用动态规划求解路径规划问题

陆浩旗

May 4, 2022

Abstract

本文将路径规划问题化为 TSP 问题，在运用动态规划精确求解。

1 问题简述

在我们的 APP 中有一项功能为路线规划功能，即用户输入要去打卡的地点，软件自动规划所需的路径。最初的版本为依次经过用户输入的地点，后续迭代中想到也许可以实现一个类似旅行商问题（TSP）的简单求解，即帮助用户规划如何走才能走最短的路程经过所有给定点。

2 模型抽象

用户输入点集 V，通过猎鹰 API 接口可以得出两点间的最短规划路径，即我们可以得到一张 n 个点的完全图，$d_{i,j}$ 表示 i 与 j 的距离。

在我们的 APP 中有一项功能为路线规划功能，即用户输入要去打卡的地点，软件自动规划所需的路径。最初的版本为依次经过用户输入的地点，后续迭代中想到也许可以实现一个类似旅行商问题（TSP）的简单求解，即帮助用户规划如何走才能走最短的路程经过所有给定点。

3 动态规划模型建立

3.1 阶段与状态

我们将用户每到达一个地点作为阶段划分的依据。设阶段的状态为 $State$，其包含两个要素：一是用户已经完成打卡的地点；二是用户所在的地点。当上述两个要素确定时，用户的状态也唯一确定下来。我们可以用集合 S_k 表示第一个要素，其包含用户到达第 k 个地点时已经访问的所有打卡点。

3.2 状态转移方程

设阶段指标函数为 $f(i, S_k)$，它表示当用户到达打卡点 i 且对应已经完成打卡的点集为 S_k 时的总路程。于是我们可以得到状态转移方程

$$f(j, S_{k+1}) = min(f(i, S_k) + d_{(i,j)}), (i \in S_k) \tag{1}$$

即第 k+1 个点为 j，枚举第 k 个点为 i，从而进行状态转移。通过对状态和地点进行遍历，我们可以得到所有可能的状态下用户的最短路程。

假如用户还规定最后需要在某一地点 j 结束，还有

$$g(S_k) = min(f(i, S_k) + d_{(i,j)}), (i \in S_k) \tag{2}$$

其中函数 $g(S_k)$ 表示用户打卡 S_k 中的所有打卡点后回到结束点的最短路程。

3.3 一些优化技巧

对于打卡点集合 S_k，由于每个打卡点都有且仅有“未被访问”和“已被访问”两种状态，我们可以分别用 0 和 1 来表示。于是, 集合 S_k 可以转化为一个 n 位的二进制数 (n 为点数)，该数的第 i 位数表示打卡点 i 的访问状态。比如，用户已经完成打卡的点集为 $S_k = \{5, 1, 2\}$，则对应的二进制数可表示为 10011。这种二进制表示方法便于与位运算结合使用，从而有利于提高编程求解的效率。优化后求答案是枚举状态的复杂度降为 O(2^N)，且可以用 for 循环方便的实现。

3.4 复杂度分析

共 N 个打卡点，2^N 种打卡点访问情况，状态复杂度（空间复杂度）为 $O(N2^N)$。用式 (1) 求解所有可能的状态下用户的最短路程 $f(State)$ 时，需要对 i, j 和 $State$ 进行遍历，对 i 和 j 复杂度均为 $O(N)$，对 $State$ 则为 $O(2^N)$，综合来看为 $O(N^2 2^N)$。用式 (2) 求解用户打卡 S_k 中的所有地点并且最终在结束点的最短路程 $g(S_k)$ 时, 需要对 i 和 $State$ 进行遍历，复杂度为 $O(N2^N)$，总复杂度 $O(N^2 2^N)$。

3.5 在 APP 中的应用

由上面的复杂度分析可知，DP 对于求解问题规模在 N<=20 以内的数据可以做到在 1s 内求解，在我们的 APP，限制于猎鹰 API 对于两点间规划距离的获取限制，我们将规划点数限制在 N<=7。