Algorithms for graph search:

A* and Multiobjective A*

The simplest problem in graph search considers the problem of **finding a path** (a series of consequent edges) **from a given start node to a destination node**.

Dijkstra's algorithm:

Used for **single objective** search in graphs. It is similar to Breadth-first search, except it keeps track of the weight of edges in the case of a **weighted graph**. It always tries to continue the path with that is the shortest at the time.

A-star Algorithm:

An extension of Dijkstra's algorithm for single objective problems, that uses a heuristic function.

The **heuristic function provides an estimate of the distance from each node to the destination node**. This estimate is taken into account when finding the best node to explore further.

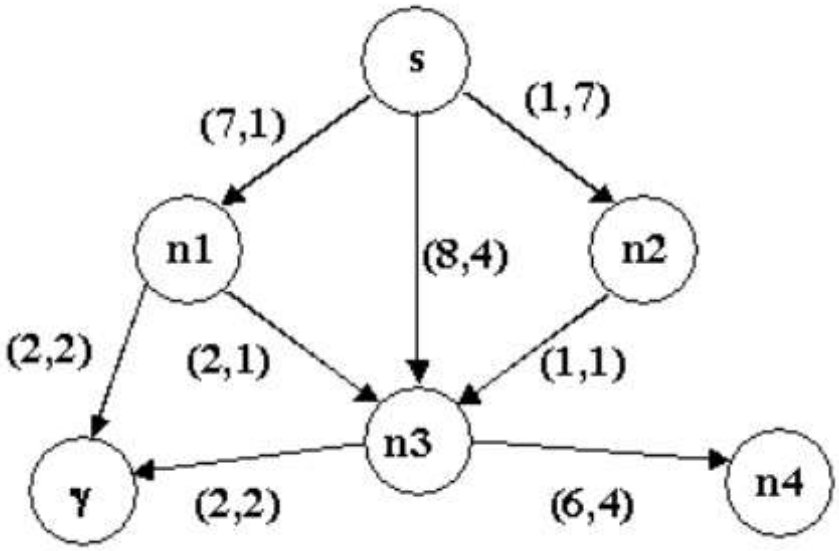
To be more precise, **the sum of the shortest path found so far to some intermediate node and the estimate from that intermediate node to the destination node** is used to evaluate how promising the given partial path is. The partial path with the lowest value gets explored further first.

As long as the heuristic function is overestimating the distance, the best solution will be found (in finite graphs).

An **example** for heuristic function for routing on a map would be the straight line distance.

The performance depends a lot on the heuristic function. With the wrong choice A* can be slower than Dijkstra's algorithm.

Multiobjective A-star Algorithm:



There were multiple attempts to extend the A* algorithm to multiobjective shortest path problems. Here, the goal is to find all Pareto optimal solutions, which is generally more than one path. Therefore, keeping track of all partial paths with promising lengths is more challanging.

The approach NAMOA* Mandow, Perez de la Cruz, 2005 (New Approach to Multiobjective A* Search) is the most efficient according to the literature, in general. The difference to the previous approach lies in the path-selection as opposed to node-selection. This means that the best (most promising) partial path is being further explored in each iteration and not all partial paths that end at the most promising node.

Pseudo code:

Table 1: A new path expansion algorithm for multiobjective A* search.

1. CREATE:
   - An acyclic search graph SG rooted in s.
   - List of alternatives, $OPEN = \{(s, \vec{g}_s, F(s, \vec{g}_s))\}$.
   - Two empty sets, $GOALN, COSTS$.
2. CHECK TERMINATION. If $OPEN$ is empty, then backtrack in SG from the nodes in $GOALN$ and return the set of solution paths with costs in $COSTS$.
3. PATH SELECTION. Select an alternative $(n, \vec{g}_n, F)$ from OPEN with $\vec{f} \in F$ non-dominated in $OPEN$, i.e.

   $$\forall (n', \vec{g}_{n'}, F') \in OPEN \quad \nexists \vec{f}' \in F' \quad | \quad \vec{f}' \prec \vec{f} \quad (3)$$

   Delete $(n, \vec{g}_n, F)$ from $OPEN$, and move $\vec{g}_n$ from $G_{op}(n)$ to $G_{cl}(n)$.
4. SOLUTION RECORDING. If $n \in \Gamma$, then
   - Include n in $GOALN$ and $\vec{g}_n$ in $COSTS$.
   - Eliminate from $OPEN$ all alternatives (x, g_x, F_x) such that all vectors in F_x are dominated by $\vec{g}_n$ (FILTERING).
   - Go back to step 2
5. PATH EXPANSION: If $n \notin \Gamma$, then

   For all successors nodes m of n that do not produce cycles in SG do:

   (a) Calculate the cost of the new path found to m: $\vec{g}_m = \vec{g}_n + \vec{c}(n, m)$.

   (b) If m is a new node
      i. Calculate $F_m = F(m, \vec{g}_m)$ filtering estimates dominated by $COSTS$.
      ii. If F_m is not empty, put $(m, \vec{g}_m, F_m)$ in $OPEN$, and put $\vec{g}_m$ in $G_{op}(m)$ labelling a pointer to n.
      iii. Go to step 2.

   else (m is not a new node), in case
   - $\vec{g}_m \in G_{op}(m)$ or $\vec{g}_m \in G_{cl}(m)$: label with $\vec{g}_m$ a pointer to n, and go to step 2.
   - If $\vec{g}_m$ is non-dominated by any cost vectors in $G_{op}(m) \cup G_{cl}(m)$ (a path to m with new cost has been found), then :
      i. Eliminate from $G_{op}(m)$ and $G_{cl}(m)$ vectors dominated by $\vec{g}_m$
      ii. Calculate $F_m = F(m, \vec{g}_m)$ filtering estimates dominated by $COSTS$.
      iii. If F_m is not empty, put $(m, \vec{g}_m, F_m)$ in $OPEN$, and put $\vec{g}_m$ in $G_{op}(m)$ labelling a pointer to n.
      iv. Go to step 2.
   - Otherwise: go to step 2.