Combining Reinforcement Learning with Arc-search Interior-Point Method for Path Planning*

Yaguang Yang†
Hampton University, Hampton, VA 23669, USA

Qiang Le‡
Hampton University, Hampton, VA 23669, USA

Isaac E. Weintraub§
Air Force Research Laboratory, Wright-Patterson AFB, OH 45433, USA.

Path planning in environments containing obstacles has numerous practical applications. The problem is challenging because it is inherently nonlinear and nonconvex. Consequently, a variety of techniques have been developed to address this problem, among which machine learning and optimal control (or optimization) have emerged as two prominent approaches. In general, machine learning methods do not require a high-fidelity model, and a trained agent can often generate a feasible path in real time. However, the resulting path is not necessarily optimal with respect to performance objectives such as minimizing path length or travel time. In contrast, optimal control and optimization methods typically rely on high-fidelity models and often require computational effort that may not satisfy real-time constraints. Nevertheless, these methods are more likely to produce optimal or near-optimal solutions. To overcome the limitations of each approach while exploiting their respective strengths, this paper proposes a framework that combines reinforcement learning with an arc-search interior-point method for path planning. Numerical simulations demonstrate that the proposed approach effectively integrates the real-time decision-making capability of reinforcement learning with the optimization performance of the arc-search interior-point method, resulting in improved path-planning performance.

Keywords: Path planning, obstacle avoidance, nonlinear optimization, arc-search.

I. Introduction

Path planning is an active area of research because of its wide range of applications [1, 31]. In real-world scenarios, path-planning problems are often subject to restricted zones or obstacles [1, 16, 27]. Consequently, these problems are generally not only nonlinear but also nonconvex, making them intrinsically more challenging than linear or convex optimization problems. Therefore, a variety of methods have been developed to address such problems [31].

Among the available approaches, two classes of methods are particularly attractive because of their respective advantages. The first class is based on optimal control and optimization techniques, including nonlinear optimal control [21], adaptive control [26], model predictive control [9], and pseudospectral methods [15]. The second class is based on machine learning techniques, particularly reinforcement learning methods such as Q-learning [5], Deep Q Networks (DQN) [12], and Deep Deterministic Policy Gradient (DDPG) [10].

Optimal control methods typically rely on high-fidelity system models and, when successful, can produce optimal solutions. However, the computational effort required to solve the resulting optimization problems is often substantial, making these methods less suitable for real-time applications. In contrast, reinforcement learning methods generally do not require high-fidelity models. Although significant computational resources may be needed during the training phase, a trained agent can generate feasible paths much faster than optimization-based methods, making reinforcement learning particularly attractive for real-time applications [16]. Nevertheless, reinforcement learning methods typically produce

†Department of Electrical and Computer Engineering Hampton University, Hampton, VA

‡Department of Electrical and Computer Engineering Hampton University, Hampton, VA

§Control Science Center, Air Force Research Laboratory, AIAA Associate Fellow. This paper is based on work performed at the Air Force Research Laboratory (AFRL) Control Science Center and is supported by AFOSR LRIR #24RQCOR002 (funded by Dr. Frederick Leve). DISTRIBUTION STATEMENT A. Approved for public release: distribution is unlimited AFRL-2026-1808 Cleared 13 APR 2026.

feasible rather than optimal solutions. Furthermore, there may exist initial conditions for which a trained agent is unable to generate a feasible path [16].

It is worthwhile to point out that the efficiency and effectiveness of optimal-control and optimization algorithms depend heavily on the selection of an initial point*. In many cases, finding a feasible initial point may itself be a challenging task. However, if a feasible path† generated by a machine-learning method is used as the initial point of an optimization algorithm, the efficiency and effectiveness of the optimization process can be significantly improved. Moreover, the quality of the path obtained by reinforcement learning may be enhanced because a feasible path can potentially be refined into an optimal path through subsequent optimization. More importantly, the application of an arc-search interior-point method (IPM) may enlarge both the feasible set and the optimal set. For applications with stringent real-time requirements, feasible or optimal paths corresponding to all relevant starting points may be computed offline and stored in a database. During real-time operation, the appropriate optimized path can then be retrieved directly once a starting point is specified.

Motivated by these considerations, this paper proposes a framework that combines machine learning with optimization techniques for path planning in the presence of obstacles. First, the problem is formulated as a nonlinear optimization problem tailored to the scenario under consideration. Although the present work considers only basic engagement zones (BEZ) [24], the proposed framework can be readily extended to more sophisticated weapon engagement zone (WEZ) models [8].

Three optimization algorithms are employed to solve the resulting nonlinear optimization problem: the MATLAB implementation of the interior-point method (MATLAB-IPM)‡, the MATLAB implementation of sequential quadratic programming (MATLAB-SQP)§, and the recently developed arc-search IPM algorithm proposed in [30].

Feasible paths for two path-planning problems, one involving a single obstacle and the other involving three obstacles, were previously obtained using DDPG in [16]. These paths are used as initial paths for MATLAB-IPM, MATLAB-SQP, and the arc-search IPM (ASIPM). When an initial path generated by DDPG enters a restricted region and therefore fails to satisfy the interior-point requirement, a simple heuristic is employed to construct an interior feasible path. Using the same set of initial paths, feasible-set maps and optimal-set maps are generated for all three optimization algorithms. The resulting comparisons demonstrate the advantages of combining DDPG with the arc-search IPM algorithm relative to the MATLAB-IPM and MATLAB-SQP approaches.

The remainder of the paper is organized as follows. Section II presents the mathematical formulation of the path-planning problem. Section III briefly reviews the arc-search IPM algorithm. Section IV discusses the implementation details and numerical results. Finally, Section V summarizes the main conclusions of the paper.

II. The problem formulation

Throughout the remainder of this paper, bold uppercase letters denote matrices, bold lowercase letters denote vectors, and ordinary letters denote scalars. For example, $\mathbf{X}$ denotes a matrix, $\mathbf{x}$ denotes a vector, and x_k denotes the k-th element of $\mathbf{x}$. To simplify the notation, the column vector $[x_1, \ldots, x_n]^{\mathrm{T}}$ is written as $(x_1, \ldots, x_n)$, and the stacked vector $[\mathbf{x}^{\mathrm{T}}, \mathbf{y}^{\mathrm{T}}]^{\mathrm{T}}$ is written as $(\mathbf{x}, \mathbf{y})$. Finally, given a vector $\mathbf{z}$, $\mathbf{Z}$ denotes a diagonal matrix whose diagonal entries are the components of $\mathbf{z}$.

Let $(\bar{x}_k, \bar{y}_k)$ denote the position of a moving object (hereafter referred to as a vehicle or an agent) at the discrete time instant t_k. Without loss of generality, let $(\bar{x}_0, \bar{y}_0)$ denote the initial position of the vehicle and $(\bar{x}_f, \bar{y}_f)$ denote its desired destination. Both positions are assumed to be specified prior to the path-planning process. The path from the initial position $(\bar{x}_0, \bar{y}_0)$ to the destination $(\bar{x}_f, \bar{y}_f)$ is divided into f segments of equal length. Let θ_k, $k = 0, \ldots, f-1$, denote the vehicle heading at the beginning of the k-th segment, and let r denote the length of each segment. The objective of the path-planning problem is to minimize the total path length while avoiding obstacles. Since the straight-line

*Throughout the remainder of this paper, we distinguish between a *starting point* of a path and an *initial point* of a nonlinear optimization algorithm. The former refers to the initial location of a path, whereas the latter refers to the initial guess for the optimization variables. Given a starting point and an initial point, a candidate path can be constructed.

†Throughout this paper, a path is said to be *feasible* if it does not enter any restricted region and reaches the destination within a prescribed tolerance. A path is said to be *optimal* if it is the shortest feasible path and reaches the destination exactly. The *feasible set* is defined as the set of starting points for which a trained agent or a given optimization algorithm is able to generate a feasible path, while the *optimal set* is defined as the set of starting points for which the trained agent or the algorithm is able to generate an optimal path.

‡MATLAB's `fmincon` implements an interior-point algorithm based on the barrier methods developed in [2, 3], together with line-search and trust-region techniques described in [25] and documented in [17].

§MATLAB's `fmincon` implements a sequential quadratic programming algorithm based on [13, 22], incorporating a presolver [7], a quadratic programming solver [11], step-size strategies [13, 19], and additional heuristic enhancements [4, 23], as documented in [17].

distance between the initial position and the destination is $\sqrt{(\bar{x}_f - \bar{x}_0)^2 + (\bar{y}_f - \bar{y}_0)^2}$, the total path length must satisfy $f \cdot r \geq \sqrt{(\bar{x}_f - \bar{x}_0)^2 + (\bar{y}_f - \bar{y}_0)^2}$. Consequently, $r \geq \frac{\sqrt{(\bar{x}_f - \bar{x}_0)^2 + (\bar{y}_f - \bar{y}_0)^2}}{f}$.

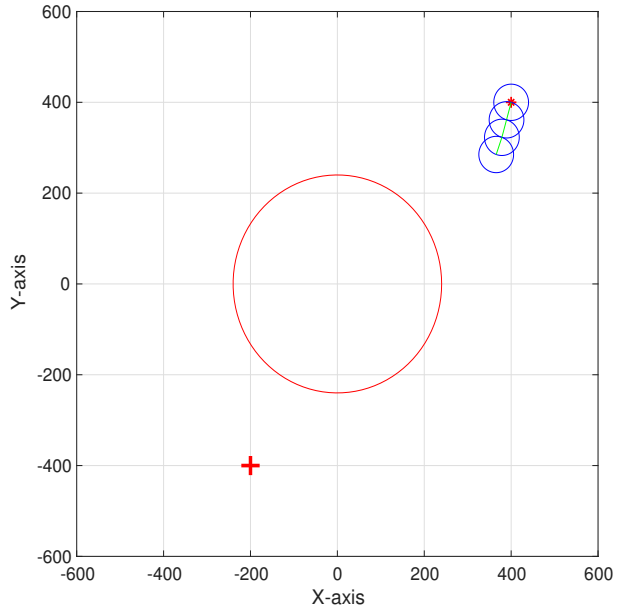

Fig. 1 Path trajectory design.

Referring to Figure 1, the red '*' denotes the starting point and the red '+' denotes the destination. The large red circle, centered at (a, b) with radius R, represents the obstacle (or basic engagement zone [27]) that the vehicle is not allowed to enter[¶]. Starting from $(\bar{x}_0, \bar{y}_0)$, the vehicle travels a distance r in the direction specified by theheading angle of θ_0 and arrive at $(\bar{x}_1, \bar{y}_1)$. It then travels another distance r in the direction specified by θ_1 and arrive at $(\bar{x}_2, \bar{y}_2)$, and the process continues in the same manner until $(\bar{x}_f, \bar{y}_f)$ is reached. The small blue circles are centered at the starting point of every $(\bar{x}_i, \bar{y}_i)$, $i = 0, 1, 2, \cdots$, and each has radius r. Since the vehicle moves a fixed distance r in every step, the position of the vehicle can be determined recursively from the heading angles. Given the initial position $(\bar{x}_0, \bar{y}_0)$, the following equations hold:

$$\bar{x}_1 = \bar{x}_0 + r\cos(\theta_0), \quad \bar{y}_1 = \bar{y}_0 + r\sin(\theta_0) \tag{1a}$$

$$\bar{x}_k = \bar{x}_{k-1} + r\cos(\theta_{k-1}) = \bar{x}_0 + r\sum_{i=0}^{k-1}\cos(\theta_i), \quad k = 1, \ldots f-1, \tag{1b}$$

$$\bar{y}_k = \bar{y}_{k-1} + r\sin(\theta_{k-1}) = \bar{y}_0 + r\sum_{i=0}^{k-1}\sin(\theta_i), \quad k = 1, \ldots f-1, \tag{1c}$$

$$\bar{x}_f = \bar{x}_0 + r\sum_{i=0}^{f-1}\cos(\theta_i), \quad \bar{y}_f = \bar{y}_0 + r\sum_{i=0}^{f-1}\sin(\theta_i). \tag{1d}$$

Since the vehicle is not allowed to enter the obstacle region $(\bar{x}_k - a)^2 + (\bar{y}_k - b)^2 \leq R^2$, and since both the initial position and the destination are assumed to lie outside the obstacle, every point along a feasible path must satisfy

$$\left[\bar{x}_0 + r\sum_{i=0}^{k-1}\cos(\theta_i) - a\right]^2 + \left[\bar{y}_0 + r\sum_{i=0}^{k-1}\sin(\theta_i) - b\right]^2 \geq R^2, \quad k = 1, \ldots, f-1. \tag{2}$$

[¶]This simple obstacle model covers several practical scenarios, including certain pursuit–evasion problems described in [24]. Furthermore, the methodology developed in this paper can be readily extended to more general constraint regions, such as those considered in [8, 27].

To restrict the vehicle's operating region, an artificial boundary is imposed.

$$\left[\bar{x}_0 + r\sum_{i=0}^{k-1}\cos(\theta_i) - a\right]^2 + \left[\bar{y}_0 + r\sum_{i=0}^{k-1}\sin(\theta_i) - b\right]^2 \le D^2, \quad k = 1, \ldots, f-1. \tag{3}$$

To avoid abrupt changes in direction, the difference in heading between two consecutive path segments is constrained by $|\theta_k - \theta_{k-1}| \le 0.5$ radians for all $k \in \{1, \ldots, f-1\}$.

Based on the above discussion, the path-planning problem considering a single circular obstacle (BEZ) is formulated as the following optimization problem.

$$\min \; r \tag{4a}$$

$$s.t. \; \bar{x}_f = \bar{x}_0 + r\sum_{i=0}^{f-1}\cos(\theta_i), \tag{4b}$$

$$\bar{y}_f = \bar{y}_0 + r\sum_{i=0}^{f-1}\sin(\theta_i), \tag{4c}$$

$$\left[\bar{x}_0 + r\sum_{i=0}^{k-1}\cos(\theta_i) - a\right]^2 + \left[\bar{y}_0 + r\sum_{i=0}^{k-1}\sin(\theta_i) - b\right]^2 - R^2 \ge 0, \quad k = 1, \ldots, f-1, \tag{4d}$$

$$D^2 - \left[\bar{x}_0 + r\sum_{i=0}^{k-1}\cos(\theta_i) - a\right]^2 - \left[\bar{y}_0 + r\sum_{i=0}^{k-1}\sin(\theta_i) - b\right]^2 \ge 0, \quad k = 1, \ldots, f-1, \tag{4e}$$

$$\theta_k - \theta_{k-1} + 0.5 \ge 0, \quad k = 1, \ldots, f-1, \tag{4f}$$

$$\theta_{k-1} - \theta_k + 0.5 \ge 0, \quad k = 1, \ldots, f-1, \tag{4g}$$

$$r \ge \frac{\sqrt{(\bar{x}_f - \bar{x}_0)^2 + (\bar{y}_f - \bar{y}_0)^2}}{f}, \tag{4h}$$

The problem consists of 2 equality constraints, given by (4b) and (4c); $2f-2$ nonlinear inequality constraints, where (4d) represents $f-1$ nonlinear inequality constraints, (4e) represents $f-1$ nonlinear inequality constraints; $2f-2$ linear inequality constraints defined by (4f) and (4g); and 1 boundary constraint defined by (4h). There are $n = f+1$ optimization variables $\mathbf{x} = (r, \theta_0, \theta_1, \ldots, \theta_{f-1})$. For convenience, the equality constraints are grouped into a vector-valued function $\mathbf{h}(\mathbf{x}) = \mathbf{0}$ whose components are (4b) and (4c). Similarly all inequality constraints are collected into a vector-valued inequality $\mathbf{g}(\mathbf{x}) \ge \mathbf{0}$ whose components are (4d), (4e), (4f), (4g), (4h)). Therefore, Problem (4) can be expressed in a standard form:

$$\min \; x_1 \tag{5a}$$

$$s.t. \; \mathbf{h}(\mathbf{x}) = \mathbf{0}, \tag{5b}$$

$$\mathbf{g}(\mathbf{x}) \ge \mathbf{0}. \tag{5c}$$

Denote $f(\mathbf{x}) = x_1$, by introducing a relaxation vector $\mathbf{s} \ge \mathbf{0}$, Problem (5) can be rewritten as:

$$\min \; f(\mathbf{x}) \tag{6a}$$

$$s.t. \; \mathbf{h}(\mathbf{x}) = \mathbf{0}, \tag{6b}$$

$$\mathbf{g}(\mathbf{x}) + \mathbf{s} = \mathbf{0}, \;\; \mathbf{s} \ge \mathbf{0}, \tag{6c}$$

where $f(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$, $\mathbf{h}(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}^m$, and $\mathbf{g}(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}^p$. In the present problem, $n = f+1$, $m = 2$, and $p = 3f-2$. To improve computational efficiency, the inequality constraints can be further partitioned into two groups: nonlinear inequality constraints and linear inequality constraints. Since the second-order derivatives associated with the linear inequality constraints are identically zero, their evaluation can be omitted during the computation of second-order information (see (9) and (13)). This separation reduces the computational burden without affecting the optimization results.

III. Optimal path planning using arc-search IPM

We adopt an arc-search interior-point method (IPM) proposed in [30] to solve the nonlinear optimization problem (6). The Lagrangian of Problem (6) is defined as

$$L(\mathbf{x}, \mathbf{y}, \mathbf{w}, \mathbf{s}, \mathbf{z}) = f(\mathbf{x}) - \mathbf{y}^{\mathrm{T}}\mathbf{h}(\mathbf{x}) - \mathbf{w}^{\mathrm{T}}(\mathbf{g}(\mathbf{x}) - \mathbf{s}) - \mathbf{z}^{\mathrm{T}}\mathbf{s}, \tag{7}$$

where $\mathbf{y} \in \mathbb{R}^m$, $\mathbf{w} > \mathbf{0} \in \mathbb{R}_+^p$, and $\mathbf{z} > 0 \in \mathbb{R}_+^p$ are multipliers. Let $\mathbf{v} = (\mathbf{x}, \mathbf{y}, \mathbf{w}, \mathbf{s}, \mathbf{z}) \in \mathbb{R}^{n+m+3p}$ denote a stacked column vector composed of primal and slack variables, and multipliers. The gradient of the Lagrangian function with respect to $\mathbf{x}$ and $\mathbf{s}$ are given by

$$\nabla_{\mathbf{x}} L(\mathbf{v}) = \nabla f(\mathbf{x}) - \nabla \mathbf{h}(\mathbf{x})\mathbf{y} - \nabla \mathbf{g}(\mathbf{x})\mathbf{w}, \tag{8}$$

and the Hessian matrix of L is given by

$$\nabla_{\mathbf{x}}^2 L(\mathbf{v}) = \nabla_{\mathbf{x}}^2 f(\mathbf{x}) - \sum_{i=1}^{m} \nabla_{\mathbf{x}}^2 h_i(\mathbf{x}) y_i - \sum_{i=1}^{p} \nabla_{\mathbf{x}}^2 g_i(\mathbf{x}) w_i. \tag{9}$$

Most optimization algorithms find the optimal solution by searching a KKT point [20], which is equivalent to solve a system of nonlinear equations:

$$\mathbf{k}(\mathbf{v}) = \begin{bmatrix} \nabla_{\mathbf{x}} L(\mathbf{v}) \\ \mathbf{h}(\mathbf{x}) \\ \mathbf{g}(\mathbf{x}) - \mathbf{s} \\ \mathbf{w} - \mathbf{z} \\ \mathbf{Z}\mathbf{s} \end{bmatrix} = \mathbf{0}, \ (\mathbf{w}, \mathbf{s}, \mathbf{z}) \in \mathbb{R}_+^{3p}, \tag{10}$$

where $\mathbf{k} : \mathbb{R}^{n+m+3p} \to \mathbb{R}^{n+m+3p}$. A sufficient condition for KKT to hold is

$$\phi(\mathbf{v}) = \mathbf{k}(\mathbf{v})^{\mathrm{T}}\mathbf{k}(\mathbf{v}) = 0. \tag{11}$$

Taking the first-order derivative of (10) yields

$$\begin{bmatrix} \nabla_{\mathbf{x}}^2 L(\mathbf{v}^k) & -\nabla \mathbf{h}(\mathbf{x}^k) & -\nabla \mathbf{g}(\mathbf{x}^k) & \mathbf{0} & \mathbf{0} \\ \left(\nabla \mathbf{h}(\mathbf{x}^k)\right)^{\mathrm{T}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \left(\nabla \mathbf{g}(\mathbf{x}^k)\right)^{\mathrm{T}} & \mathbf{0} & \mathbf{0} & -\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & -\mathbf{I} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{Z}^k & \mathbf{S}^k \end{bmatrix} \begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{y}} \\ \dot{\mathbf{w}} \\ \dot{\mathbf{s}} \\ \dot{\mathbf{z}} \end{bmatrix} = \begin{bmatrix} \nabla_{\mathbf{x}} L(\mathbf{v}^k) \\ \mathbf{h}(\mathbf{x}^k) \\ \mathbf{g}(\mathbf{x}^k) - \mathbf{s}^k \\ \mathbf{w}^k - \mathbf{z}^k \\ \mathbf{Z}^k\mathbf{s}^k - \sigma\mu_k\mathbf{e} \end{bmatrix}. \tag{12}$$

where $\mathbf{e}$ is an all-one vector of dimension p. Taking the derivative of (12) and ignoring the third-order derivative yields

$$\begin{bmatrix} \nabla_{\mathbf{x}}^2 L(\mathbf{v}^k) & -\nabla \mathbf{h}(\mathbf{x}^k) & -\nabla \mathbf{g}(\mathbf{x}^k) & \mathbf{0} & \mathbf{0} \\ \left(\nabla \mathbf{h}(\mathbf{x}^k)\right)^{\mathrm{T}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \left(\nabla \mathbf{g}(\mathbf{x}^k)\right)^{\mathrm{T}} & \mathbf{0} & \mathbf{0} & -\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & -\mathbf{I} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{Z}^k & \mathbf{S}^k \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{x}} \\ \ddot{\mathbf{y}} \\ \ddot{\mathbf{w}} \\ \ddot{\mathbf{s}} \\ \ddot{\mathbf{z}} \end{bmatrix} = \begin{bmatrix} 2[(\nabla_{\mathbf{x}}^2 \mathbf{g}(\mathbf{x}^k))\dot{\mathbf{w}}\dot{\mathbf{x}} + (\nabla_{\mathbf{x}}^2 \mathbf{h}(\mathbf{x}^k))\dot{\mathbf{y}}\dot{\mathbf{x}}] \\ -(\nabla_{\mathbf{x}}^2 \mathbf{h}(\mathbf{x}^k))^{\mathrm{T}}\dot{\mathbf{x}}\dot{\mathbf{x}} \\ -(\nabla_{\mathbf{x}}^2 \mathbf{g}(\mathbf{x}^k))^{\mathrm{T}}\dot{\mathbf{x}}\dot{\mathbf{x}} \\ \mathbf{0} \\ -2\dot{\mathbf{Z}}\dot{\mathbf{s}} \end{bmatrix}, \tag{13}$$

where the computational formulas for the elements on the right-hand side of (13) are given in [28, Appendix 1]:

$$\nabla_{\mathbf{x}}^2 \mathbf{h}(\mathbf{x})\dot{\mathbf{y}}\dot{\mathbf{x}} = \frac{\partial\left(\frac{\partial \mathbf{h}(\mathbf{x})}{\partial \mathbf{x}}\dot{\mathbf{y}}\right)}{\partial \mathbf{x}}\dot{\mathbf{x}} = \sum_{i=1}^{m} \dot{y}_i \frac{\partial}{\partial \mathbf{x}} \begin{bmatrix} \frac{\partial h_i(\mathbf{x})}{\partial x_1} \\ \vdots \\ \frac{\partial h_i(\mathbf{x})}{\partial x_n} \end{bmatrix} \dot{\mathbf{x}} = \sum_{i=1}^{m} \dot{y}_i \left(\nabla_{\mathbf{x}}^2 h_i(\mathbf{x})\right) \dot{\mathbf{x}} \tag{14}$$

$$\nabla_{\mathbf{x}}^2 \mathbf{h}(\mathbf{x})^{\mathrm{T}}\dot{\mathbf{x}}\dot{\mathbf{x}} = \left(\frac{\partial\left(\left(\frac{\partial \mathbf{h}(\mathbf{x})}{\partial \mathbf{x}}\right)^{\mathrm{T}}\dot{\mathbf{x}}\right)}{\partial \mathbf{x}}\right)^{\mathrm{T}}\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{x}}^{\mathrm{T}}\left(\nabla_{\mathbf{x}}^2 h_1(\mathbf{x})\right)\dot{\mathbf{x}} \\ \vdots \\ \dot{\mathbf{x}}^{\mathrm{T}}\left(\nabla_{\mathbf{x}}^2 h_m(\mathbf{x})\right)\dot{\mathbf{x}} \end{bmatrix} \tag{15}$$

$$\nabla^2_{\mathbf{x}}\mathbf{g}(\mathbf{x})\dot{\mathbf{w}}\dot{\mathbf{x}} = \frac{\partial\left(\frac{\partial \mathbf{g}(\mathbf{x})}{\partial \mathbf{x}}\dot{\mathbf{w}}\right)}{\partial \mathbf{x}}\dot{\mathbf{x}} = \sum_{i=1}^{n}\dot{w}_i\frac{\partial}{\partial \mathbf{x}}\begin{bmatrix}\frac{\partial g_i(\mathbf{x})}{\partial x_1}\\ \vdots \\ \frac{\partial g_i(\mathbf{x})}{\partial x_n}\end{bmatrix}\dot{\mathbf{x}} = \sum_{i=1}^{n}\dot{w}_i\left(\nabla^2_{\mathbf{x}}g_i(\mathbf{x})\right)\dot{\mathbf{x}} \tag{16}$$

$$\nabla^2_{\mathbf{x}}\mathbf{g}(\mathbf{x})^{\mathrm{T}}\dot{\mathbf{x}}\dot{\mathbf{x}} = \left(\frac{\partial\left(\left(\frac{\partial \mathbf{g}(\mathbf{x})}{\partial \mathbf{x}}\right)^{\mathrm{T}}\dot{\mathbf{x}}\right)}{\partial \mathbf{x}}\right)^{\mathrm{T}}\dot{\mathbf{x}} = \begin{bmatrix}\dot{\mathbf{x}}^{\mathrm{T}}\left(\nabla^2_{\mathbf{x}}g_1(\mathbf{x})\right)\dot{\mathbf{x}}\\ \vdots \\ \dot{\mathbf{x}}^{\mathrm{T}}\left(\nabla^2_{\mathbf{x}}g_p(\mathbf{x})\right)\dot{\mathbf{x}}\end{bmatrix}. \tag{17}$$

An observation from the fourth component of (10), (12), and (13) leads to the following lemma.

Lemma III.1 [28] If $\mathbf{w}^0 = \mathbf{z}^0$, then $\mathbf{w}^k = \mathbf{z}^k$.

This observation is important because it reduces computational cost: it is sufficient to compute either $\mathbf{w}$ or $\mathbf{z}$ but not both. Another key idea in interior-point methods is to trace the central path [18]. However, computing the central path explicitly is generally computationally impractical. To address this difficulty, arc-search techniques have been proposed, which approximate the central path by a segment of an ellipse [29].

$$\mathcal{E} = \{\mathbf{v}(\alpha) : \mathbf{v}(\alpha) = \vec{\mathbf{a}}\cos(\alpha) + \vec{\mathbf{b}}\sin(\alpha) + \vec{\mathbf{c}}, \alpha \in [0, 2\pi]\}, \tag{18}$$

Theorem III.1 [29] Suppose that the ellipse $\mathcal{E}$ defined in (18) passes through the current iterate $\mathbf{v}^k$ at $\alpha = 0$, and its first- and second-order derivatives at $\alpha = 0$ are given by $\dot{\mathbf{v}}$ and $\ddot{\mathbf{v}}$, respectively. Then the trajectory on $\mathcal{E}$ can be expressed as $\mathbf{v}(\alpha) = (\mathbf{x}(\alpha), \mathbf{y}(\alpha), \mathbf{w}(\alpha), \mathbf{s}(\alpha), \mathbf{z}(\alpha))$, which can be calculated by

$$\mathbf{v}(\alpha) = \mathbf{v}^k - \dot{\mathbf{v}}\sin(\alpha) + \ddot{\mathbf{v}}(1 - \cos(\alpha)). \tag{19}$$

To determine the largest step size α_{w_i} that preserves the nonnegativity condition $w_i \geq 0$, a set of formulas of α_{w_i} was derived in [29]. This yields the maximum admissible step size $\tilde{\alpha}_k$ such as $\mathbf{w} \geq \mathbf{0}$, given by

$$\tilde{\alpha}_k = \min_{i\in\{1,\ldots,p\}}\left\{\min\left\{\alpha^k_{w_i}, \frac{\pi}{2}\right\}\right\}. \tag{20}$$

Next a search is performed to determine $\bar{\alpha} \in (0, \tilde{\alpha}_k]$ such that the inequality constraints $\mathbf{g}(\mathbf{x}) > 0$ are satisfied. To ensure sufficient decrease of the merit function $\phi(\mathbf{v}(\alpha))$, an additional search is carried out to find $\breve{\alpha} \in (0, \bar{\alpha}_k]$ such that for some $\delta_2 \geq 0$, the following inequality holds:

$$\breve{\alpha}_k = \max\left\{\alpha \in (0, \bar{\alpha}_k] : \phi(\mathbf{v}^k(\alpha)) < \phi(\mathbf{v}^k) - \delta_2\right\}. \tag{21}$$

Finally, to ensure convergence of the arc-search IPM algorithm, $\hat{m}_k(\alpha)$ is defined in [30] as

$$\hat{m}_k(\alpha) = \min(\mathbf{Z}^k(\alpha)\mathbf{s}^k(\alpha)) - \frac{1}{2}\frac{\mathbf{Z}^0\mathbf{s}^0}{\phi(\mathbf{v}^0)}\min(\phi(\mathbf{v}(\alpha))), \tag{22}$$

where the minimum is taken component-wise. A subsequent search is then performed to determineand $\hat{\alpha} \in (0, \bar{\alpha}_k]$ such that

$$\hat{\alpha}_k = \max\left\{\alpha \in (0, \breve{\alpha}_k] : \hat{m}_k(\alpha) \geq 0\right\}. \tag{23}$$

Therefore, the arc-search IPM algorithm can be stated as follows:

Algorithm III.1 [30]

1: Parameters: $\epsilon > 0$, $\delta_1 > 0$, $\delta_2 > 0$, $\rho \in (0, \frac{1}{2})$, and $\bar{\sigma} \in [0, \frac{1}{2})$.
2: Initial point: $\mathbf{v}^0 = (\mathbf{x}^0, \mathbf{y}^0, \mathbf{w}^0, \mathbf{s}^0, \mathbf{z}^0)$ such that $(\mathbf{w}^0, \mathbf{s}^0, \mathbf{z}^0) > \mathbf{0}$, $\mathbf{g}(\mathbf{x}^0) > 0$, and $\mathbf{w}^0 = \mathbf{z}^0$.
3: **for** k=0,1,2,... **do**
4: Calculate $\mathbf{h}(\mathbf{x}^k)$, $\mathbf{g}(\mathbf{x}^k)$, $\nabla_{\mathbf{x}}\mathbf{h}(\mathbf{x}^k)$, $\nabla_{\mathbf{x}}\mathbf{g}(\mathbf{x}^k)$, and $\nabla_{\mathbf{x}}L(\mathbf{v}^k)$. If $\phi(\mathbf{v}^k) \leq \epsilon$, stop.

Calculate $\nabla^2_{\mathbf{x}}\mathbf{h}(\mathbf{x}^k)$, $\nabla^2_{\mathbf{x}}\mathbf{g}(\mathbf{x}^k)$, and $\nabla^2_{\mathbf{x}}L(\mathbf{v}^k)$.
Select σ_k such that $\bar{\sigma} \leq \sigma_k < \min\{\frac{1}{8}, \phi(\mathbf{v}^k)p/\mu^2\}$.
Calculate $\dot{\mathbf{v}}^k$ by solving (12) at $\mathbf{v} = \mathbf{v}^k$.
Calculate the righthand side of (13) using (14), (15), (16), and (17).
Calculate $\ddot{\mathbf{v}}^k$ by solving (13) at $\mathbf{v} = \mathbf{v}^k$.
Calculate $\tilde{\alpha}_k$ using (20) and search $\bar{\alpha}_k \in (0, \tilde{\alpha}_k]$ such that $\mathbf{g}(\mathbf{x}(\bar{\alpha}_k)) > \mathbf{0}$.
Search $\check{\alpha}_k \in (0, \bar{\alpha}_k]$ such that (21) holds.
Determine $\hat{\alpha}_k > 0$ using (22) and (23).
Set $\alpha_k = \min\{\hat{\alpha}_k, \check{\alpha}_k\}$.
Update $(\mathbf{x}^{k+1}, \mathbf{w}^{k+1}) = (\mathbf{x}^k, \mathbf{w}^k) - (\dot{\mathbf{x}}^k, \dot{\mathbf{w}}^k)\sin(\alpha_k) + (\ddot{\mathbf{x}}^k, \ddot{\mathbf{w}}^k)(1 - \cos(\alpha_k))$.
Set $\mathbf{s}^{k+1} = \mathbf{g}(\mathbf{x}^{k+1}) > \mathbf{0}$ and $\mathbf{z}^{k+1} = \mathbf{w}^{k+1}$.
$k + 1 \to k$.
end for

The convergence of this algorithm is proved in [30].

IV. Implementation and testing results

First, we present implementation details of Algorithm III.1.

A. Assembling systems of equations (12) and (13)

To implement the algorithm, we need to compute $\nabla_{\mathbf{x}} h_k(\mathbf{x})$ and $\nabla^2_{\mathbf{x}} h_k(\mathbf{x})$ for $k = 1, 2$, as well as $\nabla_{\mathbf{x}} g_k(\mathbf{x})$ and $\nabla^2_{\mathbf{x}} g_k(\mathbf{x})$ for $k = 1, 2, \ldots, f-1$ in order to assemble the systems of equations (12) and (13). Taking the first- and second-order derivatives of (4b) and (4c) yields the following expressions:

$$\nabla_{\mathbf{x}} h_1(\mathbf{x}) = \left(\sum_{i=0}^{f-1} \cos(\theta_i), -r\sin(\theta_0), -r\sin(\theta_1), \ldots, -r\sin\big(\theta_{f-1}\big) \right), \tag{24}$$

$$\nabla^2_{\mathbf{x}} h_1(\mathbf{x}) = \begin{bmatrix} 0 & -\sin(\theta_0) & -\sin(\theta_1) & \ldots & \ldots & -\sin\big(\theta_{f-1}\big) \\ -\sin(\theta_0) & -r\cos(\theta_0) & 0 & \ldots & \ldots & 0 \\ -\sin(\theta_1) & 0 & -r\cos(\theta_1) & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & 0 \\ -\sin\big(\theta_{f-2}\big) & 0 & \ldots & 0 & -r\cos\big(\theta_{f-2}\big) & 0 \\ -\sin\big(\theta_{f-1}\big) & 0 & \ldots & \ldots & 0 & -r\cos\big(\theta_{f-1}\big) \end{bmatrix}, \tag{25}$$

$$\nabla_{\mathbf{x}} h_2(\mathbf{x}) = \left(\sum_{i=0}^{f-1} \sin(\theta_i), r\cos(\theta_0), r\cos(\theta_1), \ldots, r\cos\big(\theta_{f-1}\big) \right), \tag{26}$$

$$\nabla^2_{\mathbf{x}} h_2(\mathbf{x}) = \begin{bmatrix} 0 & \cos(\theta_0) & \cos(\theta_1) & \ldots & \ldots & \cos\big(\theta_{f-1}\big) \\ \cos(\theta_0) & -r\sin(\theta_0) & 0 & \ldots & \ldots & 0 \\ \cos(\theta_1) & 0 & -r\sin(\theta_1) & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & 0 \\ \cos\big(\theta_{f-2}\big) & 0 & \ldots & 0 & -r\sin\big(\theta_{f-2}\big) & 0 \\ \cos\big(\theta_{f-1}\big) & 0 & \ldots & \ldots & 0 & -r\sin\big(\theta_{f-1}\big) \end{bmatrix}, \tag{27}$$

Taking the first-order derivatives of (4d), for $k = 1, \ldots, f-1$, yields

$$\nabla_{\mathbf{x}} g_k(\mathbf{x}) = \begin{bmatrix} 2\left(r\sum_{i=0}^{k-1}\cos(\theta_i) - a\right)\left(\sum_{i=0}^{k-1}\cos(\theta_i)\right) + 2\left(r\sum_{i=0}^{k-1}\sin(\theta_i) - b\right)\left(\sum_{i=0}^{k-1}\sin(\theta_i)\right) \\ -2r\left(r\sum_{i=0}^{k-1}\cos(\theta_i) - a\right)\sin(\theta_0) + 2r\left(r\sum_{i=0}^{k-1}\sin(\theta_i) - b\right)\cos(\theta_0) \\ \vdots \\ -2r\left(r\sum_{i=0}^{k-1}\cos(\theta_i) - a\right)\sin(\theta_j) + 2r\left(r\sum_{i=0}^{k-1}\sin(\theta_i) - b\right)\cos(\theta_j) \\ \vdots \\ -2r\left(r\sum_{i=0}^{k-1}\cos(\theta_i) - a\right)\sin(\theta_{k-1}) + 2r\left(r\sum_{i=0}^{k-1}\sin(\theta_i) - b\right)\cos(\theta_{k-1}) \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \tag{28}$$

To obtain the second-order derivatives of (4d), we differentiate (28) once more. This allows the entries of the Hessian matrix to be written as follows:

$$\frac{\partial^2 g_k(\mathbf{x})}{\partial r^2} = 2\left(\sum_{i=0}^{k-1}\cos(\theta_i)\right)^2 + 2\left(\sum_{i=0}^{k-1}\sin(\theta_i)\right)^2, \tag{29}$$

$$\frac{\partial^2 g_k(\mathbf{x})}{\partial r \partial \theta_i} = -2\left[\left(2r\sum_{i=0}^{k-1}\cos(\theta_i) - a\right)\sin(\theta_i) - \left(2r\sum_{i=0}^{k-1}\sin(\theta_i) - b\right)\cos(\theta_i)\right], \quad i = 0, 1, \ldots, k-1, \tag{30}$$

$$\frac{\partial^2 g_k(\mathbf{x})}{\partial \theta_j^2} = 2r\left[r + \left(r\sum_{i=0}^{k-1}\cos(\theta_i) - a\right)\cos(\theta_j) - \left(r\sum_{i=0}^{k-1}\sin(\theta_i) - b\right)\sin(\theta_j)\right], \quad j = 0, 1, \ldots, k-1. \tag{31}$$

$$\frac{\partial^2 g_k(\mathbf{x})}{\partial \theta_i \partial \theta_j} = 2r^2\sin(\theta_i)\sin(\theta_j) + 2r^2\cos(\theta_i)\cos(\theta_j), \quad i, j = 0, 1, \ldots, k-1, \quad i \neq j. \tag{32}$$

The Hessian matrix is therefore given by

$$\frac{\partial^2 g_k(\mathbf{x})}{\partial \mathbf{x}^2} = \left[\begin{array}{cccc|c} \frac{\partial^2 g_k(\mathbf{x})}{\partial r^2} & \frac{\partial^2 g_k(\mathbf{x})}{\partial r \partial \theta_0} & \cdots & \frac{\partial^2 g_k(\mathbf{x})}{\partial r \partial \theta_{k-1}} & \\ \frac{\partial^2 g_k(\mathbf{x})}{\partial r \partial \theta_0} & \frac{\partial^2 g_k(\mathbf{x})}{\partial \theta_0^2} & \cdots & \frac{\partial^2 g_k(\mathbf{x})}{\partial \theta_0 \partial \theta_{k-1}} & \\ \vdots & \vdots & \ddots & \vdots & \mathbf{0} \\ \frac{\partial^2 g_k(\mathbf{x})}{\partial r \partial \theta_{k-1}} & \cdots & \cdots & \frac{\partial^2 g_k(\mathbf{x})}{\partial \theta_{k-1}^2} & \\ \hline \multicolumn{4}{c|}{\mathbf{0}} & \mathbf{0} \end{array}\right], \quad k = 1, \ldots, f-1. \tag{33}$$

These formulas enable the assembly of the two linear systems of equations (12) and (13). Solving these systems yields the vectors $\dot{\mathbf{v}}$ an $\ddot{\mathbf{v}}$. With these quantities, the arc-search IPM can be implemented using Theorem III.1, while ensuring the conditions $\mathbf{w}^k > \mathbf{0}$, $\mathbf{s}^k > \mathbf{0}$, $\mathbf{z}^k > \mathbf{0}$, $\mathbf{g}(\mathbf{x}^k) > \mathbf{0}$, together with (21), and (23), in order to determine the apppropriate step size and updated $\mathbf{v}^{k+1}$. This process is repeated until an optimal solution $\mathbf{x}^* = (r^*, \theta_0^*, \ldots, \theta_{f-1}^*)$ is obtained. The optimal path can then be constructed from the starting location and the computed optimal solution. For large-scale problems, explicitly using the formulas to calculate $\nabla_{\mathbf{x}} h_k(\mathbf{x})$ and $\nabla_{\mathbf{x}}^2 h_k(\mathbf{x})$ for $k = 1, 2$, and $\nabla_{\mathbf{x}} g_k(\mathbf{x})$ and $\nabla_{\mathbf{x}}^2 g_k(\mathbf{x})$ for $k = 1, 2, \ldots, f-1$ can be tedious, and manually coding the large matrices is prone to errors. Therefore, automatic differentiation [20] is also employed to compute these derivatives directly from $\mathbf{h}(\mathbf{x})$ and $\mathbf{g}(\mathbf{x})$ for the path planning problem. On the other hand, since $\nabla_{\mathbf{x}}^2 h_k(\mathbf{x})$ and $\nabla_{\mathbf{x}}^2 g_k(\mathbf{x})$ for small k are sparse, using explicit formulas given in (25), (27), and (33) instead of automatic differentiation can further reduce computational cost.

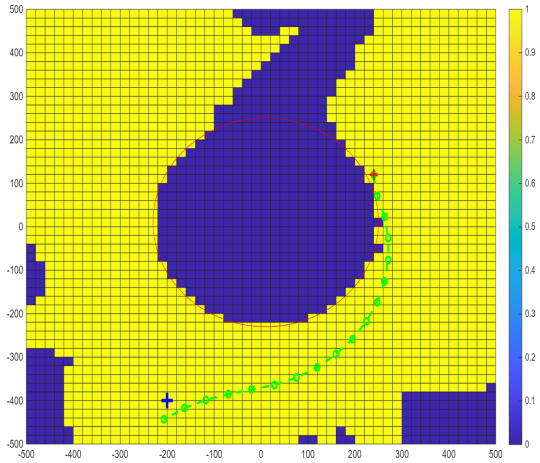

Fig. 2 Feasible set found by the trained DDPG.

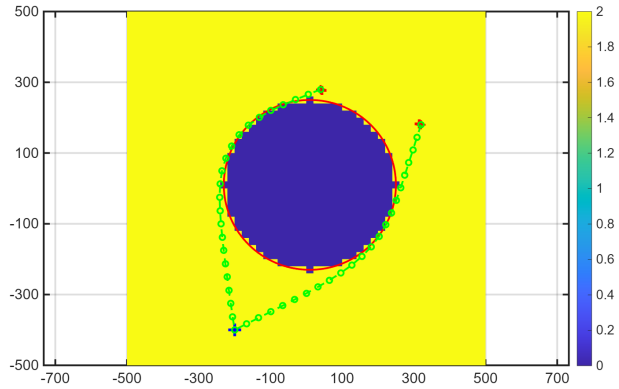

Fig. 3 Optimal set found by DDPG-arc-search IPM.

B. Path planning in the presence of single obstacle

We have shown in the previous sections that the path-planning problem can be formulated as the optimization problem (5). We propose using an arc-search interior-point method (IPM) to solve this problem. It is well known that a good initial path is important for optimization algorithms to efficiently find an optimal solution; however, there is no universal approach for generating such a path.

On the other hand, feasible paths can be generated using machine-learning methods, such as Deep Deterministic Policy Gradient (DDPG). In [16], DDPG is employed to identify a set of starting points from which the agent can safely approach the destination without entering any obstacle region. We refer to this set of starting points as the *feasible set*. A notable advantage of DDPG is that, once trained, the agent can generate a feasible path from any starting point in the feasible set to the destination in real time.

However, directly using a DDPG agent has two limitations. First, the resulting feasible path is generally not optimal. Second, the trained agent may fail to find a feasible path for starting points outside the feasible set. In this section, we combine machine-learning techniques with the arc-search IPM to exploit the strengths of both approaches. Specifically, the machine-learning agent is used to generate feasible paths, while the arc-search IPM is employed to efficiently compute optimal paths offline. For starting points for which the DDPG agent cannot generate a feasible path, the arc-search IPM is used to construct an initial path and search for an optimal solution. If an optimal solution cannot be obtained within the prescribed computational limits but a feasible solution is found, the feasible solution is also stored for subsequent real-time decision making. Consequently, the computational burden of optimization is shifted to the offline stage and does not affect real-time applications.

We define the *optimal set* as the set of starting points for which an optimization algorithm successfully computes optimal paths. More generally, we define the *feasible set* as the set of starting points for which either the DDPG agent or

the optimization algorithm can generate a feasible path to the destination. Under these definitions, the optimal set is a subset of the feasible set.

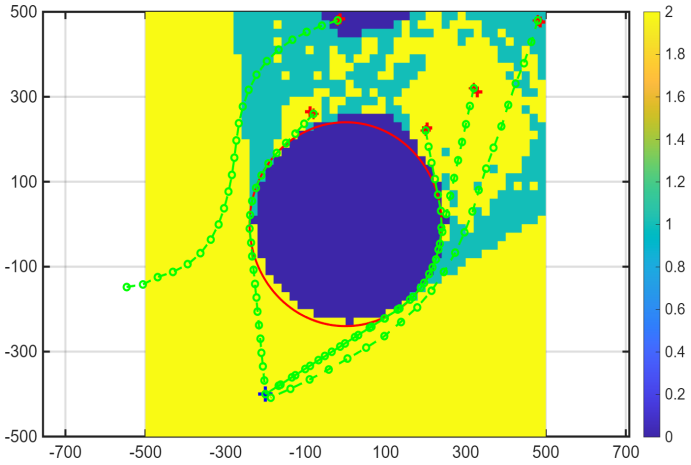

Fig. 4 Optimal and feasible sets found by DDPG-IPM implemented in Matlab fmincon.

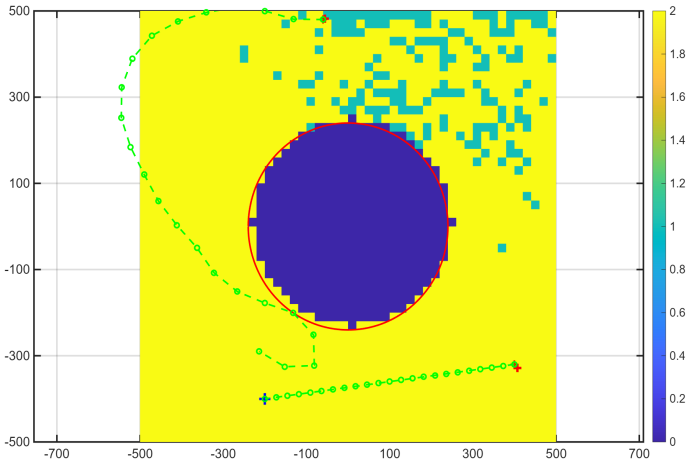

Fig. 5 Optimal and feasible sets found by DDPG-SQP implemented in Matlab fmincon.

In this section, we consider the path-planning problem in the presence of a single obstacle, as discussed in [16]. The center of the obstacle is located at $(a, b) = (0, 0)$, its radius is $R = 240$, the destination is $(\bar{x}_f, \bar{y}_f) = (-200, -400)$, and the total number of line segments is $f = 22$.

A path is regarded as feasible if the distance between its endpoint and the destination is less than 100. A path is regarded as locally optimal if the corresponding segment lengths, headings, slack variables, and Lagrange multipliers satisfy the KKT conditions.

We use three algorithms to solve Problem (4): the arc-search IPM algorithm (III.1), and the IPM and SQP algorithms implemented in Matlab's `fmincon` function with the options `interior-point` and `sqp`, respectively. To ensure a fair comparison, the same initial path is used for all three algorithms for each starting point under consideration.

For a given starting point, if a feasible path obtained by DDPG is available, it is used as the initial path. Otherwise, an initial path is generated using heuristic rules. If the heuristics fail to generate an interior point (i.e., a feasible initial path satisfying the strict inequality constraints), the analytic-center method described in [6, 32] is employed to generate an initial path.

As a reference, the feasible set of the trained DDPG agent is shown in Figure 2. In the figure, a representative starting point is marked by a red '+', the destination is marked by a blue '+', the yellow region represents the feasible set, the blue region represents the infeasible set, and a feasible path is displayed in green. The same convention is adopted in

all subsequent figures.

The feasible set and the optimal set obtained by the arc-search IPM are identical, indicating that the algorithm successfully finds an optimal path for every starting point in the feasible set (100% success rate). The result is shown in Figure 3, which demonstrates that the optimal set covers all starting points outside the obstacle. Two representative optimal paths are displayed in green.

Figure 4 shows the optimal (the area in yellow) and feasible (the area in cyan) sets obtained by the IPM implementation of Matlab's `fmincon`. The blue region outside the obstacle corresponds to starting points for which `fmincon` fails to compute an optimal solution. Although the Matlab IPM implementation fails to satisfy the optimality criterion for many starting points, it often terminates at solutions whose endpoints are very close to the destination. Clearly, the optimal set obtained by the Matlab IPM implementation is smaller than that obtained by the arc-search IPM. However, its feasible set is significantly larger than its optimal set.

Figure 5 shows the optimal (the area in yellow) and feasible (the area in cyan) sets obtained by the SQP implementation of Matlab's `fmincon`. Again, the optimal set is a subset of the feasible set, and both sets are smaller than those obtained by the arc-search IPM.

C. Path planning in the presence of multiple obstacles

In this section, we consider the path-planning problem in the presence of three obstacles, as discussed in [16]. The centers of the obstacles are located at $(a_1, b_1) = (0, 0)$, $(a_2, b_2) = (200, -400)$, and $(a_3, b_3) = (-300, 200)$, with corresponding radii $R_1 = 240$, $R_2 = 150$, and $R_3 = 100$, respectively. The destination is $(\bar{x}_f, \bar{y}_f) = (-200, -400)$, and the total number of line segments is $f = 22$. The path-planning problem is then formulated as follows:

$$\min \ r \tag{34a}$$

$$s.t. \ \bar{x}_f = \bar{x}_0 + r \sum_{i=0}^{f-1} \cos(\theta_i), \tag{34b}$$

$$\bar{y}_f = \bar{y}_0 + r \sum_{i=0}^{f-1} \sin(\theta_i), \tag{34c}$$

$$\left[\bar{x}_0 + r \sum_{i=0}^{k-1} \cos(\theta_i) - a_1\right]^2 + \left[\bar{y}_0 + r \sum_{i=0}^{k-1} \sin(\theta_i) - b_1\right]^2 - R_1^2 \geq 0, \quad k = 1, \ldots, f, \tag{34d}$$

$$\left[\bar{x}_0 + r \sum_{i=0}^{k-1} \cos(\theta_i) - a_2\right]^2 + \left[\bar{y}_0 + r \sum_{i=0}^{k-1} \sin(\theta_i) - b_2\right]^2 - R_2^2 \geq 0, \quad k = 1, \ldots, f, \tag{34e}$$

$$\left[\bar{x}_0 + r \sum_{i=0}^{k-1} \cos(\theta_i) - a_3\right]^2 + \left[\bar{y}_0 + r \sum_{i=0}^{k-1} \sin(\theta_i) - b_3\right]^2 - R_3^2 \geq 0, \quad k = 1, \ldots, f, \tag{34f}$$

$$\theta_k - \theta_{k-1} + 0.5 \geq 0, \quad k = 1, \ldots, f-1, \tag{34g}$$

$$\theta_{k-1} - \theta_k + 0.5 \geq 0, \quad k = 1, \ldots, f-1, \tag{34h}$$

$$bu(k) \geq x(k) \geq bl(k), \quad k = 1, \ldots, n, \tag{34i}$$

where $bl(k)$ and $bu(k)$ denote the lower and upper bounds of the k-th variable, respectively. Specifically, $bl(1) = 1$ and $bl(k) = -2\pi$ for $k \neq 1$, while $bu(1) = 200$ and $bu(k) = 2\pi$ for $k \neq 1$. The problem contains two equality constraints, namely (34b) and (34c); $3f$ nonlinear inequality constraints, where (34d), (34e), and (34f) each represent f nonlinear inequality constraints; $2f - 2$ linear inequality constraints given by (34g) and (34h); and $2n$ bound constraints given by (34i). The decision variable vector is $\mathbf{x} = (r, \theta_0, \theta_1, \ldots, \theta_{f-1})^T$, with $n = f + 1$ variables. As in (5), the equality constraints can be grouped into the vector equation $\mathbf{h}(\mathbf{x}) = \mathbf{0}$, whose components are given by (34b) and (34c), while the inequality constraints can be grouped into the vector inequality $\mathbf{g}(\mathbf{x}) \geq \mathbf{0}$, whose components are given by (34d), (34e), (34f), (34g), (34h), and (34i).

To improve computational efficiency, the inequality constraints are further divided into two categories: nonlinear inequality constraints, consisting of (34d), (34e), and (34f), and linear inequality constraints, consisting of (34g), (34h), and (34i). Since the second derivatives of the linear inequality constraints are identically zero, exploiting this

structure reduces the computational cost associated with Hessian evaluations. A path is regarded as locally optimal if the corresponding variables satisfy the KKT conditions for Problem (34).

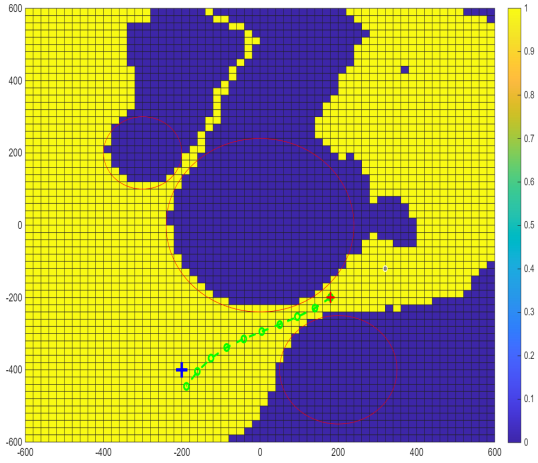

Fig. 6 Feasible set found by DDPG implemented in Matlab.

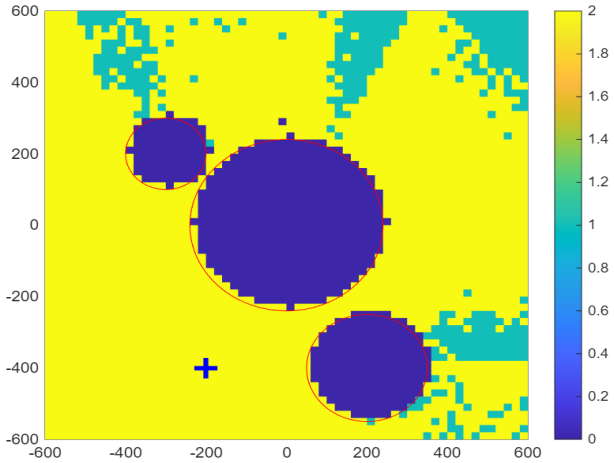

Fig. 7 Feasible and optimal sets found by DDPG-arc-search implemented in Matlab.

Again, we use the reinforcement-learning algorithm DDPG to generate an initial feasible set; see [16] for details. The feasible set obtained by DDPG is shown in Figure 6. Using the feasible paths generated by DDPG as initial paths, we then apply the arc-search IPM to determine both the feasible set and the optimal set. If the DDPG agent fails to generate a feasible path for a given starting point, an initial path is generated using heuristic rules. If the heuristics fail to generate an interior point, the analytic-center method described in [6, 32] is employed to generate an initial path. The resulting feasible set (shown in cyan) and optimal set (shown in yellow) obtained by the arc-search IPM are displayed in Figure 7. The figure shows that the arc-search IPM successfully finds a feasible path to the destination for almost every starting point under consideration.

The IPM and SQP implementations of Matlab's `fmincon` are also applied to this problem using the same initial paths as those used by the arc-search IPM. The corresponding feasible (shown in cyan) and optimal (shown in yellow) sets obtained by the Matlab IPM and SQP implementations are shown in Figures 8 and 9, respectively. Except for starting points whose shortest path to the destination is a straight line, the Matlab IPM and SQP methods perform poorly on this problem. Even for some initially feasible paths, the Matlab IPM and SQP implementations fail to maintain feasibility while attempting to improve satisfaction of the KKT conditions.

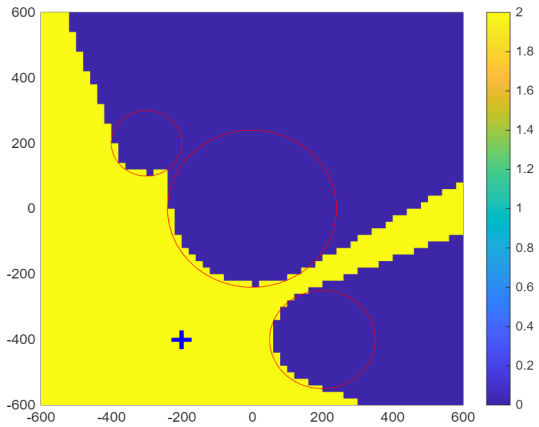

Fig. 8 Feasible and optimal sets found by DDPG-IPM implemented in Matlab.

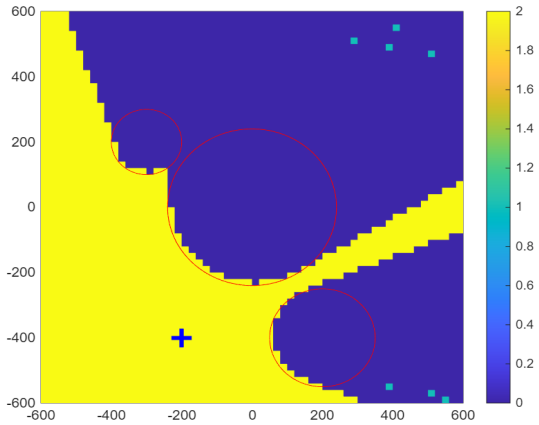

Fig. 9 Feasible and optimal sets found by DDPG-SQP implemented in Matlab.

V. Conclusions

In this paper, we developed a hybrid method combining deep deterministic policy gradient (DDPG) and an arc-search interior-point method (ASIPM), which leverages the strengths of each approach to mitigate the limitations of the other. Numerical simulations demonstrate that the proposed method is effective and attractive. Although the mathematical model is based on a basic engagement zone (BEZ), the proposed strategy can be readily extended to a weapon engagement zone (WEZ). Our future work will focus on extending the method to a more realistic setting with dynamically evolving engagement zones.

References

[1] A. Alexander, K. Venkatesan, J. Mounsef, and K. Ramanujam, A Comprehensive Survey of Path Planning Algorithms for Autonomous Systems and Mobile Robots: Traditional and Modern Approaches, IEEE Access, vol. 13, pp. 176287-176326, 2025.

[2] R.H. Byrd, J.C. Gilbert, J. Nocedal, A trust region method based on interior point techniques for nonlinear programming. Math. Program. 89, 149–185, 2000.

[3] R.H. Byrd, E. Mary Hribar, and J. Nocedal, An Interior Point Algorithm for Large-Scale Nonlinear Programming, SIAM Journal on Optimization, 9(4), pp. 877–900, 1999.

[4] R.H. Byrd, R.B. Schnabel, and G.A. Shultz, Approximate solution of the trust region problem by minimization over two-dimensional subspaces," Mathematical Programming, 40, pp 247–263, 1988.

[5] Y. Chao, and X. Xiang, A path planning algorithm for UAV based on improved Q-learning, In 2018 2nd international conference on robotics and automation sciences (ICRAS), pp. 1-5. IEEE, 2018.

[6] X. B. Chen and M. M. Kostreva, Global convergence analysis of algorithms for finding feasible points in norm-relaxed MFD, Journal of Optimization Theory and Applications 100(2), 287-309, 1999.

[7] T.F. Coleman, and A. Verma, A preconditioned conjugate gradient approach to linear equality constrained minimization, Computational Optimization and Applications, Vol. 20, No. 1, pp. 61–72, 2001.

[8] P. M. Dillon, M. D. Zollars, I. E. Weintraub, and A. Von Moll, Optimal trajectories for aircraft avoidance of multiple weapon engagement zones, Journal of Aerospace Information Systems, 2023.

[9] A. Franco, and V. Santos, Short-term path planning with multiple moving obstacle avoidance based on adaptive MPC, In 2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), pp. 1-7. IEEE, 2019.

[10] X. Gao, L. Yan, Z. Li, G. Wang, and I. Chen, Improved deep deterministic policy gradient for dynamic obstacle avoidance of mobile robot, IEEE Transactions on Systems, Man, and Cybernetics: Systems 53(6), 3675-3682, 2023.

[11] P.E. Gill, W. Murray, M.A. Saunders, and M.H. Wright, Procedures for optimization problems with a mixture of bounds and general linear constraints," ACM Trans. Math. Software, Vol. 10, pp 282–298, 1984.

[12] Y. Gu, Z. Zhu, J. Lv, L. Shi, Z. Hou, and S. Xu, DM-DQN: Dueling Munchausen deep Q network for robot path planning, Complex & Intelligent Systems 9(4), pp. 4287-4300, 2023.

[13] S.P Han, A Globally Convergent Method for Nonlinear Programming, J. Optimization Theory and Applications, 22, p. 297-309, 1977.

[14] K. Karur, N. Sharma, C. Dharmatti, and J. E. Siegel, A survey of path planning algorithms for mobile robots. Vehicles, 3(3) pp. 448-468, 2021.

[15] Q. Le and I. Weintraub, Basic engagement zone aoidance using pseudo-spectral methods, AIAA 2026, Jan 8, 2026.

[16] Q. Le, Y. Yang, and I. Weintraub, Path planning using deep deterministic policy gradient: a reinforcement learning approach, Technical report, Hampton University, 2026.

[17] Matlab, Constrained nonlinear optimization algorithms, accessed on March 31, 2026, https://www.mathworks.com/help/optim/ug/constrained-nonlinear-optimization-algorithms.html

[18] N. Megiddo, Pathways to the Optimal Set in Linear Programming, In N. Megiddo (eds), Progress in Mathematical Programming, Springer-Verlag New York, Inc, 1989.

[19] J.J. Moré, and D.C. Sorensen, Computing a Trust Region Step, SIAM Journal on Scientific and Statistical Computing, Vol. 3, pp 553–572, 1983.

[20] J. Nocedal and S. J. Wright, Numerical Optimization, Springer, 2006.

[21] G. Pepe, M. Laurenza, D. Antonelli, and A. Carcaterra, A new optimal control of obstacle avoidance for safer autonomous driving, In 2019 AEIT International Conference of Electrical and Electronic Technologies for Automotive (AEIT AUTOMOTIVE), pp. 1-6. IEEE, 2019.

[22] M.J.D. Powell, The Convergence of variable metric methods for nonlinearly constrained optimization calculations, Nonlinear Programming 3, (O.L. Mangasarian, R.R. Meyer and S.M. Robinson, eds.), Academic Press, 1978.

[23] T. Steihaug, The Conjugate Gradient Method and Trust Regions in Large Scale Optimization, SIAM Journal on Numerical Analysis, 20, pp 626–637, 1983.

[24] I. A. Von Moll, and Weintraub, Basic engagement zones. Journal of Aerospace Information Systems, 21(10), pp.885-891, 2024.

[25] R. A. Waltz, J. L. Morales, J. Nocedal, and D. Orban, An interior algorithm for nonlinear optimization that combines line search and trust region steps, Mathematical Programming, 107(3), pp. 391–408, 2006.

[26] K. Wang, C. Mu, Z. Ni, and D. Liu, Safe reinforcement learning and adaptive optimal control with applications to obstacle avoidance problem, IEEE Transactions on Automation Science and Engineering, 21(3) pp. 4599-4612, 2024

[27] I.E. Weintraub, and A. Von Moll, C.A. Carrizales, N. Hanlon, and Z.E. Fuchs, An optimal engagement zone avoidance scenario in 2-D, In AIAA SciTech 2022 Forum (p. 1587), 2022.

[28] M. Yamashita, E. Iida, and Y. Yang, An infeasible interior-point arc-search algorithm for nonlinear constrained optimization, Numerical Algorithms, 89, 249-275, 2018.

[29] Y. Yang, Arc-Search Techniques for Interior-Point Methods, CRC Press, 2020.

[30] Y. Yang, An arc-search interior-point algorithm for nonlinear constrained optimization, Computational Optimization and Applications, 90, pp. 969–995, 2025.

[31] Y. Yang, Q. Le, I. Weintraub, et. al., A survey on methods for path planning in the presence of obstacles, Technical Report, Hampton University, 2026.

[32] Y. Ye, Interior Point Algorithms: Theory and Analysis, John Wiley & Son Inc., New York, 1997.