UNIVERSITÀ DEGLI STUDI
DI MILANO

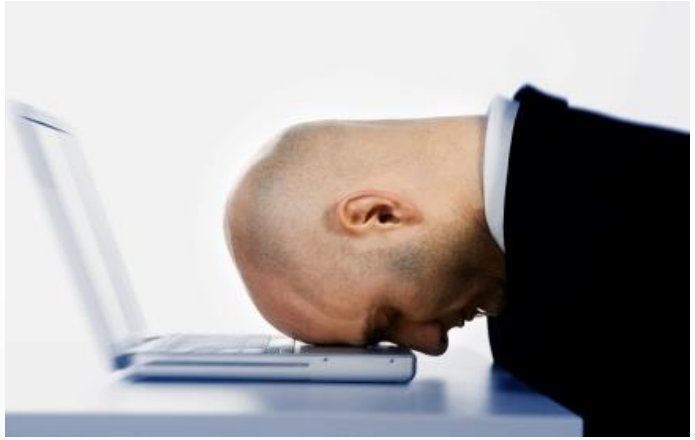
Hierarchical Pathfinding

A.I. for Video Games

Dario Maggiorini (dario@di.unimi.it)

Plans Inside Plans

- Goal: go to the office
  1. Go to the parking lot
  2. Drive car
  3. Park
  4. Go to your desk



1. Go to the parking lot
   1. Exit home
   2. Cross the street
   3. Find car
   4. Unlock car
   5. Open door
   6. Sit in the car

2. Drive car
   1. Star engine
   2. Put drive gear
   3. Press gas
   4. Go to the exit
   5. Check lanes
   6. Enter road
   7. FOLLOW WAY TO OFFICE

Plans Inside Plans

- Goal: go to the office
  1. Go to the parking lot
  2. Drive car
  3. Park
  4. Go to your desk

This is easy enough to think ... But the problem is that every action requires to be expanded to basic operations

Plans Inside Plans

- Goal: go to the office
  1. Go to the parking lot
  2. Drive car
  3. Park
  4. Go to your desk

  1. Go to the parking lot
     1. Exit home
     2. Cross the street
     3. Find car
     4. Unlock car
     5. Open door
     6. Sit in the car

Plans Inside Plans

- Goal: go to the office
  1. Go to the parking lot
  2. Drive car
  3. Park
  4. Go to your desk

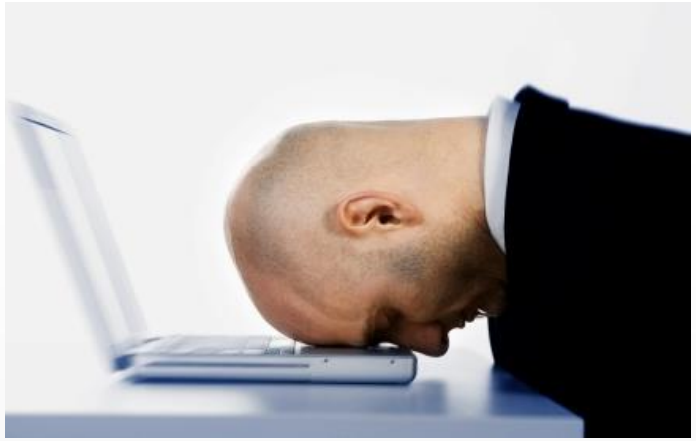
... And the expansion can occur multiple times

1. Go to the parking lot
   1. Exit home
   2. Cross the street
   3. Find car
   4. Unlock car
   5. Open door
   6. Sit in the car

2. Drive car
   1. Star engine
   2. Put drive gear
   3. Press gas
   4. Go to the exit
   5. Check lanes
   6. Enter road

Plans Inside Plans

- Goal: go to the office
  1. Go to the parking lot
  2. Drive car
  3. Park
  4. Go to your desk



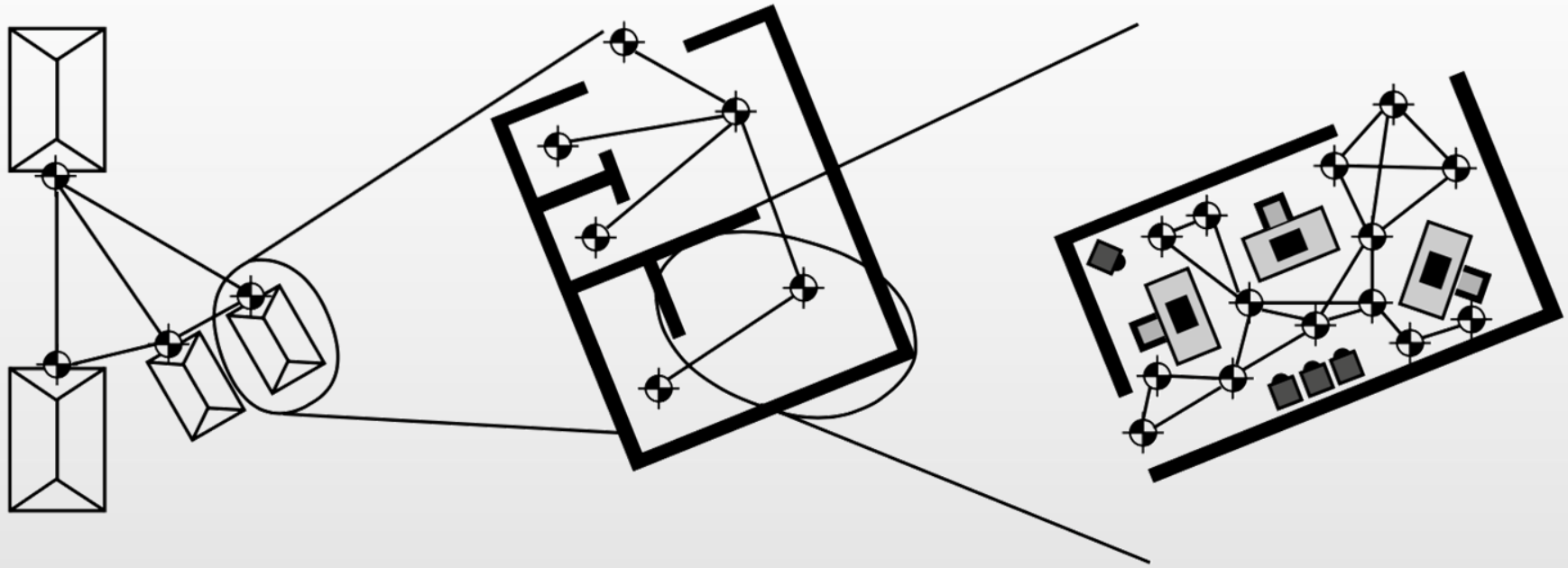
1. Go to the parking lot
   1. Exit home
   2. Cross the street
   3. Find car
   4. Unlock car
   5. Open door
   6. Sit in the car

2. Drive car
   1. Star engine
   2. Put drive gear
   3. Press gas
   4. Go to the exit
   5. Check lanes
   6. Enter road
   7. FOLLOW STREET TO OFFICE

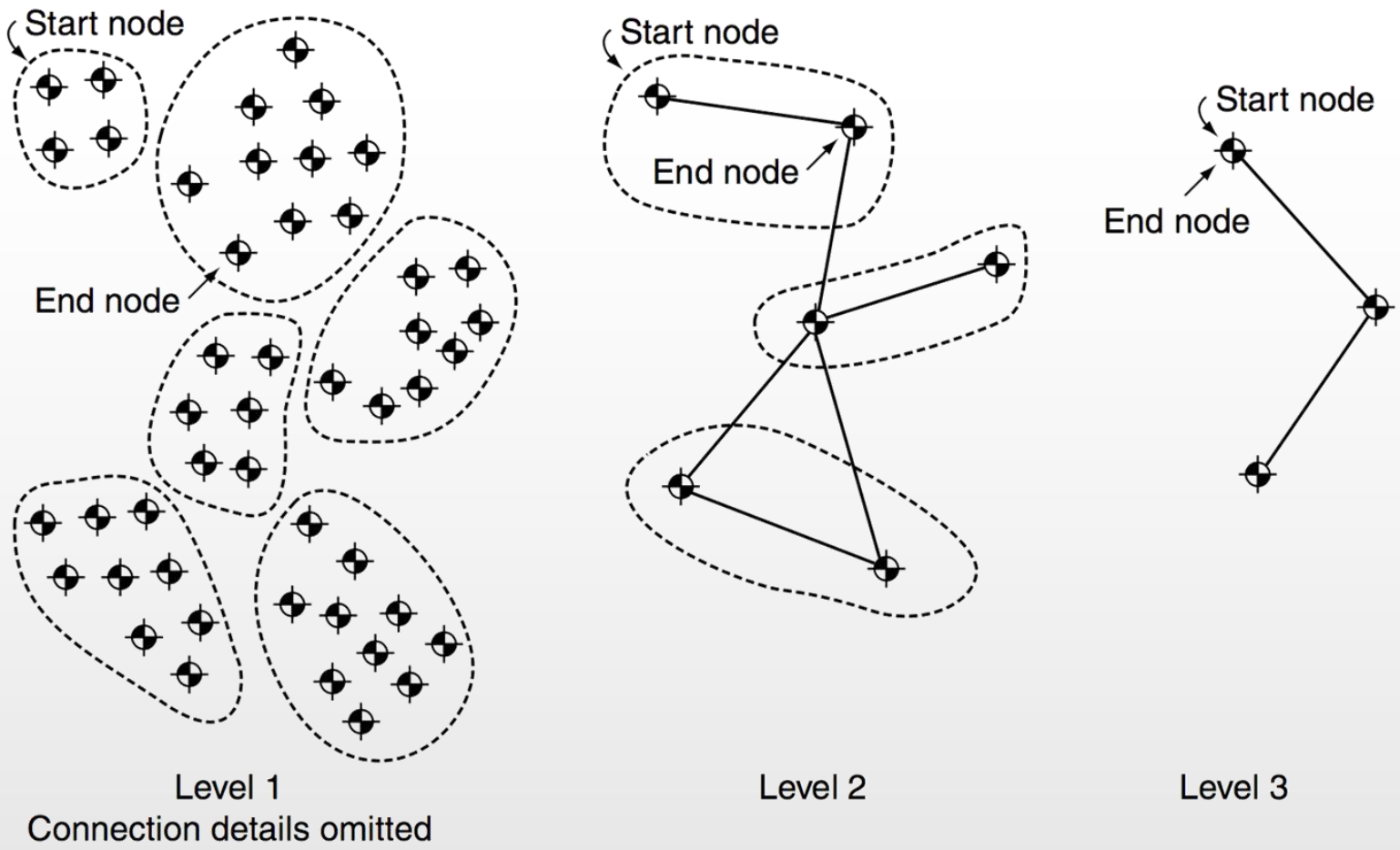
Hierarchical Pathfinding

- We can group nodes and manage them as a single location for the purpose of a higher-level pathfinding

- NOTE: **this is NOT clustering**, because we have more than one graph



Grouping Nodes Example



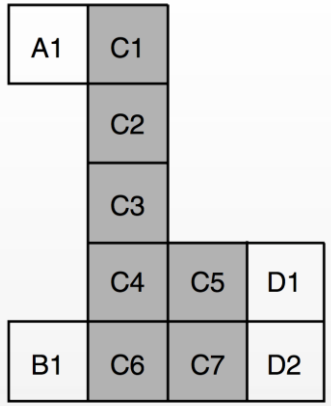
Connecting Groups

- Groups (as nodes in a higher-level graph) must be connected if there is at least a couple of lower-level nodes with and edge linking them

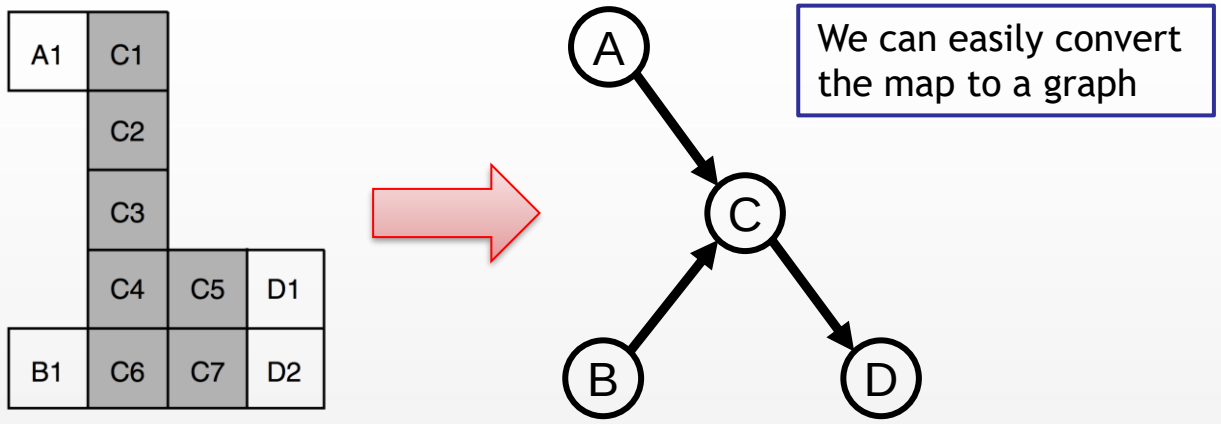
- Problem: calculating edges cost may be a pain
  It can be done
  1. Manually
  2. Using costs from lower level graph

Cost Calculation Problem

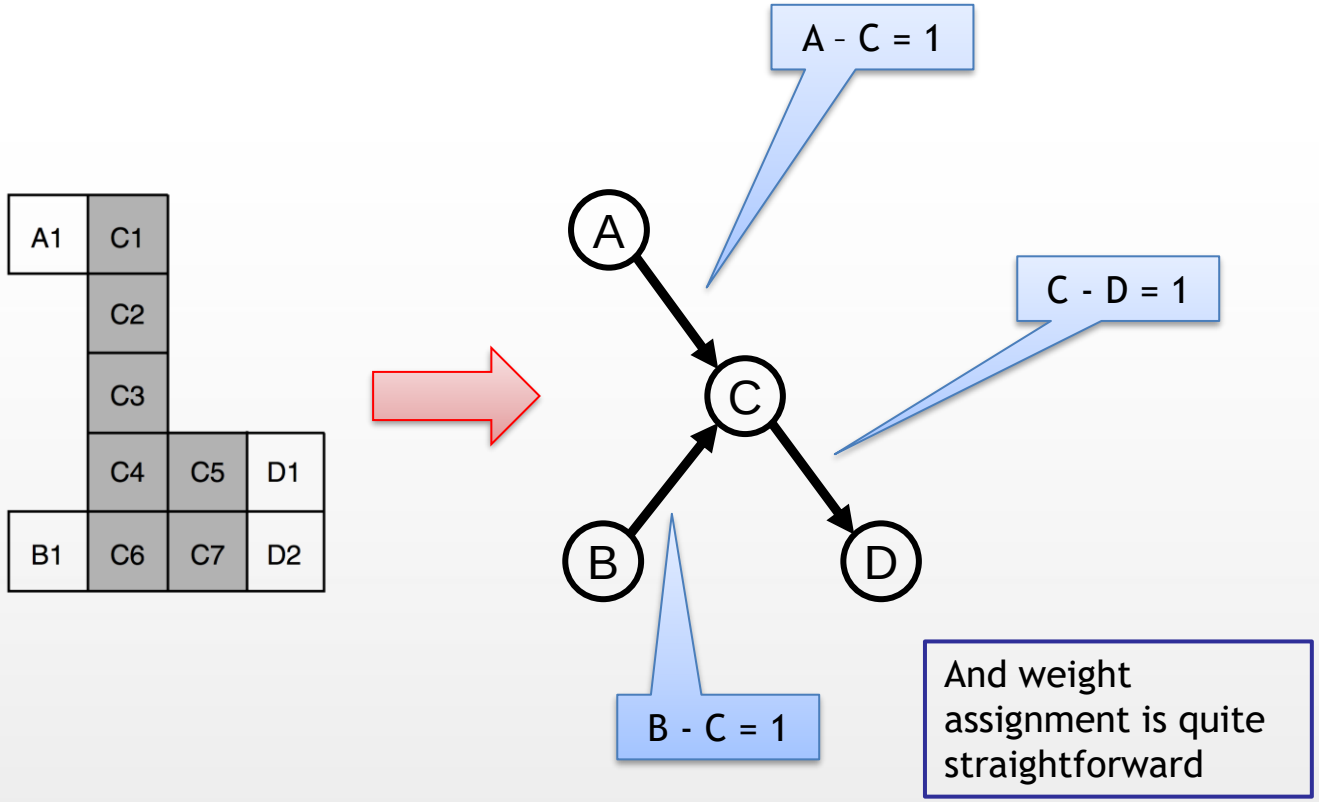
Lets' start from a square-tiled map where we have identified 4 regions: A, B, C, and D



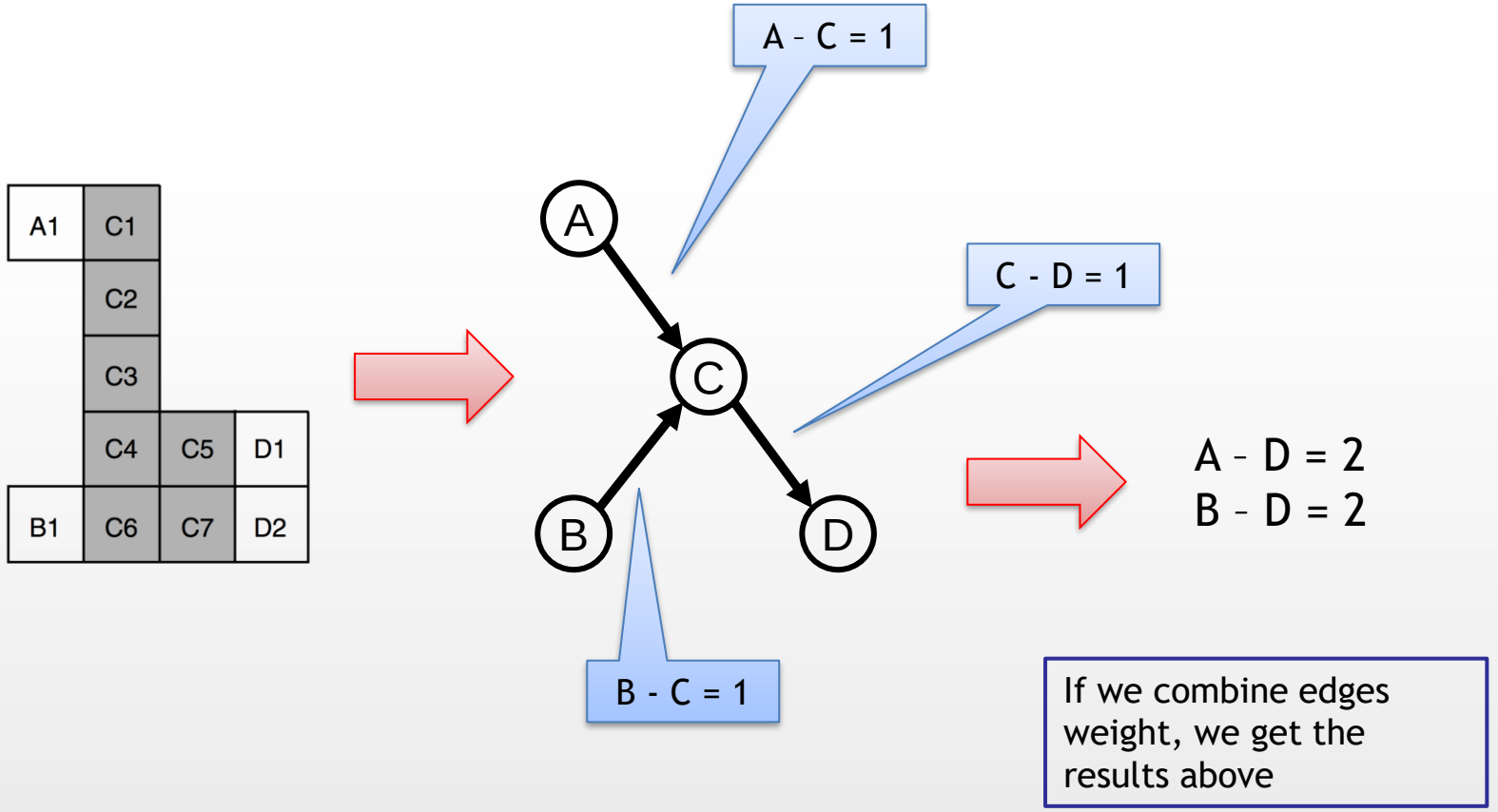
Cost Calculation Problem



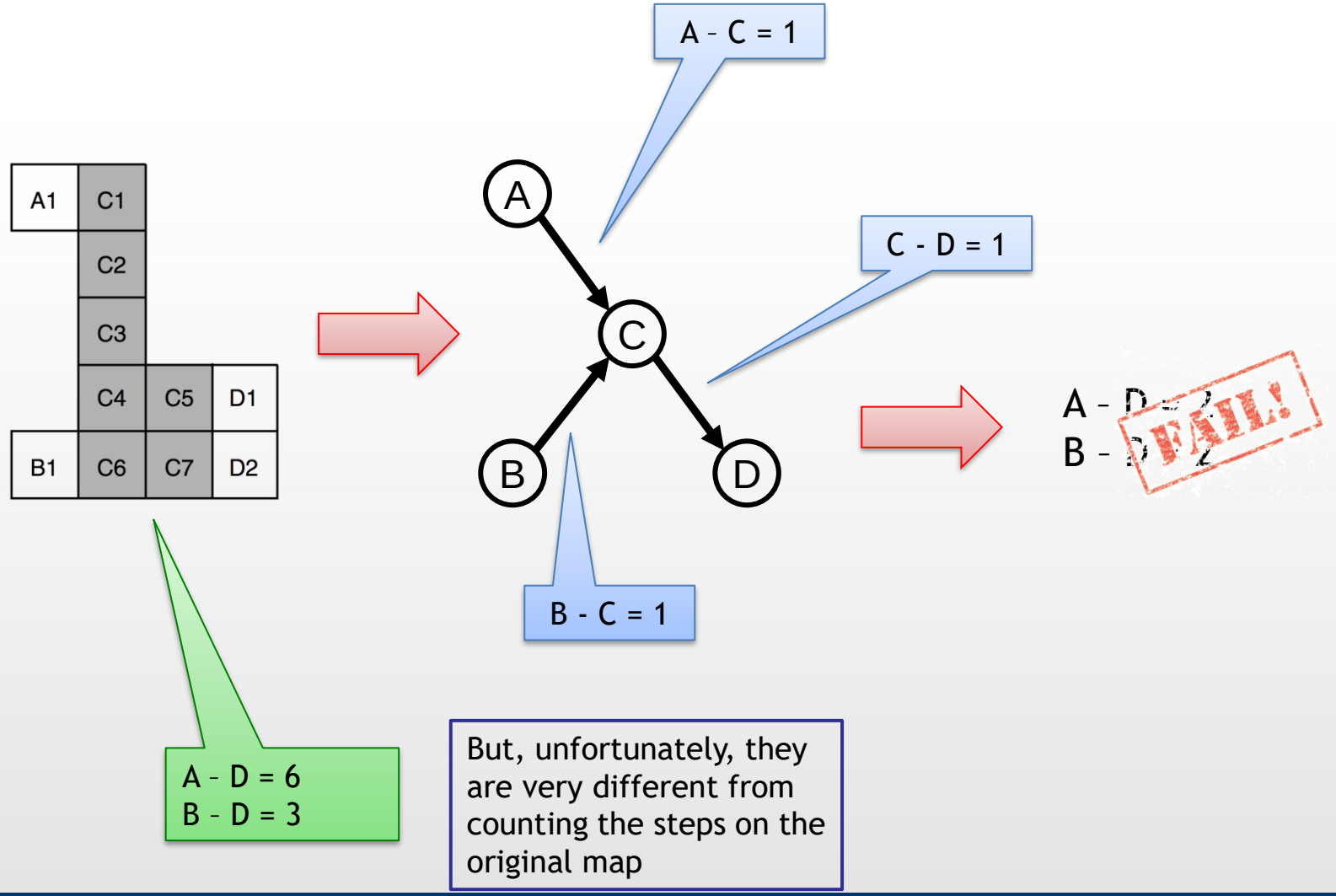
Cost Calculation Problem



Cost Calculation Problem



Cost Calculation Problem



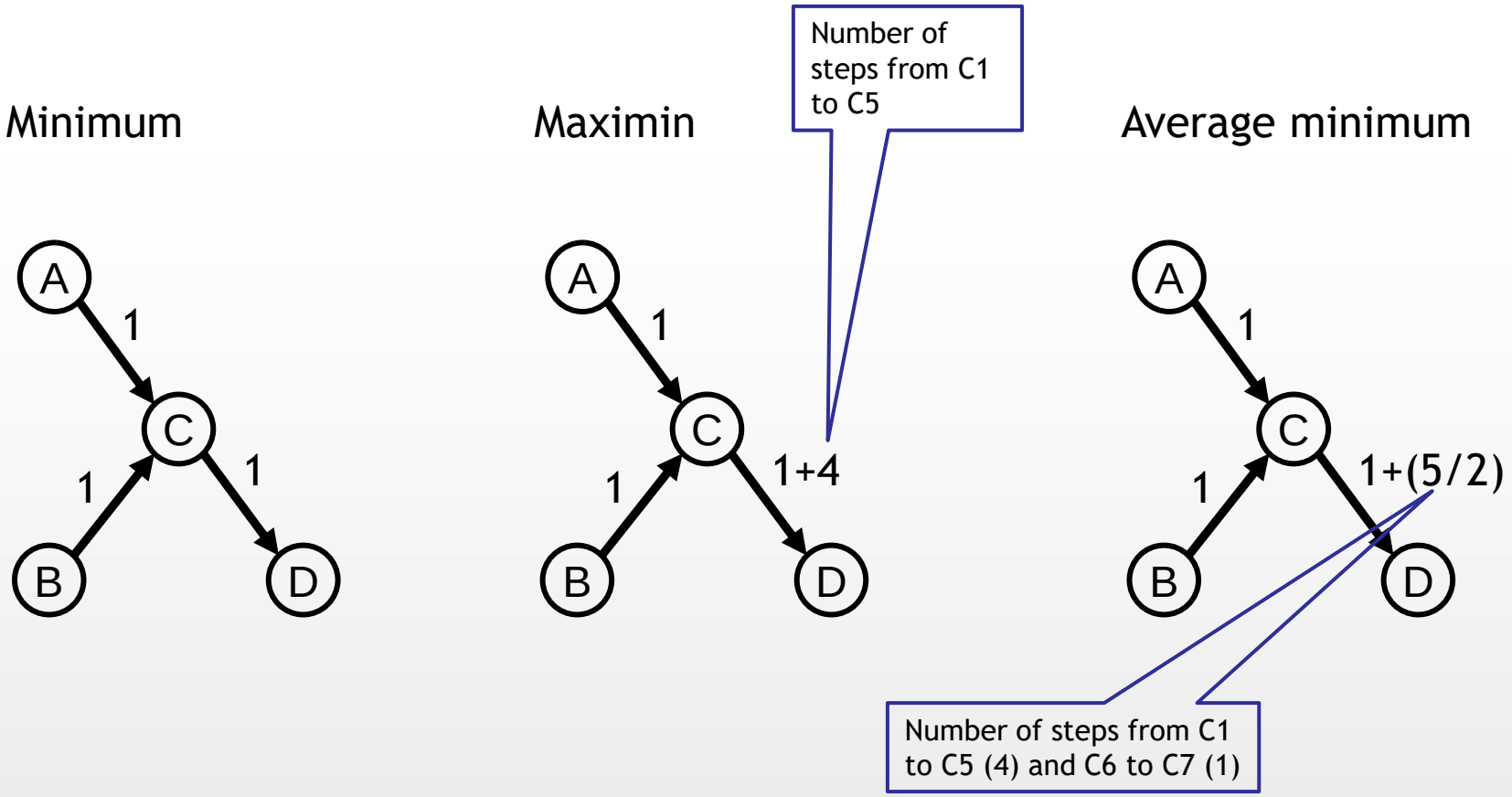
But, unfortunately, they are very different from counting the steps on the original map

Algorithms (Heuristics!) for Cost Calculation

- Minimum distance
  - The minimum weight of all edges linking the two groups
  - This is what our A* algorithm will likely use
  - An underestimation (and we like it)
- Maximin distance
  - We calculate the shortest path inside the area between each incoming node and the outgoing node, and then we take the maximum
    - This will be representative of a “maximum traversal cost” of the area to go toward the outgoing node
  - We set the weight of the outgoing edge as the traversal cost plus the link cost toward the outgoing node
    - **NOTE**: the textbook is assuming oriented graphs at this point
- Average minimum distance
  - Same calculations as in the *maximin* distance, but values are averaged instead of taking the maximum

Avoid getting confused on this

Applied Heuristics



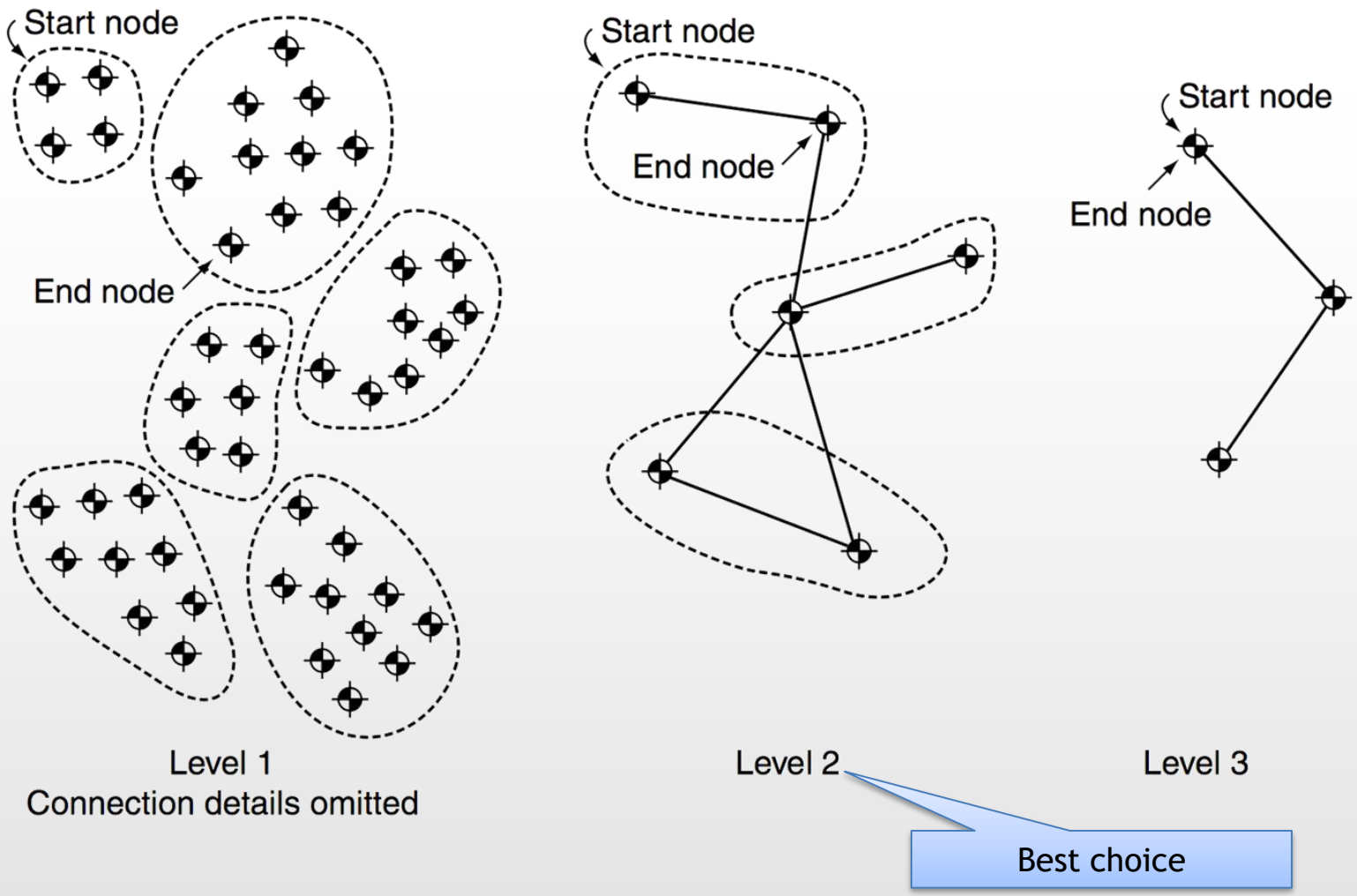
NOTE: Mind the arrows

NOTE2: Watch out for the typo “31/2” on page 259 which is actually 3½ = 3.5

Performing Pathfinding

- Basically, we apply A* (or Dijkstra) starting from the highest-level graph
- The result from each run are used to trim down nodes in the next level
- To avoid solving trivial problems, we should start from the highest level where start and goal are NOT in the same node

Where to Start Pathfinding



Progressive Calculation

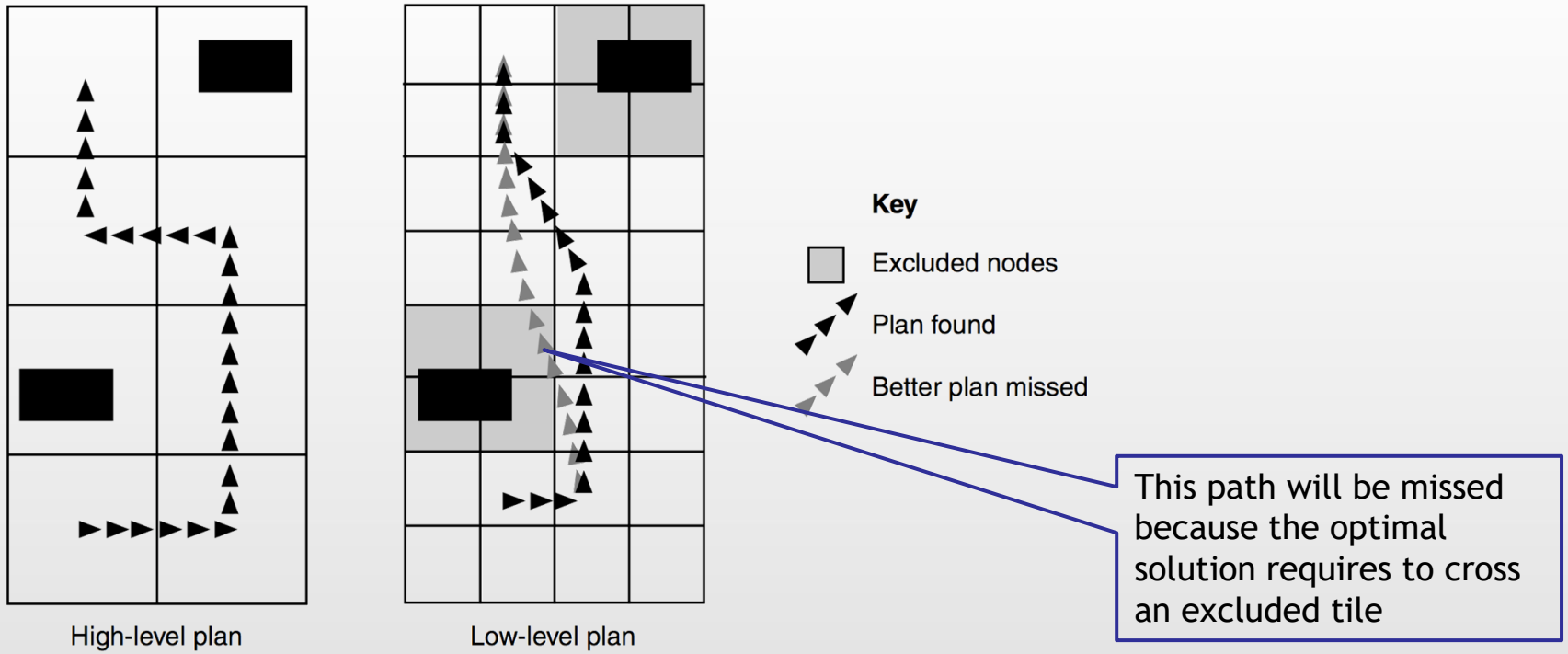
- Hierarchical pathfinding is suitable for progressive calculation

1. We calculate the path on level N
2. For each level M < N we calculate the path only for the first edge of level M+1
3. We can start navigation and then calculate other edges only when needed

BEWARE: errors in cost evaluation play a major role here.
A negligible error on a large scale may force the NPC to create a very unrealistic path at some lower level

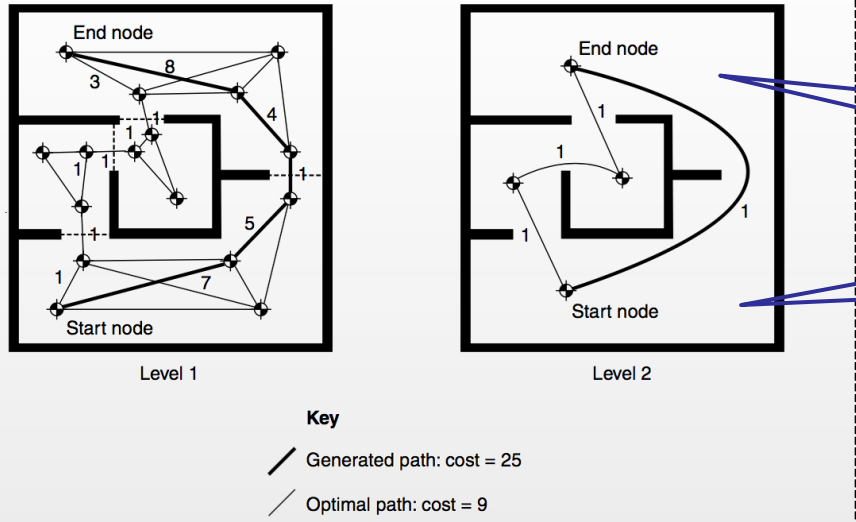
Trimming Pathfinding

- On the other hand, we can avoid progressive calculation and run a higher-level pathfinding in full. The result will be used to exclude nodes in the lower-level computation
- While this can greatly boost performances, excluding nodes may prevent us from finding optimal or more realistic paths



Effects of Errors in Costs Evaluation

- Since we have heuristics, we must live with the fact hierarchical pathfinding is giving us an approximated solution
  - There is no “golden heuristic” working in every single case



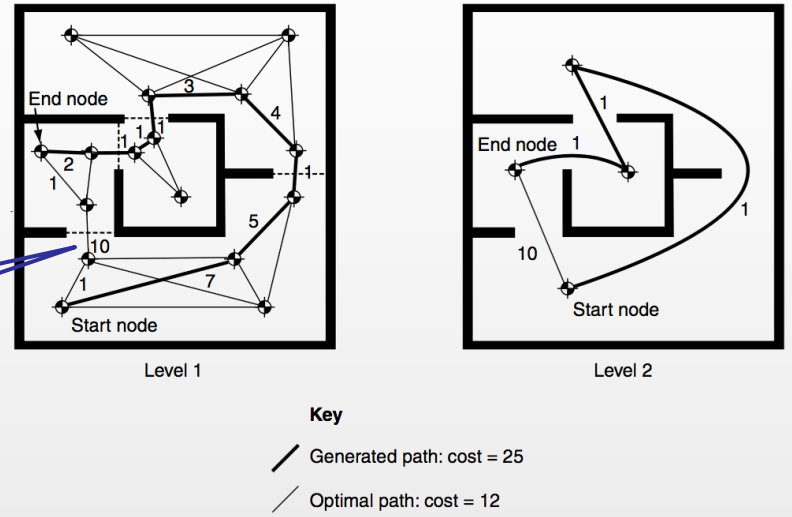
Error with minimum

Large traversal costs are ignored in two areas leading to wrong path estimation

Effects of Errors in Costs Evaluation

- Since we have heuristics, we must live with the fact hierarchical pathfinding is giving us an approximated solution
  - There is no “golden heuristic” working in every single case

One large link cost is pushing the NPC to navigate areas with even greater traversal cost



Error with maximin

References

- On the textbook
  - § 4.6.1
  - § 4.6.2
  - § 4.6.3
  - § 4.6.4