Setting Parameters for Simulated Annealing

- All heuristic algorithms (and many nonlinear programming algorithms) are affected by "algorithm parameters"
- For Simulated Annealing the algorithm parameters are
- T_o, M, α, β, maxtime
- So how do we select these parameters to make the algorithm efficient?

Handout for videotaped lecture labeled 9-2-11 (taped after regular lecture on 9-2-11)

Setting Parameters in Simulated Annealing

- As we saw in the first simulated annealing problem, the results can depend a great deal on the values of the parameter T ("temperature"), which depends upon T_o and upon α.

  can be difficult to choose

- How should we pick T_o and α?
- We can use some simple procedures to pick estimate a reasonable value (not necessarily optimal) of T_o .

  ⇒ you can shape the probability curve

  – user decides on the probability of P1 of accepting an uphill move initially and P2 the probability of accepting an uphill move after G iterations.

Rough Estimates for Good Algorithm Parameter Values

- Note that the best algorithm parameters cannot be determined exactly and that the results are probably not sensitive to small changes in algorithm parameter values.

- Recall in previous SA example there was a difference between a T_o of 100 and a T_o of 500 in the two examples. This is a difference of 500% in the algorithm parameter value-not a small change.

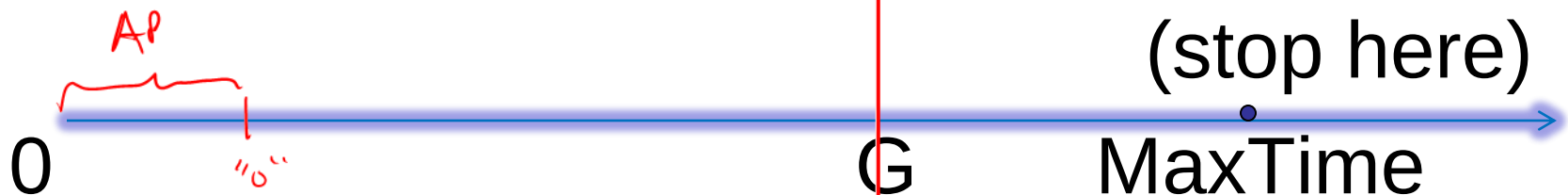
- Hence you are trying to get a rough estimate of what would be good algorithm parameter values .

BOARD : change in search over time

period to determine good parameter values

"global" search period

"local" search period.

- 

allow uphill moves(not greedy)

⇒ P large enough to reasonably allow uphill moves

(greedy)

⇒ P is small. Unlikely to make uphill moves

Numbers of objective function evaluations

Simple Approach to Parameterizing SA

- Assume M=1
- Assume
  - you want to do Maxtime iterations and you want Maxtime-G of these to be greedy. (Here greedy means having a probability of less than P2 for uphill move.)
  - The estimated range of values of Cost(S) = R
  - R= Max Cost (S)-Min Cost (S)
  - Initially you want to accept **P1** fraction of the moves.
  - By the time you have done G iterations you want to have an essentially greedy search and accept **P2** or fewer fraction of the moves.
  - P2<<P1 (For example, P1=.95 and P2 = .01)

Selecting Algorithm Parameters

- So how do we select these parameters to make the algorithm efficient?
- Recall that the principle is that initially you want to accept a relatively high percentage of uphill moves (for a minimization problem), but the more iterations you have done, the less you want to accept uphill moves.
- Acceptance of uphill moves is determined by To, M, and α. Assume M = 1, then $T_i = \alpha^i T_o$

  starting with T= To.

Let P1 be your goal for the average probability of accepting an UPHILL move initially and P2 be your goal for the probability of accepting an Uphill move after G iterations.

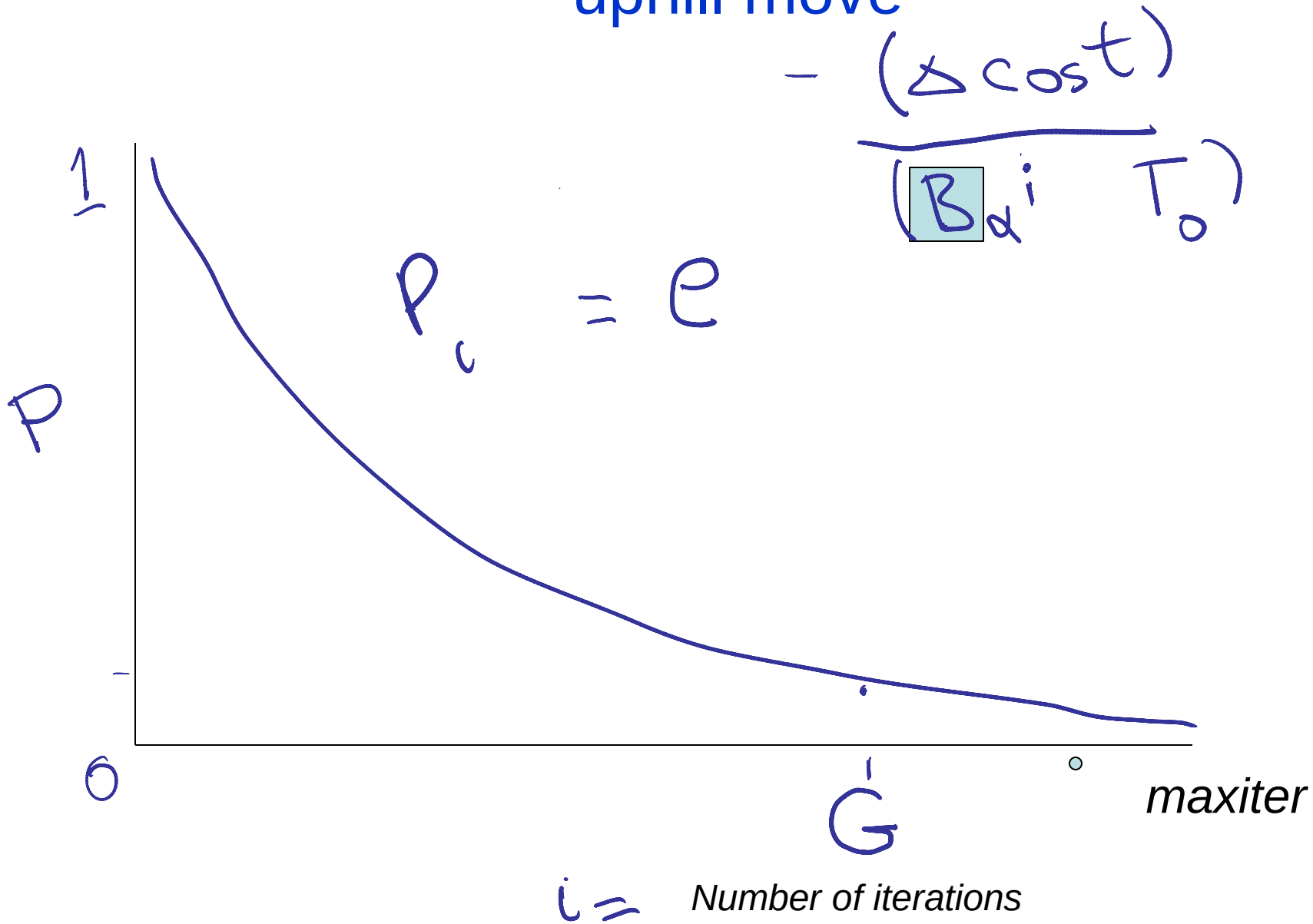
Simulated Annealing Morphs into What?

- Intially Simulated Annealing has a significant probability of accepting an uphill move.
- After many interations, Simulated Annealing becomes similar to what algorithm (Random Walk, Random Sampling, or Greedy Search)? Why?

Given $T_{o,,}$ Select α

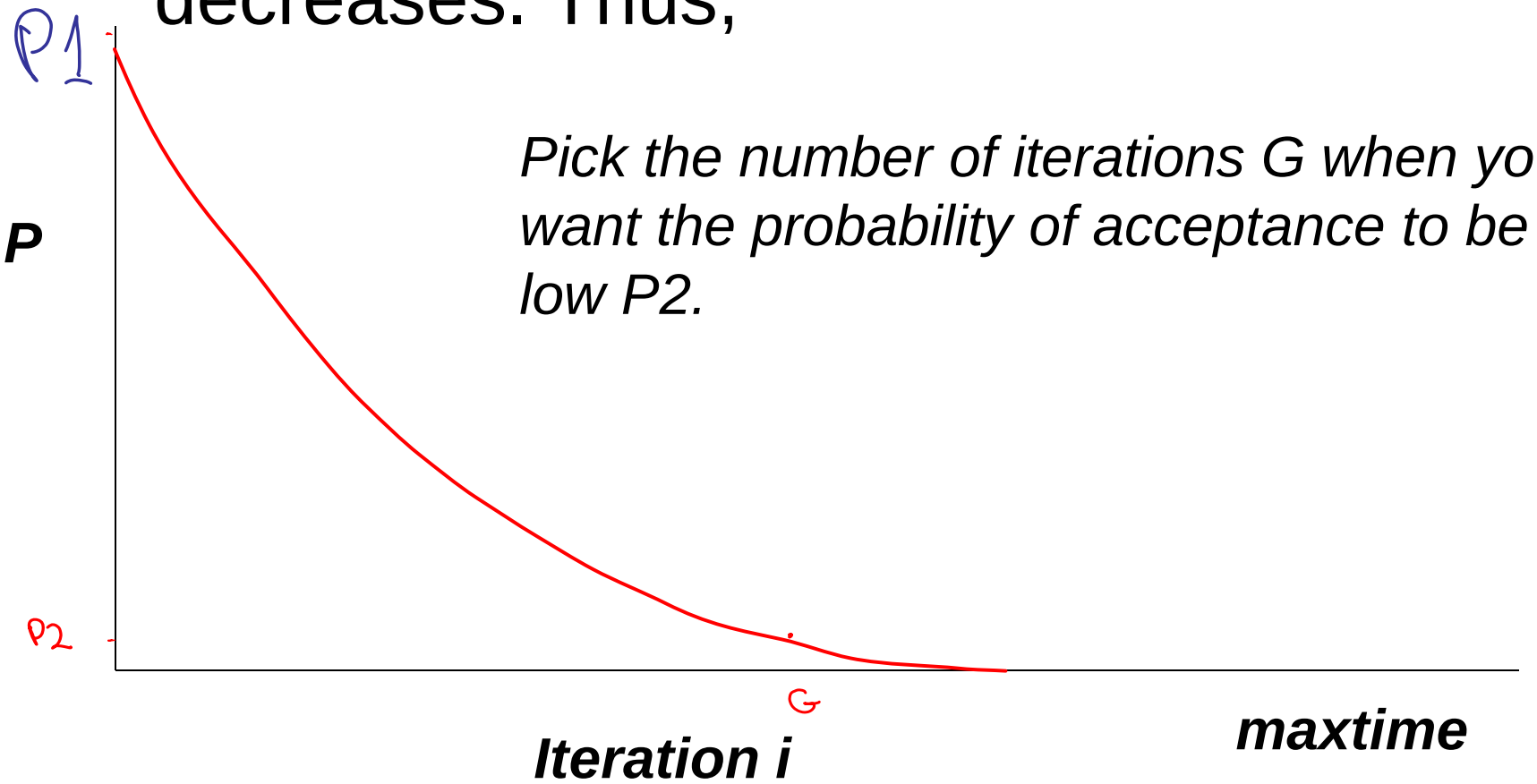
- Once α^i gets small enough after i iterations, then SA operates like __greedy__ search. WHY? P is too small unlikely to go uphill
- You have maxtime as the maximum number of iterations.
- Decide what fraction of these you want to have spent in greedy search.
- Let G= maxtime- number of iterations for greedy search and let P2>0, be a small number that allows very few uphill movements. (P2 selected by user)

BOARD P is probability of acceptance of uphill move



Effect of iteration number i on Probability

- After i iterations, $T = \alpha^i \, To$.(Decreasing)
- We know that P decreases as T decreases. Thus,



Pick the number of iterations G when you want the probability of acceptance to be a low P2.

Simple Method for Setting T_o

- Estimate the average of ΔCost. (called avgΔCost) for uphill moves in the initial iteration.

  We will discuss later ways to estimate this.
- Decide that you want initially to allow approximately Po percent of the uphill moves to Snew moves to be accepted. (User decision)
- Then given Po and avgΔCost, can you give an equation that includes T_o?

$$P1 = e^{-(\text{avg}\Delta\text{cost})/T_o}$$

is approximately what you want, and you can solve for T_o. So T_o =(-avgCost)/(ln P1)

Estimate for α

- Recall that our estimates for To and α are rough estimates of what would be good values, i.e. within an order of magnitude.
- To estimate a good value for α, we need to consider how fast do we want the probability of accept an uphill move to decline.
- We define this by the user deciding what probability (P2) do you want after G iterations.
- Recall Simulated annealing becomes like Greedy Search if the probability of uphill moves gets small and this probability is determined by α^{g} To after g iterations.

Selecting a value for α

- The probability P in kth iteration of accepting uphill move is

$P_k = \exp(-[COST(NewS)-COST(CurS)]/T_k$

We also have

$T_k = T_{k-1}\,\alpha = (T_{k-2}\,\alpha)\,\alpha = \ldots = T_o\,\alpha^k$

(The lower numbers are subscripts for iteration number. However the upper k on α is an exponential power). So the term that appears in the probability equation in the Gth iteration is

$P2 = \exp(-[COST(NewS)-COST(CurS)]/[\alpha^G\,T_o])$

P2 approximately $=\exp(-[\text{avg}\ \Delta COST)/[\alpha^G\,T_o])$

If we have already computed To, and we are given P2, G and avgΔCOST .

Hence we can compute α.

Selecting α

After G iterations and average ΔCost,
$T=\alpha^G T_o$,

so we compute the appropriate value of α using the equations on the following slide. Recall P2 is the probability of accepting an uphil move after G interation. If we assume average Δcost (="avgCost" for short) then

$$P2 = \exp(-avgCost/T)$$

Computing α , Given To , P2 and G

$$\boxed{P2 = \exp(-avgCost / T)}$$

$$P2 = \exp(-avgCost / \alpha^G T_0)$$

$$\ln P2 = \ln \exp(-avgCost / \alpha^G T_0)$$

$$\ln P2 = -avgCost / \alpha^G T_0$$

$$\alpha^G = -avgCost / T_0 \ln P2$$

$$\alpha = (-avgCost / (T_0 \ln P2))^{1/G}$$

Previous handwritten slide in videotaped lecture erroneously had "log" instead of "ln"