Random Search

Shin Yoo
SEP592, Summer 2020, School of Computing, KAIST

Random Search

- The polar opposite to the deterministic, examine-everything, search.

- Within the given budget, repeatedly generate a random solution, compare its fitness to the known best, and keep the best one.

Pros and Cons

- VERY easy to implement, inherently automatable, no bias at all.

- Depending on the problem, it may be extremely effective.

- No guidance at all: depending on the problem, it may take forever to obtain a meaningful solution.

Usage of Random Search

- The lack of any guidance provides two useful scenarios.
- First, random search should always be the default **sanity check** against your own search methodology: if it does not no better than random search, you are doing something wrong.

Usage of Random Search

- Somewhat ironically, random search is effective when the underlying problem does not give any guidance to begin with. For example:
  - “Search for the input to program A that will result in program crash”
  - In general, given an arbitrary program, you cannot measure the distance between the current program state and a crash!

Fuzz Testing

- Infinite Monkey Theorem: “Thousand monkeys at a thousand typewriters will eventually type out the entire works of Shakespeare"
- Basic idea: provide a stream of random input to the program, until it crashes (=our Shakespeare).
  - Either a stream of really random bits (naive), or
  - Well-formed input randomly mutated (more effective)