Алгоритм A*

Тимофей Фролов

07.12.2022

История создания

- Оригинальная статья - июль 1968 года
- Название ”A Formal Basis for the Heuristic Determination of Minimum Cost Paths”
- Авторы: Peter E. Hart, Nils J. Nilsson, Bertram Raphael

Используемые обозначения

- Множество вершин $\{n_i\}$
- Множество направленных рёбер $\{e_{ij}\}$, ребру e_{ij} отвечает вес c_{ij}
- Функция $\Gamma : n_i \to \{(n_j, c_{ij})\}$ - по вершине получаем всех потомков, с весами соответствующих рёбер
- Подграф $G_n \subset \{n_i\}$ множество вершин, достижимых из $n \in \{n_i\}$
- Функция $h(n_i, n_j)$ - минимальная длина пути из вершины n_i в вершину n_j

Постановка задачи

- Собственно, задача:
  - ”Стартовая” вершина $s \in \{n_i\}$
  - Множество ”целевых” вершин $T \subset G_s$
  - $h(n) := \min_{t \in T} h(n, t)$
  - Цель - найти $h(s)$ и путь, при котором оно достигается
- Дополнительные требования, чтобы можно было использовать A*
  - $\exists \delta > 0 : \forall i, j : c_{ij} > \delta$
  - Известна некоторая дополнительная информация о природе графа (объясню позднее)

Идея алгоритма

- Введём $g(n) = h(s, n)$ - минимальная длина пути из вершины s в вершину n
- Введём $f(n)$ - длина оптимального пути из s в $t \in T$, проходящего через n
  - $f(n) = g(n) + h(n)$
  - Выгодно брать точки с наименьшим $f(n)$ из всех имеющихся.
  - $f(n) = f(s)\ \forall n$, лежащей на оптимальном пути
  - $f(n) > f(s)\ \forall n$, не лежащей на оптимальном пути
  - Вычисление $f(n)$ - ???
- Вводим $\hat{f}(n)$ - аппроксимацию функции $f(n)$.
  - $\hat{f}(n) = \hat{g}(n) + \hat{h}(n)$
    - $\hat{g}(n)$ - расстояние минимального найденного пути до вершины n
    - $\hat{(h)}(n)$ - функция, которую мы вводим из дополнительных знаний о природе вершин и рёбер
    - $\forall n \in G : \hat{h}(n) \leq h(n)$
    - $\forall m, n \in G : h(m, n) + \hat{h}(n) >= \hat{h}(m)$

Собственно, алгоритм

1. Помечаем s ”открытой” , вычисляем $\hat{f}(s)$
2. Выбираем открытую вершину n с наименьшим значением $\hat{f}$ (в случае, если таких несколько, выбираем вершину с наименьшим значением $\hat{h}$)
3. Если $n \in T$, n - искомая вершина. Строим обратный путь до s, завершаем исполнение алгоритма
4. Иначе помечаем n ”закрытой” , помечаем всех не закрытых потомков n открытыми, вычисляем для них значение $\hat{f}$,, возвращаемся к пункту 2

Псевдокод

```
1: procedure AStar(s, T)
2:     open ← {s}
3:     closed ← ∅
4:     g[s] ← 0
5:     f[s] ← g[s] + h(s)
6:     while open ≠ ∅ do
7:         n ← min{open} // comparing by f
8:         if n ∈ T then
9:             break;
10:        end if
11:        open.pop(n)
12:        closed.push(n)
13:        for (e, c) ∈ Γ(n) do
14:            if e ∉ closed then
15:                open.push(e)
16:                g[e] ← g[n] + c
17:                f[e] ← g[e] + h(e)
18:                prev[e] ← n
19:            end if
20:        end for
21:    end while
22:    path = []
23:    while n ≠ s do
24:        path.pushleft(n)
25:        n ← prev[n]
26:    end while
27:    return path
28: end procedure
```

Пример работы

- Тык

Свойства алгоритма A*

- Алгоритм является допустимым
- Алгоритм является оптимальным
- Асимптотическая сложность
  - Временная сложность $O(|V|)$ (где V - множество рёбер)
    - Полиноминальная, если $|\hat{h}(x) - h(x)| = O(\log h(x))$
  - Объём используемой памяти $O(|E|)$ - храним информацию о каждой посещённой вершине
  - В случае $\hat{h}(n) = 0 \ \forall n$ A* эквивалентен алгоритму Дейкстры