Sequnce队参赛算法介绍文档

队员：秦若愚（计05），张家炜（无95），张书源（无95）

一、整体思路

此赛题本质上就是找一个最优的布尔向量输入，从输入到输出的过程等效一个黑箱（引擎模拟），最终输出成绩以及整个过程的道路状况。所以最主要的部分就是求解黑盒优化。对于黑盒优化问题，由于不能通过梯度下降去寻找局部最优解，我们在一开始确定了使用**遗传算法**。

优化之前，我们先编写了一个脚本获得各种"地图类型"的数量，发现车道的数量是35000条，也就是黑盒的输入状态空间的大小为2^{35000}。需要搜索的空间非常大，而得到一次数据需要5min左右，所以尽可能的缩小搜索空间和选择遗传算法的起始点非常重要。我们的方法是利用"白盒信息"——《用户手册》内容和模拟结果traj.txt、time.txt。我们先处理map.json获得了所有和兴趣点相连的车道id，通过引擎模拟确定上述获得的车道不能被限流，确定的和兴趣点相连的车道有，最后需要搜索的空间大小可以缩减为。虽然缩减并不明显，但是在之后使用的随机算法中减少了无谓的尝试。如果从0输入开始遗传，可以预料到前期下降的速度会非常快，同时向很多方向都会有衰减，这样遗传一轮后剩余的样本会很多，导致前几轮遗传产生的输入数量会指数级上升。再考虑到一次模拟需要5min左右，这样的做法会在一开始耽误很长时间。在这里，我们利用了模拟结果，先将access.txt修改为空文件，然后开始模拟，根据结果从中选出车流量较大或者车速较慢的n条车道，然后将其限流，在次基础上进行下一轮模拟。通过这种方式，我们得到了一个限流车道较少就达到最终平均用时850s的输入（引擎更新前的结果，更新后没模拟过）。同时我们使用了纯随机搜索，搜索状态空间中限流车道数在50条之内的样本点。我们将白盒优化的结果和随机数据结合起来，获得遗传的种子，用限行方案模拟出的平均到达时间作为该限行方案的score。我们定义了限行方案的**泛化能力**：$\alpha = \frac{1}{lanes-number}$(其中lanes-number为当前限行方案的限行车道数)，用于衡量从某个输入开始遗传最终得到更好结果的能力。我们取score最大的100个方案中泛化能力最强的10个方案作为遗传算法的起点，开始我们的遗传算法。

综上，我们的整体算法思路一下分为四步：

1. 根据map.json获得数据规模
2. 利用《用户手册》得到和兴趣点相连的车道id，然后将其从可限流车道列表移除，以缩小搜索空间，获得允许限行的车道列表Acc。
3. 考虑到直接开始遗传会耽误大量时间，选择分析输出traj.txt和time.txt，然后根据分析结果限流，每一次输入都基于上一次模拟的输入和输出结果分析。即

$$L_n = L_{n-1} + R_{n-1}$$

其中L_n是第n次模拟的输入，R_n代表分析第n次模拟结果得到的k条要限流的较慢或者车流量较大的车道。

再随机搜索样本空间中50条以内的限行方案。取平均到达时间最快的100个方案，再取其中泛化能力α最大的10个限行方案。

4. 以第3步得到的输出作为起始点，用改进的遗传算法进行搜索。

（最终第3步中我们限流的是车流量较大的车道）

二、文件清单及算法细节解释

改进的“遗传算法”步骤

在本题中，传统遗传算法遇到的最大问题是，状态数会随遗传的轮数快速上升，当遗传2-3轮后，每轮仿真需要的时间可能会需要1-2天，这对持续优化造成了很大的困难。因此在算力有限的情况下，我们使用了改进的遗传算法：

1.选取10个初始状态，作为遗传的种子集S。设初始状态为$S_i \in S\ (1 \le i \le 10)$，$S_i$对应的限行车道集合为$L_i$，限行车道的数量为$N_i$，$N_i = |L_i|$，这组限行车道对应的平均到达时间为 t_i。

2.对每个初始状态S_i，进行n_i次遗传的尝试。对第 j 次尝试，采取以下两种遗传方法：

Option a）随机取$Acc \backslash L_i$中的k_i^j条车道$L_{add_i}^j$，使用$L_{new_i}^j = L_i + L_{add_i}^j$作为新的限行方案。

Option b）随机取L_i中的m_i^j条车道$L_{rm_i}^j$，使用$L_{new_i}^j = L_i \backslash L_{rm_i}^j$作为新的限行方案。

按照概率p_i使用Option a）,按照概率$(1-p_i)$使用Option b）。得到S_i派生的n_i个状态$\{S_{new_i}^j\}$及其对应的限行方案$\{L_{new_i}^j\}$，$1 \le j \le n_i$。

3.对所有的派生状态进行仿真测试，得到平均到达时间。

4.将本轮遗传的初始状态和派生状态作为下一轮遗传的种子集合，即

$$S_{\text{总}} = \{S_i\}_{i=1}^{10} + \sum_{i=1}^{10} \{S_{new_i}^j\}_{j=1}^{n_i}$$

取$S_{\text{总}}$中score最大的20个限行方案中泛化能力α最大的10个方案，作为新的初始状态集S，再次进行步骤2。

参数的选择和更新

实验发现，泛化能力与限行方案中限制的车道数量成反比。因此在最开始挑选初始状态的时候，在score相近的情况下优先考虑限行车道少的限行方案进行遗传。

对一个初始状态，随机增加的车道和减少的车道数量与原本的限行车道数量成负相关。具体参数选择如下：

N_i	<50	50~200	200~350	>350
k_i^j	$random(20,100)$	$random(10,50)$	$random(5,30)$	$random(2,20)$
m_i^j	$random(\frac{N_i}{4},\frac{N_i}{3})$	$random(\frac{N_i}{10},\frac{N_i}{4})$	$random(\frac{N_i}{20},\frac{1N_i}{5})$	$random(1,\frac{N_i}{10})$

对一个初始状态使用Option a）的概率p_i与原本的先行车道数量成负相关，即车道数越多，将更大概率地使用减少车道地策略，具体的p_i选择如下：

N_i	<50	50~200	200~350	>350
Option A)的概率 p_i	0.9	0.8	0.6	0.3
Option b)的概率 $1-p_i$	0.1	0.2	0.4	0.7

对限行方案的奖励与惩罚

根据限行方案派生出的方案的score，可以对该限行方案的泛化能力进行实时反馈。对限行方案S_i，定义奖励因子rwd_i等于S_i派生出的限行方案排在$S_{\text{总}}$前20名的数量。再进行下一轮遗传时，根据rwd_i确定下一轮 S_i 的派生方案数n_i。具体实现为

$$n_i = \begin{cases} n_0 & \text{若}S_i\text{为新派生出的状态} \\ n_0 - \triangle n + rwd_i * n_{rwd} & \text{若}S_i\text{为遗传种子集中的状态} \end{cases}$$

由于算力受限，取$n_0 = 5, \triangle n = 2, n_{rwd} = 2$。

优化的基本过程

在使用改进的遗传算法对该问题的优化时，在状态集S上有以下两个阶段交替出现：

（1）$\min\limits_{S_i \in S}\{t_i\}$快速下降，而$Var\{t_i\}$上升。

（2）$\min\limits_{S_i \in S}\{t_i\}$变化不大，$E\{t_i\}$和$Var\{t_i\}$下降。

阶段一是泛化能力强的状态迅速派生出更优的新状态的过程。而阶段二则是群体性能的提升和集中，以及改善初始状态集的泛化能力的过程。这是因为在算法过程中，增加车道的分支往往更容易得到更优的限行方案，但当限行方案中车道数过多时，泛化能力会减弱。因此需要去除该限行方案中存在的车道冗余，才能使$\min\limits_{S_i \in S}\{t_i\}$持续下降。而减少车道的尝试往往无法使平均到达时间有大幅度的提升，因此表现为了最小时间变动不大，而均值和方差逐渐下降。

在阶段(1)中影响性能的参数主要为k_i^j，在阶段(2)中影响性能的参数主要为m_i^j。在同一个阶段中，保持k_i^j和m_i^j不变，对它们的更新在优化方案经历了一轮状态(1)和状态(2)之后进行。

参数p_i影响一次尝试中加车道和减车道的概率，一般阶段(1)中的p_i要大于阶段(2)。