Importance of your course handouts and your own handwritten notes

- In exams you will be allowed to bring in your course handouts and your own handwritten notes (on the handouts on or other pages) so you will want to have them in order (perhaps with tabs for important topics).
- You can bring in copies of the homeworks you have completed (so be sure to get copies for each member of the homework team).
- In general you cannot bring in xeroxed copies of printed text or videotaped lectures

Handout 9-7-11

Importance of your course handouts and your own notes (continued)

- Hence the course handouts (which are also posted on blackboard) will be an important resource for you during exams.
- I strongly advise you to take notes and write questions on your handouts during lecture (and/or while you watch the video tape). This will help you study and perform well on a test.

Look at VideoNote if you did not see two lectures on 8-26 and on 9-2

- You need all the earlier lectures to do the homeworks.
- The two lectures taped that were not held at a regular class time were called 8-29 and 9-2 in Video Note so you should view these and take notes on your handouts.

Additional explanation on VideoNote

- 9-5-11 was labor day so no lecture on 9-5.
- There were two lectures taped on 8-26, which were called 8-26 and 8-29 on VideoNote
- There were two lectures taped on 9-2, which were called 8-31 and 9-2 in Videonote.

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

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

we have to guess this value

Computing α , Given To , P2 and G

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

substitute in for T

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

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

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

solve for α

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

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

# of iterations user-defined

guessed

calculated

user-specified desired probability

Previous handwritten slide in 9-2 videotaped lecture erroneously had "log" instead of "ln"

Recall that there Is an M Parameter

- Sometimes you don't want to reduce T in every iterations so you let it stay constant for M iterations and then reduce T by a factor of α. ⇒ effectively turn T into a piecewise stair-step
- So T=To for k=1,…M-1

  T= α To for k=M,….,2M-1

  etc,

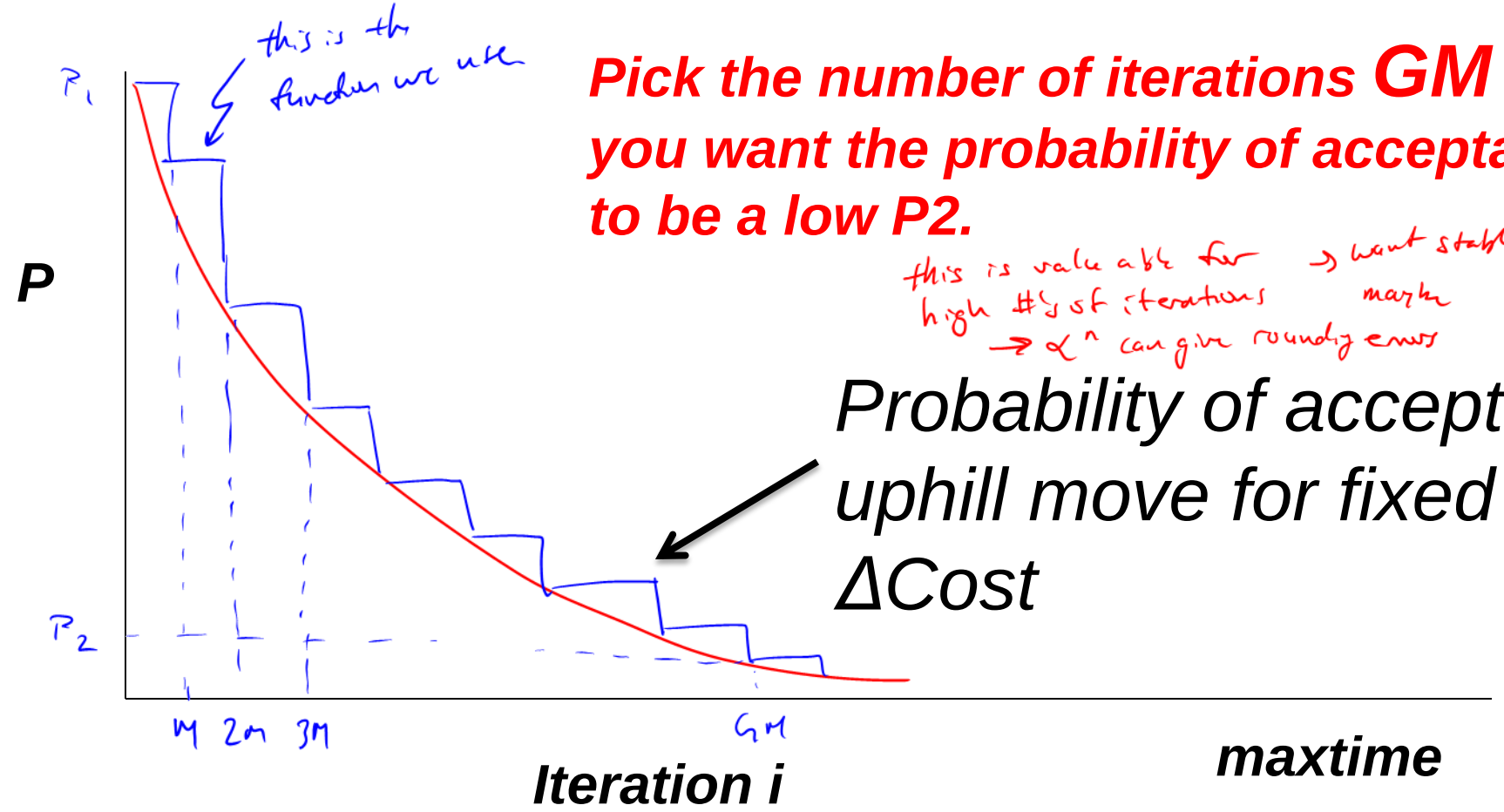
} turns the curve into a piecewise evaluation.

After iM iterations, $T= \alpha^i$ To.(Decreasing)

After GM iterations, $T=\alpha^G T_o$,

What is the effect if M>1

- After iM iterations, $T = \alpha^i To$.(Decreasing)
- Thus, for a fixed ΔCost we have:

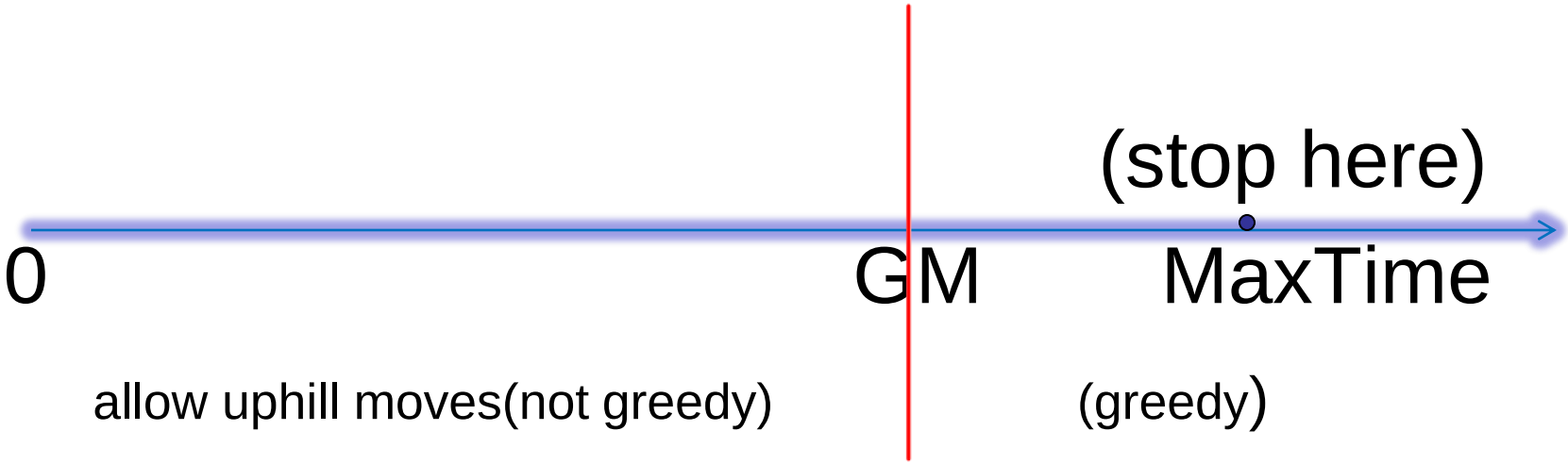


Pick the number of iterations GM when you want the probability of acceptance to be a low P2.

this is valuable for → want stable T, high #'s of iterations mark
→ α^n can give roundig errors

Probability of accepting uphill move for fixed ΔCost

BOARD : change in search over time



Numbers of objective function evaluations

Selecting α for M>1

After GM iterations and average ΔCost,
$T=\alpha^{G}\ T_{o}$,
so we compute the appropriate value of α using the equations on the following slide.

Let P2 be the probability of accepting an uphill move after ***GM*** *interations. If we assume average Δcost (="avgCost" for short) then*

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

Computing α , Given To , P2 , G and M

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

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

After GM iterations

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

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

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

this G is really G, M

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

Same formula as before with greedy point at GM evaluations.

Estimating AvgΔCost

this is the difficult thing

- Our estimates of parameters T_o and α included AvgΔCost.
- We will discuss two ways of estimating AvgΔCost:
  - Method 1-very simple
  - Method 2- do AP objective function evaluations

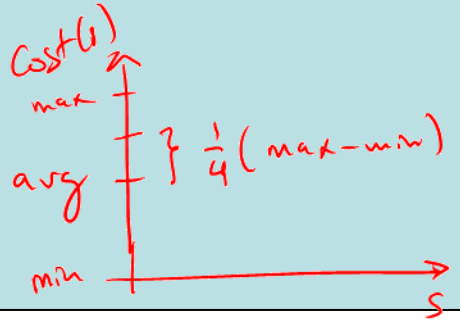
  There are many ways one could get an estimate—these are just two examples

Method 1:Estimating a good avg ΔCOST with no Cost Evaluations before starting Optimization:

- Let us assume you do not want to do any cost evaluations for estimating algorithm parameters (in order to save more cost evaluations to be done in the optimization phase).
- How would you estimate avgΔCost? Ideas???

→ you may know the minimum cost & maximum cost of your cost functions.

→ then, $\text{Avg}\Delta\text{Cost} = \frac{1}{4}(\text{Max Cost} - \text{Min Cost})$



if we pick 2 points, it is likely they will be 0.25(max-min) apart

avg cost will be $\frac{1}{2}$(Max - Min) due to the assumption that they are uniformly distributed

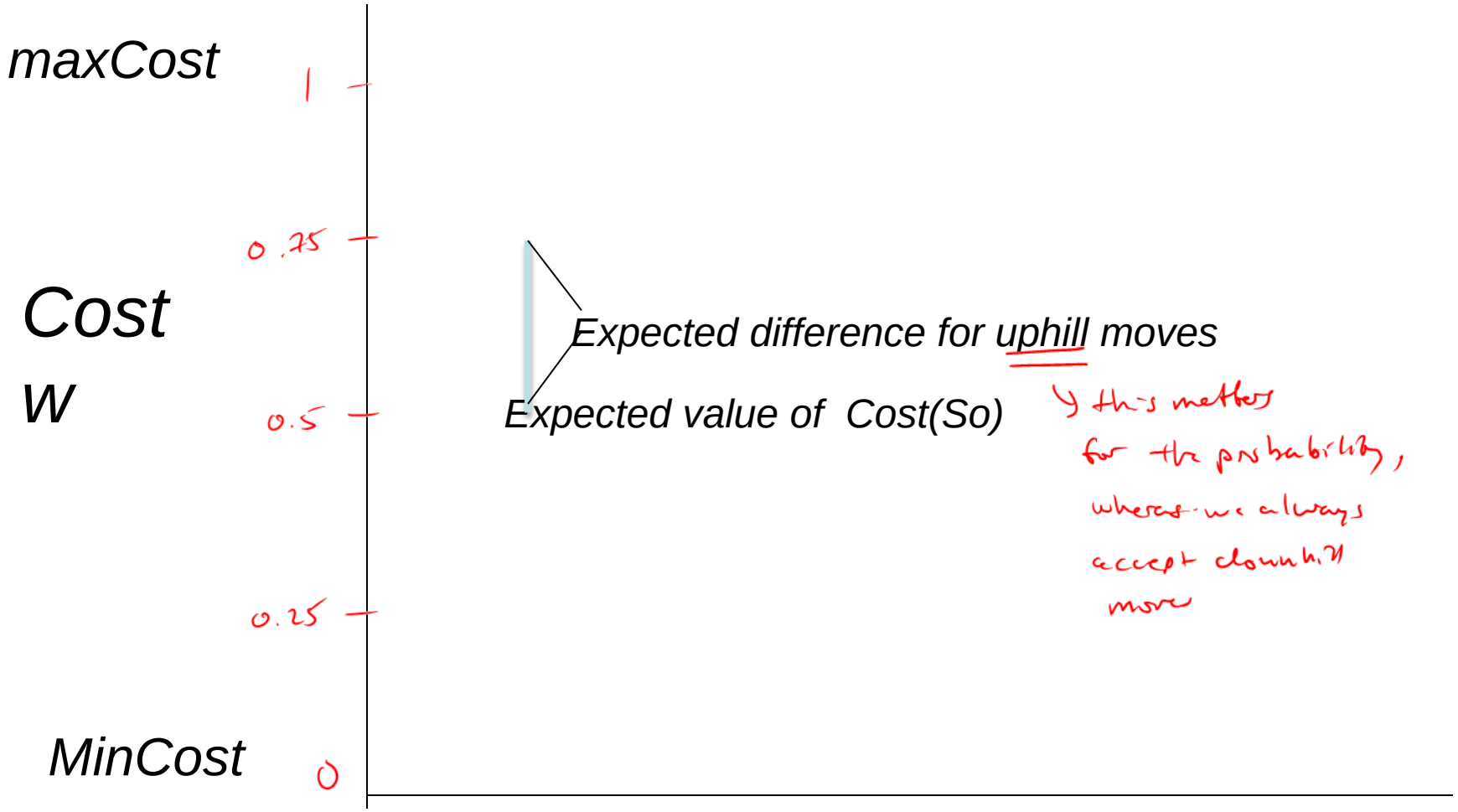
Justification for This Approximation

- Unless you have better information, assume the Cost values are uniformly distributed between the MinCost and MaxCost.
- We can put this on a graph with MaxCost corresponding to 1 and Min Cost corresponding to 0. Then we are assuming the COST is uniformly distributed between 0 and 1 so the expected cost for a random So is .5.
- **Then the uphill moves from So will be uniform between .5(Max Cost-MinCost) and (Max Cost-MinCost).**
- **Hence the average Δ Cost (for uphill moves) is (.25)*(MaxCost-MinCost)**

this is the big assumption

⇒ of course this all depends on the neighborhood.
for a small neighborhood, this will likely overestimate.
⇒ also depends on cost function, obviously

One idea for estimating avg Δcost without new cost functions but with MaxCost and Min Cost Estimate



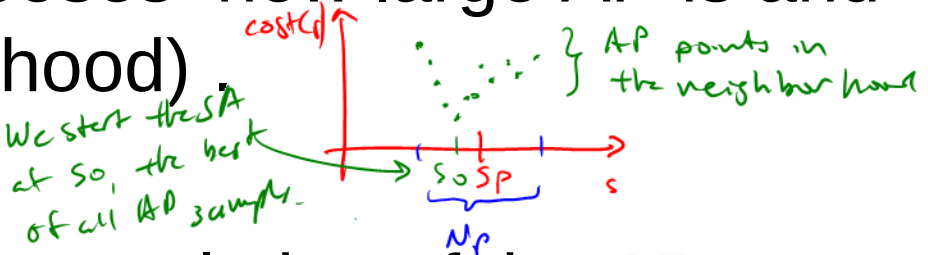
Disadvantage: ???

Method 2: Using AP cost evaluations to get a better approximation for

- In following slides is a way to set T_o that does extra evaluations of Cost (S) to estimate the cost function ΔCost.
- This is still just an approximation, but it can help get a T_o that is probably in the right order of magnitude.

Method 2: Estimating T_o by doing AP extra evaluations of Cost(S): Method 2

- Pick an Sp and evaluate Cost(S_i) for $S_i \in$ N(Sp), i=1,…,AP where the S_i are chosen at random (User chooses how large AP is and N(Sp) is neighborhood) .



- Select So is the best solution of the AP evaluations (e.g. Cost (So) $\leq$Cost(S_i) for all i.
- Thus all the other moves are uphill from So, but you only consider the moves that are in the neighborhood of So.

the Avg Δ cost is computed around the neighborhood of So NOT Sp.

We want to compute an approximation of the average cost change from AP evaluations.

- In this case we are basing our estimate of avgΔcost on the random sampling in Neighborhood around So.
- Then we compute the Δcost for each of the uphill moves in the neighborhood.
- We then take the average of the Δcost among uphill moves.

- With this new value of avgΔcost, we can now recompute the best values for α and To as described earlier.

Estimating an average Δcost around S_o given AP evaluations only in N(So) neighborhood: Method 2

$$\text{avgΔcost} \cong 1/K \sum_{i \in N(S_o)} [\text{cost}(S_i) - \text{cost}(S_o)]$$

$K = |N(S_o)|$ — cardinality of set $N(S_o)$

S_i *is in* $N(S_o)$ *and*
S_i *is uphill of* S_o

Where K is the number of uphill moves S_i *that are in neighborhood N(So) among the AP evaluations in the neighborhood* $N(S_p)$

Summary of Method 2

stochastically chosen

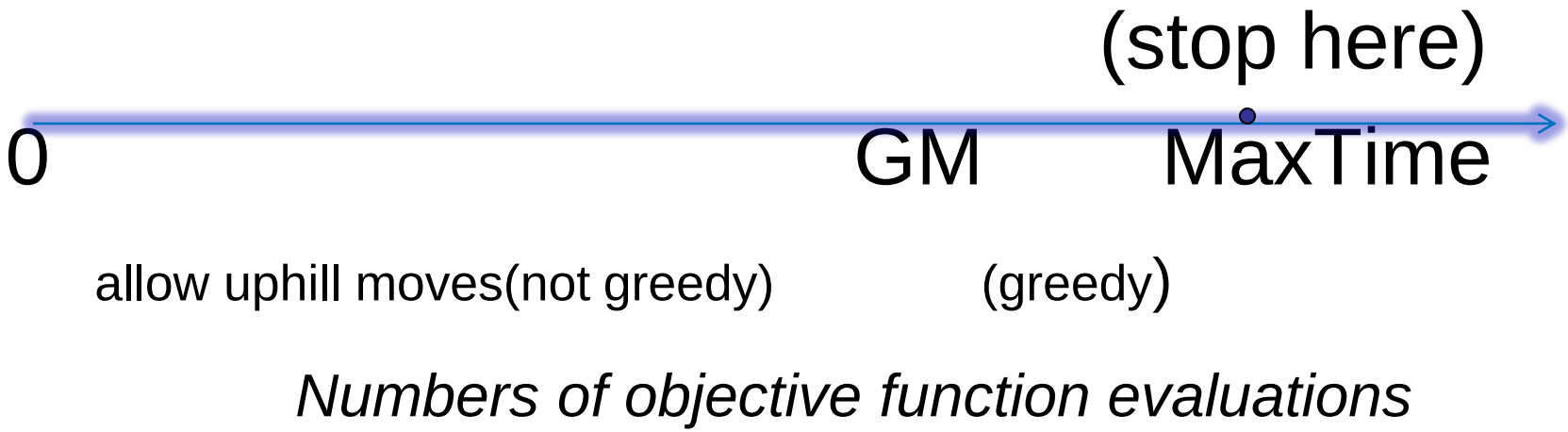
1. Pick a point Sp at random. Compute AP points in the neighborhood of Sp. Pick the best one to be S_o. ⇒ Find local minima around Sp stochastically

2. Then the estimate of avgΔCOST is based only on the points in the neighborhood $N(S_o)$.

3. If you don't have enough points in $N(S_o)$ from the initial AP points, you can compute more cost values in $N(S_o)$ (by increasing AP)

BOARD : change in search over time

- Now you need to do AP + MaxTime evaluations



What are advantages and disadvantages of a using AP objective function evaluations for selecting algorithm parameters) ?

Disadvantages:

- AP reduces number of optimization runs and takes extra programming to set up.
- The methods we have suggested look at *avgΔCost* around a neighborhood of So, but as the search progresses you will be in (unpredictably) different neighborhoods where *avgΔCost* could be quite different.
- There is no way to determine exactly what impact a better *To and α* will have on the outcome so having a really large AP to get an excellent estimate of *To and α* , probably is not worth it

Advantages and Disadvantages of Using AP Cost Evaluations to Estimate SA Parameters (continued)

Advantages:

- Having a more accurate *avgΔCost* estimate to get *To and α* has a good chance to improve the accuracy of the answer.
- You can change your starting value So to be the best answer you found among the AP evaluations. In this way you are getting both a better starting point as well as better algorithm parameters.
- How big should AP be? A good rule of thumb is set to AP to be less than a percentage of *Maxtime (e.g. AP* $\leq$ *f*Maxtime)*, where f is something like $0.0\overline{5}$).

Simulated Annealing--Summary

- Simulated Annealing is a provenly effective heuristic search method for global optimization.
- SA is similar to Greedy search in that in each iteration there is one current solution and you randomly generate the next candidate solution.
- The significant difference between SA and Greedy search is that SA has a positive probability of accepting “uphill” moves. It is this characteristic that enables SA to find global solutions whereas Greedy can only find local solutions.

Genetic Algorithms

- The heuristic search method Genetic Algorithms is inspired by the mechanism Nature uses to “improve” populations.
- Among animals, each parent has genes.
- The genes of a child are directly related to the genes of the parents.
- The ability of an organism to survive (fitness) depends directly on the child’s genes.
- In the algorithm analogy, “genes” are decision variables.

Population Genetics Suggests Heuristic

- “Survival of the fittest” means that children whose genes give high fitness are more likely to survive and produce more children.

- In this way over time, genes with characteristics beneficial for survival appear in an increasing fraction of the population and hence “fitness” of the population improves.

- It is an analogy of this principle that is the basis for the heuristic called “Genetic Algorithm”.

Definition

- Gene- basic genetic element
- Chromosone- a collection of genes
- Allele- The values a gene can take (e.g. we usually have alleles = 0 or 1).
- Examples: Let S be a binary string of length 4 so
  - Chromosome example are 1010, 0110, etc.
  - Genes are the bits in the string
  - Alleles are 0 or 1
- For most heuristic optimization cases one chromosome = “genotype” = “phenotype”
  - (The actual biological distinction is more complex.)
- ”

Definitions (continued)

- For genetic algorithms, the objective function is called “Fitness” which we will call F(S) (not Cost(S)).
- We want to maximize the fitness F(S)
- An iteration of the GA algorithm is called a “generation

Biggest Difference Between Genetic Algorithm and SA

- Simulated annealing carries the single "curS" from one iteration to the next.
- Genetic algorithms carry many solutions from one iteration to the next.
- These many solutions are called a "population"