Model-Based Stochastic Search for Large Scale Optimization of Multi-Agent UAV Swarms
基于模型的随机搜索用于多智能体无人机群体的大规模优化。

***Abstract*— 强化学习社区的近期工作表明，进化策略是一种相较于其他强化学习方法而言，更快且可扩展的替代方案。在本文中，我们展示了进化策略是基于模型的随机搜索方法的一个特例。这类算法具有良好的渐近收敛性质和已知的收敛速率。我们展示了如何利用这些方法有效地解决协作和竞争的多智能体问题。我们在两个复杂的多智能体无人机（UAV）集群战斗场景中证明了这种方法的有效性：一个场景是一队固定翼飞机必须攻击一个防守严密的基地，另一个场景是两队智能体针锋相对，以击败对方 †。**

I. Introduction

强化学习关注的是通过反复的互动和试错来最大限度地提高环境的回报。这些方法通常依赖于贝尔曼方程（Bellman equation）的各种渐进方法，包括价值函数逼近、策略梯度方法等 [1]。另一方面，演化计算（Evolutionary Computation）方面提供了一系列黑盒优化和启发式搜索方法 [2]。例如在视觉任务中，这类方法已经被用于优化神经网络的结构 [3]。

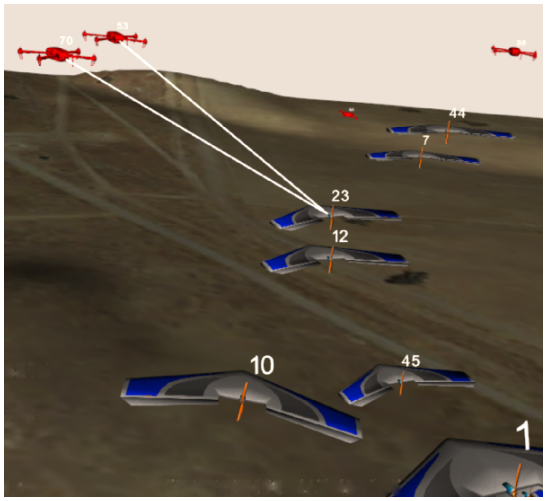


Fig. 1: SCRIMMAGE 是一个多智能体仿真环境。在这个场景中，蓝队的固定翼飞行器攻击红队的四旋翼无人机防御者。白色的线表示未命中的攻击。

最近，Salimans 等人展示了一种特定的进化计算方法变种，称为进化策略（ES），它是一种相对于其他强化学习方法快速且可扩展的替代方案，可以在 10 分钟内解决复杂的 MuJoCo 类人任务 [4]。作者们认为，ES 相比其他强化学习方法有几个优势：1）无需通过策略反向传播梯度，这打开了更广阔的策略参数化类别；2）ES 方法可以进行大规模并行化，这使得可以扩大学习规模以解决更大、更复杂的问题；3）ES 通常找到的策略比其他强化学习方法更具鲁棒性；4）ES 更擅长将学习效果归功于长期变化的策略，这有助于解决时间跨度更长、奖励稀疏的任务。在这项工作中，我们通过使用 ES 解决问题，利用了这四个优点：1）更复杂且可解析的策略架构，允许进行安全性考虑；2）一个包含许多交互元素的大规模模拟环境；3）多种随机源，包括初始条件的变化、干扰等；4）稀疏的奖励，只在长期阶段的末尾出现。

对进化计算算法的一个常见批评是缺乏收敛分析或保证。当然，对于非可微和非凸目标函数的问题，分析总是困难的。然而，我们展示了 Salimans 等人提出的进化策略算法 [4] 是一类被称为基于梯度的自适应随机搜索（GASS）[5] 的基于模型的随机搜索方法的一个特例。这类方法概括了许多随机搜索方法，如众所周知的交叉熵方法（CEM）[6]，CMA-ES[7] 等。通过将一个非可微、非凸优化问题视为梯度下降问题，我们可以得到良好的渐进收敛性质和已知的收敛速率 [8]。

在对进化策略的收敛性有了更多的信心后，我们演示了如何使用 ES 来有效地解决合作和竞争的大规模多智能体问题。许多解决多智能体问题的方法依赖于手动设计和手动调整的算法（参见 [9] 的评述）。其中一个例子是，分布式模型预测控制依赖于每个智能体上的独立 MPC 控制器，并在它们之间进行一定程度的协调 [10], [11]。这些控制器需要手动设计动态模型、成本函数、反馈增益等，并需要专门的领域知识。此外，将这些方法扩展到更复杂的问题上仍然是一个问题。进化算法也被试验为解决多智能体问题的解决方案；通常是在较小、较简单的环境中，以及政策的复杂度较低 [12], [13]。最近，一种将 MPC 与遗传算法结合的混合方法已被用于无人机集群战斗场景，用于演示为手动调整的 MPC 控制器进化成本函数 [14]。

在这项工作中，我们在两个复杂的多智能体无人机（UAV）集群战斗场景中证明了我们方法的有效性：一个场景是一队固定翼飞机必须攻击一个防守严密的基地，另一个场景是两队智能体针锋相对，以击败对方。这种场景以前在模拟环境中被考虑过，但模拟环境的真实度和复杂性较低 [15], [14]。我们利用最近开发的 SCRIMMAGE 多智能体模拟器的计算效率和灵活性进行实验（图1）[16]。我们比较了 ES 和交叉熵方法的性能。我们还展示了在竞争场景中，随着时间的推移，策略如何学习协调一种策略，以应对敌人学习做同样的事情。

II. Problem Formulation

可以将我们的问题表示为不可微、非凸优化问题：

$$\theta^* = \arg\max_{\theta \in \Theta} J(\theta) \tag{1}$$

其中 $\Theta \subset \mathbb{R}^n$ 是一个非空紧凑集，代表解空间，$J(\theta)$ 是一个非可微、非凸的实值目标函数 $J:\Theta\to\mathbb{R}$。θ 可以是我们问题的任何决策变量的组合，包括神经网络权重、PID 增益、硬件设计参数等，它们影响了返回值 J 的结果。对于强化学习问题，θ 通常代表政策的参数，J 是将政策连续应用于环境的隐函数。我们首先回顾如何使用基于梯度的自适应随机搜索方法解决这个问题，然后展示如何将 ES 算法视为这些方法的一个特例。

A. 基于梯度的自适应随机搜索

基于模型的随机搜索方法的目标是通过指定一个概率模型（因此被称为“基于模型”）进行抽样，将非可微优化问题 (1) 转化为一个可微的问题 [8]。让这个模型为 $p(\theta|\omega)=f(\theta;\omega),\omega\in\Omega$，其中 ω 是定义概率分布的参数（例如，对于高斯分布，分布完全由均值和方差 $\omega=[\mu,\sigma]$ 参数化）。那么，$J(\theta)$ 在分布 $f(\theta;\omega)$ 上的期望总是小于 J 的最优值，即：

$$\int_\Theta J(\theta)f(\theta;\omega)d\theta \le J(\theta^*) \tag{2}$$

基于梯度的自适应随机搜索（GASS）的思想是，人们可以在分布参数的空间 Ω 而不是 Θ 中进行搜索，以寻找一种最大化 (2) 中期望的分布：

$$\omega^* = \arg\max_{\omega\in\Omega}\int_\Theta J(\theta)f(\theta;\omega)d\theta \tag{3}$$

最大化这个期望相当于找到一种在最优 θ 周围最大分布的分布。然而，与最大化 (1) 不同，这个目标函数现在可以相对于 ω 连续且可微。在对分布形式的一些假设下，关于 ω 的梯度可以被推进期望值内部。

[8] 提出的 GASS 算法适用于指数族概率密度：

$$f(\theta;\omega)=\exp\{\omega^\intercal T(\theta)-\phi(\theta)\} \tag{4}$$

其中，$\phi(\theta)=\ln\int\exp(\omega^\intercal T(\theta))d\theta$，$T(\theta)$ 是充分统计量的向量。由于我们关心的是与使用高斯噪声采样的参数扰动的 ES 的关系，我们假设 $f(\theta;\omega)$ 是高斯的。此外，由于我们关注的是学习大量的参数（即神经网络的权重），我们假设每个参数上都有独立的高斯分布。然后，$T(\theta)=[\theta,\theta^2]^\intercal\in\mathbb{R}^{2n}$ 和 $\omega=[\mu/\sigma^2,-1/n\sigma^2]^\intercal\in\mathbb{R}^{2n}$，其中 μ 和 σ 是分别对应每个参数分布的均值和标准差的向量。

Algorithm 1 Gradient-Based Adaptive Stochastic Search

Require: 学习率 α_k, 样本大小 N_k, 初始策略参数 $\omega_0=[\mu_0,\sigma_0^2]^\intercal$, 平滑函数 $S()$, 小常数 $\gamma>0$.

1: **for** $k=0,1,\cdots$ **do**
2: 采样 $\theta_k^i \overset{iid}{\sim} f(\theta;\omega_k), i=1,2,\cdots,N_k$.
3: 计算返回值 $w_k^i=S(J(\theta_k^i))$ for $i=1,\cdots,N_k$.
4: 计算方差 $V_k=\hat{V}_k+\gamma I$, eq (5),(6)
5: 计算正则化参数 $\eta=\sum_{i=1}^{N_k} w_k^i$.
6: 更新参数 ω_{k+1}:
7: $\omega_{k+1}\leftarrow\omega_k+\alpha_k\frac{1}{\eta}V_k^{-1}\sum_{i=1}^{N_k}w_k^i\left(\begin{bmatrix}\theta_k^i\\(\theta_k^i)^2\end{bmatrix}-\begin{bmatrix}\mu\\\sigma^2+\mu^2\end{bmatrix}\right)$
8: **end for**

我们为这个特定的概率模型集合提出 GASS 算法（算法1），尽管更通用的指数族分布的收敛性分析是成立的。对于每次迭代 k，GASS 算法涉及到绘制 N_k 个参数的样本 $\theta_k^i \overset{iid}{\sim} f(\theta;\omega_k), i=1,2,\cdots,N_k$。然后，这些参数被用来对返回函数 $J(\theta_k^i)$ 进行采样。这些返回值通过一个塑造函数 $S(\cdot):\mathbb{R}\to\mathbb{R}^+$，然后用于计算模型参数 ω_{k+1} 的更新。

塑造函数 $S(\cdot)$ 需要是非减的，并且对于有界输入，其上下界是有限的，下界离 0 有一定距离。此外，集合 $\{\arg\max_{\theta\in\Theta}S(J(\theta))\}$ 必须是原始问题的解集 $\{\arg\max_{\theta\in\Theta}J(\theta)\}$ 的非空子集。塑造函数可以用来调整探索/利用权衡，或者在采样时处理离群值。GASS 的原始分析假定 $S_k(\cdot)$ 有一个更通用的形式，其中 S 可以在每次迭代时改变。为了简便，我们这里假设它是确定的，并且每次迭代不变。

GASS 可以被认为是二阶梯度方法，并需要估计采样参数的方差：

$$\hat{V}_k=\frac{1}{N_k-1}\sum_{i=1}^{N_k}T(\theta_k^i)T(\theta_k^i)^\intercal - \frac{1}{N_k^2-N_k}\left(\sum_{i=1}^{N_k}T(\theta_k^i)\right)\left(\sum_{i=1}^{N_k}T(\theta_k^i)\right)^\intercal. \tag{5}$$

在实际应用中，如果参数空间 Θ 非常大，如神经网络中的情况，这种方差矩阵的尺寸会是 $2n\times 2n$，其计算成本会非常高。在我们的工作中，我们通过对每个独立高斯分布的参数的独立计算方差来近似 $\hat{V}_k$。轻微地滥用一下符号，我们考虑 $\tilde{\theta}_k^i$ 作为 θ_k^i 的一个标量元素。然后，对于每个标量元素 $\tilde{\theta}_k^i$，我们有一个 2×2 的方差矩阵：

$$\hat{V}_k=\frac{1}{N_k-1}\sum_{i=1}^{N_k}\begin{bmatrix}\tilde{\theta}_k^i\\(\tilde{\theta}_k^i)^2\end{bmatrix}\begin{bmatrix}\tilde{\theta}_k^i & (\tilde{\theta}_k^i)^2\end{bmatrix} - \frac{1}{N_k^2-N_k}\left(\sum_{i=1}^{N_k}\begin{bmatrix}\tilde{\theta}_k^i\\(\tilde{\theta}_k^i)^2\end{bmatrix}\right)\left(\sum_{i=1}^{N_k}\begin{bmatrix}\tilde{\theta}_k^i & (\tilde{\theta}_k^i)^2\end{bmatrix}\right). \tag{6}$$

定理 1 表明，GASS 产生了一系列的 ω_k，这些序列收敛到一个限制集合，该集合指定了一组能够最大化 (3) 的分布。这个集合中的分布将确定如何选择 θ^* 以最终最大化 (1)。像大多数非凸优化算法一样，我们无法保证能够到达全局最大值，但使用概率模型和仔细选择形状函数应该可以帮助避免早期收敛到次优的局部最大值。证明依赖于将更新规则表述为一个广义的罗宾斯-蒙罗 Robbins-Monro 算法（参见 [8]，定理 1 和 2）。定理 1 还指定了收敛速率，以迭代次数 k，每次迭代的样本数 N_k，和学习率 α_k 来表示。在实践中，定理 1 暗示需要仔细平衡每次迭代的样本数量的增加和随着迭代进行学习率的减少。

Assumption 1

i) 学习率 $\alpha_k>0$, 当 $k\to\infty$ 时, $\alpha_k\to 0$, 并且 $\sum_{k=0}^{\infty}\alpha_k=\infty$。
ii) 样本大小 $N_k=N_0k^\xi$，其中 $\xi>0$；同时，α_k 和 N_k 共同满足 $\alpha/\sqrt{N_k}=\mathcal{O}(k^{-\beta})$。
iii) $T(\theta)$ 在 Θ 上是有界的。
iv) 如果 ω^* 是 *(3)* 的局部最大值，则 $\int_\Theta J(\theta)f(\theta;\omega)d\theta$ 的海森矩阵在 ω^* 附近是连续的和负定的。

Theorem 1
假设条件*1*成立。设定 $\alpha_k = \alpha_0/k^{\alpha}$，其中 $0 < \alpha < 1$。让 $N_k = N_0 k^{\tau-\alpha}$，其中 $\tau > 2\alpha$ 是一个常数。那么由算法*1*生成的序列 $\{\omega_k\}$ 以概率 *1* 收敛到一个极限集，收敛速度为 $\mathcal{O}(1/\sqrt{k^{\tau}})$。

B. Evolutionary Strategies

我们现在回顾一下 [4] 提出的 ES 算法，并展示它是如何是 GASS 算法的一阶近似。与 GASS 相同，ES 算法由以下两个阶段组成：1）随机扰动参数服从高斯分布。2）计算回报并计算参数的更新。ES 算法的步骤见算法2。计算出返回值后，再用合适的塑造函数 $S(\cdot)$ 作用到返回值上 [17]。Salimans 等人使用秩变换函数（rank transformation function）作为 $S(\cdot)$，他们认为这样减少了每次迭代中异常值的影响，并有助于避免局部最优。

我们现在回顾一下 [4] 提出的 ES 算法，并展示它如何是 GASS 算法的一阶近似。ES 算法由与 GASS 相同的两个阶段组成：1）随机扰动参数，其中噪声从高斯分布中采样。2）计算回报 (returns) 并计算对参数的更新。该算法在算法2中概述。计算出返回值 (回报) 后，再用合适的塑造函数 $S(\cdot)$ 作用到返回值 (回报) 上 [17]。Salimans 等人使用了一个排名变换函数作为 $S(\cdot)$，他们认为这减少了每次迭代中异常值的影响，并有助于避免局部最优。

Algorithm 2 Evolution Strategies

Require: 学习率 α_k，噪声标准差 γ，初始策略参数 θ_0，平滑函数 $S()$。

1: **for** $k = 0, 1, \cdots$ **do**
2: 从正态分布 $\mathcal{N}(0, I^{n\times n})$ 中采样 $\varepsilon_1, \cdots, \varepsilon_n$
3: 对于 $i = 1, \cdots, N_k$，计算回报 $w_k^i = S(J(\theta_k + \gamma\varepsilon_i))$
4: 更新 θ_{k+1}:
5: $\theta_{k+1} \leftarrow \theta_k + \alpha_k \frac{1}{N_k \gamma} \sum_{i=1}^{N_k} w_k^i \varepsilon_i$
6: **end for**

显然，当采样分布是点分布时，ES 算法是 GASS 算法的一个子案例。我们也可以通过忽略算法1中的第 7 行的方差项来恢复 ES 算法。ES 使用样本数 N_k 代替了 GASS 中的标准化 (归一化) 项 η。GASS 中的小常数 γ 变成了 ES 算法中的方差项。算法2中的更新规则涉及将缩放回报乘以噪声，这正好是算法1中的 $\theta_k^i - \mu$。

可以看到 ES 具有与 GASS 分析相同的渐近收敛率。虽然 GASS 是一种二阶方法而 ES 只是一阶方法，但在实践中，ES 使用近似的二阶梯度下降方法来调整学习率，从而可以使学习加速和稳定。该方法的应用的例子包括 ADAM，RMSProp，具有动量的 SGD 等，其已经显示在神经网络中非常好的表现。因此，我们可以将 ES 视为 GASS 使用的完全二阶方差更新的一阶近似。在本实验中，我们使用 ADAM [18] 来调整每个参数的学习率。正如 [4] 中所示，当使用自适应学习率时，我们发现调整采样分布的方差对效果几乎没有（只有很小的）改善。我们假设带有自适应学习率的一阶方法足以在优化神经网络时具有良好的性能。然而，对于其他类型的策略参数，GASS 的完全二阶处理可能更有用。也可以混合和匹配哪些参数需要完全方差更新，哪些可以使用一阶近似方法更新。我们使用秩变换函数 $S(\cdot)$ 并保持 N_k 为常数。

C. 学习多智能体问题的结构化策略

现在我们对 ES/GASS 方法的收敛性更有信心，我们展示了如何使用 ES 来优化一个大规模多智能体环境中的复杂策略。我们使用 SCRIMMAGE 多智能体模拟环境 [16]，因为它可以快速并行地模拟复杂的多智能体场景。我们在模拟中使用了具有 10 个和 12 个状态的动力学模型的 6 自由度固定翼飞机和四旋翼飞行器。这些动力学模型允许在现实操作范围内进行全面的运动。风力和控制噪声被建模为加性高斯噪声形式的随机干扰。地面和空中碰撞可能导致飞机被摧毁。我们还加入了一个武器模块，允许在从飞机鼻部投射出的固定锥体内对敌人进行瞄准和射击。命中的概率取决于到目标的距离以及目标向攻击者展示的总面积。该面积基于飞机的线框模型和其相对姿态。有关更多细节，请参阅我们的代码和 SCRIMMAGE 模拟器文档。

我们考虑每个智能体使用自己的策略计算自己的控制，但是策略的参数对所有智能体来说是相同的情况。这样，每个智能体可以以分散的方式控制自己，同时也允许有益的群体行为出现。此外，我们假设友方智能体可以相互通信并共享状态（参见图2）。由于我们有大量智能体 (每个队伍最多 50 个)，为了降低通信成本，我们只允许智能体在本地共享信息，即相互接近的智能体可以访问彼此的状态。在我们的实验中，我们允许每个智能体感知最近的 5 个友方智能体的状态，总共收到 $5 * 10 = 50$ 个输入状态消息。

此外，每个智能体都配备了用于探测敌方智能体的传感器。这里没有完全的状态可观测性，我们假设传感器能够感知敌方智能体的相对位置和速度。在我们的实验中，我们假设每个智能体能够感知最近的 5 个敌方智能体，总共有 35 个敌方数据维度（7 个状态 =[相对 xyz 位置、距离和相对 xyz 速度])。传感器还提供有关本方和敌方基地的相对航向和距离的信息（额外的 8 个状态)。加上智能体自身的状态（9 个状态)，策略的观测输入 $\vec{o}(t)$ 的维度为 102 维。这些输入状态被馈送到智能体的策略中，即一个具有 3 个全连接层的神经网络 $f(\vec{o}(t);\theta)$，各层大小分别为 200、200 和 50，输出 3 个标量，表示期望的相对航向 $[x_{ref}, y_{ref}, z_{ref}]$。每个智能体的神经网络具有超过 70,000 个参数。每个智能体使用与其队友相同的神经网络参数，但由于每个智能体在每个时间步进遇到不同的观测，每个智能体的神经网络策略的输出将是唯一的。也可能为每个智能体学习独特的策略；我们将这留作未来的工作。

考虑到无人机飞行中的安全性是一个重要问题，我们设计策略时考虑了安全性和控制因素。神经网络策略输出的相对航向意图由 PID 控制器用于跟踪航向。PID 控制器向飞行器提供低层控制指令 $u(t)$（推力、副翼、升降舵、方向舵)。然而，为了防止神经网络策略导致飞行器撞击地面或友方等情况发生，如果飞行器即将与某物体碰撞，我们使用回避航向来替换神经网络的航向输出。这有助于将学习过程集中在如何智能地与环境和队友互动上，而不是学习如何避免明显的错误。此外，通过以结构化且可解释的方式设计策略，将更容易将从模拟中学到的策略直接应用到现实世界中。由于策略中的神经网络部分不产生低层指令，它对不同的低层控制器、动力学、PID 增益等是不变的。这有助于学习更具可迁移性的用于真实世界应用的策略。

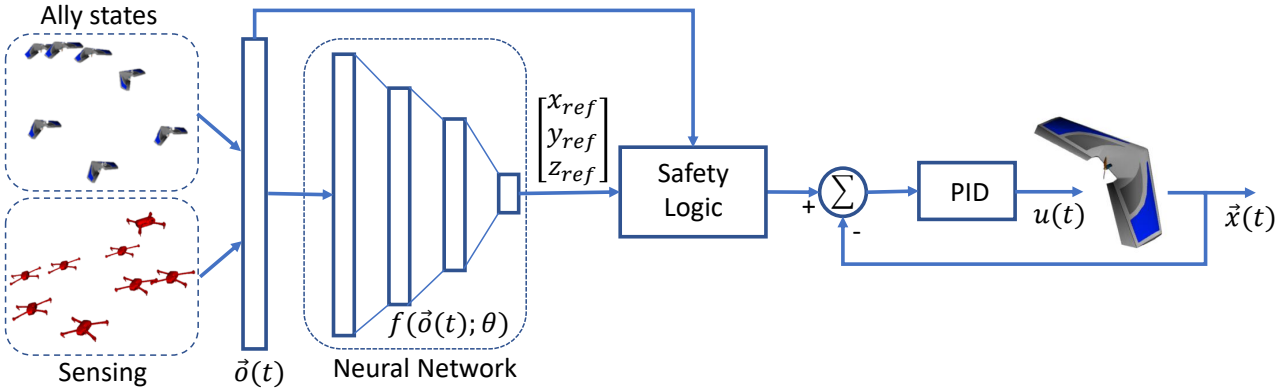


Fig. 2: 每个智能体的策略示意图。附近的友机状态和感知到的敌机，基地位置等，以及智能体自己的状态被输入到到神经网络中，产生相对 xyz 坐标系中的参考目标。目标被送入安全逻辑模块，用于检查与邻机或地面是否碰撞。最终产生出一个参考目标，该目标被传递给 PID 控制器，进而为智能体提供低级控制（推力、副翼、升降舵、方向舵）。

III. Experiments

我们考虑两种场景：一种是基地攻击场景，其中一支由 50 架固定翼飞机组成的团队必须攻击由 20 个四旋翼无人机防守的敌方基地；以及一个团队对抗任务，上述两个团队同时学习击败对方。在这两项任务中，我们使用以下奖励：

$$J = 10 \times (\#\text{kills}) + 50 \times (\#\text{collisions with enemy base}) - 1e-5 \times (\text{distance from enemy base at end of episode}) \tag{7}$$

奖励函数鼓励空对空作战，以及对敌基地的自杀式袭击（例如一群携带有效载荷的廉价一次性无人机）。最后一部分是鼓励飞机在学习的初始阶段向敌人基地方向移动。

A. Base Attack Task

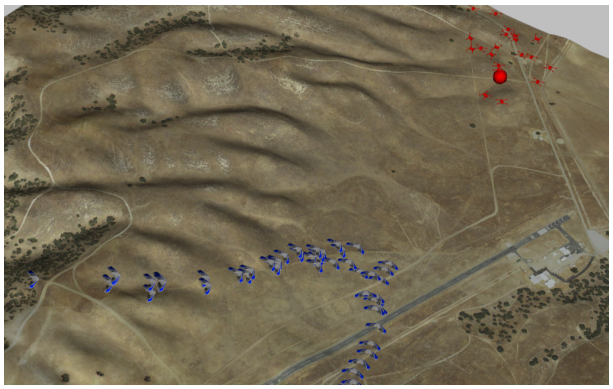


Fig. 3: 基本攻击任务的快照。蓝色固定翼组（左下方）的目标是攻击红色基地（红点，右上方），同时避开或攻击红色旋翼飞行器的防守。

在这个场景中，一支由 50 架固定翼飞机组成的机队必须攻击一个由 20 架四旋翼飞机防守的敌方基地（图3）。四旋翼飞机使用人工策略，在没有敌人的情况下，它们均匀分散以覆盖整个基地。在有敌人的情况下，它们会选择最近的敌人作为目标，调整高度，并进行连续射击。我们使用了 $N_k = 300$、$\gamma = 0.02$，时间步长为 0.1 秒，总的回合长度为 200 秒。双方初始位置在场地两端的固定区域内随机分布。训练在一台装有 Xeon Phi CPU（244 个线程）的机器上进行了两天的全并行化处理。

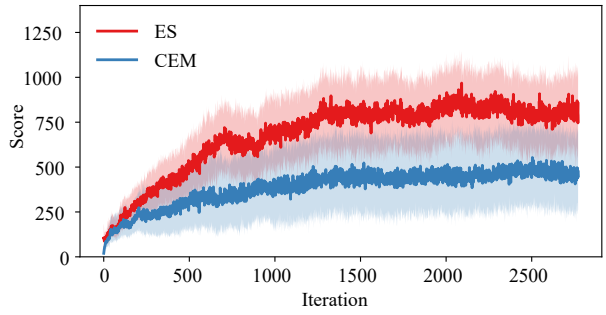
在训练过程中，我们发现固定翼飞机机队学会了一种策略，他们迅速形成 V 字队形并接近基地。一些飞机进行自杀式攻击敌方基地，而其他飞机开始进行空中格斗。我们还将我们实现的 ES 方法与众所周知的交叉熵方法（CEM）进行了比较。结果显示，CEM 的表现明显不如 ES（图4）。我们假设这是因为 CEM 丢弃了一部分采样的参数，因此对于 (3) 的梯度估计更差。与其他完全二阶方法（如 CMA-ES 或完全二阶 GASS 算法）的比较是不现实的，因为神经网络中参数的数量很大，计算这些参数的协方差存在极大的计算困难。

B. Two Team Competitive Match

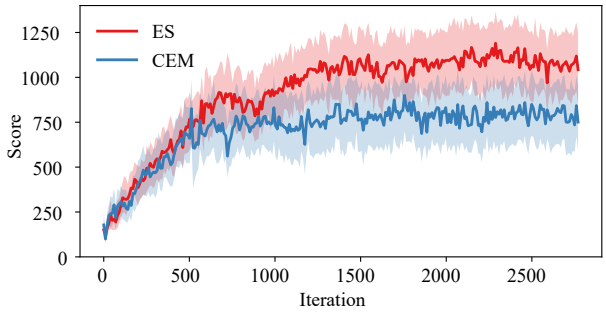
我们考虑的第二个场景是两个团队，每个团队都配备了自己独特的策略来训练他们的飞机（图5）。在每个迭代中，生成 $N_k = 300$ 个模拟，每个模拟都有不同的随机扰动，而且每个团队都有不同的扰动。根据与对手扰动策略对战所得到的分数，计算每个策略的更新。结果是，每个团队都学会了在每个迭代中击败各种对手行为。我们观察到，两个团队的行为很快接近了纳什均衡，双方都试图击败尽可能多的对手飞机，以防止更高得分的自杀攻击（详见补充视频）。最终结果是两个团队相互消灭，以平分结束（图6）。我们假设通过让每个团队与一些过去的敌对团队行为竞争，或者通过构建一个策略库进行选择，可以学习到更多样化的行为，这是进化计算社区经常讨论的话题 [19]。

IV. Conclusion

我们已经展示了进化策略在学习具有数千个参数的复杂任务上的适用性，无论是在竞争性还是合作性的多智能体环境中。通过展示进化策略与更为可理解的基于模型的随机搜索方法之间的联系，我们能够对未来算法设计提供借鉴。未来的工作将包括针对混合参数化进行实



(a) Training



(b) Testing

Fig. 4: 每个迭代的基地攻击任务得分。上图：训练期间由扰动策略获得的得分。平均得分较低，因为这些得分是由随机扰动值参数化的策略产生的。下图：训练过程中由更新的策略参数获得的得分。红色曲线表示进化策略算法，蓝色曲线表示交叉熵方法。粗线是中位数，阴影区域是第 25/75 四分位数边界。

验，例如优化神经网络权重和 PID 增益。在这种情况下，对非神经网络参数的二阶处理可能更为有益，因为系统的行为可能对非神经网络参数的扰动更为敏感。另一个研究方向可能是为团队中的每个智能体优化独特的策略。还可以比较其他用于训练神经网络的进化计算策略，包括使用更多样化种群的方法 [20]，或更多遗传算法类型的启发式方法 [21]。

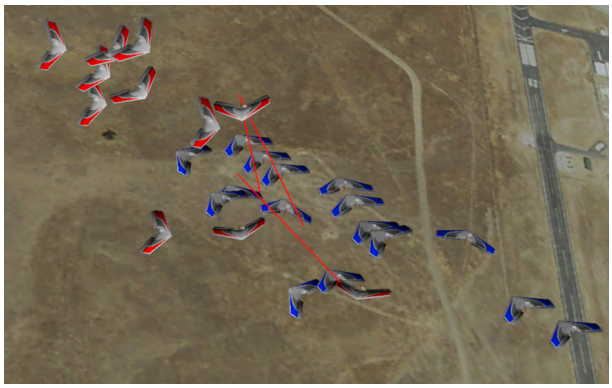
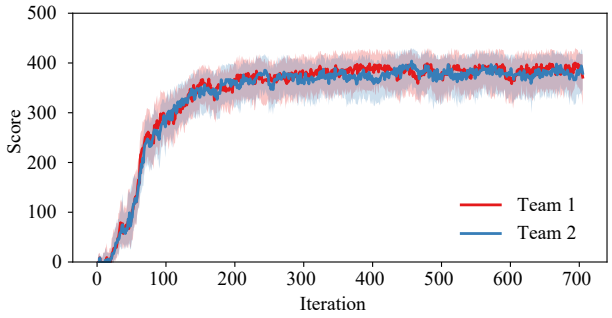
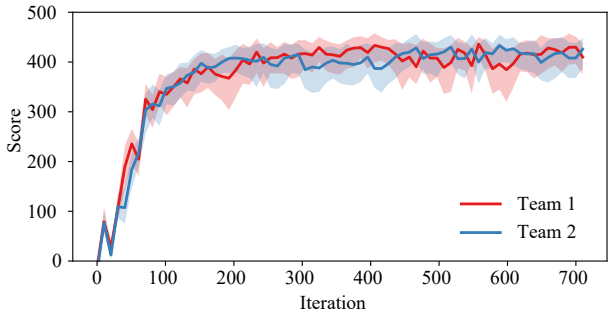


Fig. 5: 两队对抗快照。两队的目标都是在遭受最小损失的同时击落所有敌机，或攻击敌方基地。红线表示成功击中对方



(a) Training



(b) Testing

Fig. 6: 对抗竞赛每次的迭代分数。上图：训练时各团队获得的分数，其中的策略在训练期间由随机扰动值参数化。下图：测试时获得的分数，使用已更新的策略参数训练的框架。红色和蓝色曲线分别显示队 1 和队 2 的分数。

References

[1] Y. Li, "Deep Reinforcement Learning: An Overview," *ArXiv e-prints*, Jan. 2017.

[2] K. Stanley and B. Bryant, "Real-time neuroevolution in the NERO video game," *IEEE transactions on*, 2005. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=1545941

[3] O. J. Coleman, "Evolving Neural Networks for Visual Processing," Thesis, 2010.

[4] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, "Evolution Strategies as a Scalable Alternative to Reinforcement Learning," *ArXiv e-prints*, Mar. 2017.

[5] J. Hu, "Model-based stochastic search methods," in *Handbook of Simulation Optimization*. Springer, 2015, pp. 319–340.

[6] S. Mannor, R. Rubinstein, and Y. Gat, "The cross entropy method for fast policy search," in *Machine Learning-International Workshop Then Conference-*, vol. 20, no. 2, 2003, Conference Proceedings, p. 512.

[7] N. Hansen, "The CMA evolution strategy: A tutorial," *CoRR*, vol. abs/1604.00772, 2016. [Online]. Available: http://arxiv.org/abs/1604.00772

[8] E. Zhou and J. Hu, "Gradient-based adaptive stochastic search for non-differentiable optimization," *IEEE Transactions on Automatic Control*, vol. 59, no. 7, pp. 1818–1832, 2014.

[9] L. Panait and S. Luke, "Cooperative multi-agent learning: The state of the art," *Autonomous Agents and Multi-Agent Systems*, vol. 11, no. 3, pp. 387–434, 2005. [Online]. Available: http://link.springer.com/10.1007/s10458-005-2631-2

[10] J. B. Rawlings and B. T. Stewart, "Coordinating multiple optimization-based controllers: New opportunities and challenges," *Journal of Process Control*, vol. 18, no. 9, pp. 839–845, 2008.

[11] W. Al-Gherwi, H. Budman, and A. Elkamel, "Robust distributed model predictive control: A review and recent developments," *The Canadian Journal of Chemical Engineering*, vol. 89, no. 5, pp.

1176–1190, 2011. [Online]. Available: http://doi.wiley.com/10.1002/cjce.20555

[12] G. B. Lamont, J. N. Slear, and K. Melendez, "UAV swarm mission planning and routing using multi-objective evolutionary algorithms," in *IEEE Symposium Computational Intelligence in Multicriteria Decision Making*, no. Mcdm, 2007, Conference Proceedings, pp. 10–20.

[13] A. R. Yu, B. B. Thompson, and R. J. Marks, "Competitive evolution of tactical multiswarm dynamics," *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans*, vol. 43, no. 3, pp. 563–569, 2013.

[14] D. D. Fan, E. Theodorou, and J. Reeder, "Evolving cost functions for model predictive control of multi-agent uav combat swarms," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, ser. GECCO '17. New York, NY, USA: ACM, 2017, pp. 55–56. [Online]. Available: http://doi.acm.org/10.1145/3067695.3076019

[15] U. Gaerther, "UAV swarm tactics: an agent-based simulation and Markov process analysis," 2015. [Online]. Available: https://calhoun.nps.edu/handle/10945/34665

[16] K. J. DeMarco. (2018) SCRIMMAGE multi-agent robotics simulator. [Online]. Available: http://www.scrimmagesim.org/

[17] D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, J. Peters, and J. Schmidhuber, "Natural evolution strategies." *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 949–980, 2014.

[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[19] K. O. Stanley and R. Miikkulainen, "Competitive coevolution through evolutionary complexification," *Journal of Artificial Intelligence Research*, vol. 21, pp. 63–100, 2004.

[20] E. Conti, V. Madhavan, F. Petroski Such, J. Lehman, K. O. Stanley, and J. Clune, "Improving Exploration in Evolution Strategies for Deep Reinforcement Learning via a Population of Novelty-Seeking Agents," *ArXiv e-prints*, Dec. 2017.

[21] F. Petroski Such, V. Madhavan, E. Conti, J. Lehman, K. O. Stanley, and J. Clune, "Deep Neuroevolution: Genetic Algorithms Are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning," *ArXiv e-prints*, Dec. 2017.