UNIVERSITÀ DEGLI STUDI
DI MILANO

# Goal Oriented Behavior

*A.I. for Video Games*

Davie Gadia (gadia@di.unimi.it) and Dario Maggiorini (dario@di.unimi.it)

# Goal Orientation

- Decision techniques we used so far are reactive
  - A set of external inputs is used to take a decision

- What if we want to set an overall goal for our agent and then let it do whatever it takes to reach that goal?
  - Kill a boss
  - Get extremely rich
  - Survive as long as possible

# GOB as a General Term

- Goal-Oriented is some kind of general term used to indicate any technique taking into account goals or desires

- Many techniques fall under this label
  - E.g., utility-based decision-making

- Using these techniques, we choose an action to execute from a set
  - The selection is based on current goals

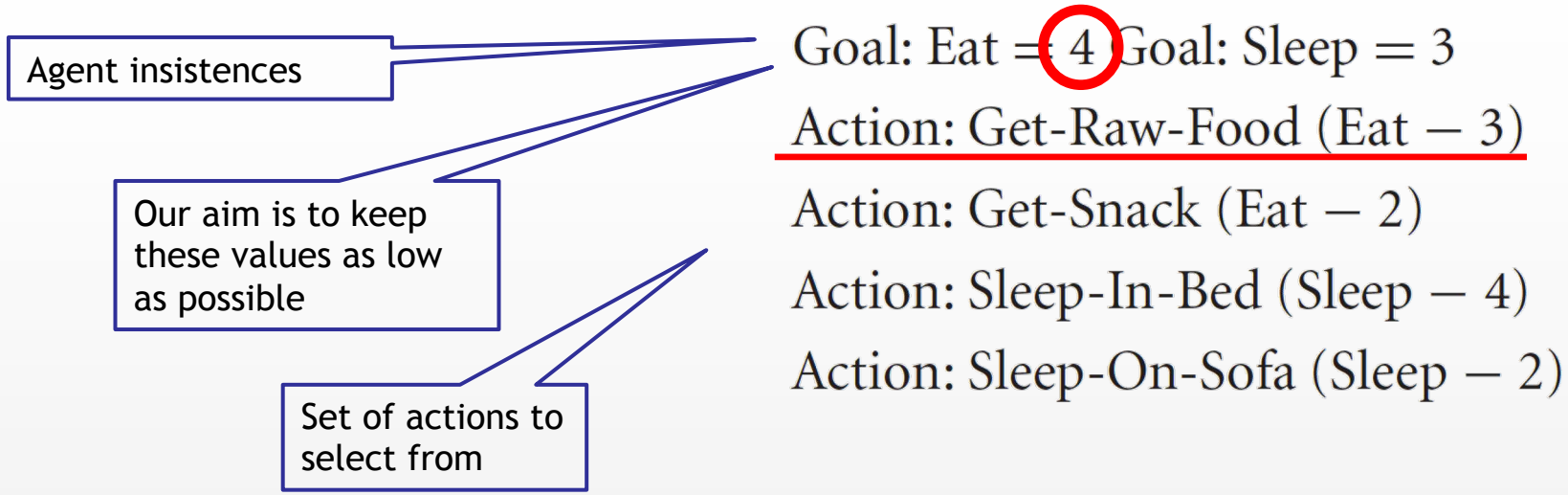
# Elements of GOB

- Goal (*motive*)
  - Describes what we want to do
  - An agent may have multiple goals active at the same time
- Insistence
  - A numeric value associated to a goal indicating the compulsion to fulfill it
  - A value of zero means the goal has been satisfied (has been reached)
    - On the contrary, the maximum value is left to the designer
  - Insistence may just drop temporarily and never get fully satisfied
    - E.g., hunger. You eat to survive, but eventually get hungry again
- Action
  - Something that the agent can do to pursue its goal(s)
  - Every action is rated against each applicable goal

For every action, we know the effect on each goal

# Simple Selection Approach

- Pick an action in a way that makes the agent look intelligent




- In a simple selection approach, we identify:
  1. The most insistent goal
  2. The most rewarding action for that goal

Randomize if you have more than one valid option

# But No One is Perfect

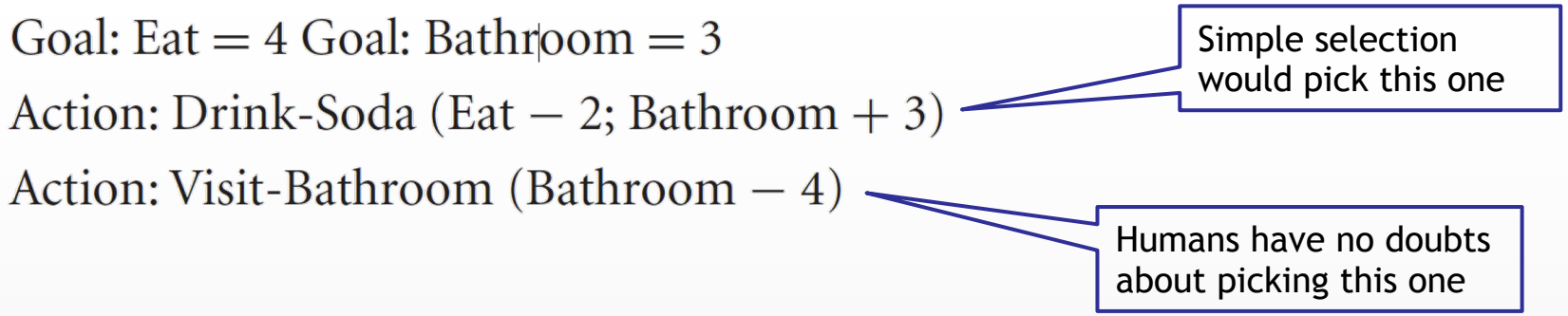
- Goal: Eat = 5 Goal: Sleep = 4
- Action: Get-Raw-Food (Eat – 3)
- Action: Get-Snack (Eat – 2)
- Action: Sleep-In-Bed (Sleep – 4)
- Action: Sleep-On-Sofa (Sleep – 2)

Good solution, picked by simple selection

Could this be an overall better solution for the agent?

# Overall Utility Approach

- Pick the action giving the best result over all the goals




- We define the discontentment as the combination of insistences over all goals

- In the overall utility approach we select the action minimizing discontentment

# Evaluating Discontentment

- The sum of the insistences may not be a good idea

In this example we assume all max values are the same. If not, we can just normalize before scaling

Discontentment = 14 (10 + 4)

Goal: Eat = 10 Goal: Bathroom = 4
Action: Drink-Soda (Eat -4; Bathroom +1)
Afterward: Eat = 6; Bathroom = 5

Discontentment = 11

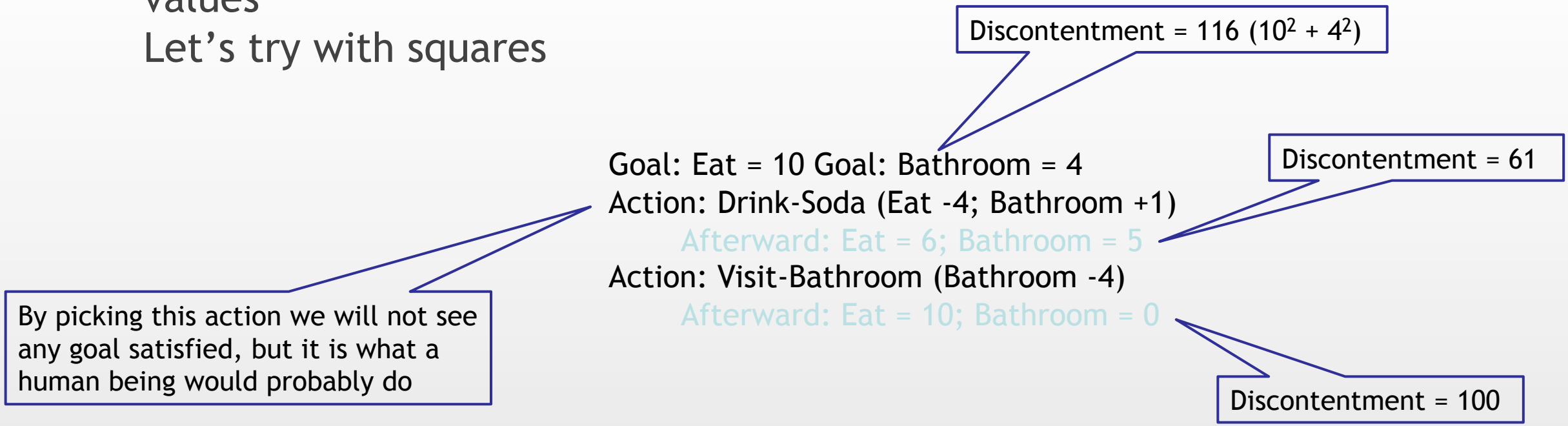
Action: Visit-Bathroom (Bathroom -4)
Afterward: Eat = 10; Bathroom = 0

Discontentment = 10

Nevertheless, a real person would never pick this action because the compulsion to eat is much stronger than going to the bathroom

# Evaluating Discontentment

- The sum of the insistences may not be a good idea
- But we want to give priority to high insistence values
  - A simple solution is to apply a nonlinear scaling and amplify higher values
    Let's try with squares



# Taking Time Into Account

- Every action may take time to complete
  - This accounts also for:
    1. Multi-steps actions
    2. Eventual movement or commuting time required

- Timing may be an accurate evaluation or just a guesstimation. This is not a problem as long as it is:
  1. Consistent
  2. A distance

I will assume you know the properties of a distance function

# Discontentment and Timing

- Simple way
  - We evaluate discontentment as for an instant action, and give priority to shorter actions when discontentment is the same

- Best way
  - We evaluate discontentment taking into consideration the effects of the extra time

Goal: Eat = 4 changing at + 4 per hour
Goal: Bathroom = 3 changing at + 2 per hour
Action: Eat-Snack (Eat − 2) 15 minutes
■ afterwards: Eat = 2, Bathroom = 3.5: Discontentment = 16.25
Action: Eat-Main-Meal (Eat − 4) 1 hour
■ afterwards: Eat = 0, Bathroom = 5: Discontentment = 25
Action: Visit-Bathroom (Bathroom − 4) 15 minutes
■ afterwards: Eat = 5, Bathroom = 0: Discontentment = 25

Calculated using squares

This is easy. Then ... Why should we consider the simple way at all?

# Discontentment and Timing

- Simple way
  - We evaluate discontentment as for an instant action, and give priority to shorter actions when discontentment is the same

- Best way
  - We evaluate discontentment taking into consideration the effects of the extra time

Goal: Eat = 4 changing at + 4 per hour
Goal: Bathroom = 3 changing at + 2 per hour
Action: Eat-Snack (Eat − 2) 15 minutes
■ afterwards: Eat = 2, Bathroom = 3.5: Discontentment = 16.25
Action: Eat-Main-Meal (Eat − 4) 1 hour
■ afterwards: Eat = 0, Bathroom = 5: Discontentment = 25
Action: Visit-Bathroom (Bathroom − 4) 15 minutes
■ afterwards: Eat = 5, Bathroom = 0: Discontentment = 25

PROBLEM: sometimes these values are not part of the agent internal knowledge

SOLUTION: we use the simple way to select the action and then observe the system to estimate the change rate.

This is going to simulate some kind of "growing experience" in the agent!

# Estimating Changes

- The most convenient way to estimate the rate change is by observing instantaneous change rates and using an Exponential Weighted Moving Average
  - Big name for an easy formula

```
rateSinceLastTime = changeSinceLastTime / timeSinceLast
basicRate = 0.95 * basicRate + 0.05 * rateSinceLastTime
```

- If the change rate is constant, the agent will adjust the estimation after each action involving that goal
  - In the same way we gain experience from repeating an action

# Planning and GOB

- Planning actions is important when dealing with GOB
- We want to find the sequence giving us the best result on the long run
  - I.e., bringing the agent closer to its goals
- We talk about Goal Oriented Action Planning, or GOAP
- GOAP is tricky because, while planning, actions may be
  1. Inter-dependent (enabling and disabling each other)
     - I can shoot the enemy only if I picked up the gun
  2. Sharing resources
     - Driving a car and driving a truck will consume the same fuel but produce different results
  3. Constrained by time
     - I must arrive to destination by sunset
     - I cannot leave my shelter at nighttime

# Overall Utility GOAP

- Basically, we consider all possible combination of *K* steps and pick the one with minimum final discontentment

- We can implement this as a deep-first tree visit on the states tree
  - The states tree is the combination of all possible states the agent can assume by applying the rule
    - If a state is a node, having 6 actions available implies 6 branches leaving the node
    - Technically, the state tree is infinite
      - This is why we usually limit the exploration to K steps
  - Explore the tree using an array
    - Never use a recursive approach

- Easy, but damn inefficient and unacceptably time consuming
  - Complexity will be $O(nm^K)$
    - Where *n* is the number of goals, *m* is the number of actions, an K the number of steps to plan

# Overall Utility GOAP

- Basically, we consider all possible combination of *K* steps and pick the one with minimum final discontentment

- We can implement this as a deep-first tree visit on the states tree
  - The states tree is the combination of all possible states the agent can assume by applying the rule
    - If a state is a node, having 6 actions available implies 6 branches leaving the node
    - Technically, the state tree is infinite
      - This is why we usually limit the exploration to K steps
  - Explore the tree using an array
    - Never use a recursive approach

- Easy, but damn inefficient and unacceptably time consuming
  - Complexity will be $O(nm^K)$
    - Where *n* is the number of goals, *m* is the number of actions, and K the number of steps to plan

**WARNING!**

We cannot give for granted it is always a tree, because if action B reverse action A, we will be looping back to an old state.

Nevertheless, it can be demonstrated that an infinite tree is equivalent to a graph with loops with states duplication

# A Faster States Tree Visit Approach

- We can reduce the computational time considerably by excluding actions leading to a higher discontentment
  - Mathematically speaking, this is the same as accepting a local minimum as a solution

- This may not provide to a 100% believable behaviour

# Single Goal Version: GOAP with IDA*

- If we have one single goal to fulfill A* can be our best friend ... again!
  1. Because we are actually aiming to something (we have a single destination)
  2. Because it is a graph, after all

- Nevertheless, the concept of “best path” must be redefined
  - The shortest in term of number of actions?
  - The shortest in term of time spent to reach the last stage?
  - The one with the lowest amount of resources used?
    - Ammo
    - Mana
    - Money

- Once we defined a cost metric, we run A* and look for the path with the lowest cost

# IDA*

- Problem: our goal could actually be unachievable
  - This may sit in the goal declaration itself
    - Survive as long as possible (in an MMO)
    - Get as rich as possible (in a stock exchange simulation)
- The standard A* will not terminate
- We must content ourselves to go as close as possible to satisfying the goal
  - Using a progressive maximum depth when calculating A* will allow us to achieve this
- We introduce IDA*: Iterative Deepening A*

# IDA* Algorithm

1. A threshold value is set to the estimated distance of the starting point to the goal
2. A deep-first exploration is performed with two constraints
   - The cost of a path must be lower than the threshold
   - There is a maximum acceptable deepness
3. If the goal is reached the algorithm terminates
4. Otherwise, the threshold is set to the lowest cost greater than the old threshold found during exploration and the algorithm goes back to step 2

REMEMBER!
We must also avoid visiting the same configuration more than once

# References

- On the book
  - § 5.7 (excluded 5.7.7)