Simulated Annealing

Contents

1 Recall

1.1 Heuristic search

Definition.

- The universe $\mathcal{X}$ is a finite set of elements which are possible (not necessarily feasible) solutions
- $X \in \mathcal{X}$ is feasible if it satisfies the constraints of the problem
- $P(X)$ is the profit of X
- A neighbourhood function is a function

$$N : \mathcal{X} \to 2^{\mathcal{X}}$$

  where $2^{\mathcal{X}}$ is the set of all subsets of $\mathcal{X}$.

- Given N, a neighbourhood search is an algorithm, possible randomized, which takes a feasible solution $X \in \mathcal{X}$ and returns either a feasible solution $Y \in N(X) \setminus \{X\}$ or `fail`

The idea is that if we are at a feasible solution, then we consider its neighbourhood, and following the neighbourhood search algorithm we update X to Y (or we fail).

1.2 Hill climbing

The simplest example of heuristic search is hill climbing.

Algorithm: GenericHillClimbing

```
Select a feasible X in the universe
searching = true
while searching
    try to get a feasible solution Y in N(X) with P(Y) > P(X) (randomly or exhaustively)
    if Y = fail
        searching = false
    else
        X = Y
return X
```

2 Simulated Annealing

How can we get out of local maxima? One approach is to have some chance of going down rather than up. If this chance is large things will be quite jittery, we won't make much progress but nor will be get stuck. If this chance is small things will be close to hill climbing.

Simulated annealing is when we start with a large chance of going down and then decrease the probability as the algorithm goes on. The name is in analogy with of annealing of metal where a high temperature lets the atoms move around and fill in holes in the crystal structure, and controlled cooling maintains this as the atoms calm down.

For simulated annealing suppose we are at X, let $Y \in N(X)$ be a feasible solution returned by the neighbourhood search. If $P(Y) > P(X)$ then move to Y, otherwise with probability $e^{(P(Y)-P(X))/T}$ move to Y, where T decreases by $T = \alpha T$ at each iteration $0 < \alpha < 1$. The generic shape of the algorithm is

Algorithm: GenericSimulatedAnnealing

```
Select a feasible X in the universe
c=0
T=T0
bestX=X
while c <= cmax
    try to get a feasible solution Y in N(X)
    if not(Y = fail)
        if P(Y) > P(X)
            X = Y
            if P(X) > P(bestX)
                bestX = X
        else
            r = rand(0,1)
            if r < exp((P(Y)-P(X))/T)
                X = Y
    c = c+1
    T = alpha*T
return bestX
```

2.1 Knapsack problem by simulated annealing

To use simulated annealing for the knapsack problem make the following choices

- $N(X) = \{X \in \{0,1\}^n : d(X,Y) = 1\}$ where d is the Hamming distance
- Given X, generate a random $Y \in N(X)$ by choosing a random index $0 \le j \le n-1$ and swapping that bit. Then
$$w(Y) = \begin{cases} w(X) + w_j & \text{if } x_j = 0 \\ w(X) - w_j & \text{if } x_j = 1 \end{cases}$$
and
$$P(Y) - P(X) = \begin{cases} p_j & \text{if } x_j = 0 \\ -p_j & \text{if } x_j = 1 \end{cases}$$
- begin with the feasible solution $(0, 0, \ldots, 0)$.

This gives the following algorithm

Algorithm: SimulatedAnnealingKnapsack

```
c = 0
T = T0
X = [0,...,0]
CurW = 0
bestX = X
while c <= cmax
    j random integer 0<= j <= n-1
    Y = X
    y(j) = 1-x(j)
    if not(y(j) = 1 and CurW + w(j) > M)    (the condition inside the not is the fail)
        if y(j) = 1    (that is if the profit increased)
            X = Y
            CurW = CurW + w(j)
            if P(X) > P(bestX)
                bestX = X
        else
            r = rand(0,1)
            if r < exp(-p(j)/T)
                X = Y
                CurW = CurW - w(j)
    c = c+1
    T = alpha*T
return bestX
```

It remains to select T_0, α and c_{max} depending on the problem instance. T_0 should be large enough to have a high probability of accepting a downwards move initially, so looking at the profits in the problem we want e^{-p_j/T_0} to be more than, say, $3/4$ for all j, while remaining less than 1 for all j, if possible.

Kreher and Stinson took the following weights and profits:

weight	profit
70	135
73	139
77	149
80	150
82	156
87	163
90	173
94	184
98	192
106	201
110	210
113	214
115	221
118	229
120	240

They took capacity 750 and $T_0 = 1000$ and tried different values for α and c_{max}. They ran 10 runs for each set of parameters. Note that the optimal profit for this instance is 1458. The result of their runs is the following table (from p177)

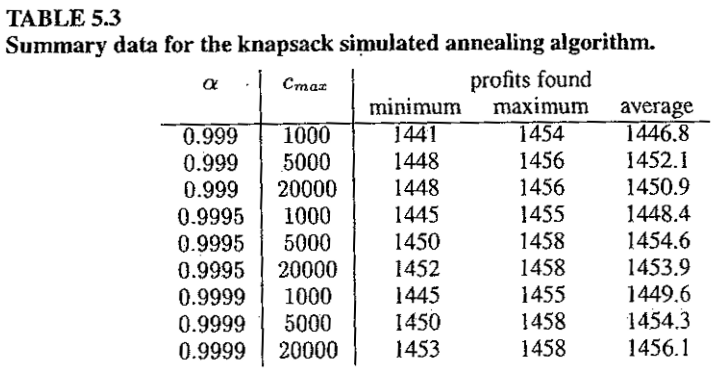

TABLE 5.3
Summary data for the knapsack simulated annealing algorithm.

α	c_{max}	profits found		
		minimum	maximum	average
0.999	1000	1441	1454	1446.8
0.999	5000	1448	1456	1452.1
0.999	20000	1448	1456	1450.9
0.9995	1000	1445	1455	1448.4
0.9995	5000	1450	1458	1454.6
0.9995	20000	1452	1458	1453.9
0.9999	1000	1445	1455	1449.6
0.9999	5000	1450	1458	1454.3
0.9999	20000	1453	1458	1456.1

Thanks for a nice semester everyone, and good luck with the future.