Routing

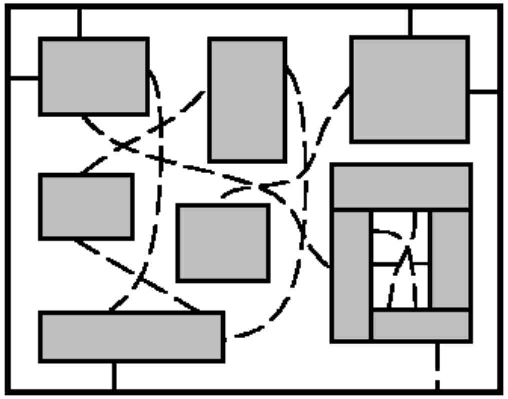
Routing

placement

•Generates a "loose" route for each net.

•Assigns a list of routing regions to each net without specifying the actual layout of wires.

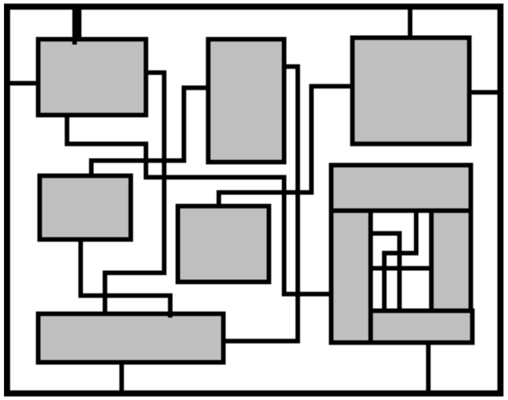
global routing



Global routing

•Finds the actual geometric layout of each net within the assigned routing regions.

detailed routing

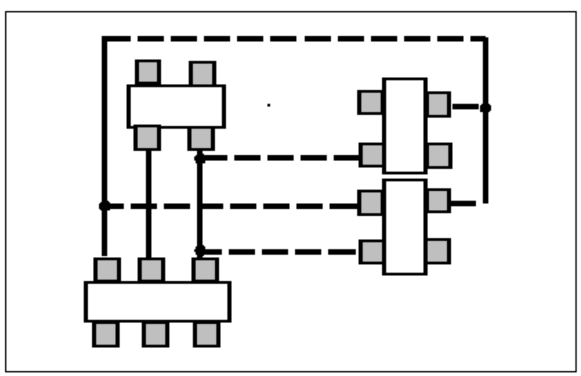


Detailed routing

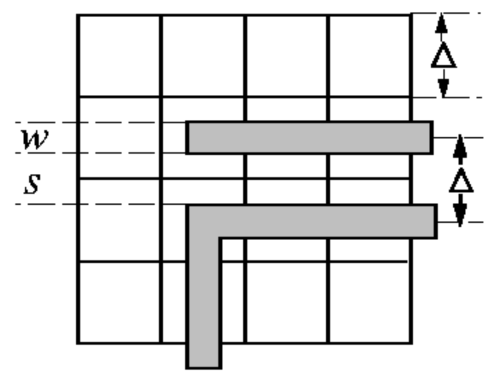
compaction

Routing Constraints

- 100% routing completion + area minimization, under a set of constraints:
  - Placement constraints: usually based on fixed placement
  - Number of routing layers
  - Geometrical constraints: must satisfy design rules
  - Timing constraints (performance-driven routing): must satisfy delay constraints
  - Others



Two-layer routing



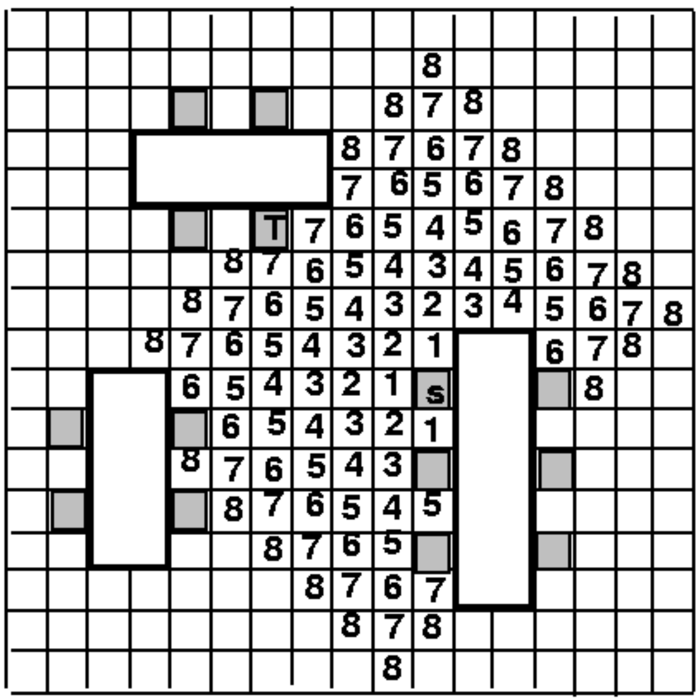
Geometrical constraint

Maze Router: Lee Algorithm

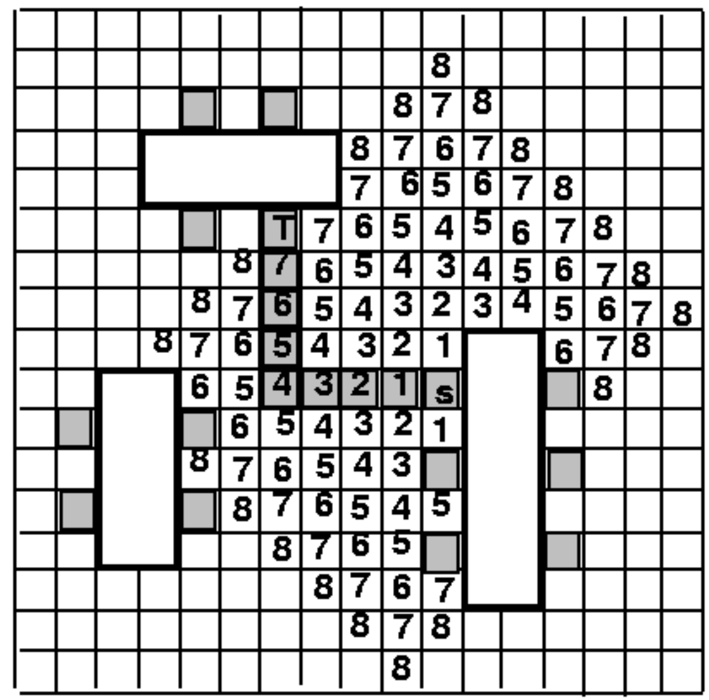
- Lee, "An algorithm for path connection and its application," *IRE Trans*. Electronic Computer, EC-10, 1961.
- Discussion mainly on single-layer routing
- Strengths
  - Guarantee to find connection between 2 terminals if it exists.
  - Guarantee minimum path.
- Weaknesses
  - Requires large memory for dense layout
  - Slow
- Applications: global routing, detailed routing

Lee Algorithm

- Find a path from *S* to *T*.



Filling (Wave propagation)

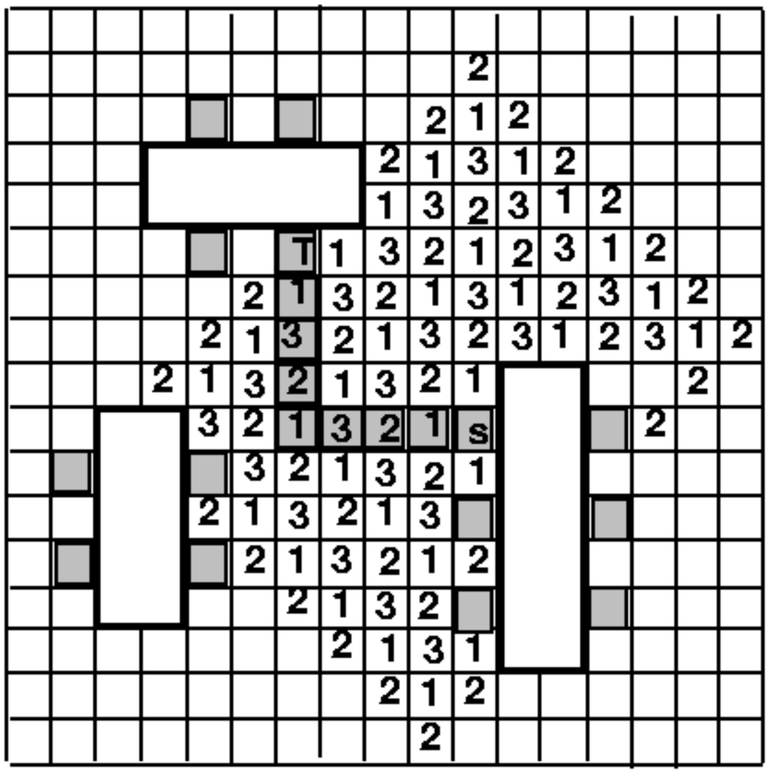


Retrace

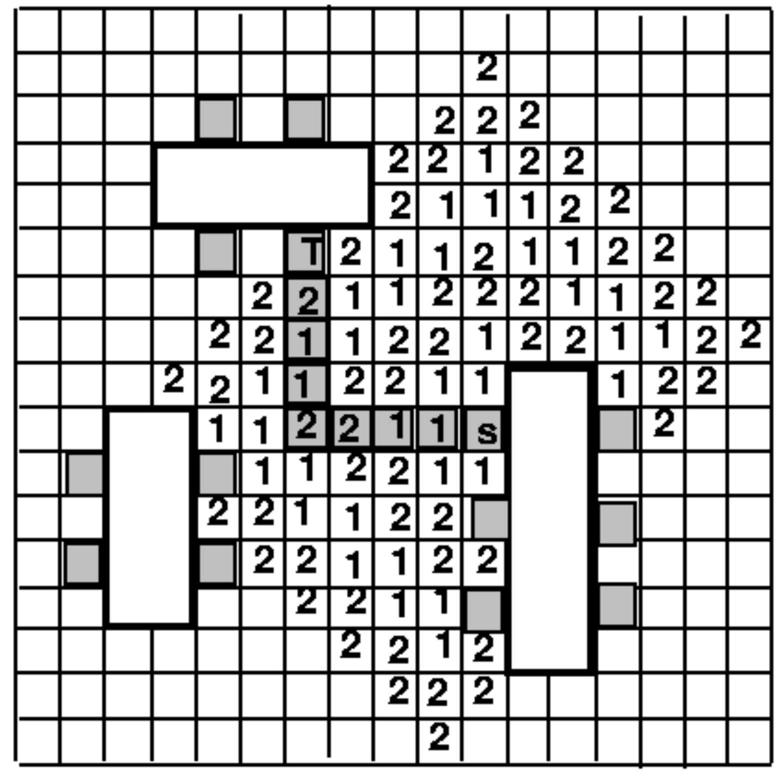
- Time & space complexity for an *M**N* grid: *O*(*MN*) (**huge!**)

Reducing Memory Requirement

- Akers's Observation (1967)
  - Adjacent labels for *k* are either *k*-1 or *k*+1.
  - Want a labeling scheme such that each label has its preceding label different from its succeeding label.
- **Way 1:** coding sequence 1,2,3,1,2,3,…; states: 1, 2, 3, empty, *blocked* (3 bits required)
- **Way 2:** coding sequence 1,1,2,2,1,1,2,2,…; states: 1, 2, empty, *blocked* (need only 2 bits)



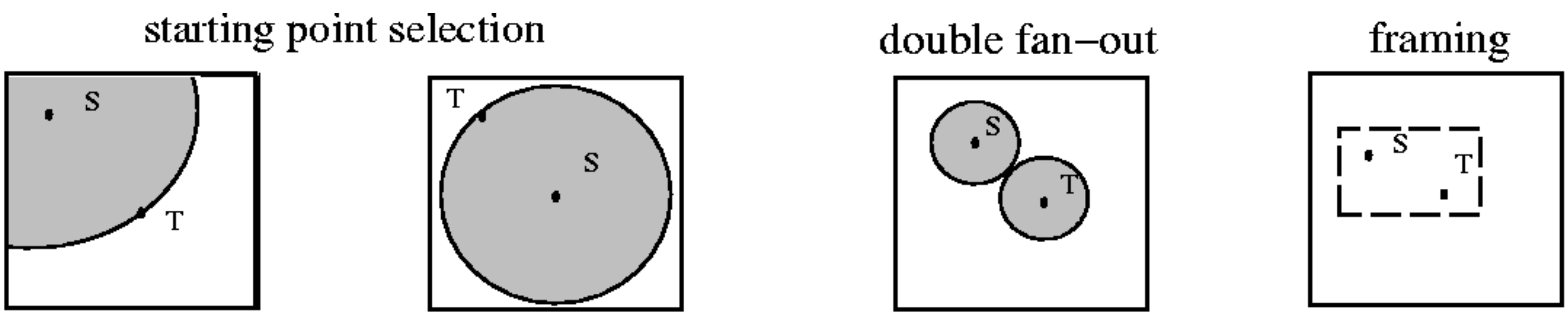
Sequence: 1, 2, 3, 1, 2, 3, ...



Sequence: 1, 1, 2, 2, 1, 1, 2, 2, ...

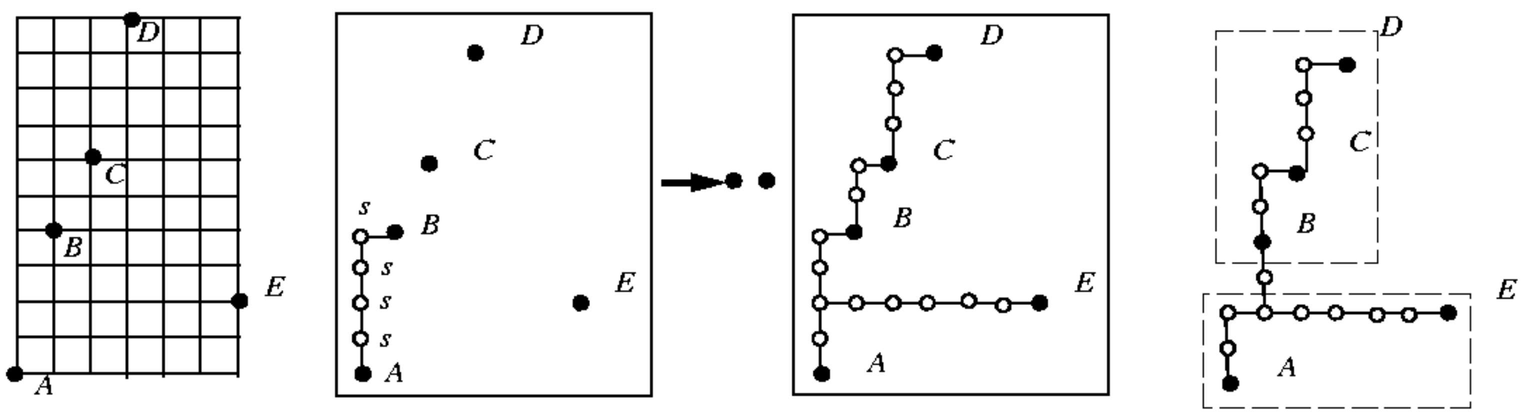
Reducing Running Time

- **Starting point selection:** Choose the point farthest from the center of the grid as the starting pint.
- **Double fan-out:** Propagate waves from both the source and the target cells.
- **Framing:** Search inside a rectangle area 10-20% larger than the bounding box containing the source and target.
  - Need to enlarge the rectangle and redo if the search fails.



Connecting Multi-Terminal Nets

- Step 1: Propagate wave from the source s to the closet target.
- Step 2: Mark ALL cells on the path as *s*.
- Step 3: Propagate wave from ALL *s* cells to the other cells.
- Step 4: Continue until all cells are reached.
- Step 5: Apply heuristics to further reduce the tree cost.

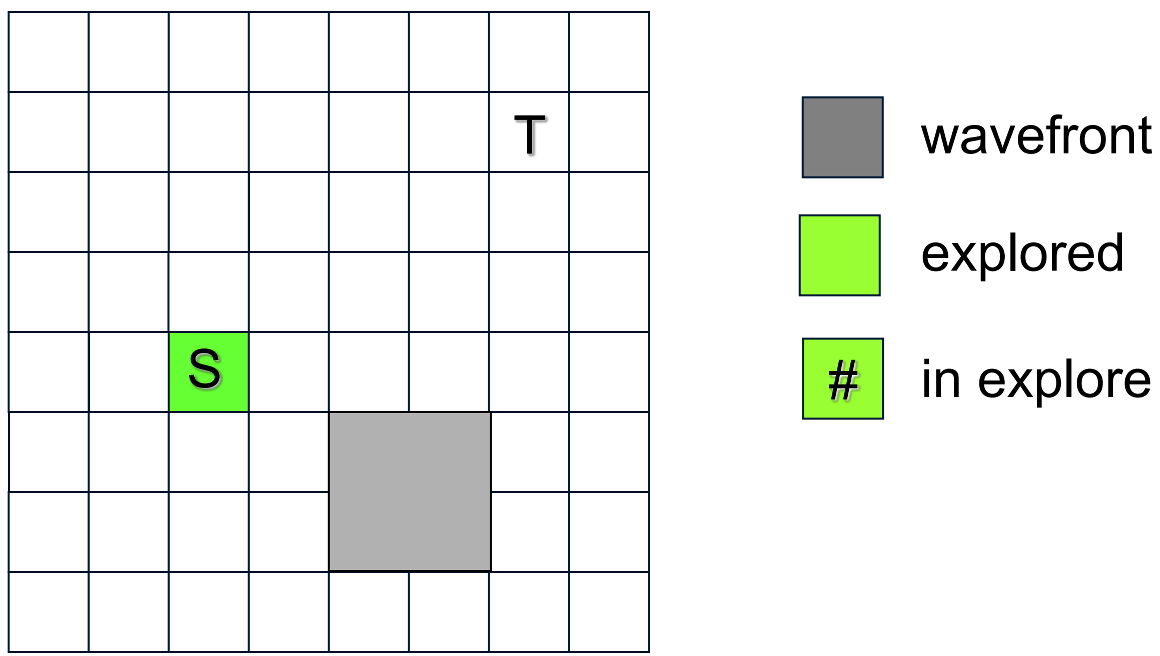


A*-Search Routing

- Maze search is also called blind search since it searches the routing region in a blind way.
- A*-search is also called the best-first search
  - uses function $f(x) = g(x) + h(x)$ to evaluate the cost of a path x
    - $g(x)$: the cost from the source to the current node of x
    - $h(x)$: the estimated cost from the current node of x to the target
- A*-search first searches the route that is most likely to lead towards the target.
  - BFS is a special case of A*-search where $h(x) = 0$ for all x
- Good property:
  - if $h(x)$ is admissible (never overestimates the actual cost from the current node to the target), then A*-search is optimal

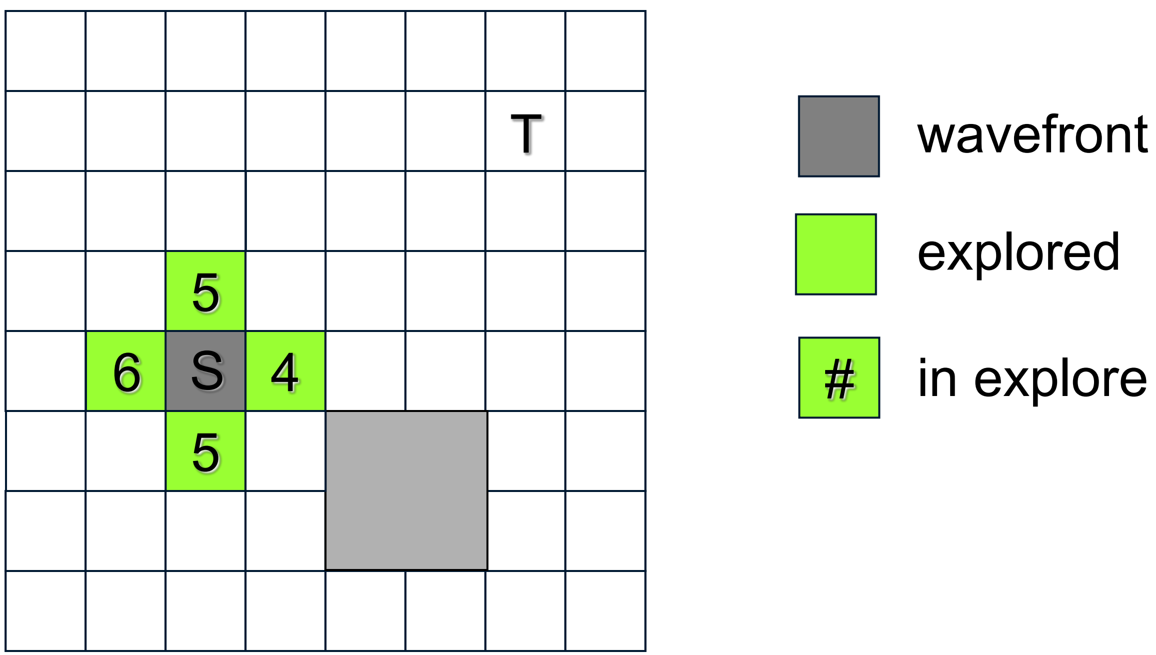
An Example of A*-Search Routing

- Cost function for a path x: $f(x) = g(x) + h(x)$
  - $g(x)$: the label from the source S to the current node of x (*i.e.*, the lable used in maze routing)
  - $h(x)$: $\max(\text{dist}_x(T, x), \text{dist}_y(T, x))$



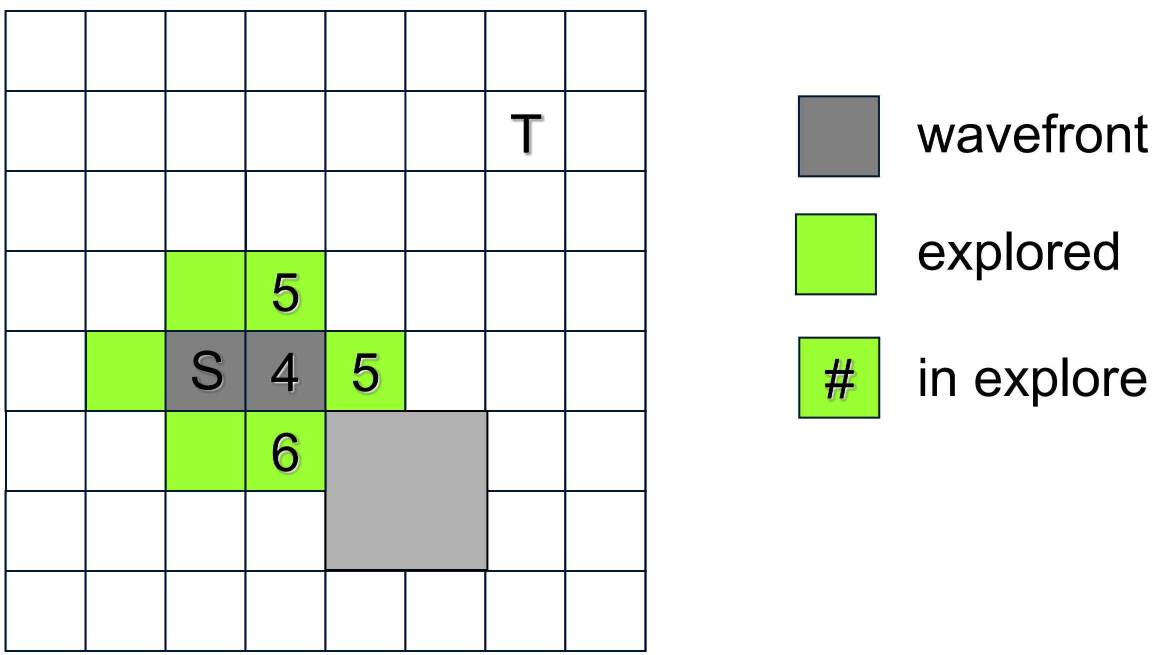
An Example of A*-Search Routing

- Cost function for a path x: $f(x) = g(x) + h(x)$
  - $g(x)$: the label from the source S to the current node of x (*i.e.*, the lable used in maze routing)
  - $h(x)$: $\max(\text{dist}_x(T, x), \text{dist}_y(T, x))$



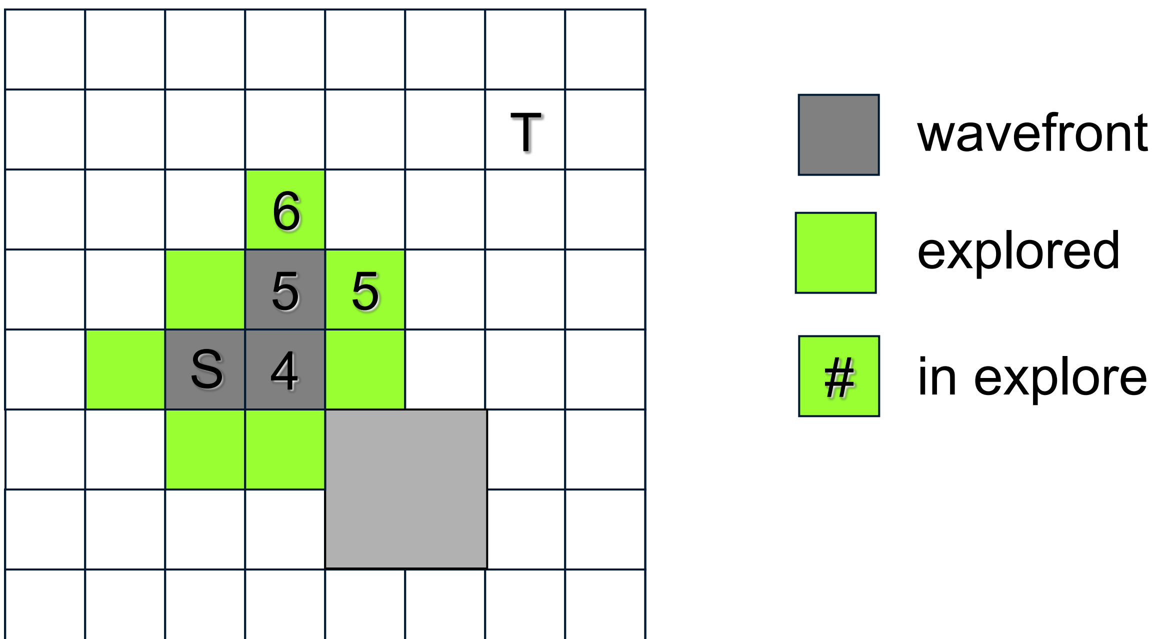
An Example of A*-Search Routing

- Cost function for a path x: $f(x) = g(x) + h(x)$
  - $g(x)$: the label from the source S to the current node of x (*i.e.*, the lable used in maze routing)
  - $h(x)$: $\max(\mathrm{dist}_x(T, x), \mathrm{dist}_y(T, x))$



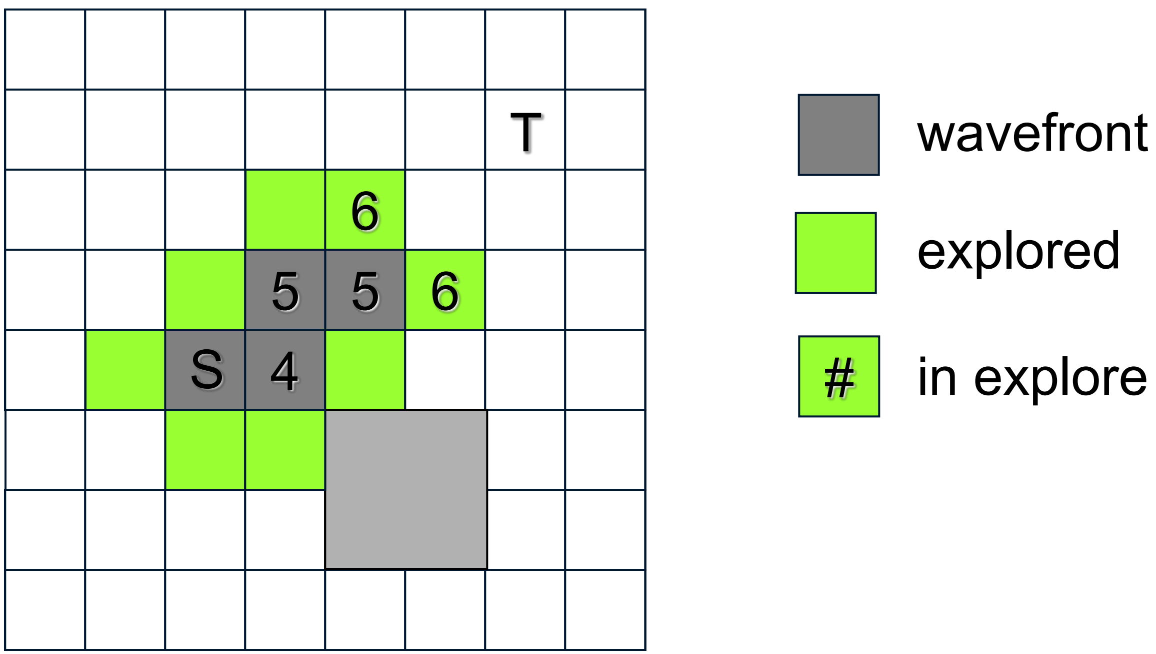
An Example of A*-Search Routing

- Cost function for a path x: $f(x) = g(x) + h(x)$
  - $g(x)$: the label from the source S to the current node of x (*i.e.*, the lable used in maze routing)
  - $h(x)$: $\max(\mathrm{dist}_x(T, x), \mathrm{dist}_y(T, x))$



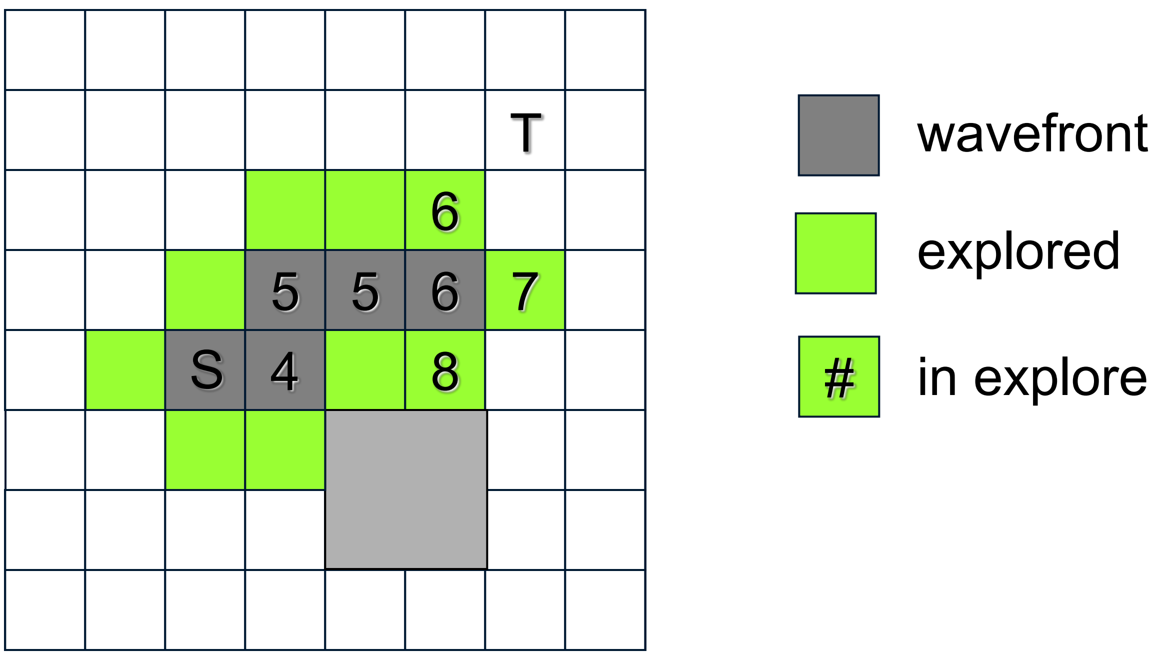
An Example of A*-Search Routing

- Cost function for a path x: $f(x) = g(x) + h(x)$
  - $g(x)$: the label from the source S to the current node of x (*i.e.*, the lable used in maze routing)
  - $h(x)$: $\max(\mathrm{dist}_x(T, x), \mathrm{dist}_y(T, x))$



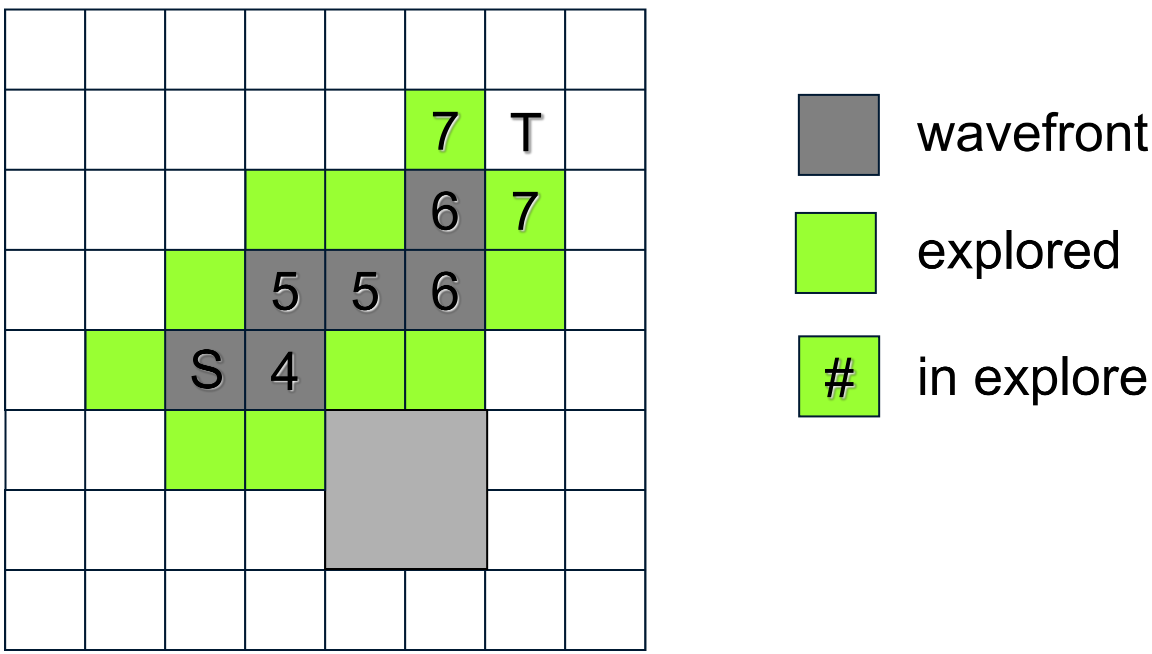
An Example of A*-Search Routing

- Cost function for a path x: $f(x) = g(x) + h(x)$
  - $g(x)$: the label from the source S to the current node of x (*i.e.*, the lable used in maze routing)
  - $h(x)$: $\max(\mathrm{dist}_x(T, x), \mathrm{dist}_y(T, x))$



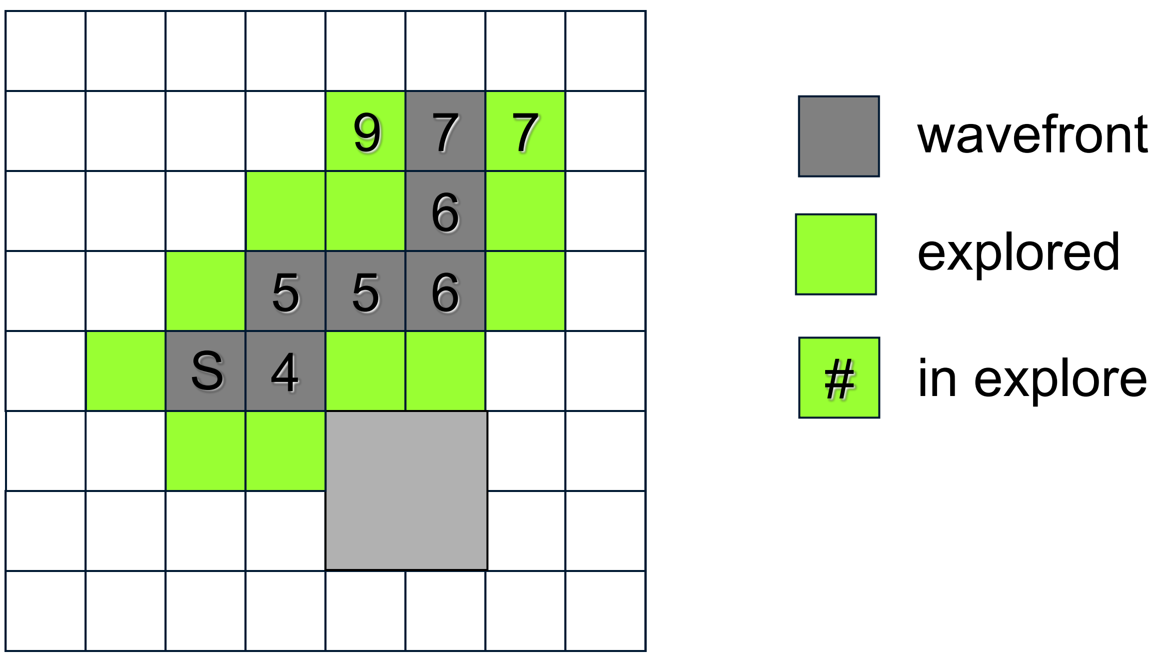
An Example of A*-Search Routing

- Cost function for a path x: $f(x) = g(x) + h(x)$
  - $g(x)$: the label from the source S to the current node of x (*i.e.*, the lable used in maze routing)
  - $h(x)$: $\max(\text{dist}_x(T, x), \text{dist}_y(T, x))$



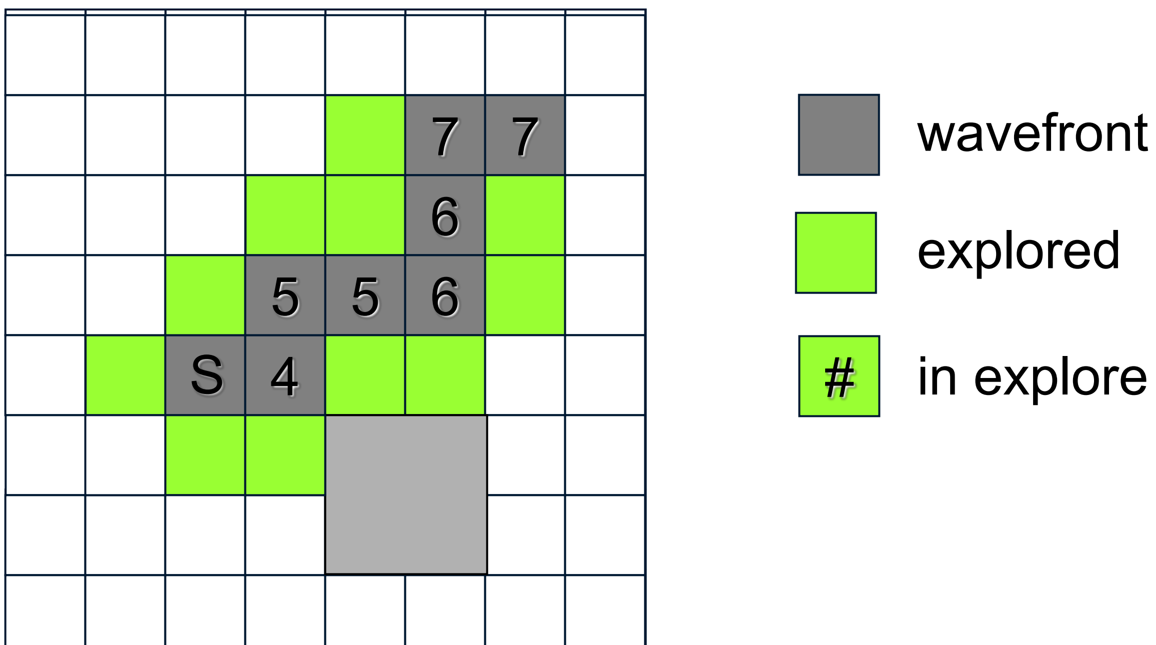
An Example of A*-Search Routing

- Cost function for a path x: $f(x) = g(x) + h(x)$
  - $g(x)$: the label from the source S to the current node of x (*i.e.*, the lable used in maze routing)
  - $h(x)$: $\max(\mathrm{dist}_x(T, x), \mathrm{dist}_y(T, x))$



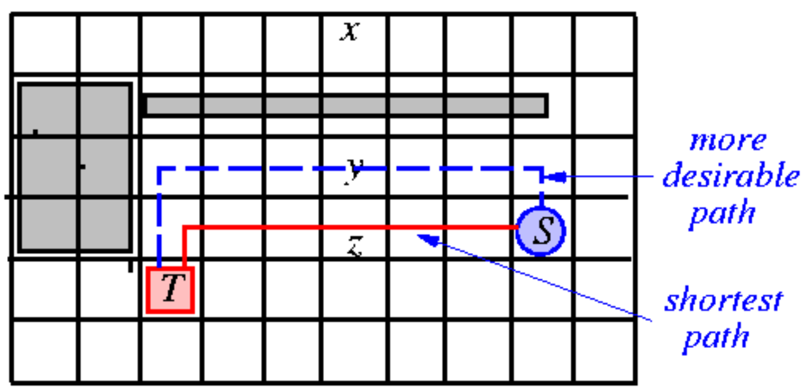
An Example of A*-Search Routing

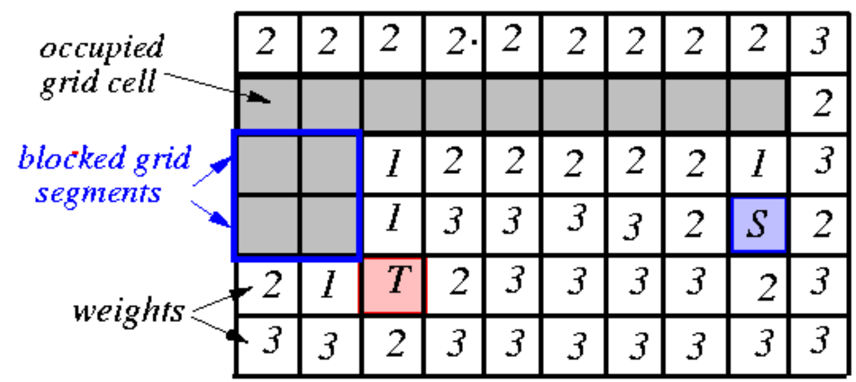
- Cost function for a path x: $f(x) = g(x) + h(x)$
  - $g(x)$: the label from the source S to the current node of x (*i.e.*, the lable used in maze routing)
  - $h(x)$: $\max(\text{dist}_x(T, x), \text{dist}_y(T, x))$



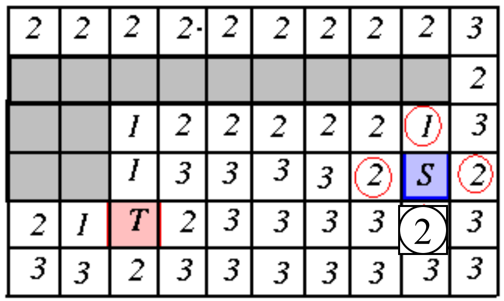
Routing on a Weighted Grid

- Motivation: finding more desirable paths
- *weight*(*grid cell*) = # *of unblocked grid cell segments* - 1

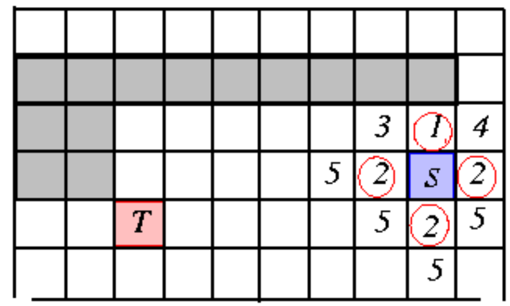




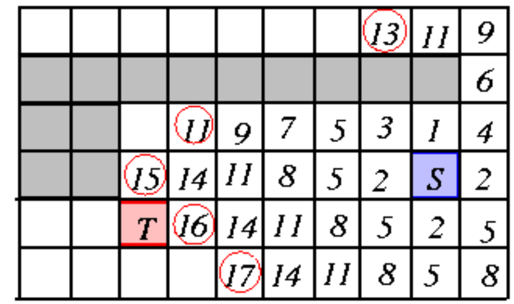
A Routing Example on a Weighted Grid

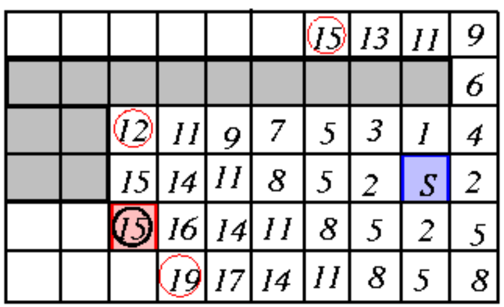


initialize cell weights

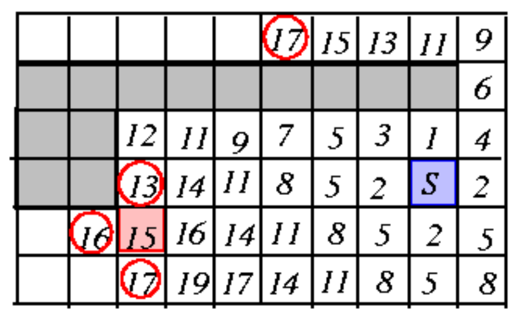


wave propagation

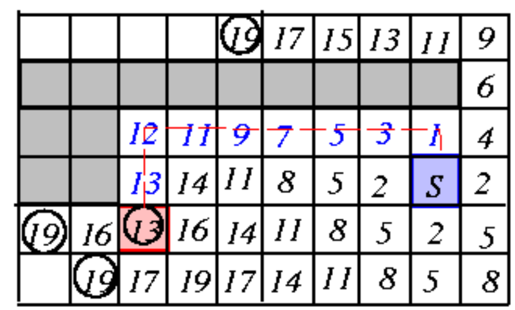




first wave reaches the target



finding other paths

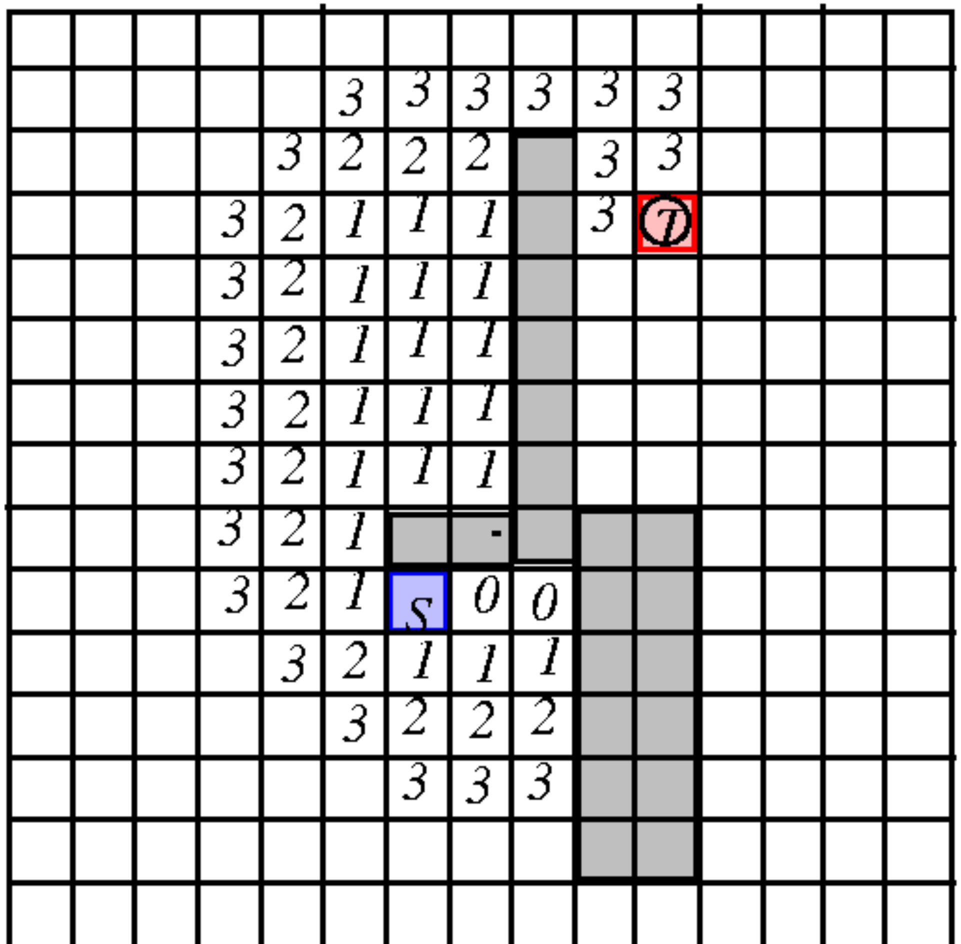


min−cost path found

Hadlock's Algorithm

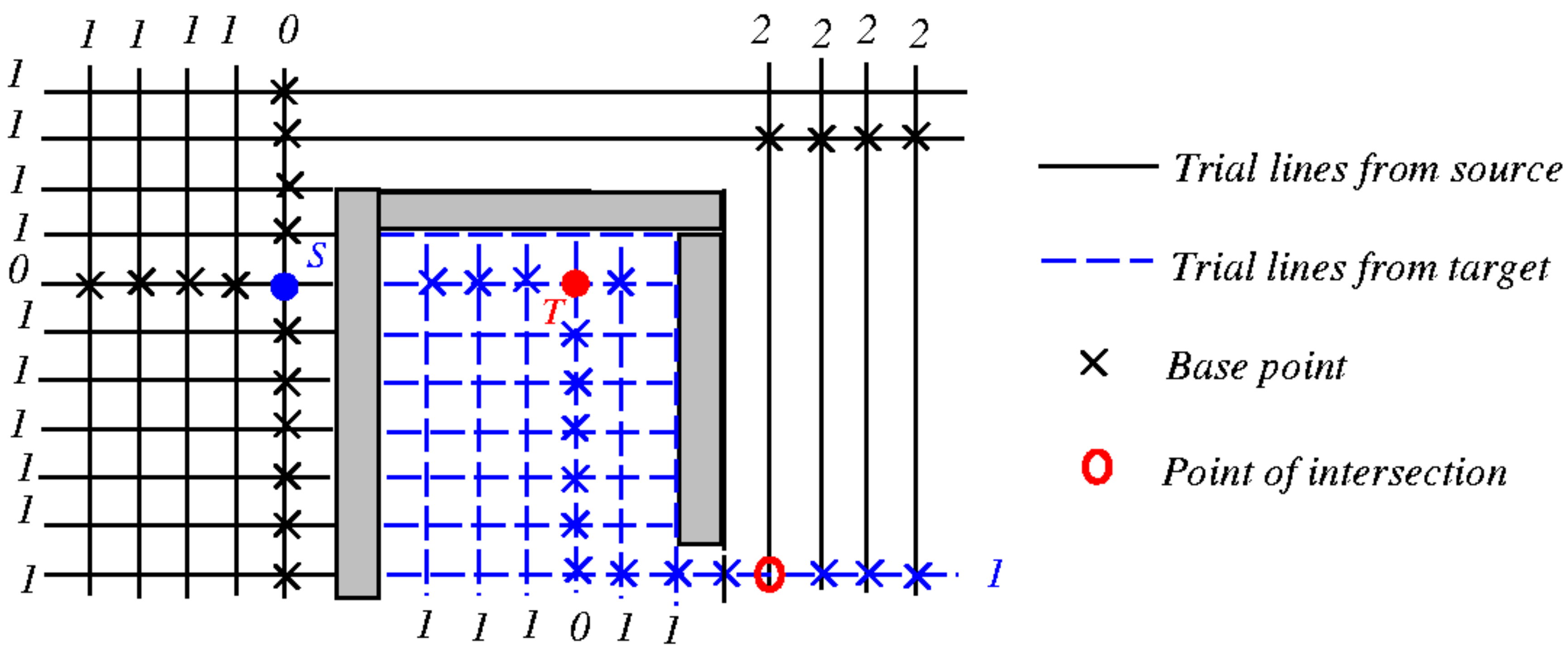
- Hadlock, "A shortest path algorithm for grid graph," *Networks*, 1977.
- Uses detour number (instead of labeling wavefront in Lee's router)
  - Detour number, $d(P)$: # of grid cells directed **away from** its target on path P.
  - $MD(S,T)$: the Manhattan distance between S and T.
  - Path length of P, $l(P)$: $l(P) = MD(S,T) + 2d(P)$.
  - $MD(S,T)$ fixed! $\Rightarrow$ Minimize $d(P)$ to find the shortest path.
  - For any cell labeled i, label its adjacent unblocked cells **away from** T $i + 1$; label i otherwise.
- Time and space complexities: $O(MN)$, but substantially reduces the # of searched cells.
- Finds the shortest path between S and T.

Hadlock's Algorithm(cont'd)



Mikami-Tabuchi's Algorithm

- Mikami & Tabuchi, "A computer program for optimal routing of printed circuit connectors," *IFIP*, H47, 1968.
- Every grid point is an escape point.



Hightower's Algorithm

- Hightower, "A solution to line-routing problem on the continuous plane," DAC-69.
- A single escape point on each line segment.
- If a line parallels to the blocked cells, the escape point is placed just past the endpoint of the segment.