Slides for VideoTaped Talk

- This video taped talk replaces the regular lecture on Aug. 29.

- You are expected to view the video tape. These slides are NOT intended to replace the video taped lecture.
- (There will also be two lectures videotaped on 9-2 to make up for missed lecture on 8-31. Those slides will be posted separately and both lectures are available on VideoNote.)

handout for video taped lecture replaces 8-29 lecture that covers the beginning of Simulated annealing. (video was Taped 8-26-11)

Homework: Random Walk and Random Sampling

- In random sampling and random walk the next point where cost is evaluated is picked at random and is not dependent upon the value of the Cost(S) at the current point S.
- The Homework describes these two methods and gives you a computer code.
- (You need to log onto Blackboard to get the homework assignment and the computer code. Instructions for getting onto Blackboard are on the Syllabus distributed.)

Hard Problems versus Easy Problems

- For" Polynomial time problems",

  $f(n) \leq O(n^k) = c\, n^k$, for some k and c

Where $f(n)$ is the time for computing the solution to the problem and n is the number of decision variables.

Polynomial time problems are easier problems than "exponential time problems".

However, for k large, polynomial problems can also take a great deal of time for solution. For example, let k=3. If it takes 1 min to solve a problem with n = 10 , How long does it take to solve it with n 10 times as large so n=100?

ANS $f(n) = cn^k$ $\quad 1\text{min} = c\,(10)^3 \quad c = \frac{1}{1000}$

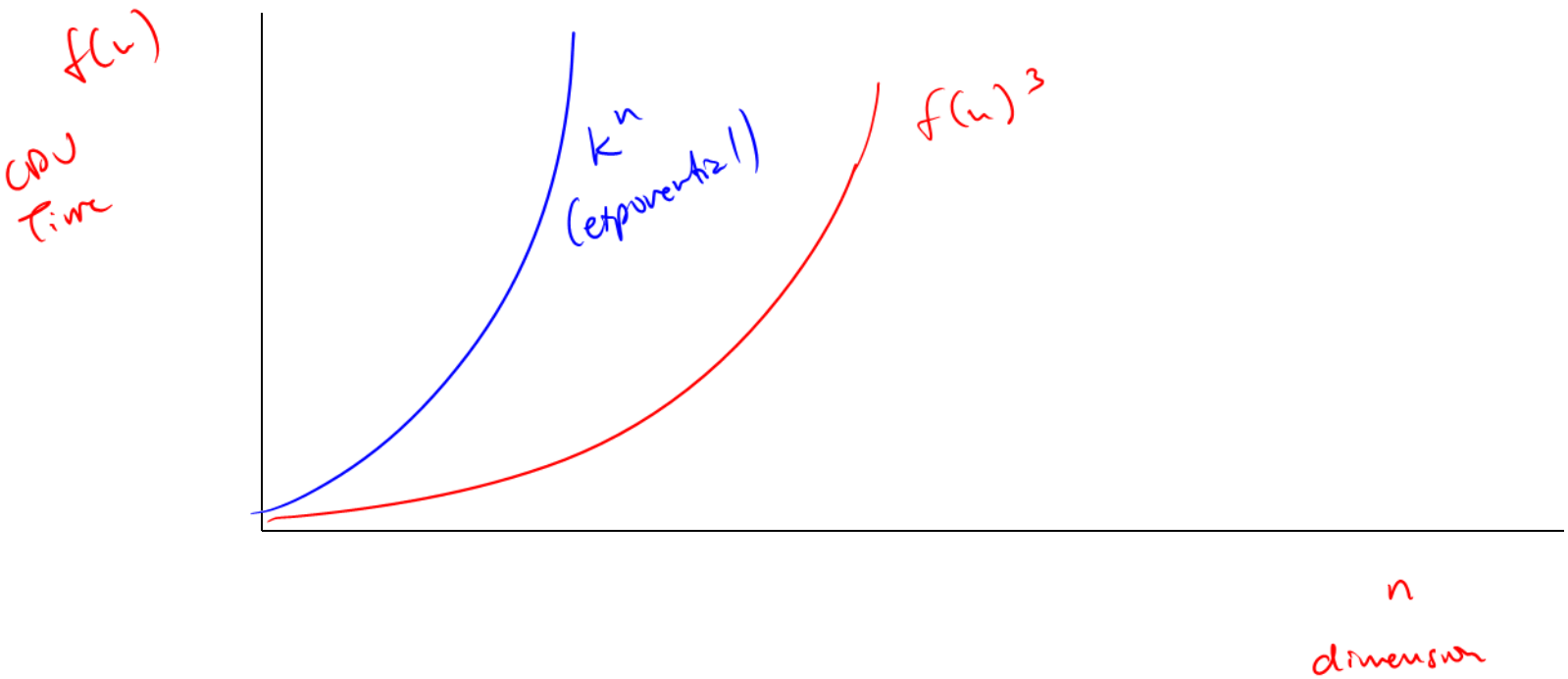
Note: $O(n^3)$ problems are very common

$f(100) = \frac{1}{1000}(100)^3 = (1e^{-3})(1e^{6}) = 1e^{3}$

$\boxed{1000 \text{ min}}$

Effect of Dimension on CPU time required to find solution

- Hence even for polynomial time algorithms, computation can grow quickly as dimension increases



Hard Problems ⇒ NP problems

- There are many problems for which there are no polynomial time algorithms and the problems are exponential time problems.
- These are known as Hard or Intractable problems.
- The xeroxed text gives an example of a number of Hard Problems.
- For Hard problems, we usually have to be satisfied with an approximate solution to the problem.
-

Hard Problems and Heuristics

- A. heuristic technique is a method which seeks **good** (i.e.. near optimal) solutions at a reasonable computational cost without being able to guarantee optimality, and possibly not feasibility.
- There are many problems for which mathematically based methods do guarantee optimality (or at least a solution that is arbitrarily close to the true optimal solution), if enough evaluations of Cost(S) are done.
- However for at least 2000 classes of problems, these clever methods are

  what does this mean?

  – **NP Hard**, meaning from a practical point of view that it is not feasible (in the worst case) to carry out enough iterations of the algorithm to guarantee the final solution is arbitrarily close to the optimal.
- Hence, for these cases it is **not proven** that the more mathematical algorithms (with an inadequate number of iterations) will perform better than heuristic methods since the number of evaluations of Cost(S) is limited.

Hard Problems

- The evidence for the worth of different methods on different classes of large scale problems is largely **empirical** for NP complete problems.
- For example, the simplex method (a clever mathematically based method) for linear programming is NP Hard, but it has been shown empirically to be very effective at finding optimal solutions in relatively few iterations. ⇒ math says NP, in practice it's pretty effective.
- There are many problems for which heuristics perform as well as or better than existing mathematical algorithms given that we are only willing to perform a limited number of iterations.
- In addition there are many problems for which no optimization method exists except heuristic methods.

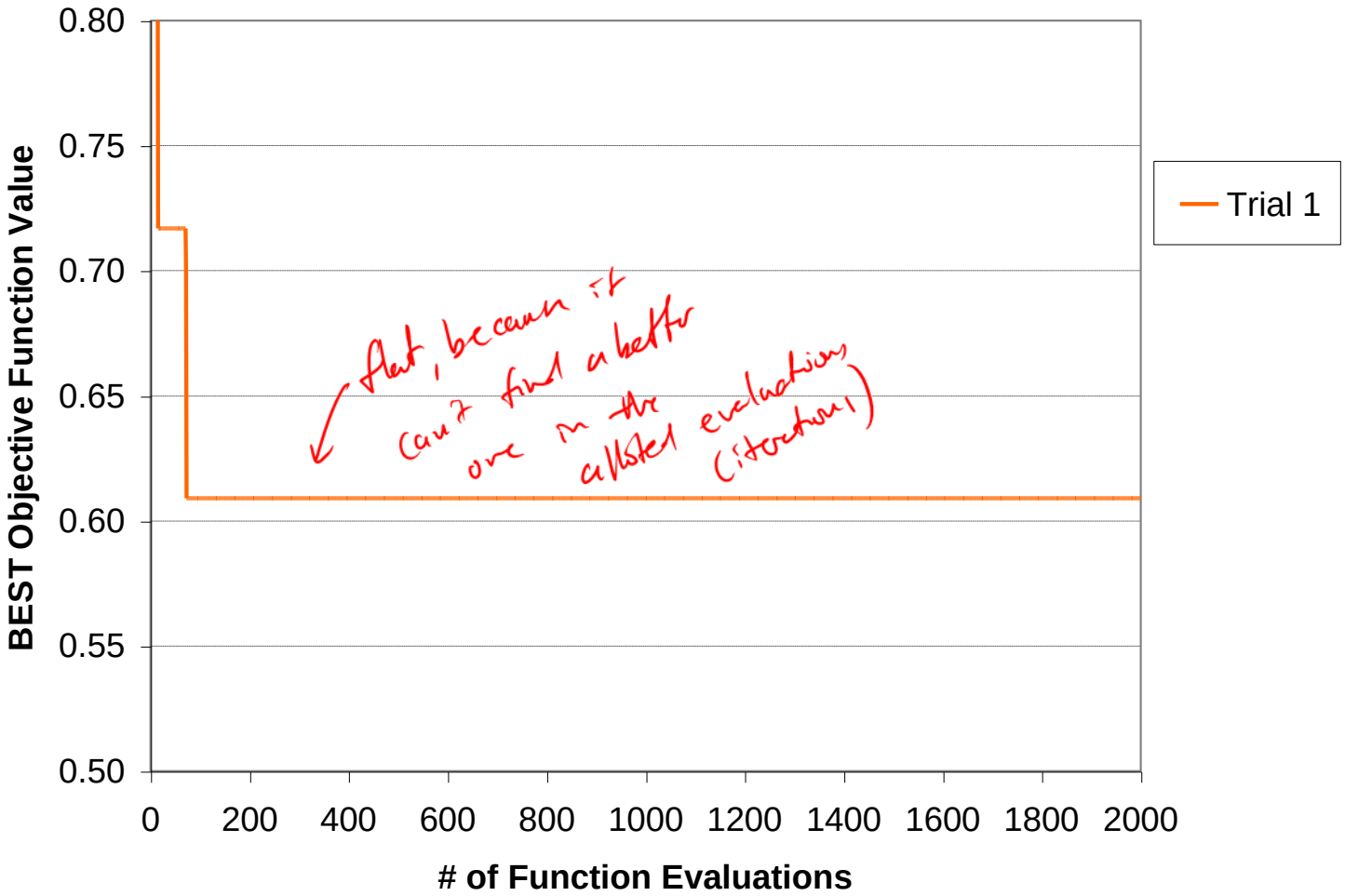
Hard Problems

- ***Bottom Line***:
- For many problems there is no feasible way to guarantee finding the optimal solution.
- Heuristics are a very useful tool for finding good (near optimal) solutions.

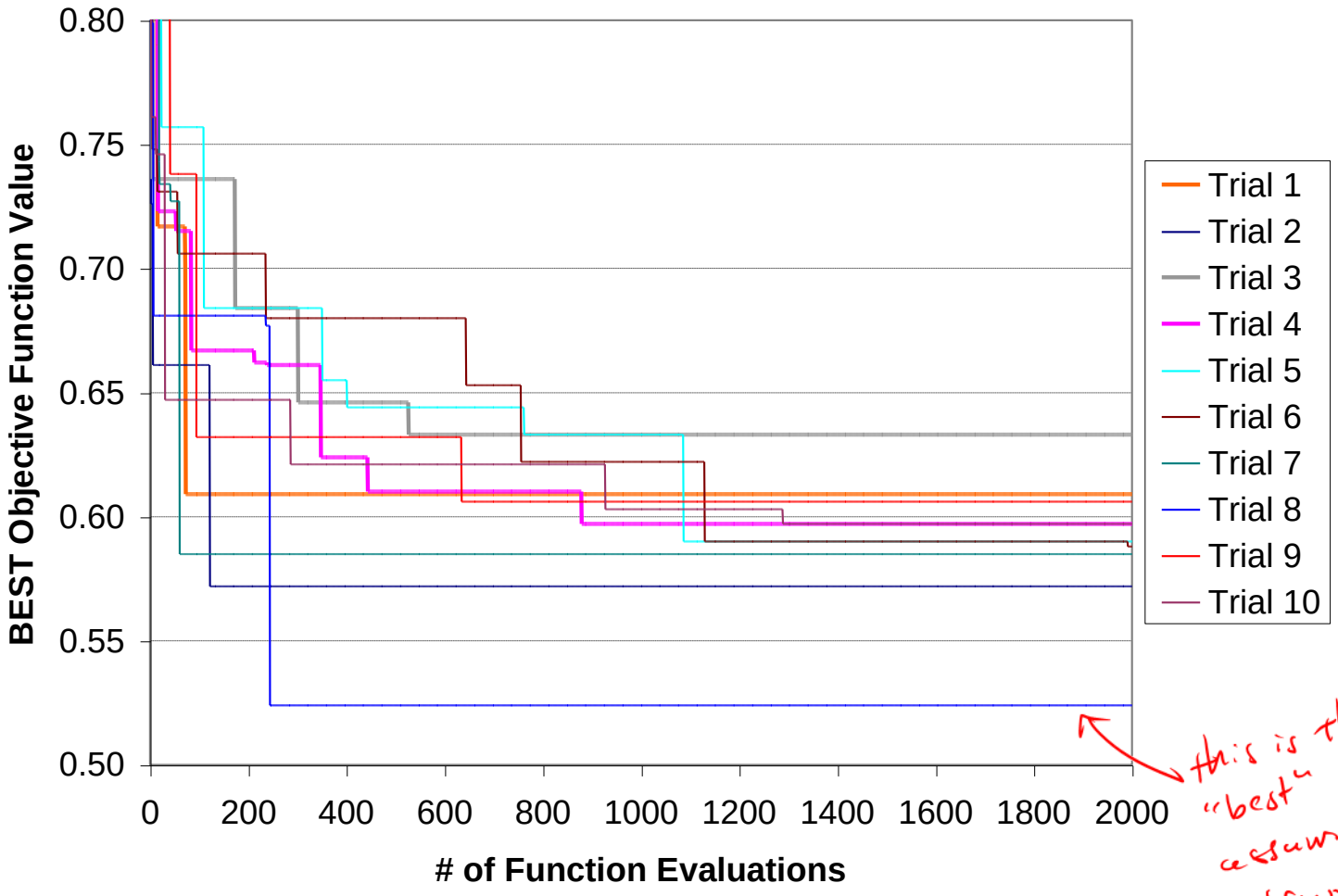
Homework Problem

- The homework asks you to plot the “averages” of multiple trials of stochastic algorithm results.
- Often you are asked to plot the best solution found by a given number of evaluations.
- In the past there has been some confusion about what this means.
- The following slides are designed to help you understand.
- In these slides the “algorithm” is MCS (Monte Carlo Sampling). You don’t need to know the algorithm, since all you are doing is plotting the algorithm output.

Algorithm Performance Assessment over Multiple Trials: Example for MCS

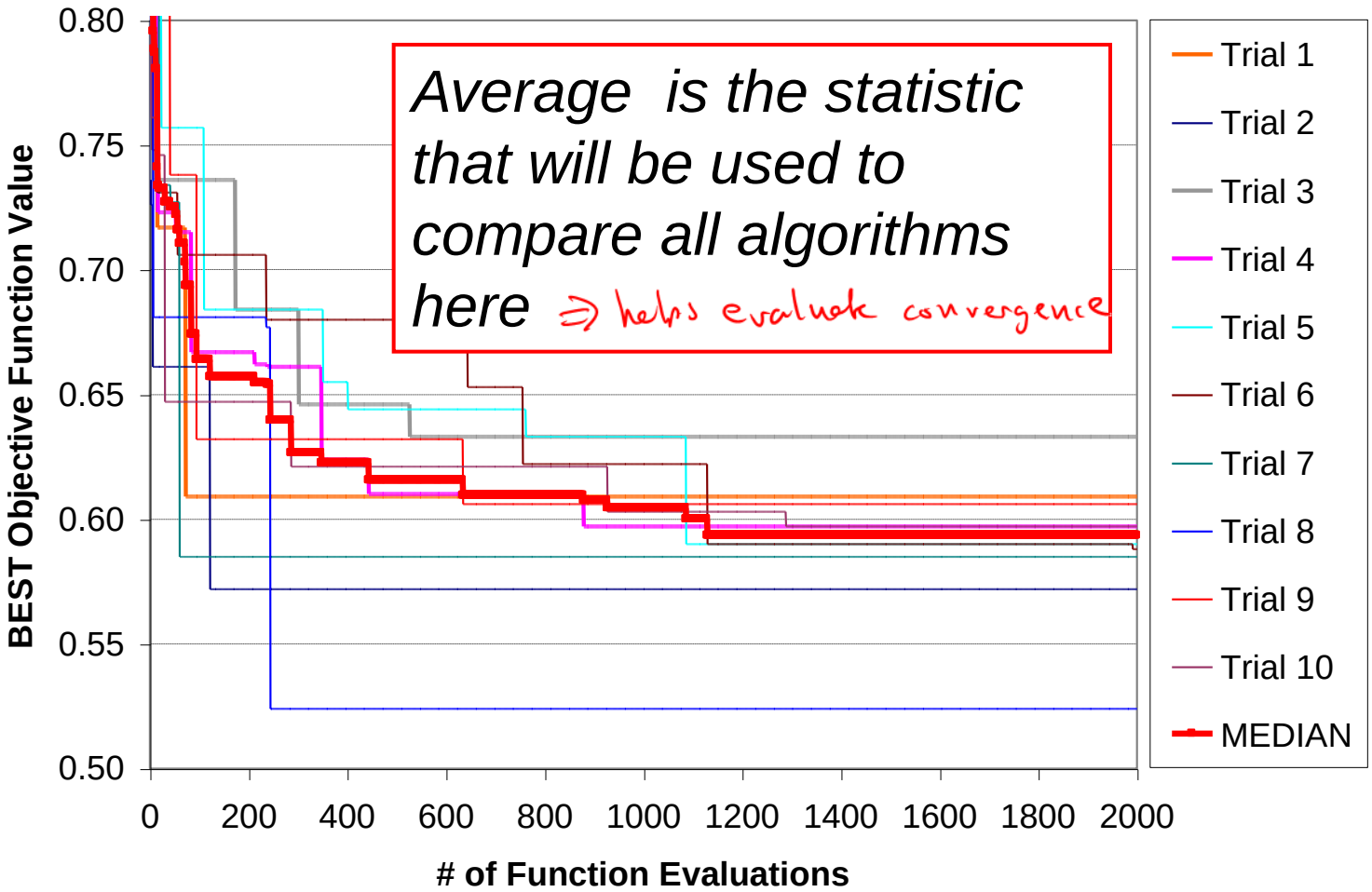


Algorithm Performance Assessment over Multiple Trials: Example for MCS



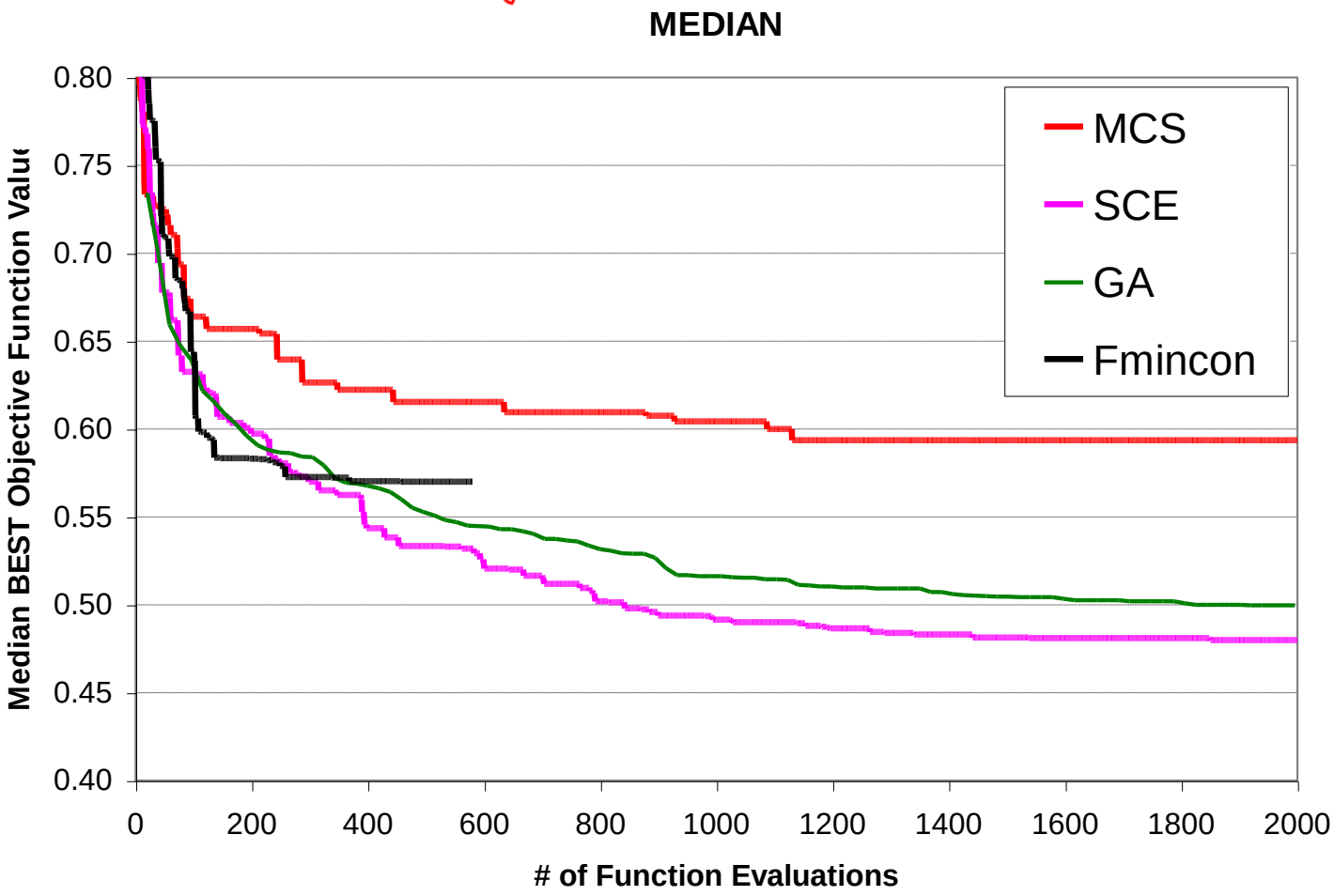
this is the "best" simulation assuming we're trying to minimize

Algorithm Performance Assessment over Multiple Trials: Example for MCS



Example 1 (9 Parameters)

⇒ evaluation of convergence rates of different algorithms. We use the average to do this.



SIMULATED ANNEALING (Reading 49-55)

- Simulated Annealing is based on a representation of a cooling substance.
- In “real” annealing a material is kept at a temperature for a period of time and then the temperature is dropped at distinct points.
- The molecular structure can change in non optimal ways when the material is hot and changes in a more greedy fashion when it is good.

Simulated Annealing Algorithm

- An algorithm inspired by the (metal) annealing process was developed by Metropolis in 1953.
- In 1983, Kirkpatrick applied this idea to a combinatorial optimization problem.
- The algorithm has an initial "temperature" T and a rate α at which the temperature cool.

not really a temperature
just the analogy.
it does make it more
"interesting" though.

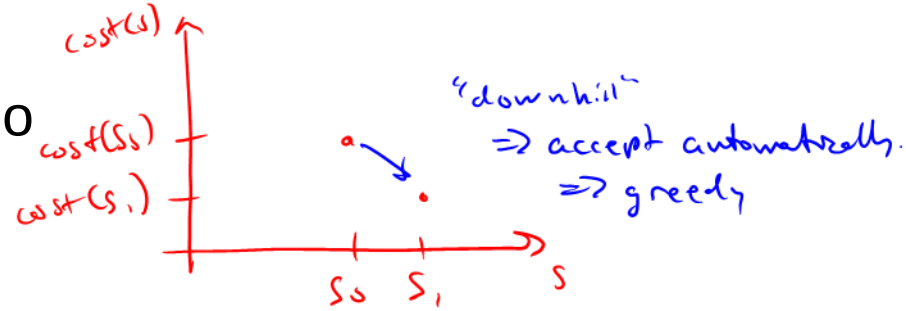
Algorithm Parameters in Heuristics

- Recall that the local greedy search algorithm had parameter (SEARCHRANGE) that determined the size of the neighborhood.
- This is called an “algorithm parameter” since it affects the performance of the algorithm.
- We will discuss later how to determine good values for these algorithm parameters.

SIMULATED ANNEALING

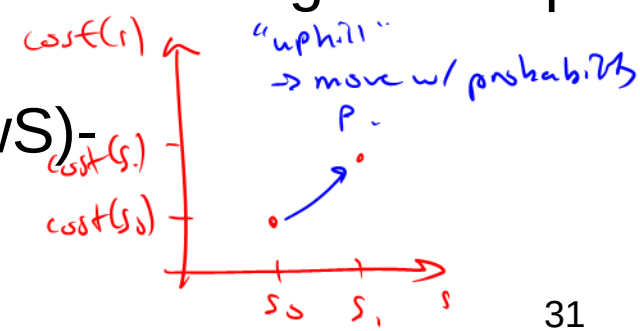
- The simplest version of the algorithm is:

- Step 0. Select the initial guess So for the value of the solution. Set the initial temperature To , and other algorithm parameters including α, β, maxtime. (We assume here that M = 1 and …. $0<\alpha<1$) Initialization: let CurS=So and Time = 1.
- Let BestS = S_o



Simulated Annealing (continued)

- Step 1. (Metropolis step) *CurS is the current value of the estimate of the S that minimizes COST(S), which is the objective function*.
- a. Select at random one element NewS from the neighborhood N(CurS).
- Step 2 If COST(NewS) < COST(CurS) then replace CurS by New S and go to Step 1.
- If COST(NewS) ≥ COST(CurS), then with probability P, replace CurS by New S and go to Step 1. *(This means you are accepting an uphill move with a probability P).* With probability 1-P, do not replace Cur S and go to Step 1.
- The probability P = exp(-[COST(NewS)-COSTS)]/[T*constant])



Simulated Annealing

- Step 2. If Cost(CurS)< Cost(BestS), replace BestS with CurS. Let T = α T and Time =Time + 1. If Time $\leq$ Maxtime go to Step 1. Otherwise stop.
- Note: $\alpha < 1$ so T is decreasing. As T gets smaller, it becomes less likely that uphill moves will be accepted.
- The version above is simplified because it assumes T is changed after each evaluation of J(NewS) (which is equivalent to M=1). These details will be added later.
- Stopped here videotaped lecture 8-26