## 1 简单baseline计划：

## 完成标准的读入，并获取相应的信息。工作台状态，机器人状态。

1. 处理模块。
   * 1. 按类别获取工作台的index。
     2. 机器人只买1，2，3工作台材料，卖给4，5，6，9工作台。
     3. 对于每个机器人来说，每个帧数找性价比最高的1，2，3，卖给最近的4，5，6，9。
     4. 机器人途中路线完全直线。不计算加速减速带来的时间损失。
     5. 不考虑碰撞与时间系数，带来价值的损失。
2. 机器人行为响应模块，每个机器人包含五个行为。4\*5=20

Notes：性价比计算：获利价格/路程。

不理解：系统的demo示例为什么存在，到达目标点的减速行为（没必要，最大速度\*一帧时间，即6\*0.02=0.12m，可认为过程是连续的）。应该是因为机器人进行了交易行为，要去下一个目标点，进行转向导致的。如果前后目标点方向一致，应该不会存在转向行为。

最坏的情况就是前后两个目标点，方向相反。导致机器人从-6到6的加速过程，需要（1帧加速0.3m/s，12/0.3=40帧）。

消耗性工作台可一直接收产品（周期一帧）。

图表, 折线图

描述已自动生成

**角速度=夹角**

**角速度=PI\*tanh（夹角）**

**对于如下工作台状态**

1. **2 3 2 4(12)**

文本

描述已自动生成

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 2 | 4**(12)** |
| Robot1 |  |  |  |  |  |
| Robot2 |  |  |  |  |  |

首先，给原材料1，2，3的生产工作台赋值10的优先级

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 2 | 4**(12)** |
| Robot1 | 10 | 10 | 10 | 10 |  |
| Robot2 | 10 | 10 | 10 | 10 |  |

**检测是否存在 类似于（12） 的节点，给所有2的优先级加10，其他同理**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 2 | 4 |
| Robot1 | 10 | 20 | 10 | 20 | 0 |
| Robot2 | 10 | 20 | 10 | 20 | 0 |

**如果机器人手上有2，则2产品优先级-1，避免同时拿一样的**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 2 | 4 |
| Robot1 | 10 | 19 | 10 | 19 | 0 |
| Robot2 | 10 | 19 | 10 | 19 | 0 |

**对于如下工作台状态**

1. **2 3 2 4(12)**

**优先级表如下**

**（4没生产出来，优先级为0）一个机器人手上有2**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 2 | 4 |
| Robot1 | 10 | 19 | 10 | 19 | 0 |
| Robot2 | 10 | 19 | 10 | 19 | 0 |

**当4生产好了4(12)的状态不存在，优先级改为如下**

**1 2 3 2 4(可拿)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 2 | 4 |
| Robot1 | 10 | 10 | 10 | 10 | 20 |
| Robot2 | 10 | 10 | 10 | 10 | 20 |

再检测是否存在产品对应可卖节点，不可卖的产品优先级置0

如果4当前没有可买节点

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 2 | 4 |
| Robot1 | 10 | 10 | 10 | 10 | 0 |
| Robot2 | 10 | 10 | 10 | 10 | 0 |

对于所有robot，拿了以后就近卖，拿的优先级就一定程度避免了死锁发生，如果还是发生了死锁（当前机器人发现卖不出去），就命令机器人丢弃。

更新优先级表的时机：机器人发生了卖的操作或丢弃操作就更新

为什么每个机器人要维护自己的优先级：

避免robot1更改优先级，导致robot2靶工作站变化，走到一半掉头

每个机器人只有再自己手上没东西那一刻才会更新自己那一行的优先级