Pseudocode for URLearning algorithms

Brandon Malone

May 10, 2017

Abstract

This document describes the algorithms implemented in the URLearning software package. The document *does not* necessarily describe the inner behavior of the solvers, which often contain optimizations to improve performance.

1 A* Search Algorithm

The A* search algorithm uses a best-first expansion policy to explore the order graph. Algorithm 1 gives a high-level overview of this search strategy. The primary operations are (immediate) duplicate detection and maintaining the priority queue.

In practice, the `contains` operations are implemented using a single hash table which contains all generated nodes. Flags indicate whether the nodes are in the *open* or *closed* lists.

Further, (pointers to) nodes in the *open* list are also maintained in a priority queue, which is implemented on top of a standard heap. The `update` operation leverages the observation that node priorities only ever improve; if a duplicate is worse, it is simply ignored.

Algorithm 1 A* Search Algorithm

function ASTAR(sparse parent graphs with $BestScore(\cdot)$, admissible heuristic h)
  $start \leftarrow \emptyset$
  $Score(start) \leftarrow 0$
  push$(open, start, h(start)$
  while len$(open) > 0$ **do**
    $\mathbf{U} \leftarrow$pop$(open)$
    if $\mathbf{U}$ is goal **then** ▷ A shortest path is found
      $\mathcal{N}^* \leftarrow$ construct a network from the shortest path
      return $\mathcal{N}^*$
    put$(closed, \mathbf{U})$
    for each $X \in \mathbf{V} \setminus \mathbf{U}$ **do** ▷ Generate successors
      $\mathbf{U}' \leftarrow \mathbf{U} \cup \{X\}$
      $g \leftarrow BestScore(X, \mathbf{U}) + Score(\mathbf{U})$
      if contains$(closed, \mathbf{U}')$ **then** ▷ Closed list DD
        if $g < Score(\mathbf{U}')$ **then** ▷ reopen node
          remove$(closed, \mathbf{U}')$
          push $(open, \mathbf{U}', g + h(\mathbf{U}'))$
          $Score(\mathbf{U}') \leftarrow g$
      else if contains$(open, \mathbf{U}')$ **then** ▷ Open list DD
        if $g < Score(\mathbf{U}')$ **then** ▷ better path
          update$(open, \mathbf{U}', g + h(\mathbf{U}'))$
          $Score(\mathbf{U}') \leftarrow g$
      else ▷ New node
        push $(open, \mathbf{U}', g + h(\mathbf{U}'))$
        $Score(\mathbf{U}') \leftarrow g$