2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

Multi-Robot Path Deconfliction through Prioritization by Path Prospects

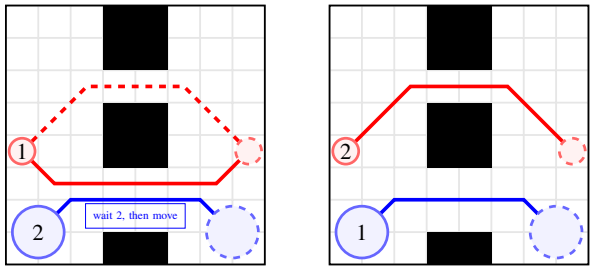
Wenying Wu*, Subhrajit Bhattacharya†, Amanda Prorok*

***Abstract*— This work deals with the problem of planning conflict-free paths for mobile robots in cluttered environments. Since centralized, coupled planning algorithms are computationally intractable for large numbers of robots, we consider decoupled planning, in which robots plan their paths sequentially in order of priority. Choosing how to prioritize the robots is a key consideration. State-of-the-art prioritization heuristics, however, do not model the coupling between a robot's mobility and its environment. This is particularly relevant when prioritizing between robots with different degrees of mobility. In this paper, we propose a prioritization rule that can be computed online by each robot independently, and that provides consistent, conflict-free path plans. Our innovation is to formalize a robot's *path prospects* to reach its goal from its current location. To this end, we consider the number of homology classes of trajectories, which capture *distinct* prospects of paths for each robot. This measure is used as a prioritization rule, whenever any robots enter negotiation to deconflict path plans. We perform simulations with heterogeneous robot teams and compare our method to five benchmarks. Our method achieves the highest success rate, and strikes a good balance between makespan and flowtime objectives.**

I. Introduction

Technological advances are enabling the large-scale deployment of robots to solve various types of problems in logistics and transport [9, 12, 14]. The commonality of many of these applications is that they require methods that assign and guide individual robots to goal locations on collision-free paths. The challenge of providing fast, optimal and complete solutions to this problem is very current, as we continue to complexify the problem domain by considering increasingly large and heterogeneous robot teams in navigation-constrained, cluttered environments. In light of these developments, our work focuses on the *coupling* between a robot's mobility traits and the built environment. In particular, we posit that a robot's ability to reach its goal can be measured, and that by integrating this measure in planning routines, better joint path plans can be found.

Approaches to multi-robot path planning can generally be described as either centralized (assuming the existence of a central component that knows the state of the whole robot system) or decentralized (where no single component has the full picture, but cooperation must still be achieved). Centralized methods can be further categorized according to whether they are coupled or decoupled. Coupled approaches operate in the joint configuration space of all the robots, allowing for completeness (e.g., see [13, 22]). However, solving for optimality is NP-hard [23], and although significant progress has been made towards alleviating the computational load, e.g., [11, 18, 22], these approaches still scale poorly in environments with a high number of path conflicts. On the other hand, decoupled approaches plan for each robot separately, and solve conflicts between paths as they arise, to ensure that collisions with other robots are avoided. Approaches to decoupled planning include sequential programming [6], vehicle prioritization [20] and velocity tuning [15]. These methods offer improved scalability, but often at the cost of completeness and optimality [2].



Makespan: 8.8
Flowtime: 16.6

Makespan: 8.6
Flowtime: 15.4

Fig. 1: An example problem where considering path diversity is important for the prioritization. The red robot has two possible paths, whereas the larger blue robot only has one. On the left, the red robot has first priority. It takes the shorter of its two paths, however this forces the blue robot to wait in place until it can follow. On the right, the blue robot, with lower path diversity, has first priority. The red robot is able to take its alternative path to avoid it, giving a faster overall solution.

Prioritized planning, first proposed in [10] as a centralized strategy, is a very efficient method because it allows robots to plan sequentially in space-time in order of priority, eschewing the combinatorial complexities of coupled approaches. In this approach, each robot plans a minimum-cost path to its goal that avoids the computed trajectories of any higher-priority robots. Clearly, the chosen priority order will affect the solution found. It is generally desirable to optimize metrics such as makespan (the time at which the last robot in the team arrives), flowtime (the sum of all robots' travel times), or success rate (completion); targeting the optimization of either one of these objectives (but commonly not all simultaneously), researchers have proposed heuristics for choosing a planning order [2, 16, 20, 21].

The original prioritized planning idea in [10] used a *fixed* total priority ordering, and has been adapted in various papers to work in a decentralized manner (e.g., [5, 21]). However, choosing how to prioritize the team of robots still remains a key consideration. Moreover, as the operational conditions of the robots vary throughout time, and environments are in general not static, it is crucial to consider *online* (dynamic) priority schemes. Although several dynamic priority schemes have been considered thus far (e.g., [1, 7, 8, 17, 20]), none

*Wenying Wu and Amanda Prorok are with the University of Cambridge, UK: {ww329|asp45}@cam.ac.uk. †Subhrajit Bhattacharya is with Lehigh University, USA: sub216@lehigh.edu

of these schemes account for the coupling between robot mobility and the environment, and hence, may fail to find better solutions. Such a scenario is exemplified in Fig. 1, which shows how considering a robot's path diversity leads to a reduction of both flowtime and makespan.

In this work, we focus on how the coupling between the built environment and a robot's mobility traits determines its path options to reach its goal. In specific, we propose a decentralized planning method that makes use of a novel prioritization rule based on an estimate of the robot's *path prospects*. The key idea that underpins this method is that individual robots have distinct path prospects within the same environment, due to unique conditions arising from kinematic, dynamic, or environmental constraints. The purpose of this work is to provide a formal introduction to the concept of path prospects, and a demonstration of its utility in path planning for (potentially heterogeneous) multi-robot teams.

Contributions. In this work, we propose a novel prioritization heuristic that is based on the number of *path prospects* that represent a robot's *distinct* path options to reach its goal. This prioritization rule that has two key components: **(1)** a method that computes the *distinct number of path options* a robot has to reach its goal, based on theory from algebraic topology, and **(2)**, a method that defines the *area of relevance*, within which these path options are computed. We show how our prioritization rule is embedded in a decentralized, online planning algorithm to de-conflict robot trajectories between robots with potentially different mobility capabilities. We prove that this dynamic planning algorithm provides a partial ordering over the robot set, and hence, is cycle-free. Our results demonstrate that the planned solutions provide competitive makespan and flowtime performance, at very high success rates.

II. Problem Description

We consider a $D-$dimensional workspace $\mathcal{W} \subseteq \mathbb{R}^D$ and a set of B static obstacles $\mathcal{O} = \{o_1, \ldots, o_B\}$ with $o_i \subset \mathcal{W}$. A team of N robots $\mathcal{R} = \{r_1, \ldots, r_N\}$ navigate in this workspace. The robot team may be heterogeneous in size; the effective space occupied by robot r_n is ρ_n.

Graph representation. Each robot travels along the edges of a directed graph $G_n = \langle \mathcal{V}_n, \mathcal{E}_n \rangle$, which allows only feasible motion. This graph can be arbitrarily dense, and can represent motion with high granularity (as such it can accommodate a wide range of motion); furthermore, it does not need to be fully represented in memory, but instead, can be generated online (e.g., by sampling motion primitives). In particular, a robot r_n that travels along edges in G_n cannot collide with any obstacles in $\mathcal{O}$. The set $\mathcal{V}_n$ is defined by vertices $v_i = \langle \mathbf{x}_i, t_i \rangle$ with $\mathbf{x}_i \in \mathcal{W}$ and $t_i \in \mathbb{R}^+$. The set $\mathcal{E}_n$ is defined by directed edges $e_{ij} : \mathbb{R}^+ \mapsto \mathbb{R}^D$, between vertex v_i and v_j, such that $e_{ij}(t_i) = \mathbf{x}_i$, $e_{ij}(t_j) = \mathbf{x}_j$, and $t_i \leq t_j$. In other words, the graph G_n exists in a $(D+1)$-dimensional space, where the last dimension represents time.

Labeled assignment. Robot r_n is assigned a start location $\mathbf{s}_n \in \mathcal{W}$ (corresponding to vertex v_i with $\mathbf{x}_i = \mathbf{s}_n$ and $t_i = 0$). Similarly, robot r_n is assigned a goal location $\mathbf{g}_n \in \mathcal{W}$ (corresponding to a set of vertices v_i with $\mathbf{x}_i = \mathbf{g}_n$ and $t_i \in \mathbb{R}^+$). A labeled assignment $\mathcal{A}$ is a set of tuples $\{\langle \mathbf{s}_1, \mathbf{g}_1 \rangle, \ldots, \langle \mathbf{s}_N, \mathbf{g}_N \rangle\}$, for all robots in $\mathcal{R}$.

Conflict-free trajectories. A robot r_n has a trajectory $\pi_n : \mathbb{R}^+ \mapsto \mathcal{W}$ that represents a sequence of edges traversed in G_n such that two consecutive edges share a common vertex. A trajectory π_n is said to be *satisfying* if $\pi_n(0) = \mathbf{s}_n$ and there exists a time t_n^f such that $\pi_n(t_n^f) = \mathbf{g}_n$. A robot r_n navigating along this path defines a volume $V(\pi_n, \rho_n)$ that depends on its size. To coordinate the navigation in $\mathcal{W}$, two robots r_n and r_m can share their path plans with each other if they are within communication range, i.e., if their positions are separated by a quantity less than c. We denote by $\mathcal{N}_n$ the set of all robots within communication range of r_n (including r_n). We make use of a function $\textsc{Trim}(G_n, \rho_n, V(\pi_m, \rho_m))$ that removes all unfeasible paths in G_n that would collide with the volume defined by robot r_m. Any path in the graph returned by TRIM is ensured to be *conflict-free* with the path π_m of robot r_m.

In order to facilitate the definition of a given robot's configuration space, we define the notion of an *effective obstacle*, which is a set of original obstacles in $\mathcal{O}$, such that no trajectories in a given graph passes between them (see Figure 6). Specifically, a robot r_n has a set of effective obstacles $\tilde{\mathcal{O}}_n = \{\tilde{o}_1, \ldots, \tilde{o}_{\tilde{B}}\}$, $\tilde{B} \leq B$, with $\tilde{o}_i \subseteq \mathcal{O}$ and $\cup_i \tilde{o}_i = \mathcal{O}$ and $\cap_i \tilde{o}_i = \emptyset$.

Figure 2 shows a labeled assignment for two robots that must plan minimum-cost trajectories from their start positions to their goal positions. Figure 3 demonstrates how robot r_2 circumnavigates the path plan of robot r_1, after execution of $\textsc{Trim}(G_2, \rho_2, V(\pi_1, \rho_1))$.

Assumptions. We assume that a robot is able to check for collisions between its own planned path and another robot's. To facilitate this, we assume all their clocks are synchronized. Messaging delay and clock misalignment can be accommodated (e.g., through time-stamps), however, the deviations must be negligible with respect to robot dynamics (i.e., the speed at which the motion graph is traversed). We assume that robot detections are always mutual (when they come into communication range).

Objective. Our goal is to find a method that has a high success rate and that strikes a good balance between minimizing the mean flowtime ($\sum_n t_n^f / N$) and minimizing the makespan ($\max_n t_n^f$), such that each robot r_n follows a satisfying trajectory π_n which is conflict-free with all other robots' paths. We note that, in general, makespan and flowtime objectives demonstrate a pairwise Pareto optimal structure, and cannot be simultaneously optimized [23].

III. Decentralized Planning

Our decentralized path planning algorithm can be broken down into two components: *coordination*, where we consider how robots communicate and negotiate a priority ordering; and *local planning*, where we consider how an individual robot plans a trajectory to its goal given the plans of other robots within communication range. We make use of the following definitions.

Definition 1 (Priority ordering)**.** *A priority ordering $\prec$ is such that a robot $r_n \in \mathcal{R}$ with priority ξ_n is of higher priority than robot r_m with priority ξ_m iff $\xi_n \prec \xi_m$.*

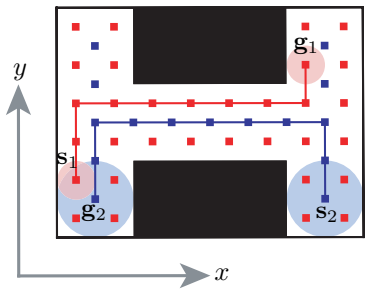


Fig. 2: Planar workspace with two robots, r_1 and r_2, and their respective start and goal positions. Robot r_2 has an effective size ρ_2 that is twice that of robot r_1. The minimum-cost paths would result in a collision.

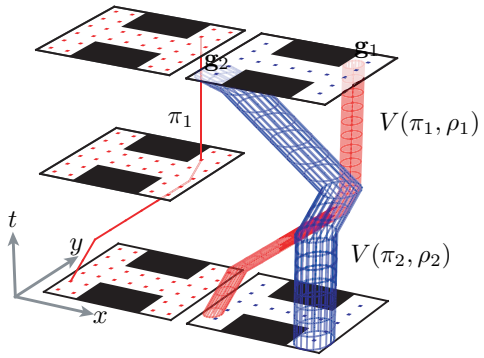


Fig. 3: On the left, we plot the space-time graph G_1 with a minimum-cost trajectory π_1 for robot r_1. On the right, we see how trajectory π_2 sweeps a volume $V(\pi_2, \rho_2)$ that does not intersect with $V(\pi_1, \rho_1)$.

Definition 2 (Ordered robot set)**.** *Given a priority ordering $\prec$ on a set of robots $\mathcal{R}$, the pair $(\mathcal{R}, \prec)$ is a strict partially ordered robot set.*

Definition 3 (Ordered robot neighborhood)**.** *Given a priority ordering $\prec$, for a given robot r_n, $H_n = \{r_m | \xi_m \prec \xi_n$ and $r_m \in \mathcal{N}_n\}$ is the set of robots with higher priority, and $L_n = \{r_m | \xi_m \succ \xi_n$ and $r_m \in \mathcal{N}_n\}$ is the set of robots with lower priority. The neighborhood of robot r_n defined as*

Algorithm 1: Dynamic Prioritized Path Planning

$H_n \leftarrow \emptyset$ // list of higher priority robots
$L_n \leftarrow \emptyset$ // list of lower priority robots
$\pi_n \leftarrow$ ComputeNewPlan(H_n)
while True **do**
  if *new robot r_m (with priority ξ_m) in range c* **or** *received new priority ξ_m from r_m* **then**
    $\xi_n \leftarrow$ ComputePriority(r_n)
    $H_{n,old} \leftarrow H_n$ and $L_{n,old} \leftarrow L_n$
    $H_n \leftarrow \{r_i | \xi_i \prec \xi_n \wedge r_i \in H_{n,old} \cup L_{n,old} \cup \{r_m\}\}$
    $L_n \leftarrow \{r_i | \xi_i \succ \xi_n \wedge r_i \in H_{n,old} \cup L_{n,old} \cup \{r_m\}\}$
    if $H_{n,old} \neq H_n$ **then**
      $\pi_n \leftarrow$ ComputeNewPlan(H_n)
  else if *r_m left range c* **then**
    $L_n \leftarrow L_n \setminus r_m$
    if $r_m \in H_n$ **then**
      $H_n \leftarrow H_n \setminus r_m$
      $\pi_n \leftarrow$ ComputeNewPlan(H_n)
  else if *receive new plan π_m from r_m* **then**
    if $r_m \in H_m$ **then**
      $\pi_n \leftarrow$ ComputeNewPlan(H_n)

Algorithm 2: Re-plan trajectory

Function: ComputeNewPlan(H_n)
$G \leftarrow G_n$
for $r_m \in H_n$ **do**
  ReceivePlanFrom(r_m)
  $G \leftarrow$ Trim$(G, \rho_n, V(\pi_m, \rho_m))$
$\pi_n \leftarrow$ PlanPathFromCurrentPosition$(G, \mathbf{g}_n)$
$\xi_n \leftarrow$ ComputePriority(r_n)
BroadcastPlanIfChanged(π_n)
BroadcastPriorityIfChanged(ξ_n)
return π_n

$\mathcal{N}_n$ is strongly connected (by symmetry of communication). By definition, the robot neighborhood $\mathcal{N}_n$ is an ordered robot set $(\mathcal{N}_n, \prec)$.

A. Coordination Strategy

Our coordination strategy is detailed in Algorithm 1, and is based on two main elements, described as follows:

Computation of an ordered robot neighborhood: Each robot is able to detect other robots when they come into the communication range c, and when they leave it. A robot r_n has a priority score ξ_n, which it can compute locally by a function ComputePriority (see Section IV). Each robot r_n maintains two lists of robots currently in its range: H_n contains the robots with higher priority whilst L_n contains the robots with lower priority. In a dynamic priority scheme, r_n recomputes ξ_n whenever a new robot comes into range. It then broadcasts this updated priority value to ensure all robots within range (i.e., in its neighborhood) have a consistent plan.

Re-planning: Re-planning is triggered for r_n in three cases: *(i)* when a new robot comes into range that has a higher priority, *(ii)* when an updated plan is received from a higher priority robot, or *(iii)*, when a robot r_m broadcasts a new priority ξ_m. When r_n re-plans, it calls a function ComputeNewPlan that takes into account the planned trajectories of robots with higher priority (in H_n). The robot then communicates its new plan to robots in L_n.

Proposition 1. *Algorithm 1 is deadlock-free.*

Proof. Since each robot r_n in $\mathcal{R}$ executes Alg. 1, the result is a collection of ordered robot neighborhoods $\mathcal{N}_n$, $\forall n$. If two robot neighborhoods $\mathcal{N}_n$ and $\mathcal{N}_m$ share a common robot r_j, then, by transitivity, there must be a partial ordering in $\mathcal{N}_n \cup \mathcal{N}_m$, since Algorithm 1 ensures that robot r_j can only have one priority score ξ_j that is broadcast. Hence, Algorithm 1 constructs an ordered robot set $(\mathcal{R}, \prec)$. Since partial orderings are acyclic, no planning deadlocks can arise. □

B. Local Planning

Each robot handles the computation of its own minimum-cost trajectory from its current location to its goal location $\mathbf{g}_n$ (see function ComputeNewPlan). The resulting trajectory avoids the static obstacles in the environment as well as the

planned paths of any higher-priority robots. In our implementation, all robots use the HCA* algorithm proposed in [19] which applies A* search to a space-time map, and uses a reservation table to record the trajectories of other robots to be avoided. This effectively implements the function TRIM. The complexity of TRIM is $O(|\pi_m| log |E_n|)$ assuming the usage of a fast spatial lookup structure, such as a quadtree. When the spatial lookup can be done in constant time, the complexity becomes $O(|\pi_m|)$. The complexity of PLANPATHFROMCURRENTPOSITION is $O(|E_n| + |V_n| log |V_n|)$, or, when $V_n \propto E_n$, $O(|V_n| log |V_n|)$). Hence, the complexity of Alg. 2 is $O(|\mathcal{N}_n|\,|\pi_m|\, log |E_n| + |E_n| + |V_n| log |V_n|)$.

This local planning approach is general, in that any path planning algorithm that is able to avoid dynamic obstacles with known trajectories can be used; indeed, it is even possible for different robots to use different algorithms, so long as an implementation of the function TRIM, which reconciles heterogeneous space-time graphs, can be embedded into the planning function.

IV. PRIORITIZATION BASED ON PATH PROSPECTS

During navigation, when robots come within communication range, they enter negotiations to deconflict their path plans. To facilitate this negotiation, we implement a rule that prioritizes robots with *fewer path options*. Our prioritization rule has two key components: **(1)** a method that estimates the *number of options* a robot has to reach its goal, and **(2)**, a method that defines the *area* within which these path options are computed. The following paragraphs detail our approach.

A. Homology Classes of Trajectories

To develop a method for **(1)** above, we build on theory from algebraic topology. For a particular robot, we consider the trajectories in different *homology classes* as the path prospects for the robot. Homology classes (of trajectories) in an environment represent topologically distinct classes of trajectories (Figure 5). For example, each homology class can be thought to be capturing the width of a passage or separation between two obstacles, and depending on the size/kinematic capabilities of a robot, that passage (and hence the corresponding homology class) may or may not be available to the robot. This gives an effective means of designing a prioritization heuristic without having to consider the set of all possible paths in the discrete domain that the robots can physically take to reach their respective goals.

Two trajectories connecting the same start and goal points on a planar domain are said to be in the same homology class if the closed loop formed by the two trajectories are *null homologous*, i.e., it forms the oriented boundary of a two-dimensional obstacle-free region [4]. The homology class of a loop, in turn, can be quantified by winding numbers around the connected components of effective obstacles, with the null homologous class having zero winding number around every obstacle. Thus, in a planar domain with z connected components of effective obstacles, the *homology invariant* of a loop is given by a vector of integers, $[h_1, h_2, \cdots, h_z] \in \mathbb{Z}^z$, where h_i is the winding number around the i^{th} obstacle.

However, there are infinitely many homology classes since a trajectory can loop/wind around the same obstacle multiple times, and for every different number of windings the class assigned to the loop is different. In order to prevent the separate counting of the multi-looping homology classes, one can compute the homology invariants in the "mod 2" coefficient [3], $\mathbb{Z}_2 = \mathbb{Z}/2\mathbb{Z} = \{0, 1\}$. Simply put, the homology invariant in the $\mathbb{Z}_2$ coefficients become $[h_1, h_2, \cdots, h_z] \mod 2 \in \mathbb{Z}_2^z$. Doing so identifies all the even winding numbers to 0 and all the odd winding numbers to 1, thus preventing the creation of separate homology classes for loops that wind around obstacles multiple times (Figure 4). $\mathbb{Z}_2^z$ is a finite set, and in fact has 2^z elements. Thus, the number of $\mathbb{Z}_2$ coefficient homology classes in a planar domain with z connected components of effective obstacles is 2^z, which we use in the construction of heuristics in the path prospect algorithm.

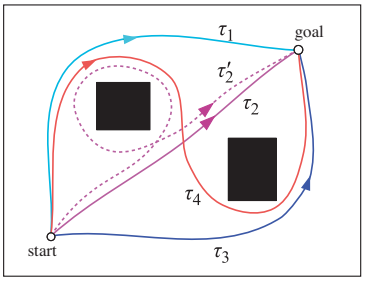


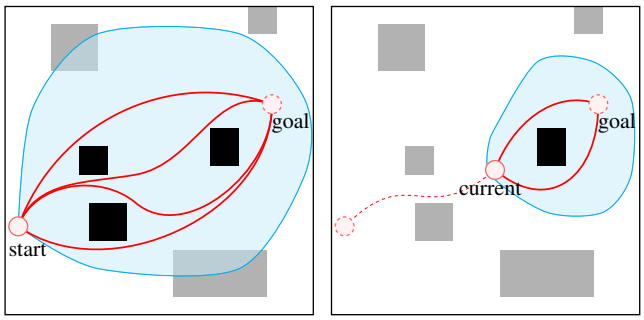
Fig. 4: Homology classes of trajectories. τ_2 and τ_2' are in different classes in regular homology, but map to the same class in $\mathbb{Z}_2$-coefficient homology.

B. Path Prospect Algorithm

We use the number of $\mathbb{Z}_2$ coefficient homology classes in an area with z effective obstacles to return an estimate of a robot r_n's *path prospects* $P_n^{(t)}$ at time t in that area. Next, we develop a method for computing a relevant area (and its associated vertices), to define the component **(2)**, above.

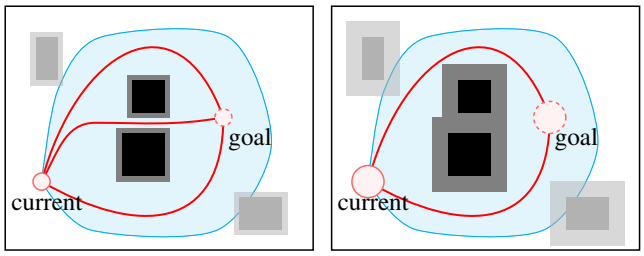
A robot r_n's *path prospects* $P_n^{(t)}$ are an indicator of the number of distinct paths to goal $\mathbf{g}_i$ from its current location at time t. This can be estimated by counting the effective obstacles $\tilde{\mathcal{O}}_n$ which r_n will likely come across as it moves *towards* its goal $\mathbf{g}_n$ from its current position. Specifically, we do not wish to count any effective obstacles that lie *behind* the robot, given $\mathbf{g}_n$ and its current location. To achieve this, we define a set of *forwards vertices* $F_n^{(t)} \subseteq \mathcal{V}_n$ and count only the effective obstacles whose areas intersect the area in $\mathcal{W}$ containing all vertices in $F_n^{(t)}$ and the edges that link them.

To define $F_n^{(t)}$, we use the notion of true distance as proposed in [19]. The true distance of a vertex $v \in \mathcal{V}_n$ is the length of the shortest satisfying path in G_n from v to $\mathbf{g}_n$. We define $F_n^{(t)} \subseteq \mathcal{V}_n$ to be the set of vertices that are reachable from r_n's location at time t by only transitioning from a vertex v_i to a vertex v_j, if v_j can still lead to paths that are shorter than the estimated longest true distance of the robot team. Note that the longest true distance can be estimated locally by broadcasting TRUEDISTANCE$(\mathbf{s}_n, \mathbf{g}_n)$ (this value is time-invariant) along with priority ξ_n in Algorithm 1. Upon receiving such a message, each robot can update its estimate by taking the maximum between the current estimate and the value of the message. The set of forwards vertices can



(a) At $t = 0$, $P_n^{(0)} = 2^3$. **(b)** At $t = 17$, $P_n^{(17)} = 2^1$.

Fig. 5: Illustration of path prospects for a robot navigating to its goal, computed for two different moments in time. In (a), only 4 representative paths out of 8 are shown, for clarity.



(a) $P_n^{(t)} = 2^2$. **(b)** $P_m^{(t)} = 2^1$.

Fig. 6: Example where two robots with different sizes have different path prospects. In (b), the two central obstacles merge into a single effective obstacle. The lighter borders around each obstacle depict their inflation by the robots radius ρ_n, which is one method for computing effective obstacles.

Algorithm 3: Path Prospects

Input : current position of r_n: v_n, goal location $\mathbf{g}_n$, untrimmed graph G_n, effective obstacles $\tilde{\mathcal{O}}_n$, estimated longest path length T
Output: path prospects $P_n^{(t)}$

$F_n^{(t)} \leftarrow$ GETFORWARDSVERTICES$(v_n, \mathbf{g}_n, G_n, T)$
$A \leftarrow$ COMPUTEAREA$(F_n^{(t)}, \mathcal{E}_n)$
$\kappa \leftarrow 0$
for $o \in \tilde{\mathcal{O}}_n$ **do**
  if $o \cap A = o$ **then**
    $\kappa \leftarrow \kappa + 1$ // count this obstacle
return 2^κ

Algorithm 4: Compute Set of Forwards Vertices

Function: GETFORWARDSVERTICES$(v, \mathbf{g}, G, T)$

visited $\leftarrow \emptyset$
priority_queue $\leftarrow \{v\}$ // prioritizes by smallest t
while priority_queue $\neq \emptyset$ **do**
  $q \leftarrow$ POPSMALLEST(priority_queue) with $q = \langle \mathbf{x}_q, t_q \rangle$
  if $\mathbf{x}_q \notin$ visited **then**
    neighbours $\leftarrow$ FINDNEIGHBOURS(G, q)
    for $n \in$ neighbours with $n = \langle \mathbf{x}_n, t_n \rangle$ **do**
      if $t_n +$ TRUEDISTANCE$(n, \mathbf{g}) \leq T$ **then**
        APPEND(priority_queue,n)
    visited $\leftarrow$ visited $\cup \{\mathbf{x}_q\}$
return visited

be computed using a variant of Dijkstra's algorithm (see Algorithm 4), with a complexity of $(|E_n| + |V_n| log |V_n|)$.

Figure 5 illustrates the path prospects for a robot navigating towards its goal, at two consecutive moments in time.

C. Prioritization Heuristic

We use the path prospect algorithm (Algorithm 3) to prioritize robots with conflicting paths. For robots r_n and r_m, we define the ordering $\prec$ such that

$$P_n^{(t)} < P_m^{(t)} \Leftrightarrow \xi_n \prec \xi_m. \quad (1)$$

Priority orderings are negotiated through Algorithm 1. By prioritizing robots that have fewer path prospects, we force those robots that have more options to deviate from their preferred (best) plans, and to give way to the robots that have fewer options. Figure 6 illustrates how different robot sizes affect the available path prospects (and hence the priority ordering).

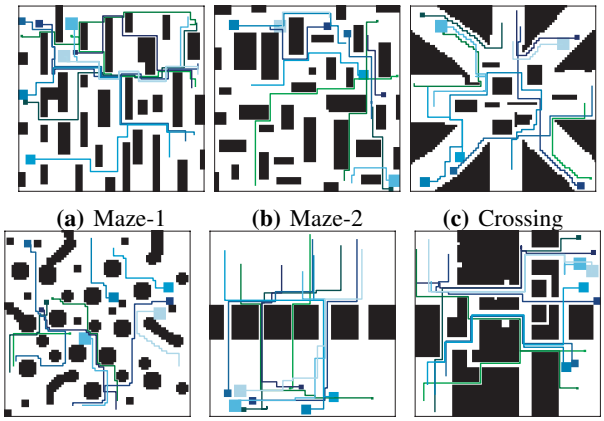
V. EVALUATION

We implement our method in grid-worlds. This allows us to easily create valid graphs G_n for all robots, implement the corresponding TRIM function, and create a set of effective obstacles $\tilde{\mathcal{O}}_n$ for any robot r_n by inflating original obstacles in $\mathcal{O}$ by ρ_n. We note that this dilation can be done more generally by applying Minkowski addition.

We evaluate the performance of our method in two experiments. The first experiment (**S1**) tests the performance of our method in a large environment with a large number of robots. We produce a map of size 150×150 and use a team of 100 robots of size ranging from 1 to 4, in equal proportion, with a communication range of 50. We generate 500 problems and record the performance of all seven algorithms. The second experiment (**S2**) tests the methods across different types of environment. We generate six different cluttered grid-worlds, depicted in Figure 7, of size 75×75. We use a team of 10 robots of five different sizes, with two robots per size, and sizes ranging from 1 to 5. For each base environment, we generate 500 problems (random assignments), and record the performance of the solutions provided by our algorithm (with two alternative tie-break options to guarantee strict orderings), as well as by five additional benchmark algorithms (described below). We solve each problem across communication ranges c that vary between 30 and 50.

A. Benchmarks

In order to test the efficacy of our prioritization method, we perform an ablation analysis. The aim of this ablation study is to identify the efficacy of our proposed *path prospects* heuristic by isolating its two key components: *(i)* the spatial area within which it is applied, and *(ii)* the consideration of the robot-environment coupling. To this end, we implement seven variant schemes for online decentralized prioritization; we focus our experimental analysis to comparable methods that scale *at most linearly* with the number of neighboring robots. Four of these schemes incorporate state-of-the art heuristics, two of the schemes represent our proposed method, and the final scheme incorporates a random rule:



(a) Maze-1 (b) Maze-2 (c) Crossing

(d) Clutter (e) Corridor (f) Tunnel

Fig. 7: Examples of path solutions (blue lines) for the six maps used in our problem sets. In each problem, 10 robots (blue squares) of five different sizes are assigned random start and goal positions.

(1) ***Naive Surroundings (NS):*** This prioritization heuristic follows the idea in [7], whereby robots with the most cluttered surrounding workspace are prioritized. Our implementation of this method counts the number of original obstacles in $\mathcal{O}$ within a range $z = 30$ (which corresponds to the best performing range found via grid-search). This variant does not consider the coupling between robot mobility and the environment, and we term it *naive*. We break ties by prioritizing robots with longest remaining paths. **(2)** ***Coupled Surroundings (CS):*** This prioritization heuristic also follows [7], yet we adapt it to consider the coupling between robot mobility and the environment, whereby effective obstacles in $\tilde{\mathcal{O}}$ are counted (instead of original obstacles). When robot priorities are equal, we tie-break by giving a higher priority to the robot that has the longest remaining path. **(3)** ***Longest First (LF):*** This method prioritizes the robot that has the longest remaining path to its goal, which corresponds to the heuristic used in [20]. When robot priorities are equal, we tie-break by giving a random priority order. **(4)** ***Forwards Looking (FL):*** We consider a *naive* approach that disregards the coupling of robot mobility and the environment. It is naive in that it uses original obstacles in $\mathcal{O}$ instead of obstacles in $\tilde{\mathcal{O}}$ to compute the number of path options. The number of path options is considered in the area that contains paths with a cost less than the cost of the currently longest path known, as specified by Alg. 4. We tie-break this method by prioritizing robots with the longest remaining paths. **(5)** ***Path Prospects (PP-R):*** This method implements our path prospect algorithm. We tie-break randomly. **(6)** ***Path Prospects (PP-LF):*** This method implements our path prospect algorithm. We tie-break with longest-first. **(7)** ***Random (R):*** Finally, we also implement a prioritization rule that randomly assigns the priority order.

B. Results

We evaluate the seven algorithm variants by computing two performance metrics: we consider the percent increase in makespan and flowtime, over the ideal makespan and flowtime, respectively, that assumes a collision-free world without robot interactions.

Figure 8 shows results for the large map used in experiment **S1**. Our two proposed methods ***PP-R*** and ***PP-LF*** lie on the empirical Pareto front (i.e., lowest values over both dimensions). Numerical results for experiment **S2** show that our two proposed methods ***PP-R*** and ***PP-LF*** lie on the empirical Pareto front (i.e., lowest values over both dimensions) across different environments. Compared to ***LF***, our method provides a valuable trade-off when flowtime is important. When comparing ***PP-R*** and ***LF*** to ***CS***, ***CS*** incurs a loss of performance in makespan or flowtime performance, or both. This indicates that the *area* within which path options are considered is important. Overall, our two methods consistently outperform the naive variant, ***FL***, which uses the same area for computing path prospects (i.e., forwards vertices), but disregards the robots' mobility within this area. This demonstrates the importance of considering the coupling between the robot and its environment.

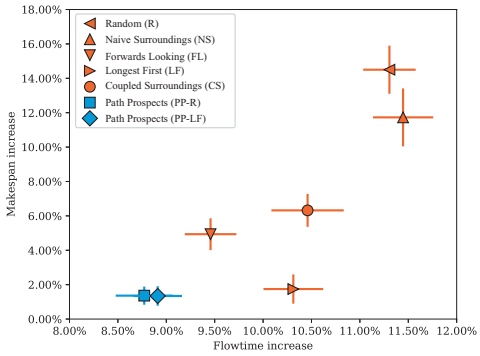


Fig. 8: Experiment **S1** obtained on a large map of size 150×150 with 100 robots. Percentage increase over the ideal flowtime and ideal makespan, for the seven variant prioritization heuristics. We show a 95% confidence interval. Blue nodes correspond to path prospect heuristics, red nodes represent the alternate benchmarks.

We also compute the success rates for the seven algorithms, averaged over all environments. The highest success rates are achieved by our two methods, ***PP-R*** at 95.7% and ***PP-LF*** at 94.1%, with all other methods in the range 82%-93%.

VI. Conclusion

We presented a method for online prioritized path planning for teams of robots. Our decentralized, decoupled planning algorithm provides a deadlock-free means of negotiating path plans among robots, and uses a prioritization heuristic that is based on $\mathbb{Z}_2$ coefficient homology classes, which quantifies a robot's number of available path options. The usage of algebraic topology gives an effective means of designing the prioritization heuristic without having to consider the set of all possible paths in the discrete domain that the robots can physically take to reach their respective goals. We compared our method to five alternate heuristics to show that it provides an interesting alternative to other prioritized, scalable methods by operating on a different point of the Pareto front; in most environments test, our method provides the only point on the Pareto front.

VII. Acknowledgement

We gratefully acknowledge the support of ARL grant DCIST CRA W911NF-17-2-0181. This work incorporates results from the research project Co-Evolving Built Environments and Mobile Autonomy for Future Transport and Mobility funded by the Centre for Digital Built Britain, under InnovateUK grant number RG96233.

REFERENCES

[1] K. Azarm and G. Schmidt. Conflict-free motion of multiple mobile robots based on decentralized motion planning and negotiation. In *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, volume 4, pages 3526–3533. IEEE, 1997.

[2] M. Bennewitz, W. Burgard, and S. Thrun. Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobile robots. *Robotics and autonomous systems*, 41(2-3):89–99, 2002.

[3] S. Bhattacharya, R. Ghrist, and V. Kumar. Persistent homology for path planning in uncertain environments. *IEEE Transactions on Robotics (T-RO)*, 31(3):578–590, March 2015. DOI: 10.1109/TRO.2015.2412051.

[4] S. Bhattacharya, M. Likhachev, and V. Kumar. Topological constraints in search-based robot path planning. *Autonomous Robots*, 33(3):273–290, October 2012. DOI: 10.1007/s10514-012-9304-1.

[5] M. Cáp, P. Novák, M. Seleckỳ, J. Faigl, and J. Vokffnek. Asynchronous decentralized prioritized planning for coordination in multi-robot system. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 3822–3829. IEEE, 2013.

[6] Y. Chen, M. Cutler, and J. P. How. Decoupled multiagent path planning via incremental sequential convex programming. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 5954–5961. IEEE, 2015.

[7] C. M. Clark, T. Bretl, and S. Rock. Applying kinodynamic randomized motion planning with a dynamic priority system to multi-robot space systems. In *Aerospace Conference Proceedings, 2002. IEEE*, volume 7, pages 7–7. IEEE, 2002.

[8] A. A. Deshpande and K. R. Nataraj. A review: Priority based motion control of multiple robot systems. *International Journal of Innovative Research in Science, Engineering and Technology*, 2(1), 2013.

[9] J. Enright and P. R. Wurman. Optimization and coordinated autonomy in mobile fulfillment systems. In *Automated action planning for autonomous mobile robots*, pages 33–38, 2011.

[10] M. Erdmann and T. Lozano-Pérez. On multiple moving objects. *Algorithmica*, 2(1-4):477, 1987.

[11] C. Ferner, G. Wagner, and H. Choset. ODrM* optimal multirobot path planning in low dimensional search spaces. In *2013 IEEE International Conference on Robotics and Automation*, pages 3854–3859. IEEE, 2013.

[12] P. Grippa, D. A. Behrens, C. Bettstetter, and F. Wall. Job selection in a network of autonomous uavs for delivery of goods. *Robotics: Science and Systems*, 2017.

[13] W. Hönig, T. S. Kumar, L. Cohen, H. Ma, H. Xu, N. Ayanian, and S. Koenig. Multi-agent path finding with kinematic constraints. In *Twenty-Sixth International Conference on Automated Planning and Scheduling*, 2016.

[14] N. Hyldmar, Y. He, and A. Prorok. A fleet of miniature cars for experiments in cooperative driving. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[15] K. Kant and S. W. Zucker. Toward efficient trajectory planning: The path-velocity decomposition. *The international journal of robotics research*, 5(3):72–89, 1986.

[16] K. Okumura, M. Machida, X. Défago, and Y. Tamura. Priority inheritance with backtracking for iterative multi-agent path finding. *IJCAI*, 2019.

[17] R. Regele and P. Levi. Cooperative multi-robot path planning by heuristic priority adjustment. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 5954–5959. IEEE, 2006.

[18] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219:40–66, 2015.

[19] D. Silver. Cooperative pathfinding. In *Artificial intelligence and interactive digital entertainment*, volume 1, pages 117–122, 2005.

[20] J. P. Van Den Berg and M. H. Overmars. Prioritized motion planning for multiple robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 430–435. IEEE, 2005.

[21] P. Velagapudi, K. Sycara, and P. Scerri. Decentralized prioritized planning in large multirobot teams. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 4603–4609. IEEE, 2010.

[22] G. Wagner and H. Choset. M*: A complete multirobot path planning algorithm with performance bounds. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 3260–3267. IEEE, 2011.

[23] J. Yu and S. M. LaValle. Structure and intractability of optimal multi-robot path planning on graphs. In *AAAI*, 2013.