EE365: Informed Search

Dijkstra's algorithm

$v_s = 0$
$v_i = \infty$ for all $i \neq s$
$F = \{s\}$
while $F \neq \emptyset$
    $i = \underset{i \in F}{\operatorname{argmin}} \, v_i$ // extract vertex i
    $F = F \setminus \{i\}$
    if $i \in \mathcal{T}$ *terminate* // found target
    for $j \in \mathcal{N}_i$
        if $v_j > v_i + g_{ij}$
            $v_j = v_i + g_{ij}$
            $F = F \cup \{j\}$

- explores $\mathcal{V}$ closest first
- stops upon reaching the target set
- needs $\mathbf{dist}(i, \mathcal{T}) \geq 0$ for all i

$A^\star$ algorithm

$v_s = 0$
$v_i = \infty$ for all $i \neq s$
$F = \{s\}$
while $F \neq \emptyset$
    $i = \underset{i \in F}{\operatorname{argmin}}\, v_i + h_i$ // *modified extraction rule*
    $F = F \setminus \{i\}$
    if $i \in \mathcal{T}$ *terminate*   // found target
    for $j \in \mathcal{N}_i$
        if $v_j > v_i + g_{ij}$
            $v_j = v_i + g_{ij}$
            $F = F \cup \{j\}$

- h_i is an *estimate* of the distance from i to the target $\mathbf{dist}(i, \mathcal{T})$
- h is called the *heuristic* function
- idea is to *guide* the search to look first in directions suggested by the heuristic

Informed search

- h is the *heuristic* function
- h_i is an estimate of the optimal *cost to go* from i to the target
- search first in directions with smallest estimated *total cost*
- a good choice of h reduces the number of vertices explored by the search
- and reduces the number of steps before termination
- called *informed search*
- correspondingly, shortest path algorithms without heuristics are called *uninformed search*

Reduction to Dijkstra's algorithm

- construct *transformed graph*, with weights $\hat{g}_{ij} = g_{ij} + h_j - h_i$
- applying Dijkstra to the transformed graph is the same as applying $A^\star$ to the original graph

Reduction to Dijkstra's algorithm

for any path $u \to w \to x \to \ldots \to y \to z$

$$\hat{g}(u \rightsquigarrow z) = g(u \rightsquigarrow z) + h_z - h_u$$

because

$$\hat{g}(u \rightsquigarrow z) = g_{uw} + h_w - h_u + g_{wx} + h_x - h_w + \cdots + g_{yz} + h_z - h_y$$

- we'll see that $A^\star$ finds the shortest path in the transformed graph (Dijkstra)
- with target vertex t, algorithm $A^\star$ therefore minimizes $g(s \rightsquigarrow t) + h_t$

Reduction to Dijkstra's algorithm

- let $\hat{v}$ be the distance estimate in Dijkstra's algorithm
- let v be the distance estimate in $A^\star$
- then the algorithms are the same, with $\hat{v}_i = v_i + h_i - h_s$, because
  - $\hat{v}_j - \hat{v}_i + \hat{g}_{ij} = v_j - v_i + g_{ij}$ so the same edges are relaxed
  - $\operatorname*{argmin}_i \hat{v}_i = \operatorname*{argmin}_i v_i + h_i$ so the same vertices are extracted

Admissible heuristics

the function h is called *admissible* if, for all $i \in \mathcal{V}$,

$$h_i \leq \mathbf{dist}(i, \mathcal{T})$$

- if h is admissible, then

$$\begin{aligned} \widehat{\mathbf{dist}}(i, \mathcal{T}) &= \min_{j \in \mathcal{T}} \widehat{\mathbf{dist}}(i, j) \\ &= \min_{j \in \mathcal{T}} (\mathbf{dist}(i, j) + h_j - h_i) \\ &= \mathbf{dist}(i, \mathcal{T}) - h_i \\ &\geq 0 \end{aligned}$$

- hence admissibility implies that $\widehat{\mathbf{dist}}(i, \mathcal{T}) \geq 0$ for all i
- this is precisely the condition required by Dijkstra's algorithm
- if h is admissible, then $A^\star$ will terminate with a shortest path from s to $\mathcal{T}$

Consistent heuristics

the function h is called *consistent* if $h_x = 0$ for all $x \in \mathcal{T}$ and for all $i, j \in \mathcal{V}$,

$$g_{ij} + h_j - h_i \geq 0$$

- a *Bellman inequality*
- also called a *monotone* heuristic
- hence, for any path, $g(i \rightsquigarrow j) \geq h_i - h_j$
- implies admissibility, since

$$\begin{aligned} \mathbf{dist}(i, \mathcal{T}) &= \min_{j \in \mathcal{T}} \mathbf{dist}(i, j) \\ &\geq \min_{j \in \mathcal{T}} (h_i - h_j) \\ &= h_i \end{aligned}$$

Consistent heuristics

- if h is consistent then the weights in the transformed graph are *nonnegative*
- with nonnegative weights, Dijkstra extracts each vertex once, and never revisits vertices
- hence $A^\star$ never *backtracks*

Constructing heuristics

- relax constraints on the allowed actions; gives an admissible heuristic
- pointwise maximum of admissible (consistent) heuristics is admissible (consistent)

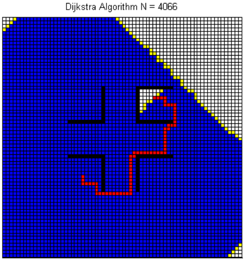
Example: Two dimensional grid

- Estimate the distance to target through the Manhattan distance:

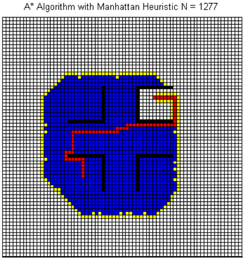
$$h_u = |u_x - t_x| + |u_y - t_y|$$

- Manhattan distance is a lower bound, since it assumes no obstacles
- in fact, it is a consistent heuristic

Left: Uninformed search $h = 0$, $N = 4066$



Right: Heuristic search $N = 1277$.



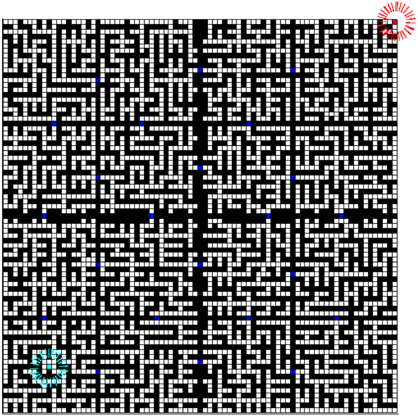
Two dimensional maze

Problem: find shortest path in the following maze.

- Starting position is with cyan.
- target position with red.
- waypoints between squares are denoted with blue.

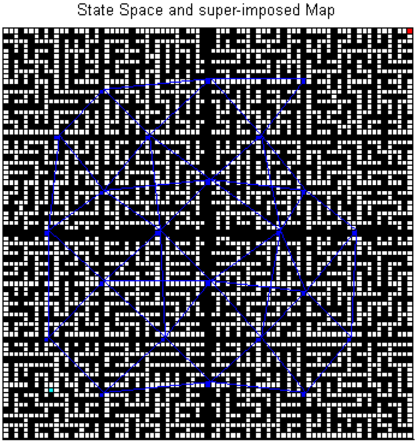
Two dimensional maze

Problem: Find the shortest path between starting position and target.



Waypoints graph

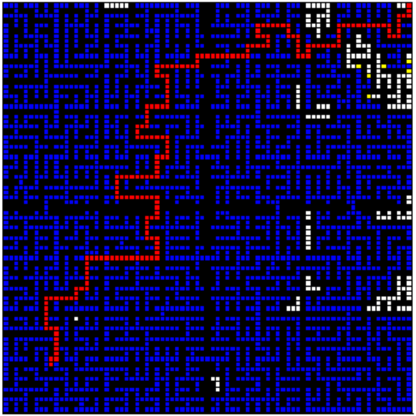
Waypoints between blocks and connectivity pattern.



Two dimensional maze

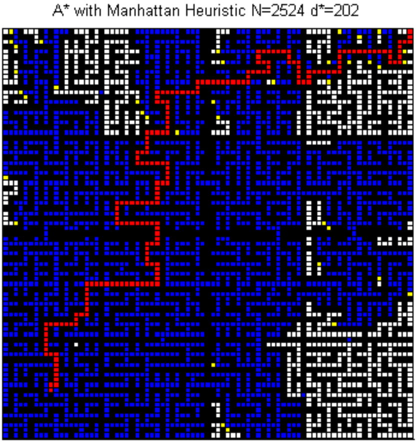
- Using **uninformed search** $h = 0$, we essentially have to explore the whole space before we find the shortest path.

A* with Zero Heuristic(Dijkstra N=3132 d*=202



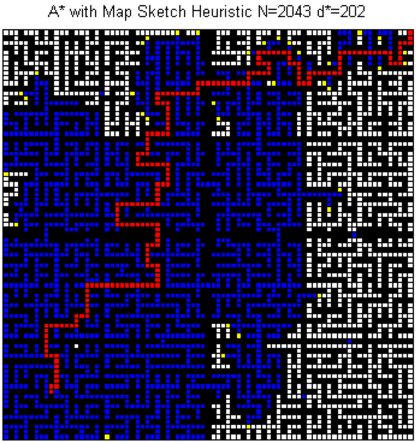
Two dimensional maze

- **Manhattan Distance Estimate**: $\hat{h}_u = |u_x - t_x| + |u_y - t_y|$. Essentially assumes there are no obstacles (relaxes constraints).



Two dimensional maze

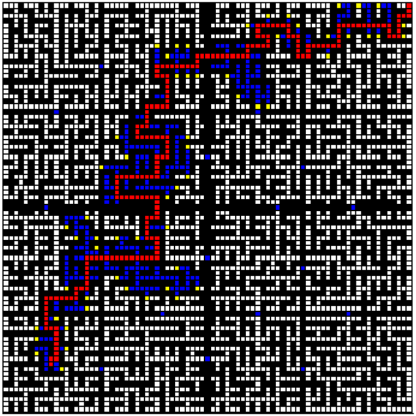
- **Waypoints Graph** with *Manhattan Distance Weights*. Essentially assumes there are no obstacles in going from one waypoint to the other.



Two dimensional maze

- **Waypoints Graph** with *Computed Pairwise Distance Weights*. Essentially assumes there are no obstacles in going from the point to the closest waypoint and from the last waypoint to the target.

A* with Map Exact Heuristic N=492 d*=202



Search strategies

- both Dijkstra and $A^\star$ are guaranteed to find the *optimal solution* if it exists
- heuristics *change the sequence* in which vertices are searched
- $A^\star$ heavily used in practice
- most common limitation is available memory
- further refinements possible to trade-off time/memory