粒子群加速算法：

假设任务M，数量为m（按顺序可完全执行）；AGV（或RGV）为A，数量为a；则定义任务矩阵为：，其中i表示第i个任务，j表示第j个AGV，并拥有以下约束：

（1）

（2）

式1式2联合约束表述为每项任务仅能由一个AGV完成，并且每项任务必定会被执行。

原始粒子群算法公式：

（3）

（4）

式3为速度更新公式，式4为位置更变公式，其中w表示惯性权重；C1、C2通常表示单点跨度权重，可理解为学习率；Random为(0,1)的随机数；V为当前速度；表示更新后的速度；表示单个粒子的最优点；表示全局粒子的最优点；P表示单个粒子的当前位置；表示更新后的单个粒子位置。

由于本问题的离散性，粒子群加速算法对于该问题需要作出一些改动。首先明确以下几点：

1. 路径的确定完全取决于立体货架的主通路，定义为MainPath；以及电梯位置，定义为Elevator。因此路径的选择在算法的迭代中采取完全随机的策略。如此一来，即使是完全相同的任务-AGV配置也会产生不同的结果，一定程度上可以减轻算法陷入局部最优的情况。
2. 粒子群加速算法使用于任务调度，也即对于所有任务的分配AGV策略。
3. 对于任务M采取类似模拟退火算法的方式进行任务的震荡编排。预期随着总体迭代次数的增加，震荡变化越大或者越小，也即任务列表从一开始的完全顺序到一定程度的打乱，或反之。（暂未实现，接口已预留）

于是算法流程如下：

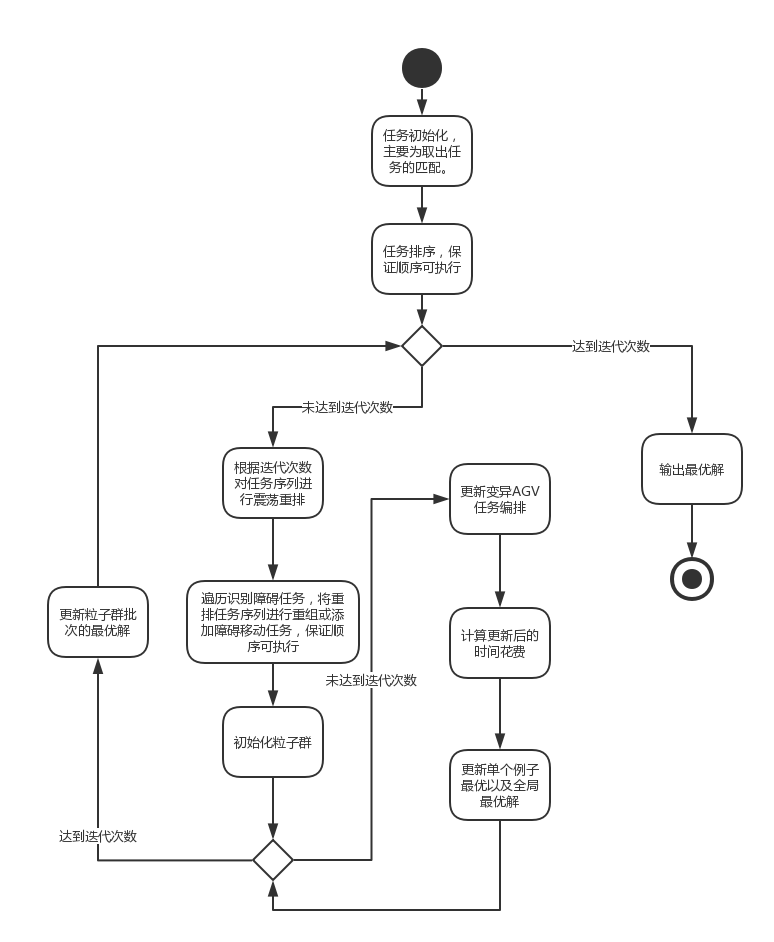


图1. 总体算法流程图

上图1中所示为总体的算法流程图，下面将进行详细的讲解。

最初的任务初始化主要操作是根据需要出库的出库单号进行整张地图的扫描，选取需要出库的货物的商品位置和信息，并对任务对象进行初始化，封装到一个列表中，这里定义为MissionList。

任务排序：对MissionList进行反复扫描，把出库任务封装进一个任务矩阵，定义为TakeOutMissionMatrix；把入库任务封装进一个入库任务列表SetInMissionList。此处有两种情况：

1. 第一次扫描能够有直接出库的商品，此时TakeOutMissionMatrix的第一个列表肯定为能够直接出库的商品；而之后的所有列表都是被阻碍的。
2. 所有TakeOutMissionMatrix的列表任务都是被阻碍的。

任务震荡重排：根据迭代次数调整震荡率，把TakeOutMissionMatrix和SetInMissionList的所有任务按照震荡率进行重排。默认情况下TakOutMissionMatrix的排序是按照顺序依次展开拼接到列表尾。

识别阻碍任务：有些任务是无法直接取出的，特别是在顺序完全打乱的情况下。因此需要把所有阻碍物作为任务提前，进行重排保证整个任务序列能够顺序可执行。此处有两种情况：

1. 阻碍物不是任务，添加额外的阻碍物移动任务。
2. 阻碍物是任务，直接把该任务提前。

初始化粒子群：上述完成重排的任务序列数量、AGV数量以及粒子数量，初始化粒子群。初始化中也先做一遍时间耗费计算，对单个粒子最优解以及单批次全局最优解进行初始化。

变异更新AGV任务编排：根据最开始的式1式2的约束，把原始的粒子群更变公式进行如下改造：

（5）

式5为更改之后的任务分配矩阵更新公式。的定义如上文所示，不再赘述。此处的w取值为1；C1、C2取值为2；Random为(-1,1)的随机数，主要是为了减轻所有粒子集中化而造成陷入局部最优的情况，因而采取距离感知但是方向未知的方式。去除了速度的概念，直接使用距离进行更新（实验效果，去除速度更能收敛到耗时较少的策略）。

但是式5更新之后的数值是不能直接使用的，还需要满足式1式2的约束，因此对式5进一步操作如下：

（6）

（7）

（8）

（9）

（10）

上式6、7联合主要是为了把所有第i项任务的分配情况都置为大于0的数；式8为第i项任务的分配情况总和；式9表示对第i项任务的分配情况进行重新赋值，赋值手段为前j个AGV的值的总和；式10表示对式9计算结果的归一化。

经过上式计算之后，所有AGV对于第i项任务的计算值就可转换为0~1的概率覆盖范围。对于第i项任务，若第j台AGV的覆盖范围越大，则分配到任务的概率越高。选取0~1的随机数，分配给对应的AGV以第i项任务。

先前保证任务序列顺序可执行就是为了防止分配完任务之后发生任务死锁。

计算时间花费：对M项任务的A个AGV进行无限循环遍历。使用基本累加值进行时间累加，此处定义为1。初始化时的每个任务的花费时间为-1。

1. 对每台AGV计算一个已完成任务的时间消耗值，为所有已完成任务的时间花费+等待时间。
2. 对未完成任务，若时间花费未计算，则计算时间花费并把附加任务（例如AGV的主路径更换、就位等等）添加到对应AGV的任务序列中。若完成时间花费计算，则计算当前时间累加值是否大于当前已完成任务的时间累加值+当前未完成任务的时间花费+当前未完成任务的等待时间。若大于，则标记当前未完成任务为已完成任务，否则继续下一轮循环。
3. 检查是否所有AGV的任务序列都是已完成状态，若存在未完成，则返回1继续循环。

更新最优解：得出当前一次迭代的解之后，比较单个粒子的最优解以及单批次粒子群最优解。若更优，则替换最优解。最终得出的最优解即为本次计算的最优解。目前未加入碰撞检测以及入库任务的计算和操作，还有任务序列震荡方法需要考虑。