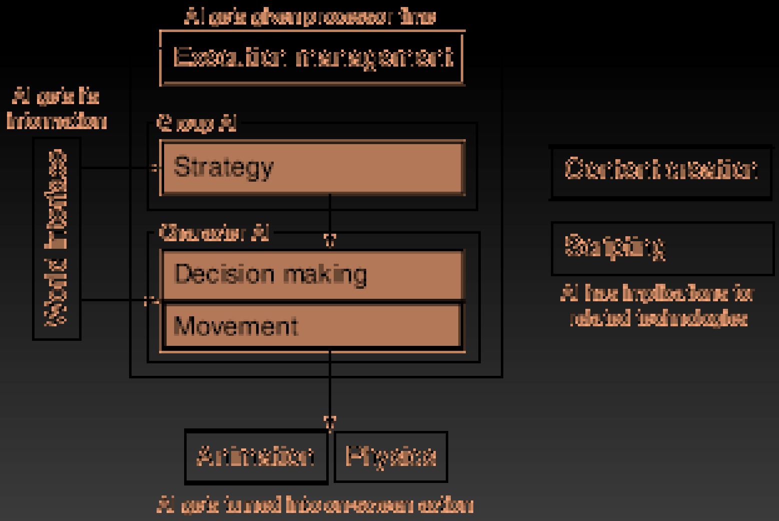
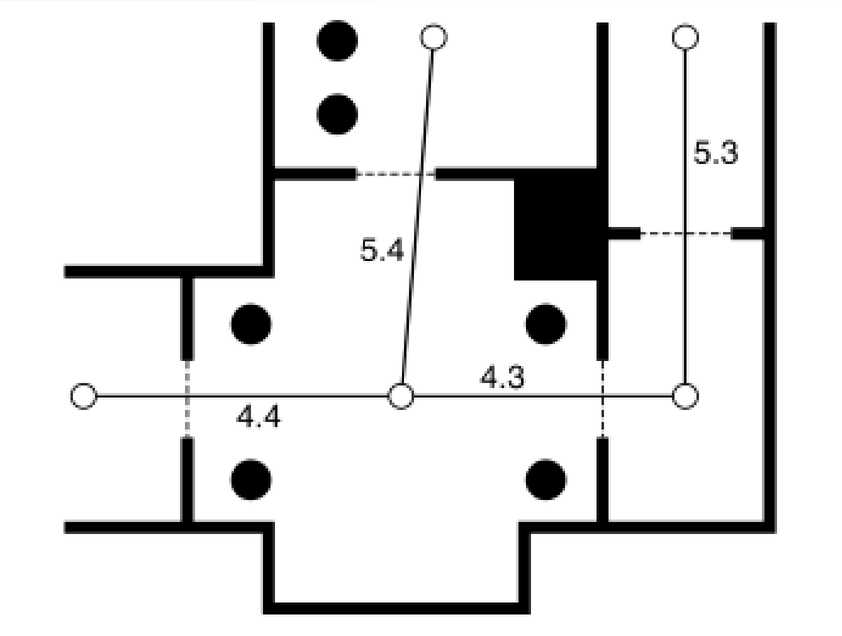
Path Finding

- Graphs
- Search Algorithms
- Heuristic A*
- Any-time ARA*
- Tactical Path Finding
- Navigation Mesh
- Dynamic graph and incremental algorithms
- Movement Path Planning



Weighted Graph for Path Finding



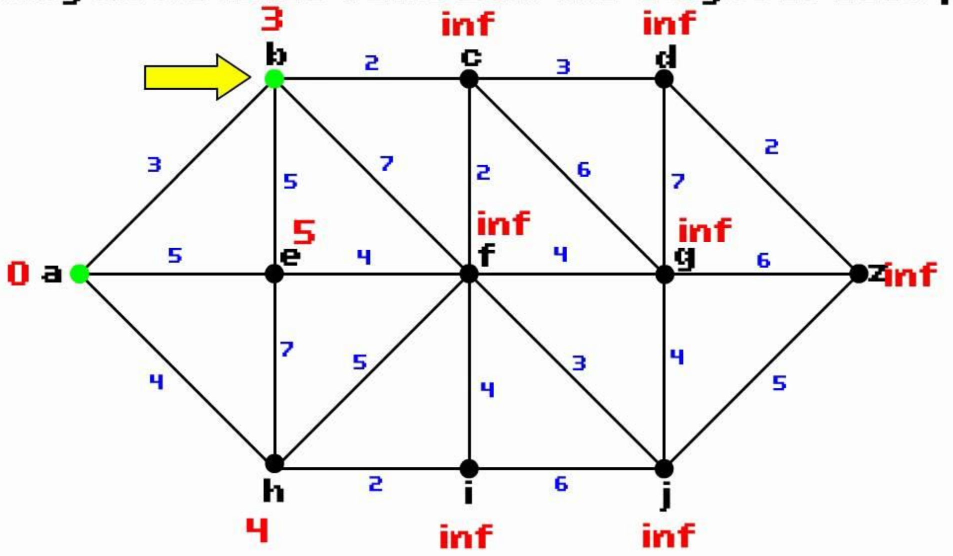
Representation:

- Adjacency List
- Incidence Matrix
- Adjacency Matrix

Search Algorithms

Dijkstra From one to all

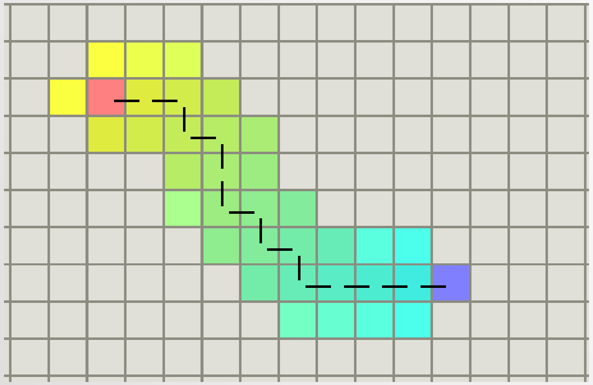
The distance of the adjacent vertices is calculated by adding its distance value with the weight of each path.



Heuristics Search

A* Search

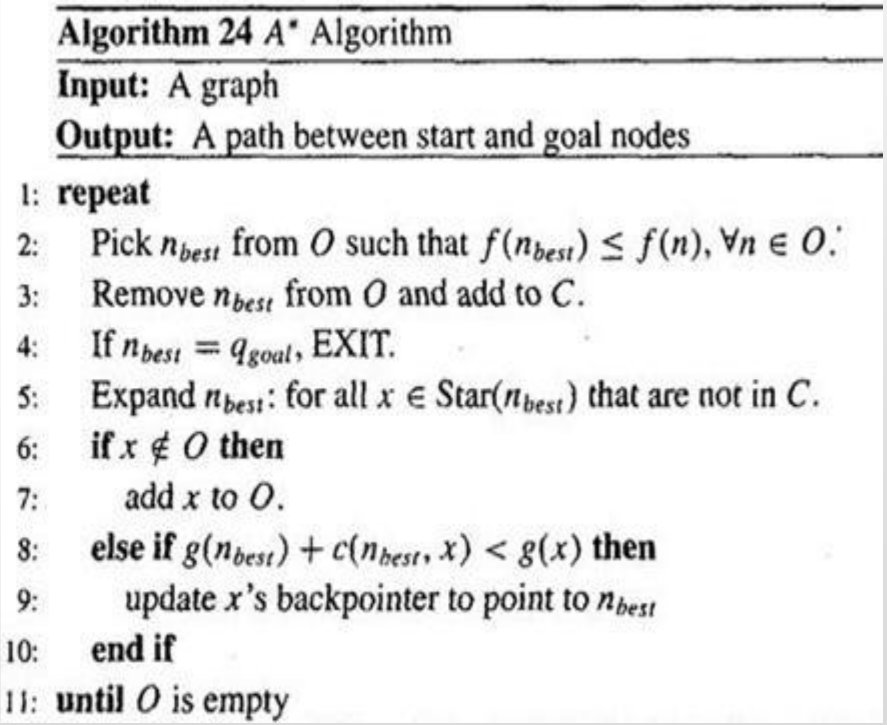
Current Optimal Distance + Cost



Neighbours of the starting point. In each iteration algorithm looks for the value of a function f(n) = g(n) + h(n) for each node n.

For each node with a value of function f(n), the algorithm selects the node with this minimum value and expands the neighboors of this selected node n. A* search also remembers the nodes visited in each iteration

O = Open set, or priority queue which includes the nodes that are subject to search at an iteration step
C = Closed set, the visited nodes so far
c(n1,n2) = the length of the edge connecting n1 and n2
g(n) = total length covered so far in order to reach node n
h(n) = heuristic cost function, or the Euclidian distance from node n to goal
f(n) = g(n) + h(n) main criteria for search and selection evaluation



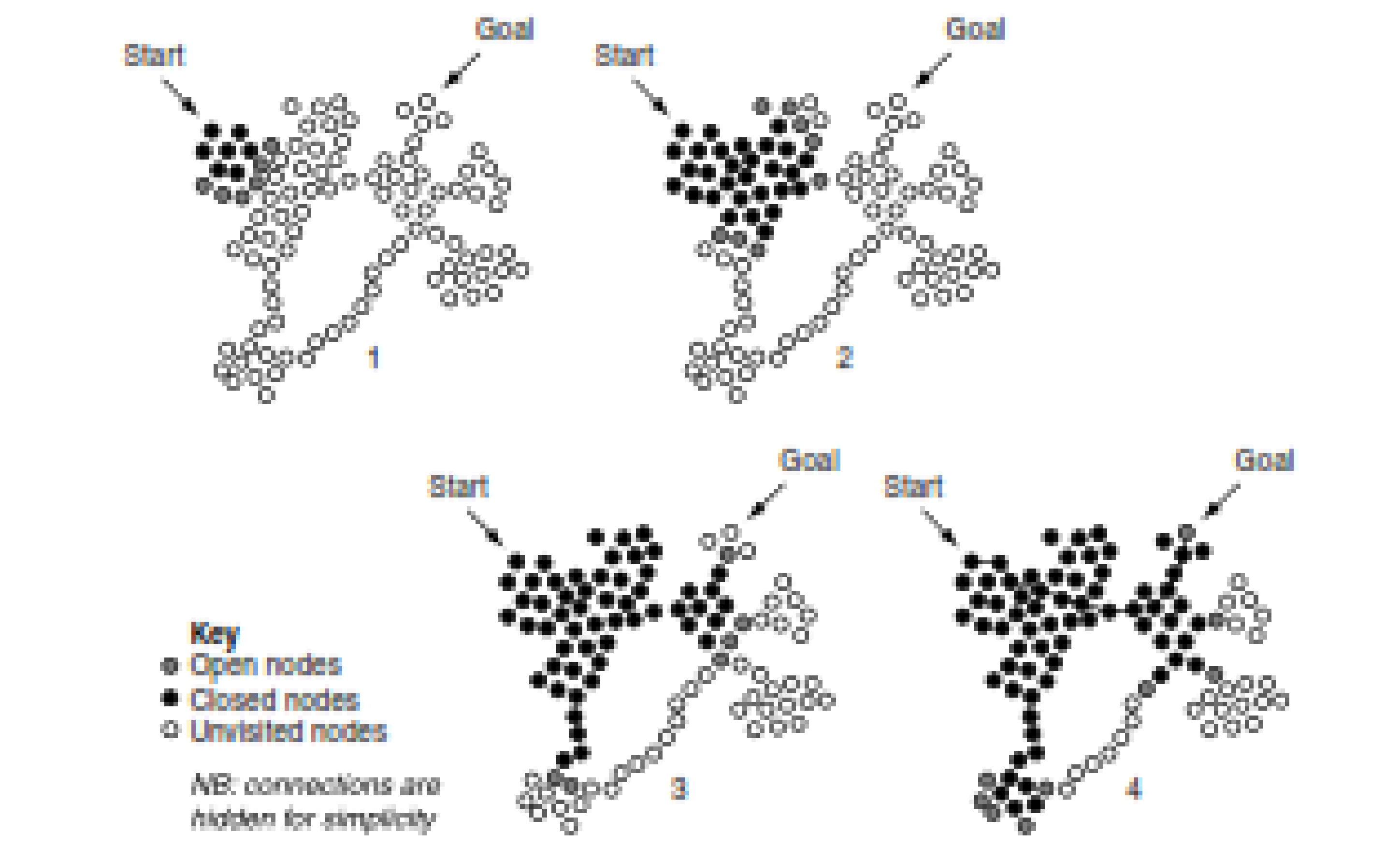
Algorithm 24 A^* Algorithm

Input: A graph

Output: A path between start and goal nodes

```
repeat
    Pick $n_{best}$ from $O$ such that $f(n_{best}) \leq f(n), \forall n \in O$.
    Remove $n_{best}$ from $O$ and add to $C$.
    If $n_{best} = q_{goal}$, EXIT.
    Expand $n_{best}$: for all $x \in Star(n_{best})$ that are not in $C$.
    if $x \notin O$ then
        add $x$ to $O$.
    else if $g(n_{best}) + c(n_{best}, x) < g(x)$ then
        update $x$'s backpointer to point to $n_{best}$
    end if
until $O$ is empty
```

A* Iterations



Self-study

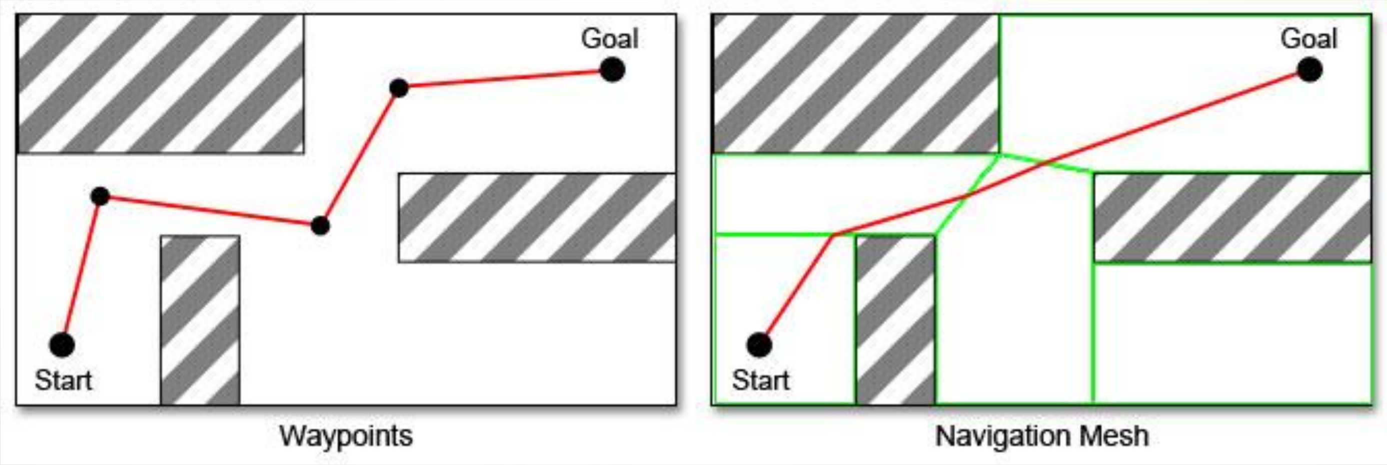
Different Graph Search Algorithms

http://web.cs.wpi.edu/~cs4341/b03/Projects/Project1/Solutions/solutions_hw1.html

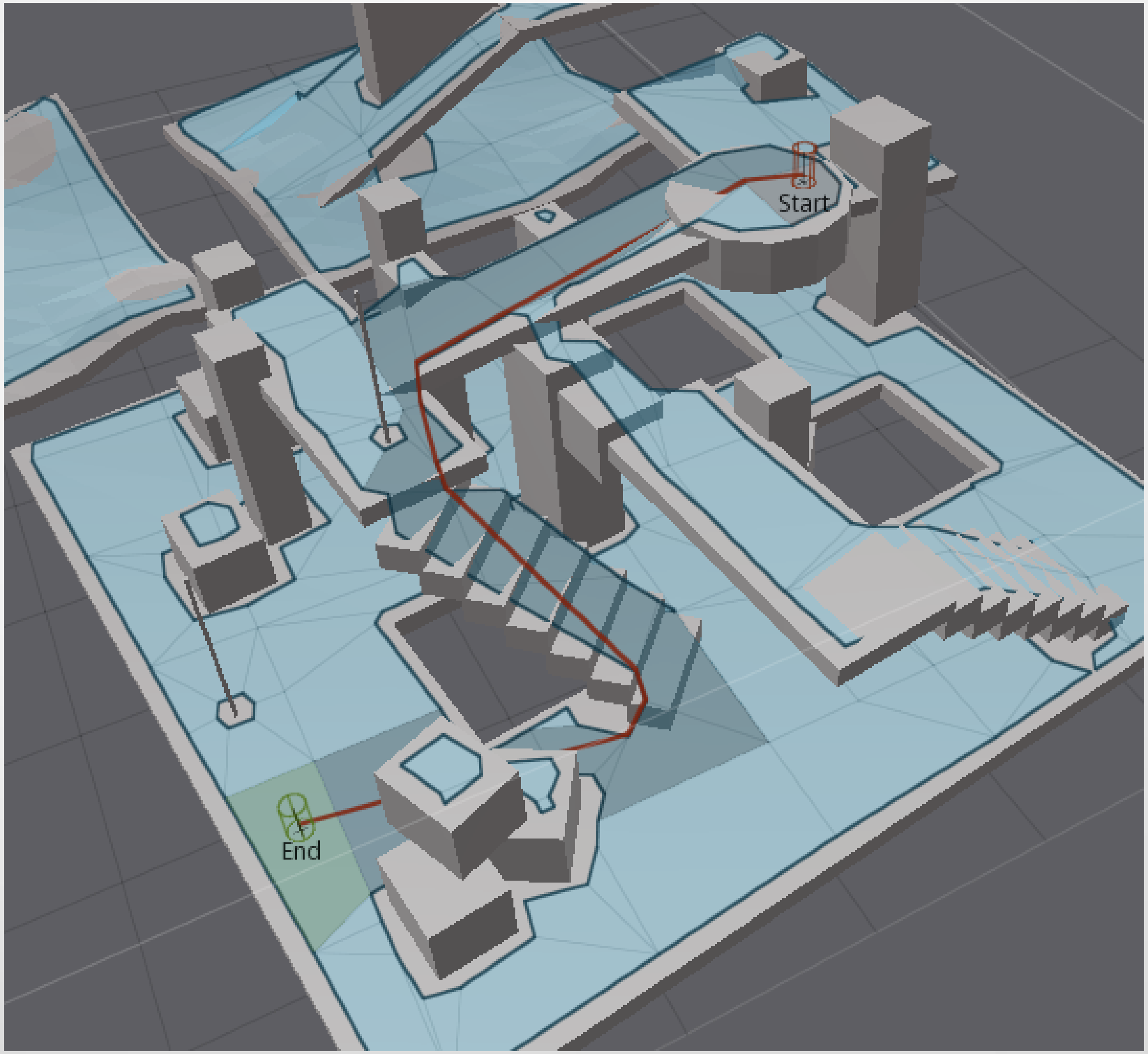
Incremental and Any-time Algorithms

http://www.aaai.org/ocs/index.php/ICAPS/ICAPS12/paper/viewFile/4724/4735

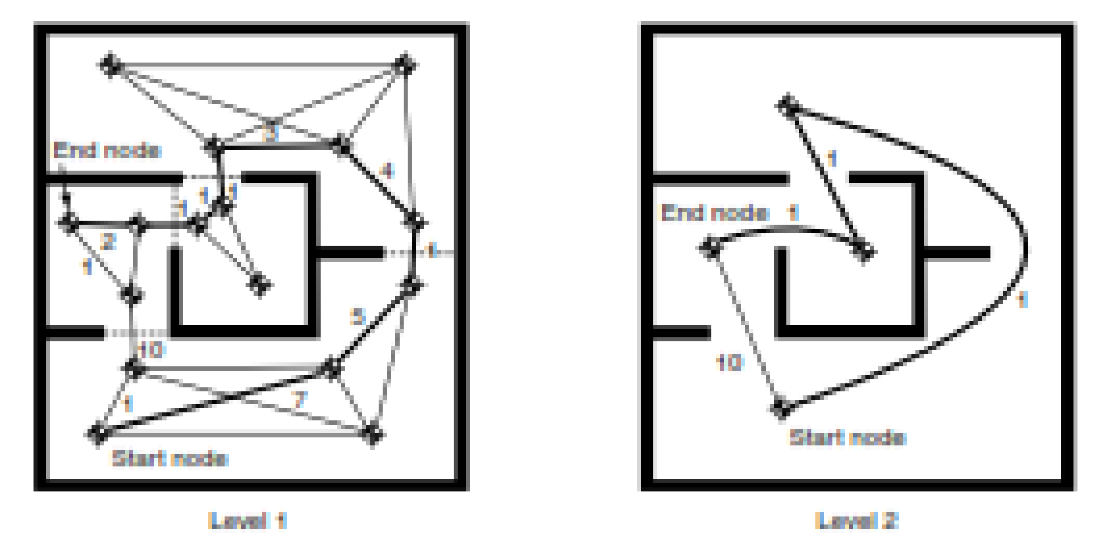
Voronoi-based Navigation Mesh



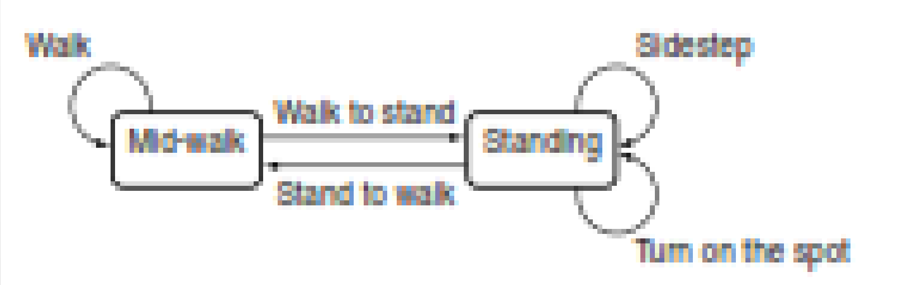
Path Finding using NavMesh without NavLinks



Hierarchical Path Planning



Position controlled by finite state machine


Walk
Mid-walk
Walk to stand
Stand to walk
Standing
Sidestep
Turn on the spot