Discrete Optimization

The Knapsack Problem: Greedy Algorithms Intuition

Goals of the Lecture

- Introduce greedy algorithms
  - What would Indiana Jones do?

The Temple is Collapsing

$1 Million
2kg

$1 Million
2kg

$1 Million
2kg

$10 Million
5kg

$10 Million
5kg

$14 Million

can we
do better?

$13 Million
8kg

$7 Million
3kg

Maximum
Capacity
10kg

Greedy Algorithms

- Idea: “Take the most valuable items first”
  - greedy algorithm

- For one problem, there are many possible greedy algorithms.
  - some will do better than others

Until Next Time

Citations

Stone Foundation Tablet with Inscription of Gudea - 41221 (http://commons.wikimedia.org/wiki/File:Sumerian _-_Stone_Foundation_Tablet_with_Inscription_of_Gudea_-_Walters_41221_-_View_A.jpg). Artist Unknown. Walters Art Museum [Public domain, CC-BY-SA-3.0 (http://creativecommons.org/licenses/by-sa/3.0)], via Wikimedia Commons

Stone Foundation Tablet with Inscription of Gudea - 41220 (http://commons.wikimedia.org/wiki/File:Sumerian_-_Stone_Foundation_Tablet_with_an_Inscription_of_Gudea_-_Walters_41220_-_View_A.jpg). Artist Unknown. Walters Art Museum [Public domain, CC-BY-SA-3.0 (http://creativecommons.org/licenses/by-sa/3.0) via Wikimedia Commons

Ring with the engraved portrait of Ptolemy VI Philometor (http://commons.wikimedia.org/wiki/File:Ring_with_engraved_portrait_of_ Ptolemy_VI_Philometor_(3rd%E2%80%932nd_century_BCE)_-_2009.jpg<http://commons.wikimedia.org/wiki/File:Ring_with_engraved_portrait_of_Ptolemy_VI_Philometor_(3rd-2nd_century_BCE)_-_2009.jpg>) By Unknown. (Photographed by PHGCOM in 2009.) [Public domain], via Wikimedia Commons

the mask of agamemnon (http://www.flickr.com/photos/rosemania/5705122218/) by Xuan Che (http://www.flickr.com/people/rosemania/) CC BY-2.0 (http://creativecommons.org/licenses/by/2.0/deed.en)

Terracotta Warrior (http://www.flickr.com/photos/59627558@N00/4677378806/) by fixermark (http://www.flickr.com/photos/59627558@N00/) CC BY 2.0 (http://creativecommons.org/licenses/by/2.0/deed.en)

Discrete Optimization

The Knapsack Problem: Greedy Algorithms

Goals of the Lecture

- Compare different greedy algorithms
  - knapsack example

What is Greedy?

- Assume the it's "easy" to build a feasible solution

- Key Idea:
  - Build a solution by picking items one at a time

- Called: greedy algorithms or heuristics

The Temple is Collapsing

$1 Million
2kg

$1 Million
2kg

$1 Million
2kg

$10 Million
5kg

$10 Million
5kg

Which items to take?

$13 Million
8kg

$7 Million
3kg

Maximum
Capacity
10kg

Knapsack Greedy Algorithms

- Idea 1:
  - More items is best, start with small ones and take as many as you can

Idea: The More Items the Better

$1 Million
2kg

$1 Million
2kg

$1 Million
2kg

$10 Million
5kg

$10 Million
5kg

Total: $10 Million

$13 Million
8kg

$7 Million
3kg

Maximum
Capacity
10kg

Knapsack Greedy Algorithms

- Idea 1: ($10 Million)
  - More items is best, start with small ones and take as many as you can
- Idea 2:
  - Valuable items are best, start with the most valuable items

Idea: More Valuable is Better

$1 Million
2kg

$1 Million
2kg

$1 Million
2kg

$10 Million
5kg

$10 Million
5kg

Total: $14 Million

$13 Million
8kg

$7 Million
3kg

Maximum
Capacity
10kg

Knapsack Greedy Algorithms

- Idea 1: ($10 Million)
  - More items is best, start with small ones and take as many as you can
- Idea 2: ($14 Million)
  - Valuable items are best, start with the most valuable items
- Idea 3:
  - Value density! dollars per kilogram

Idea: The More Items the Better

0.5
M/kg

$1 Million
2kg

$1 Million
2kg

$1 Million
2kg

2.0
M/kg

$10 Million
5kg

$10 Million
5kg

Total: $18 Million

1.8
M/kg

$13 Million
8kg

2.3
M/kg

$7 Million
3kg

Maximum
Capacity
10kg

Knapsack Greedy Algorithms

- Is $18 million dollars the best we can do?
  - optimal?

Total: $20 Million!

$10 Million
5kg

$10 Million
5kg

Maximum
Capacity
10kg

Greedy Algorithms Overview

- For one problem, there are **many** possible greedy algorithms.
  - some will do better than others
  - depends on the input!
- Advantages
  - quick to design and implement
  - can be very fast
- Problems
  - no quality guarantees (in general)
  - quality can vary widely on the input
  - problem feasibility needs to be “easy”

The Essence of this Class

- We can always start with greedy

- Going beyond greedy
  - Constraint Programming
  - Local Search
  - Mixed Integer Programming

- Ways to
  - reliably find feasible solutions
  - reliably build high-quality solutions
    - robust to different inputs
  - ideally, proving those solutions are the best

Until Next Time

Citations

Stone Foundation Tablet with Inscription of Gudea - 41221 (http://commons.wikimedia.org/wiki/File:Sumerian _-_Stone_Foundation_Tablet_with_Inscription_of_Gudea_-_Walters_41221_-_View_A.jpg). Artist Unknown. Walters Art Museum [Public domain, CC-BY-SA-3.0 (http://creativecommons.org/licenses/by-sa/3.0)], via Wikimedia Commons

Stone Foundation Tablet with Inscription of Gudea - 41220 (http://commons.wikimedia.org/wiki/File:Sumerian_-_Stone_Foundation_Tablet_with_an_Inscription_of_Gudea_-_Walters_41220_-_View_A.jpg). Artist Unknown. Walters Art Museum [Public domain, CC-BY-SA-3.0 (http://creativecommons.org/licenses/by-sa/3.0) via Wikimedia Commons

Ring with the engraved portrait of Ptolemy VI Philometor (http://commons.wikimedia.org/wiki/File:Ring_with_engraved_portrait_of_ Ptolemy_VI_Philometor_(3rd%E2%80%932nd_century_BCE)_-_2009.jpg<http://commons.wikimedia.org/wiki/File:Ring_with_engraved_portrait_of_Ptolemy_VI_Philometor_(3rd-2nd_century_BCE)_-_2009.jpg>) By Unknown. (Photographed by PHGCOM in 2009.) [Public domain], via Wikimedia Commons

the mask of agamemnon (http://www.flickr.com/photos/rosemania/5705122218/) by Xuan Che (http://www.flickr.com/people/rosemania/) CC BY-2.0 (http://creativecommons.org/licenses/by/2.0/deed.en)

Terracotta Warrior (http://www.flickr.com/photos/59627558@N00/4677378806/) by fixermark (http://www.flickr.com/photos/59627558@N00/) CC BY 2.0 (http://creativecommons.org/licenses/by/2.0/deed.en)

Discrete Optimization

The Knapsack Problem: Modeling

Goals of the Lecture

- How to formalize an optimization task as a mathematical model

The (1-Dimensional) Knapsack Problem

- Given a set of items **I**, each item i $\in$ **I**
  characterized by
  - its weight w_i
  - its value v_i

  and
  - a capacity K for a knapsack

  find the subset of items in **I**
  - that has maximum value
  - does not exceed the capacity K of the knapsack

Optimization Models

- How to model an optimization problem
  - choose some decision variables
    - they typically encode the result we are interested in
  - express the problem constraints in terms of these variables
    - they specify what the solutions to the problem are
  - express the objective function
    - the objective function specifies the quality of each solution
- The result is an optimization model
  - It is a declarative formulation
    - specify the “what”, not the “how”
  - There may be many ways to model an optimization problem

A Knapsack Model

- Decision variables
  - x_i denotes whether item i is selected in the solution
    - $x_i = 1$ means the item is selected
    - $x_i = 0$ means that it is not selected
- Problem constraint
  - The selected item cannot exceed the capacity of the knapsack $\sum_{i \in I} w_i x_i \leq K$
- Objective function
  - Captures the total value of the selected items $\sum_{i \in I} v_i x_i$

A Knapsack Model

- Putting it all together

$$\begin{aligned}
&\text{maximize} && \sum_{i \in I} v_i \; x_i \\
&\text{subject to} \\
& && \sum_{i \in I} w_i x_i \leq K \\
& && x_i \in \{0, 1\} \quad (i \in I)
\end{aligned}$$

Exponential Growth

- How many possible configurations?
  - (0,0,0,...,0), (0,0,0,...,1), ..., (1,1,1,...,1)
- Not all of them are feasible
  - They cannot exceed the capacity of the knapsack
- How many are they?
  - $2^{|I|}$
- How much time to explore them all?
  - 1 millisecond to test a configuration
  - if $|I| = 50$, it will take 1,285,273,866 centuries

Until Next Time

Discrete Optimization

The Knapsack Problem: Dynamic Programming

Goals of the Lecture

- How to find the **BEST** knapsack solutions
- Using dynamic programming

Dynamic Programming

- Widely used optimization technique
  - for certain classes of problems
  - heavily used in computational biology

- Basic principle
  - divide and conquer
  - bottom up computation

Dynamic Programming

- Basic conventions and notations
  - assume that `I` = {1,2,...,n}
  - O(k,j) denotes the optimal solution to the knapsack problem with capacity k and items [1..j]

$$\begin{aligned} \text{maximize} \quad & \sum_{i \in 1..j} v_i \; x_i \\ \text{subject to} \quad & \\ & \sum_{i \in 1..j} w_i x_i \leq k \\ & x_i \in \{0,1\} \;\; (i \in 1..j) \end{aligned}$$

- We are interested in finding out the best value O(K,n)

Recurrence Relations (Bellman Equations)

- Assume that we know how to solve
  - O(k,j-1) for all k in 0..K
- We want to solve O(k,j)
  - We are just considering one more item, i.e., item j.
- If $w_j \leq k$, there are two cases
  - Either we do not select item j, then the best solution we can obtain is O(k,j-1)
  - Or we select item j and the best solution is v_j + O(k-w_j,j-1)
- In summary
  - O(k,j) = max(O(k,j-1), v_j + O(k-w_j,j-1)) if $w_j \leq k$
  - O(k,j) = O(k,j-1) otherwise
- Of course
  - O(k,0) = 0 for all k

Recurrence Relations

- We can write a simple program

```
int O(int k,int j) {
   if (j == 0)
     return 0;
   else if (w_j <= k)
     return max(O(k,j-1),v_j + O(k-w_j,j-1));
   else
     return O(k,j-1)
}
```

- How efficient is this approach?

Recurrence Relations - Fibonacci Numbers

- We can write a simple program for finding fibonacci numbers

```
int fib(int n) {
   if (n == 0 || n == 1)
      return 1;
   else
        return fib(n-2) + fib(n-1);
}
```

- How efficient is this approach?
  - we are solving many times the same subproblem
    - fib(n-1) requires fib(n-2) which we have already solved
    - fib(n-3) requires fib(n-4) which we have already solved

Dynamic Programming

- Compute the recursive equations bottom up
  - start with zero items
  - continue with one item
  - then two items
  - ...
  - then all items

$$
\begin{aligned}
&\text{maximize} && 5x_1 + 6x_2 + 3x_3 \\
&\text{subject to} \\
& && 4x_1 + 5x_2 + 2x_3 \leq 9 \\
& && x_i \in \{0, 1\} \quad (i \in 1..3)
\end{aligned}
$$

Dynamic Programming - Example

- How to find which items to select?

Capacity	0
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

Take items 1 and 2

Trace back

Dynamic Programming - Example

$$\begin{aligned} \text{maximize} \quad & 16x_1 + 19x_2 + 23x_3 + 28x_4 \\ \text{subject to} \quad & 2x_1 + 3x_2 + 4x_3 + 5x_4 \leq 7 \\ & x_i \in \{0, 1\} \quad (i \in 1..4) \end{aligned}$$

$$x_1 = 1, x_2 = 0, x_3 = 0, x_4 = 1$$

Capacity	0	1	2	3	4
0					
1					
2					
3					
4					
5					
6					
7					

Dynamic Programming

- What is the complexity of this algorithm?
  - time to fill the table
  - i.e., O(K n)

- Is this polynomial?
  - How many bits does K need to be represented on a computer?
    - log(K) bits
  - Hence the algorithm is in fact exponential in terms of the input size
    - pseudo-polynomial algorithm
    - “efficient” when K is small

Until Next Time

Discrete Optimization

The Knapsack Problem: Branch and Bound

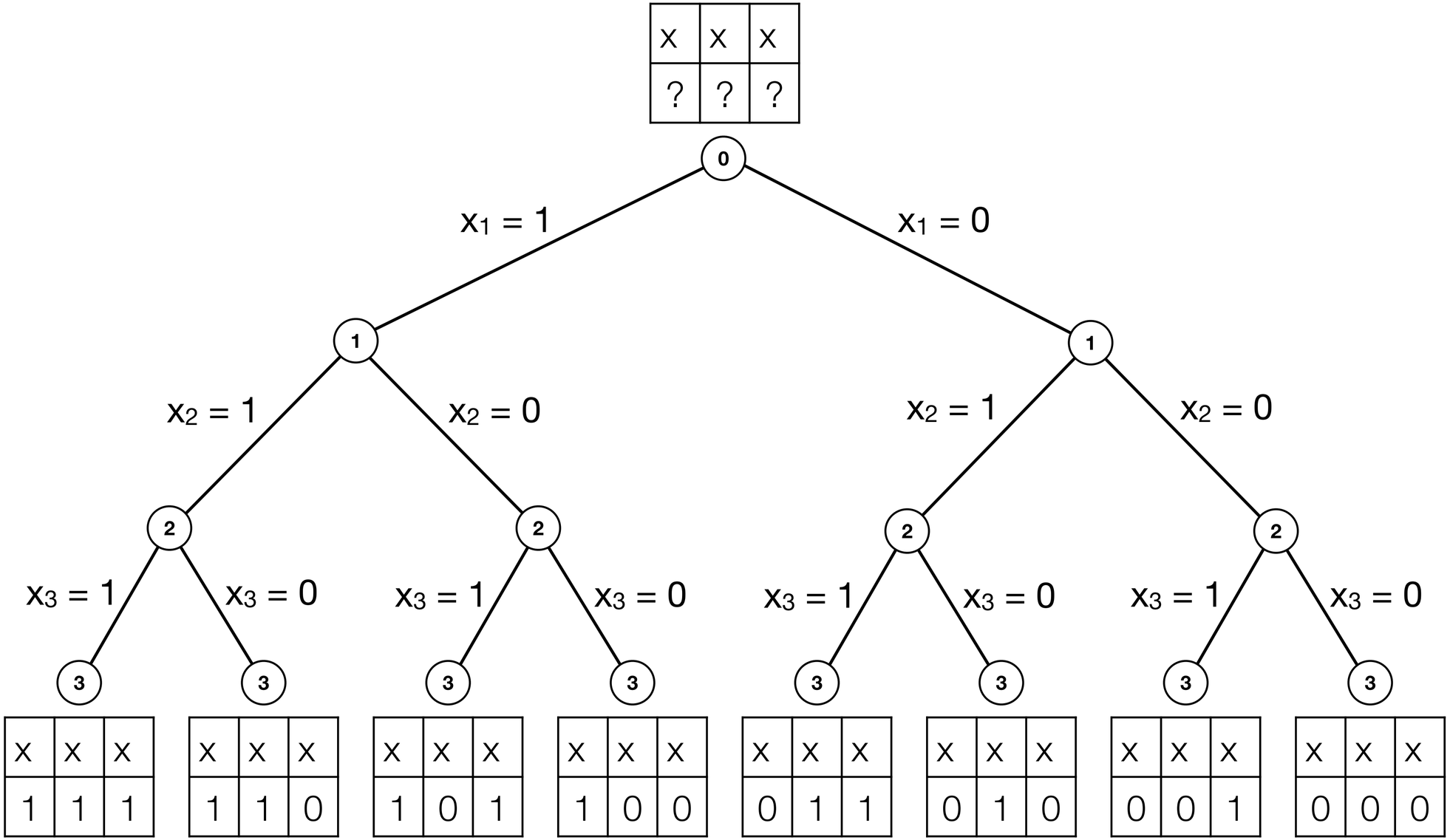
Goals of the Lecture

- Introduce branch and bound
- The value of relaxation

One-Dimensional Knapsack

$$\begin{aligned}
\text{maximize} \quad & 45x_1 + 48x_2 + 35x_3 \\
\text{subject to} \quad & \\
& 5x_1 + 8x_2 + 3x_3 \leq 10 \\
& x_i \in \{0, 1\} \quad (i \in 1..3)
\end{aligned}$$

Exhaustive Search



Branch and Bound

- Iterative two steps
  - branching
  - bounding
- Branching
  - split the problem into a number of subproblems
    - like in exhaustive search
- Bounding
  - find an ***optimistic estimate*** of the best solution to the subproblem
    - maximization: upper bound
    - minimization: lower bound

Branch and Bound

- How to find this optimistic estimate?
  - Relaxation!

- Optimization is the art of relaxation

A Knapsack Model

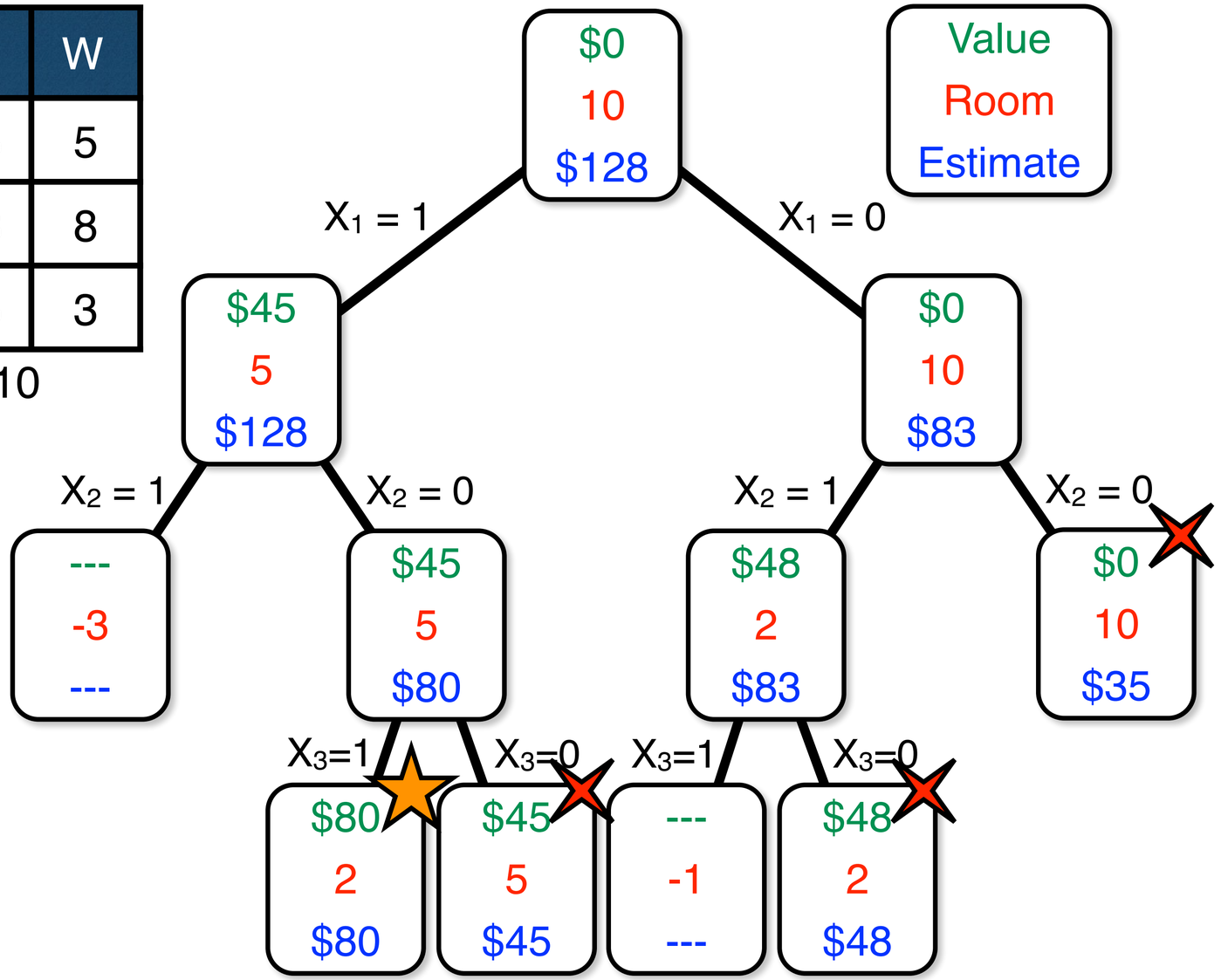
$$
\begin{aligned}
\text{maximize} \quad & 45x_1 + 48x_2 + 35x_3 \\
\text{subject to} \quad & \\
& 5x_1 + 8x_2 + 3x_3 \leq 10 \\
& x_i \in \{0, 1\} \quad (i \in 1..3)
\end{aligned}
$$

- What can we relax?
  - we can relax the capacity constraint

Depth-First Branch and Bound

i	V	W
1	45	5
2	48	8
3	35	3

K = 10



A Knapsack Model

$$
\begin{aligned}
\text{maximize} \quad & 45x_1 + 48x_2 + 35x_3 \\
\text{subject to} \quad & \\
& 5x_1 + 8x_2 + 3x_3 \leq 10 \\
& x_i \in \{0, 1\} \quad (i \in 1..3)
\end{aligned}
$$

- Can we relax something else?

A Knapsack Model

- What if the items are bars of Belgian chocolate?
  - In that case, we could actually take a fraction of the bar!

subject

- This is
  - we will
  - we rela

A Knapsack Model

- Can we solve a knapsack when we can take parts of the items?
  - order the items by decreasing value of V_i/W_i
  - "most value per kilo"

$$\begin{aligned} \text{maximize} \quad & 45x_1 + 48x_2 + 35x_3 \\ \text{subject to} \quad & \\ & 5x_1 + 8x_2 + 3x_3 \leq 10 \\ & 0 \leq x_i \leq 1 \quad (i \in 1..3) \end{aligned}$$

A Knapsack Model

- How to solve the relaxation now?
  - select the items while the capacity is not exhausted
  - select a fraction of the last item

$$\begin{aligned} \text{maximize} \quad & 45x_1 + 48x_2 + 35x_3 \\ \text{subject to} \quad & \\ & 5x_1 + 8x_2 + 3x_3 \leq 10 \\ & 0 \leq x_i \leq 1 \quad (i \in 1..3) \end{aligned}$$

- In this example,
  - $V_1/W_1 = 9$, $V_2/W_2 = 6$, $V_3/W_3 = 11.7$
  - select items 3 and 1
  - select 1/4 of item 2
  - estimation: 92

A Knapsack Model

- Why is correct?

$$\text{let } x_i = \frac{y_i}{v_i}$$

$$\begin{aligned} &\text{maximize} && \textstyle\sum_{i \in 1..j} y_i \\ &\text{subject to} && \\ & && \textstyle\sum_{i \in 1..j} \frac{w_i}{v_i} y_i \leq K \\ & && 0 \leq y_i \leq 1 \quad (i \in 1..j) \end{aligned}$$

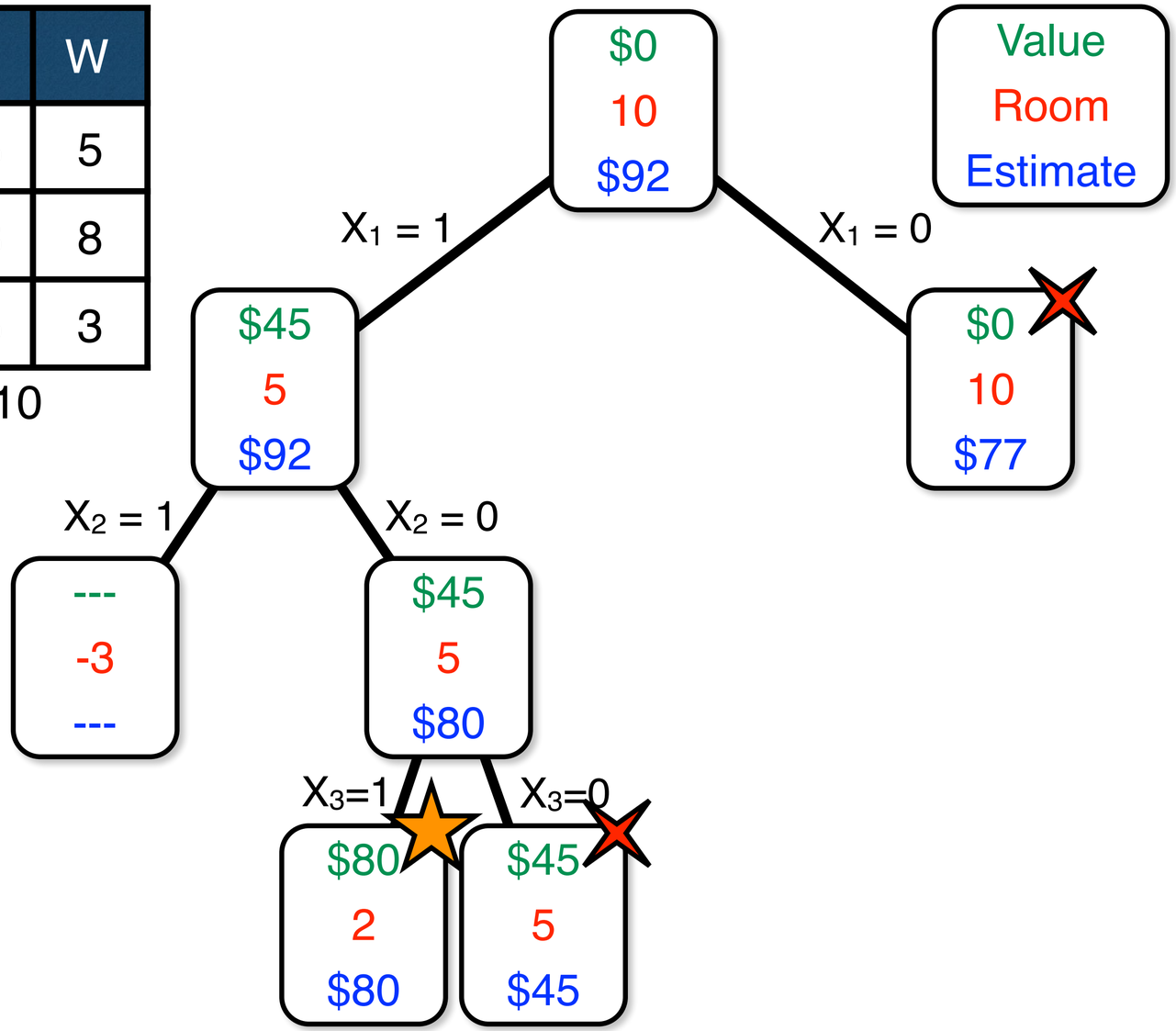
A Knapsack Model

- Of course, if the items are archeological artifacts...

Depth-First Branch and Bound

i	V	W
1	45	5
2	48	8
3	35	3

K = 10



Until Next Time

Citations

Stone Foundation Tablet with Inscription of Gudea - 41221 (http://commons.wikimedia.org/wiki/File:Sumerian _-_Stone_Foundation_Tablet_with_Inscription_of_Gudea_-_Walters_41221_-_View_A.jpg). Artist Unknown. Walters Art Museum [Public domain, CC-BY-SA-3.0 (http://creativecommons.org/licenses/by-sa/3.0)], via Wikimedia Commons

Stone Foundation Tablet with Inscription of Gudea - 41220 (http://commons.wikimedia.org/wiki/File:Sumerian_-_Stone_Foundation_Tablet_with_an_Inscription_of_Gudea_-_Walters_41220_-_View_A.jpg). Artist Unknown. Walters Art Museum [Public domain, CC-BY-SA-3.0 (http://creativecommons.org/licenses/by-sa/3.0) via Wikimedia Commons

Buddha at the Moment of Victory (http://commons.wikimedia.org/wiki/File:Thai_-_Buddha_at_the_Moment_of_Victory_-_Walters_542775.jpg). Artist Unknown. Walters Art Museum [Public domain, CC-BY-SA-3.0 (http://creativecommons.org/licenses/by-sa/3.0)], via Wikimedia Commons

Hoxne Hoard two gold bracelets (http://commons.wikimedia.org/wiki/File:Hoxne_Hoard_two_gold_bracelets_side.JPG) by Fæ (http://commons.wikimedia.org/wiki/User:F%C3%A6) [CC-BY-SA-3.0 (http://creativecommons.org/licenses/by-sa/3.0)], via Wikimedia Commons

Ring with the engraved portrait of Ptolemy VI Philometor (http://commons.wikimedia.org/wiki/File:Ring_with_engraved_portrait_of_ Ptolemy_VI_Philometor_(3rd%E2%80%932nd_century_BCE)_-_2009.jpg<http://commons.wikimedia.org/wiki/File:Ring_with_engraved_portrait_of_Ptolemy_VI_Philometor_(3rd-2nd_century_BCE)_-_2009.jpg>) By Unknown. (Photographed by PHGCOM in 2009.) [Public domain], via Wikimedia Commons

Calice du sacre Tau (http://commons.wikimedia.org/wiki/File:Calice_du_sacre_Tau.jpg) By Vassil (http://commons.wikimedia.org/wiki/User:Vassil) (Own work) [Public domain], via Wikimedia Commons

Citations

the mask of agamemnon (http://www.flickr.com/photos/rosemania/5705122218/) by Xuan Che (http://www.flickr.com/people/rosemania/) CC BY-2.0 (http://creativecommons.org/licenses/by/2.0/deed.en)

Terracotta Warrior (http://www.flickr.com/photos/59627558@N00/4677378806/) by fixermark (http://www.flickr.com/photos/59627558@N00/) CC BY 2.0 (http://creativecommons.org/licenses/by/2.0/deed.en)

Mmmmm Chocolate [274/366] (http://www.flickr.com/photos/sackton/8041853745) by Tim Sackton (http://www.flickr.com/photos/sackton/) CC BY-SA 2.0 (http://creativecommons.org/licenses/by-sa/2.0/deed.en)

Discrete Optimization

The Knapsack Problem: Search Strategies

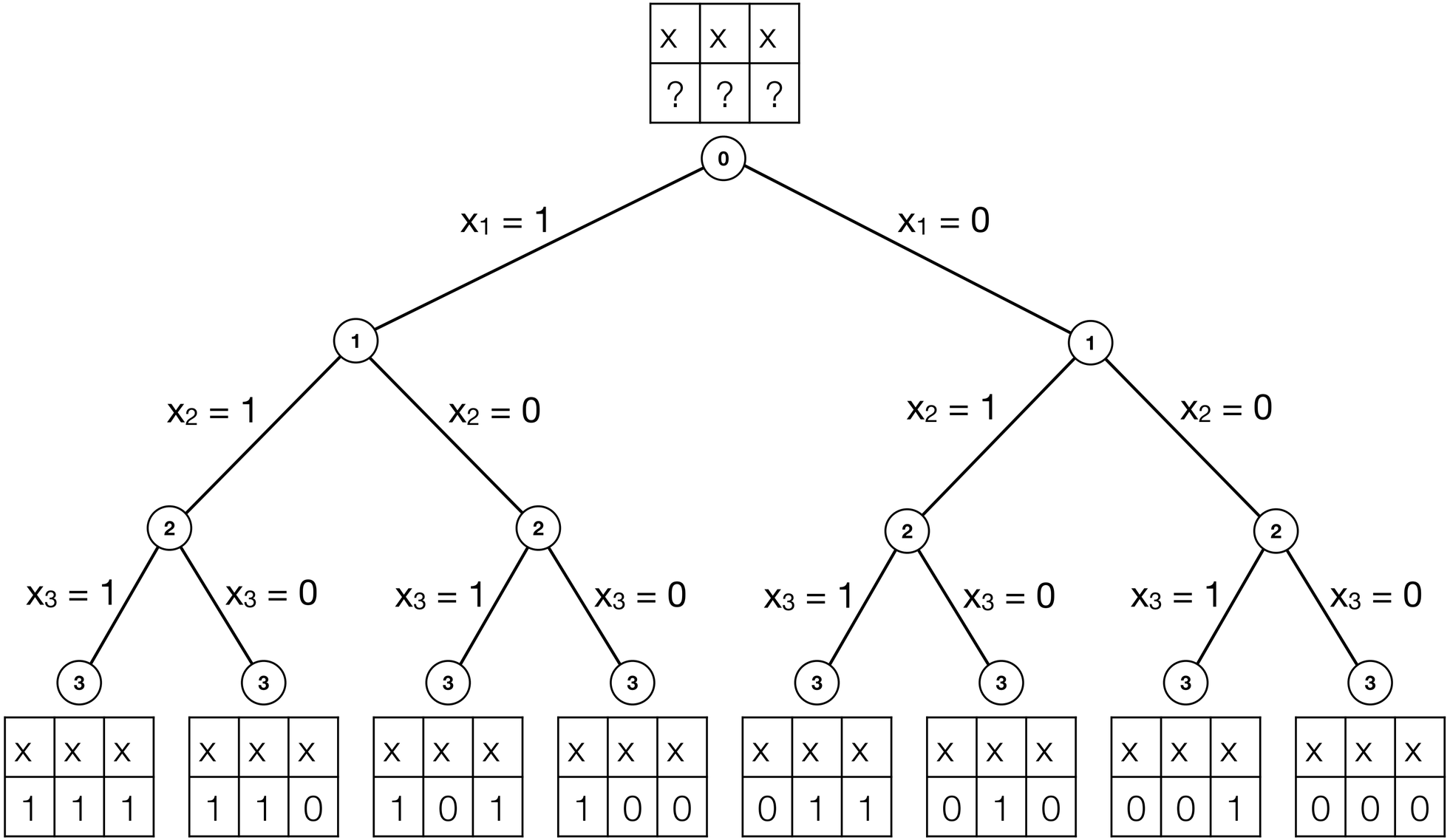
Goals of the Lecture

- Introduce search strategies for branch and bound

A Knapsack Model

- Of course, if the items are archeological artifacts...

Exhaustive Search



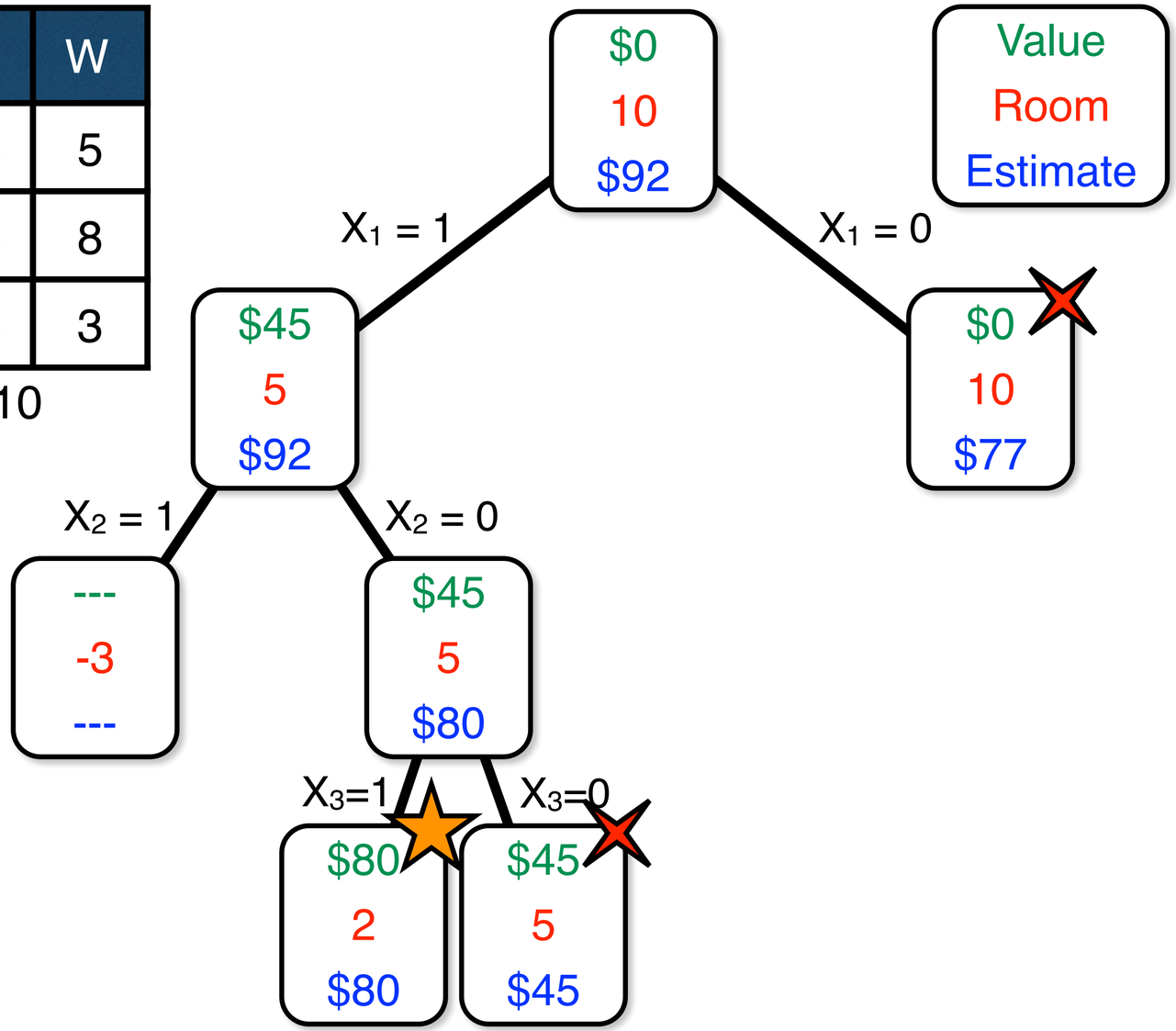
Branch and Bound

- Search Strategies
  - depth-first, best-first, least-discrepancy
  - many others
- Depth-first
  - prunes when a node estimation is worse than the best found solution
  - memory efficient
- Best-First
  - select the node with the best estimation
- Least-Discrepancy
  - trust a greedy heuristic

Depth-First Branch and Bound

i	V	W
1	45	5
2	48	8
3	35	3

K = 10



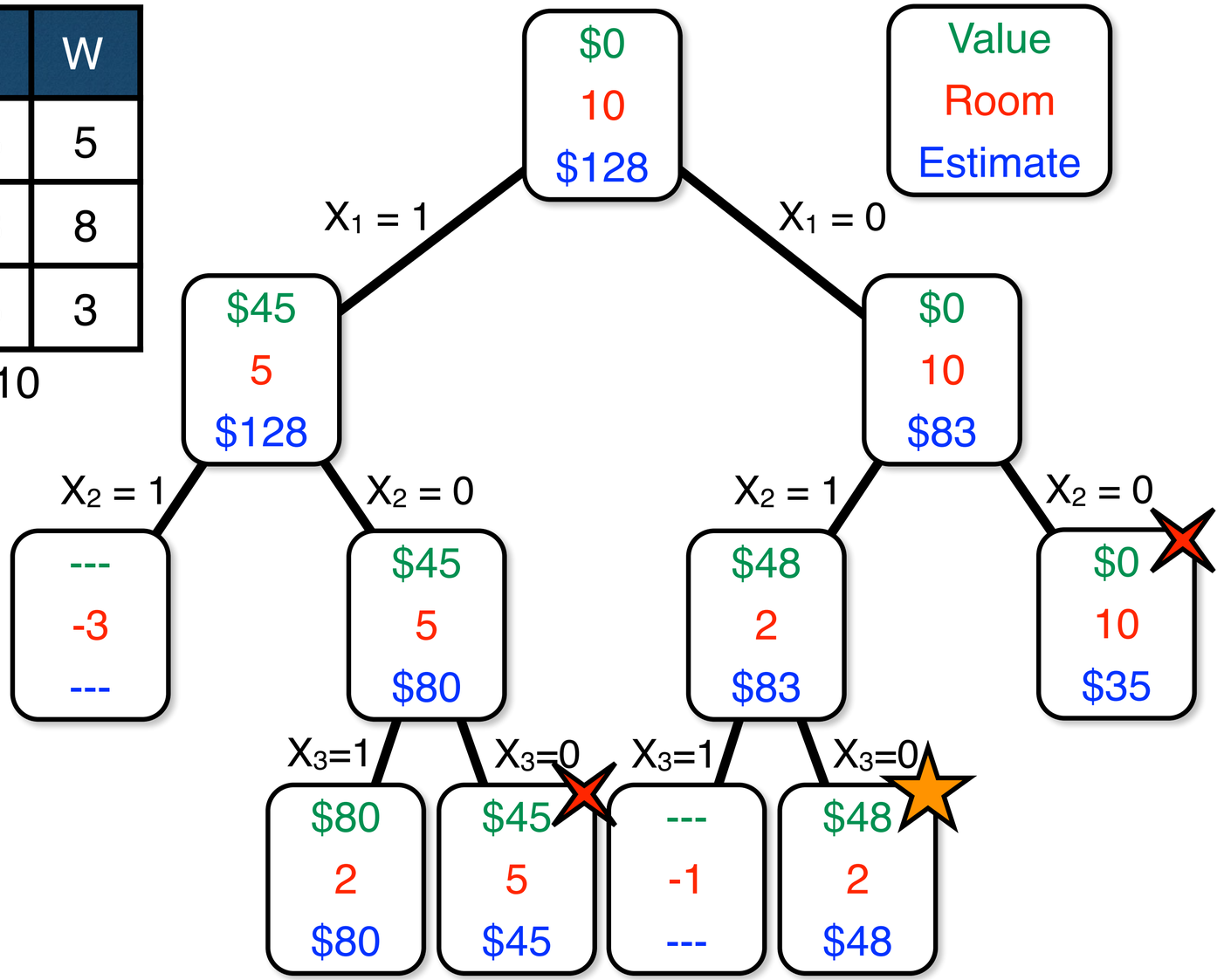
Branch and Bound

- Depth-First
  - select the node to the left
  - when does it prune?
    - when it finds a new node worse than the found solution
  - is it memory efficient?
    - exaggerate!

Best-First Branch and Bound

i	V	W
1	45	5
2	48	8
3	35	3

K = 10



Branch and Bound

- Best-First
  - select the node with the best estimation
  - when does it prune?
    - when all the nodes are worse than a found solution
  - is it memory efficient?
    - exaggerate!

Limited Discrepancy Search

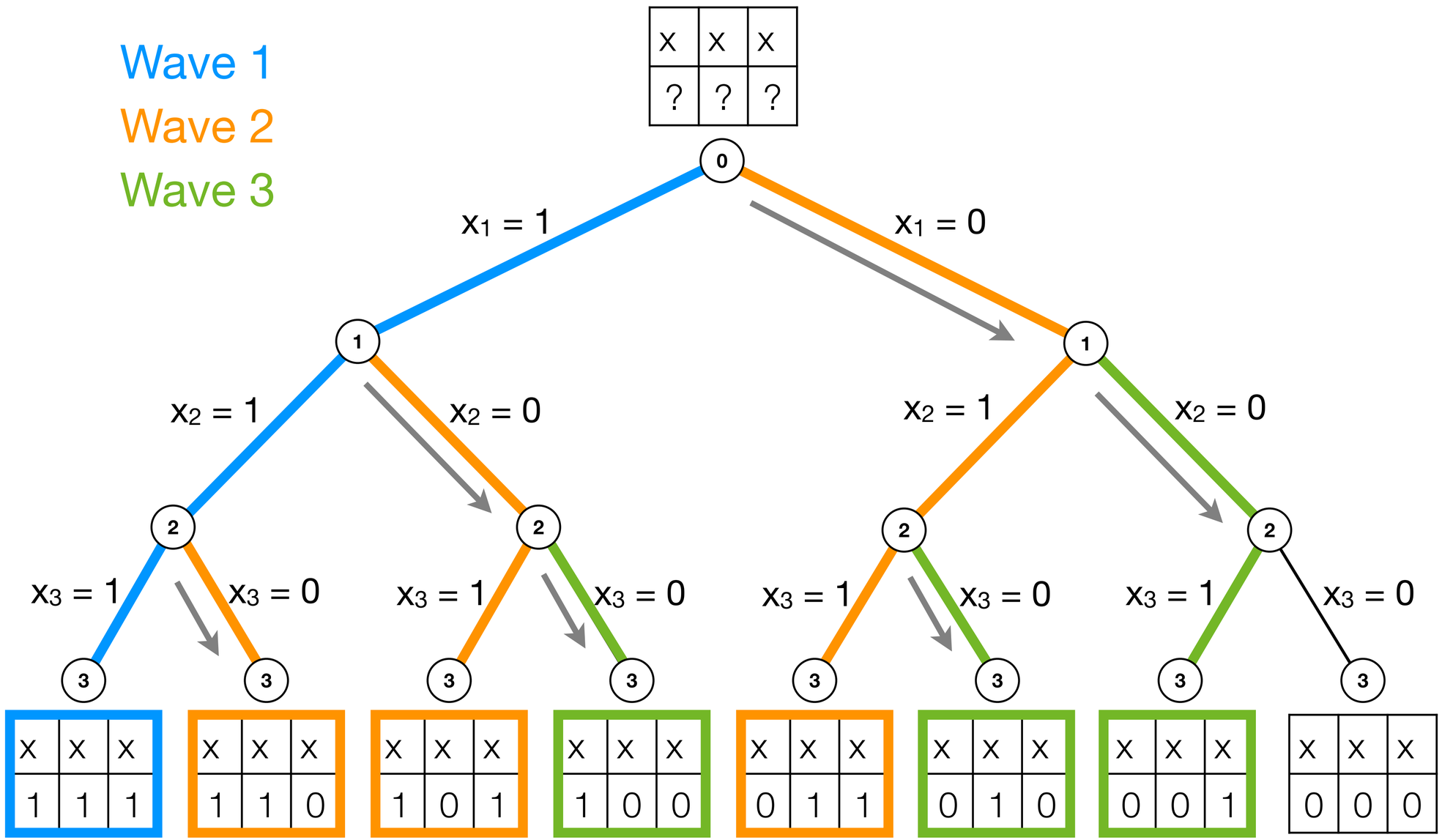
- Assume that a good heuristic is available
  - it makes very few mistakes
  - the search tree is binary
  - following the heuristic means branching **left**
  - branching **right** means the heuristic was wrong

- Limited Discrepancy Search (LDS)
  - explore the search space in increasing order of mistakes
  - trusting the heuristic less and less

Limited Discrepancy Search

- Limited Discrepancy Search (LDS)
  - explore the search space in increasing order of mistakes
  - trusting the heuristic less and less
- Explores the search space in waves
  - no mistake
  - one mistake
  - two mistakes
  - ....

LDS Waves

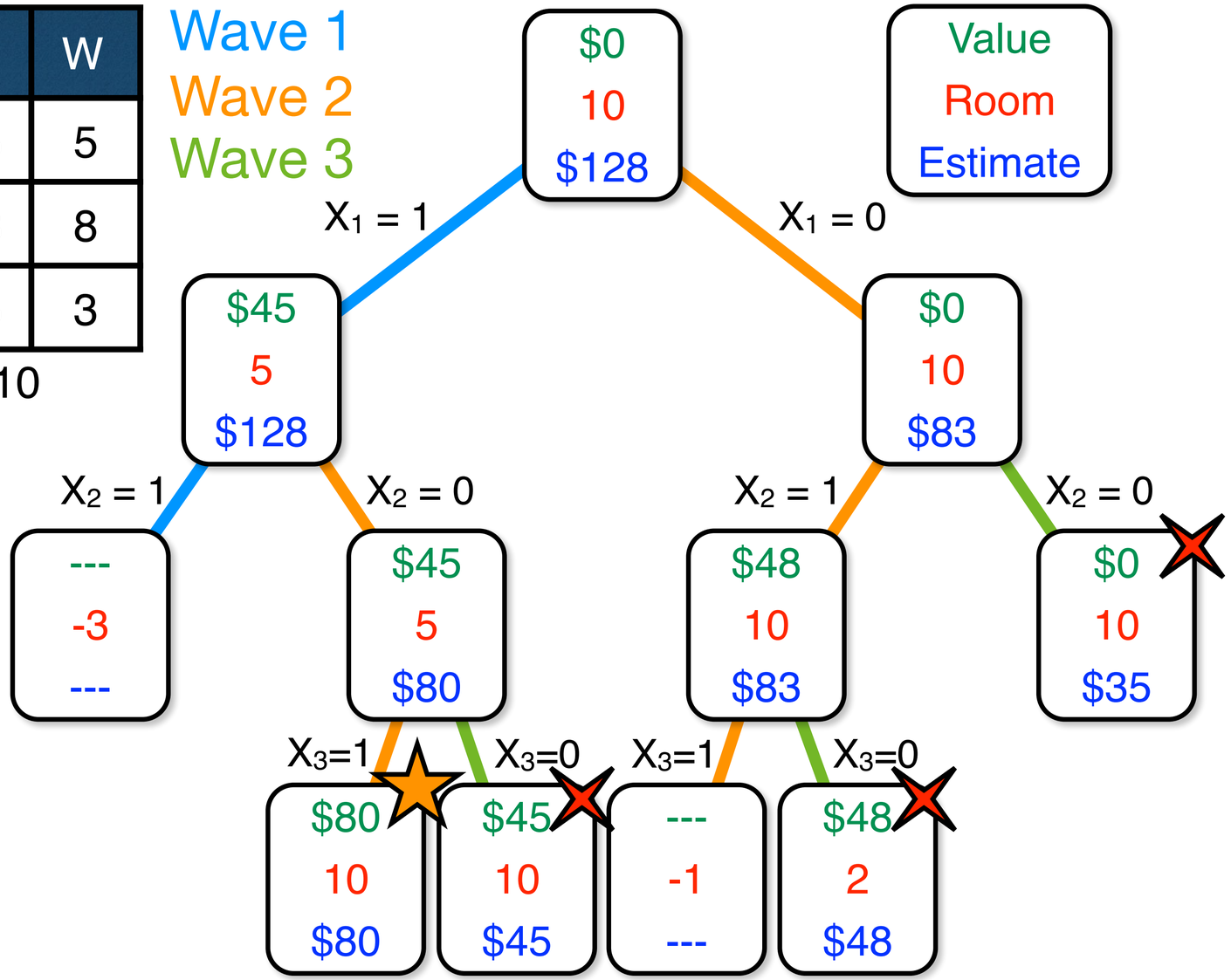

Wave 1
Wave 2
Wave 3
x x x
? ? ?
0
$x_1 = 1$
$x_1 = 0$
1
1
$x_2 = 1$
$x_2 = 0$
$x_2 = 1$
$x_2 = 0$
2
2
2
2
$x_3 = 1$
$x_3 = 0$
$x_3 = 1$
$x_3 = 0$
$x_3 = 1$
$x_3 = 0$
$x_3 = 1$
$x_3 = 0$
3
3
3
3
3
3
3
3
x x x
1 1 1
x x x
1 1 0
x x x
1 0 1
x x x
1 0 0
x x x
0 1 1
x x x
0 1 0
x x x
0 0 1
x x x
0 0 0

LDS Branch and Bound

i	V	W
1	45	5
2	48	8
3	35	3

K = 10

Wave 1
Wave 2
Wave 3



Branch and Bound

- Least Discrepancy Search
  - trust the greedy heuristic
  - when does it prune?
  - is it memory efficient?
    - compared to depth-first and best-first?

Search Strategies

- Relaxation and Search
  - a match made in discrete optimization heaven
- Can be problem specific
- Can you think of a new one?

Until Next Time

Citations

Stone Foundation Tablet with Inscription of Gudea - 41221 (http://commons.wikimedia.org/wiki/File:Sumerian _-_Stone_Foundation_Tablet_with_Inscription_of_Gudea_-_Walters_41221_-_View_A.jpg). Artist Unknown. Walters Art Museum [Public domain, CC-BY-SA-3.0 (http://creativecommons.org/licenses/by-sa/3.0)], via Wikimedia Commons

Stone Foundation Tablet with Inscription of Gudea - 41220 (http://commons.wikimedia.org/wiki/File:Sumerian_-_Stone_Foundation_Tablet_with_an_Inscription_of_Gudea_-_Walters_41220_-_View_A.jpg). Artist Unknown. Walters Art Museum [Public domain, CC-BY-SA-3.0 (http://creativecommons.org/licenses/by-sa/3.0) via Wikimedia Commons

Buddha at the Moment of Victory (http://commons.wikimedia.org/wiki/File:Thai_-_Buddha_at_the_Moment_of_Victory_-_Walters_542775.jpg). Artist Unknown. Walters Art Museum [Public domain, CC-BY-SA-3.0 (http://creativecommons.org/licenses/by-sa/3.0)], via Wikimedia Commons

Hoxne Hoard two gold bracelets (http://commons.wikimedia.org/wiki/File:Hoxne_Hoard_two_gold_bracelets_side.JPG) by Fæ (http://commons.wikimedia.org/wiki/User:F%C3%A6) [CC-BY-SA-3.0 (http://creativecommons.org/licenses/by-sa/3.0)], via Wikimedia Commons

Ring with the engraved portrait of Ptolemy VI Philometor (http://commons.wikimedia.org/wiki/File:Ring_with_engraved_portrait_of_ Ptolemy_VI_Philometor_(3rd%E2%80%932nd_century_BCE)_-_2009.jpg<http://commons.wikimedia.org/wiki/File:Ring_with_engraved_portrait_of_Ptolemy_VI_Philometor_(3rd-2nd_century_BCE)_-_2009.jpg>) By Unknown. (Photographed by PHGCOM in 2009.) [Public domain], via Wikimedia Commons

Calice du sacre Tau (http://commons.wikimedia.org/wiki/File:Calice_du_sacre_Tau.jpg) By Vassil (http://commons.wikimedia.org/wiki/User:Vassil) (Own work) [Public domain], via Wikimedia Commons

Citations

the mask of agamemnon (http://www.flickr.com/photos/rosemania/5705122218/) by Xuan Che (http://www.flickr.com/people/rosemania/) CC BY-2.0 (http://creativecommons.org/licenses/by/2.0/deed.en)

Terracotta Warrior (http://www.flickr.com/photos/59627558@N00/4677378806/) by fixermark (http://www.flickr.com/photos/59627558@N00/) CC BY 2.0 (http://creativecommons.org/licenses/by/2.0/deed.en)

Discrete Optimization

Assignments: Getting Started

Goals of the Lecture

- How to get started on a new problem
- Different approaches are rewarded equally

Assignment Design

- Emulation of the Real World
  - your boss tells you, *“we need to solve this problem”*
  - doesn’t care how you do it, just wants results!

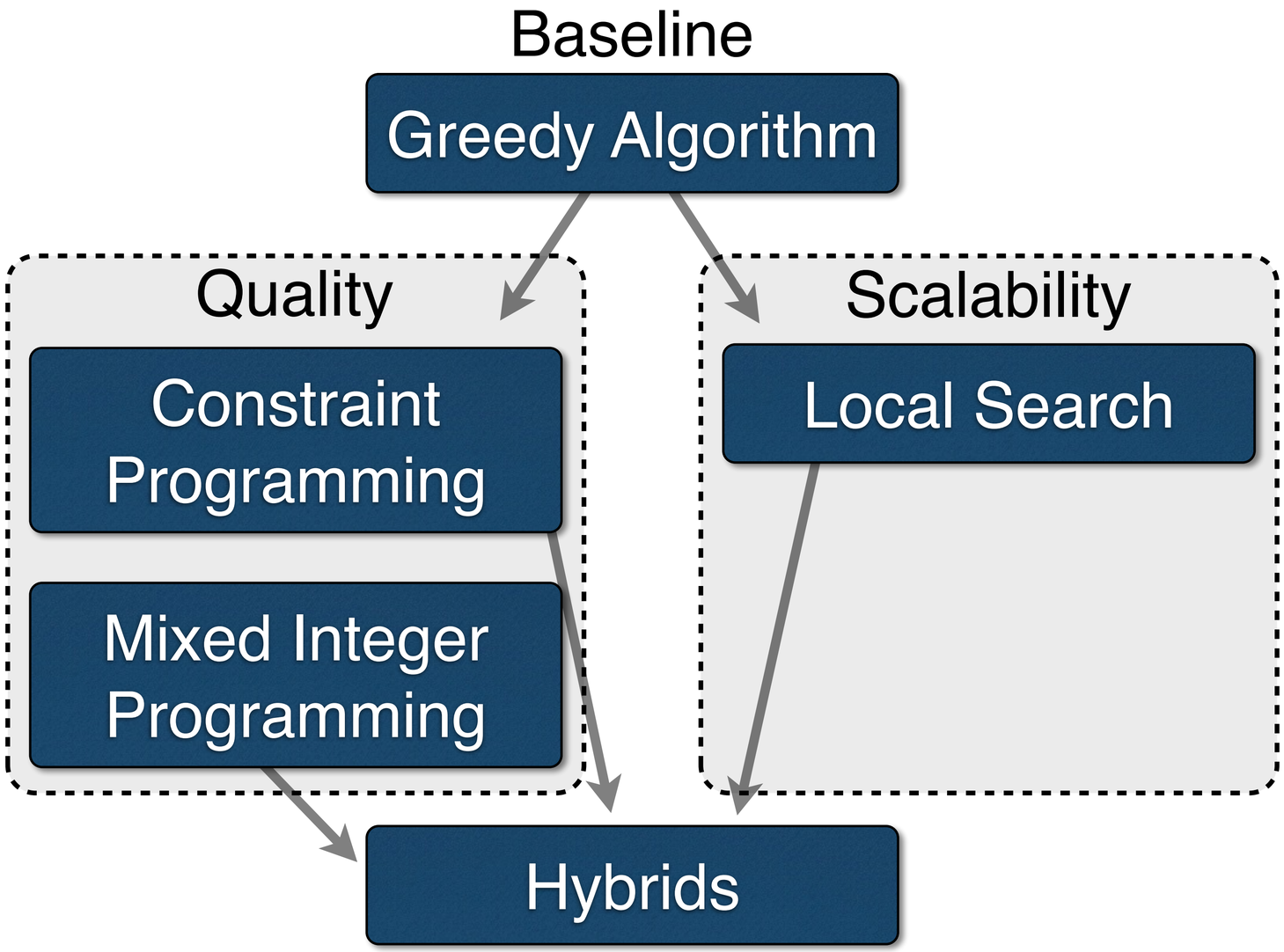
- How can you solve the problem?
  - Use your optimization **hats**!

- The assignments are about being creative with your hats and having hunches and trying them out

Getting Started

- Imagine yourself in a company
- Implement the simplest thing you can imagine
  - for example, a greedy solution
  - see how well it does, inspect the solution
- Try to understand its shortcomings, and fix them
  - smarter greedy
  - hats you have learned: DP, CP, LS, MIP
- Relax
  - Give quality guarantee

Getting Better



Grading Design

- There is no silver bullet in optimization
  - Just one approach will not work for all the parts of an assignment
  - We reward taking a scalability approach and taking a quality approach.

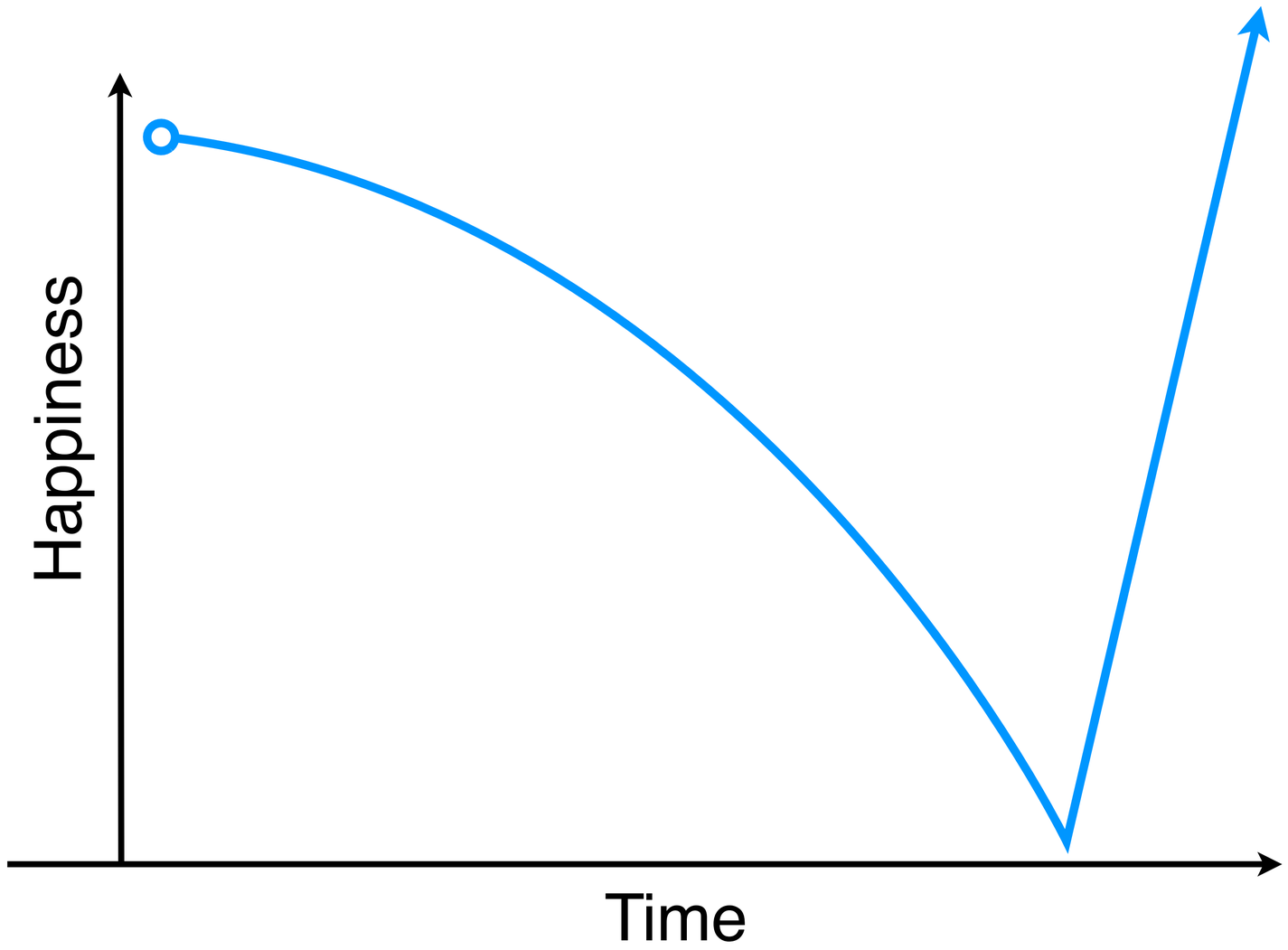
- A little math related to grading, target points for an assignment 7*6 = 42
  - High-quality, less scalable 10*4 + 3*2 = 46
  - Scalable, lower quality solution 7*6 = 42
  - Both are viable approaches!

Moving On

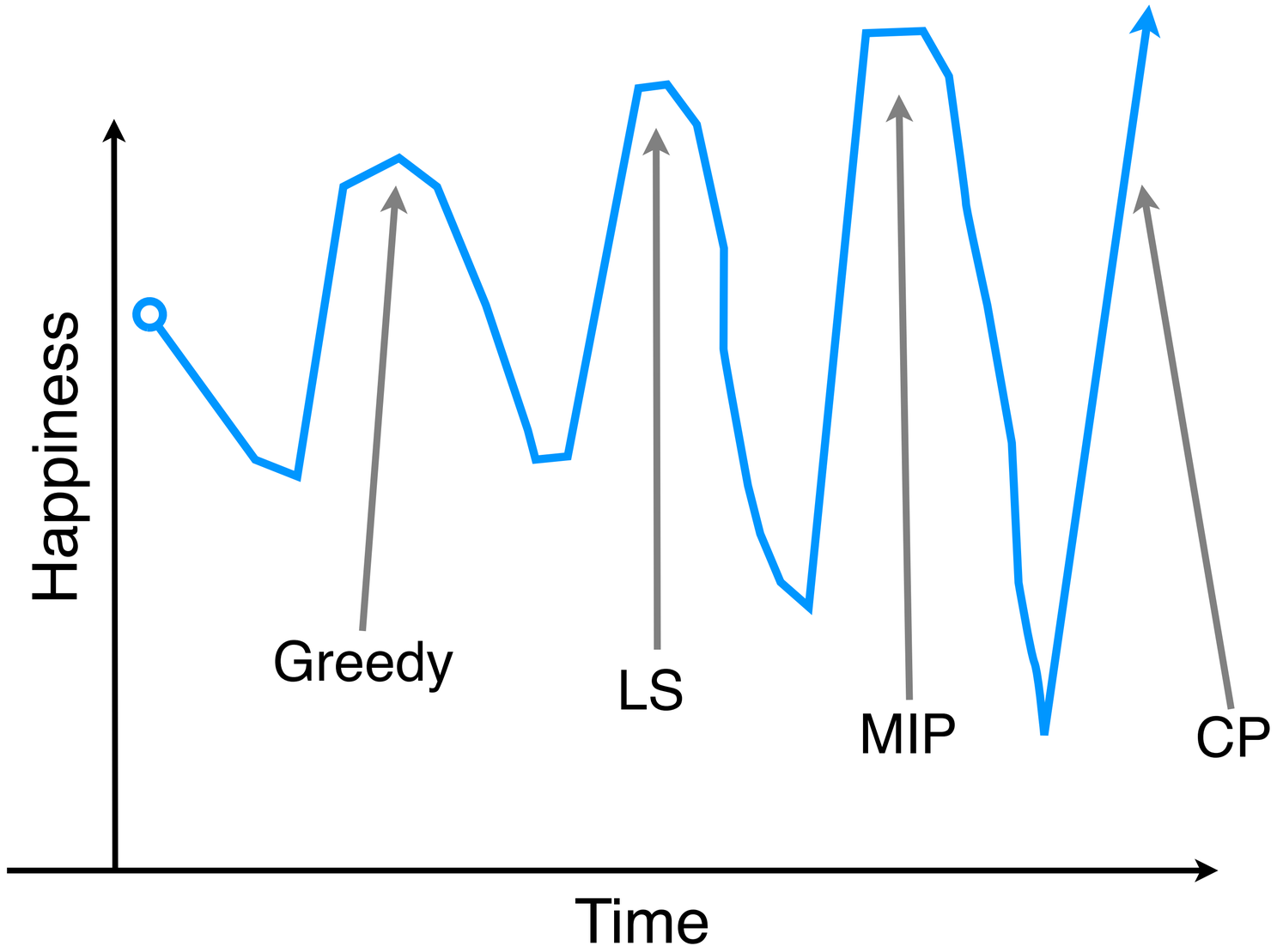
- Optimize your time
  - As the course continues you will, get smarter, skills will improve, and have better ideas
  - Problems will seem easer
  - If you get stuck, move on, and come back to that problem later

- There is time at the end of the course, just for “touching up” your solutions

Enjoying the Journey



Enjoying the Journey



Have Fun!

Discrete Optimization

Assignments: Knapsack

The Knapsack Problem

The Knapsack Problem

- *n* Items
- Value V_i
- Weight W_i
- X_i take item *i*

$$\text{maximize:} \quad \sum_{i \in 1 \ldots n} v_i x_i$$

$$\text{subject to:} \quad \sum_{i \in 1 \ldots n} w_i x_i \leq K$$

$$x_i \in \{0, 1\} \quad (i \in 1 \ldots n)$$

I/O Format

$$\text{maximize:} \quad \sum_{i \in 1 \ldots n} v_i x_i$$

$$\text{subject to:} \quad \sum_{i \in 1 \ldots n} w_i x_i \leq K$$

$$x_i \in \{0, 1\} \quad (i \in 1 \ldots n)$$

Input

```
n K
v_1 w_1
v_2 w_2
...
v_n w_n
```

Output

```
obj opt
x_1 x_2 x_3 ... x_n
```

An Example

$$\text{maximize:} \quad \sum_{i \in 1 \ldots n} v_i x_i$$

$$\text{subject to:}$$

$$\sum_{i \in 1 \ldots n} w_i x_i \leq K$$

$$x_i \in \{0, 1\} \quad (i \in 1 \ldots n)$$

Input

```
4 11
8 4
10 5
15 8
4 3
```

Output

```
19 0
0 0 1 1
```

Implementing Your solver.py

```
import os

def solveIt(inputData):
    ### Modify this code to run your optimization algorithm
    lines = inputData.split('\n')

    ...

    outputData = str(value)+' '+str(0)+'\n'
    outputData += ' '.join(map(str,taken))
    return outputData
```

Input

```
4 11
8 4
10 5
15 8
4 3
```

Output

```
19 0
0 0 1 1
```

Implementing an External Solver

solverJava.py

```
def solveIt(inputData):
    tmpFileName = 'tmp.data'
    tmpFile = open(tmpFileName,'w')
    tmpFile.write(inputData)
    ...
    ### Runs the command: java Solver -file=tmp.data
    process = Popen(['java','Solver','-file='+tmpFileName], stdout=PIPE)
    (stdout, stderr) = process.communicate()
    ...
    return stdout.strip()
```

Solver.java

```
class Solver {
    public static void main(String[] args) {
        new Solver().run(args);
    }
    public void run(String[] args) {
        ...
        System.out.println(value+" 0");
        for(int i=0; i < items; i++){
            System.out.print(taken[i]+" ");
        }
        System.out.println("");
    }
}
```

Testing Your Solver

```
> python solver.py ./data/ks_4_0
18 0
1 1 0 0

>
```

Grading Your Solver

```
> python submit.pyc
==
== Knapsack Solution Submission
==
Login (Email address):
Submission Password (from the programming assignments page.
This is NOT your own account's password):

== Connecting to Coursera ...
Hello! These are the assignment parts that you can submit:
1) Knapsack Problem 1
2) Knapsack Problem 2
3) Knapsack Problem 3
4) Knapsack Problem 4
5) Knapsack Problem 5
6) Knapsack Problem 6
0) All
Please enter which part(s) you want to submit (0-6):
```

Grading Your Solver

```
== Connecting to Coursera ...
Hello! These are the assignment parts that you can submit:
1) Knapsack Problem 1
2) Knapsack Problem 2
3) Knapsack Problem 3
4) Knapsack Problem 4
5) Knapsack Problem 5
6) Knapsack Problem 6
0) All
Please enter which part(s) you want to submit (0-6):
```

```
Please enter which part(s) you want to submit (0-6): 3
```

```
Please enter which part(s) you want to submit (0-6): 0
```

```
Please enter which part(s) you want to submit (0-6): 4, 6
```

Assignment Tips

- Dynamic Program
  - How much space do you need?

- Branch and Bound
  - What is the fastest way to calculate the relaxation value?

Have Fun!

Citations

Stone Foundation Tablet with Inscription of Gudea - 41221 (http://commons.wikimedia.org/wiki/File:Sumerian _-_Stone_Foundation_Tablet_with_Inscription_of_Gudea_-_Walters_41221_-_View_A.jpg). Artist Unknown. Walters Art Museum [Public domain, CC-BY-SA-3.0 (http://creativecommons.org/licenses/by-sa/3.0)], via Wikimedia Commons

Stone Foundation Tablet with Inscription of Gudea - 41220 (http://commons.wikimedia.org/wiki/File:Sumerian_-_Stone_Foundation_Tablet_with_an_Inscription_of_Gudea_-_Walters_41220_-_View_A.jpg). Artist Unknown. Walters Art Museum [Public domain, CC-BY-SA-3.0 (http://creativecommons.org/licenses/by-sa/3.0) via Wikimedia Commons

Buddha at the Moment of Victory (http://commons.wikimedia.org/wiki/File:Thai_-_Buddha_at_the_Moment_of_Victory_-_Walters_542775.jpg). Artist Unknown. Walters Art Museum [Public domain, CC-BY-SA-3.0 (http://creativecommons.org/licenses/by-sa/3.0)], via Wikimedia Commons

Hoxne Hoard two gold bracelets (http://commons.wikimedia.org/wiki/File:Hoxne_Hoard_two_gold_bracelets_side.JPG) by Fæ (http://commons.wikimedia.org/wiki/User:F%C3%A6) [CC-BY-SA-3.0 (http://creativecommons.org/licenses/by-sa/3.0)], via Wikimedia Commons

Ring with the engraved portrait of Ptolemy VI Philometor (http://commons.wikimedia.org/wiki/File:Ring_with_engraved_portrait_of_ Ptolemy_VI_Philometor_(3rd%E2%80%932nd_century_BCE)_-_2009.jpg<http://commons.wikimedia.org/wiki/File:Ring_with_engraved_portrait_of_Ptolemy_VI_Philometor_(3rd-2nd_century_BCE)_-_2009.jpg>) By Unknown. (Photographed by PHGCOM in 2009.) [Public domain], via Wikimedia Commons

Calice du sacre Tau (http://commons.wikimedia.org/wiki/File:Calice_du_sacre_Tau.jpg) By Vassil (http://commons.wikimedia.org/wiki/User:Vassil) (Own work) [Public domain], via Wikimedia Commons

Citations

the mask of agamemnon (http://www.flickr.com/photos/rosemania/5705122218/) by Xuan Che (http://www.flickr.com/people/rosemania/) CC BY-2.0 (http://creativecommons.org/licenses/by/2.0/deed.en)

Terracotta Warrior (http://www.flickr.com/photos/59627558@N00/4677378806/) by fixermark (http://www.flickr.com/photos/59627558@N00/) CC BY 2.0 (http://creativecommons.org/licenses/by/2.0/deed.en)

Discrete Optimization

Exploring the Material

Goals of the Lecture

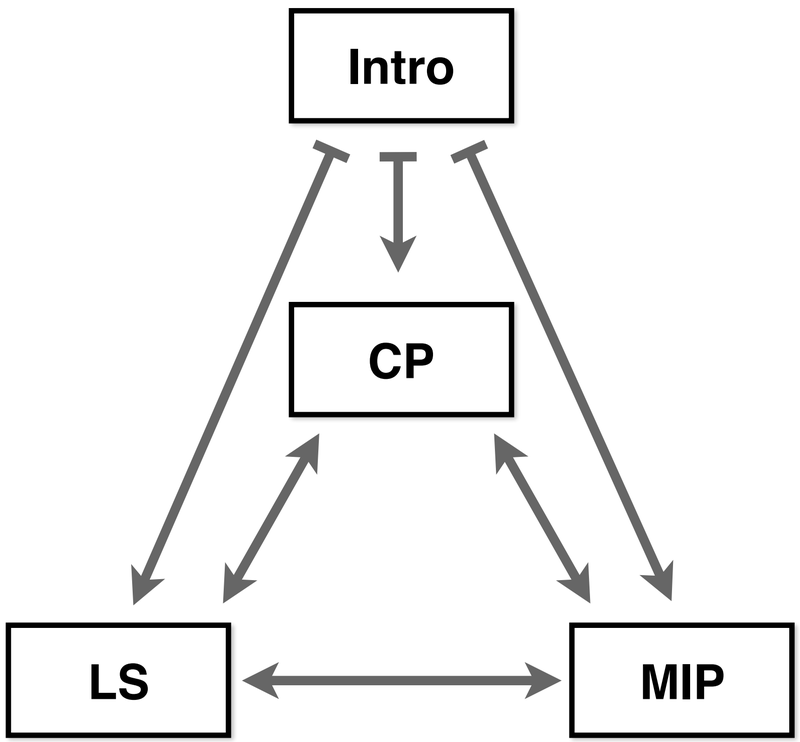
- Exploring the rest of the course material
- Designing your own study plan

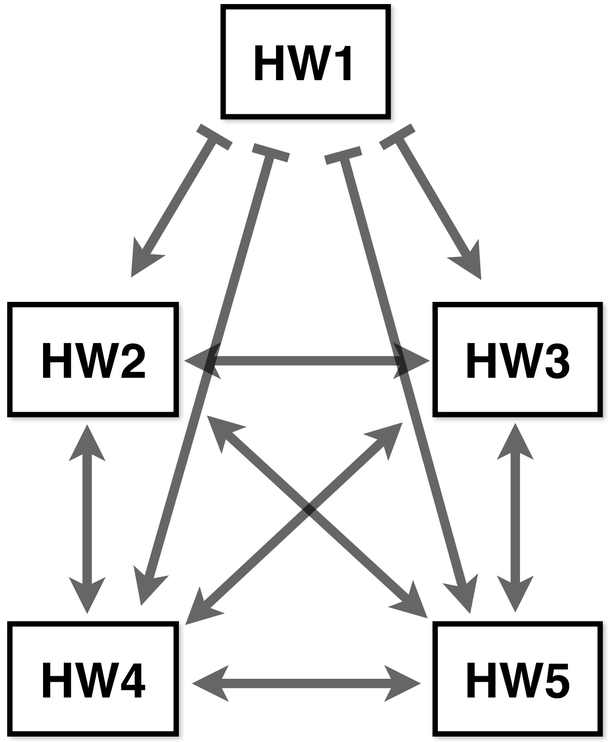
Congratulations!

- You have made it this far!

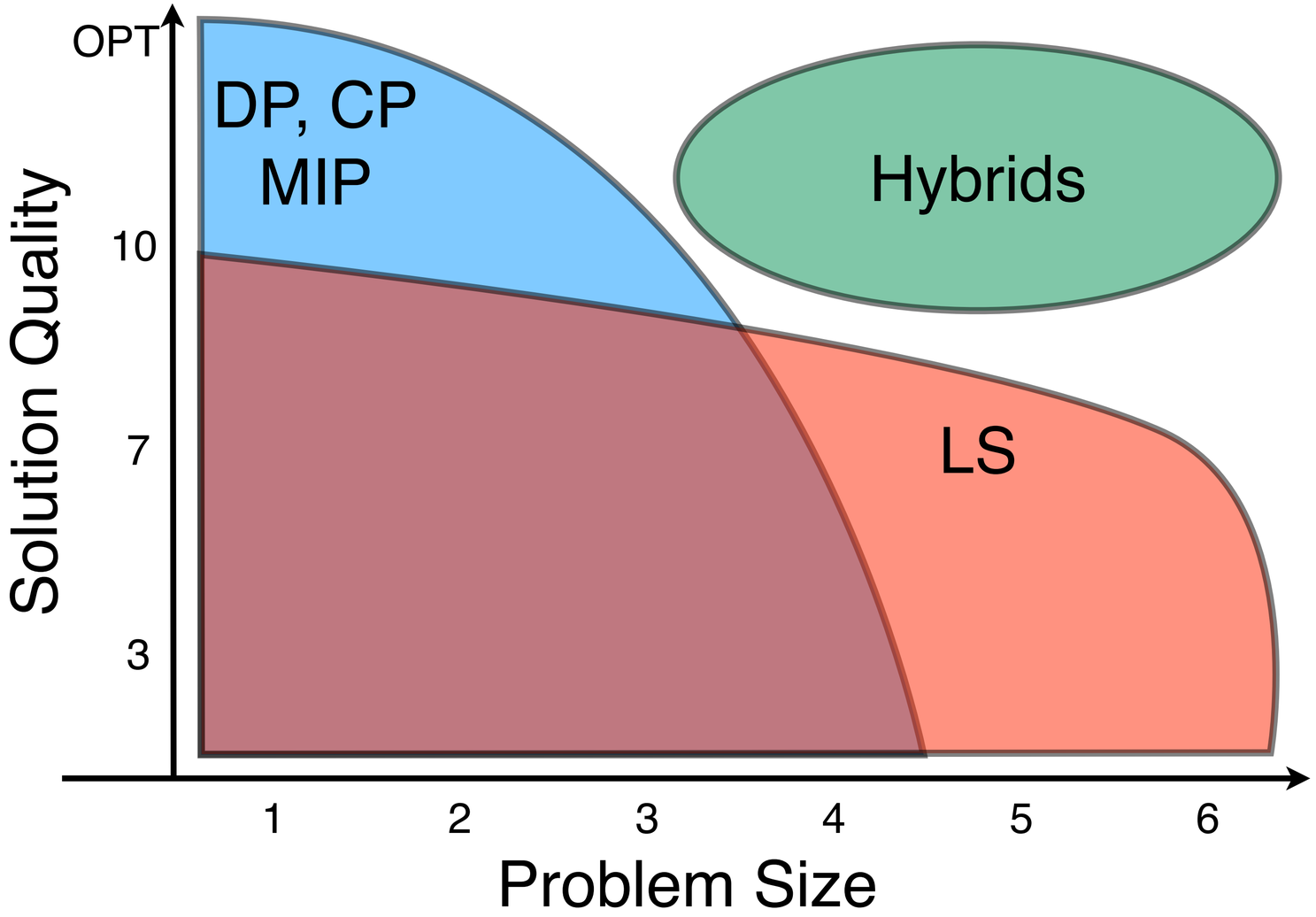
- Passed the Knapsack assignment and getting ready to explore the other course topics
  - Constraint Programming (CP)
  - Local Search (LS)
  - Mixed Integer Programming (MIP)

Open Course Design





Optimization Landscape



Pick Your Own Optimization Adventure

- There are many viable paths through the course
- Most problem sets have 6 parts graded on a 10 point scale
  - Quality based solution approach (CP,DP,MIP)
    - 10*4 + 3*2 = 46 points
  - Scalability based solution approach (LS)
    - 7*6 = 42 points
  - Either is sufficient to get a certificate
- It may take both to get a distinction certificate

Matching Your Interest to the Topic

- Constraint Programming
  - like solving puzzles
  - lots of logic / discrete mathematics
- Mixed Integer Programming
  - grounded in linear algebra
  - lots of continuous mathematics
- Local Search
  - intuition based, most significant coding
  - writing efficient code really helps
  - lots of staring at the terminal

Until Next Time

- We hope you enjoy your adventure in to Discrete Optimization