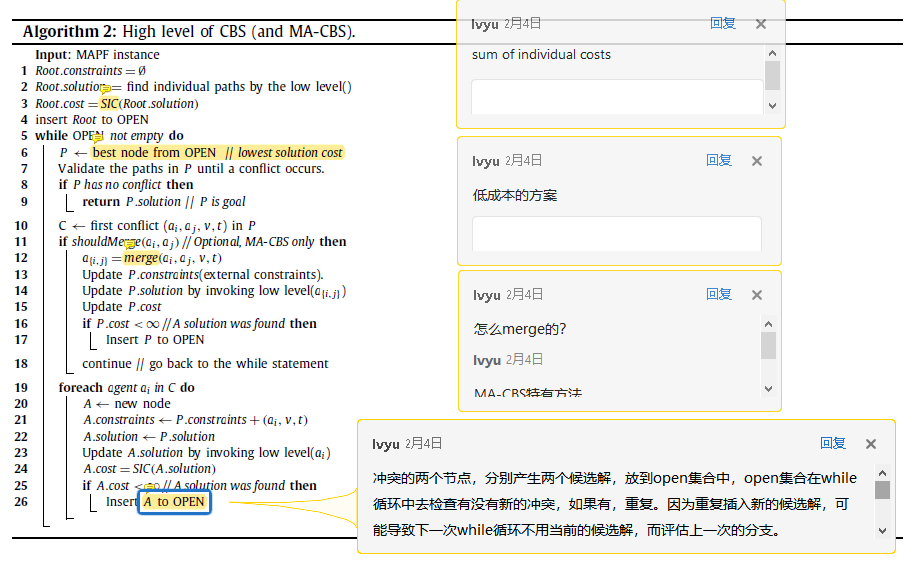
### cbs算法分析

cbs（conflict-based search algorithm）算法是当前最先进的MAPF算法之一。算法是由Guni Sharon等在2014年底发表。算法伪码参考下图，分为高层和底层两段，高层负责冲突的检测，新方案的必选，底层利用A\*算法等产生新的解决方案。



cbs发现第一个冲突，比如agent1和agent2之后，立刻对冲突的两个agent分别构建两个分支，agent1分支为agent1避让分支，agent2分支为agent2避让分支。以agent1分支为例，每个分支只对第一个冲突进行一次避让操作，也即设置冲突导致的新约束，调用底层A\*算法，让agent1对第一个冲突的点或边进行重新寻路，绕过冲突点，该分支的其它agent的路径均不改变，这样形成新的一个MAPF候选解，存入open集合，等待下一次重新进行冲突检测和解更新。每次冲突会有两个分支，其中优先会选择低成本的分支进行下轮的路径搜索。

底层的A\*算法是Peter Hart在1968年发表。其是Dijkstra算法的增强版本，在工程和游戏领域被广泛使用。

A\*算法有三个核心点：

（1）open，close

open放的是等待评估的节点，算法初始运行，为起点，之后从open中选出最小代价点作为当前节点，开始探索地图。

close放从open中移除的已经评估过的点。

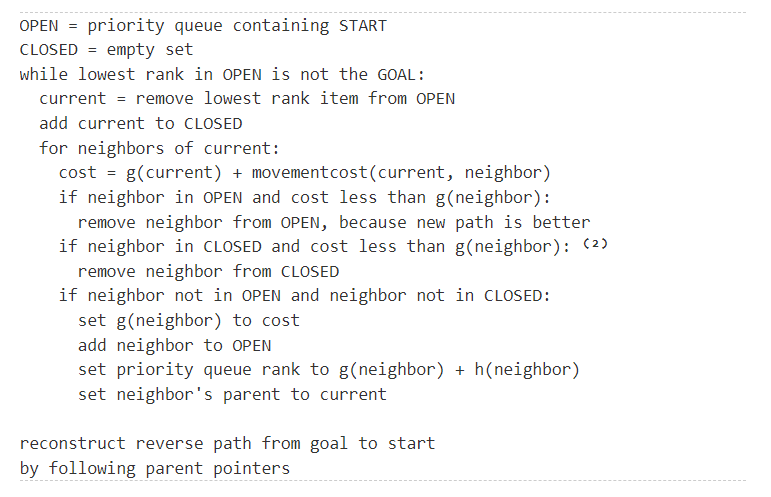
（2）f=g+h

当前评估点的所有邻居点（可达）被选出，计算其f值，并对所有邻点计算f值，根据邻点为探索过，待探索，未探索的三类不同情况，进行处置。

（3）对当前点的邻居点的处置

如果邻点是探索过（close），但探索时的f值的g的部份高于当前点的g值与到邻点的代价之和，则更新邻点，并将其重新放到open集合供下一次评估；如果邻点是等待探索（open），但探索时的f值的g的部份高于当前点的g值与到邻点的代价之和，则更新邻点；如果邻点未探索，则计算邻点的g，f，并把邻点放进open集合。

全过程需要保持当前点与邻点的父子关系，即当前点是前驱，邻点是后续，从时间是父子关系（先有父后有子），便于找到终点后，回溯整条路径。



基础的cbs算法存在的问题是：对于某些场景非常耗时，在agent数量较多，或地图太大的应用场景无法使用。虽然它能够得到完备的最优解，但搜索的时间太长。

以我们项目的地图场景，存在相当多的单向车道长走廊，如果采用cbs不优化，必然导致无法及时求解出最优解（主流配置PC机器上运行，32x32的地图耗时超过1小时（23步的长单通道）。

静态规划需要优化到实时水平。

