**THU-70250403, Convex Optimization (Fall 2020)** **Homework: 12**

# Complete Coverage Path Planning

*Lecturer: Li Li* `li-li@tsinghua.edu.cn`

*Student:* 杨润钊 陈一帆 谢佳辰 严相杰 于铭瑞

## 1 项目描述

假设二维空间中存在一个尺寸为 $0.57m \times 0.7m$ 清洁车，如图1所示。

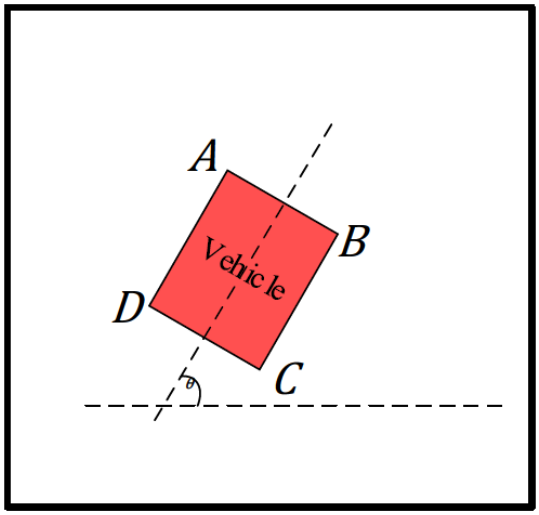


图 1: 垃圾车模型

现有图2所示的垃圾清扫场景，场景覆盖 $10m \times 10m$ 的矩形区域，数字编号的 14 个点表示垃圾的位置，字母编号的 4 个多边形表示实体障碍的位置。任务是为清洁车设计一条可以清除场景中所有垃圾的无碰撞且光滑的路径，并使得路径长度尽可能短。

## 2 数学建模

根据任务描述，这是一个典型的路径规划问题，一般来说路径规划是为了搜索代价最小的路径, 需要给出的条件包括：地图，起始地点，目标地点，损失函数。对应于本项目的任务，规划需求如下：

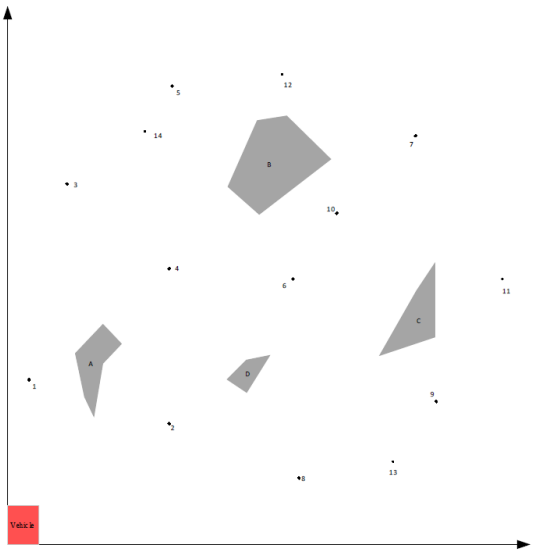


图 2: 场景描述

1）路径尽可能短；

2）清洁车覆盖所有的垃圾；

3）清洁车避开所有的障碍；

4）清洁车旋转角（图 1 的$\theta$）不超过 $0.2\pi$。

为了定量完成这个任务，我们先引入一些数学记号。路径 $\boldsymbol{z}$ 由一系列航点表示：

$$\boldsymbol{z} = \left[\boldsymbol{z}(0)^T, \boldsymbol{z}(1)^T, \ldots, \boldsymbol{z}(n)^T\right]^T, \tag{1}$$

其中 $\boldsymbol{z} = [x(k), y(k), \theta(k)]^T$ 表征第 $k$ 时刻清洁车的状态，$P(k) = [x(k), y(k)]^T, \theta(k)$ 分别表征清洁车的中心位置和方位角（如图1所示）。完成决策变量 $\boldsymbol{z}$ 的定义后，我们可以将“清洁车”要完成的任务建模为如下优化问题。

**目标函数**

为了满足最小化路径的要求，定义目标函数如下：

$$\min_{\boldsymbol{z}} J(\boldsymbol{z}) = \sum_{k=0}^{n-1} \sqrt{(x(k+1) - x(k))^2 + (y(k+1) - y(k))^2}. \tag{2}$$

**约束条件**

对于第 1 个约束，定义如下：

$$\forall m_i, \exists j, \text{ s.t.m } _i \in V(j), \tag{3}$$

其中，$m_i$ 为编号 $i$ 的垃圾位置，$V(j)$ 为清洁车在第 $j$ 采样时刻的覆盖区域。

对于第 2 个约束，矩形车在任意采样时刻与所有的多边形均无交集。换言之如果障碍是凸的，此时矩形和多边形可看作若干半平面的交，我们只需判断这些半平面构成的可行区域是否为空即可检查碰撞与否；对于非凸障碍，可以将多边形分割成若干凸子集或者利用非凸障碍的凸包近似来估计。

对于最后一个约束，定义 $\alpha(P(k)-P(k-1), P(k+1)-P(k))$ 为向量 $P(k)-P(k-1)$ 和 $P(k+1)-P(k)$ 的夹角，如图3所示，因此这个约束可以写作：

$$\alpha(P(k)-P(k-1), P(k+1)-P(k)) \leq \alpha_{\max}, \tag{4}$$

其中，$\alpha_{\max}$ 为最大允许转角。

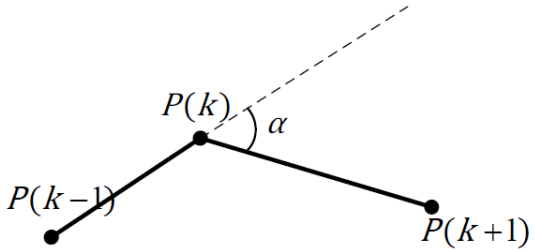


图 3: 清洁车的转向角度

## 3 实现细节

**约束条件 1**

在清洁车进行垃圾拾取时，定义拾取成功的条件是：清洁车在第 $j$ 采样时刻的覆盖区域 $V(j)$ 覆盖了垃圾点坐标。

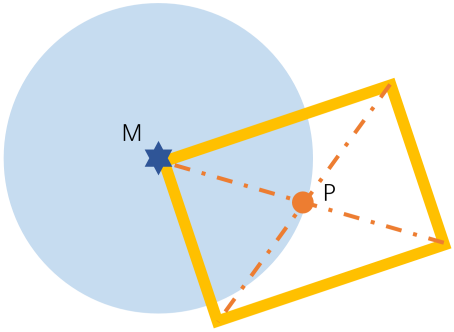


图 4: 清洁圆模型

首先对于问题进行简化。由于在第 $j$ 采样时刻的清洁车角度可以任意定义，因此定义以清洁车对角线为直径，以垃圾点为圆心的圆（后文简称为收集圆），如图4所示。因此当清洁车的中心坐标处于收集圆中时，可以通过旋转垃圾车，使其一角指向垃圾以实现拾取垃圾的要求。

因此针对于该问题的约束条件 1，只需要对于垃圾车的位置进行优化，保证其处于收集圆中，并根据垃圾车与垃圾相对位置确定清洁车方向角，即可完成收集。问题转化为，如何以最短的路径遍历所有清洁圆。

再进一步，可以将该问题分解成解清洁圆的路径顺序 $M(k)$ 与各个清洁圆的接触位置 $P(k)$。如果两个单位圆间没有障碍，其间距离小于两圆心连线距离，大于两圆心连线距离减清洁圆直径。因此可以的不同的路径顺序进行上下界估计，大大缩小其可能解的范围。再通过凸优化方式对于接触位置进行优化，以求得该遍历顺序下最短路径，如图5所示，并排除所以下界小于该最优解的遍历顺序。最后对于可能解的优化结果进行排序，即可选出最优解。

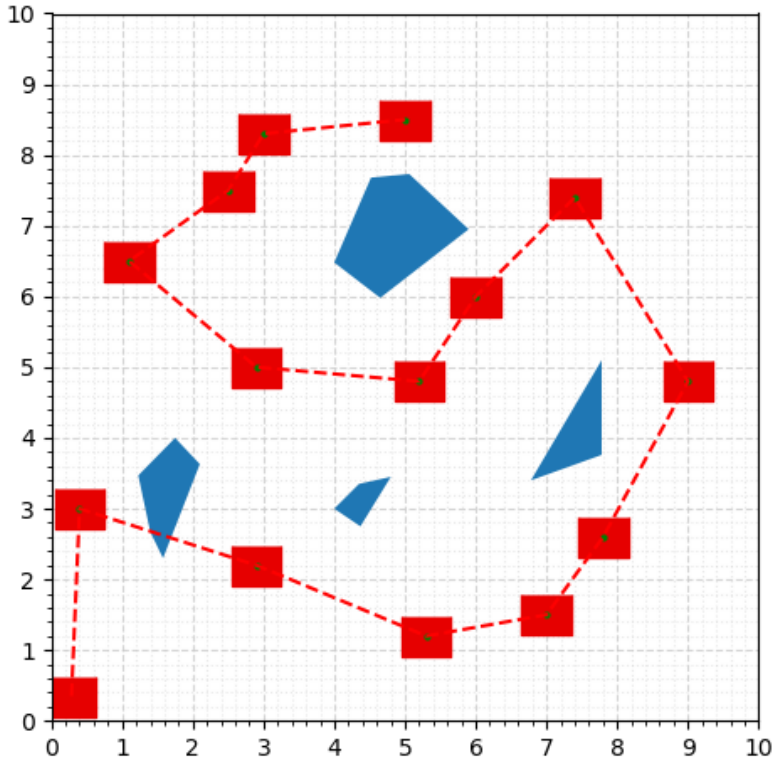


图 5: 路径遍历最优解

首先确定路径顺序。

以清洁圆圆心作为需要遍历中心位置。这是一个旅行推销员问题（Travelling salesman problem, TSP），该问题有大量次优解求解方法。但是由于本问题位置点较少较为简单，因此本文使用较为保守的遍历方式进行最优解搜索。首先计算清洁圆之间的距离。同时因为存在障碍物干扰，因此需要针对连线之间有障碍物的圆的距离进行适当矫正，并获得所有清洁圆之间的距离矩阵。进而对于所有路径方式进行遍历。

在遍历过程中为了加速运算，进行部分经验性约束。在搜索下一个路径点时，只遍历距当前点距离最短的 $n$ 点；同时如果下一个路径点的距离大于最近待搜路径点的 $k$ 倍时，直接舍弃。可以通过选择 $k$ 与 $n$ 改变遍历效果。

然后确定接触位置。

我们定义如下优化问题

$$\min_{\boldsymbol{z}} \ J(\boldsymbol{z}) = \sum_{k=0}^{n-1} \parallel P(k+1) - P(k) \parallel_2 \tag{5}$$

$$\begin{aligned} s.t. \ \parallel P(1) - M(1) \parallel_2 \leq \ d/2 \\ \cdots \\ \parallel P(n) - M(n) \parallel_2 \leq \ d/2 \end{aligned} \tag{6}$$

其中 $P(k)$ 为路径中第 k 个点，$M(k)$ 为以确定的遍历路径中第 k 个清洁圆心，d 为小车的对角线长度，即清洁圆大小。该问题为凸问题，可以使用凸优化方法求解，获得每个清洁圆的接触位置 $P(k)$。

**约束条件 2**

我们基于所谓的可视图路径规划算法解决原优化问题的避障约束，最原始的可视图法由 Lozano-Perez 和 Wesley 在 1979 年提出 [1]，是机器人领域全局运动规划的经典算法，应用到我们的场景中，将清洁车建模为质点，障碍物用多边形表示，设 $V_o$ 是所有障碍物的顶点构成的集合，将起始点 $S$、目标点 $G$ 和障碍物的顶点进行组合连接形成可视图，要求起始点和障碍物各顶点之间、目标点和障碍物各顶点之间以及各障碍物顶点与顶点之间的连线均不能穿越障碍物，即直线是“可视的”，给图中的边加上权值，即构成可视图 $G(V, E)$，其中点集 $V = V_o \bigcup \{S, G\}$，如图6所示，$O_1, O_2$ 为两个已知大小和位置的多边形障碍物，$E$ 为所有“可视边”的集合。由于任意两直线的顶点都是“可视”的，从当前位置沿着这些直线到达目标点的所有路径均是车的无碰路径。搜索最优路径的问题就转化为从起始点 $S$ 到目标点 $G$ 经过这些可视直线的最短距离问题。

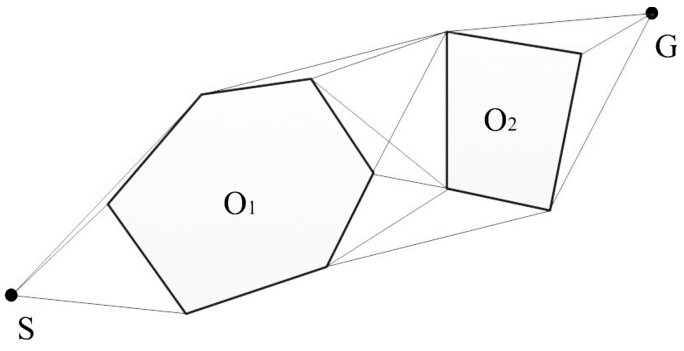


图 6: 可视图示意图

然而实际场景中，清洁车的尺寸不能忽略，我们对多边形障碍物的轮廓进行 padding 处理，填充大小为清

洁车的半短轴，并且多边形顶点附近通过增加采样点使得轮廓更加光滑，如图7所示，用清洁车质心位置代表车在某时刻的位置，多边形障碍物某一转角用 $\overline{P_1OP_2}$ 表示，填充后的障碍物轮廓用 $\overline{P_1'A_1}$、$\overset{\frown}{A_1A_4}$ 和 $\overline{A_4P_2'}$ 表示，$\overline{SA_2}$，$\overline{GA_3}$，$\overline{P_1'A_1}$，$\overline{P_2'A_4}$ 分别是圆 $O$ 的四条切线，且 $\overline{OA_1}=\overline{OA_2}=\overline{OA_3}=\overline{OA_4}=$ 清洁车半短轴，那么在无障碍时，注意到 $\overline{SG}$ 会直接穿越障碍物，从起始点 $S$ 到目标点 $G$ 最短路径为 $\overline{SOG}$，在有障碍物时最短路径为 $\overline{SA_2}\rightarrow\overset{\frown}{A_2A_3}\rightarrow\overline{A_3G}$。

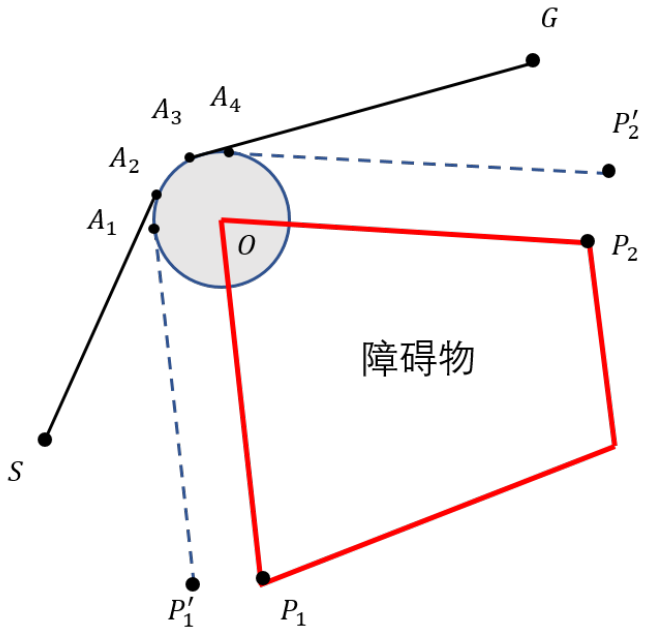


图 7: 避障示意图

**约束条件 3**

如果不考虑式 (4) 中的转角约束，则最优路径将如图8所示

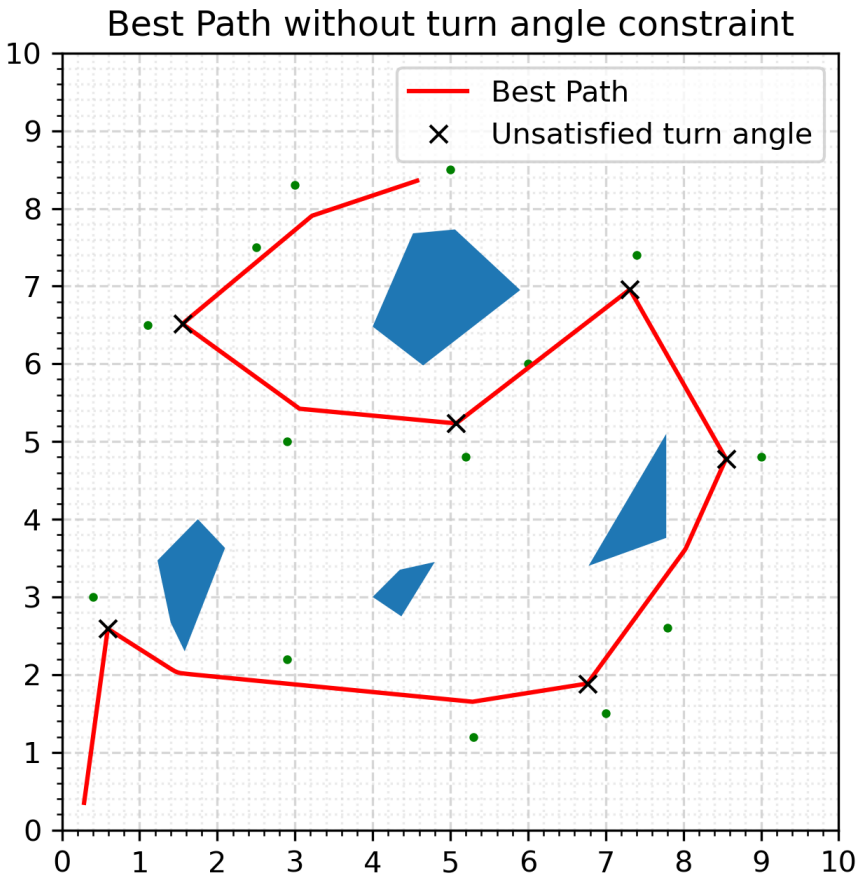


图 8: 无转向角度约束下最优路径

其中黑色叉号标注的是不满足式 (4) 中的转角约束的路径点。为了解决此约束，我们用后处理的思想，对图8中展示的路径进行修正，即可得到满足式 (4) 的新路径。同时，对约束 1 做进一步的放缩，即可得到满足以上所有约束的最终路径，具体方法如下。

首先对路径进行修正，我们采用如算法1所示的处理方式，可以确保所有路径点都满足式 (4) 中的转角约束

**算法 1** 转角修正

**给定:** 最佳路径 $\{P_i\}$

**输出:** 修正后的最佳路径 $\{P_i\}$

1: **重复运行:**

2: 确定图9中的 $P_a, P_b$

3: 将 $P_2$ 从 $\{P_i\}$ 中删除

4: 将 $P_a, P_b$ 插入 $\{P_i\}$ 中原 $P_2$ 所在的位置

5: **直到:** $\{P_i\}$ 所有的所有路径点都满足约束

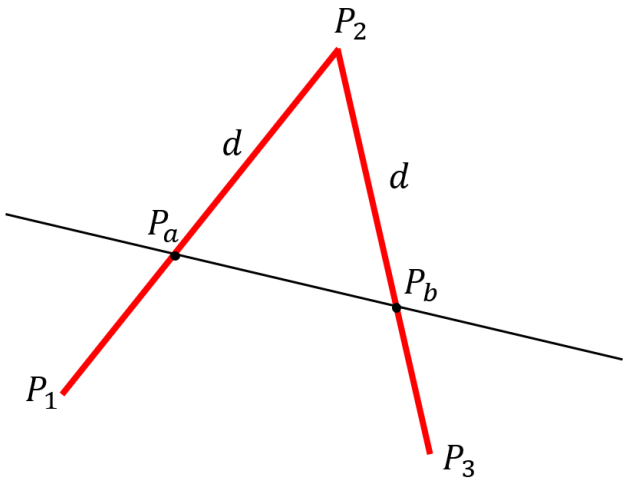


图 9: 路径点修正示意图

在图9中，路径点 $P_2$ 不满足约束。为此，我们分别沿 $P_2P_1$ 与 $P_2P_3$ 两个方向截取长度 $d$，得到新的路径点

$$\begin{aligned} P_a &= P_2 + d \cdot \frac{P_2P_1}{|P_2P_1|} \\ P_b &= P_2 + d \cdot \frac{P_2P_3}{|P_2P_3|} \end{aligned} \tag{7}$$

注意，在算法1中，长度 $d$ 需要随着循环的进行递减，以确保 $d < ||P_xP_y||$，本文采用的递减方法为 $d := d * 0.1$

由于上述方法产生新路径点 $P_a, P_b$ 会“远离”旧路径点 $P_2$，为了确保修正后的的路径依然满足约束 1，我们在约束 1 的方程6中乘以一个放缩系数 $\gamma$，即

$$\| P(i) - M(i) \|_2 \leq \frac{d}{2} * \gamma, \ 1 \leq i \leq n \tag{8}$$

图10展示了采用算法1修正后的放大细节。

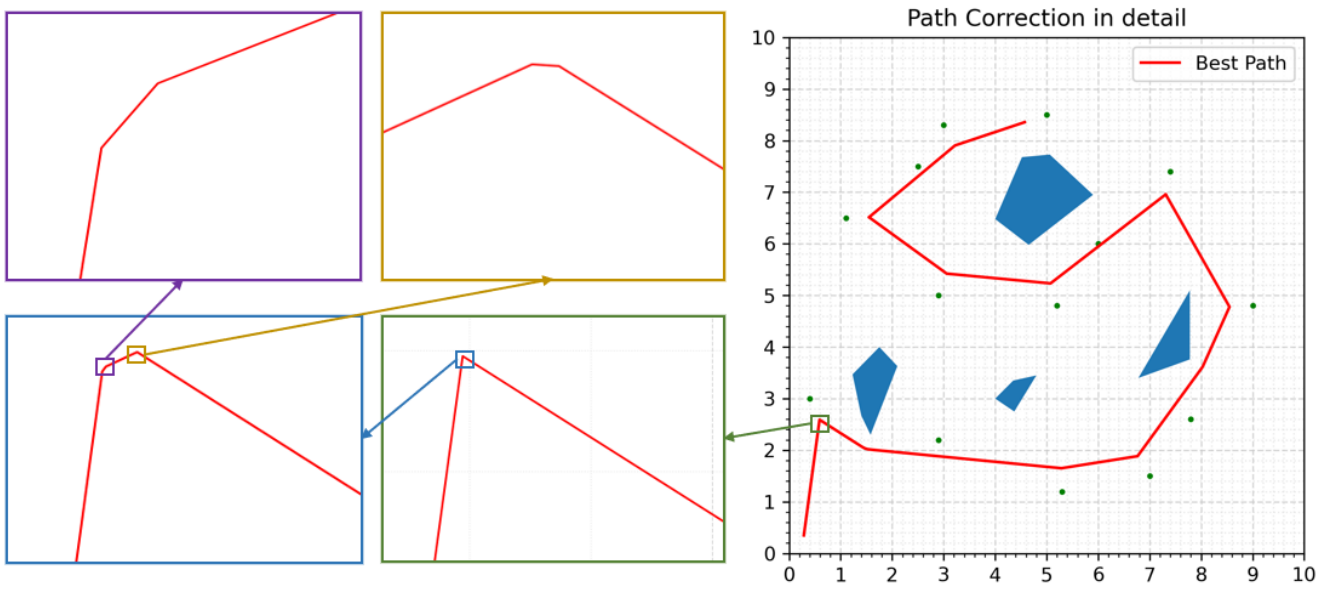


图 10: 路径点修正示意图

## 4 结果

我们用 python 语言编写本程序，并使用 gekko 框架调用 APOPT 求解器对上述优化问题进行求解。在 Intel Xeon 3.7GHz 处理器上求解优化问题，耗时约 2.8 秒。

最终结果如图11所示，其中包含 57 步，总路程为 24.867793，**建议放大图像查看细节**。

路径图文件与路径数据文件见附件，本文代码已上传至https://github.com/RichealYoung/CleaningRobot。

## 参考文献

[1] T. Lozano-Pérez and M. A. Wesley, “An algorithm for planning collision-free paths among polyhedral obstacles,” *Communications of the ACM*, vol. 22, no. 10, pp. 560–570, 1979.

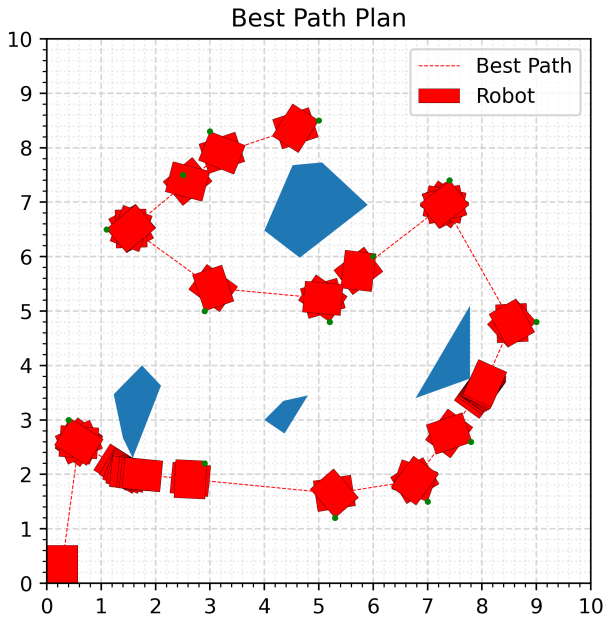


图 11: 最优路径规划示意图