CS101: Intro to Computing
Fall 2015

Lecture 22

Administrivia

- Homework 13 due today
- Midterm 2 on Monday
  - Practice exam released today
- Homework 14 released ***after*** the exam

REVIEW


10
20
30
40
50

```
def f(x):
  if x==(31,25,14):
    return 10000
  else:
    return 0
```

What does x represent in this code?

a) The combination

b) The reward

c) The safe

d) The moon

```
max_score=0
best=None
n=range(0,60)
for x in ??? :
  if f(x)>max_score:
    best=x
    max_score=f(x)
```

What should replace the ???

a) itertools.product(n,3)

b) itertools.permutations(n,3)

c) itertools.combinations(n,3)

OPTIMIZATION

Optimization

- Given a function f(x), find the x such that f(x) is maximized/minimized
- Goal: search through the domain for the optimum x
- Many clever ways to do this, but let's start with something naïve

Example

On vacation, you found n items of varying weight and value.

Your bag has a weight limit of 50 pounds.

What is the best set of items to take on the flight?

Set up problem

```
import numpy as np
n=10
items=range(n)
weights=np.random.rand(n)*50
values=np.random.rand(n)*100
```

Set up function

```
def f(x,w,v):
    total_weight=0
    total_value=0
    for i in x:
       total_weight+=w[i]
       total_value+=v[i]
   if total_weight>=50:
        return 0
     else:
       return total_value
```

Brute-force search

- Search through the ***entire*** domain of f
- Also called an “exhaustive search”
- How did we do this for our problem?

Brute force optimization

```
best_value=0.0
best_items=[]

for i in range(0,n+1):
  for x in itertools.combinations(items,i):
    v=f(x,weights,values)
    if v>best_value:
      best_value=v
      best_items=x
```

HEURISTIC OPTIMIZATION

Heuristic

- Technique to trade optimality for speed
- ***NOT*** guaranteed to find the best solution
- ***NOT*** guaranteed to be efficient
- Take a shortcut, take risks
- Find a solution that's "good enough"

Local Optima

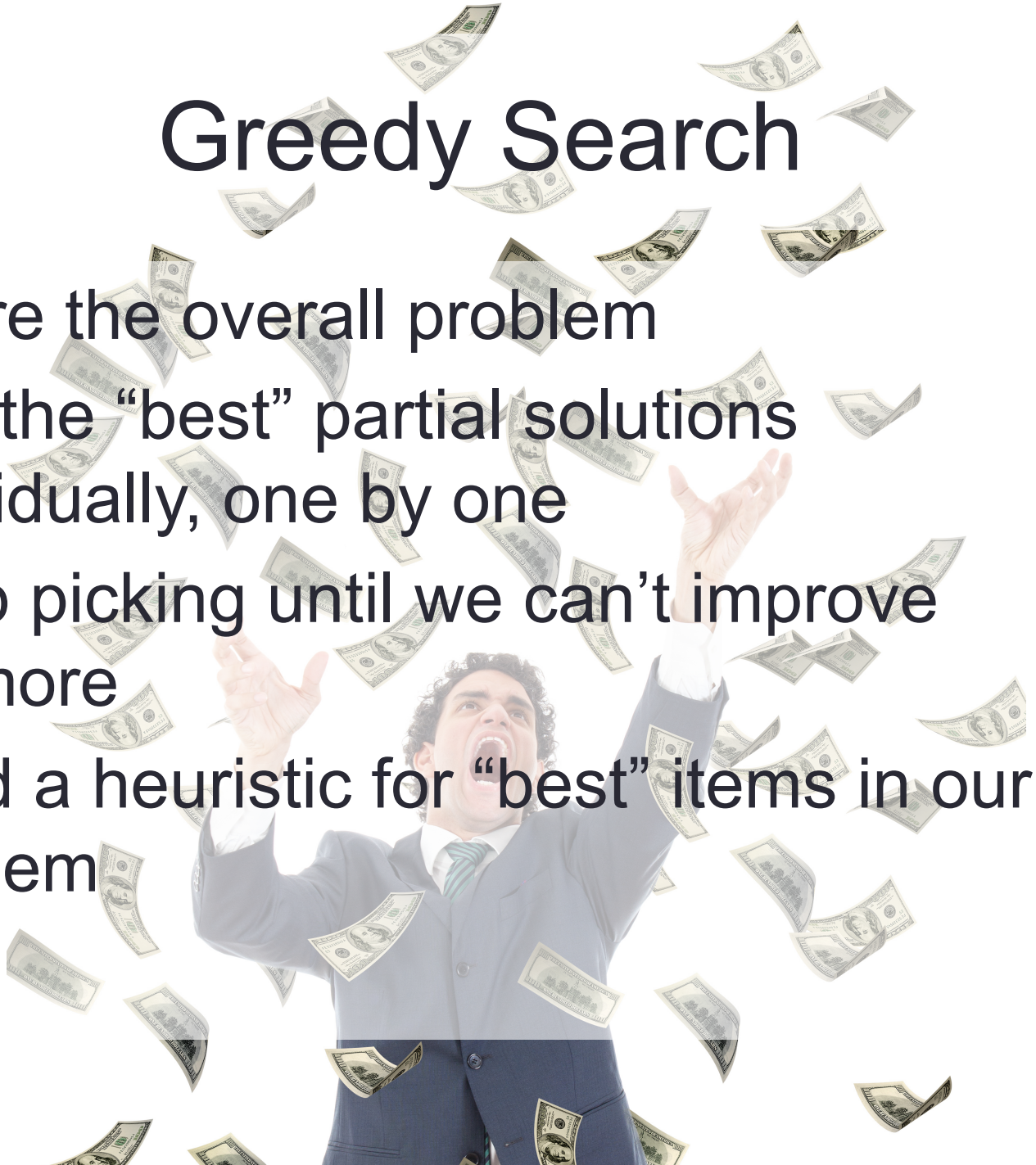
- When using these approaches, it is possible to get “stuck”
- Reach the “top” of ***a*** hill
- But we were not on top of the ***tallest*** hill
- This is called a ***local optimum***
- True optimum is the ***global optimum***

Random search

- Randomly sample the domain of f
- Might not find the *true* optimum
- Might find one that is “good enough”
- How can we sample from the domain of our problem?

Greedy Search

- Ignore the overall problem
- Pick the “best” partial solutions individually, one by one
- Keep picking until we can’t improve anymore
- Need a heuristic for “best” items in our problem



Hill Climbing

- Keep track of our “current” solution
- Change one element of the solution
- If change improves the solution, this is new “current” solution
- Repeat until we can’t improve anymore

Steepest Ascent Hill Climbing

- Keep track of our “current” solution
- Incrementally change ***all*** elements of the solution
- Pick the change with best improvement
  - “Steepest” slope
- If change improves the solution, this is new “current” solution
- Repeat until we can’t improve anymore