WAP: Wavefront Arithmetic Propagation Algorithm and Implementation for Motion Planning Among Dynamic Obstacles

Shijie Bao[1][0009−0008−3258−1117], Shizhe Zhao[1][0000−0002−2351−6152], and Zhongqiang Ren[1,†][0000−0003−2880−8653]

[1]Global College, Shanghai Jiao Tong University, Shanghai, China
{thetoror,shizhe.zhao,zhongqiang.ren}@sjtu.edu.cn

Abstract. This paper considers Motion Planning among Dynamic Obstacle (MPDO), which seeks collision-free path for a point robot with maximum speed limit from given start to goal position in 2D continuous space among dynamic obstacles along known trajectories while minimizing the arrival time. Existing algorithms can either find feasible solutions for MPDO without solution optimality guarantees, or solve special cases of MPDO to optimality using continuous Dijkstra paradigm. A major difficulty in MPDO is to compute the reachable space of the robot among dynamic obstacles which involves boolean operation over complex point sets. This paper proposes a new method Wavefront Arithmetic Propagation (WAP) to address this difficulty by converting these complex boolean operations to efficient arithmetic operations on a new data structure which we refer to as the waveseg graph. Additionally, we implemented the proposed WAP, which is rare in the field of continuous Dijkstra, and compare against mixed integer programming (MIP) baselines. Results show that, WAP can find optimal paths in complex environments among tens of dynamic obstacles in less than a second while achieving earlier arrival time than the baseline methods.

Keywords: Motion and Path Planning, Dynamic Obstacles

1 Introduction

Motion planning is of fundamental importance in robotics [5, 10, 14, 15]. This paper considers Motion Planning among Dynamic Obstacle (MPDO), which seeks a collision-free path for a point robot with maximum speed limit from a given start to a goal position in 2D continuous space while minimizing the arrival time. In particular, the obstacles in MPDO can be non-convex, static, transient (i.e., appear or disappear at any time), or dynamic (i.e., moving along known trajectories). When the obstacles are static, there are algorithms for finding an optimal or bounded sub-optimal solution in polynomial time [5, 14, 15]. In the presence of dynamic obstacles, this problem is known to be NP-Hard [4].

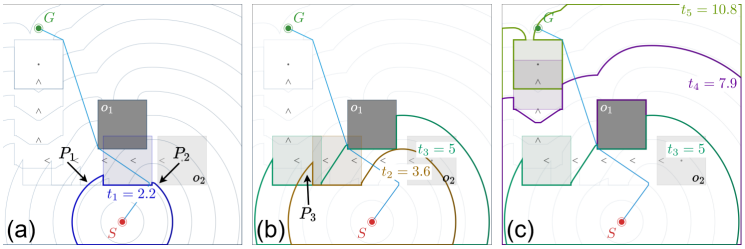


Fig. 1: An MPDO instance and the earliest-arrival solution produced by WAP. There is a static obstacle o_1 (dark square) and a dynamic obstacle o_2, with light gray squares indicating the start and gray trianglar waypoints indicating its trajectory. The time axis is discretized with $t_\Delta = 0.1$. The light blue curves show the boundaries of the reachable sets for every 1 second. The blue line from S to G is the computed earliest-arrival path. (a) The deep blue curve shows the boundary of the reachable set at time $t_1 = 2.2$. For boundary points P_1 and P_2, the robot cannot reach them earlier than t_1 (b) The brown and green curves show the boundary of the reachable set at times $t_2 = 3.6$ and $t_3 = 5$ respectively. Due to dynamic obstacles, reachable points may become inaccessible at later times. For example, point P_3 is reachable at time t_2 but blocked by the obstacle o_2 at time t_3. (c) The violet and olive green curves show the boundary of the reachable set at times $t_4 = 7.5$ and $t_5 = 10$ respectively. WAP terminates when the reachable set contains the goal point G at time t_5.

Existing methods can find optimal solutions for only a few special cases of MPDO. When obstacles are convex polygons moving along fixed directions with constant speeds, an optimal solution can be found by constructing an "accessibility graph" [8] in 3D where time is added as the third dimension. When obstacles are convex polygons, mixed-integer programming [1, 21, 24, 26] can address the problem. When obstacles are polygons, static or transient, Continuous Dijkstra (CD) paradigms [7,11,17,18] can be leveraged to compute an optimal solution by alternating between propagating the wavefront and detecting events where the wavefront intersects with each other or with the obstacles. When both obstacles and robot path are rectilinear, an improved version of the CD with reduced time complexity has recently been developed in [16]. To address transient obstacles, existing CD methods often rely on assumptions on "monotonous paths" [7, 16], which, intuitively speaking, only consider monotonous growing reachable sets and cannot handle the case where dynamic obstacles may re-enter the previously reachable sets as time evolves. Finally, while these CD-based methods find optimal solutions in theory, we are not aware of any implementations or any numerical results of these algorithms.

To address the challenges, we propose a new method Wavefront Arithmetic Propagation (WAP) that can handle general (non-convex) obstacles as opposed to mixed-integer programming [1, 21], and can plan general paths among dynamic obstacles as opposed to the monotonous paths in CD [7, 16]. While WAP also relies on the idea of wavefront propagation, WAP is fundamentally different from the existing CD-based methods. Instead of relying on the notion of events as

in CD, WAP discretizes the time-axis into time steps, and propagates the wavefronts between time steps. WAP is inspired by the Huygens-Fresnel principle [3]: this provides an intuitive understanding that finding the reachable space can be viewed as independently propagating each point on the wavefront and taking the union of the results. Then, at each time step, WAP computes the resulting wavefronts based on boolean operations over the previous wavefronts, the newly swept areas and the obstacles. There is no obvious way to directly conduct these boolean operations as it is hard to represent arbitrary set of points. We bypass this difficulty by (i) introducing the concept of "wave segment" (waveseg) and a data structure Waveseg Graph (WGs), to capture the topology of the wavefront for boolean operations; (ii) converting the boolean operations over sets to arithmetic operations in WGs; and (iii) developing efficient algorithms to conduct these arithmetic operations.

Besides the proposed algorithm, another contribution is the implementation and numerical evaluation, which is rare in the field of CD for motion planning. We compare WAP against recent mixed-integer programming (MIP) approaches [21, 26] that also rely on discrete time step representation and can find optimal solutions among convex polygonal obstacles. The results show that WAP runs 2.93× to 4.37× faster than the baselines, provides optimality guarantees in arrival time, and achieves near-optimal path lengths. Our video shows a complex environment with 30 dynamic obstacles where WAP finds an optimal solution in under a second.

2 Other Related Work

While finding the optimum for MPDO is challenging, there are many algorithms to compute a *feasible* solution. By discretizing the workspace into a graph, search-based planners [20, 23] can find an optimal solution within the graph, whose quality is determined by the resolution of the discretization. Path-velocity decomposition methods [13] can also be leveraged to find a feasible solution by first finding a path among static obstacles and then finding the speeds along the path to avoid the dynamic obstacles. Sampling-based methods [6, 12] have been applied for generalizations of MPDO with motion constraints and can be used to find feasible solutions. To quantify the deviation of these feasible solutions from the optimum, one can find tight lower bounds [22] to provide posteriori sub-optimality bound of the obtained solution.

By discretizing the time dimension, one can also formulate the MPDO as a mixed-integer linear program (MILP) [1, 21, 24] or mixed-integer conic program (MICP) [26, 27], which uses big-M constraints to encode obstacle collision constraints at each time step, and invoke solvers such as Gurobi to solve the formulation to optimality. We compare our WAP against this type of methods.

3 Problem Description

Let $\mathcal{W} \subset \mathbb{R}^2$ denote a bounded 2D workspace. Let $t_\Delta > 0$ denote a fixed time interval and the time dimension is discretized into time steps $t_i = i \cdot t_\Delta, i = 0, 1, 2 \cdots$. For each time t_i, let $\mathbb{O}_i = \{o_i^1, o_i^2, ...\}$ denote a set of time-dependent transient obstacles. We assume all obstacles are arbitrary finite-edge polygons and their edges do not overlap[1]. The boundaries of obstacles are traversable, i.e., each $o \in \mathbb{O}_i$ is an open point set. Let O_i denote the unified non-traversable space, i.e., $O_i = \bigcup_{o \in \mathbb{O}_i} o$.

The robot is a point that can move in any direction with a speed limit of $v_{max} \in \mathbb{R}^+$. Let $\vec{s}, \vec{g} \in \mathcal{W}$ represent the start and goal location of the robot. Let $\pi : \langle \vec{p}_0, \vec{p}_1, \ldots, \vec{p}_n \rangle$, where $\vec{p}_0 = \vec{s}$, $\vec{p}_n = \vec{g}$, denote a path of the robot that reach $\vec{g}$ at time t_n, and let $\vec{p}_i$ denote the i-th waypoint at time t_i. MPDO seeks a path π from s to g with the minimum t_n that satisfies: (i) speed constraint: $\forall i \leq n, |\vec{p}_i - \vec{p}_{i+1}| \leq v_{max} t_\Delta$, and; (ii) collision free: $\forall i \leq n, \vec{p}_i \notin O_i$.

Remark 1. In this problem, we only consider transient obstacles that appear or disappear at each time step. The trajectory of a moving obstacle can be converted to transient obstacles by sampling its position at each time step. This problem setting does not explicitly consider collision avoidance between two adjacent time steps during the transition. In practice, this can be avoided by applying obstacle inflation that considers $v_{max} t_\Delta$.

4 Methods

4.1 Basic Concepts

We employ some basic concepts from topology that will be used in this work.

Definition 1. *For a given point set $S \subset \mathcal{W}$, let $Int(S)$ and ∂S denote the* interior *and* boundary *respectively. S is a regular set (RS) if $\partial(Int(S)) = \partial S$ [25] (Fig 2(a) illustrates these concepts). Let '$\neg$' denote the complement operator of a point set, i.e., $\neg S = \mathcal{W} \backslash S$. Let $B(\vec{p}, d)$ denote a closed RS in circle centered at $\vec{p}$ with radius d, and let $B_\epsilon(\vec{p}) := \lim_{\epsilon \to 0} B(\vec{p}, \epsilon)$.*

Let R_n denote the set of points that the robot can reach at time t_n, starting from t_0 at position $\vec{s}$. Then R_n can be represented recursively as follows:

$$\begin{cases} R_0 = \{\vec{s}\}, & \text{(base case)} \\ U_n = R_{n-1} \oplus B_\Delta, & \text{(propagation)} \\ R_n = U_n \cap \neg O_n, & \text{(restriction)} \end{cases} \tag{1}$$

where $B_\Delta = B(\vec{0}, v_{max} t_\Delta)$ and $\oplus$ is the Minkowski sum of two sets, i.e., $X \oplus Y = \{\vec{x} + \vec{y} | \vec{x} \in X, \vec{y} \in Y, X, Y \subset \mathcal{W}\}$. In Eq. (1), U_n denotes the point set that all points in R_{n-1} moves in all directions in $\mathcal{W}$, and we refer to this process as *propagation*. Then U_n is clipped by O_n to ensure the reachable set is collision-free at t_n, and we refer to this process as *restriction*.

[1] We can always merge polygons with overlapping edges into one.

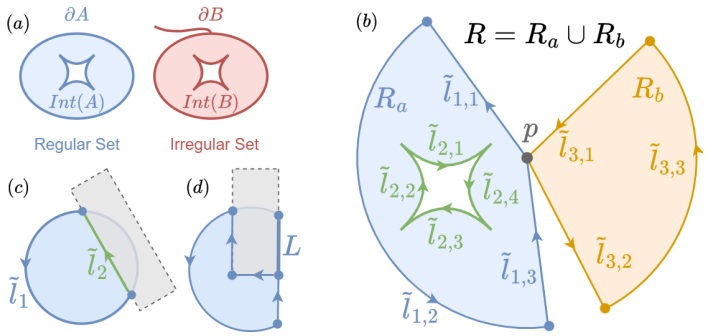


Fig. 2: (a) Examples of regular sets and irregular sets. (b) A reachable set R with two components R_a, R_b and a hole bounded by wavesegs $\tilde{l}_{2,1}, \cdots, \tilde{l}_{2,4}$. (c) A full-circle waveseg is split into a circular arc (blue) and a segment (green) by the boundary of an obstacle. (d) The restriction introduces the component L, which is not a regular set.

Definition 2. *(Wavefront and Wavesegs) The boundary of the reachable set at t_n, ∂R_n, is called* wavefront. *∂R_n can be expressed as a set of directional curves $\{\tilde{l}_1, \tilde{l}_2, ..\}$, where each $\tilde{l} \in \partial R_n$ is called a* waveseg. *The direction of waveseg will be defined in Def. 3. A waveseg $\tilde{l}$ is a closed set consisting of three parts $l^s, l^e, \tilde{l}^o$. Here, l^s and l^e are start and end points of the curve, respectively, while $\tilde{l}^o$ is the open set of l, excluding l^s and l^e. This means that $\tilde{l}^o$ never intersects with other wavesegs. Specifically, when the waveseg $\tilde{l}$ is a full circle, we can choose an arbitrary point $p \in \tilde{l}$ and set $l^s = l^e = p$. All wavesegs form multiple loops without any self-intersection (See in Exp 1).*

Definition 3. *(Direction of Wavesegs) The direction of $\tilde{l}$ at a point $\vec{p} \in \tilde{l}$ is defined by the unit vector tangent to $\tilde{l}$ at $\vec{p}$, pointing from l^s to l^e, denoted as $\vec{p}(\tilde{l})$. For a reachable set R, the directions of wavesegs determine the interior and exterior of R. Let $n^l_{\curvearrowleft}(\vec{p})$ and $n^l_{\curvearrowright}(\vec{p})$ denote the unit normal vector of $\vec{p}(\tilde{l})$ pointing to left and to right w.r.t. l, respectively. Let $\mathbb{N}(\tilde{l}) = \{\vec{p} \oplus B_\epsilon(\vec{p}) | \vec{p} \in \tilde{l}\}$ denote all nearby points of $\tilde{l}$. A point $\vec{q} \in \mathbb{N}(\tilde{l})$ is left (or right)[2] to $\tilde{l}$ if $\exists \vec{p} \in \tilde{l}$ that $n^l_{\curvearrowleft}(\vec{p}) \cdot (\vec{q} - \vec{p}) > 0$ (or $n^l_{\curvearrowright}(\vec{p}) \cdot (\vec{q} - \vec{p}) > 0$ for right). For a point p that is not on the boundary ($p \notin \partial R$), p is in interior (exterior) of R if (i) $\exists \vec{q}$ that is left (right) to any waveseg of ∂R; and (ii) there exists a continuous path that connects $\vec{q}$ and $\vec{p}$ without intersecting any waveseg. Jordan Curve Theorem [9] ensures that a point $\vec{p} \in R$ cannot be both in the interior and exterior of R.*

Example 1. (Components of Reachable Set) In Fig. 2(b), the regular set R can be represented by two regular sets $\{R_a, R_b\}$. The boundary of R consists of three loops $\tilde{l}_{1,...}$ (blue), $\tilde{l}_{2,...}$ (green) and $\tilde{l}_{3,...}$ (orange). Their directions define the interior and exterior of R. Specifically, all points inside the region defined

[2] We define the left/right relationship only for nearby points of a waveseg.

by green wavesegs are exterior points, creating a hole in R_a. Also note that $\partial R_a \cap \partial R_b = \{p\}$, while $Int(R_a) \cap Int(R_b) = \emptyset$.

Exp. 1 shows that, due to time-dependent transient obstacles, R_n may contains multiple reachable regions with holes, and ∂R may form multiple loops. In the remaining content, we refer the different reachable regions and loops as components of R and ∂R respectively. For a given waveseg $\tilde{l}$, let $R(\tilde{l})$ and $\partial R(\tilde{l})$ denote the corresponding component that $\tilde{l}$ is part of.

Remark 2. At any time $t_n > 0$, each waveseg can be represented by either a circular arc or segment. Intuitively, during the propagation from t_{n-1} to t_n, for a point $\vec{p} \in \partial R_{n-1}$, $\vec{p} \oplus B_\Delta$ is a subset of U_n, and $\partial(\vec{p} \oplus B_\Delta)$ has a single waveseg $\tilde{l}$ (i.e., a full circle). Then the restriction may introduce a segment from the border of an obstacle that splits $\tilde{l}$ into a circular arc $\tilde{l}_1$ and a segment $\tilde{l}_2$ (see in Fig 2(c)). We will show that wavesegs of a propagation can still be represented by arcs and segments (see in Exp 2).

Note that R_n guarantees that a collision-free path exists for all points $\vec{p} \in R_n$. However, the lengths of these paths are not necessarily the same. Since our objective is to minimize arrival time, we never split a reachable set based on the path length metric.

Remark 3. R_n may contains regular set (e.g., U_n), segments L and points P (e.g., R_0), where L and P are not part of RS. Fig. 2(d) shows an scenario of generating a segmental reachable set that not be part of any regular set. To simplify discussion, for the remainder of the paper, we will refer to R_n by its regular set component and disregard L, P. In Sec. 4.4, we will show that the proposed method still works when L, P are considered.

Now we demonstrate the result of propagating a waveseg $\tilde{l}$. Let $S_w(\tilde{l}) \subset U_n$ denote the newly swept region of a waveseg $\tilde{l}$ propagates from t_{n-1} to t_n, i.e.:

$$S_w(\tilde{l}) = S_w(l^s) \cup S_w(l^e) \cup S_w(\tilde{l}^o) \supseteq (\tilde{l} \oplus B_\Delta) \backslash R_{n-1}, \tag{2}$$

where $S_w(\tilde{l}^o)$ consists of points generated by moving all points in $\tilde{l}$ to the right direction of $\tilde{l}$, i.e., $\{\vec{p} + v_{max} t_\Delta n^l_\frown(\vec{p}) | \vec{p} \in \tilde{l}^o\}$; and $S_w(l^s)$ (or $S_w(l^e)$) is a quarter circle centered at l^s (or l^e) and propagates in direction $n^l_\frown(l^s)$ (or $n^l_\frown(l^e)$) without overlapping with $S_w(\tilde{l}^o)$. Exp. 2 demonstrates the propagation.

Example 2. (Waveseg Propagation) Fig. 3(a,b) shows the result of propagating a single waveseg $\tilde{l}$. $S_w(l^o)$ is shown in blue, dashed arrows at $v \in \tilde{l}$ are in direction $n^l_\frown(\vec{v})$ with length $v_{max} t_\Delta$. $S_w(\tilde{l}^s), S_w(\tilde{l}^e)$ are shown in green. We can see that using quarter circles is enough to represent the propagated region such that $S_w(\tilde{l}) \supseteq (\tilde{l} \oplus B_\Delta) \backslash R_{n-1}$. Figure 3(c,d) demonstrates the result of propagating a wavefront. We can see that all wavesegs expand towards the exterior region, resulting in a larger reachable region (represented by the outer blue and green wavesegs) and a smaller hole (represented by the inner blue wavesegs).

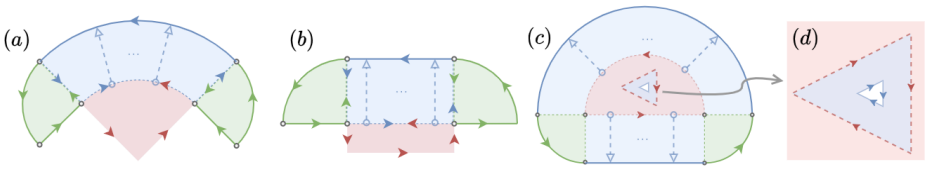


Fig. 3: Red areas represent a subset of the reachable region from the previous iteration. Green and blue areas represent the swept regions generated by propagation. Solid arrows indicate the direction of wavesegs, while hollow arrows show the movement direction of a point during propagation. (a) Propagation of a single circular waveseg; (b) Propagation of a single segment waveseg; (c) Propagation on a wavefront; (d) Propagation of a hole in the wavefront;

U_n in Eq (1) can be expressed as an union of multiple regular sets (RSs)

$$U_n = R_{n-1} \cup \Big(\bigcup_{\tilde{l} \in \partial R_{n-1}} S_w(\tilde{l}) \Big). \tag{3}$$

Eq (3) suggests that we can compute U_n and R_n via boolean operators on finite number of RSs. For this purpose we introduce the concept, *indicator scalar field* (ISF) of a regular set $X \subset \mathcal{W}$; and we define boolean operators to compute ISF of X from subsets of it.

Definition 4. *(Indicator Scalar Field) For an RS, $X \subset \mathcal{W}$, $\mathcal{F}(\partial X)(\cdot)$ defines an indicator scalar field (ISF), for a point $\vec{p} \in \mathcal{W}$:*

$$\mathcal{F}(\partial X)(\vec{p}) = \begin{cases} 0, \ \textit{if } \vec{p} \notin \big(\partial X \cup Int(X)\big) & \textit{(exterior point)} \\ 1, \ \textit{if } \vec{p} \in \partial X & \textit{(boundary point)} \\ 2, \ \textit{if } \vec{p} \in Int(X) & \textit{(interior point)}, \end{cases} \tag{4}$$

Next we define the boolean operators of $\mathcal{F}(\cdot)$ for two RSs $X_1, X_2 \subset \mathcal{W}$.

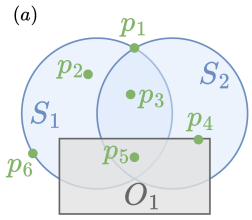
Union ISF of $X = X_1 \cup X_2$ is expressed as follows:

$$\mathcal{F}\big(\partial X\big)(\vec{p}) = \begin{cases} f\big(\mathcal{F}(\partial X_1) + \mathcal{F}(\partial X_2)\big) & \text{for } \vec{p} \notin \partial X_1 \cap \partial X_2 \quad (5) \\ r(X_1 \cup X_2, \partial X_1 \cap \partial X_2, \vec{p}) & \text{for } \vec{p} \in \partial X_1 \cap \partial X_2 \ , \quad (6) \end{cases}$$

where $f(x) = \min(x, 2)$ is a saturation function indicates that if $\vec{p}$ is an interior point any of X_1, X_2 then it must be an interior point of $X_1 \cup X_2$; and $r(X, D, \vec{p})$ is a repairing function that considers $B_\epsilon(\vec{p})$ to ensure $\mathcal{F}(\partial X)$ is defined when $\vec{p}$ locates at both boundary of X_1 and X_2, that

$$r(X, D, \vec{p}) = \begin{cases} 0, & \text{if } \forall \vec{q} \in B_\epsilon(\vec{p}) \backslash D, \ \text{that } \mathcal{F}(\partial X)(\vec{q}) = 0 \\ 2, & \text{if } \forall \vec{q} \in B_\epsilon(\vec{p}) \backslash D, \ \text{that } \mathcal{F}(\partial X)(\vec{q}) = 2 \\ 1, & \text{otherwise.} \end{cases} \tag{7}$$

Eq (7) indicates that a $\vec{p}$ is an exterior (or interior) point of X if all nearby points $B_\epsilon(\vec{p}) \backslash D$ are exterior (or interior) points, while the point $\vec{p}$ is on the boundary if there exist both interior and exterior points in $B_\epsilon(\vec{p}) \backslash D$.



(b)

$\mathcal{F}(\partial\cdot)(\cdot)$	p_1	p_2	p_3	p_4	p_5	p_6
S_1	1	2	2	0	2	1
S_2	1	0	2	2	2	0
O_1	0	0	0	1	2	0
R	1	2	2	1	0	1

$R = (S_1 \cup S_2) \cap \neg O_1$

Fig. 4: (a) A regular set $R = (S_1 \cup S_2) \cap \neg O_1$ is defined by boolean operations on multiple regular sets. (b) The ISF value of points $p_1, \cdots, p_6$ in each regular set.

Complement ISF of the complement set is:

$$\mathcal{F}(\partial(\neg X)) = 2 - \mathcal{F}(\partial X), \tag{8}$$

which turns all interior points to exterior (and vice-versa) while leaving all boundary points unchanged.

Intersection ISF of $X = X_1 \cap X_2$ is expressed as follows:

$$\mathcal{F}\big(\partial X\big)(\vec{p}) = \begin{cases} 2 - f(4 - (\mathcal{F}(\partial X_1) + \mathcal{F}(\partial X_2))), & \text{for } \vec{p} \notin \partial X_1 \cap \partial X_2, \\ r(X_1 \cap X_2, \partial X_1 \cap \partial X_2, \vec{p}) \ \ \text{(Eq. (7))}, & \text{for } \vec{p} \in \partial X_1 \cap \partial X_2, \end{cases} \tag{9}$$

where Eq. (9) comes from De Morgan's laws that $X_1 \cap X_2 = \neg(\neg X_1 \cup \neg X_2)$ with Eq. (5) and (8). Exp. 3 shows how to leverage the boolean operators to compute the ISF value of a point.

Example 3. Fig. 4(a) shows a regular set $R = (S_1 \cup S_2) \cap \neg O_1$ that consists of multiple subsets, and Fig. 4(b) shows the ISF value of points $p_1, \cdots, p_6$ in S_1, S_2, O_1 and R. By applying Eqs. (5)–(9), we can compute the ISF values as shown in Fig. 4(b). For instance, $\mathcal{F}(\partial R)(p_4) = 2 - f(4 - (2 + 1)) = 1$ (boundary point) and $\mathcal{F}(\partial R)(p_5) = 2 - f(4 - (2 + 0)) = 0$ (exterior point). Note that computing $\mathcal{F}(\partial R)(p_1)$ requires the repairing function (Eq. (7)), as p_1 lies on the boundary of both S_1 and S_2. In Sec. 4.2, we will show that our method does not need to address such case.

We should strictly distinguish between a set of regular sets and their composition via boolean operators. For example, in Fig. 4, $\mathbb{R} = \{S_1, S_2, O_1\}$ is a set of regular sets, while R is a new regular set composed by boolean operators on $\mathbb{R}$. $\mathbb{R}$ and R represent different set of wavefronts. Next, we introduce *Waveseg Graph*, which allows us to compute R_n within finite steps by leveraging the concept of ISF.

Definition 5. *(Waveseg Graph) For a set of regular sets, $\mathbb{P} = \{P_1, \ldots, P_n\}$, let $\tilde{l}_{a,i}$ denote the i-th waveseg of $P_a \in \mathbb{P}$. A Waveseg Graph (WG), denoted as $G^w(\mathbb{P}) = (V, E)$ is a directed graph constructed from all wavesegs in $\mathbb{P}$. In short, V consists of the ending points of all wavesegs in $\mathbb{P}$ and the points located on*

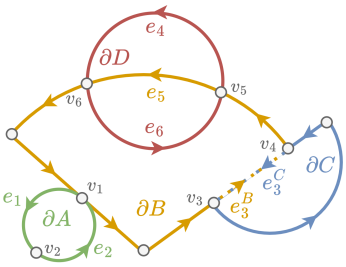


Fig. 5: Waveseg Graph, $G^w(A, B, C, D)$, is built from regular sets A, B, C, D. Each regular set forms a loop in WG. Vertex v_1 is a tangent point between ∂A and ∂B, splitting ∂A into e_1, e_2. Here, v_2 is an arbitrarily chosen endpoint for the full-circle waveseg of ∂A. Two wavefronts, $\partial B, \partial C$ have a waveseg overlapping at v_3 and v_4, forming a duplicated edge $e_3 = e_3^B = e_3^C$, which is indicated by the dashed line. Edges e_4, e_5, e_6 connect vertices v_5, v_6, where e_4, e_5 are in the same direction, indicating that G^w can be a multi-graph.

multiple wavesegs. Each $e \in E$ is a subset of one or more wavesegs. Formally speaking:
Vertices *For a single waveseg $\tilde{l}$, its start and end points are graph vertices, i.e., $l^s, l^e \in V$. For a pair of wavesegs $\tilde{l}_{a,i} \in P_a, \tilde{l}_{b,j} \in P_b$, where $a \neq b$: (1) if $\tilde{l}_{a,i}$ and $\tilde{l}_{b,j}$ intersect at $\vec{p}$, then $\vec{p} \in V$; (2) if $\tilde{l}_{a,i}$ and $\tilde{l}_{b,j}$ overlap along a curve whose endpoints are $\vec{p}$ and $\vec{q}$, then $\vec{p}, \vec{q} \in V$.*
Edges *A waveseg $\tilde{l}$ is split into multiple edges by $v \in V \cap \tilde{l}$, for an edge $e = (u, v)$, let $A(e)$ denote the set of wavesegs intersect or overlap with e: (1) if u, v are from ending points of overlapped wavesegs $\tilde{l}_1, \tilde{l}_2$, then e has both directions of $\tilde{l}_1, \tilde{l}_2$, and $A(e) \supseteq \{\tilde{l}_1, \tilde{l}_2\}$; (2) otherwise e is subset of a single $\tilde{l}$ and has the same direction as $\tilde{l}$, and $A(e) = \{\tilde{l}\}$. We refer to an edge e for which $|A(e)| > 1$ as a* duplicated edge*. All edges are open set excluding their end points, i.e., $V \cap E = \emptyset$.*

WG is a directed cyclic graph where each cycle represents a loop on the wavefront of $P_i \in \mathbb{P}$. WG is also a multi-graph, meaning there can be multiple edges between the same pair of vertices. For example, in Fig. 5, $e_4 = (v_5, v_6)$ and $e_5 = (v_5, v_6)$ are different edges. Next, we show an important property of WG that enables us to compute the reachable set R_n within finite steps.

Lemma 1. *Consider a WG, $G^w(\mathbb{P}) = (V, E)$, where $\mathbb{P} = \{P_1, \ldots, P_n\}$. Then for any points $\vec{p}, \vec{q}$ on the same edge $e \in E$, for all $P_i \in \mathbb{P}$, we have:*

$$\mathcal{F}(\partial P_i)(\vec{p}) = \mathcal{F}(\partial P_i)(\vec{q}). \tag{10}$$

Furthermore, for any $\vec{p}, \vec{q}$ on a continuous curve $c \in \mathcal{W}$, if $c \cap G^w(\mathbb{P}) = \emptyset$ then Eq. (10) also holds.

Sketch Proof Assume there are an edge $e \in E$, a pair of points $\vec{p}, \vec{q} \in e$ and a RS $P_i \in \mathbb{P}$, such that $\mathcal{F}(P_i)(\vec{p}) \neq \mathcal{F}(P_i)(\vec{q})$. Then, moving from $\vec{p}$ to $\vec{q}$ along e must cross the boundary of a RS $P_j \in \mathbb{P}$ (where i, j might be the same), which contradicts the assumption that $\vec{p}, \vec{q} \in e$. □

4.2 WAP Algorithm

We solve MPDO by finding the minimum n such that $g \in R_n$. Computing R_n is equivalent to finding all wavesegs of ∂R, which can be achieved by identifying all

Algorithm 1 WAP

1: **procedure** WAP($\vec{s}, \vec{g}, v_{max}, t_\Delta, \mathcal{W}$)
2: $n, R_0 \leftarrow 0, \{\vec{s}\}$
3: **while** $g \notin R_n$ **do** ▷ Check whether $\mathcal{F}(\partial R_n)(g) = 0$
4: $n \leftarrow n + 1$
5: $\mathbb{P}_n \leftarrow \textsc{Propagation}(R_{n-1}, v_{max}, t_\Delta)$ ▷ Eq. (1) and (2)
6: $G^w \leftarrow \textsc{BuildWG}(\mathbb{P}_n, \mathbb{O}_n)$ ▷ Def. 5
7: $\lambda_1(\cdot), \lambda_2(\cdot) \leftarrow \textsc{CalcAllLambda}(G^w, \mathbb{P}_n, \mathbb{O}_n)$ ▷ Alg. 2
8: $\partial R_n \leftarrow \{e \in G^w \mid \gamma(\lambda_1(e), \lambda_2(e)) = 1\}$ ▷ Eq. (13)
9: **if** $R_n = \emptyset$ **then**
10: **return** No Solution
11: **return** n ▷ g is reachable at t_n

points $\vec{p} \in \mathcal{W}$ such that $\mathcal{F}(\partial R_n)(\vec{p}) = 1$. For this purpose, we can constructed a WG (Def. 5) from all regular sets (RSs) involved in propagation and restriction. According to Lemma 1, we only need to evaluate a single point on each edge of the WG.

Next we establish the expressions to compute $\mathcal{F}(\partial R_n)(\vec{p})$ when $\vec{p}$ is not on a duplicated edge. The handling of duplicated edges is discussed in Sec. 4.4. For simplicity, in the remainder of the paper, we assume that duplicated edges do not exist unless explicitly mentioned.

Let $\mathbb{P}_n = \{R_{n-1}\} \cup \{S_w(\tilde{l}) | \tilde{l} \in \partial R_{n-1}\}$ denote all RSs involved in propagation, and let $G^w : (E, V)$ denote the WG constructed from $\mathbb{P}_n$ and $\mathbb{O}_n$ (obstacle set at t_n). For a point $\vec{p}$ on an edge of G^w, let

$$\begin{cases} \lambda_1(\vec{p}) &= \sum_{P \in \mathbb{P}_n} \mathcal{F}(\partial P)(\vec{p}), & (11) \\ \lambda_2(\vec{p}) &= \sum_{O \in \mathbb{O}_n} \mathcal{F}(\partial O)(\vec{p}), & (12) \\ \gamma(a, b) &= 2 - f(2 - f(a) + f(b)) \ , & (13) \end{cases}$$

Lemma 2. *$\mathcal{F}(\partial R_n)(\vec{p}) = \gamma(\lambda_1(\vec{p}), \lambda_2(\vec{p}))$ holds when $\vec{p}$ is not on a duplicated edge of G^w.*

Proof (Proof sketch). Since $\vec{p}$ is not on a duplicated edge, the repairing function (Eq. (7)) is never invoked, so the saturation function distributes over sums: $f(a + f(b + c)) = f(a + b + c)$ when $a, b, c > 0$. Applying the union operator (Eq. (5)) to $\mathbb{P}_n$ yields $\mathcal{F}(\partial U_n)(\vec{p}) = f(\lambda_1(\vec{p}))$, and similarly $\mathcal{F}(\partial O_n)(\vec{p}) = f(\lambda_2(\vec{p}))$. The result follows by applying the complement operator (Eq. (8)) to O_n and the intersection operator (Eq. (9)) to $U_n \cap \neg O_n$.

Alg. 1 demonstrates the key procedures of WAP. In each iteration, we first propagate all wavesegs from the previous iteration according to Eq. (2) and obtain $\mathbb{P}_n$ (line 5). Then we build the Waveseg Graph, $G^w(\mathbb{P}_n, \mathbb{O}_n)$, based on Def. 5 (line 6). Next, we compute the value of λ_1, λ_2 of all edges (line 7), which will be discussed next. Finally, according to Eq. (13), we can identify all wavesegs

Algorithm 2 Compute λ_1, λ_2 of All Edges

1: **procedure** CalcAllLambda($G^w, \mathbb{P}_n, \mathbb{O}_n$)
2: $E_{todo} \leftarrow E(G^w)$ ▷ Edge set to be evaluated
3: **for** $e \in E_{todo}$ **do**
4: $m \leftarrow$ middle point of e
5: $\lambda_1(e) \leftarrow \sum_{P \in \mathbb{P}_n} \mathcal{F}(\partial P)(m)$ ▷ Eq. (11)
6: $\lambda_2(e) \leftarrow \sum_{O \in \mathbb{O}_n} \mathcal{F}(\partial O)(m)$ ▷ Eq. (12)
7: **if** use spread **then** ▷ Improvement by *spread*, discussed in Sec. 4.3
8: $E_{spread} \leftarrow$ Spread($e, \lambda_1, \lambda_2, \emptyset, G^w, \mathbb{P}_n, \mathbb{O}_n$) ▷ Alg. 3
9: $E_{todo} \leftarrow E_{todo} \backslash E_{spread}$
10: **return** λ_1, λ_2
11: **procedure** $\mathcal{F}(\partial R)(p)$
12: $q, cnt \leftarrow (\infty, \infty), 0$ ▷ q is far enough that outside R
13: $\tilde{qp} \leftarrow$ Ray(q, p)
14: **for** $\tilde{l} \in \partial R$ **do**
15: **for** $v \in$ Intersect($\tilde{qp}, \tilde{l}$) **do** ▷ Compute all intersection points
16: **if** $|vp| = 0$ **then**
17: **return 1** ▷ On boundary
18: $\Delta \leftarrow$ sign($\tilde{qp} \cdot n^l_{\curvearrowleft}(\tilde{v})$) ▷ sign$(x) \Rightarrow -1 : x < 0, 0 : x = 0, 1 : x > 0$
19: $cnt \leftarrow cnt + \Delta$
20: **return** $2 \times cnt$

of ∂R_n by selecting those edges for which $\gamma(\lambda_1(e), \lambda_2(e)) = 1$ (line 8). The algorithm terminates at the first time R_n contains g (line 3), which can be determined by evaluating $\mathcal{F}(\partial R_n)(g)$. There are two cases when there is no solution: (i) R_n becomes an empty set, meaning that there is no reachable point from t_n, then the algorithm terminates and confirms that there is no solution (line 10); (ii) The algorithm keeps iterating and never terminate, e.g., when $\vec{s}, \vec{g}$ are in different connected components.

Alg. 2 computes λ_1, λ_2 for all edges in WG (line 3 to 10). Lines from 7 to 9 are enhancements that will be discussed in Sec. 4.3. To compute $\lambda_1(e), \lambda_2(e)$, we firstly select a point $m \in e$ (line 4) then evaluate the summation of $\mathcal{F}(\cdot)(m)$ according to Eq. (12) and (11) (line 5 to 6). Here m can be chosen arbitrarily (Lemma 1), in our case, it is the middle point of e.

Procedure $\mathcal{F}(\partial R)(\vec{p})$ (Alg 2, line 12 to 20) computes the ISF value at $\vec{p}$ of a regular set R according to Def 4. Line 12 to 20 can be viewed as an adaption of the well-known *Ray Crossing* algorithm [19], which tests whether a point is in a polygon. The main difference is that, we compute intersections between a ray $\vec{qp}$ and a waveseg $\tilde{l} \in \partial R$ (line 15), where $\tilde{l}$ can be either a circular arc or a segment (Remark 2). For each intersection point v, if $|vp| = 0$, we can confirm $\vec{p}$ is on boundary and directly return (line 17). Otherwise, we use the dot product between $\vec{qp}$ and $n^l_{\curvearrowleft}(\tilde{v})$ to determine whether $\vec{qp}$ is entering the interior (or exterior), and we then increase (or decrease) the counter accordingly (line 18). Specially, if $\vec{qp} \cdot n^l_{\curvearrowleft}(\tilde{v}) = 0$, it implies that $\vec{qp}$ is tangent to a circular waveseg or collinear with a segment waveseg. Both cases should not affect the

Algorithm 3 Spread λ_1 and λ_2 to Other Edges

procedure SPREAD($e : (u, v), \lambda_1, \lambda_2, E_{done}, G^w, \mathbb{P}_n, \mathbb{O}_n$)
  $E_{done} \leftarrow E_{done} \cup \{e\}$ ▷ Record e has been done
  ▷ All edges in G^w at v are arranged by polar angle in CCW
  $nxt \leftarrow$ NEXTADJ(v, e) ▷ Pick the edge next to e
  if $nxt \in E_{done}$ **then** ▷ Each edge only be evaluated once.
    return E_{done}
  $p_e, p_n := p_e \in e, p_n \in nxt$ ▷ Pick a point on e and n.
  $\Delta\lambda_1(e \to nxt) \leftarrow 0$
  $\Delta\lambda_2(e \to nxt) \leftarrow 0$
  if $R(nxt) \in \mathbb{P}_n$ **then**
    $\Delta\lambda_1(e \to nxt) \leftarrow \int_{p_e}^{p_n} \nabla\mathcal{F}(\partial P)(x) \cdot dx$ ▷ Integration according to Eq. (14)
  else
    $\Delta\lambda_2(e \to nxt) \leftarrow \int_{p_e}^{p_n} \nabla\mathcal{F}(\partial P)(x) \cdot dx$
  $\lambda_1(nxt) \leftarrow \lambda_1(e) + \Delta\lambda(e \to nxt)$
  $\lambda_2(nxt) \leftarrow \lambda_2(e) + \Delta\lambda(e \to nxt)$
  $E_{done} \leftarrow E_{done} \cup$ SPREAD($nxt, \lambda_1, \lambda_2, E_{done}, G^w, \mathbb{P}_n, \mathbb{O}_n$) ▷ Spread recursively
  return E_{done}

counter. Note that lines 15 to 18 might be affected by numeric errors, leading to an incorrect ISF value, which may alter the topology of the reachable region. Our implementation mitigates this by using an ε-distance test for point coincidence and restarting an iteration from a different starting edge whenever unclosed boundaries are detected.

4.3 Improvement Through Gradient Integration

Computing $\lambda_1(e), \lambda_2(e)$ for all $e \in E$ can be expensive, we now present a method to speed up this procedure. Since $\mathcal{F}(\partial R)(\cdot)$ is defined in a continuous space, we can construct its gradient field, $\nabla\mathcal{F}(\partial R)$. This enables us to define $\mathcal{F}(\partial R)(\vec{p})$ as a result of gradient integration. Specifically, consider a regular set R, a waveseg $\tilde{l} \in \partial R$, and any point $\vec{p} \in \tilde{l}$. Let $\vec{q}_l, \vec{q}_r \in B_\epsilon(\vec{p})$ denote nearby points of $\vec{p}$ left and right to $\tilde{l}$ respectively. The gradient field can be constructed as follows:

$$\begin{cases} \nabla\mathcal{F}(\partial R)(\vec{p}) & = 0, \quad \forall \vec{p} \notin \partial R \\ \int_{\vec{p}}^{\vec{q}} \nabla\mathcal{F}(\partial R)(\vec{r}) \cdot d\vec{r} & = 0, \quad \forall \vec{q} \in \tilde{l}^o \\ \int_{\vec{q}_r}^{\vec{p}} \nabla\mathcal{F}(\partial R)(\vec{r}) \cdot d\vec{r} & = 1 \\ \int_{\vec{q}_l}^{\vec{p}} \nabla\mathcal{F}(\partial R)(\vec{r}) \cdot d\vec{r} & = -1 \end{cases} \tag{14}$$

Eq. (14) indicates that when a point $\vec{q}$ is not on boundary, the gradient of $\vec{q}$ is $\vec{0}$. For any $\vec{q}, \vec{p} \in \tilde{l}$, the gradient integration result from $\vec{q}$ to $\vec{p}$ is always 0. If a path enters the boundary from the exterior (or interior), the gradient integration increases by 1 (or decreases by 1), and if a path exits the boundary to the exterior (or interior), the gradient integration decreases by 1 (or increases by 1).

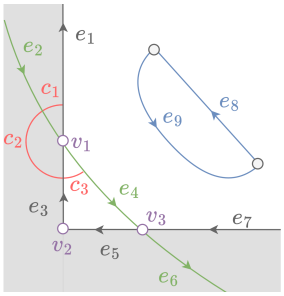


Fig. 6: A subgraph of $G^w(\mathbb{P}_n, \mathbb{O}_n)$ is shown, where shaded area represents a non-traversable region due to obstacles. The dark edges $\{e_1, e_3, e_5, e_7\}$ are on the boundary of obstacles from $\mathbb{O}_n$; green and blue edges are on the boundary of regular set from $\mathbb{P}_n$. Assume we have $\lambda_1(e_1) = 2, \lambda_2(e_1) = 1$. By using *spread*, and following the curves c_1, c_2, c_3 (red), we can compute $\lambda_1(\cdot), \lambda_2(\cdot)$ for other edges that are in the same connected component as e_1.

The concept of gradient integration allows us to compute the ISF value at $\vec{p}$ from any point whose ISF value is known, rather than always from a far away point $\vec{q}$ (Alg. 2, line 12). Specially, we have

$$\mathcal{F}(\partial R)(\vec{p}) = \mathcal{F}(\partial R)(\vec{q}) + \int_{\vec{q}}^{\vec{p}} \nabla \mathcal{F}(\partial R)(x) \cdot dx \tag{15}$$

This means that, once the values $\lambda_1(e), \lambda_2(e)$ have been evaluated (Alg. 2 line 5,6), we can use them to compute λ_1, λ_2 of all adjacent edges on G^w without expensive computation on line 5 and 6. We refer to this procedure as *spread* (Alg. 3). Exp. 4 illustrates this procedure.

Example 4. Consider a subgraph of WG, $G^w(\mathbb{P}_n, \mathbb{O}_n)$ as shown in Fig. 6. The dark edges $\{e_1, e_3, e_5, e_7\}$ are on wavesegs from $\mathbb{O}_n$, green and blue edges are on wavesegs from $\mathbb{P}_n$. Assume we have $\lambda_1(e_1) = 2, \lambda_2(e_1) = 1$, and we begin the spreading from e_1. First, we sort all edges connected to v_1 in a counter-clockwise (CCW) direction, resulting in the edge list $[e_1, e_2, e_3, e_4]$. Next, we integrate along the curve c_1 to compute $\lambda_1(e_2)$ and $\lambda_2(e_2)$. Recall that $\mathbb{P}_n, \mathbb{O}_n$ only contribute to λ_1, λ_2 respectively (Eq. (11), (12)). For $\lambda_1(e_2)$, we are integrating from the interior of $R(e_2) \in \mathbb{P}_n$ to the boundary of $R(e_2)$, so $\lambda_1(e_2) = \lambda_1(e_1) - 1 = 1$; for $\lambda_2(e_2)$, we are integrating from the boundary of $R(e_1) \in \mathbb{O}_n$ to the interior of $R(e_1)$, so $\lambda_2(e_2) = \lambda_2(e_1) + 1 = 2$. The same procedure applies along c_2, c_3 to compute values for e_3, e_4 and all other edges in the same connected component. In practice, each edge is evaluated only once. Blue edges e_8, e_9 cannot be spread because they are in a different connected component from e_1.

Spread (Alg. 3) explores G^w in a depth-first manner. For an edge $e : u \to v$ who has been evaluated, we sort all adjacent edges of e at v by polar angle in CCW, and visit the next nxt immediately after e (Line 4). This guarantees that there is no other wavesegs between e and nxt, allowing us to directly apply Eq. (14) (Line 11). After computing the $\lambda_1(nxt), \lambda_2(nxt)$, we visit adjacent edges of nxt (Line 16) recursively. The recursion guarantees that each edge is visited only once (Line 5), ensuring that it terminates in a finite number of steps.

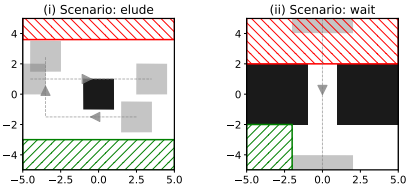


Fig. 7: Test scenarios: (a)*elude* and (b)*wait*. Black polygons are static obstacles. Gray polygons indicate the start and end positions of dynamic obstacles, with their trajectories indicated by gray dashed lines and arrows. In each scenario, the start and goal locations are randomly sampled from the green and red regions, respectively.

4.4 Discussion

Irregular Sets in R_n. As discussed in Remark 3, the reachable point set R_n may contains irregular sets, which are intentionally ignored in our method. The reason is that, the operation $R_{n-1} \oplus B_\Delta$ in propagation (Eq. (1)) makes all irregular sets to regular sets. Hence, when we build WG and run procedures in Alg. 2, the theoretical foundations (Lemma 1 and 2) still hold.

Duplicated Edges. When a point $\vec{p}$ lies on a duplicated edge of the WG (i.e., shared by multiple wavesegs), the formula in Lemma 2 does not directly apply. However, in most cases, such edges can still be treated as non-duplicated: if the edge lies in the interior of X_P (or X_O), it does not affect the final boundary and can be safely ignored. The remaining cases reduce to a base scenario involving exactly two overlapping regular sets, where the ISF value is directly determined by the relative orientation of the two wavesegs.

Optimality. The reachable set R_n at each iteration (Eq. 1) is complete. Because any point $p \notin R_n$ is either unreachable due to speed limit (i.e., not in $U_n = R_{n-1} \oplus B_\Delta$) or blocked by obstacles (i.e., $U_n \cap \neg O_n$). Thus, WAP never miss any feasible collision-free path, and WAP terminate at the first iteration that R_n covers the goal, ensuring the earliest arrival time.

Runtime Complexity. Let N_i denote the number of wavesegs at the i-th iteration. The complexity of a single iteration at t_i is $O(N_i^2 \cdot log(N_i^2))$, where N_i^2 arises from the number of edges in WG, and $log(N_i^2)$ is due to computing the intersections between wavesegs using a K-D tree [2]. Existing work [7] suggests that, in the worst case, n wavesegs can generate $O(n^2)$ new wavesegs in a single propagation when the time dimension is continuous. As time increases, the number of wavesegs can grow exponentially in the worst case, and the overall runtime complexity of our method is non-polynomial. A more precise complexity bound that takes into account discretized time could be an interesting direction for future work.

5 Experimental Results

We compare our WAP with MIP methods CP-MILP [21] and MICP [26]. All code is implemented in `C++` and compiled by `g++-11` with the `-O2` flag. All experiments run on an Intel® Core™ i7-14700KF CPU. The code and datasets will be publicly available after acceptance.

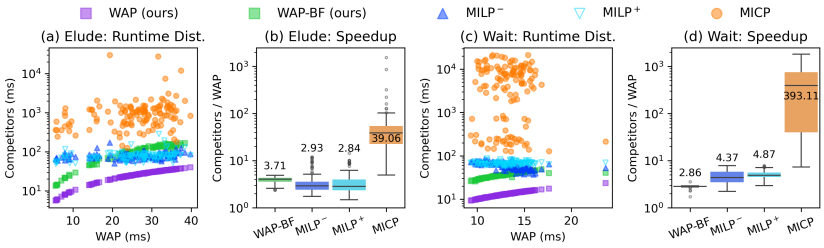


Fig. 8: Runtime comparison. (a) and (c) show the runtime of all instances in the scenarios *elude* and *wait*. Each point (x, y) represents the result of an instance, where x is the runtime of WAP, and y represents the runtime of the corresponding method; smaller values are better. (b) and (d) show the speedup ratio of WAP compared to competitors; a value greater than 1 indicates that WAP is faster.

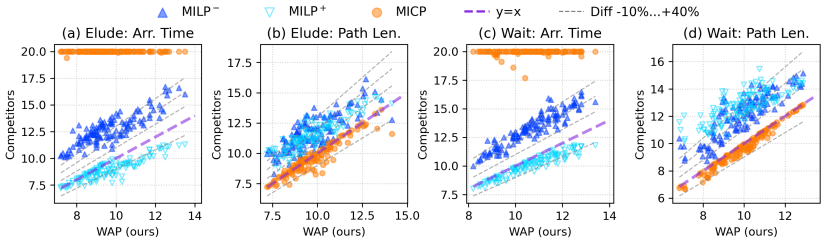


Fig. 9: Solution quality comparison. (a) and (c) show the arrival times (Arr. Time). (b) and (d) show the path lengths (Path Len.). The purple dashed line, $y = x$, represents the path quality of WAP. A point of a competitor above this line indicates that the competitor has worse quality than WAP in the corresponding instance.

Baselines Both CP-MILP [21] and MICP [26] work in continuous space under discretized time and max speed constraint. Additionally, they have a maximum time step constraint, T_{max}, and only consider solutions with arrival times before T_{max}. In this experiment, we set the $t_\Delta = 0.1$, $v_{max} = 1$ and $T_{max} = 20$ (i.e., 200 time steps). CP-MILP minimizes arrival time but uses axis-aligned speed bounds, so the per-step reachable set is a square rather than B_Δ. We evaluate it in two configurations: MILP$^-$ ($v_{max,x} = v_{max,y} = \frac{1}{\sqrt{2}}$, total max speed 1) and MILP$^+$ ($v_{max,x} = v_{max,y} = 1$, total max speed $\sqrt{2}$). MICP uses the same reachable set as WAP but minimizes travel distance; we use it as a path length reference with a 5% optimality gap. We also compare WAP with the brute-force version, i.e., without *spread*, referred to as *WAP-BF*.

Dataset We test on two representative scenarios *elude* and *wait* (Fig. 7). Each scenario includes 128 instances. The *elude* shows a situation with a static obstacle in the middle and two dynamic obstacles moving in workspace. The *wait* shows a situation where a dynamic obstacle moves across the corridor, resulting in the robot waiting for the dynamic obstacle to pass. WAP-BF has a 68% success

rate in *elude* and 96% in *wait*, due to numerical errors: it invokes PointInRS approximately two orders of magnitude more frequently than WAP, incurring a higher probability of numerical failure per iteration. WAP requires far fewer ray-casting calls per component because *spread* derives most ISF values from adjacent edges directly. All other baselines also achieve 100% success rate. We exclude failed instances from WAP-BF results below.

Result Fig. 8 shows the runtime comparison. We can see that, in both scenarios, WAP is significantly faster than the others. Specifically, consider the median of speedup ratio compared to MILP$^-$ (Fig. 8(b,d)), WAP is 2.93× faster in *elude* and 4.37× faster in *wait*. We can also see that *spread* in WAP achieves a speed-up of 3.71× in the *elude* scenario and 2.86× in the *wait* scenario compared to brute force. Fig. 9 compares the solution quality. We can see that WAP has a smaller arrival time compared to MILP$^-$ in both scenarios, as WAP guarantees optimality in arrival time. MILP$^+$ has $\approx 10\%$ smaller arrival time (Fig. 9(a,c)) because it is configured with a larger maximum speed. The arrival times of MICP are mostly around 20 (T_{max}) because MICP only optimizes the path length. Fig. 9(b,d) show that, the path length of WAP is within $\pm 10\%$ to *MICP*, indicating that WAP also achieves near-optimal path length.

6 Conclusion and Future Work

This work considers Motion Planning Among Dynamic Obstacles (MPDO) in continuous space and aims to minimize the arrival time. We propose an arithmetic-based wavefront propagation method, WAP, to solve the problem in discretized time steps. Experiment shows that, our method can be 2.93 to 4.37 times faster than baselines and achieves a near-optimal path length.

Limitations and Future Work There are some limitations in our current method and can be addressed in future work. First, WAP may never terminate if the instance is unsolvable. This issue can be resolved by conducting connectivity analysis in each iteration. Second, Alg. 2 may suffer from numerical errors, which impact the result of reachable set R_n. Future work includes developing methods to identify and repair numerical errors during planning.

Acknowledgements

This work was supported by the Natural Science Foundation of China under Grant 62403313, and the Natural Science Foundation of Shanghai under Grant 24ZR1435900.

References

1. Afonso, R.J., Maximo, M.R., Galvao, R.K.: Task allocation and trajectory planning for multiple agents in the presence of obstacle and connectivity constraints with mixed-integer linear programming. International Journal of Robust and Nonlinear Control **30**(14), 5464–5491 (2020)
2. Bentley, J.L.: K-d trees for semidynamic point sets. In: Seidel, R. (ed.) Proceedings of the Sixth Annual Symposium on Computational Geometry, Berkeley, CA, USA, June 6-8, 1990. pp. 187–197. ACM (1990)
3. Born, M., Wolf, E.: Principles of optics: electromagnetic theory of propagation, interference and diffraction of light. Elsevier (2013)
4. Canny, J., Reif, J.: New lower bound techniques for robot motion planning problems. In: 28th Annual Symposium on Foundations of Computer Science (sfcs 1987). pp. 49–60. IEEE (1987)
5. Choset, H., Lynch, K.M., Hutchinson, S., Kantor, G.A., Burgard, W.: Principles of robot motion: theory, algorithms, and implementations. MIT press (2005)
6. Frazzoli, E., Dahleh, M.A., Feron, E.: Real-time motion planning for agile autonomous vehicles. Journal of guidance, control, and dynamics **25**(1), 116–129 (2002)
7. Fujimura, K.: Motion planning amid transient obstacles. The International journal of robotics research **13**(5), 395–407 (1994)
8. Fujimura, K., Samet, H.: Planning a time-minimal motion among moving obstacles. Algorithmica **10**(1), 41–63 (1993)
9. Hales, T.C.: The jordan curve theorem, formally and informally. The American Mathematical Monthly **114**(10), 882–894 (2007)
10. Halperin, D., Salzman, O., Sharir, M.: Algorithmic motion planning. In: Handbook of Discrete and Computational Geometry, pp. 1311–1342. Chapman and Hall/CRC (2017)
11. Hershberger, J., Suri, S.: An optimal algorithm for euclidean shortest paths in the plane. SIAM Journal on Computing **28**(6), 2215–2256 (1999)
12. Hsu, D., Kindel, R., Latombe, J.C., Rock, S.: Randomized kinodynamic motion planning with moving obstacles. The International Journal of Robotics Research **21**(3), 233–255 (2002)
13. Kant, K., Zucker, S.W.: Toward efficient trajectory planning: The path-velocity decomposition. The international journal of robotics research **5**(3), 72–89 (1986)
14. Latombe, J.C.: Robot motion planning, vol. 124. Springer Science & Business Media (2012)
15. LaValle, S.M.: Planning algorithms. Cambridge university press (2006)
16. Maheshwari, A., Nouri, A., Sack, J.R.: Shortest paths among transient obstacles. Journal of Combinatorial Optimization **43**(5), 1036–1074 (2022)
17. Mitchell, J.S.: An optimal algorithm for shortest rectilinear paths among obstacles in the plane. In: Abstracts of the First Canadian Conference on Computational Geometry. vol. 22 (1989)
18. Mitchell, J.S.: Shortest paths among obstacles in the plane. In: Proceedings of the ninth annual symposium on Computational geometry. pp. 308–317 (1993)
19. o'Rourke, J.: Computational geometry in C. Cambridge university press (1998)
20. Phillips, M., Likhachev, M.: Sipp: Safe interval path planning for dynamic environments. In: 2011 IEEE International Conference on Robotics and Automation. pp. 5628–5635. IEEE (2011)

21. Ren, Z., Philip, A.G., Zhao, S., Rathinam, S., Choset, H.: Cp-milp: Mixed integer linear programming for multi-agent motion planning with linear dynamics. IEEE Robotics and Automation Letters **10**(12), 12573–12579 (2025)
22. Ren, Z., Rathinam, S., Choset, H.: A lower bounding framework for motion planning amid dynamic obstacles in 2d. In: Algorithmic Foundations of Robotics XV. pp. 540–556. Springer International Publishing, Cham (Dec 2022)
23. Ren, Z., Rathinam, S., Likhachev, M., Choset, H.: Multi-objective safe-interval path planning with dynamic obstacles. IEEE Robotics and Automation Letters **7**(3), 8154–8161 (2022)
24. Richards, A., How, J.P.: Robust variable horizon model predictive control for vehicle maneuvering. International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal **16**(7), 333–351 (2006)
25. Steen, L.A., Seebach, J.A., Steen, L.A.: Counterexamples in topology, vol. 18. Springer (1978)
26. Zhao, S., Liu, Y., Choset, H., Ren, Z.: Mixed integer conic programming for multi-agent motion planning in continuous space. In: 2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 10540–10547 (2025)
27. Zhao, S., Philip, A.G., Rathinam, S., Choset, H., Ren, Z.: Cb-gcs: Conflict-based search on the graph of convex sets for multi-agent motion planning. In: 2025 IEEE 21st International Conference on Automation Science and Engineering (CASE). pp. 2208–2214 (2025)