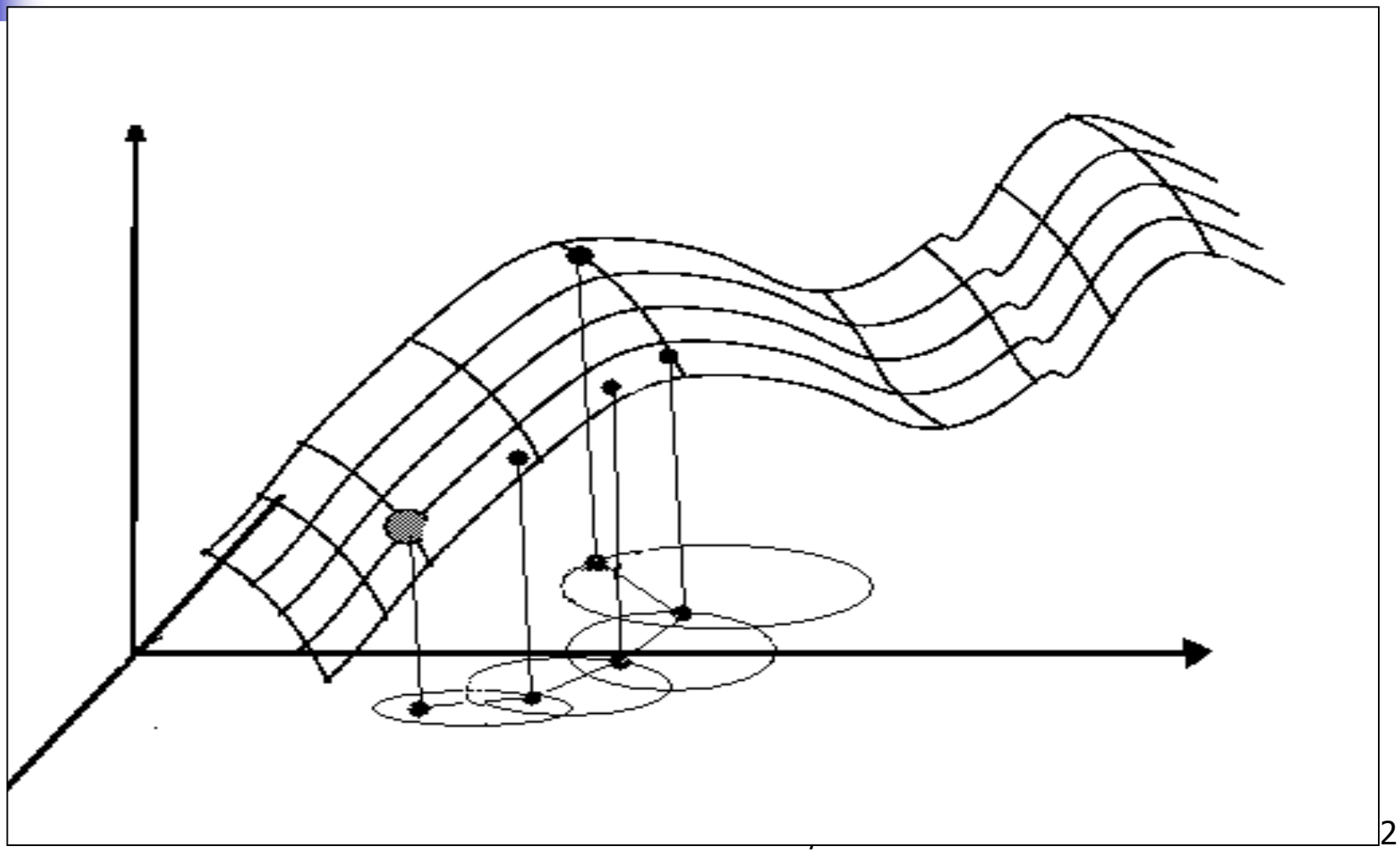
Artificial Intelligence Methods (G52AIM)

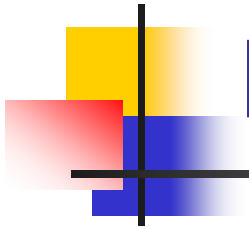
Dr Rong Qu

rxq@cs.nott.ac.uk

Simulated Annealing

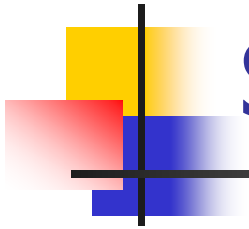
Hill Climbing – local optima





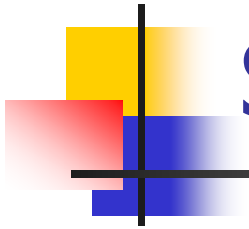
Hill Climbing – local optima

- Problem: Gets stuck at local minima!
  - Moves are **always to better states**
- Possible solutions
  - Try several runs, starting at different initial positions
  - Increase the size of the neighborhood (e.g. in TSP try 3-opt rather than 2-opt)



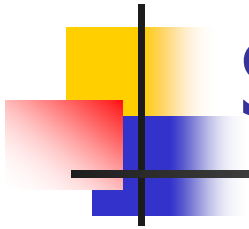
Simulated Annealing vs. HC

- Hill-climbing
  - moves are always to better states
- Simulated annealing
  - To escape a local optimum we must allow worsening moves
  - In a controlled way, allow downwards (“wrong-way”, worsening) steps



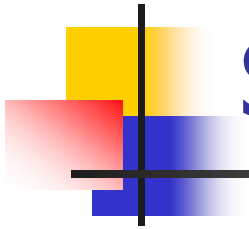
Simulated Annealing vs. HC

- Hill-climbing
  - might consider many possible moves
  - evaluate many solutions, can be too expensive
- Simulated annealing
  - randomly select one state in the neighbourhood
  - decide whether to accept it or not
  - **better moves are always accepted**
  - **worsening moves are sometimes selected**



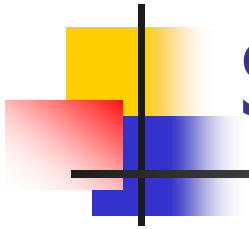
Simulated Annealing

- Motivated by the physical annealing process
- Material is heated and slowly cooled into a uniform structure
- Simulated annealing mimics this process
- The first SA algorithm was developed in 1953 (Metropolis)



Simulated Annealing

- **Kirkpatrick (1982) applied SA to optimisation problems**
  - Kirkpatrick, S , Gelatt, C.D., Vecchi, M.P. 1983. *Optimization by Simulated Annealing*. Science, vol 220, No. 4598, pp 671-680



Simulated Annealing - acceptance

- The law of thermodynamics states that at temperature, *t*, the probability of an increase in energy of magnitude, δE, is given by

$$P(\delta E) = \exp(-\delta E / kt)$$

Where *k* is a constant known as Boltzmann's constant

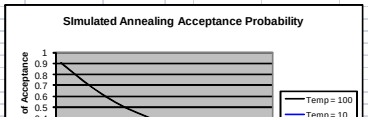
Simulated Annealing - acceptance

$$P = \exp(-c/t) > r$$

- where
  - c is change in the evaluation function
  - t is the current temperature
  - r is a random number between 0 and 1
- Example

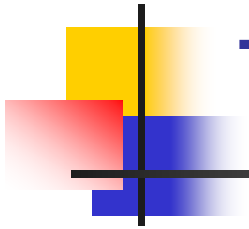
Change in Evaluation Function	Temperature of System	exp(-C/T)
10	100	0.904837418
20	100	0.818730753
30	100	0.740818221
40	100	0.670320046
50	100	0.60653066
60	100	0.548811636
70	100	0.496585304
80	100	0.449328964
90	100	0.40656966
100	100	0.367879441
110	100	0.332871084
120	100	0.301194212
130	100	0.272531793
140	100	0.246596964
150	100	0.22313016
160	100	0.201896518
170	100	0.182683524
180	100	0.165298888
190	100	0.149568619
200	100	0.135335283

Change in Evaluation Function	Temperature of System	exp(-C/T)
10	10	0.367879441
20	10	0.135335283
30	10	0.049787068
40	10	0.018315639
50	10	0.006737947
60	10	0.002478752
70	10	0.000911882
80	10	0.000335463
90	10	0.00012341
100	10	4.53999E-05
110	10	1.67017E-05
120	10	6.14421E-06
130	10	2.26033E-06
140	10	8.31529E-07
150	10	3.05902E-07
160	10	1.12535E-07
170	10	4.13994E-08
180	10	1.523E-08
190	10	5.6028E-09
200	10	2.06115E-09



To accept or not to accept - SA?

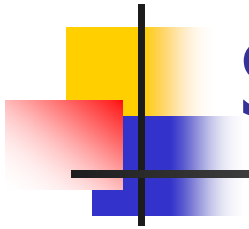
Change	Temp	exp(-C/T)		Change	Temp	exp(-C/T)
0.2	0.95			0.2	0.1	
0.4	0.95			0.4	0.1	
0.6	0.95			0.6	0.1	
0.8	0.95			0.8	0.1	



To accept or not to accept - SA?

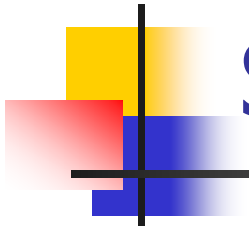
Change	Temp	exp(-C/T)		Change	Temp	exp(-C/T)
0.2	0.95	0.810		0.2	0.1	0.13583
0.4	0.95	0.656		0.4	0.1	0.018339
0.6	0.95	0.532		0.6	0.1	0.0024852
0.8	0.95	0.431		0.8	0.1	0.000335

- Need to use a scientific calculator to calculate exp()
- Note that the temperatures here are just examples and not recommendations for values to use in real implementations.



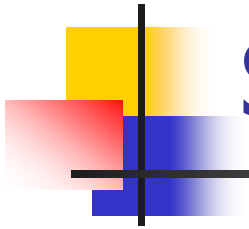
Simulated Annealing - acceptance

- The probability of accepting a worse state is a function of both the temperature of the system and the change in the cost function
- As the temperature decreases, the probability of accepting worse moves decreases
- If t = 0, no worse moves are accepted (i.e. hill climbing)



SA – implementation

- The most common way of implementing an SA algorithm is to implement hill climbing with an accept function and modify it for SA
- The example shown here is taken from Russell/Norvig (Artificial Intelligence : A Modern Approach)

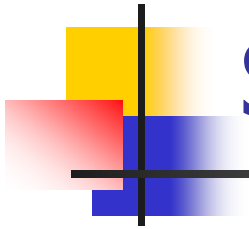


SA – algorithm

Function SIMULATED-ANNEALING(*Problem*, *Schedule*)
returns a solution state

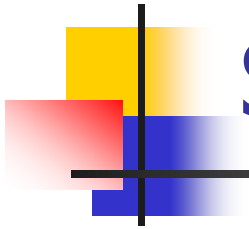
Inputs
Problem: a problem
Schedule: a mapping from time to temperature

Local Variables
Current: a node
Next: a node
T: a "temperature" controlling the probability of downward steps



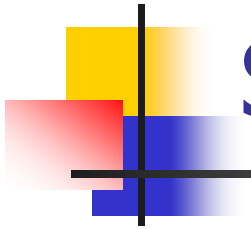
SA – algorithm

Current = MAKE-NODE(INITIAL-STATE[*Problem*])
For t = 1 **to** ∞ **do**
    T = *Schedule*[t]
    If T = 0 **then return** *Current*
    Next = a randomly selected successor of *Current*
    ΛE = VALUE[*Next*] – VALUE[*Current*]
    if ΛE > 0 **then** *Current* = *Next*
    else *Current* = *Next* only with probability exp(-ΛE/T)



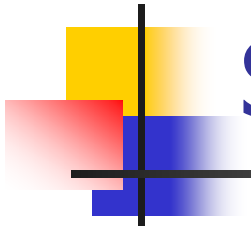
SA – algorithm

- The cooling schedule is *hidden* in this algorithm - but it is important (more later)
- The algorithm assumes that annealing will continue until temperature is zero - this is not necessarily the case



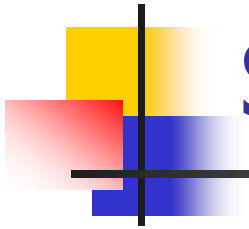
SA – cooling schedule

- Starting Temperature
- Final Temperature
- Temperature Decrement
- Iterations at each temperature



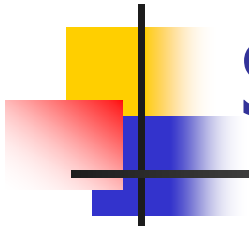
SA – cooling schedule

- Starting Temperature
  - Must be *hot* enough to allow moves to *almost* neighbourhood state (else we are in danger of implementing hill climbing)
  - Must *not* be so hot that we conduct a random search for a period of time
  - Problem is finding a suitable starting temperature



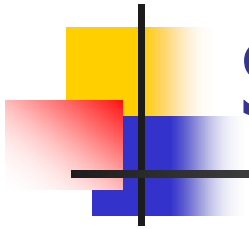
SA – cooling schedule

- Starting Temperature
  - If we know the maximum change in the cost function we can use this to estimate
  - Start high, reduce quickly until about 60% of worse moves are accepted. Use this as the starting temperature
  - Heat rapidly until a certain percentage are accepted the start cooling



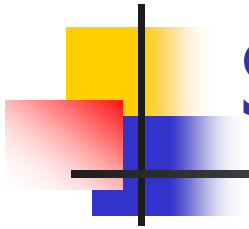
SA – cooling schedule

- Final Temperature
  - It is usual to let the temperature decrease until it reaches zero
    However, this can make the algorithm run for a lot longer, especially when a geometric cooling schedule is being used
  - In practise, it is not necessary to let the temperature reach zero because the chances of accepting a worse move are almost the same as the temperature being equal to zero



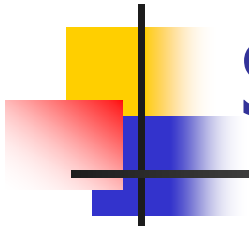
SA – cooling schedule

- Final Temperature
  - Therefore, the stopping criteria can either be a suitably low temperature or when the system is "frozen" at the current temperature (i.e. no better or worse moves are being accepted)
  - Example: online demo



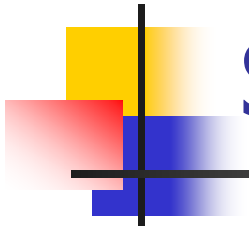
SA – cooling schedule

- Temperature Decrement
  - Theory states that we should allow enough iterations at each temperature so that the system stabilises at that temperature
  - Unfortunately, theory also states that the number of iterations at each temperature to achieve this might be exponential to the problem size



SA – cooling schedule

- Temperature Decrement
  - We need to compromise
  - We can either do this by doing a large number of iterations at a few temperatures, a small number of iterations at many temperatures or a balance between the two



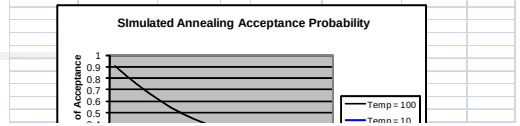
SA – cooling schedule

- Temperature Decrement
  - Linear: *temp = temp - x*
  - Geometric: *temp = temp* * α
    - Experience has shown that α should be between 0.8 and 0.99
    - Of course, the higher the value of α, the longer it will take to decrement the temperature to the stopping criterion

SA – cooling schedule

Change in Evaluation Function	Temperature of System	exp(-C/T)
10	100	0.904837418
20	100	0.818730753
30	100	0.740818221
40	100	0.670320046
50	100	0.60653066
60	100	0.548811636
70	100	0.496585304
80	100	0.449328964
90	100	0.40656966
100	100	0.367879441
110	100	0.332871084
120	100	0.301194212
130	100	0.272531793
140	100	0.246596964
150	100	0.22313016
160	100	0.201896518
170	100	0.182683524
180	100	0.165298888
190	100	0.149568619
200	100	0.135335283

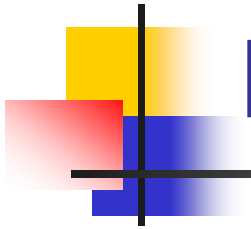
Change in Evaluation Function	Temperature of System	exp(-C/T)
10	10	0.367879441
20	10	0.135335283
30	10	0.049787068
40	10	0.018315639
50	10	0.006737947
60	10	0.002478752
70	10	0.000911882
80	10	0.000335463
90	10	0.00012341
100	10	4.53999E-05
110	10	1.67017E-05
120	10	6.14421E-06
130	10	2.26033E-06
140	10	8.31529E-07
150	10	3.05902E-07
160	10	1.12535E-07
170	10	4.13994E-08
180	10	1.523E-08
190	10	5.6028E-09
200	10	2.06115E-09



- Iterations at each temperature
  - A constant number of iterations at each temperature
  - Another method, first suggested by (Lundy, 1986) is to only do one iteration at each temperature, but to decrease the temperature *very* slowly. The formula used is

$$t = t/(1 + \beta t)$$

where β is a suitably small value



Learning Objectives

- SA basics
  - Cooling schedule (4 elements)
  - Acceptance criteria
- HC basics
  - Problem of local optima
- Be able to implement HC and SA in your coursework