}} \quad \text{where} \quad \Delta E = E_{\text{next}} - E_{\text{current}} \]

The probability is:

- **Modulated through a temperature parameter** \( T \):
  - for \( T \to \infty \) the probability of any move approaches 1
  - for \( T \to 0 \) the probability that a state with smaller value is selected goes down and approaches 0

- **Cooling schedule:**
  - Schedule of changes of a parameter \( T \) over iteration steps

---

Simulated annealing

function \text{SIMULATED-ANNEALING}(\text{problem, schedule}) \text{ returns a solution state}

inputs: \text{problem, a problem}

\text{schedule, a mapping from time to "temperature"}

static: \text{current, a node}

next, a code

\( T \), a "temperature" controlling the probability of downward steps

current := \text{MAKE-NODE(INITIAL-STATE[problem])}

for \( t \leftarrow 1 \) to \( \infty \) do

\( T \leftarrow \text{schedule}[t] \)

if \( T = 0 \) then return \text{current}

next := a randomly selected successor of \text{current}

\( \Delta E \leftarrow \text{Value}[\text{next}] - \text{Value}[\text{current}] \)

if \( \Delta E > 0 \) then \text{current} := next

else \text{current} := \text{next} only with probability \( e^{\Delta E / T} \)
Simulated annealing algorithm

- **Simulated annealing algorithm**
  - developed originally for modeling physical processes
    (Metropolis et al, 53)

- **Properties:**
  - If temperature $T$ is decreased slowly enough the best configuration (state) is always reached

- **Applications:**
  - VLSI design
  - airline scheduling

Simulated evolution and genetic algorithms

- Limitations of **simulated annealing**:
  - Pursues one state configuration at the time;
  - Changes to configurations are typically local

**Can we do better?**

- Assume we have two configurations with good values that are quite different
- We expect that the combination of the two individual configurations may lead to a configuration with higher value (Not guaranteed !!!)

This is the idea behind **genetic algorithms** in which we grow a population of candidate solutions generated from combination of previous configuration candidates
Genetic algorithms

**Algorithm idea:**
- **Create a population of random configurations**
- **Create a new population through:**
  - Biased selection of pairs of configurations from the previous population
  - Crossover (combination) of selected pairs
  - Mutation of resulting individuals
- **Evolve the population over multiple generation cycles**

- **Selection of configurations to be combined:**
  - **Fitness function** = value of the objective function measures the quality of an individual (a state) in the population

---

**Reproduction process in GA**

- Assume that a state configuration is defined by a set variables with two values, represented as 0 or 1

---

(a) Initial Population  (b) Fitness Function  (c) Selection  (d) Cross-Over  (e) Mutation