HSEA-第一次作业

丁豪 人工智能学院 181220010
181220010@smail.nju.edu.cn

摘　要：此次作业为使用启发式搜索算法玩双人pacman游戏，我采用的是带有时间限制的A-star搜索算法，在每个等级均获得了平均每轮超过1w分的成绩。

关键词：启发式搜索，A-star算法，gvgai框架

任务一：理解游戏

通过简单的试玩，我发现游戏中获得加分的方法主要有：

- 吃金币、水果、绿宝石。分数较少，≤ 10。
- 吃完绿宝石变为powered状态后，触碰ghost。分数较多，一次+40分。

同时游戏失败的方式只有一种：

- 在非powered状态下触碰ghost，终止整个游戏。

因此算法设计的思路主要分为以下三个部分：

- 防止游戏失败，即在not powered状态下避免直接接触ghost。
- 在游戏继续进行的过程中，尽量多的触发power kill ghost +40奖励。
- 在无法触发powered状态（绿宝石吃完）时，尽量多的收集地图上的resources，来提高最终分数。

任务二：实现搜索算法

算法分为**搜索+决策**两部分。搜索部分实现了基础A-star框架，即综合考虑路径代价以及期望代价之和$f(s)=g(s)+h(s)$来选取节点进行扩展，维护“frontier”与"searched"两个列表，一直搜索直到时间不够。决策部分，从"frontier"或"searched"列表中挑选$f(s)$最小的节点（当前已经发现的最佳状态），并回溯到根节点，返回对应的第一个动作。

- 我们为A-star节点专门建一个类，在其中存储如下属性，并在找到更优到达此节点的路径时通过封装的`update_father`方法来更改其父节点、父节点动作、g(s)的值。

```
public StateObservationMulti state; // 节点对应的状态
public AStarNode father; // 父节点
public Types.ACTIONS act; // 父节点如何操作转移到此
public double f_value,g_value,h_value; // 总目标、路径代价、启发式函数

public void update_father(AStarNode fnode,Types.ACTIONS a,double g,double h)
```

- 搜索部分的**路径代价g(s)**定义为每一步增加一个常数，以在其他条件相同情况下会选择最短路径。

```
double stepCost=100;
...
// 新节点的g值，为父节点g+stepCost
AStarNode tmpNode = new AStarNode(ob, pick, a, pick.g_value + stepCost, h(ob));
```

- 搜索部分的**启发式函数h(s)**由几个分量构成，每个分量乘以相应的权重值，最终结果为这些分量的加权和。

1. 游戏失败惩罚（无权重，直接返回一个很大的惩罚值）（存在一个小bug：会导致avatar不吃最后一块金币，来避免游戏结束，但影响只有1分，可以忽略不计）

```
if (ob.isGameOver()) return Double.MAX_VALUE / 2; // 除以二防止溢出
```

2. score：游戏得分本身是很重要的启发

```
int score_cost = -1000;
...
value += score_cost * ob.getGameScore(id);
```

3. 墙体的改变：经验表明，不移动任何墙体也可以赢得游戏，且改变墙体容易卡住部分路径或资源，使最终可得的分数上限降低。因此我们鼓励不要移动墙体，给每一块被移动的墙体一个惩罚。

```
int dif_cost = 500; // 为score_cost/2，使得不会为了吃金币而穿墙
...
value += dif_cost*diff;
```

4. 如果处于powered状态，则优先寻找最近的ghost来接近，此cost反映了离最近ghost的距离。这里额外添加了一个常数power_const，以抵消突然增加的dis_cost，防止ghost因为h(s)增加而不去吃绿宝石。此外在实验过程中，发现如果两个avatar追同一个ghost可能卡死，因此给他们分别分配不同的追踪任务。

```
int ghost_dis_cost = 1000; // 比dif_cost大，使得追猎ghost时可以穿墙
int power_const = -12000; // 1000*12，用于奖励进入powered状态，防止h(s)反常上升
int hunt_npc = 2; // 每个Avatar追猎的ghost数量，分别追[:hunt_npc]与[-hunt_npc:]，防止抢夺同一个ghost导致死锁
...
value += ghost_dis_cost * mindis;
value += power_const;
```

5. 如果处于非powered状态，则优先寻找最近的资源。资源列表的大小随着剩余资源种类的数量变化，因此这里直接获取资源列表中最后一种资源。

```
int resource_dis_cost = 100; // 不用很大，只需要在没有方向的时候提供一个启发
int index = ob.getImmovablePositions().length-1;
...
value += resource_dis_cost * mindis;
```

6. 随机增加一个小的数字，使相同f_value状态总可以挑选出“最优”，而不具有系统性归纳偏好（如总是选择序号小的）

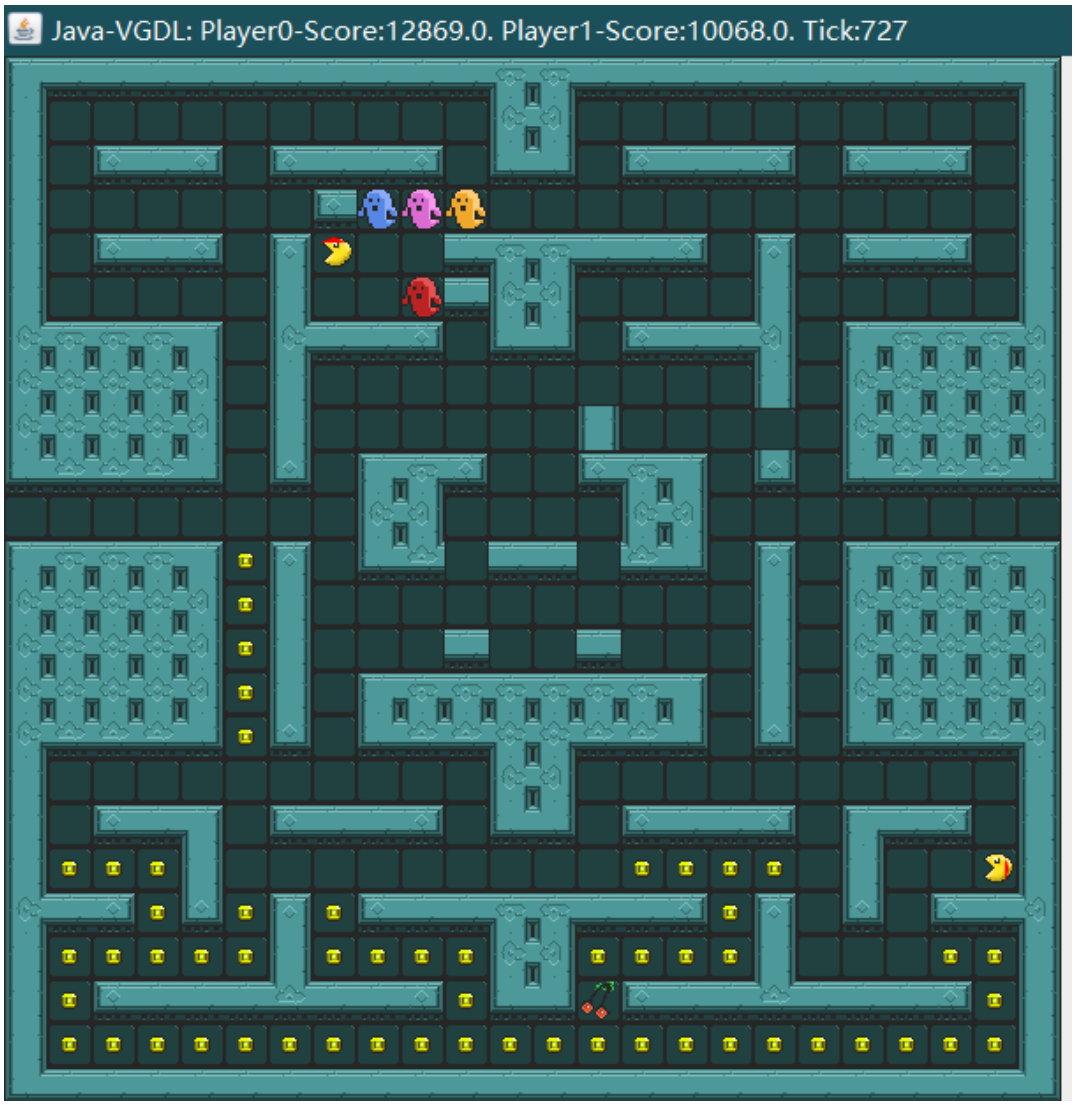
```
return rand.nextInt(10) + value;
```

- 决策过程按照frontier > searched > random的顺序挑选最优动作，如果frontier非空则在其中找到最优状态并回溯到根节点的动作返回，其次如果searched非空则在其中找到最优状态回溯到根节点的动作返回。最后如果两个列表都为空，则返回随机动作。

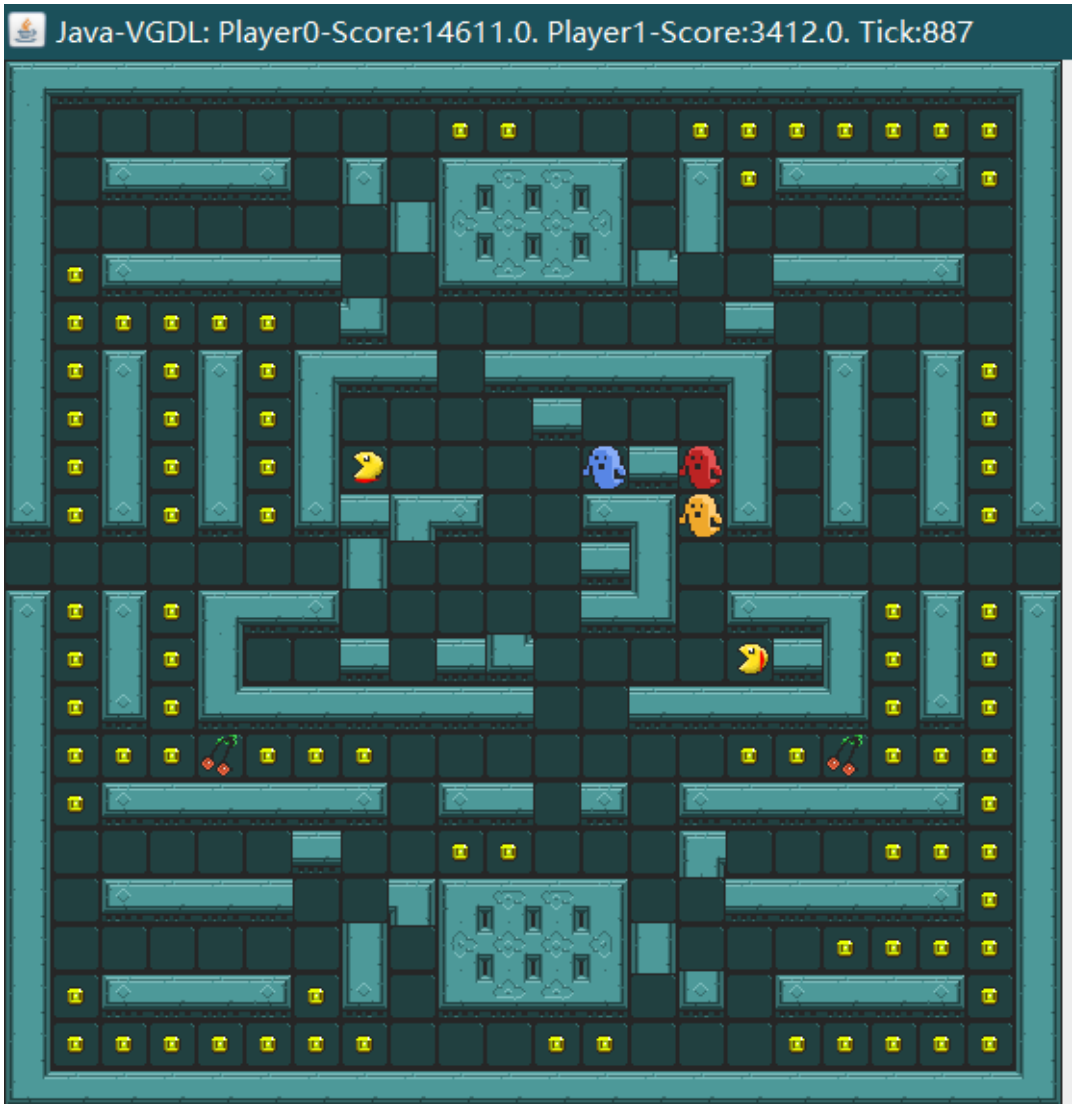
```
Types.ACTIONS best_act;
if (frontier.size()>0) {
    best_act = getBestAct(frontier);
}
else if (searched.size()>0){
    best_act = getBestAct(searched);
}
else {
    best_act = actions.get(rand.nextInt(actions.size()));
    System.out.println(id+" RandomAct "+best_act.name());
}
return best_act;
```

任务三：调整算法

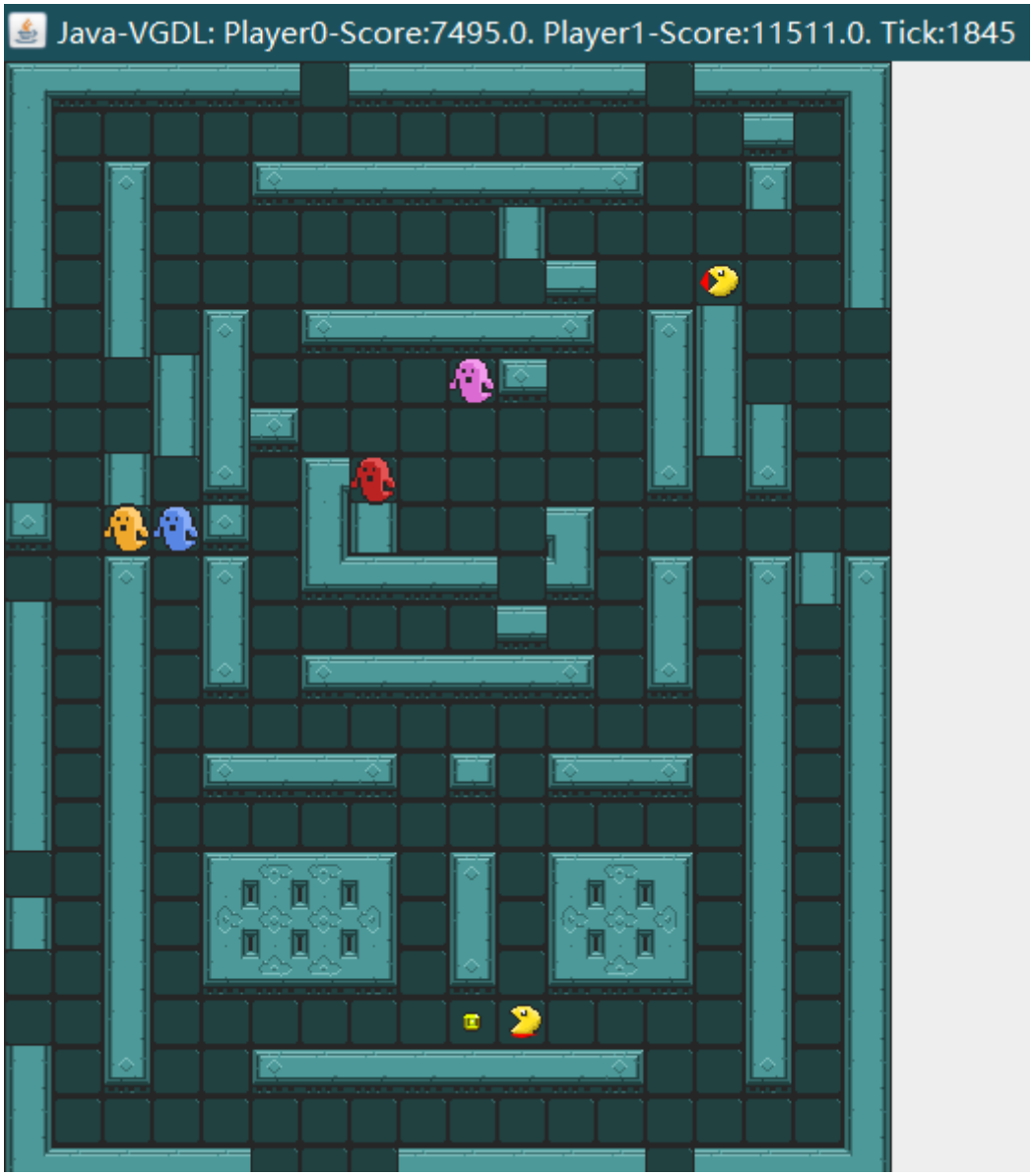
- 经过不断总结新的启发式函数和改进已有启发式函数的实现，以及对于已有启发式函数分量的不断调参，最终取得了不错的性能。具体启发式函数的细节见上一节描述。
- 算法使用A-star搜索，其展开节点数量与耗时成正相关。经过调试，最终在确定每步100ms的情况下（ACTION_TIME = 100）大约可以进行不到10轮迭代，展开不到40个节点。最终实验分数结果如下：（由于游戏内置ghost的随机数与全局种子独立，因此如果不能复现结果，请多尝试几次即可）
    - lv0-平均 20034，最高 22973



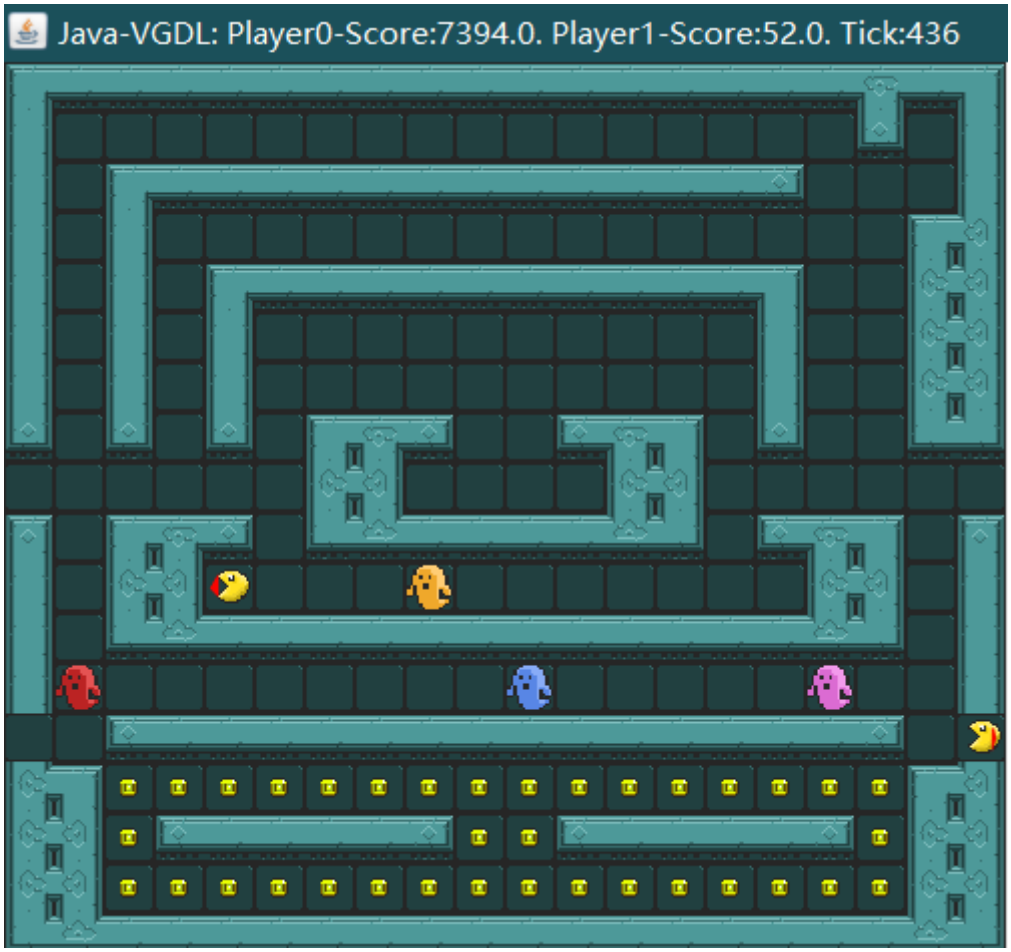
- lv1-平均 12363，最高 18023



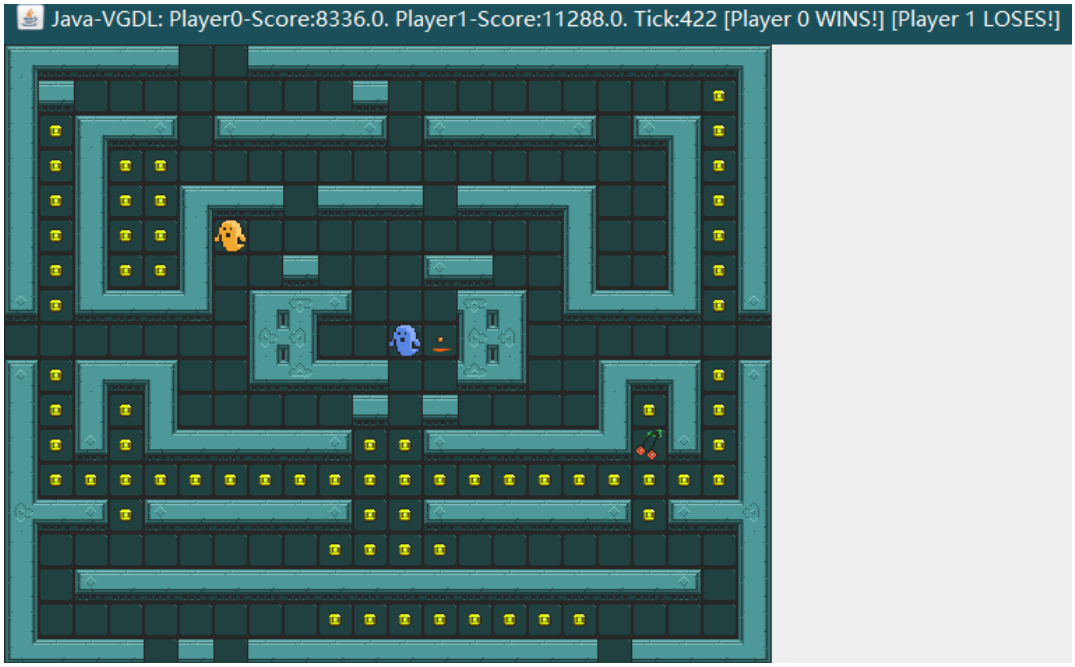
- lv2-平均 17050，最高 19006



- lv3-平均 4235，最高 7446



- lv4-平均 16460，最高 19624



总结

在本次实验中，我实现了基于A-star的搜索和基于启发式函数的策略选择，获得了较为理想的分数。然而，仍然存在不能完全吃光全图资源、部分情况下两个Avatar争抢同一个资源、ghost随机化导致模拟状态中不会死亡的动作导致真实游戏死亡、仅剩最后一块资源时不会主动获取以结束游戏等问题。这些问题的产生很大一部分原因受制于启发式搜索的限制，人为总结的启发式函数，总是只能对应于某些特定情况，当时空环境发生变化时无法有效刻画游戏中不同状态下动作选择的优劣。未来也许会尝试使用强化学习来解决此类问题。

鸣谢

特别感谢薛轲学长在本次实验中给予我的帮助，通过和他的讨论加深了我对游戏的理解，以及解答了游戏中固定随机种子后每次游戏结果仍然存在不确定性的问题。