通过大量安全盒进行快速路径规划

Tobia Marcucci、Parth Nobel、Russ Tedrake 和 Stephen Boyd

arXiv:2305.01072v2 [cs.RO] 2 Jan 2024

摘要—我们提出了一种快速算法，用于设计受限于一组轴对齐盒内的光滑路径（或轨迹）。我们考虑这类安全盒数量较多的情况，并且可以离线对其进行基本预处理（例如求取它们的交集）。在运行时，我们能够快速生成给定初始位置和终止位置之间的光滑路径。我们的算法设计的轨迹能够保证在任何时刻都是安全的，并且在不存在此类轨迹时能够检测出不可行性。我们的算法基于两个能够高效求解的子问题：在加权图中寻找最短路径，以及求解（多个）凸优化控制问题。我们通过大规模数值示例展示了所提出的路径规划器，并提供了一种高效的开源软件实现：fastpathplanning。

索引词—运动与路径规划、优化与最优控制、碰撞规避、凸优化。

I. 引言

路径规划是几乎所有自主系统的核心问题。无人驾驶汽车、无人机、自主飞行器、机器人操作臂和足式机器人，只是依赖路径规划算法在环境中导航的系统中的几个示例。路径规划问题在许多方面都可能颇具挑战性。环境可能是动态的，即会随时间变化；也可能由于含噪传感器测量 [1]–[4] 而具有不确定性。计算可能需要满足严格的实时要求 [5]–[7]。多个机器人之间缺乏集中协调的交互可能导致博弈论问题 [8]–[11]。在本文中，我们考虑这样一类问题：需要在一个完全已知且静态的环境中寻找一条单一的光滑路径，而该环境可能非常庞大且导航起来十分复杂。例如，无人机检查工业厂房，或移动机器人在大型仓库中运输包裹时，就会遇到这种情况。

与以往的方法类似 [12], [13], ，我们假设环境由一系列安全集描述，系统或机器人可以在其中自由移动而不与障碍物发生碰撞。我们的问题是找到一条包含在这些安全集的并集中、连接给定起点和终止点的光滑路径。我们考虑安全集为轴对齐盒且数量众多（数千或数万个）的情况。请注意，将复杂环境分解为盒子时可以采用近似（保守）方式，并且可以使用现有算法 [14]–[18], 的简单变体以及针对运动学量身定制的方法来计算。

T. Marcucci 和 R. Tedrake 任职于麻省理工学院电气工程与计算机科学系，美国马萨诸塞州剑桥市 02139。P. Nobel 和 S. Boyd 任职于斯坦福大学电气工程系，美国加利福尼亚州斯坦福市 94305。通讯作者为 T. Marcucci：tobiam@mit.edu。

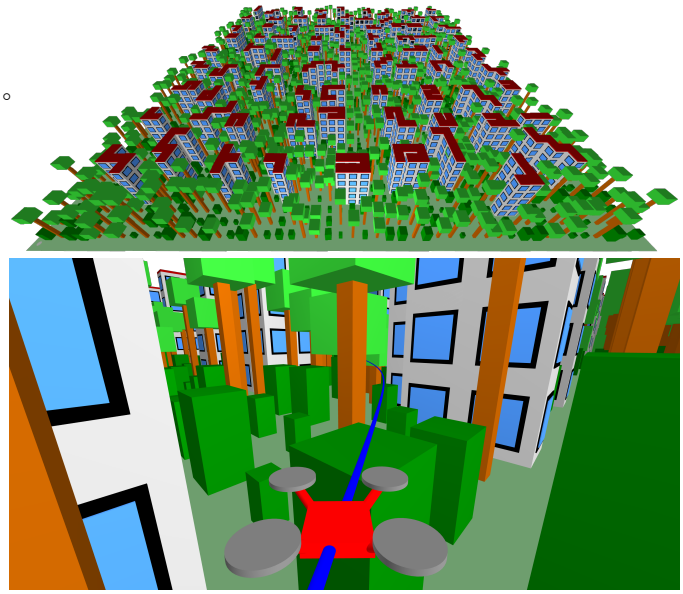


图1. 四旋翼无人机飞越模拟村庄的路径规划。上图。 村庄由建筑物、树木和灌木组成。自由空间使用一万多个安全盒进行分解。下图。 四旋翼无人机飞行的快照。光滑路径连接村庄两个相对的角落，并保证始终无碰撞。在线规划时间仅需几秒。

树木 [19]–[22]。专注于盒状安全集，使我们能够大幅加速算法中的多个部分。

我们的路径规划方法由离线部分和在线部分组成。在离线预处理中，我们构建一个存储安全盒交集的图，并求解一个凸规划问题，为该图的边标注近似距离。由于环境是静态的，这些计算只需执行一次；根据问题规模的不同，所需时间从不到一秒到几十秒不等。在在线部分，我们首先利用离线构建的图，设计一条连接给定初始点和终止点的短多边形曲线。随后，我们求解一系列凸最优控制问题，将这条多边形曲线转换为一条使给定目标函数最小化的光滑路径。我们的算法的在线运行时间主要由这些控制问题决定；不过，其求解时间仅随路径经过的盒子数量线性增加 [23]。因此，对于中等规模的问题，在线规划时间约为百分之一秒；对于超大规模的问题，则约为一秒。以图 1 所示的四旋翼无人机飞行为例，现有技术需要几秒钟才能找到一条穿过不到一百个安全盒的路径

在相同的时间内，我们的规划器可以设计出一条穿过一万多个盒子的路径。

所提出的算法是完备的：如果存在连接初始位置和最终位置的平滑路径，它总能找到这样的路径；否则，它会证明该规划问题不可行。此外，通过使用 B´ézier 曲线进行路径参数化，我们可以保证平滑轨迹在任何时刻都安全，而不仅仅是在有限个采样点上。我们的方法采用启发式方法：尽管它通常能规划出代价较低的路径，但不能保证以最优方式或在最优值的固定百分比范围内解决规划问题。本文所用技术已在配套的开源软件包 fastpathplanning 中实现。

A. 相关工作

过去五十年来，人们开发了各种各样的路径规划算法。[24,Part 2] 对现有文献中的技术作了出色的综述。下面我们回顾与我们的方法最为相关的技术。

与本文方法最为接近的是 [13] 提出的 GCS（凸集图）。与我们的方法类似，GCS 通过一系列经过预处理并构成图的安全集来规划平滑路径。借助 [25]，的优化框架，它为规划问题构造了一个紧的凸松弛，并通过低成本的舍入策略恢复出无碰撞轨迹。得益于这一流程，GCS 还能为其规划的轨迹提供紧的最优性界。另一方面，GCS 试图通过单个凸规划来解决规划问题，因此存在若干局限。首先，目前 GCS 无法高效处理路径加速度及更高阶导数上的代价或约束，而这些正是本文分析的问题中的核心部分。其次，对于本文所考虑的数量极大的安全集，GCS 无法扩展至相应规模。所提出的算法与 GCS 的思路不同：它利用快速图搜索，以启发式方式解决规划问题中的离散部分，随后才使用凸优化来调整连续路径。这种划分牺牲了最优性保证，但保留了算法完备性，并使我们能够快速为超大规模规划问题找到低代价路径。

要设计能够避开障碍物的最优平滑路径，一种自然的方法是采用混合整数规划。早期的混合整数规划形式针对多面体障碍物，为每个障碍物的每个面都设置一个二元变量，以确保轨迹点不会发生碰撞 [26]–[28]。相较之下，[12]中的规划形式利用 [17] 中的算法，以凸区域覆盖全部或部分无碰撞空间，并通过强制每个轨迹段完全位于至少一个凸区域内来确保安全。这样可以提高混合整数规划的效率，因为每个轨迹段在每个安全集上只需一个二元变量。 [12] 中考虑的路径规划问题与我们的方法基本相同，但我们的算法能在极短时间内求解规模大得多的问题（见 §VII-D中的比较）。

无碰撞路径规划的两种常见方法是局部非凸优化 [29]–[34] 和基于采样的算法 [35]–[37]。前一种方法能够处理运动学约束和动力学约束，但容易陷入局部极小值；如果环境中有许多障碍物，往往难以找到可行轨迹。尽管已有多种方法尝试缓解这一问题 [38]–[40],，但当环境复杂时，基于采样的算法通常更可靠（事实上，这类算法是概率完备的）。然而，基于采样的方法在高维空间中可能表现不佳，也不太适合设计平滑路径。与 [13], 类似，我们提出的方法可以看作是基于采样的算法的一种推广：以无碰撞集合取代无碰撞样本。我们不再对环境进行密集采样，而是用大型安全盒填充环境。这将组合复杂度降至最低，并使我们能够通过高效的凸优化，在开阔空间中规划路径 [41]。

将环境分解为安全集（或单元格）在反馈运动规划中也很常见。在此类方法中，通过将导航函数与分段光滑向量场结合来构造反馈规划：前者决定单元格之间的离散转换，后者使单元格中的所有状态流向下一个单元格 [24, §8.4]。类似地， [42]中的方法利用离散抽象 [43]为在多边形环境中移动的平面机器人生成可证明正确的控制策略。在鲁棒运动规划中，漏斗 [32], [44], 管道 [45], 屏障函数 [9], [46],和正不变集 [47]–[50] 经常用于抽象掉连续动力学，并将规划问题简化为离散搜索。尽管这些方法与我们的算法在理念上相似，但那些论文中提出的方法所考虑的问题性质与我们的问题不同。我们的目标不是综合反馈策略，也不显式推理动力学和扰动。我们的目标是设计低成本的安全光滑路径，而我们问题中的挑战在于环境的复杂性（即安全盒的数量）。

最后，本文使用 B´ézier 曲线对平滑路径进行参数化。这些曲线具有多种性质，使其特别适用于凸优化，并且在过去十五年中已广泛应用于路径规划和最优控制领域 [13], [51]–[59]。

B. 概述

本文的组织结构如下。我们在 §II 中陈述路径规划问题，并概述我们的算法。该算法可分为三个部分，其中一个为离线部分，两个为在线部分。§III 介绍离线预处理，该步骤不使用路径端点或具体的目标函数。§IV 展示第一个在线阶段：在安全盒内寻找一条连接给定路径端点的短多边形曲线。§V 介绍第二个在线阶段：将多边形曲线转换为一条成本较低的安全光滑路径。§VI 总结路径规划器的主要性质。§VII 通过多次数值实验评估算法的性能。最后，在

§VIII 中，我们介绍该方法在更一般规划问题上的一些扩展。

II. 路径规划

本节将陈述路径规划问题，并从较高层次介绍我们算法的组成部分。

A. 问题陈述

我们考虑设计一条光滑路径，该路径从给定的初始点 $p^{\text{init}} \in \mathbf{R}^d$ 出发，通往给定的终止点 $p^{\text{term}} \in \mathbf{R}^d$，并位于$\mathbf{R}^d$ 中。我们将路径表示为函数 $p\colon [0,T] \to \mathbf{R}^d$，其中$T$ 是 traversing 该路径所需的时间。除初始点和终止点约束外，

$$p(0) = p^{\text{init}}, \qquad p(T) = p^{\text{term}},$$

我们还要求路径保持在给定的安全点集合 $\mathcal{S} \subset \mathbf{R}^d$ 内:

$$p(t) \in \mathcal{S}, \quad t \in [0,T].$$

我们假设安全集 $\mathcal{S}$ 是若干轴对齐盒的并集,

$$\mathcal{S} = \bigcup_{k=1}^{K} \mathcal{B}_k,$$

with

$$\mathcal{B}_k = \{x \in \mathbf{R}^d \mid l_k \le x \le u_k\}, \quad k = 1, \dots, K.$$

这里的不等式均按元素计算，且盒边界满足 $l_k \le u_k$ for $k = 1, \dots, K$。

我们考虑具有 D 连续导数的路径，并将目标函数设为这些导数的平方 L_2范数的加权和，

$$J = \sum_{i=1}^{D} \alpha_i \int_0^T \|p^{(i)}(t)\|_2^2 \, dt, \tag{1}$$

其中， $p^{(i)}$ 表示 p的第 i 阶导数， α_i 为非负权重。

路径规划问题为

$$\begin{array}{ll} \text{minimize} & J \\ \text{subject to} & p(0) = p^{\text{init}}, \quad p(T) = p^{\text{term}}, \\ & p(t) \in \mathcal{S}, \quad t \in [0,T]. \end{array} \tag{2}$$

优化变量为路径 p。问题数据包括目标权重 α_i、终止时间 T、初始点和终止点 p^{init} 和 p^{term}，以及安全集 $\mathcal{S}$（由盒边界 l_k 和 u_k指定）。此表述仅包含路径规划问题的基本组成部分。例如，这里指定了初始位置和终止位置，但未对初始导数和终止导数施加约束。在 §VIII 中，我们将讨论这些简单扩展中的多种情况，并重点说明对我们的方法所需的修改。

路径规划问题 (2) 是无限维的，但我们将候选路径限制为分段 B´ézier 曲线，这些曲线由有限组控制点参数化。

B. 安全映射

问题 (2) 具有凸二次目标函数、两个线性等式约束，以及安全约束；一般而言，安全约束并非凸约束。安全约束是一个无限集合的析取约束，它们强制要求对每个 $t \in [0,T]$, 点 $p(t)$ 至少位于盒子 $\mathcal{B}_k$ 中的一个内。因此，确保路径 p 安全等价于找到一个函数 $s\colon [0,T] \to \{1, \dots, K\}$，使得

$$p(t) \in \mathcal{B}_{s(t)}, \quad t \in [0,T].$$

$s(t) \in \{1, \dots, K\}$ 表示路径在时间 t选择的安全盒，而整体函数 s 可以视为路径的安全映射 。

我们的安全映射将具有有限数量的转换，即，形式如下

$$s(t) = \begin{cases} s_1 & t \in [t_0, t_1] \\ s_2 & t \in (t_1, t_2] \\ \vdots & \\ s_N & t \in (t_{N-1}, t_N], \end{cases} \tag{3}$$

其中 $0 = t_0 < t_1 < \cdots < t_N = T$。我们将把$s_1, \dots, s_N$称为安全映射 s 的盒序列 ，并将$T_1 = t_1 - t_0, \dots, T_N = t_N - t_{N-1}$称为遍历时间。

就安全映射而言，路径规划问题是

$$\begin{array}{ll} \text{minimize} & J \\ \text{subject to} & p(0) = p^{\text{init}}, \quad p(T) = p^{\text{term}}, \\ & l_{s(t)} \le p(t) \le u_{s(t)}, \quad t \in [0,T], \end{array} \tag{4}$$

其中，变量为路径 p 和安全映射 s。我们注意到，如果盒序列 $s_1, \dots, s_N$ 固定，则问题 (4) 可简化为一个非凸但连续的最优控制问题，其决策变量为路径 p 和遍历时间 $T_1, \dots, T_N$。如果我们进一步固定遍历时间，那么安全映射便完全确定，此时问题 (4) 就变成一个具有二次目标函数和线性约束的凸最优控制问题。

C. 可行性

如果安全映射满足以下条件，我们就称其为可行的

$$\begin{array}{l} p^{\text{init}} \in \mathcal{B}_{s_1}, \quad p^{\text{term}} \in \mathcal{B}_{s_N}, \\ \mathcal{B}_{s_j} \cap \mathcal{B}_{s_{j+1}} \neq \emptyset, \quad j = 1, \dots, N-1. \end{array} \tag{5}$$

第一个条件表明，盒序列中的第一个盒和最后一个盒分别覆盖初始点和终止点。第二个条件要求每两个相邻的盒子都相交；因此，可以沿着一条连续路径遍历该盒序列。

重要的是，当且仅当存在可行安全映射时，路径规划问题 (4) 才是可行的。为说明这一点，注意如果路径 p 和安全映射 s 对于 (4) 是可行的，那么该安全映射必须满足 (5) 中的两个条件。（特别地，第二个条件由 p 的连续性推出，这确保了对于所有 $j = 1, \dots, N-1$, $p(t_j) \in \mathcal{B}_{s_j} \cap \mathcal{B}_{s_{j+1}}$。）反过来，假设某个安全映射是可行的，并令 $p_j \in \mathcal{B}_{s_j} \cap \mathcal{B}_{s_{j+1}}$ 对于 $j = 1, \dots, N-1$。则以节点 $p^{\text{init}} = p_0$、 $p_1, \dots, p_{N-1}, p_N = p^{\text{term}}$ 为节点的多边形曲线是

完全包含于安全集 $\mathcal{S}$ 中。通过以下步骤，我们构造一条路径 p，使其具有 D 个连续导数，并沿多边形曲线移动（因此是安全的）。我们选取任意时刻 $0 = t_0 < t_1 < \cdots < t_N = T$。我们选择多边形曲线的任意光滑时间参数化，使其满足插值条件 $p(t_j) = p_j$（对于 $j = 0, \ldots, N$），以及导数约束 $p^{(i)}(t_j) = 0$（对于 $i = 1, \ldots, D$ 和 $j = 1, \ldots, N-1$）。尽管多边形曲线存在折点，路径 p 仍然是可微的 D 次，因为它在每个折点处都会完全停止。将该路径与可行安全映射配对后，我们便得到问题 (4) 的一个可行解。

D. 方法概述

这里我们从高层次介绍路径规划算法的三个阶段，具体细节将在后续章节中说明。

离线预处理：离线预处理仅使用安全集 $\mathcal{S}$，即安全盒 $\mathcal{B}_1$、……、$\mathcal{B}_K$。在此阶段，我们构造一个线图 G，其顶点对应于两个盒子交集中的点，边则连接位于同一个盒子中的点对。将该图视为$\mathbf{R}^d$的子集时，它完全位于安全集中，可用于快速设计连接给定起点和终止点的安全多边形曲线。与顶点关联的点称为代表点，并经过优化以尽量减小线图中各边的欧几里得长度总和。这是一种启发式方法，可用于缩短多边形曲线的长度。

1: **过程** 离线预处理
2：计算安全盒 $\mathcal{B}_1$、……、$\mathcal{B}_K$3：构造线图 G4：优化代表点
5: **结束过程**

多边形阶段：在这里，我们寻找一条多边形曲线 $\mathcal{C}$，它连接 p^{init} 和 p^{term}，完全包含在安全集$\mathcal{S}$中，并且长度较短。首先，通过求解在线图中离线构建的最短路径问题，对该曲线进行初始化。随后，通过一个迭代过程对其进行缩短：在该过程中，我们交替执行两个步骤，即在固定盒序列的情况下最小化曲线长度，以及在固定多边形曲线的情况下更新盒序列。

1: **过程**多边形阶段
2：连接 p^{init} 与 p^{term} 使用安全多边形曲线 $\mathcal{C}$3: **当**未收敛**时**4: 修正盒序列 $s_1, \ldots, s_N$ 并缩短曲线5: 固定曲线并改进盒序列6: **结束循环**
7: **结束过程**

平滑阶段：在此阶段，我们固定多边形阶段中确定的盒序列s_1，……，s_N，并沿曲线 $\mathcal{C}$ 遍历这些盒。如上所述，这会将问题 (4) 转化为一个连续但非凸的最优控制问题。为了解决这一控制问题，我们首先使用一种简单的启发式方法来估计初始遍历时间 T_1，……，T_N。然后，我们在两个凸最优控制问题之间交替进行。在第一个问题中，我们固定遍历时间（从而指定整个安全映射 s），并优化路径的形状。在第二个问题中，我们通过求解该非凸最优控制问题的局部凸近似，尝试改进遍历时间。

1: **过程** SMOOTHPHASE2: 固定盒序列 $s_1, \ldots, s_N$
3: 估算遍历时间 $T_1, \ldots, T_N$4: **当**未收敛 **时执行**
5: 固定遍历时间并优化路径 p6: 尝试改进遍历时间7: **结束循环**8: **结束过程**

III. 离线预处理

在本节中，我们将描述算法的离线部分。以下步骤也将在本节末尾通过一个简单的示例进行说明。

A. 线图

我们首先计算与安全盒相关的线图。该图的顶点是相交的安全盒对，边则连接共享一个盒子的交集对。形式上，线图是一个无向图$G = (\mathcal{V}, \mathcal{E})$，其顶点是

$$\mathcal{V} = \{\{k, l\} \subseteq \{1, \ldots, K\} \mid \mathcal{B}_k \cap \mathcal{B}_l \neq \emptyset,\ k \neq l\},$$

和边

$$\mathcal{E} = \{\{v, w\} \subseteq \mathcal{V} \mid v \cap w \neq \emptyset,\ v \neq w\}.$$

线图这一名称源于这样一个事实：G 可以等价地定义为我们这组盒子的交图的线图。

线图使我们能够高效地构造有安全性保证的多边形曲线。考虑线图中的一条路径。对于该路径中的每个顶点，在对应盒子的交集中选择$\mathbf{R}^d$ 中的一个点。然后构造一条经过这些点的多边形曲线。该曲线中的每条线段都对应线图中的一条边，因此也对应一个安全盒。根据构造方式，该安全盒完全包含这条线段。由此可知，整条多边形曲线都是安全的。

由于计算两个盒子的交集是一个简单操作，即使盒子数量 K 非常庞大，我们也能非常高效地构造线图 G。我们的实现基于 [60,§2]中的技术。

B. 代表点

下一步是为线图的每个顶点选择一个代表点，即为每对相交的盒子选择一个代表点。为了启发式地缩短按上述方法构造的多边形曲线，我们将这些点放置在靠近线图中相邻顶点的位置。更正式地说，记 $x_v \in \mathbf{R}^d$ 为顶点 $v \in \mathcal{V}$的代表点，我们最小化所有由边连接的代表点对之间的欧几里得距离之和：

$$\begin{array}{ll} \text{minimize} & \sum_{\{v,w\} \in \mathcal{E}} \|x_v - x_w\|_2 \\ \text{subject to} & x_v \in \mathcal{B}_k \cap \mathcal{B}_l, \quad v = \{k, l\} \in \mathcal{V}. \end{array} \tag{6}$$

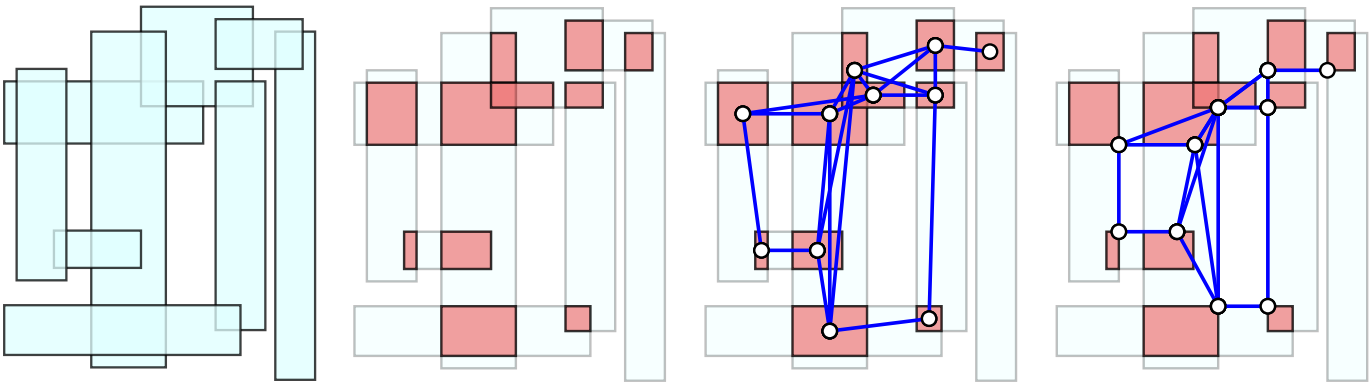


图 2. 安全盒的离线预处理。左侧。 安全盒。中左。 安全盒的两两交集。中右。 线图，其中顶点位于盒交集中，边连接共享同一个安全盒的交集。右侧。 经过优化的代表点构成的线图。

这里的变量是代表点 x_v、 $v \in \mathcal{V}$。每个点都受约束，必须位于相应的盒交集中，而该交集本身是一个轴对齐盒。这是一个凸优化问题，可表示为二阶锥规划（SOCP），并可高效求解 [41,§4.4.2], [61]。

按照式 (6) 优化代表点 x_v 的位置后，为线图中的每条边 $\{v, w\}$ 赋予权重 $\|x_v - x_w\|_2$。

C. 示例

我们通过一个小型问题来说明离线预处理，该问题将在整篇论文中作为示例。该问题在 $d = 2$ 维度中包含 $K = 9$ 个安全盒，如图 2 所示。左图显示了安全盒，中左图显示了它们的盒交集（当两个以上盒子相交时，部分区域会重叠）。这些交集对应于线图的 $|\mathcal{V}| = 11$ 个顶点。在中右图中，我们将线图的 $|\mathcal{E}| = 20$ 条边显示为连接盒交集中心的线段。右图显示了经过优化的代表点，这些点使线图各边上的欧几里得距离总和最小，即 (6) 的一个解。请注意，图中有些 20 条边彼此重叠。还要注意，将线图视为 $\mathbf{R}^2$，的子集时，它完全包含于安全集内，因为每条边至少位于一个安全盒中。

IV. 多边形阶段

现在我们描述算法的第一个在线阶段，在此阶段，我们设计一条长度较短的安全多边形曲线 $\mathcal{C}$，连接 p^{init} 和 p^{term}。本节末尾给出了以下步骤的示意图，我们将在那里继续讨论贯穿全文的示例。

A. 最短路径问题

我们使用线图 G 来初始化多边形曲线 $\mathcal{C}$。我们通过添加两个新顶点来扩充线图，这两个顶点的代表点分别为 p^{init} 和 p^{term}。在 p^{init}与所有包含 p^{init}的安全盒的交集之间添加一条边，即，所有满足 $p^{\text{init}} \in \mathcal{B}_k$ 或 $p^{\text{init}} \in \mathcal{B}_l$的顶点 $\{k, l\} \in \mathcal{V}$。对 p^{term} 也进行类似的操作。至于线图中的其他边，这些新边的权重设为它们所连接的代表点之间的欧几里得距离。随后，我们求出从初始点到终止点的最短路径，并沿着这条路径连接代表点，从而恢复出一条初始多边形曲线$\mathcal{C}$ 。如上所述，这条曲线是安全的，因为它的每条线段都包含在一个安全盒内。

这一步最短路径步骤用于确定我们的路径规划问题是否可行。如果扩展后的线图中，与 p^{init} 和 p^{term} 关联的顶点之间不存在路径，则路径规划问题 (4) 不可行。反之，如果这两个顶点之间存在路径，则路径规划问题可行，因为可以按照 §II-C 中的方法构造一条可行轨迹。

识别所有包含初始点和终止点的安全盒这一问题称为刺点问题；鉴于构建线图时已完成预计算，这一步所需时间可以忽略不计 [60]。使用经过优化的 Dijkstra 算法实现（例如，scipy [62] 提供的实现），求最短路径也非常快。

B. 缩短多边形曲线

得益于对（6）中代表点的优化，我们的初始多边形曲线 $\mathcal{C}$ 通常较短。然而，在线获知初始点和预处理期间未知的终止点后，我们可以进一步缩短这条曲线。该过程通过迭代完成：对于固定的盒序列，我们交替求解使曲线长度最小化的凸规划，并在固定多边形曲线的情况下改进盒序列。

多边形曲线的优化：用 $\mathcal{C}$ 表示当前迭代中的曲线（初始化为最短路径问题的解）。设 N 为 $\mathcal{C}$ 中的线段数， $y_0, \ldots, y_N \in \mathbf{R}^d$ 为曲线节点，其中 $y_0 = p^{\text{init}}$ 且 $y_N = p^{\text{term}}$。对于 $j = 1, \ldots, N$，另设 s_j 为覆盖 y_{j-1}与 y_j之间线段的安全盒索引。固定当前曲线经过的盒序列 $s_1, \ldots, s_N$，并优化节点 y_j的位置

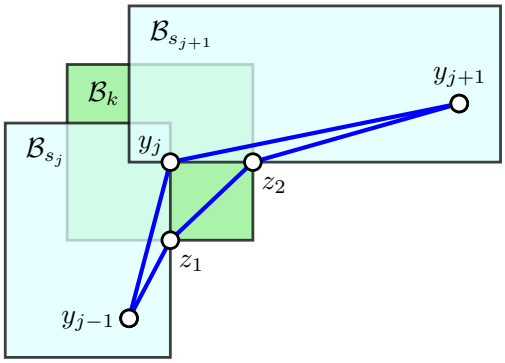


图 3. 通过在当前盒序列中插入一个新盒（$\mathcal{B}_k$ 绿色），缩短多边形曲线 $\mathcal{C}$ 。

使曲线长度最小化。这将得到如下问题

$$\begin{array}{ll}\text{minimize} & \sum_{j=1}^{N}\|y_j - y_{j-1}\|_2 \\ \text{subject to} & y_0 = p^{\text{init}}, \quad y_N = p^{\text{term}}, \\ & y_j \in \mathcal{B}_{s_j} \cap \mathcal{B}_{s_{j+1}}, \quad j = 1, \ldots, N-1,\end{array} \tag{7}$$

变量为 $y_0, \ldots, y_N$。这是一个带有带状约束的小型二阶锥规划，求解效率很高，所需时间仅与 N 线段 [23]的数量呈线性关系。

式（7）得到的最优节点定义了我们的新曲线 $\mathcal{C}$。我们假设这些节点互不相同，因为如果有两个节点重合，总可以删去其中一个。

改进盒序列：求解问题（7）后，节点 $y_0, \ldots, y_N$ 使给定盒序列 $s_1, \ldots, s_N$对应的曲线长度最小。然而，如图 3 所示，插入一个新盒可能为进一步缩短多边形曲线创造空间。在第二步中，我们寻求一个包含当前曲线、并保证曲线长度减小的新盒序列。由于安全集是盒子，这一步将非常迅速。

对于所有 $j = 1, \ldots, N-1$，我们求解一个刺点问题，找出所有包含曲线节点 y_j 的盒子 $\mathcal{B}_k$。然后，考虑在盒序列中将索引 k 插入到 s_j 和 s_{j+1} 之间。如图 3 所示，这一插入会产生问题 (7) 的一个新实例，其中变量 y_j 被替换为两个变量：

$$z_1 \in \mathcal{B}_{s_j} \cap \mathcal{B}_k, \qquad z_2 \in \mathcal{B}_k \cap \mathcal{B}_{s_{j+1}}.$$

选择 $z_1 = z_2 = y_j$ 会得到问题 (7) 这个新实例的一个可行解，且不会改变曲线 $\mathcal{C}$ 的长度。因此，当且仅当该可行解不是最优解时，插入 $\mathcal{B}_k$ 才会使曲线变短。

我们固定 $z_1 = z_2 = y_j$，并检查新实例 (7) 的最优性条件是否能够满足。如 §A 中所述，这项检查可归结为寻找一个向量 $\lambda \in \mathbf{R}^d$，使其满足以下不等式：

$$\begin{array}{lll}\|\lambda\|_2 \le 1, & L_1(\lambda - \lambda_1) \ge 0, & L_2(\lambda - \lambda_2) \le 0, \\ & U_1(\lambda - \lambda_1) \le 0, & U_2(\lambda - \lambda_2) \ge 0.\end{array} \tag{8}$$

这里的向量 λ_1, $\lambda_2 \in \mathbf{R}^d$ 是固定且已知的，由以下公式给出

$$\lambda_1 = \frac{y_j - y_{j-1}}{\|y_j - y_{j-1}\|_2}, \qquad \lambda_2 = \frac{y_{j+1} - y_j}{\|y_{j+1} - y_j\|_2}.$$

矩阵 L_1 和 U_1 选取盒约束 $y_j \in \mathcal{B}_{s_j} \cap \mathcal{B}_k$ 中的非活动不等式索引（L_1 用于下界，U_1 用于上界）。类似地，L_2 和U_2 选取 $y_j \in \mathcal{B}_k \cap \mathcal{B}_{s_{j+1}}$中的非活动不等式。

检查 (8) 中的不等式是否可满足非常快。事实上，由于 L_1、L_2、U_1和 U_2 是选择矩阵，(8) 中对应的不等式只对 λ 的部分分量施加界限。我们用 $c_1 \le \lambda \le c_2$ 表示这些界限，其中 $c_1 \in (\mathbf{R} \cup \{-\infty\})^d$ 和 $c_2 \in (\mathbf{R} \cup \{\infty\})^d$ 是两个适当的向量。然后，可以计算满足这些界限且欧几里得范数最小的向量 λ：

$$\lambda^\star = \min\{c_2, \max\{c_1, 0\}\}, \tag{9}$$

其中，最小值和最大值均按元素计算。由此可知，$\lambda^\star$ 的范数大于 1，当且仅当不等式组(8)无解；这等价于插入盒子 $\mathcal{B}_k$ 后使曲线 $\mathcal{C}$缩短。

对于每个满足$\lambda^\star$ 的范数大于 1 的索引 $j = 1, \ldots, N-1$，我们都会在序列中插入一个新盒子。如果同一索引j对应多个满足此条件的盒子，我们会选择使 $\lambda^\star$ 的范数最大的那个（§A 中说明了这一启发式方法的依据）。更新盒序列后，我们通过求解问题(7)的新实例来优化曲线 $\mathcal{C}$ 。重复此过程，直到对每个曲线节点 j 和每个盒子 k，上述条件都不再成立。

C. 示例

图 4 延续了前面的示例，并展示了多边形曲线的构造。初始位置 p^{init}和终止位置 p^{term} 分别以黑色圆盘表示，位于左下方和右下方。左图展示了增广线图，其中这两点分别与其相邻顶点相连。初始点 p^{init} 有两个相邻顶点，而终止点 p^{term} 只有一个。中间偏左的图展示了从初始点到终止点的最短路径。在中间偏右的图中，我们固定曲线必须经过的盒子，并通过求解二阶锥规划(7)来最小化曲线长度。在右图中，我们在盒序列中插入一个新盒子，并再次优化曲线节点。在此示例中，多边形阶段经过一次迭代后收敛。

V. 平滑阶段

平滑阶段是我们算法的最后一个阶段。它从多边形曲线 $\mathcal{C}$ 出发，构造一条适用于我们规划问题的光滑路径 p，并且目标函数值较小。对于路径参数化，我们使用分段 B´ézier 曲线，即一系列平滑连接的 B´ézier 曲线。（有时也称为复合 B´ézier 曲线。）本节首先回顾这类曲线的一些基本性质。接下来，我们将介绍用于设计光滑路径的最优控制问题。最后，我们将完成这个贯穿全文的示例。

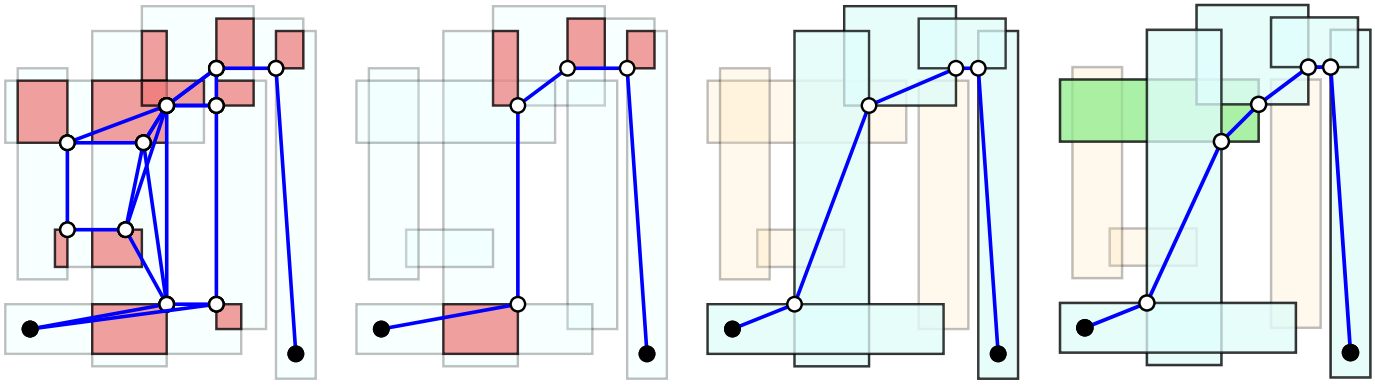


图 4. 算法的多边形阶段。左图。 添加了 p^{init} 和 p^{term}的线图，以黑色圆盘表示。左中图。 从 p^{init} 到 p^{term}的最短路径。右中图。 安全盒序列保持固定，并通过凸优化缩短多边形曲线。右图。 在序列中插入一个新的盒（以绿色显示），并再次缩短曲线。由于无法进一步缩短，多边形阶段在一次迭代后收敛。

A. B´塞zier 曲线

A 贝塞尔曲线是使用伯恩斯坦多项式构造的。次数为 M 的伯恩斯坦多项式定义在区间 $[a,b] \subset \mathbf{R}$ 上，其中 $b > a$，其形式为

$$\beta_n(t) = \binom{M}{n} \left(\frac{t-a}{b-a}\right)^n \left(\frac{b-t}{b-a}\right)^{M-n}, \quad n = 0, \ldots, M.$$

对于 $t \in [a,b]$ 伯恩斯坦多项式均为非负，并且根据二项式定理，它们的总和为 1。因此，标量 $\beta_0(t) \ldots, \beta_M(t)$ 可以被视为凸组合的系数。使用这些系数来组合给定的一组控制点 $\gamma_0, \ldots, \gamma_M \in \mathbf{R}^d$，我们得到一条 B´ézier 曲线：

$$\gamma(t) = \sum_{n=0}^{M} \beta_n(t)\gamma_n.$$

B´ézier 曲线 $\gamma: [a,b] \to \mathbf{R}^d$ 是一个次数为 M 的多项式函数。图 5 展示了一个 B´ézier 曲线示例，其中 $d = 2$ 和 $M = 4$。下面我们列出一些将在本节后续部分用到的重要性质。

端点：一条 B´ézier 曲线从其第一个控制点开始，到其最后一个控制点结束，即，

$$\gamma(a) = \gamma_0, \qquad \gamma(b) = \gamma_M. \tag{10}$$

利用这一性质，只需将每段曲线的最后一个控制点与下一段曲线的第一个控制点设为相等，即可使分段 B´ézier 曲线（即我们的路径）保持连续。

控制多胞体：由于贝塞尔曲线上的每个点都是控制点的凸组合，因此整条曲线都包含在控制点的凸包中：

$$\gamma(t) \in \mathbf{conv}\{\gamma_0, \ldots, \gamma_M\}, \tag{11}$$

对于所有 $t \in [a,b]$ 均成立。这个凸包称为 B´ezier 曲线 γ 的控制多胞体。由这一性质可知，如果所有控制点都位于一个凸集内（在我们的例子中是一个盒），那么 B´ezier 曲线也位于该凸集内。

导数：B´ezier 曲线 γ 的导数 $\gamma^{(1)}$ 也是一条 B´ezier 曲线。它的次数为 $M-1$，其控制点由差分方程给出

$$\gamma_n^{(1)} = \frac{M}{b-a}(\gamma_{n+1} - \gamma_n), \quad n = 0, \ldots, M-1. \tag{12}$$

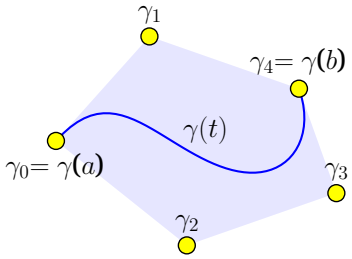


图 5. 控制点为 $\gamma_0, \ldots, \gamma_M$，$M = 4$ 的贝塞尔曲线。曲线从 $\gamma(a) = \gamma_0$ 开始，在 $\gamma(b) = \gamma_M$ 结束，并完全位于控制点的凸包内，图中以阴影标示。

由此迭代可知，任意阶 $i \geq 1$ 的导数 $\gamma^{(i)}$ 都是一条次数为 $M - i$的 B´ezier 曲线。此外，分段 B´ezier 曲线的导数也是分段 B´ezier 曲线，并且可以利用端点性质(10)来保证其连续性。

平方 L_2 范数：贝塞尔曲线 γ 的 L_2 范数平方可通过以下表达式 [63,§3.3] 表示为控制点的函数：

$$\int_a^b \|\gamma(t)\|_2^2 \, dt = (b-a)Q(\gamma_0, \ldots, \gamma_M), \tag{13}$$

其中 Q 是定义如下的凸二次函数：

$$Q(\gamma_0, \ldots, \gamma_M) = \frac{1}{2M+1} \sum_{m=0}^{M} \sum_{n=0}^{M} \frac{\binom{M}{m}\binom{M}{n}}{\binom{2M}{m+n}} \gamma_m^T \gamma_n.$$

利用此公式，我们还可以计算分段贝塞尔曲线的平方 L_2 范数及其导数。

B. 非凸最优控制问题

在平滑阶段，我们将注意力限制在分段 B´ezier 曲线上，这些曲线遍历与曲线 $\mathcal{C}$ 相同的盒序列s_1、…、s_N。这将路径规划问题 (4) 转化为一个有限维、仅含连续变量但非凸的最优控制问题。本小节将阐述这一控制问题，下一小节则介绍我们求解该问题的方法。

变量：问题 (4) 中的变量是安全映射 s 和路径 p。这里盒序列是固定的，因此安全映射仅由遍历时间 T_1、. . . 、T_N指定，它们是我们控制问题中的第一组变量。对于路径参数化，我们使用一条分段 B´ezier 曲线，其中包含 N 段（每段对应路径必须经过的一个安全盒）。每一段，即子路径，都是一条 B´ezier 曲线

$$p_j : [t_{j-1}, t_j] \to \mathbf{R}^d, \qquad j = 1, \ldots, N.$$

这些 B´ézier 曲线的次数为 M，其控制点

$$p_{j,0}, \ldots, p_{j,M} \in \mathbf{R}^d, \quad j = 1, \ldots, N, \tag{14}$$

是我们控制问题中的第二组变量。

对于 $i = 1, \ldots, D$，子路径的导数 $p_j^{(i)}$ 是次数为 $M - i$ 的 B´ézier 曲线。我们的最后一组变量是这些导数的控制点:

$$p_{j,0}^{(i)}, \ldots, p_{j,M-i}^{(i)} \in \mathbf{R}^d, \quad i = 1, \ldots, D, \ j = 1, \ldots, N. \tag{15}$$

为简化记法，我们有时将式 (14) 中的控制点记为 $p_{j\,0}^{(0)},,\ldots,p_j^{(0)},_M$，其中上标表示零阶导数。

约束：我们利用 B´ézier 曲线的性质来构建控制问题的约束。

利用端点性质 (10)，问题 (4) 中的边界条件可以简单地表示为

$$p_{1,0} = p^{\text{init}}, \qquad p_{N,M} = p^{\text{term}}. \tag{16}$$

同样，我们通过以下方式保证路径的连续性和可微性:

$$p_{j,M-i}^{(i)} = p_{j+1,0}^{(i)}, \quad i = 0, \ldots, D, \ j = 1, \ldots, N-1. \tag{17}$$

性质 (11) 表明，B´ézier 曲线位于其控制多胞体内。因此，为了确保子路径 p_j 完全包含在相应的安全盒 $\mathcal{B}_{s_j}$ 内，只需约束其控制点:

$$l_{s_j} \le p_{j,n} \le u_{s_j}, \quad j = 1, \ldots, N, \ n = 0, \ldots, M. \tag{18}$$

由于每条子路径 p_j 都被约束在一个安全盒内，因此整条分段 B´ézier 曲线 p都是安全的。

子路径导数的控制点需要满足与 (12) 类似的差分方程:

$$p_{j,n}^{(i)} = \frac{M-i+1}{T_j}\left(p_{j,n+1}^{(i-1)} - p_{j,n}^{(i-1)}\right),$$
$$i = 1, \ldots, D, \ j = 1, \ldots, N, \ n = 0, \ldots, M-i. \tag{19}$$

请注意，这些等式约束是非线性的，因为控制点和遍历时间都是我们优化问题中的变量。

最后，遍历时间必须为正 ，

$$T_j > 0, \qquad j = 1, \ldots, N, \tag{20}$$

并求和至终止时间，

$$\sum_{j=1}^{N} T_j = T. \tag{21}$$

目标函数：我们将目标函数 (1) 中的积分拆分为 N 项之和（每条子路径对应一项）：

$$J = \sum_{i=1}^{D} \alpha_i \sum_{j=1}^{N} J_{i,j}.$$

我们利用式（13）将每一项表示为控制点和遍历时间的函数:

$$J_{i,j} = \int_{t_{j-1}}^{t_j} \|p_j^{(i)}(t)\|_2^2 \, dt = T_j Q(p_{j,0}^{(i)}, \ldots, p_{j,M-i}^{(i)}), \tag{22}$$

对于 $i = 1$、.....、D 以及 $j = 1$、.....、N。请注意，二次函数 Q 是凸的，但其与遍历时间T_j 的乘积使我们的目标函数变为非凸。

优化问题：收集所有分量，我们得到优化问题。

$$\begin{aligned} &\text{minimize} \quad J \\ &\text{subject to} \quad \text{constraints (16) to (21).} \end{aligned} \tag{23}$$

该程序具有最优控制问题的结构，其中差分方程 (19) 充当动力系统，将各变量随时间联系起来。结合非凸目标项 (22)，这一非线性差分方程使问题成为非凸问题。然而，与其无限维对应形式 (4) 类似，如果固定遍历时间，该问题便可简化为一个凸二次规划 (QP) [41, §4.4] 。事实上，这样做会使差分方程变为线性方程，并使目标函数成为凸二次函数。

曲线次数与可行性：如果子路径的次数满足 $M \ge 2D + 1$，则可保证问题 (23) 可行。事实上，与 §II-C 中的讨论类似，这个最低次数确保每条子路径都可以是线段，且其端点处的前 D 阶导数均为零。这样，整体路径 p 就可以呈现为安全多边形曲线 $\mathcal{C}$ 的形状，同时满足可微性约束。另请注意，次数 M 必须至少为 $D+1$，因为连续性和可微性约束 (17) 固定了每条子路径的 $D + 1$ 个控制点的值。在本文余下部分，我们将使用次数为 $M = 2D + 1$ 的曲线，以确保问题 (23) 始终可行。不过在实践中，我们发现，取更接近 $D + 1$ 的 M 值时，问题也几乎总是可行。

C. 通过凸交替求解

我们通过交替求解投影问题 和切线问题来求解非凸规划 (23)，二者都是凸最优控制问题。与规划方法的其他部分一样，这一步采用启发式方法：它保证能找到 (23) 的一个可行解，但该解不一定是最优解。

初始化：我们首先计算遍历时间 $T_1, \ldots, T_N$的初始估计值，使其满足约束 (20) 和 (21)。为此，我们假设以恒定速度沿多边形曲线 $\mathcal{C}$ 行进。分配给第 j 个盒的时间窗口 T_j 等于节点 y_j 和 y_{j-1}之间的距离除以曲线 $\mathcal{C}$ 的总长度，再乘以终止时间 T。

尽管这种启发式方法非常简单，但我们发现它对大多数问题都很有效。当然，也可以对遍历时间进行更精确的初始估计，但根据我们的经验，这些估计带来的复杂性增加很少值得。

投影问题：在这一步中，我们固定当前的遍历时间值（按上述方式初始化），并求解控制问题 (23)。如上所述，这是一个凸二次规划，其作用是将当前迭代点投影到 (23) 的非凸可行集上。得益于其最优控制结构和带状约束，这些二次规划的求解时间仅随路径 [23] 遍历的盒子数量 N 线性增长。

切线问题：在这一步中，我们尝试通过求解 (23) 的一个凸近似来改进对遍历时间的估计。

我们引入辅助变量，用以表示遍历时间与路径导数控制点的乘积：

$$q_{j,n}^{(i)} = T_j p_{j,n}^{(i)}, \tag{24}$$

对于 $i = 1, \dots, D$, $j = 1, \dots, N$，以及 $n = 0, \dots, M - i$。使用这些变量，非线性差分方程 (19) 变为线性方程，

$$q_{j,n}^{(i)} = (M - i + 1)\left(p_{j,n+1}^{(i-1)} - p_{j,n}^{(i-1)}\right),$$

非凸目标项 (22) 变为二次函数除以线性函数 [41, §3.2.6]，

$$J_{i,j} = \frac{Q(q_{j,0}^{(i)}, \dots, q_{j,M-i}^{(i)})}{T_j}.$$

（请注意，分子为凸函数且分母为正的二次函数除以线性函数是凸函数，并且可以通过二阶锥表示 [61, §2.3]。）

我们的问题中仅剩的非凸性是非线性等式约束 (24)，我们只需在当前遍历时间 $\bar{T}_j$ 和控制点 $\bar{p}_j^{(i)}{}_{,\,n}$（通过求解投影问题得到）附近对其进行线性化：

$$q_{j,n}^{(i)} = -\bar{T}_j \bar{p}_{j,n}^{(i)} + T_j \bar{p}_{j,n}^{(i)} + \bar{T}_j p_{j,n}^{(i)}.$$

由于该线性化在远离标称点时可能不准确，我们还加入了一个信赖域约束

$$\frac{1}{1+\kappa} \le \frac{T_j}{\bar{T}_j} \le 1 + \kappa, \qquad j = 1, \dots, N. \tag{25}$$

这将遍历时间的最大相对变化限制在 $\kappa > 0$ 以内。

由此得到的问题是一个二阶锥规划，它在局部近似非凸规划 (23)，并尝试通过沿非线性方程 (24) 的切空间迈出一步来改进当前解。与投影问题一样，其求解时间仅随 N 线性增长。我们只保留其解中的最优遍历时间 $T_1^\star, \dots, T_N^\star$，然后求解一个新的投影问题，以获得一条新的可行路径。如果最优目标值低于上一个投影问题的目标值，我们就接受新的时间并更新路径。否则，我们保留之前的时间和路径。

信赖域更新：每次迭代后，无论是否成功，我们都会减小信赖域参数 κ 的值。一个简单的方法是将 κ 除以参数 $\omega > 1$。然而，使用这条规则时，可能会出现连续两次迭代都对遍历时间作出相同的无效更新。具体来说，如果某次迭代未成功，且切线问题中计算出的转换时间不在信赖域 (25) 的边界上，那么在缩小信赖域后，这些时间仍可能是可行的（因而也是最优的）。为防止这种情况，我们采用一种稍复杂的更新方式：

$$\kappa^+ = \frac{1}{\omega}\left(\max\left\{\frac{\bar{T}_1}{T_1^\star}, \frac{T_1^\star}{\bar{T}_1}, \dots, \frac{\bar{T}_N}{T_N^\star}, \frac{T_N^\star}{\bar{T}_N}\right\} - 1\right) \le \frac{\kappa}{\omega}.$$

这里，κ 和 κ^+ 分别是当前和更新后的信赖域参数。括号中的项是事后看来至少会使一个信赖域约束 (25) 起作用的 κ 最小值。如果这些约束中已有一个处于起作用状态，则该规则简化为$\kappa^+ = \kappa/\omega$。如果没有任何信赖域约束处于起作用状态，且本次迭代未成功，则会缩短信赖域，使得上一个切线问题的解对下一个问题不可行。

终止条件：切线问题的最优值小于或等于其前一个投影问题的最优值。当这一差值除以投影问题的代价后小于固定容差$\varepsilon > 0$时，我们终止算法。此时，我们最后求解一次投影问题，并返回迄今为止找到的最优路径。

参数的选择：我们发现，对于大多数问题，κ 的值可以直接初始化为 1。当我们的遍历时间初始化较准确时，较大的 ω（例如，$\omega = 5$）通常效果良好；否则，较小的值（例如，$\omega = 2$）更为有效。本文讨论的数值实验中，我们使用 $\omega = 3$。终止容差的合理取值为 $\varepsilon = 10^{-2}$。

D. 示例

我们通过展示路径规划算法的平滑阶段来结束这个贯穿全文的示例。我们寻求一条$D = 3$ 次连续可微的路径，其总时长T等于 1。我们使用次数为 $M = 2D+1 =$7 的 B´ézier 曲线。我们采用目标权重 $\alpha_1 = \alpha_2 = 0$ 和 $\alpha_3 > 0$，即，我们的目标函数会惩罚路径三阶导数（或加加速度）的平方 L_2 范数。为简化分析，我们选择权重 α_3，使得该问题的全局最小值等于 1。

在图 6 左侧，我们展示了在多边形阶段计算出的曲线，并以青色突出显示了对应的安全盒序列。在图 6 左侧中央，我们展示了通过求解第一个投影问题得到的路径，其遍历时间使用恒定速度启发式进行初始化。路径经过的每个盒子都标有两个数字：上方的百分比表示该盒子的遍历时间 T_j 与终止时间 T 之比；下方的数字表示子路径 p_j 的代价 $J_{3,\,j}$。红色阴影区域是每个盒子内的 B´ézier 控制多胞体婷婷五月

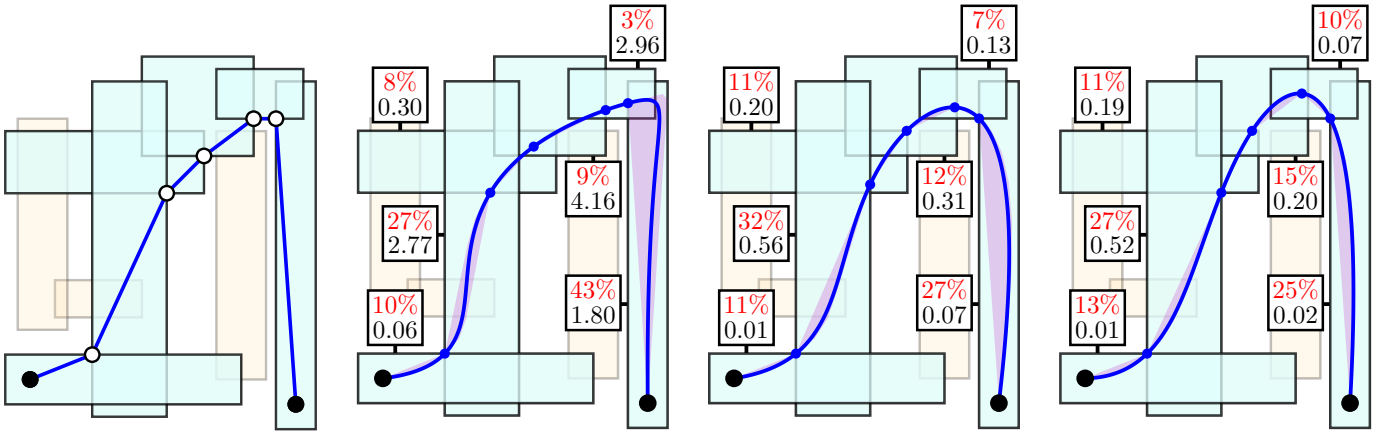


图 6. 我们算法的平滑阶段。左。 来自多边形阶段的曲线及其对应的安全盒序列。中左。 使用多边形曲线估算初始遍历时间，并优化出第一条光滑路径。对于路径经过的每个盒子，标签在顶部显示遍历时间（以终止时间归一化），在底部显示轨迹段的代价。红色阴影区域是 B´ézier 曲线的控制多胞体。中右。 改进遍历时间，并对路径进行第二次优化。右。 最后一次（第四次）迭代得到的路径，其代价与全局最小值相差不到 0.01%。

盒。我们求解切线问题，更新时间遍历时间，并再次求解投影问题。所得路径如图 6 中右侧中央所示。经过四次迭代后，平滑阶段结束，所得路径如图 6 最右侧所示。

初始路径（左侧中间）的代价为 12.04。第一次迭代后的路径（右侧中间）的代价为 1.27，减少了 89%。最终路径（右侧）的代价为 1.0001，基本上就是问题的全局最小值。尽管我们用于初始化遍历时间的简单启发式方法并不准确，但我们的算法只需很少几次迭代就能收敛。

VI. 算法效率与保证

我们简要概述我们的方法的主要特性。

完备性： 我们的算法是完备的：如果存在一条路径，它就能找到一条连接起点和终点位置的安全光滑路径；否则，它会证明问题不可行。通过多边形阶段开始时最短路径问题的解，几乎可以立即判定可行性。如果该问题不可行，那么起点和终止点就无法通过连续曲线连接。反之，如果最短路径问题可行，那么如 §V-B 末尾所述，我们的方法总能找到一条光滑的可行路径。当然，如果安全盒只是对更复杂空间的近似，那么我们方法的完备性也受限于这种近似的保守性。

次优性： 我们的算法是一种启发式方法，不保证能以最优方式（或在固定的最优性容差范围内）求解问题 (2)。在实践中，次优性的主要来源是盒序列的选择：该序列在多边形阶段之后便被冻结，且未考虑路径规划问题的实际目标函数。求解非凸问题 (23) 是次优性的另一个来源；我们在 §V-C 中采用的交替方法旨在优先减少迭代次数，而非降低最终路径的代价。另外两项（影响较小的）近似是使用 B´zier 曲线进行路径参数化，以及安全约束 (18) 的充分性。

路径规划器中的某些启发式步骤并不影响算法完备性，但对于减少上述最优性损失可能发挥重要作用。例如，我们可以直接将盒交集的中心作为代表点，而不必像式 (6) 那样对其进行优化。不过，采用这种选择时，后续的最短路径问题通常会选出效率较低的盒序列。我们方法中另外两个主要的启发式环节是 §IV-B 中对多边形曲线的迭代缩短，以及 §V-C 中遍历时间的初始化。

运行时间： 我们的方法的离线和在线运行时间主要由凸优化问题决定。二阶锥规划 (6) 是最耗时的预处理步骤，但即使对于规模极大的路径规划问题，也能高效求解。此外，我们注意到，该子问题只需以中等甚至较低精度求解。多边形阶段中的二阶锥规划 (7) 几乎不耗时，因为它规模很小且具有带状约束。此外，该阶段通常在四五次迭代内收敛。（更严格地说，该阶段的迭代次数上限为安全盒的数量 K ；由于每次迭代至少会向序列中添加一个盒，而且重复使用盒并非最优。）平滑阶段中的投影二次规划和切线二阶锥规划可能规模较大，但其耗时仅随经过的盒子数量线性增长，而且可以用中等精度求解。还需注意，信赖域 (25) 在平滑阶段的迭代过程中会按几何级数缩小。因此，经过少数几次迭代（采用 §V-C 中给出的参数时通常为四到八次）后，投影问题与切线问题实际上完全相同，平滑阶段也随之终止。

VII. 数值实验

本节通过多次数值实验分析我们方法的性能。每项实验均使用软件实现 fastpathplanning 中的默认值运行，下面将对此作简要介绍。

计算在一台配备 2.4 GHz 八核 Intel Core i9 处理器和 64 GB 内存的计算机上进行。

为提高代码可读性并加快原型开发，当前版本的 fastpathplanning 使用 CVXPY [64] 构建凸优化问题，并将其传递给求解器。由此产生的开销对于某些问题而言甚至可能比实际求解器运行时间高出数倍。由于直接与求解器通信可以使该开销降至可忽略不计的程度，因此本文报告的运行时间中已剔除在 CVXPY 内部耗费的时间。

A. 软件包

本文介绍的算法已在开源 Python 软件包 fastpathplanning 中实现，该软件包可从 https://github.com/cvxgrp/ fastpathplanning 获取。对于图计算（例如线图的构建），我们使用 NetworkX 3.2 [65]。对于在线图中求解最短路径问题，我们使用 scipy 1.11.3 [62]。凸优化问题使用 CVXPY 1.4.1 [64], 进行描述，并使用 Clarabel 0.6.0 求解器 [66]求解。

以下是 fastpathplanning 使用方法的一个基本示例。

```
1 import fastpathplanning as fpp 2 3 # 离线预处理
4 I
=
# 
间
fp
16
```

矩阵 L 和 U 分别包含每个安全盒 $\mathcal{B}_k$, $k = 1, \ldots, K$ 的下界 l_k 和上界 u_k。这些矩阵的维度为 $K \times d$，且未在上述代码中显式定义。在第 6 行中，它们用于实例化安全集 $\mathcal{S}$ （作为对象 S）。这一行执行离线预处理，即，我们构建线图并优化代表点。在第 13 行中，函数 plan 根据安全集、初始点和终止点、终止时间以及目标系数，求出一条光滑路径 p。路径所具有的连续导数数量 D 等于列表 alpha 的长度。默认情况下，B´ézier 曲线的次数设置为 $M = 2D + 1$。通过传入时间 $t \in [0, T]$，可以像调用函数一样调用路径对象 p，如第 17 行所示。（它还包含其他属性，例如 B´ézier 控制点列表，以及路径经过的安全盒 $s_1, \ldots, s_N$。）

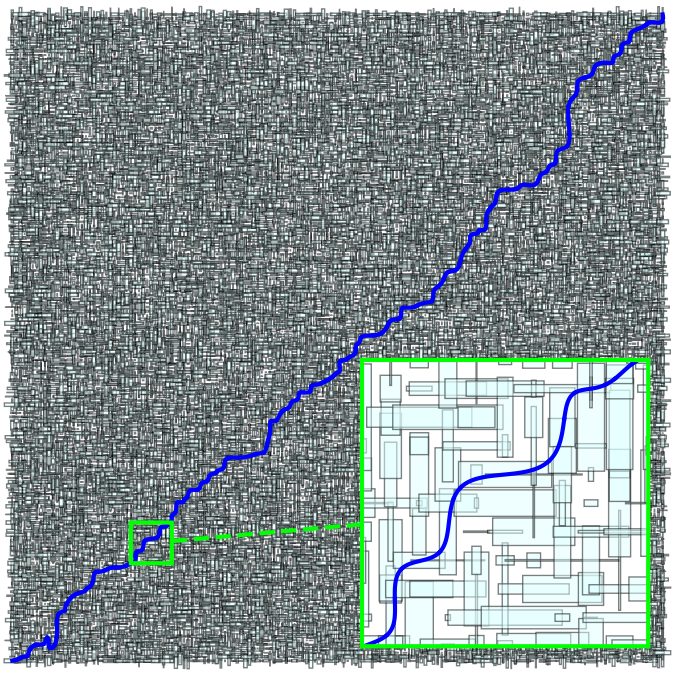


图 7. 规模扩展研究中最大的实例，包含 $K = 25{,}600$ 个安全盒，并展示了最终路径。

B. 规模扩展研究

在我们的第一个示例中，我们考虑 $d = 2$ 维度中的路径规划问题，并分析算法性能与安全盒数量 K 之间的关系。

我们按如下方式生成问题（2）的一个实例。我们构建一个包含 P^2 个具有整数坐标$\{1, \ldots, P\}^2$的点的正方形网格。将网格中的每个点作为一个安全盒 $\mathcal{B}_k$ 的中心。每个盒均以相同的概率沿水平方向或垂直方向延伸。盒的短边和长边分别从区间 $[0, 0.5]$ 和 $[0, 2]$，中均匀随机抽取。

我们使用此过程生成六个可行路径规划问题，其网格边长分别为 $P = 5$，10，20，40，80，160。这些问题中的盒子数量随后为

$$K = P^2 = 25,\ 100,\ 400,\ 1{,}600,\ 6{,}400,\ 25{,}600.$$

终止时间取为 $T = P$，代价权重为 $\alpha_1 = 0$ 和 $\alpha_2 = \alpha_3 = 1$。路径连续可微 $D = 3$ 次，贝塞尔曲线的次数为 $M = 2D + 1 = 7$。初始位置为左下方盒子的中心，$p^{\text{init}} = (1, 1)$；终止位置为右上方盒子的中心，$p^{\text{init}} = (P, P)$。这些实例中规模最大的一个（包含 $K = 25{,}600$ 个安全盒）如图 7 所示。

计算时间如图 8 所示，分为离线预处理、多边形阶段和平滑阶段。较小规模的实例可在百分之几秒或十分之几秒内求解。对于图 7 中规模最大的实例，离线处理时间为 25 秒，而在线多边形阶段和平滑阶段分别耗时 0.12 秒和 3.4 秒。考虑到一些固定开销，我们可以看到，预处理时间随着盒子数量的增加而近乎线性增长。

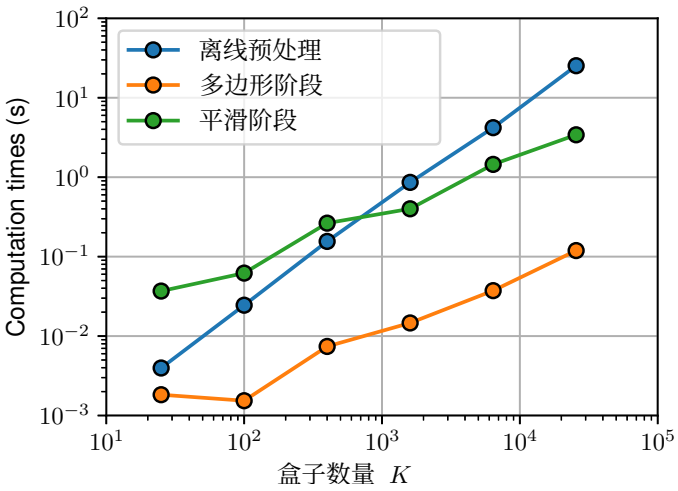


图 8. 规模扩展研究的计算时间，细分为离线预处理、多边形阶段和平滑阶段。

表 I 规模扩展研究中各实例的规模

盒子总数 K	25	100	400	1,600	6,400	25,600
顶点 $\lvert\mathcal{V}\rvert$	38	179	804	3,298	12,816	52,308
边 $\lvert\mathcal{E}\rvert$	127	708	3,583	15,613	58,351	241,348
路径盒子 N	7	12	35	53	113	241

K （双对数图中的单位斜率），而在线运行时间的增长速度则更慢。表 I 显示了本分析中所有问题的线图 G 中的顶点数$|\mathcal{V}|$ 和边数 $|\mathcal{E}|$ ，以及最终路径经过的盒子数 N 。

我们发现，在多边形阶段和平滑阶段，迭代次数基本不受问题规模的影响。多边形阶段的迭代次数在 1 到 4 次之间，平滑阶段则在 5 到 6 次之间。

C. 大型示例

在第二个示例中，我们为一架四旋翼无人机规划一条穿越多障碍物环境的路径。四旋翼无人机的构型空间是六维的：三个坐标指定质心的位置，另外三个坐标指定其朝向。然而，只要给定一条四次可微的质心路径，就总能重建出四旋翼无人机朝向的动力学可行轨迹，以及所需的控制推力 [67]。这一便利特性称为微分平坦性，借助它，我们只需在 $d = 3$ 个维度中求解路径规划问题，就能规划四旋翼无人机的飞行。

四旋翼无人机的环境如图 1 顶部所示，类似于一个拥有多栋建筑物和茂密植被的村庄。该村庄建立在一个包含 $P^2 = 50^2 = 2,500$ 个点的正方形网格上，这些点将地面划分为 $(P-1)^2$ 个边长为 1 的正方形单元格。索引为 $(i, j) \in \{1,\dots,P-1\}^2$ 的单元格，其左下角坐标为 $(i, j) \in \mathbf{R}^2$，右上角坐标为 $(i+1, j+1) \in \mathbf{R}^2$。每个单元格内都有以下障碍物之一：建筑物、灌木或树木。建筑物总数为 $9^2 = 81$ 。这些单元格每栋建筑物占据的单元格通过一次从索引为 $(i, j) \in \{5, 10, \dots, 40, 45\}^2$ 的单元格出发、长度为 5 的随机游走确定。因此，每栋建筑物最多可覆盖六个单元格，相邻建筑物也可能相连。建筑物的构造方式保证四旋翼无人机在其他单元格中飞行时不会与其发生碰撞；无人机的碰撞几何体以半径为 0.1 的球体进行保守估计。每栋建筑物的高度为 5.0。在未被建筑物占据的单元格中，以相同概率设置灌木或树木。灌木和树木位于各自单元格的中心。灌木的底面为正方形，其边长在 0.2 到 0.7 之间均匀随机选取，高度为底面边长的两倍。树木的树冠用边长为 0.8 的立方体表示。树冠中心的高度在 1.0 到 4.5 之间均匀随机选取。树干的横截面为边长 0.2 的正方形。

为构建安全集 $\mathcal{S}$ ，我们使用轴对齐盒，在每个单元格中独立分解自由空间。建筑物完全占据其所在单元格，因此这些单元格中不使用任何安全盒。灌木周围的自由空间分解为五个安全盒：灌木周围四个，顶部一个。同样，对于树木，我们在树干周围设置四个安全盒，并在树冠顶部设置一个安全盒。我们对这些盒子进行适当收缩，以计入四旋翼无人机的碰撞几何体。使用这种方法分解图 1 中的环境，共需 $K = 10,150$ 个安全盒。由此得到的线图有 $|\mathcal{V}| = 70,907$ 个顶点和 $|\mathcal{E}| = 1,022,782$ 条边。

如 [67]，所示，规划四旋翼无人机的路径时，一个自然的目标函数是四阶导数（即跃度变化率）的平方 L_2 范数。因此，我们将代价权重设为 $\alpha_1 = \alpha_2 = \alpha_3 = 0$ 和 $\alpha_4 = 1$。我们设计的路径连续可微 $D = 4$ 次，并使用次数为 $M = 2D + 1 = 9$ 的 B´ezier 曲线。终止时间设为 $T = P = 50$。四旋翼无人机从环境 $p^{\text{init}} =$ 的左下角（1, 1, 0）起飞，并在右上角 $p^{\text{init}} =$（P, P, 0）降落。根据 [67]，的结果可以看出，为使四旋翼无人机水平起飞和停止，且平动速度和角速度均为零，必须满足以下边界条件：

$$p^{(i)}(0) = p^{(i)}(T) = 0, \quad i = 1, \dots, 3.$$

为使我们的算法能够处理这些约束所需的小幅修改将在第 VIII 节中说明。

安全盒的离线预处理耗时 101 秒，其中式 (6) 中的代表点由商用求解器 MOSEK 10.0 计算得出。多边形阶段耗时 0.22 秒，并在 5 次迭代后收敛。平滑阶段耗时 7.5 秒，共迭代 8 次。最终路径中包含 135 个盒子。图 1 底部展示了四旋翼无人机沿着我们算法生成的路径飞行。四旋翼无人机飞行的视频可在 https://youtu.be/t9UWIi9NyxM 查看。

在此示例以及其他只惩罚路径跃度变化率的问题中，我们对遍历时间的初始猜测相当不准确，因此初始轨迹的代价很高。然而，平滑阶段的第一次迭代就

足以将代价降低 84.3%，最终轨迹的代价比初始轨迹低 99.3%。

D. 与混合整数优化的比较

解决问题 (2) 的一个非常自然的方法是采用混合整数（全局）优化 [12]。我们以将本方法与这些技术进行比较来结束实验。作为基准，我们采用 §III–V 中展示的简单示例，因为混合整数方法不适用于规模更大的问题。

为使用混合整数优化求解问题 (2)，我们将路径参数化为分段 B´ézier 曲线，由时长相等的 N 个子路径组成，遍历时间为 $T_j = T/N$ $j = 1, \ldots\ldots, N$。我们编写了一个混合整数规划，它与问题 (23) 相同，但遍历时间 T_j 在此处取固定值，并且用析取约束替代安全条件 (18)。该析取约束要求每个子路径 p_j 至少包含在一个安全盒 $\mathcal{B}_k$ 中，并使用二进制变量 $\sigma_{j,\,k} \in \{0,1\}$ $j = 1, \ldots\ldots, N$ 和 $k = 1, \ldots\ldots, K$ 进行编码。由于我们的安全集是轴对齐盒，因此该约束可以写成以下简单形式：

$$\sum_{k=1}^{K} l_k \sigma_{j,k} \leq p_{j,n} \leq \sum_{k=1}^{K} u_k \sigma_{j,k},$$

对于 $j = 1, \ldots\ldots, N$和 $n = 0, \ldots\ldots, M$。二进制变量还受到“独热”约束的限制

$$\sum_{k=1}^{K} \sigma_{j,k} = 1,$$

对于 $j = 1, \ldots\ldots, N$。由此得到的问题是一个混合整数二次规划（MIQP）。

我们针对分段 B´ézier 曲线中逐渐增加的子路径数量，依次求解一系列混合整数二次规划问题：$N = 9$、……、18。选择最小值 $N = K = 9$，因为最优路径可能必须经过每个安全盒。将 B´ézier 曲线的次数设为 $M = 2D + 1 = 7$ 后，混合整数方法便具有与我们的方法相同的完备性保证，即，当且仅当原始规划问题 (2) 可行时，混合整数二次规划才可行。较大的 N 值能带来更灵活的路径参数化，并降低混合整数二次规划的最优值。不过，这也会增加混合整数二次规划的求解时间；在最坏情况下，求解时间与二进制变量可能赋值的数量 K^N 成正比。请注意，由于 $N \geq K$，这种最坏情况下的运行时间会随着盒子数量 K 的增加呈超指数增长。

我们的方法为该示例设计的路径如图6右侧所示，其代价为1.0001，基本上就是问题的全局最小值（其单位代价为1）。我们算法的离线预处理（图2）、多边形阶段（图4）和平滑阶段（图6）
分别耗时1.0、1.7和14.5毫秒。将这三个阶段的总耗时（17.2毫秒）与我们路径的代价一同以黄色星号标示在图9中。

为求解混合整数二次规划问题，我们考虑三种最先进的商用求解器：CPLEX 22.1.1、Gurobi 10.0 和 MOSEK 10.0。图9报告了这些求解器的求解时间

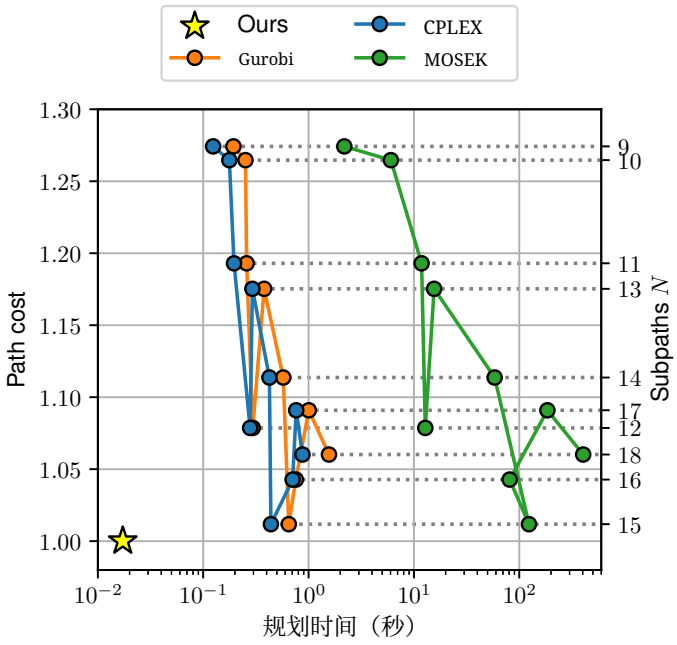


图9. 我们的算法与混合整数优化的比较。黄色星号表示我们的方法。三条曲线展示了随着路径参数化中使用的子路径数量N 增加，不同商用求解器（CPLEX、Gurobi 和 MOSEK）的性能。

以及它们的路径成本随子路径数量N 变化的情况。对于 $N = 9$，混合整数二次规划的最优值为 1.27，高于我们的方法，因为在混合整数二次规划中，每条子路径的持续时间都是固定的，求解器无法对行进时间进行精细优化。最快的求解器是 CPLEX，耗时 124 毫秒，比我们的方法慢六倍。使混合整数二次规划达到最低最优值的子路径数量为 $N = 15$。该最优值为 1.01，虽然更接近，但仍高于我们方法找到的路径成本。在这种情况下，CPLEX 同样是最快的求解器，耗时 439 毫秒（比我们的算法慢 26 倍）。

正如预期的那样，随着问题规模增大，混合整数方法很快就变得不切实际。例如，通过混合整数二次规划求解规模扩展研究§VII-B中最小的问题（具有 $N = K$ 和 $M = 2D + 1$），我们得到的路径成本约为我们的方法的三分之一。然而，我们的规划器仅耗时43毫秒（包括离线预处理），而CPLEX求解该混合整数二次规划耗时584秒，并且在找到成本低于我们的方法的可行路径之前，分支定界耗时25秒。其他求解器则更慢。

VIII. 扩展

最后，我们简要介绍如何将本文提出的技术扩展到更一般的路径规划问题。

初始和最终导数：我们的方法可以轻松处理路径导数的边界值。只需像处理式 (16) 中的端点约束那样，在问题 (23) 中使用控制点 $p_1^{(i)},\,0$ 和 $p_N^{(i)},\,M-i$ 指定这些边界值，其中 $i \in \{1, \ldots, D\}$。

当导数存在边界条件时，我们的算法还需要进行一项额外修改，这关系到 §V-C 中遍历时间的估算。初始化遍历时间时，不再以恒定速度走完整条多边形曲线$\mathcal{C}$，而只以恒定速度走 $\mathcal{C}$的中间部分；对于第一段和最后一段，则将能提供足够变量以满足边界条件和可微性约束的最低阶导数设为常数。例如，在 §VII-C 的四旋翼问题中，为了找到一种时间参数化，使其前三阶导数在端点处为零，且连续可微四次，我们需要在$\mathcal{C}$ 的起始段和末尾段令七阶导数保持恒定。

最后，我们注意到，如果对导数施加边界条件，则可能需要增加第一条和最后一条 B´ézier 曲线的次数，以保持我们算法的完备性。例如，如果初始位置靠近安全集的边界，且初始速度指向外部，则可能需要许多控制点来设计一个不会离开安全集的急转弯。

凸安全集：假设安全集是轴对齐盒，这对我们算法的离线部分非常方便，因为可以非常高效地求出一组盒子之间的两两交集 [60]。在多边形阶段，我们也利用了这一假设，具体体现在多重刺穿问题以及 §IV-B 中对盒序列的改进上。对于更一般的凸安全集，这些计算会更加复杂，并可能显著降低我们算法的速度。例如，检查两个凸集是否相交需要求解一个凸优化问题，例如，当这些集合为多面体时，则需要求解一个线性规划。然而，如果每个凸安全集都配备一个轴对齐包围盒，则可以恢复我们方法的部分效率。

未指定终止时间：在某些应用中，指定固定的终止时间 T 并不容易，因此最好让规划算法自动选择该值。在这些情况下，我们还会对 T 施加惩罚（例如，采用固定权重 $\alpha_0 > 0$的线性代价$\alpha_0 T$ ），以防止原始目标函数 J 使终止时间变得任意长。我们可以非常自然地将此方法扩展到这类问题。在 §V-C 中初始化遍历时间时，现在还需要对路径的总持续时间作出初始猜测。随后，通过求解 §V-C 中的切线问题来改进这一猜测；此时终止时间 T 成为线性约束 (21) 中的一个变量，且代价函数包含时间惩罚（例如， $\alpha_0 T$）。

导数约束：在我们的框架中，也很容易纳入对路径导数的凸约束。事实上，路径导数是分段 B´ezier 曲线，与式（18）中的安全约束类似，通过约束其控制点，可以确保它们在任何时刻都位于某个凸集中。若终止时间 T 固定，加入这些约束会破坏我们算法的完备性。具体而言，§II-C 中的可行性论证不再成立，即使原始路径规划问题可行，我们在式（23）中对分段 B´ezier 路径的优化也可能不可行。然而，如果像上文所述，让 T 成为一个优化变量，那么算法完备性将得以恢复。这是因为，沿着一条足够缓慢的曲线行进，就能满足任何包含原点于其内部的导数约束。

多个路径点：在某些路径规划问题中，我们需要设计一条单一的光滑路径，使其按顺序插值或经过给定的一系列中间路径点。为将此方法扩展到这类问题，我们重复多边形阶段中的步骤，以连接每一对相邻路径点，从而得到一条满足所有插值约束的多边形曲线。同样，在平滑阶段，我们将多个形式为 (23) 的问题合并为一个程序，其中每段分段 B´ézier 曲线具有固定端点，并受到与相邻曲线平滑连接的约束。整体路径经过各路径点的时间随后由平滑阶段自动确定。最后，只需要求路径满足$p^{(i)}(0) = p^{(i)}(T)$, $i = 0, \ldots\ldots, D$，即可生成经过所有路径点的周期轨迹。这些条件可立即转化为关于初始和最终 B´ézier 子路径控制点的线性约束。

致谢

本研究得到美国海军研究办公室（ONR）项目编号 N00014-22-1-2121 的资助。事实上，本研究正是该资助促成的合作所直接带来的成果。Parth Nobel 部分获得美国国家科学基金会研究生研究奖学金计划项目编号 DGE-1656518 的支持。本文所表达的任何意见、研究结果、结论或建议均属于作者个人观点，不一定反映美国国家科学基金会的立场。Stephen Boyd 部分获得由 InnoHK 资助、位于中国香港特别行政区的新兴智能系统人工智能芯片中心 ACCESS 的支持。

参考文献

[1] S. LaValle 和 R. Sharma，“变化且部分可预测环境中的运动规划”，《国际机器人研究杂志》，第 16 卷，第 6 期，第 775–805 页，1997 年。[2] P. Fiorini 和 Z. Shiller，“利用速度障碍在动态环境中进行运动规划”，《国际机器人研究杂志》，第 17 卷，第 7 期，第 760–772 页，1998年。[3] S. Petti 和 T. Fraichard，“动态环境中的安全运动规划”，载于智能机器人与系统国际会议。IEEE/RJS，2005年，第 2210–2215 页。[4] N. Du Toit 和 J. Burdick，“动态、不确定环境中的机器人运动规划”，*IEEE*《机器人学汇刊》，第 28 卷，第 1 期，第 101–115 页，2011年。[5] E. Frazzoli、M. Dahleh 和 E. Feron，“敏捷自动驾驶车辆的实时运动规划”，《制导、控制与动力学杂志》，第 25 卷，第 1 期，第 116–129 页，2002年。[6] Y. Kuwata、J. Teo、G. Fiore、S. Karaman、E. Frazzoli 和 J. P. How，“面向城市自动驾驶的实时运动规划”，*IEEE*《控制系统技术汇刊》，第 17 卷，第 5 期，第 1105–1118 页，2009年。[7] C. Katrakazas、M. Quddus、W.-H. Chen 和 L. Deka，“自主道路驾驶的实时运动规划方法：现状与未来研究方向”，《交通研究*C*辑：新兴技术》，第 60 卷，第 416–442 页，2015年。[8] D. Sadigh、S. Sastry、S. A. Seshia 和 A. D. Dragan，“利用对人类行为影响的自动驾驶汽车规划”，载于机器人学：科学与系统，第 2 卷。美国密歇根州安娜堡，2016年，第 1–9 页。[9] L. Wang、A. D. Ames 和 M. Egerstedt，“无碰撞多机器人系统的安全屏障证书”，*IEEE*《机器人学汇刊》，第 33 卷，第 3 期，第 661–674 页，2017年。

[10] A. Liniger 和 J. Lygeros，“自主竞速的非合作博弈方法”，IEEE《控制系统技术汇刊》，第 28 卷，第 3 期，第 884–897 页，2019年。[11] R. Spica、E. Cristofalo、Z. Wang、E. Montijano 和 M. Schwager，“面向自主双人无人机竞速的实时博弈论规划器”，IEEE《机器人学汇刊》，第 36 卷，1403，2020 5，1389 no. pp. – 。[12] R. Deits 和 R. Tedrake，“杂乱环境中无人机的高效混合整数规划”，载于国际机器人与自动化会议。IEEE，2015年，第 42–49 页。[13] T. Marcucci、M. Petersen、D. von Wrangel 和 R. Tedrake，“利用凸优化绕开障碍物进行运动规划”，《科学·机器人学》，第 8 卷，第 84 期，文章编号 eadf7843，2023年。[14] J.-M. Lien 和 N. Amato，“多边形的近似凸分解”，载于第二十届计算几何年度研讨会论文集，2004年，第 17–26 页。[15] N. Ayanian 和 V. Kumar，“含障碍物环境中机器人群组的抽象与控制器”，载于国际机器人与自动化会议。IEEE，2010年。[16] M. Ghosh、N. M. Amato、Y. Lu 和 J.-M. Lien，“利用相对凹度进行快速近似凸分解”，《计算机辅助设计》，第 45 卷，第 2 期，第 494–504 页，2013年。[17] R. Deits 和 R. Tedrake，“通过半定规划计算无障碍自由空间中的大型凸区域”，载于《机器人算法基础 XI》。Springer，2015年，第 109–124 页。[18] P. Werner、A. Amice、T. Marcucci、D. Rus 和 R. Tedrake，“利用可见性图的团覆盖，以少量凸集近似机器人构型空间”，arXiv 预印本 arXiv:2310.02875，2023年。[19] A. Amice、H. Dai、P. Werner、A. Zhang 和 R. Tedrake，“为机器人操作臂寻找并优化构型空间中经过认证的无碰撞区域”，载于《机器人算法基础 XV》。Springer，2022年，第 328–348 页。[20] M. Verghese、N. Das、Y. Zhi 和 M. Yip，“用于机器人规划与控制中可扩展代理碰撞检测的构型空间分解”，IEEE《机器人与自动化快报》，第 7 卷，第 2 期，第 3811–3818 页，2022年。[21] H. Dai、A. Amice、P. Werner、A. Zhang 和 R. Tedrake，“无碰撞构型空间的认证多面体分解”，arXiv 预印本 arXiv:2302.12219，2023年。[22] M. Petersen 和 R. Tedrake，“利用非线性规划扩展构型空间中的凸无碰撞区域”，arXiv 预印本 arXiv:2303.14737，2023年。[23] Y. Wang 和 S. Boyd，“利用在线优化实现快速模型预测控制”，IEEE《控制系统技术汇刊》，第 18 卷，第 2 期，第 267–278 页，2009年。[24] S. LaValle，《规划算法》。剑桥大学出版社，2006年。[25] T. Marcucci、J. Umenberger、P. Parrilo 和 R. Tedrake，“凸集图中的最短路径”，arXiv 预印本 arXiv:2101.11565v4，2021年。[26] T. Schouwenaars、B. De Moor、E. Feron 和 J. How，“多车辆路径规划的混合整数规划”，载于欧洲控制会议。IEEE，2001年，第 2603–2608 页。[27] A. Richards、J. Bellingham、M. Tillerson 和 J. How，“多架无人机的协调与控制”，载于AIAA制导、导航与控制会议及展览，2002年，第 4588 页。[28] D. Mellinger、A. Kushleyev 和 V. Kumar，“异构四旋翼团队的混合整数二次规划轨迹生成”，载于国际机器人与自动化会议。IEEE，2012年，第 477–483 页。[29] F. Augugliaro、A. Schoellig 和 R. D’Andrea，“四旋翼机队无碰撞轨迹生成：序列凸规划方法”，载于智能机器人与系统国际会议。IEEE/RJS，2012年，第 1917–1922 页。[30] J. Schulman、Y. Duan、J. Ho、A. Lee、I. Awwal、H. Bradlow、J. Pan、S. Patil、K. Goldberg 和 P. Abbeel，“使用序列凸优化和凸碰撞检测进行运动规划”，《国际机器人研究杂志》，第 33 卷，第 9 期，第 1251–1270 页，2014年。[31] X. Liu 和 P. Lu，“通过凸优化求解非凸最优控制问题”，《制导、控制与动力学杂志》，第 37 卷，第 3 期，第 750–765 页，2014年。[32] A. Majumdar 和 R. Tedrake，“用于实时鲁棒反馈运动规划的漏斗库”，《国际机器人研究杂志》，第 36 卷，第 8 期，第 947–982 页，2017年。[33] R. Bonalli、A. Cauligi、A. Bylard 和 M. Pavone，“Gusto：通过序列凸规划实现有保证的序列轨迹优化”，载于2019年国际机器人与自动化会议（ICRA）。IEEE

，2019，第 6741–6747 页。

[34] X. Zhang、A. Liniger 和 F. Borrelli，《基于优化的碰撞规避》，IEEE《控制系统技术汇刊》，第 29 卷，第 3 期，第 972–983 页，2020年。[35] L. Kavraki、P. Svestka、J.-C. Latombe 和 M. Overmars，《高维构型空间中用于路径规划的概率路线图》，IEEE机器人与自动化汇刊，第 12 卷，第 4 期，第 566–580 页，1996年。[36] S. LaValle，《快速探索随机树：一种新的路径规划工具》，TR 98-11，爱荷华州立大学计算机科学系，1998年。[37] S. Karaman 和 E. Frazzoli，《用于最优运动规划的基于采样的算法》，《国际机器人研究杂志》，第 30 卷，第 7 期，第 846–894 页，2011年。[38] N. Ratliff、M. Zucker、J. A. Bagnell 和 S. Srinivasa，《CHOMP：用于高效运动规划的梯度优化技术》，载于 2009年 IEEE机器人与自动化国际会议。IEEE，2009年，第 489–494 页。[39] M. Kalakrishnan、S. Chitta、E. Theodorou、P. Pastor 和 S. Schaal，《STOMP：用于运动规划的随机轨迹优化》，载于 2011年 IEEE机器人与自动化国际会议。IEEE，2011年，第 4569–4574 页。[40] K. Hauser，《学习问题-最优解映射：分析及其在机器人学全局优化中的应用》，IEEE《机器人学汇刊》，第 33 卷，第 1 期，第 141–152 页，2016年。[41] S. Boyd 和 L. Vandenberghe，《凸优化》。剑桥大学出版社，2004年。[42] C. Belta、V. Isler 和 G. J. Pappas，《多边形环境中机器人运动规划与控制的离散抽象》，IEEE《机器人学汇刊》，第 21 卷，第 5 期，第 864–874 页，2005年。[43] R. Alur、T. A. Henzinger、G. Lafferriere 和 G. J. Pappas，《混杂系统的离散抽象》，《IEEE会刊》，第 88 卷，第 7 期，第 971–984 页，2000年。[44] R. Tedrake、I. Manchester、M. Tobenkin 和 J. Roberts，《LQR树：通过平方和验证实现反馈运动规划》，《国际机器人研究杂志》，第 29 卷，1052，2010年 8，1038 期，第 – 页。[45] S. Singh、A. Majumdar、J.-J. Slotine 和 M. Pavone，《通过收缩理论和凸优化实现鲁棒在线运动规划》，载于 2017年 IEEE机器人与自动化国际会议（ICRA）。IEEE，2017年，第 5883–5890 页。[46] A. D. Ames、X. Xu、J. W. Grizzle 和 P. Tabuada，《面向安全关键系统的基于控制屏障函数的二次规划》，IEEE自动控制汇刊，第 62 卷，第 8 期，第 3861–3876 页，2016年。[47] A. Weiss、F. Leve、M. Baldwin、J. R. Forbes 和 I. Kolmanovsky，《在 $SO(3) \times \mathbb{R}^3$ 上使用正不变约束可容许集进行航天器受约束姿态控制》，载于 2014年美国控制会议。IEEE，2014年，第 4955–4960 页。[48] A. Weiss、C. Petersen、M. Baldwin、R. S. Erwin 和 I. Kolmanovsky，《用于航天器障碍规避的安全正向不变集》，《制导、控制与动力学杂志》，第 38 卷，第 4 期，第 720–732 页，2015年。[49] K. Berntorp、R. Bai、K. F. Erliksson、C. Danielson、A. Weiss 和 S. Di Cairano，《用于安全集成车辆运动规划与控制的正不变集》，IEEE智能车辆汇刊，第 5 卷，第 1 期，第 112–126 页，2019年。[50] C. Danielson、K. Berntorp、A. Weiss 和 S. Di Cairano，《使用不变集运动规划器为受扰动的不确定系统进行鲁棒运动规划》，IEEE自动控制汇刊，第 65 卷，第 10 期，第 4456–4463 页，2020年。[51] J.-w. Choi、R. Curry 和 G. Elkaim，《基于贝塞尔曲线的自主地面车辆路径规划》，载于《电气与电子工程进展》——2008年世界工程与计算机科学大会 IAENG 特刊。IEEE，2008年，第 158–166 页。[52] M. Flores，《使用非均匀有理B样条基函数为受约束非线性动力系统进行实时轨迹生成》。加州理工学院，2008年。[53] B. Lau、C. Sprunk 和 W. Burgard，《使用样条为移动机器人进行运动学-动力学规划》，载于智能机器人与系统国际会议。IEEE/RJS，2009年，第 2427–2433 页。[54] M. Elbanhawi、M. Simic 和 R. Jazar，《使用B样条曲线为类汽车机器人进行连续路径平滑》，《智能与机器人系统杂志》，第 80 卷，第 23–56 页，2015 年。[55] J. Park 和 H. J. Kim，《使用相对安全飞行走廊为多架四旋翼飞行器进行快速轨迹规划》，载于智能机器人与系统国际会议。IEEE/RSJ，2019年，第 596–603 页。[56] F. Koolen，《使用非线性质心模型进行人形机器人的平衡控制与运动规划》，博士论文，马萨诸塞州理工学院。

，2020年。

[57] J. Tordesillas、B. T. Lopez、M. Everett 和 J. P. How，《Faster：面向未知环境导航的快速安全轨迹规划器》，*IEEE*《机器人学汇刊》，第 38 卷，第 2 期，第 922–938 页，2021年。[58] N. Csomay-Shanklin、A. Taylor、U. Rosolia 和 A. Ames，《不确定非线性系统的多速率规划与控制：模型预测控制和控制李雅普诺夫函数》，*arXiv* 预印本 *arXiv:2204.00152*，2022年。[59] ¨O. Arslan 和 A. Tiemessen，《自适应贝塞尔次数约简与分割：实现计算高效的运动规划》，*IEEE*《机器人学汇刊》，第 38 卷，第 6 期，第 3655–3674 页，2022年。[60] A. Zomorodian 和 H. Edelsbrunner，《用于盒相交的快速软件》，载于第十六届计算几何年度研讨会论文集，2000年，第 129–138 页。[61] M. Lobo、L. Vandenberghe、S. Boyd 和 H. Lebret，《二阶锥规划的应用》，《线性代数及其应用》，第 284 卷，第 1–3 期，第 193–228 页，1998 年。[62] P. Virtanen 等，《SciPy 1.0：Python科学计算的基础算法》，《自然方法》，第 17 卷，第 261–272 页，2020年。[63] R. Farouki 和 V. Rajan，《伯恩斯坦形式多项式的算法》，《计算机辅助几何设计》，第 5 卷，第 1 期，第 1–26 页，1988年。[64] S. Diamond 和 S. Boyd，《CVXPY：一种嵌入 Python 的凸优化建模语言》，《机器学习研究杂志》，第 17 卷，第 83 期，第 1–5 页，2016年。[65] A. Hagberg、D. Schult 和 P. Swart，《使用 NetworkX 探索网络结构、动态特性和功能》，载于第七届 *Python*科学会议论文集，G. Varoquaux、T. Vaught 和 J. Millman 编，帕萨迪纳（美国加利福尼亚州），2008年，第 11–15 页。[66] P. Goulart 和 Y. Chen，《Clarabel 求解器文档》，2023年。[在线]。网址：https://oxfordcontrol.github.io/ClarabelDocs/stable/ [67] D. Mellinger 和 V. Kumar，《四旋翼飞行器的最小跃度变化率轨迹生成与控制》，载于国际机器人与自动化会议。IEEE，2011年，第 2520–2525 页。

附录 A

在本附录中，我们推导不等式（8），它们用于多边形阶段，以改进曲线 $\mathcal{C}$经过的盒序列。

为简化符号，在本附录中我们令 $a = y_{j-1}$、$b = y_{j+1}$和 $y = y_j$。此外，我们用 l 和 u 表示界定轴对齐盒 $\mathcal{B}_{s_j} \cap \mathcal{B}_{s_{j+1}}$的下界和上界。同样，我们令 l_1 和 u_1 界定盒 $\mathcal{B}_{s_j} \cap \mathcal{B}_k$，并令 l_2 和 u_2 界定盒 $\mathcal{B}_k \cap \mathcal{B}_{s_{j+1}}$。我们比较图 3 所示两个问题的最优值。第一个是

$$\begin{array}{ll} \text{minimize} & \|y-a\|_2 + \|b-y\|_2 \\ \text{subject to} & l \le y \le u, \end{array}$$

其中唯一的变量是 y。第二个是

$$\begin{array}{ll} \text{minimize} & \|z_1-a\|_2 + \|z_2-z_1\|_2 + \|b-z_2\|_2 \\ \text{subject to} & l_1 \le z_1 \le u_1, \quad l_2 \le z_2 \le u_2, \end{array} \tag{26}$$

其中变量为 z_1 和 z_2。令 $y^\star$ 为第一个问题的解，我们已知该解，因为我们已经求解了（7）。我们想检查对于第二个问题，选择 $z_1 = z_2 = y^\star$ 是否最优。为此，我们寻找满足互补松弛条件且对偶可行的（26）问题的拉格朗日乘子 [41,§5.5]。

互补松弛条件为

$$\begin{aligned} \lambda_1 = \tfrac{y^\star - a}{\|y^\star - a\|_2}, \quad (y^\star - l_1)^T \nu_1^+ = (y^\star - l_2)^T \nu_2^+ = 0, \\ \lambda_2 = \tfrac{b - y^\star}{\|b - y^\star\|_2}, \quad (u_1 - y^\star)^T \nu_1^- = (u_2 - y^\star)^T \nu_2^- = 0, \end{aligned}$$

其中，乘子 λ_1、$\lambda_2 \in \mathbf{R}^d$ 分别与式（26）中的第一个和最后一个目标函数项配对，ν_1^+、$\nu_1^- \in \mathbf{R}^d$ 与第一个盒约束中的下限和上限配对，而 ν_2^+、$\nu_2^- \in \mathbf{R}^d$ 与第二个盒约束配对。问题（26）的对偶问题约束为

$$\begin{gathered} \nu_1^+,\ \nu_1^-,\ \nu_2^+,\ \nu_2^- \ge 0, \\ \|\lambda\|_2,\ \|\lambda_1\|_2,\ \|\lambda_2\|_2 \le 1, \\ \lambda - \lambda_1 + \nu_1^+ + \nu_1^- = \lambda_2 - \lambda + \nu_2^+ + \nu_2^- = 0, \end{gathered}$$

其中，乘子 $\lambda \in \mathbf{R}^d$ 与式（26）中的第二个成本项相对应。

我们令 L_1 和 U_1 为分别选取满足 $l_1 < y^\star$ 和 $y^\star < u_1$的条目的矩阵。对于限制 l_2 和 u_2，我们以类似方式定义 L_2 和 U_2。经过一些变换，上述两组条件可归结为式（8）中的不等式。唯一的变量是 λ，因为根据互补松弛条件，λ_1 和 λ_2 的值是固定的（且已知）。

最后，我们注意到，式（9）中的拉格朗日乘子 $\lambda^\star$ 的范数可以解释为图 3 中点 z_1 和 z_2 之间交换的弹性力，并且反映了允许这些点分离所产生的成本下降。这启发我们采用这样的启发式方法：插入向量 $\lambda^\star$ 范数最大的盒。