arXiv:2304.01765v1 [math.OC] 4 Apr 2023

有向图上多智能体路径寻找解的局部优化

S. Ardizzoni[1], I. Saccani[1],L. Consolini[1],M. Locatelli[1]1意大利帕尔马大学工程与建筑系，Parco Area delle Scienze,181/A，帕尔马，意大利

摘要——**在次优多智能体路径寻找求解器中，基于规则的算法尤其具有吸引力，因为它们是完备的。即使在拥挤场景中，它们也能找到可行解，使每个智能体到达其目标位置，并避免出现死锁情形。然而，基于规则的算法通常给出的解远长于最优解。本文的主要贡献是提出一种用于多智能体路径寻找的迭代局部搜索过程。我们从一个可行的次优解出发，在该解的邻域内进行局部搜索，以寻找更短的解。我们反复执行这一过程，直到解无法再缩短为止。最终得到的解仍然是次优解，但总体而言，其质量远高于初始解。我们采用动态规划来执行局部搜索过程。在这方面，搜索具有局部性对于降低算法的时间复杂度至关重要。事实上，若采用标准动态规划，所探索状态的数量会随智能体数量呈指数级增长。我们将看到，引入局部性约束后，局部动态规划问题的求解时间相对于智能体数量仅呈多项式增长。**

一、引言

我们研究多智能体路径寻找（MAPF）问题。考虑一个有向图和一组智能体。每个智能体占据一个不同的节点，并可移动到空闲节点，即未被其他智能体占据的节点。多智能体路径寻找问题旨在计算一个移动序列，使所有智能体抵达分配给它们的目标节点，同时避免碰撞。研究动机主要来自自动导引车（AGV）车队的管理。自动导引车在仓库的不同位置之间运送物品。每辆自动导引车沿着预定义路径行驶，这些路径连接存放或处理物品的位置。我们将这些路径的布局表示为一个有向图。节点表示物品被取放的位置，以及用于路径规划的其他位置。有向弧表示连接这些位置的预先计算路径。当多辆自动导引车在狭小场景中行驶时，每辆车都会成为其他车辆的障碍物。在某些情况下，车队可能陷入死锁，导致所有车辆都无法抵达目标位置。因此，即使在拥挤的配置下，找到多智能体路径寻找问题的可行解也十分重要。

文献综述。许多研究致力于寻找多智能体路径寻找（MAPF）的最优解（即移动次数最少的解）。例如，基于冲突的搜索（CBS）是一种两层算法，它根据各个智能体之间的冲突构建搜索树（见 [9]）。然而，寻找多智能体路径寻找问题的最优解是 NP-hard 的（见 [12]），且计算时间会随着智能体数量的增加而呈指数级增长。因此，当智能体数量相对较少时，通常采用最优求解器；反之，当智能体数量较多时，通常采用次优求解器。在这种情况下，目标是快速为各个智能体找到路径，而要保证给定的解是最优解往往难以实现。其中，基于搜索的求解器旨在提供高质量的解，但并不完备（即它们不一定总能找到可行解）。一个典型例子是分层合作式 A* （HCA*） [10], ，其中智能体按照预先设定的顺序逐一规划。相比之下，基于规则的方法针对不同情形制定了特定的移动规则。它们优先考虑以较低的计算成本保证完备性，而非解的质量。该领域最早的重要成果之一来自 Kornhauser 的论文 [6],，其中提出了一种基于规则的程序，用于解决多智能体路径寻找问题（或判定该问题不存在可行解）。Kornhauser 表明，使用其方法找到的解具有立方级路径长度复杂度。然而，据我们所知，该程序从未得到完整实现。近期两种相关的基于规则的算法是 TASS [5]和*Push andRotate* [11] [1]。TASS 是一种基于树的智能体交换策略，在任意树上都具有完备性；而*PushandRotate*能够求解至少包含两个空位（即未占用顶点）的图上的所有多智能体路径规划实例。文献 [7]提出了一种将图转换为树的方法（如 [4]所示），并使用 TASS 求解转换后的问题。基于规则的算法也适用于至少具有两个未占用节点的有向图。具体而言，文献 [2] 提出了 diSC 算法，该算法能够求解强连通有向图上的任意多智能体路径规划实例；所谓强连通有向图，是指从任一节点出发都能到达其他任一节点的有向图。另一篇相关文献是 [3], ，其求解的是一类特定的双连通有向图上的多智能体路径寻找问题；这类图是指强连通有向图，忽略边的方向后得到的无向图没有割点。所提出的 diBOX 算法关于节点数具有多项式复杂度。

研究动机与贡献陈述。

在次优多智能体路径寻找求解器中，基于规则的算法尤其具有吸引力，因为它们是完备的。即使在拥挤场景中，它们也能找到可行解，使每个智能体都到达其目标位置，从而避免死锁情形。然而，基于规则的算法通常给出的解远长于最优解。这是其在工业应用中的一项关键限制。

本文的主要贡献是在多智能体路径寻找中引入迭代局部搜索过程。我们从

一个可行的次优解开始，例如基于规则的算法所提供的解。我们在该解的邻域中执行局部搜索，以寻找更短的解。随后，我们迭代地重复这一过程，直到无法再缩短该解为止。最终，我们得到一个仍然是次优的解，但通常其质量远高于初始解。

我们使用动态规划来执行局部搜索过程。在这方面，搜索具有局部性这一特性，对于降低算法的时间复杂度至关重要。事实上，原则上可以通过动态规划解决一般的多智能体路径寻找问题。然而，探索状态的数量会随着智能体数量呈指数级增长，因此我们无法将标准动态规划应用于涉及大量智能体的问题。正如我们将看到的，引入局部性约束后，可以在时间复杂度仅随智能体数量多项式增长的情况下，解决（局部）动态规划问题（见定理 IV.1）。

II. 问题定义

A. 多智能体路径寻找问题

设 $G = (V, E)$ 是一个有向图，其顶点集为 V，边集为 E。我们为每个智能体分配一个唯一标签，并令集合 P 包含这些标签。一个**配置**是一个函数 $\mathcal{A}$: $P \to V$，它为每个智能体指定其占据的顶点。如果一个配置是单射（即每个顶点至多被一个智能体占据），则称其为有效配置。集合 $\mathcal{C} \subset \{P \to V\}$ 表示所有有效配置。

假设时间已离散化。在每个时间步，每个智能体占据一个顶点并执行单个动作。动作有两种类型：等待和移动。我们用 ι 表示等待动作。执行此动作的智能体会在当前顶点再停留一个时间步。我们用 $u \to v$ 表示移动动作。在这种情况下，智能体从当前顶点u 移动到相邻顶点 v（即 $(u, v) \in E$）。因此，单个智能体的所有可能动作集合为

$$\bar{E} = E \cup \{\iota\}.$$

函数 $\rho: \mathcal{C} \times E \to \mathcal{C}$ 是一个部分定义的转移函数，使得 $\mathcal{A}' = \rho(\mathcal{A}, u \to v)$ 表示将一个智能体从 u 移动到 v 后得到的配置：

$$\mathcal{A}'(q) := \begin{cases} v, & \text{if } \mathcal{A}(q) = u; \\ \mathcal{A}(q), & \text{otherwise}. \end{cases} \tag{1}$$

记号 $\rho(\mathcal{A}, u \to v)!$ 表示该函数定义良好。换句话说，当且仅当 $(u, v) \in E$ 且$\mathcal{A}' \in \mathcal{C}$ 时，$\rho(\mathcal{A}, u \to v)!$。此外，$(\forall \mathcal{A} \in \mathcal{C})\ \rho(\mathcal{A}, \iota)!$ 且 $\rho(\mathcal{A}, \iota) = \mathcal{A}$。

由于智能体的移动可以是同步的，因此在每个时间步，一个动作是 $\mathcal{E} = \bar{E}^{|P|}$, $a = (a_1, \ldots, a_{|P|})$ 中的一个元素，其中 a_i 是智能体 i 的单步移动。我们可以将函数 $\rho: \mathcal{C} \times E \to \mathcal{C}$ 扩展为 $\rho: \mathcal{C} \times \mathcal{E} \to \mathcal{C}$，方法是令$\mathcal{A}' = \rho(\mathcal{A}, a)$ 等于沿边 a_i 移动智能体 i 后得到的配置（若 $a_i = \iota$，则智能体不移动）。在这种情况下，$(\forall a \in \mathcal{E}, \mathcal{A} \in \mathcal{C})\ \rho(\mathcal{A}, a)!$ 当且仅当满足以下条件：

1) $\mathcal{A}' \in \mathcal{C}$：两个或更多智能体不能在同一时间步占据同一个顶点；
2) $\forall i = 1, \ldots, |P|$，如果 $a_i = (u, v)$，则 $\nexists j \in \{1, \ldots, |P|\}$使得 $a_j = (v, u)$：两个智能体不能在单个时间步内交换位置。

我们将规划表示为动作的有序序列。将 $\mathcal{E}$ 中的元素视为一种语言的符号会很方便。我们用 $\mathcal{E}^*$ 表示 $\mathcal{E}$的克莱尼星号，即由 $\mathcal{E}$ 中的元素组成的、长度任意的有序序列集合，以及空串 ϵ:

$$\mathcal{E}^* = \bigcup_{i=1}^{\infty} \mathcal{E}^i \cup \{\epsilon\}.$$

我们将函数 $\rho: \mathcal{C} \times \mathcal{E} \to \mathcal{C}$ 扩展为 $\rho: \mathcal{C} \times \mathcal{E}^* \to \mathcal{C}$。($\forall s \in \mathcal{E}^*, e \in \mathcal{E}, \mathcal{A} \in \mathcal{C}$) $\rho(\mathcal{A}, se)!$ 当且仅当 $\rho(\mathcal{A}, s)!$ 且 $\rho(\rho(\mathcal{A}, s), e)!$，并且如果 $\rho(\mathcal{A}, se)!$，则 $\rho(\mathcal{A}, se) = \rho(\rho(\mathcal{A}, s), e)$。

注意，ϵ 是一个平凡规划，它使所有智能体和空位都保持在各自的位置。

我们用 $\mathcal{E}^*_{\mathcal{A}}$ 表示满足 $\rho(\mathcal{A}, f)$ 有定义的规划集合:

$$\mathcal{E}^*_{\mathcal{A}} = \{f \in \mathcal{E}^* : \rho(\mathcal{A}, f)!\}.$$

检测是否存在可行解的问题如下:

定义 *2.1*：（**可行性多智能体路径寻找问题**）。给定一个有向图 $G = (V, E)$、一个智能体集合 P、一个初始有效配置 $\mathcal{A}^s$ 和一个最终有效配置 $\mathcal{A}^t$，求一个规划 f，使得 $\mathcal{A}^t = \rho(\mathcal{A}^s, f)$。

现在，对于一个可行规划 f，我们将 $|f|$ 定义为规划 f 的长度，即所有智能体通过规划 f 到达最终配置所需的时间步数。此外，给定 $k \in \mathbb{N}$，我们用 f_k 表示 f 的第 k 个前缀（也就是 f 的长度为 k 的前缀，由 f 的前 k 个动作组成）。注意，$|f_k| = k$。

我们的目标是在最小化全局累积代价函数的同时，解决给定的多智能体路径规划实例。我们采用称为最大完工时间的代价函数，其值等于最后一个智能体到达目的地的时间（即各智能体个体代价的最大值）。

定义 *2.2*：（**优化多智能体路径规划问题**）。给定 $\mathcal{A}^s$和 $\mathcal{A}^t$ 在有向图 G上的初始和最终有效配置，具有最大完工时间的优化多智能体路径规划问题定义为

$$\begin{aligned} &\min |f| \\ &\text{s.t. } \mathcal{A}^t = \rho(\mathcal{A}^s, f) \\ &f \in \mathcal{E}^*_{\mathcal{A}^s}. \end{aligned} \tag{2}$$

文献中还使用了其他代价函数。代价总和例如，是所有智能体到达目标位置且之后不再离开的时间步数之和。不幸的是，研究表明，求得最优解，即最小的最大完工时间 或代价总和，是 NP-hard 的 [12]。因此，本文中

我们提出一种方法，可在多项式时间内找到质量较高的次优解。

B. 距离

如前所述，我们提出了一种基于探索参考计划的邻域的解法。为定义邻域，我们引入顶点、配置和规划之间的距离。设 $G = (V, E)$ 为一个有向图，P 为一组智能体。我们将顶点 u 到顶点 v 的距离定义为 G 中从 v 到 u 的最短路径的长度：

$$d : V \times V \to \mathbb{N} \qquad d(u, v) = \ell(\pi_{vu}),$$

其中，π_{vu} 是 G 中从 v 到 u 的最短路径，而 $\ell(\pi_{vu})$ 是该路径的长度，定义为 π_{vu} 中边的数量。注意，d 不是对称的，因为 π_{uv} 和 π_{vu} 可能不同。接下来，我们将配置 $\mathcal{A}^1$ 到配置 $\mathcal{A}^2$ 的距离定义为两个配置中各智能体所在顶点之间距离的总和：

$$d : \mathcal{C} \times \mathcal{C} \to \mathbb{N} \qquad d(\mathcal{A}^1, \mathcal{A}^2) = \sum_{p \in P} d(\mathcal{A}^1(p), \mathcal{A}^2(p)).$$

最后，我们定义 $\mathcal{E}^*$ 中两个规划之间的非对称距离。为此，我们将每个规划关联到 $\mathcal{L} = \{\psi$ 中的一个函数：$\mathbb{N} \to \mathcal{C}\}$，如下所示

$$\Phi_{\mathcal{A}} : \mathcal{E}^*_{\mathcal{A}} \to \mathcal{L}$$

$$f \to \psi_f(k) := \begin{cases} \rho(\mathcal{A}, f_k), & k < |f|, \\ \rho(\mathcal{A}, f), & k \geq |f|, \end{cases}$$

其中，ψ_f 是一个函数，它将每个 $k \in \mathbb{N}$映射到第 k步的配置，该配置取决于f的第 k个前缀。我们将规划 f 与规划 g 之间的距离定义为相应函数 $\Phi_{\mathcal{A}}(f)$ $\Phi_{\mathcal{A}}(g)$之间的距离：

$$d : \mathcal{E}^*_{\mathcal{A}} \times \mathcal{E}^*_{\mathcal{A}} \to \mathbb{N} \qquad d(f, g) := \bar{d}(\Phi_{\mathcal{A}}(f), \Phi_{\mathcal{A}}(g)).$$

¯我们可以用多种方式定义 d，从而得到规划 f 与 g之间距离的不同定义：

1） ∞-距离：

$$\bar{d}_\infty(\Phi_{\mathcal{A}}(f), \Phi_{\mathcal{A}}(g)) := \max_{1 \leq k \leq \min\{|f|,|g|\}} d(\psi_f(k), \psi_g(k));$$

2）1距离：

$$\bar{d}_1(\Phi_{\mathcal{A}}(f), \Phi_{\mathcal{A}}(g)) := \sum_{k=1}^{\min\{|f|,|g|\}} d(\psi_f(k), \psi_g(k));$$

3）最大-最小距离：

$$\bar{d}^*_\infty(\Phi_{\mathcal{A}}(f), \Phi_{\mathcal{A}}(g)) := \max_{k \in \mathbb{N}} \min_{h \in \mathbb{N}} d(\psi_f(k), \psi_g(h));$$

4）和-最小距离：

$$\bar{d}^*_1(\Phi_{\mathcal{A}}(f), \Phi_{\mathcal{A}}(g)) := \sum_{k=1}^{\min\{|f|,|g|\}} \min_{h \in \mathbb{N}} d(\psi_f(k), \psi_g(h)).$$

也就是说，采用 ∞距离时，规划 f 和g 之间的距离对应于相对于时间步 k而言，相应配置之间距离的最大值。采用1距离时，该距离对应于相对于时间步 k而言，在 k 处相应配置之间距离的总和。采用最大-最小距离（分别为和-最小距离）时，该距离对应于相对于 k而言，规划 f 在步骤k 到达的配置与规划 g所经过的所有配置集合之间距离的最大值（分别为总和）。容易看出，对于每一对规划f、 g，采用1距离得到的距离是四种距离中最大的，而采用最大-最小距离得到的距离最小。定义这些距离后，我们可以引入优化多智能体路径规划问题（2）的一种有趣变体，即受给定规划约束的优化多智能体路径规划问题。当我们已有某个多智能体路径规划实例的次优解 f_0，并希望为同一问题找到另一个解，使其与 f_0 相差不大且质量更好（即长度更短）时，就会遇到这一问题。给定有向图 G上的初始和最终有效配置 $\mathcal{A}^s$ 和 $\mathcal{A}^t$，给定满足 $\mathcal{A}^t = \rho(\mathcal{A}^s, f_0)$的 $f_0 \in \mathcal{E}^*_{\mathcal{A}^s}$（即$f_0$ 是该多智能体路径规划实例的一个可行解），以及$r \in \mathbb{N}$ 和规划之间的距离 d，受 f_0 约束的最大完工时间优化多智能体路径规划问题定义如下

$$\begin{aligned} &\min |f| \\ &\text{s.t. } \mathcal{A}^t = \rho(\mathcal{A}^s, f) \\ &\qquad f \in \mathcal{E}^*_{\mathcal{A}^s},\ d(f, f_0) \leq r. \end{aligned} \tag{3}$$

C. 问题（*2*）和（*3*）的域缩减

在问题 (2) 和 (3) 中，变量 f 属于定义良好的计划集合 $\mathcal{E}^*_{\mathcal{A}^s}$。为了缩小这两个问题的可行集基数，我们利用一些不变性。具体而言，我们在规划集合 $\mathcal{E}^*_{\mathcal{A}^s}$ 上定义两个等价关系，使得对于同一等价类中的所有规划，问题 (2) 和 (3) 的目标函数取值相同。此外，当且仅当同一等价类中的所有规划都可行时，一个规划才可行。这样，我们就可以将问题 (2) 和 (3) 转化为等价问题，并以等价类集合为优化域。请注意，等价类集合对应于我们将在动态规划求解算法中使用的状态集。我们将在 $\mathcal{E}^*_{\mathcal{A}}$上考虑以下两个等价关系。

定义 *2.3*：设 $f_0 \in \mathcal{E}^*_{\mathcal{A}}$ 为参考计划。给定f, $g \in \mathcal{E}^*_{\mathcal{A}}$，则

1) $f \sim_1 g$ if and only if
   a) $|f| = |g|$;
   b) $\rho(\mathcal{A}, f) = \rho(\mathcal{A}, g)$.
2) $f \sim_2 g$ if and only if
   a) $f \sim_1 g$;

b) $d(f, f_0) = d(g, f_0)$.

我们用 $\tilde{\mathcal{E}}^i_{\mathcal{A}}$ 表示 $\mathcal{E}^*_{\mathcal{A}}$上所有 $\sim_i$ 的等价类集合。设 $\hat{f} \in \tilde{\mathcal{E}}^i_{\mathcal{A}}$ 和 $f \in \mathcal{E}^*_{\mathcal{A}}$ 为 $\hat{}$ 等价类 f的代表元。我们定义：

- f的长度，$|\hat{f}| := |f|$；
- 一个新的转移函数，

$$\rho^* : \mathcal{C} \times \tilde{\mathcal{E}}^i_{\mathcal{A}} \to \mathcal{C},$$

$$\rho^*(\mathcal{A}, \hat{f}) := \rho(\mathcal{A}, f).$$

- f_0 到 f_0 （ f_0 所属的等价类）的距离， d（f， f_0）：$= d$（f， f_0）（仅当 $i = 2$时）。

$\hat{}$请注意， $|f|$ 定义良好，因为根据定义，等价类 f 中的所有元素都具有相同的长度。类似地，ρ^* 和 d 也定义良好，因为对于等价类 f中的所有元素 f_1， f_2 $\hat{}$， ρ（$\mathcal{A}$， f_1）$= \rho$（$\mathcal{A}$， f_2），并且对于 $\sim_2$，d（f_1， f_0）$= d$（f_2， f_0）。

令 α_1： $\tilde{\mathcal{E}}^1_{\mathcal{A}} \to \mathbb{N} \times \mathcal{C}$ 使其满足

$$\alpha_1(\hat{f}) = (|\hat{f}|, \rho^*(\mathcal{A}, \hat{f})). \tag{4}$$

此函数是定义良好的，因为如果 $f_1 \sim_1 f_2$ ，那么 $|f_1| = |f_2|$ ，且 $\rho(\mathcal{A}, f_1) = \rho(\mathcal{A}, f_2)$。此外， α_1 是 $\hat{}\hat{}\hat{}\hat{}$ 单射，因为，如果 f_1 和 f_2 满足 $\alpha_1(f_1) = \alpha_1(f_2)$， $\hat{}\hat{}\hat{}\hat{}$ 那么 $|f_1| = |f_2|$ 且 $\rho^*(\mathcal{A}, f_1) = \rho^*(\mathcal{A}, f_2)$，因此， $\hat{}\hat{}f_1 = f_2$。

令 α_2： $\tilde{\mathcal{E}}^2_{\mathcal{A}} \to \mathbb{N} \times \mathcal{C} \times \mathbb{N}$ 定义如下：

$$\alpha_2(\hat{f}) = (|\hat{f}|, \rho^*(\mathcal{A}, \hat{f}), d(\hat{f}, f_0)). \tag{5}$$

由于若 $f_1 \sim_2 f_2$， 则$|f_1| = |f_2|$， $\rho(\mathcal{A}, f_1) = \rho(\mathcal{A}, f_2)$且 $d(f_1, f_0) = d(f_2, f_0)$，因此该函数定义良好。$\hat{}\hat{}$ 此外， α_2 是单射，因为若 f_1 和 f_2 满足$\hat{}\hat{}\hat{}\hat{}\hat{}\alpha_1(f_1) = \alpha_2(f_2)$，则 $|f_1| = |f_2|$， $\rho^*(\mathcal{A}, f_1) = \rho^*(\mathcal{A}, f_2)$， $\hat{}\hat{}\hat{}\hat{}d(f_1, f_0) = d(f_2, f_0)$，因此 $f_1 = f_2$。

由于 $f \sim_i g$, $i = 1$, 2，意味着 $|f| = |g|$， $\rho(A^s, f) = \rho(A^s, g)$，且 $d(f, f_0) \le r \leftrightarrow d(g, f_0) \le r$，因此问题（2）和（3）在等价关系 $\sim_i$ 下保持不变。因此，给定 $\hat{f}_0 \in \tilde{\mathcal{E}}^i_{\mathcal{A}^s}$，问题（3）（问题（2）也类似）可在等价类集合上定义如下：

$$\begin{aligned} &\min |\hat{f}| \\ &\text{s.t. } \mathcal{A}^t = \rho^*(\mathcal{A}^s, \hat{f}) \\ &\hat{f} \in \tilde{\mathcal{E}}^i_{\mathcal{A}^s},\ d(\hat{f}, f_0) \le r. \end{aligned} \tag{6}$$

D. 邻域

根据第 II-B 节定义的距离以及第 II-C 节对等价类的定义，我们可以定义顶点、配置和等价类的邻域。此外，我们还能够对这类邻域的基数进行上界估计。为了评估探索这些邻域所需的时间，这种估计是必要的；而探索邻域正是本文所提出方法中的一项核心操作，

首先，我们定义一个以 $v \in V$ 为中心、半径为 $r \in \mathbb{N}$ 的球 $\mathcal{B}_r(v)$：

$$\mathcal{B}_r(v) := \{u \in V : d(u, v) \le r\}.$$

¯我们用 $\mathcal{B}_r(v)$ 表示球的边界，即在定义中用 $=$ 替换 $\le$ 得到的集合。令 $\phi = outdeg(G)$ 为有向图 $G = (V, E)$ 的最大出度。以下命题给出了 $\mathcal{B}_r(v)$ 基数的一个上界。

命题 *II.1*：成立。

$$|\bar{\mathcal{B}}_r(v)| \le \phi^r. \tag{7}$$

证明。令 n_h 为与 v 距离为 h 的节点数。注意， $n_1 \le \phi$，且 $\forall h \ge 2$ $n_h \le n_{h-1}(\phi - 1)$。由归纳法可得， $n_h \le \phi(\phi - 1)^{h-1}$ $\forall h \ge 1$。因此，球的边界上的节点数的上界为

$$|\bar{\mathcal{B}}_r(v)| \le \phi(\phi - 1)^{r-1} \le \phi^r.$$

接下来，我们定义一个以 $\mathcal{A} \in \mathcal{C}$ 为中心、半径为 r 的球：

$$\mathcal{B}_r(\mathcal{A}) := \{\mathcal{A}^* \in \mathcal{C} : d(\mathcal{A}^*, \mathcal{A}) \le r\},$$

¯，记其边界为 $\mathcal{B}_r(\mathcal{A})$。以下命题给出了球的基数上界。

命题 *II.2*：有

$$|\mathcal{B}_r(\mathcal{A})| \le 1 + \frac{(r + k - 1)!}{(r - 1)!(k - 1)!}\phi^r. \tag{8}$$

证明。 首先，我们求出与 $\mathcal{A}$ 的距离为 h 的配置数上界，也就是以 $\mathcal{A}$ 为中心、半径为 h 的球的边界的基数上界：

$$\bar{\mathcal{B}}_h(\mathcal{A}) = \left\{ \mathcal{A}^* \in \mathcal{C} : \sum_{i=1}^{|P|} d(\mathcal{A}(p_i), \mathcal{A}^*(p_i)) = h \right\}.$$

设 $(h_1, \cdots, h_{|P|})$ 是 h 的一个 $|P|$-分解（即 $\sum_{i=1}^{|P|} h_i = h$）。满足 $d(\mathcal{A}(p_i), \mathcal{A}^*(p_i)) = h_i$ 的配置数上界为

$$\prod_{i=1}^{|P|} |\bar{\mathcal{B}}_{h_i}(\mathcal{A}(p_i))| \le \prod_{i=1}^{|P|} \phi^{h_i} = \phi^h.$$

根据标准的组合论论证， h 的 $|P|$-分解数为

$$\frac{(h + (|P| - 1))!}{h!(|P| - 1)!}.$$

因此，半径为h的球的边界的基数可由以下上界限定：

$$|\bar{\mathcal{B}}_h(\mathcal{A})| = \frac{(h + (|P| - 1))!}{h!(|P| - 1)!} \prod_{i=1}^{|P|} |\bar{\mathcal{B}}_{h_i}(\mathcal{A}(p_i))| \le$$

$$\le \frac{(h + (|P| - 1))!}{h!(|P| - 1)!}\phi^h,$$

并且， $\mathcal{B}_r(\mathcal{A})$中的配置总数可按如下方式进行高估：

$$|\mathcal{B}_r(\mathcal{A})| = 1 + \sum_{h=1}^{r} |\bar{\mathcal{B}}_h(\mathcal{A})| \leq$$

$$\leq 1 + \sum_{h=1}^{r} \frac{(h+|P|-1)!}{h!(|P|-1)!}\phi^h \leq 1 + r\,\frac{(r+|P|-1)!}{r!(|P|-1)!}\phi^r.$$

最后，给定半径 $r \in \mathbb{N}$，我们定义 $f_0 \in \mathcal{E}_{\mathcal{A}}^*$ 的以下**邻域**：

$$\mathcal{N}_r(f_0) := \{\hat{g} \in \tilde{\mathcal{E}}_{\mathcal{A}}^i : |\hat{g}| \leq |f_0|, d(\hat{g}, f_0) \leq r\}.$$

这里我们考虑基于最大-最小距离的距离。以下命题给出了 $\mathcal{N}_r(f_0)$ 基数的上界。注意，由于最大-最小距离是所考虑距离中最小的，因此命题给出的上界对于此前讨论的基于其他距离的距离也有效。

命题 *II.3*：有：

$$|\mathcal{N}_r(f_0)| \leq |f_0|^2 \left(1 + \frac{(r+k-1)!}{(r-1)!(k-1)!}\phi^r\right), \qquad (9)$$

其中 $k = |P|$ 和 $\phi = outdeg(G)$。

证明$\mathcal{E}_{\mathcal{A}}^i$设 $f\hat{f} \notin \mathcal{E}_{\mathcal{A}}^*$，且 $\hat{} f \in \mathcal{B}_r(f_0)$。设 $\hat{}\hat{} f$ 为 f 的代表元，并设 $I = \{1, \cdots, |f|\}$ 和 $J = \{1, \cdots, |f_0|\}$。则：

$$\alpha_1(\mathcal{N}_r(f_0)) \subset \left(I \times \bigcup_{j \in J} \mathcal{B}_r(\psi_{f_0}(j))\right).$$

$\hat{}$ 确实，对于 f 的每个代表元 f，都有 $|f| \leq |f_0|$ $\hat{}$；此外， $d(f,\ f_0) \leq r$ 蕴含

$$\max_{1 \leq i \leq |f|} \min_{1 \leq j \leq |f_0|} d(\psi_f(i), \psi_{f_0}(j)) \leq r,$$

特别是，对于 $i = |f|$，存在 j 使得

$$d(\rho(\mathcal{A}, f), \psi_{f_0}(j)) \leq r,$$

这意味着 $\rho(\mathcal{A},\ f) \in \mathcal{B}_r(\psi_{f_0}(j))$。因此，回想起 α 是单射，

$$|\mathcal{N}_r(f_0)| \leq \left|I \times \bigcup_{j \in J} \mathcal{B}_r(\psi_{f_0}(j))\right|.$$

那么，同样根据式 (8)，我们有：

$$|\mathcal{N}_r(f_0)| \leq |f_0| \left(\sum_{j=1}^{|f_0|} |\mathcal{B}_r(\psi_{f_0}(j))|\right) \leq$$

$$\leq |f_0|^2 \left(1 + \frac{(r+k-1)!}{(r-1)!(k-1)!}\phi^r\right).$$

命题 *II.4*： f_0 的半径为 r 的邻域，其基数相对于节点数是多项式级的。特别地， $\exists C = C(r) \in \mathbb{R}$ 使得

$$|\mathcal{N}_r(f_0)| \leq |f_0|^2 \left(1 + C(r+k)^r \phi^r\right),$$

其中 $k = |P|$ 和 $\phi = outdeg(G)$。

证明。注意到

$$\frac{(r+k-1)!}{(r-1)!(k-1)!} \leq \frac{(r+k)^r}{(r-1)!}.$$

定义 $C := {}^{1}\,\frac{}{(r-1)!}$，并回顾命题 II.3 的结果，即得所证。

III. 迭代邻域搜索

本节将介绍所提出的迭代方法，用于检测问题（2）的次优解，或在等价类集合上定义的该问题的等价对应问题的次优解。相对于参考解的邻域而言，返回的解是局部最优的。在每次迭代中，我们求解问题（6）的一个实例。更具体地说，该算法以一个可行解 f_0作为输入，该解的质量可能较差。例如，我们可以通过基于规则的算法获得 f_0 。我们的目标是改进 f_0，得到一个更短的解。为此，我们使用动态规划算法求解问题（6）。也就是说，我们通过算法*Dynprog*在邻域 $\mathcal{N}_r(f_0)$中搜索长度短于 f_0 的规划；我们将在下文介绍该算法。如果无法获得比 f_0 更短的解（即， f_0 是局部最优的），则停止算法。否则，如果获得了改进的解 f^*，我们将参考解重新定义为 $f_0 = f^*$，并再次求解问题（6）。我们重复这一过程，直到无法进一步缩短当前解。

该算法可归类为**邻域搜索**算法（见 [8]）。

要定义邻域 $N_r(f_0)$，我们可以使用 II-B 节中介绍的任意距离函数。在我们的数值实验中，我们使用了和-最小距离。算法 1 给出了该过程的步骤。

算法 1 邻域搜索

输入： $f_0,\ r,\ \mathcal{A}^s,\ \mathcal{A}^t$
输出： f^*
$f^* \leftarrow f_0$
do
  $f_0 \leftarrow f^*$
  $f^* \leftarrow DynProg(f_0, r, \mathcal{A}^s, \mathcal{A}^t)$
当 $|f^*| < |f_0|$
返回 f^*

在算法 1 中， r 是局部搜索半径， $\mathcal{A}^s$ 是初始配置，A^t 是最终配置。

IV. 动态规划算法

为搜索问题（6）的最优解，我们采用一种动态规划（DP）算法。在一般的 DP 问题中，给定一个状态空间 S，其中 $A \subset S$ 是目标状态；一个扩展函数 $g\colon S \to P(S)$，其中P 是 S的幂集；以及一个目标函数 $c : S \to R$。从初始状态 $s_0 \in S$出发，我们使用函数 g 迭代扩展状态，以探索状态空间，并计算使目标函数最小的$s_t = g^n(s_0) \in A$ 。

在我们的情形中，状态表示关系 $\sim_2$的等价类，该关系在第 II-C 节中定义。具体而言，我们使用单射α_2：
$\tilde{\mathcal{E}}_{\mathcal{A}}^2 \to \mathbb{N}\times\mathcal{C}\times\mathbb{N}$，将每个等价类 ˆ ˆ ˆ$f$关联到一个三元组 $(\beta,\ \gamma,\ \sigma) = \alpha_2(f)$，其中 β是 f的长度，γ是将 f的一个代表元计划应用于初始状态 $\mathcal{A}^s$后得到的配置，而 σ 是 f 的一个代表元与参考计划 f_0之间的距离。具体而言，状态空间为

$$\mathcal{S} := \alpha_2(\tilde{\mathcal{E}}_{\mathcal{A}}^2) \subset \mathbb{N}\times\mathcal{C}\times\mathbb{N},$$

其中，α_2 在（5）中定义。由于 α_2 是单射，$\mathcal{S}$ 和 $\tilde{\mathcal{E}}_{\mathcal{A}}^2$ 一一对应。每个状态 $s = (\beta,\gamma,\sigma) \in \mathcal{S}$表示等价类：

$$\alpha_2^{-1}(s) = \{f \in \mathcal{E}_{\mathcal{A}} : \beta = |f|,\ \gamma = \rho(\mathcal{A}^s, f),\\ \sigma = d(f, f_0)\}.$$

初始状态为 $s_0 = \alpha_2(\epsilon) = (0, \mathcal{A}^s, 0)$。我们使用优先队列 Q 存储尚未访问的状态。开始时，$Q = \{s_0\}$。我们基于长度在 $\mathcal{S}$上定义偏序。也就是说，如果 $s_1 = (\beta_1,\gamma_1,\sigma_1)$ $s_2 = (\beta_2, \gamma_2, \sigma_2)$，则 $s_1 < s_2$ 当且仅当 $\beta_1 < \beta_2$。我们按照这一偏序对 Q中的元素排序。

如果满足以下条件，则状态 $s_1 = (\beta_1,\gamma_1,\sigma_1)$ 支配 $s_2 = (\beta_2,\gamma_2,\sigma_2)$：

- $\beta_1 \le \beta_2$，
- $\gamma_1 = \gamma_2$，
- $\sigma_1 \le \sigma_2$。

换言之，如果对应于 s_1 和 s_2的规划 f_1、f_2满足以下性质，则 s_1 支配 s_2 ：规划 f_1 的长度不超过 f_2，f_1 和 f_2 导向相同的最终配置，并且 f_1 与参考解 f_0 的距离不大于 f_2与参考解的距离。如果 s_1 支配s_2，我们可以丢弃 s_2。一般而言，我们会从 Q 中移除所有被支配的状态。我们还定义了以下转移函数，它可以（在某些情况下）将新状态加入优先队列：

$$\tilde{\rho} : \mathcal{S}\times\mathcal{E} \to \mathcal{S}$$

$$\tilde{\rho}((\beta,\gamma,\sigma),e) := (\beta+1, \rho(\gamma,e), \sigma + \min_{k\in\mathbb{N}} d(\rho(\gamma,e), \psi_{f_0}(k))).$$

将此函数应用于状态 $s = (\beta,\ \gamma,\ \sigma)$：

- 将等价类 $\alpha_2^{-1}(s)$ 的长度加1；
- 会更新等价类的最终配置，并将函数 $\rho(\gamma,\ e)$ 应用于$\alpha_2^{-1}(s)$ 的最终配置，其中 e 是选定的边集；
- 更新 σ，添加更新后的最终配置 $\rho(\gamma,\ e)$ 与参考计划 f_0 之间计算得出的最小距离。我们用.....表示

$$\Sigma := \{\tilde{\rho}((\beta,\gamma,\sigma),e) \ :\ e \in \mathcal{E} \text{ and } \rho(\gamma,e) \in \mathcal{B}_r(\gamma)\} \subset \mathcal{S},$$

通过将转移函数 $\tilde{\rho}$ 应用于当前状态（$\beta,\ \gamma,\ \sigma$）以及 $\mathcal{E}$ 中所有可能的动作，可以生成并导向配置 $\mathcal{B}_r(\gamma)$ 的新状态集合。此外，我们用.....表示

$$\Gamma := \alpha_2(\mathcal{N}_r(f_0)) \subset \mathcal{S},$$

邻域搜索过程中可以访问的状态集合。

A. 算法

动态规划算法如算法 2 所述。算法内部维护的优先队列 Q 是一个状态集合，并按其代表元的长度 β 排序。函数 insert($Q,\ x$) 插入一个状态 x，同时维护 Q 的偏序。函数 remove($Q,\ x$) 从 Q 中移除 x。队列的队首，即 β 最小的状态，记为 $Q[0]$。算法从初始状态 s_0 开始探索状态空间。每次迭代时，都会从队列中取出 β 最小的状态。如果取出的状态是目标状态（即 $\gamma = \mathcal{A}^t$），算法便停止，并返回问题（6）的一个最优解ˆ 的代表元（对于给定的等价类 f，函数 ˆrepr(f) 返回该类的一个代表元）。否则，算法会根据先前定义的转移函数，使用函数 expand($s,\ f_0,\ r$) 来寻找新状态。如果一个新状态未被支配，则将其加入队列 Q。此外，还会从 Q 中移除 Q 内所有被新加入状态支配的状态。请注意，对于更复杂的

算法 2 带支配关系的动态规划

输入：$f_0,\ r,\ \mathcal{A}^s,\ \mathcal{A}^t$
输出：f
$s_0 \leftarrow (0, \mathcal{A}^s, 0)$
插入(Q, s_0)
当 $Q \neq \emptyset$ **时**:
  $s = (\beta,\ \gamma,\ \sigma) \leftarrow Q[0]$
  如果 $\gamma = \mathcal{A}_t$ **那么**
    $f \leftarrow$repr($\alpha_2^{-1}(s)$)
    $Q \leftarrow \emptyset$
  else
    $\Sigma \leftarrow$ 扩展（$s,\ f_0,\ r$）
    对于 $s_k \in \Sigma$，**执行**
      如果 s_k 未被支配于 Q **则**
        插入（$Q,\ s_k$）
        对于 $s_i \in Q$，**执行**
          如果 s_k 支配 s_i **则**
            移除($Q,\ s_i$)
返回 f

解的效果要好得多。

定理 *IV.1*：算法 1 和 2 关于图中节点数的时间复杂度为多项式时间。证明。在算法 2 中，时间复杂度为 $O(|Q|^2\cdot|\Sigma|)$。集合 Q 和 Σ 随每次迭代而变化，但始终是 Γ的子集。因此，每次迭代时，其基数始终满足以下上界：

$$|Q| \le |\Gamma|, |\Sigma| \le |\Gamma|.$$

注意到 α_2 是单射，并利用命题 II.4 的结论，

$$|\Gamma| = |\mathcal{N}_r(f_0)| \le |f_0|^2\left(1 + C(1+k)^r\phi^r\right),$$

其中 $k = |P|$ 和 $\phi = outdeg\,(G)$。因此，算法 2 的时间复杂度为

$$O(|f_0|^6\,(1 + C(r + k)^r \phi^r)^3).$$

算法 1最多调用 $|f_0|$ 次算法 2，因此其时间复杂度为 $O(|f_0|^7(1 + C(r + k)^r \phi^r)^3)$。

V. 实验结果

我们在不同的图上进行了两组不同的实验，并采用两种不同的方式生成初始解。该算法使用*C++* 编程语言编写，并在配备 16 GB 内存的第 *11* 代英特尔*(R)* 酷睿*(TM) i7-1165G7 @ 2.80GHz*处理器上运行。

A. 具有顺序生成初始解的随机图

在第一组实验中，我们生成了随机有向图，节点数 $|V|$ 从 20 到 100，步长为 10，边数 $|E|$ 等于 $4|V|$。图的生成方式为创建 $|E|$ 随机的节点有序对，并利用这些有序对构建有向图。我们只选取强连通图。智能体数量 $|P|$从 2 到 10 不等， $\mathcal{A}^s$ 和 $\mathcal{A}^t$ 则随机生成。在这些实验中，为生成初始解，我们一次将一个智能体移动到其目标节点，所遵循的路径是：在移除当前被其他所有智能体占据的节点后所得到的图中，从其源节点到目标节点的最短路径；其中，被占据的节点可以是其他智能体的目标节点，也可以是其源节点，具体取决于这些智能体是否已经移动。需要注意的是，该过程并不完备，即当可行解存在时，它不保证能够找到可行解，也不保证能够确定不存在可行解。我们生成了 100 个随机图；对于节点数和智能体数量的每一种组合，上述过程都能够返回一个可行解。经过一些调整后，我们将邻域的半径 r 设为 5；事实证明，这是在邻域中找到的解的质量与探索邻域所需的时间之间的一个良好折中（注意，该时间会随 r呈指数增长）。给定初始解 f_0 和所提出方法返回的最终解 f^*，最终解相对于初始解的降幅百分比等于 $100\frac{f_0 - f^*}{f_0}$ %。在图 1 和图 2 中，我们分别报告了 $|P|$ 和 $|V|$每种组合下平均降幅百分比的中位数以及运行时间（以秒为单位）。值得指出的是，随着智能体数量增加，降幅百分比往往会降低。一种可能的解释是，对于智能体数量较多的情况，当用于生成初始解的顺序过程能够返回可行解时，该解本身已经较好，因而无法得到大幅改进。在第二组实验中没有观察到这一现象，因为其中采用了不同的过程来生成初始可行解。

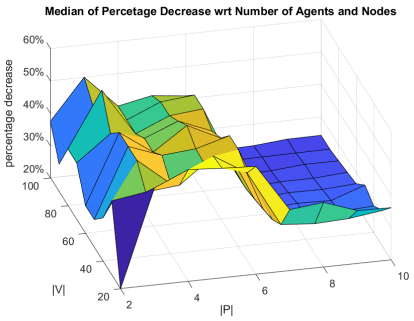


图 1：每个 $|P|$ 和 $|V|$ 的平均百分比降幅。

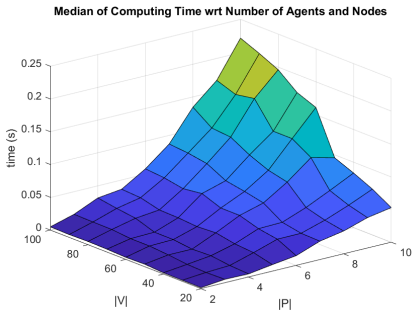


图 2：每个 $|P|$ 和 $|V|$ 的平均运行时间。

请注意，智能体数量越多，平均百分比降幅越低。这可以用寻找初始解时所采用算法的特性来解释：在节点数量不变的情况下增加智能体数量，可行实例会大幅减少，而且找到的解更难改进。

B. 具有多个双连通分量的强连通图，以及基于规则生成的初始解

在第二组实验中，我们生成了具有强连通性和多个双连通分量的图。生成此类图的过程见 [2]。智能体数量 $|P|$ 从2到6不等，节点数从20到50不等，增量为10。初始配置 $\mathcal{A}^s$ 和最终配置$\mathcal{A}^t$ 均为随机生成。在这种情况下，初始解（如果存在）通过 [2]中描述的算法生成。需要注意的是，该算法是完备的，即当可行解存在时，它总能返回一个可行解。然而，与第一组实验中用于生成初始解的算法所返回的解相比，这种初始解的质量通常较低（即，初始规划通常更长）。这也可能解释了为什么这些实验中的下降百分比（见图3）相较于第一组实验显得更大，并且还往往随着智能体数量的增加而增加（这与随机图的情况不同）。同样，在进行了一些调参后，我们将邻域 r 的半径设为3。与第一组实验一样，图3和图4分别报告了每种 [2]和 $|P|$ 组合下的平均下降百分比中位数以及运行时间（单位：秒）的中位数，

分别为每种 $|P|$ 和 $|V|$组合下的平均下降百分比和运行时间（单位：秒）的中位数。需要注意的是，第二组实验所采用的图似乎比随机图更具挑战性。实际上，计算时间更长，并且随着智能体数量的增加而迅速增长。

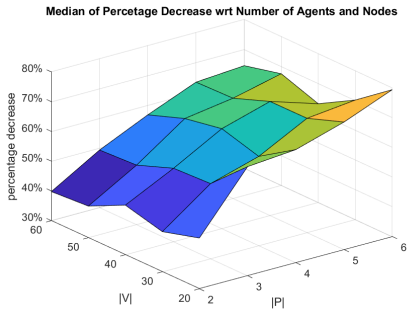


图 3：每个 $|P|$ 和 $|V|$的平均下降百分比。

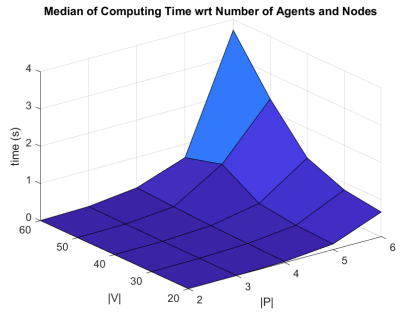


图 4：每个 $|P|$ 和 $|V|$的平均运行时间。

VI. 结论与未来工作

我们提出了一种用于多智能体路径寻找的迭代局部搜索过程，旨在缩短已知的可行解。我们得到的解仍是次优解，但总体而言，其质量远胜于初始解。所提出的算法相对于智能体数量具有多项式时间复杂度（见定理 IV.1），且其计算耗时适用于工业应用。我们可以从多个方面扩展这些结果，这将是未来研究的重点：

- 我们可以定义不同于第 II-B 节所述的局部性约束。例如，可以限制相对于参考解修改路径的智能体数量。或者，可以对解偏离参考解的时间区间数量设定上界。

- 我们可以在真实的工业场景中测试所提出的方法。

- 我们可以改进定理 IV.1 中给出的复杂度界，因为目前的界相当粗略。

参考文献

[1] Ebtehal Turki Saho Alotaibi 和 Hisham Al-Rawi。《Push and spin：一种完备的多机器人路径规划算法》。载于：*2016*年第*14*届国际控制、自动化、机器人与视觉会议（*ICARCV*）。泰国普吉岛：IEEE，2016年11月，第1–8页。[2] S. Ardizzoni 等。《强连通有向图上的多智能体路径规划》。载于：*2022*年 *IEEE* 第*61*届决策与控制会议（*CDC*）。墨西哥坎昆：IEEE，2022年12月，第7194–7199页。[3] Adi Botea、Davide Bonusi 和 Pavel Surynek。《在强双连通有向图上求解多智能体路径规划》。载于：人工智能研究杂志 62（2018年6月），第273–314页。[4] Gilad Goraly 和 Refael Hassin。《图上的多色鹅卵石移动》。英文。载于：*Algorithmica* 58.3（2010年11月），第610–636页。[5] Mokhtar Khorshid、Robert Holte 和 Nathan Sturtevant。《一种用于非最优多智能体路径规划的多项式时间算法》。载于：组合搜索国际研讨会论文集 2.1（2021年8月），第76–83页。[6] D. Kornhauser、G. Miller 和 P. Spirakis。《图上的卵石移动协调、置换群的直径及其应用》。载于：*1984*年第*25*届计算机科学基础年会。美国佛罗里达州辛格岛：IEEE，1984年，第241–250页。[7] Athanasios Krontiris、Ryan Luna 和 Kostas Bekris。《从可行性测试到多智能体路径规划器》。载于：组合搜索国际研讨会论文集 4.1（2021年8月），第114–122页。[8] David Pisinger 和 Stefan Ropke。《大邻域搜索》。载于：元启发式手册 （2019年），第99–127页。[9] Guni Sharon 等。《用于最优多智能体路径规划的基于冲突的搜索》。英文。载于：人工智能 219（2015年2月），第40–66页。[10] David Silver。《协作式路径规划》。载于：*AAAI*人工智能与互动数字娱乐会议论文集 1.1（2021年9月），第117–122页。[11] B. de Wilde、A. W. Ter Mors 和 C. Witteveen。《Push and Rotate：一种完备的多智能体路径规划算法》。载于：人工智能研究杂志51（2014年10月），第443–492页。[12] Jingjin Yu 和 Steven LaValle。《图上最优多机器人路径规划的结构与难解性》。载于：*AAAI*人工智能会议论文集。第27卷，第1期。2013年，第1443–1449页。