COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

GRADO EN INGENIERÍA MATEMÁTICA E INTELIGENCIA ARTIFICIAL

TRABAJO FIN DE GRADO

Towards intelligent coordination of autonomous platforms for wildfire control through deep reinforcement learning

Author: Pablo Tuñón Laguna

Director: Miguel Leiva Vélez

Madrid, June 2026

I declare, under my responsibility, that the Project submitted under the title

Towards intelligent coordination of autonomous platforms for wildfire control through deep reinforcement learning

at the ICAI School of Engineering of Comillas Pontifical University in the academic year 2025/2026 is my own work, original and unpublished, and has not previously been submitted for any other purpose.

The Project is not plagiarized, either wholly or partially, and the information taken from other documents is duly referenced.

Signed: Pablo Tuñón Laguna Date: 15 / 06 / 2026

Submission of the project is authorized

THE PROJECT DIRECTOR

Firmado Electrónicamente por TUÑON LAGUNA PABLO - ***2762** el día 23/06/2026 con un certificado emitido por AC FNMT Usuarios

Signed: Miguel Leiva Vélez Date: 15 / 06 / 2026

LEIVA VELEZ MIGUEL - 02581929H

Digitally signed by LEIVA VELEZ MIGUEL - 02581929H
Date: 2026.06.24 12:51:11 +02'00'

Acknowledgments

I would like to thank my father for encouraging me to pursue a career in engineering since I was a child, as well as for his unconditional support and for teaching me how to approach problems and make use of tools.

I would also like to thank my family for their constant support and encouragement, for always being there for me, and for making both this thesis and these academic years possible.

Finally, I would like to thank my director, Miguel Leiva, for his guidance, support, and patience. Thank you for those meetings during weekends or after exhausting days, which were extremely valuable to me, and for helping me understand the foundations and challenges that I faced throughout the development of this thesis.

HACIA LA COORDINACIÓN INTELIGENTE DE PLATAFORMAS AUTÓNOMAS PARA EL CONTROL DE INCENDIOS FORESTALES MEDIANTE APRENDIZAJE POR REFUERZO PROFUNDO

Autor: Pablo Tuñón Laguna
Director: Miguel Leiva Vélez

Resumen

La extinción coordinada de incendios forestales mediante medios aéreos es un problema complejo de optimización, control colectivo de múltiples plataformas y gestión de recursos, marcado por su alta dimensionalidad, su observabilidad parcial y una dinámica no lineal y estocástica. El aprendizaje por refuerzo profundo puede alcanzar un alto grado de optimalidad en problemas de este tipo, aunque sigue siendo difícil de aplicar, y todavía no dispone de una referencia común sobre la que comparar estrategias de coordinación. Presentamos *FireCoopBench*, un *benchmark* reproducible y parametrizable para la extinción cooperativa sobre un plano discretizado con propagación estocástica del fuego, optimización de recursos y observabilidad parcial. Sobre él entrenamos una política MAPPO (un método estándar de *policy gradient* extendido para que varios agentes aprendan a la vez) de parámetros compartidos con crítico centralizado, *entity attention*, asignación húngara en la observación, recurrencia y recompensa de *team spirit*, caracterizando cada componente en el Capítulo 6 (cuatro por ablación y el crítico centralizado mediante la comparación con IPPO). La política resultante, evaluada a lo largo de varios episodios estadísticamente diferenciados, resuelve equipos de hasta 15 drones sin necesidad de reentrenamiento o *fine-tuning*, y el tamaño mínimo de equipo cooperativo necesario crece de 4 a 15 con la dificultad. Comparamos frente a aprendices independientes (*independent learners*) y a los controladores heurísticos más conocidos evaluados en este estudio preliminar, y analizamos la robustez de la política aprendida.

Palabras clave: Aprendizaje por refuerzo profundo; cooperación multiagente; extinción de incendios; mecanismos de atención; escalabilidad; generalización; entrenamiento centralizado y ejecución descentralizada.

Resumen ejecutivo

1 Introducción

Los incendios forestales son un desastre natural costoso y peligroso cuya frecuencia y severidad se prevé que aumenten con el cambio climático [1], y su propia extinción es cara y arriesgada bajo observabilidad parcial del frente de fuego. Las plataformas aéreas autónomas (vehículos no tripulados de ala fija y rotatoria capaces de descargar agente extintor) ofrecen una vía

para escalar la respuesta, pero requieren estrategias de coordinación que puedan *entrenarse*, *validarse* y *reproducirse*. El aprendizaje por refuerzo tiene décadas de historia en control y optimización, pero solo al unirse con las redes neuronales profundas se volvió tratable en problemas de alta dimensión, estocásticos y parcialmente observables. Lo hemos visto en Atari y Go, en StarCraft y, más recientemente, en el control en tiempo real del plasma de un tokamak. La extinción cooperativa de incendios es precisamente uno de esos problemas, y por eso la pregunta es oportuna ahora. Este es el ámbito del aprendizaje por refuerzo multiagente (MARL), y el nicho en el que se enmarca este trabajo.

Sin embargo, la literatura de MARL aplicado a incendios presenta una carencia metodológica: no se dispone de un entorno de referencia parametrizable y reproducible sobre el que comparar con rigor las estrategias de coordinación. Este trabajo se propone cubrir ese vacío.

2 Objetivos

El objetivo principal es diseñar, implementar y estudiar un *benchmark* multiagente reproducible para la extinción cooperativa de incendios forestales, junto con una política cooperativa sólida cuyos componentes hayan sido validados de forma independiente. La hipótesis central es que una única política, entrenada de forma centralizada pero ejecutada de manera distribuida por cada agente (paradigma CTDE) y dotada de un crítico centralizado y de las invariancias estructurales adecuadas, puede aprender una estrategia de coordinación que escale a equipos de distinto tamaño. Un mecanismo de atención sobre las entidades del entorno permite operar con un número variable de agentes y, como ventaja añadida, generalizar a tamaños de equipo nunca observados durante el entrenamiento. El trabajo persigue cuatro objetivos secundarios, en consonancia con la planificación del proyecto (Anexo B): (i) implementar un entorno modular de simulación de incendios, parametrizable y reproducible, dotado de un *curriculum* de dificultad incremental; (ii) construir un *pipeline* completo de entrenamiento y validación basado en DRL/MARL y entrenar sobre él la política cooperativa; (iii) evaluar bajo observabilidad parcial las estrategias de coordinación así obtenidas y compararlas con los algoritmos de DRL/MARL y los controladores heurísticos pertinentes; y (iv) mejorar la robustez y la generalización mediante *domain randomization* y un escenario de fuego adversario.

3 Descripción del modelo/sistema/herramienta

El *benchmark*, *FireCoopBench*, se estructura como una escalera de dificultad de tres niveles que comparten una misma interfaz y un mismo modelo de fuego. Fija las partes que deben permanecer constantes para que una comparación sea justa, el entorno y las métricas de evaluación (KPIs), y deja abiertas las que el usuario puede modificar: la red neuronal, la función de recompensa, los hiperparámetros y el propio algoritmo de aprendizaje. Como punto de partida proponemos una función de recompensa por defecto, que sirve de línea base sin

ser parte inmutable del *benchmark*. El entorno cooperativo es un plano discretizado sobre el que de 3 a 10 drones deben extinguir conjuntamente un incendio que se propaga de forma estocástica, gestionando a la vez un depósito de agua finito que debe recargarse en ubicaciones fijas. El entorno es parcialmente observable: cada dron percibe únicamente una porción local y estructurada por entidades del mundo, esto es, su propio estado, los k incendios más cercanos como un conjunto y un cupo de posiciones (*slots*) reservado para los demás drones.

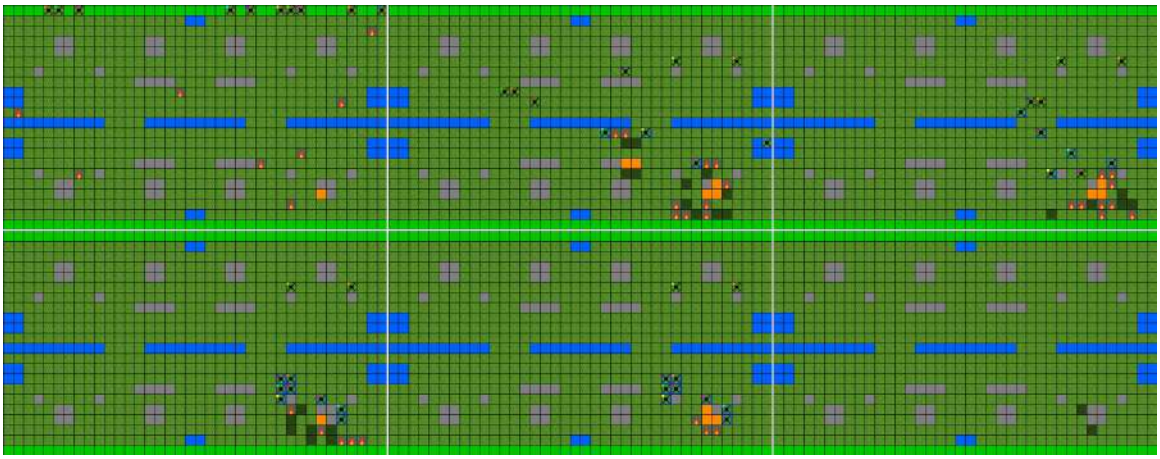

Figura 1: Diseño del entorno cooperativo de *FireCoopBench*: secuencia temporal (de izquierda a derecha) de un episodio con $N = 10$ drones en dificultad máxima. Se aprecian el frente de fuego en propagación, los drones cooperando para confinarlo y las estaciones de recarga de agua sobre el plano discretizado del entorno.

Sobre el entorno cooperativo entrenamos una política *MAPPO* (*Multi-Agent PPO*) con parámetros compartidos [2], dotada de cinco componentes de diseño, cada uno motivado por separado y caracterizado en el Capítulo 6 (cuatro mediante ablación y el crítico centralizado mediante la comparación con IPPO). El primero es un *crítico centralizado*: MAPPO sigue el paradigma actor-crítico, en el que el crítico estima el valor esperado de cada situación y guía así la mejora de la política, y aquí ese crítico consume una porción del estado global accesible solo durante el entrenamiento. El segundo es un *encoder* de observaciones con *entity attention* [3], que trata los incendios cercanos y los compañeros como conjuntos invariantes a permutaciones. El tercero es una *asignación húngara incorporada a la observación*, que dota a la política de un *bias inductivo* explícito de reparto de roles. El cuarto es una red recurrente *LSTM*, que aporta memoria para lidiar con la observabilidad parcial del entorno y anticipar su evolución. El quinto es una función de recompensa modulada por un parámetro de *team spirit* que equilibra las dinámicas colectivas y las individuales. El entrenamiento emplea un *curriculum* de dificultad y una ventaja contrafactual de tipo COMA. La misma red neuronal se reutiliza para cualquier tamaño de equipo, lo que hace posible la evaluación con un número variable de agentes.

4 Resultados

El resultado central es un hallazgo de generalización *out-of-distribution* (OOD) al tamaño de equipo. El tamaño mínimo de equipo cooperativo necesario para «resolver» un escenario, definido como extinguir al menos el 80 % del incendio, crece de forma sostenida desde 4 drones en la dificultad más baja hasta 15 en la más alta, lo que ofrece una caracterización cuantitativa y nítida del requisito de cooperación (Figura 2). En consecuencia, la política resultante, evaluada a lo largo de varios episodios estadísticamente diferenciados y entrenada con equipos de 3 a 10 drones, resuelve equipos de hasta 15 sin necesidad de reentrenamiento o *fine-tuning*.

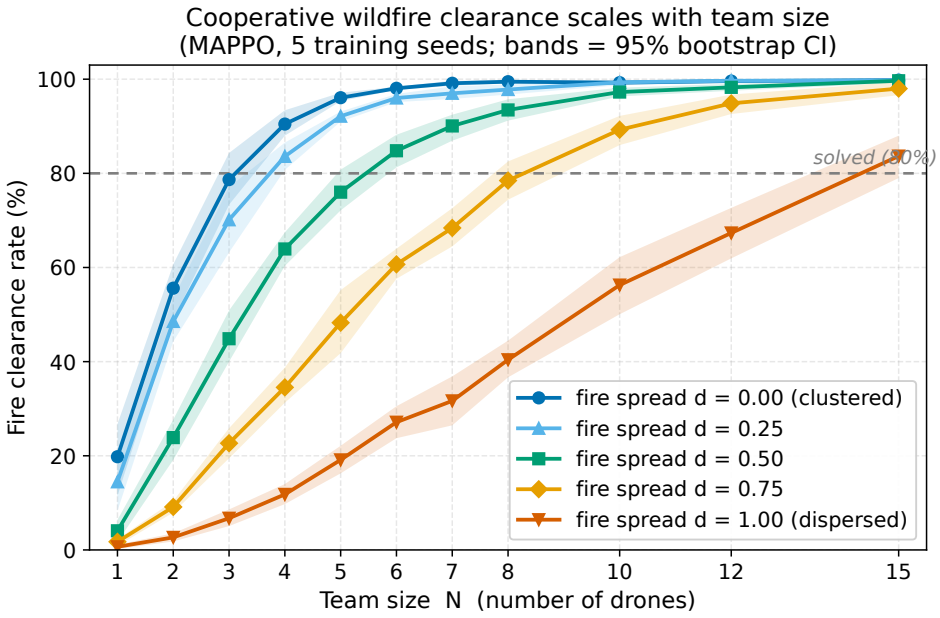

Figura 2: Porcentaje de extinción en función del tamaño de equipo N y de la dificultad d, promediado sobre cinco semillas de entrenamiento. Las bandas son intervalos de confianza *bootstrap* al 95 %. El N mínimo con el que el equipo extingue al menos el 80 % de los episodios crece de forma sostenida con la dificultad ($4 \rightarrow 15$).

Los cinco componentes de diseño se caracterizan por separado: cuatro de ellos, la *entity attention*, el *bias inductivo* húngaro, la LSTM y la recompensa de *team spirit*, mediante ablación componente a componente; el crítico centralizado, mediante la comparación con un aprendiz independiente (IPPO). De todos ellos, la *entity attention* resulta ser el componente individualmente más determinante. La comparación entre algoritmos muestra que, siguiendo la tendencia actual en muchas áreas de control y optimización, las políticas aprendidas con DRL mejoran de forma significativa a los controladores heurísticos más conocidos evaluados en este estudio preliminar (*greedy* y de asignación húngara), lo que aísla la aportación del *aprendizaje*

frente a la asignación óptima codificada a mano. En cambio, el aprendiz independiente (IPPO) iguala a la política con crítico centralizado en todas las métricas de umbral. Este resultado negativo es informativo, aunque conviene matizarlo: la ventaja de un crítico centralizado depende de cuánta información añada su visión global sobre lo que ya capta la observación local de cada agente, y cuando la observabilidad parcial no es muy severa ese margen es pequeño y puede quedar compensado por la distinta representación interna que aprenden el actor y el crítico. En este *benchmark*, por tanto, la coordinación parece provenir de la compartición de parámetros y de la percepción más que del entrenamiento centralizado. Por último, el análisis de robustez compara dos maneras de gastar un mismo presupuesto de entrenamiento. Presentar las dificultades de forma gradual, de la más fácil a la más difícil (un *curriculum*), concentra el entrenamiento en el régimen difícil y, en estas condiciones, consigue mejor extinción en él que sortear una dificultad al azar en cada episodio (*domain randomization*).

5 Conclusiones

De este trabajo se extraen tres conclusiones. La primera es que la combinación de mecanismos de atención con el esquema de entrenamiento centralizado y ejecución distribuida (CTDE) basta para hacer emerger dinámicas colectivas y cooperativas en un entorno complejo, lo que apunta a un potencial directo para problemas reales de coordinación de múltiples plataformas. La segunda es que disponer de un *benchmark* compartido como *FireCoopBench* es un primer paso valioso para que organismos públicos y privados desarrollen y comparen sus propios algoritmos sobre una base común. La tercera es que el entorno empleado es todavía una simplificación: trabaja sobre un plano discretizado y no incorpora aún sistemas realistas de guiado, navegación y control, ni una capa perceptiva con sensores de alta fidelidad. El trabajo futuro debe, por tanto, aumentar la complejidad y el realismo de las plataformas, del entorno, del control y de las perturbaciones. En esa línea, el paso natural siguiente es transferir la estrategia de coordinación validada a herramientas de simulación de mayor fidelidad y paradigmas de entrenamiento más realistas, en coherencia con la metodología incremental de «prototipar en 2D y transferir después» adoptada desde el inicio del proyecto.

6 Referencias

[1] IPCC, “Climate Change 2022: Impacts, Adaptation and Vulnerability – Chapter 13: Europe”, Intergovernmental Panel on Climate Change, inf. téc., 2022.

[2] C. Yu, A. Velu, E. Vinitsky y col., “The Surprising Effectiveness of PPO in Cooperative, Multi-Agent Games”, en *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2022. eprint: 2103.01955.

[3] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Póczos, R. Salakhutdinov y A. Smola, “Deep Sets”, en *NeurIPS*, 2017.

TOWARDS INTELLIGENT COORDINATION OF AUTONOMOUS PLATFORMS FOR WILDFIRE CONTROL THROUGH DEEP REINFORCEMENT LEARNING

Author: Pablo Tuñón Laguna
Director: Miguel Leiva Vélez

Abstract

Coordinating a fleet of aircraft to put out a wildfire is, at bottom, a complex optimisation problem: the collective control of several platforms and the management of a shared, finite resource under high dimensionality, partial observability of the fire front, and nonlinear, stochastic dynamics. It is also a multi-agent reinforcement learning problem for which there is still no standard testbed. We introduce *FireCoopBench*, a reproducible and parameterisable benchmark for cooperative suppression on a 22×38 gridworld with stochastic fire propagation, water logistics, and partial observability. The benchmark fixes the environment and the key performance indicators against which methods are compared, while leaving the policy network, the reward, the hyperparameters, and the learning algorithm open for the experimenter to modify; the reward we ship is a sensible default that can be rebalanced. On the cooperative environment we train a parameter-sharing Multi-Agent Proximal Policy Optimisation (MAPPO) policy, a policy-gradient method extended so that several agents learn at once, under the centralized-training/decentralized-execution scheme (the policy is trained centrally but executed in a distributed way, each drone acting on its own local observation). The policy adds a centralized critic, entity-attention observations, a Hungarian assignment carried in the observation, recurrence, and a team-spirit reward; each component is characterised in Chapter 6 (four by ablation, the centralized critic by the IPPO comparison). Averaged over training seeds, the policy solves the benchmark for teams of up to 15 drones with no retraining, and the minimum cooperating team size grows from 4 to 15 as difficulty increases. We compare against independent learners and against the best-known classical controllers evaluated in this preliminary study, and we analyse robustness.

Keywords: Multi-agent reinforcement learning; Cooperative wildfire suppression; MAPPO; Reproducible benchmark; Variable-team-size generalisation; Centralized training, decentralized execution; Domain randomisation.

Executive Summary

1 Introduction

Wildfires are a costly and dangerous natural hazard whose frequency and severity are projected to rise under continued climate change [1], and suppression itself is expensive and risky for the crews involved. Autonomous aerial platforms (fixed- and rotary-wing UAVs equipped to

drop suppressant) offer a way to scale the response, but they require coordination strategies that can be trained, validated, and reproduced. Seen through the lens of control, the task is a complex optimisation problem: the collective control of several platforms sharing a finite resource, under high dimensionality, partial observability of the fire front, and nonlinear, stochastic propagation dynamics. Reinforcement learning has a decades-long history in control and optimisation, yet only its pairing with deep neural networks made problems of this kind tractable, as the progression from Atari and Go to StarCraft and real-time tokamak plasma control illustrates. Cooperative wildfire suppression sits in that same family, and that is what makes the question timely. It is the domain of multi-agent reinforcement learning (MARL), and the setting of this thesis.

Across the wildfire MARL literature, however, a methodological gap persists: there is no parameterised, reproducible benchmark on which coordination strategies can be compared rigorously. This thesis closes that gap.

2 Objectives

The main objective is to design, implement, and study a reproducible multi-agent benchmark for cooperative wildfire suppression, together with a strong cooperative policy whose components have each been validated independently. The central hypothesis is that a single policy, trained on a variable number of agents and equipped with a centralized critic and the right structural invariances, can learn a coordination strategy that generalises to team sizes never seen during training. The work pursues four secondary objectives, mirroring the project plan (Annex B): (i) implement a modular wildfire-simulation environment that is parameterisable and reproducible, with a difficulty curriculum; (ii) build an end-to-end DRL/MARL training and validation pipeline; (iii) evaluate multi-agent coordination strategies under partial observability and compare the DRL/MARL algorithms and the best-known classical controllers that were within reach for this preliminary study; and (iv) improve robustness and generalisation through domain randomisation and an adversarial-fire setup.

3 Description of the Model/System/Tool

The benchmark, *FireCoopBench*, is structured as a three-rung difficulty ladder sharing a common interface, fire model, and reward structure. The cooperative environment is a 22×38 grid on which 3 to 10 drones must jointly suppress a stochastically spreading wildfire while managing a finite water tank that must be refilled at fixed locations. The benchmark is partially observable: each drone sees only a local, entity-structured slice of the world (its own state, the k nearest fires as a set, and slots for the other drones).

On the cooperative environment we train a parameter-sharing Multi-Agent Proximal Policy Optimisation (MAPPO) policy [2] under the centralized-training/decentralized-execution paradigm: the policy is trained centrally, with access to global information, but is executed in

a distributed fashion, each drone acting only on its own local observation. The policy carries five architectural components, each motivated separately and each characterised in Chapter 6 (four by ablation, the centralized critic by the IPPO comparison): a *centralized critic* that consumes a global-state slice during training only; an *entity-attention* observation encoder [3] that treats the nearest fires and the teammates as permutation-invariant sets; a *Hungarian assignment-in-observation* that gives the policy an explicit role-allocation prior; an *LSTM* for short-horizon memory under partial observability; and a constant *team-spirit reward* that mixes per-agent and team objectives. The reward we use is a deliberate default, a starting baseline that the benchmark leaves free to rebalance rather than a fixed part of the problem; training also uses a difficulty curriculum and a counterfactual, COMA-style advantage. The same policy network is used for any team size, which is what makes the variable-N evaluation possible.

4 Results

The central result is a *variable-team-size, out-of-distribution generalisation* finding. The minimum cooperating team size required to "solve" a scenario (clearance $\geq 80\%$) grows monotonically from 4 at the easiest difficulty to 15 at the hardest, which quantifies the cooperation requirement directly (Figure 1). The seed-averaged policy, trained on teams of 3–10 drones, then solves the benchmark for teams of up to 15 drones with no retraining.

The five architectural choices are characterised separately. Four of them (entity attention, the Hungarian prior, the LSTM, and the team-spirit reward) are studied by single-component ablation, and the centralized critic by comparison against an independent learner (IPPO); entity attention turns out to be the most load-bearing of the five. In the algorithm comparison, every learned policy clears far more fire than the best-known classical controllers evaluated in this preliminary study (a greedy controller and a Hungarian-assignment controller), which isolates what learning adds on top of an optimal assignment. The independent-learner baseline (IPPO), by contrast, *matches* the centralized-critic policy on every solved-threshold metric. We read this as an informative negative result, with one qualification: how much a centralized critic helps depends on how much its global view adds over what each agent already sees locally, and when partial observability is mild that margin is small and can be offset by the differing internal representations that the actor and the critic learn. On this benchmark, and within the budget of this study, centralized training brought no measurable benefit, and the coordination seems to come from parameter sharing and perception rather than from centralized credit assignment. A robustness analysis compares two ways of spending the same training budget. Introducing the difficulties gradually, from easiest to hardest (a curriculum), spends more of training on the hard regime and, under these settings, clears it better than drawing a difficulty at random in every episode (domain randomisation).

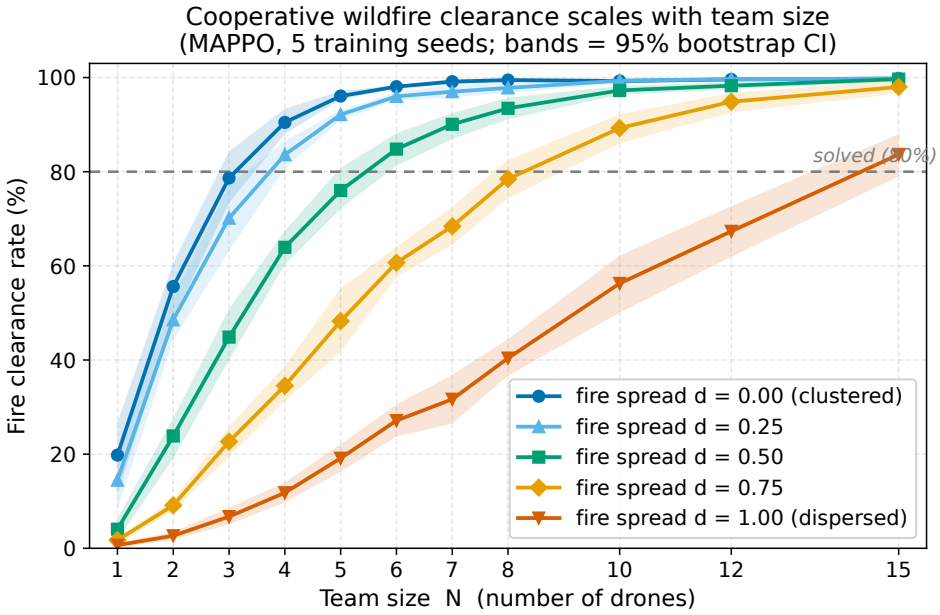

Figure 1: Clearance rate as a function of team size N and difficulty d, averaged over five training seeds. Bands are 95% bootstrap confidence intervals. The minimum N at which the team clears at least 80% of episodes grows monotonically with difficulty ($4 \rightarrow 15$).

5 Conclusions

These results turn an isolated wildfire-MARL result into a reproducible benchmark on which methods can be compared, accompanied by an architecture whose components are characterised in Chapter 6 (four by ablation, the centralized critic by the IPPO comparison) and by a clear characterisation of the cooperation requirement as a function of team size and difficulty. That the seed-averaged policy also generalises to team sizes never seen in training is best read as a welcome bonus rather than the central claim: the benchmark and the component analysis are the contributions that should outlast it. The limitations are stated plainly: the benchmark is a two-dimensional gridworld and does not yet incorporate realistic guidance, navigation and control systems, or a perception layer built on high-fidelity sensors. The natural next step (transferring the validated coordination strategy to higher-fidelity simulation tools and training paradigms) is left as future work, in keeping with the incremental "prototype in 2D, then transfer" methodology adopted from the start of the project.

6 References

[1] IPCC, "Climate change 2022: Impacts, adaptation and vulnerability – chapter 13: Europe", Intergovernmental Panel on Climate Change, Tech. Rep., 2022.

[2] C. Yu, A. Velu, E. Vinitsky, *et al.*, "The surprising effectiveness of PPO in cooperative, multi-agent games", in *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2022. eprint: 2103.01955.

[3] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Póczos, R. Salakhutdinov, and A. Smola, "Deep sets", in *NeurIPS*, 2017.

Contents

List of Figures

List of Tables

Chapter 1 Introduction

A wildfire never presents itself as a single, well-posed problem. It can start as a flare-up barely fifty metres across and still threaten lives and property within the hour, and it can grow into a moving front several kilometres long that splits and recombines as wind and terrain dictate. At any scale it appears as a crew exposed on a ridge whose only safe corridor may close within minutes, and as an air tanker overhead that must decide, in a glance, which flank to soak first, knowing another drop will not be ready for half an hour once the tank runs dry. A useful response must work whether the fire is fifty metres or fifty kilometres wide. The people and aircraft tasked with the response see only fragments of this scene at any instant (a screen of smoke here, a flare-up there, a radio report from a sector they cannot watch directly), and yet they must act together, immediately, with little opportunity to pause and re-plan once a flank shifts. This is the everyday reality of wildfire suppression, and it is becoming both more common and more severe.

The trend is not speculative. Climate projections indicate that the frequency and intensity of wildfires will rise across most regions over the coming decades, lengthening fire seasons and pushing fires into terrain and weather once considered marginal [1]. The fire seasons of recent years in southern Europe, Canada, and Australia have already made the human and economic stakes plain, and have exposed the operational ceiling of the current response: large, fast-moving fires, rapidly shifting weather, degraded communications, and only a partial view of where the fire actually is. The binding constraint is increasingly not the willingness to act but the ability to coordinate many suppression assets, in real time, with incomplete information. That is precisely the kind of problem this thesis is about.

Stated in general terms, the challenge is to reach high levels of optimality in a world that is high-dimensional, stochastic, partially observed, and changing under the agents' own actions, while the consequences of an action may surface only many steps later. That last point is a fourth difficulty in its own right: long-horizon decision-making, in which credit for a good or bad outcome must be assigned back across a long chain of decisions. Cooperation sits on both sides of this picture. Having many platforms act on incomplete local views is part of what makes the problem hard, yet a team that coordinates well is also the most promising way to cover a large, fast-moving front. Posed this way, the problem is one of the oldest and hardest goals in automatic decision-making, and until recently it lay stubbornly out of reach.

Why now: from a decades-old idea to a usable tool. The mathematical machinery for "learning to act well over time" is not new. Reinforcement learning, the idea of an agent improving its behaviour by trial and error against a long-run reward, grew out of optimal control, operations research, and dynamic programming, and has decades of history behind it. For most of that history it was confined to small, low-dimensional, fully observed problems. It could not ingest raw high-dimensional inputs, it was brittle under heavy stochasticity, and it had no principled way to act when the agent could see only part of its world. What changed in the last decade was the pairing of reinforcement learning with *deep neural networks*, which learn their own internal representations directly from raw data rather than relying on

hand-engineered features. This pairing is what finally let the approach scale to the properties that had previously kept it out of problems like the one above: high dimensionality, heavy stochasticity, partial observability, and the long horizons over which credit must be assigned.

A few landmarks trace the progression from toy settings toward the kind of regime wildfire suppression inhabits. A single network learned to play dozens of Atari games directly from screen pixels, reaching human-level scores without game-specific engineering [2]. Deep value and policy networks coupled with tree search then mastered the game of Go, whose search space had long been considered intractable for classical methods, first by defeating a human professional and then by surpassing every prior system while learning entirely from self-play [3], [4]. The same family of methods reached grandmaster level at *StarCraft II*, a real-time game with an enormous action space, long horizons, and genuine partial observability of the opponent [5], and sustained coordinated five-versus-five team play in *Dota 2*, where several agents must cooperate over extended horizons [6]. For an engineering audience the most relevant step is that deep reinforcement learning has since moved out of games and into safety-critical physical control: a learned controller now regulates the magnetic field of a tokamak fusion reactor in real time, shaping and stabilising the plasma where hand-tuned controllers had struggled [7]. Running through all of these is the same thread. Neural networks made high-dimensional, stochastic, partially observed control tractable, and, for a team of agents, multi-agent control as well. This is why credible autonomous coordination of many platforms has become a realistic goal only in the last few years, and why the question this thesis asks is now worth asking.

The specific challenge: cooperation under partial observability. Wildfire suppression by a team of autonomous aerial platforms (fixed- and rotary-wing UAVs able to drop water or suppressant) sits squarely in this newly reachable regime, and adds the difficulties peculiar to having many agents. Each platform sees only its own neighbourhood of a spreading fire, carries a finite water tank and must periodically leave the front to refill, and must choose actions that are good for the team rather than merely for itself. The standard way to train such teams is *centralized training with decentralized execution* (CTDE): a centralized critic or mixing network may consult global information while learning, but each deployed policy then acts on its local observation alone. A widely used approach is multi-agent PPO (MAPPO) [8], a policy-gradient method extended to teams (the full method family is reviewed in Chapter 2), which recent large-scale benchmarking reports among the more consistent performers on fully cooperative tasks [9].

The gap. Despite real progress, the wildfire-suppression sub-field has not yet adopted the evaluation discipline that adjacent cooperative-MARL sub-fields now take for granted. Existing studies typically report a single task instance, solved by a single algorithm, without component ablations, without confidence intervals over training seeds, and without any account of how performance scales as the team grows. There is, in short, no parameterised, reproducible benchmark on which coordination strategies can be compared rigorously. The

general-purpose MARL benchmarking wave of the last two years [9] has shown what such a benchmark should provide; this thesis carries that machinery over to wildfire suppression.

Hypothesis. The scientific question that organises the thesis can be stated as a single falsifiable hypothesis:

> *A single parameter-shared policy, trained on a variable number of agents and equipped with the right structural invariances, can learn complex emergent behaviour and high levels of coordination, also generalising to team sizes never seen during training.*

If true, this would mean that team coordination need not be re-learned for every fleet size, which would be of clear operational value. Everything that follows is designed to test this hypothesis and to identify *which* ingredients are responsible when it holds.

This thesis. The core goal of this thesis is to find a genuinely good neural architecture for cooperative wildfire suppression, and, in the process of building and testing it, to put together a benchmark that can be released publicly so that the community can design and compare methods for this pressing problem on common ground. We call that benchmark *FireCoopBench*: a three-rung difficulty ladder that culminates in cooperative suppression on a 22×38 grid with 3–10 drones (Chapter 5). On the cooperative environment we train a parameter-sharing MAPPO policy equipped with a centralized critic, an entity-attention observation encoder, a Hungarian assignment carried in the observation, an LSTM for short-horizon memory, and a constant team-spirit reward. The components of this architecture are evaluated in depth in Chapter 6; the role of each of them is taken up there, and the point here is only that the hypothesis of the previous paragraph is the question the experiments are built to answer.

Contributions. The contributions of the thesis are:

- A reproducible wildfire-suppression benchmark, *FireCoopBench*, with a difficulty curriculum and a formal "solved" evaluation protocol, designed for open-source release with pinned dependencies and fixed seeds.
- A cooperative neural architecture for the benchmark, evaluated in depth (parameter-shared MAPPO with a centralized critic, an entity-attention encoder, a Hungarian assignment carried in the observation, an LSTM, and a high constant team-spirit reward), which across generalisation and robustness evaluation campaigns, and against the best-known baselines examined in this study, shows clear improvements in coordination and in the range of team sizes it can handle. Which of its components matters most is studied in the ablation of Section 6.5.

Scope and intended path. The 2D benchmark presented here is the first stage of a deliberately incremental "prototype-in-2D-then-transfer" methodology: a fast, reproducible gridworld in which cooperative-MARL ideas can be validated rigorously before being carried over to more complex, higher-fidelity simulation tools and training paradigms with continuous action. That transfer is explicit future work (Chapter 8).

Document outline. The remainder of this thesis is organised as follows.

- Chapter 2: foundations of deep reinforcement learning, partial observability, and multi-agent RL.
- Chapter 3: state of the art on wildfire modelling and cooperative MARL, and the specific gap this thesis addresses.
- Chapter 4: motivation, objectives, and alignment with the United Nations Sustainable Development Goals.
- Chapter 5: the benchmark, the cooperative architecture, and the training procedure.
- Chapter 6: experimental results.
- Chapter 7: a qualitative reading of the learned behaviour.
- Chapter 8: limitations and future work.

Chapter 2 Theoretical Background

This chapter shares the theoretical foundations on which the proposed methodology is built. Each named ingredient of the principal policy is introduced at the level of *what it is and why it exists*; the precise way they are wired together is deferred to Chapter 5. Throughout, the convention is to give the plain-language intuition first and the equation second, and to keep the mathematics to the minimum that makes a claim falsifiable rather than merely plausible.

2.1 Reinforcement learning and Markov decision processes

Reinforcement learning (RL) formalises the problem of an agent that learns to act well over time by trial and error, using only a scalar *reward* signal as feedback rather than labelled examples of the correct action. The agent and the world it inhabits are modelled together as a *Markov decision process* (MDP), the standard object of sequential decision-making under uncertainty [10]. An MDP is the tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, whose components have direct physical readings in our setting:

- $\mathcal{S}$ is the set of *states*, that is, a complete description of the world at one instant (in our domain: where every fire, drone, building and water cell is, and how much water each drone carries);
- $\mathcal{A}$ is the set of *actions* available to the agent (move, extinguish, wait);
- $P(s' \mid s, a)$ is the *transition function*, the probability of landing in state s' after taking action a in state s. This is where the stochastic fire spread lives;
- $R(s, a)$ is the *reward*, the immediate scalar feedback for acting;
- $\gamma \in [0, 1)$ is the *discount factor*, which expresses how much a unit of reward earned now is preferred to the same unit earned later.

The defining assumption is the *Markov property*: the next state depends only on the current state and action, not on the full history of how the agent got there. The current state is therefore a sufficient statistic for predicting the future. (Section 2.2 relaxes exactly this assumption, which is what makes our problem hard.)

The agent's behaviour is a *policy* $\pi(a \mid s)$, a (generally stochastic) mapping from states to a distribution over actions. The agent does not optimise the immediate reward but the *return*, the discounted sum of all future rewards from time t onward:

$$G_t \;=\; \sum_{k=0}^{\infty} \gamma^k \, R_{t+k}. \tag{2.1}$$

The discount $\gamma < 1$ keeps this sum finite and encodes a preference for sooner rather than later: extinguishing a fire now is worth more than promising to extinguish it in a hundred

steps. The objective of reinforcement learning is to find a policy that maximises the *expected* return from the start,

$$J(\pi) \;=\; \mathbb{E}_{\pi}[\,G_0\,] \;=\; \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R_t\right], \tag{2.2}$$

where the expectation is taken over the randomness in both the policy and the environment's transitions.

Two derived quantities organise almost every RL algorithm. The *state-value function* $V^{\pi}(s) = \mathbb{E}_{\pi}[G_t \mid S_t = s]$ is the expected return of following π from state s; it answers "how good is it to be here?". The *action-value function* $Q^{\pi}(s,a) = \mathbb{E}_{\pi}[G_t \mid S_t = s, A_t = a]$ answers the finer question "how good is it to take action a here, and follow π thereafter?". Their difference,

$$A^{\pi}(s,a) \;=\; Q^{\pi}(s,a) \;-\; V^{\pi}(s), \tag{2.3}$$

is the *advantage*: how much better (or worse) a particular action is than the policy's average behaviour in that state. The advantage is the precise signal a policy-gradient method needs (Section 2.3), and a multi-agent refinement of it is what counterfactual credit assignment supplies (Section 2.9).

What kept this framework confined to small, tabular problems for decades was the need to represent π, V or Q explicitly for every state. The turning point was the use of *deep neural networks* as function approximators that learn their own features directly from raw, high-dimensional input: a single network trained end-to-end reached human-level play on dozens of Atari games (29 of 49 tested) from screen pixels alone, with no game-specific engineering [2]. That result is the reason "reinforcement learning" and "deep reinforcement learning" are used almost interchangeably in this thesis: every method we employ represents its policy and value functions as neural networks.

2.2 Partial observability: POMDPs and Dec-POMDPs

The MDP of Section 2.1 assumes the agent sees the full state. A drone over a wildfire does not: it perceives only its own neighbourhood (a local window of fire, a few nearby teammates, its own tank level) and never the whole front at once. This is *partial observability*, and it is modelled by a *partially observable MDP* (POMDP): the agent no longer receives the state s but an *observation* o drawn from an observation function $O(o \mid s)$ that reveals only a fragment of s. The damaging consequence is that the Markov property is lost at the level of observations: the same observation can correspond to many different true states with different correct actions, so a policy that maps the current observation to an action (a *memoryless* policy) is provably insufficient in general. The standard remedy is to let the policy depend on the *history* of past observations and actions, or on a learned summary of it; this is precisely the role of recurrence (Section 2.8).

When several agents act together, the right object is the *decentralised POMDP* (Dec-POMDP) [11]. A Dec-POMDP extends the POMDP to a team of N agents that share a single team reward but each receive their *own* private observation and choose their *own*

action, with no ability to see the others' observations or to communicate the full picture at execution time. Formally it adds a joint action space $\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_N$, a joint observation space, and a *shared* scalar reward R that depends on the joint action. Two features of this model are decisive for everything that follows. First, the reward is a property of the *team*, not of any individual: a drone is rewarded for the fire the team puts out, not merely the fire it personally touches. This is what makes the problem genuinely cooperative rather than a set of independent control problems, and it is the formal home of the team-spirit reward (Section 2.5). Second, solving a Dec-POMDP optimally is known to be intractable in the worst case (NEXP-complete: the cost grows faster than any polynomial in the problem size), which is exactly why the field reaches for approximate, learning-based methods rather than exact planners. Our benchmark is, in these terms, a finite-horizon Dec-POMDP with a parameterised stochastic transition function.

2.3 Value-based and policy-gradient methods; actor–critic

Two broad families of algorithms optimise the objective in Equation (2.2). *Value-based* methods learn an action-value function Q and act greedily (or near-greedily) with respect to it; the policy is implicit, read off as "take the action with the highest Q". The deep Q-network of [2] is the canonical example. Value-based methods are sample-efficient and natural for discrete actions, but acting greedily requires maximising over the action set, and they can be awkward when one wants a directly stochastic, easily differentiable policy, both of which matter in the multi-agent setting.

Policy-gradient methods take the complementary route: they represent the policy directly as a parameterised, differentiable function $\pi_\theta(a \mid s)$ and improve it by gradient ascent on $J(\pi_\theta)$. The foundational result that makes this possible is the *policy-gradient theorem*, which gives an estimator of $\nabla_\theta J$ that does not require differentiating through the unknown environment dynamics. In its advantage form it reads, in one line,

$$\nabla_\theta J(\pi_\theta) \;=\; \mathbb{E}_{\pi_\theta}\big[\, \nabla_\theta \log \pi_\theta(a \mid s)\; A^{\pi_\theta}(s,a) \,\big]. \tag{2.4}$$

The intuition is simple: nudge the parameters so as to make actions that turned out better than average (positive advantage) more probable, and actions that turned out worse less probable, with the magnitude of the nudge scaled by how much better or worse. The earliest practical instance, REINFORCE [12], used the raw return G_t in place of the advantage; the advantage form reduces the variance of the estimator dramatically, because subtracting the state-value baseline $V^\pi(s)$, which does not change the expectation in Equation (2.4), removes the part of the signal that is common to all actions in a state.

This last observation motivates the *actor–critic* architecture, which underlies every method this thesis trains. An *actor* is the policy π_θ; a *critic* is a learned estimate of a value function (here V) used to compute the advantage that scales the actor's updates. The two are trained jointly: the critic learns to predict returns, the actor uses the critic's advantage estimate to improve, and the loop repeats. Algorithm 1 states this generic loop; the specific methods of the next subsections are refinements of it, not departures from it.

Algorithm 1 Generic on-policy actor–critic loop

1: Initialise policy parameters θ (actor) and value parameters ϕ (critic)
2: **repeat**
3: Run policy π_θ in the environment to collect a batch of transitions (s_t, a_t, r_t, s_{t+1})
4: Estimate the advantage $\hat{A}_t$ for each transition using the critic V_ϕ ▷ e.g. via GAE, Eq. (2.6)
5: Update the actor: ascend the policy-gradient estimate of Eq. (2.4) using $\hat{A}_t$ ▷ PPO clips this step, Eq. (2.5)
6: Update the critic: descend the squared error between $V_\phi(s_t)$ and the observed return target
7: **until** converged

2.4 Proximal Policy Optimisation and advantage estimation

Plain policy gradients are notoriously unstable: a single large update can move the policy so far that the data just collected no longer describes it, and performance collapses. *Proximal Policy Optimisation* (PPO) [13] is the now-standard fix, and the algorithm on which both our principal policy and its baselines are built. Its idea is to improve the policy while explicitly forbidding it from changing too much in any one update. Let $\rho_t(\theta) = \pi_\theta(a_t \mid s_t) \,/\, \pi_{\theta_{\text{old}}}(a_t \mid s_t)$ be the probability ratio between the new and the old policy for an action that was actually taken. PPO maximises the *clipped surrogate objective*

$$L^{\text{CLIP}}(\theta) \;=\; \mathbb{E}_t\Big[\, \min\Big(\rho_t(\theta)\,\hat{A}_t,\; \text{clip}\big(\rho_t(\theta),\, 1-\epsilon,\, 1+\epsilon\big)\,\hat{A}_t\Big)\Big], \tag{2.5}$$

where $\hat{A}_t$ is the estimated advantage and ϵ (typically 0.1–0.2) is the clip width. The mechanism is exactly the cautious one just described: when an action was good ($\hat{A}_t > 0$) the objective stops rewarding further increases in its probability once the ratio exceeds $1+\epsilon$; when it was bad ($\hat{A}_t < 0$) the objective stops once the ratio drops below $1-\epsilon$. The min makes the bound pessimistic, so the policy never takes a step large enough to be unjustified by the data in hand. PPO's appeal is that it delivers most of the stability of more elaborate trust-region methods with the simplicity of a first-order optimiser, which is why it scales to the long-horizon, many-agent regime this thesis works in.

PPO still needs the advantage estimate $\hat{A}_t$ that Equation (2.5) consumes. The standard choice is *generalised advantage estimation* (GAE) [14], which trades off bias against variance with a single parameter $\lambda \in [0, 1]$. Writing the one-step *temporal-difference residual* as $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$, the error between the critic's prediction and a one-step-better estimate, GAE sums those residuals with an exponentially decaying weight,

$$\hat{A}_t^{\text{GAE}(\gamma,\lambda)} \;=\; \sum_{l=0}^{\infty} (\gamma\lambda)^l\, \delta_{t+l}. \tag{2.6}$$

Setting $\lambda = 0$ recovers the low-variance, higher-bias one-step estimate δ_t; setting $\lambda = 1$ recovers the high-variance, unbiased Monte-Carlo estimate. Intermediate values ($\lambda \approx 0.95$ is

typical and is what we use) give the smooth bias–variance compromise that makes long-horizon training stable. GAE is the concrete realisation of step 4 of Algorithm 1.

2.5 Multi-agent RL: independent learners and CTDE

Moving from one agent to a team introduces a difficulty that has no single-agent analogue: *non-stationarity*. If every agent learns simultaneously, then from any one agent's point of view the environment, which now includes the other, changing agents, is a moving target, and the convergence guarantees that single-agent RL relies on no longer hold.

The simplest response is to ignore the problem: *independent learning*, in which each agent runs its own single-agent algorithm (here, PPO) treating the others as part of the environment. Independent PPO (IPPO) is exactly this, and it is the baseline against which we measure the value of anything more sophisticated. It is often surprisingly strong, and, as Chapter 6 reports, on our benchmark it is statistically indistinguishable from the centralized method, a deliberate and informative negative result.

The shared team reward of a Dec-POMDP also admits an intermediate framing of the reward each agent learns from, the *team-spirit reward*. Rather than training every agent purely on the global team outcome or purely on its own local outcome, each agent's reward is a blend of the two: a convex combination of the agent's individual contribution and the shared team reward, governed by a single team-spirit coefficient. The coefficient interpolates between a fully selfish objective at one extreme and a fully cooperative, shared-reward objective at the other, and can be set or scheduled to control how strongly each agent is asked to value the team's success over its own. (The specific coefficient and schedule used in this thesis are given in Chapter 5.)

The dominant alternative is *centralized training with decentralized execution* (CTDE). The key observation is that the non-stationarity is only a problem at *training* time, and at training time we are not constrained by what a deployed drone can see: a simulator can hand the learner global information that no individual agent will have in the field. CTDE exploits this by letting a *centralized critic* (used only to compute advantages during training) condition on the joint state or joint observations, so its value estimates are stationary with respect to the whole team, while each *actor* remains a function of its own local observation alone and can therefore be deployed independently. Three landmark CTDE algorithms span the design space and recur throughout this thesis:

- **MADDPG** [15] extends the deterministic actor–critic to multiple agents, giving each agent a centralized critic over all agents' observations and actions during training.
- **QMIX** [16] is a value-based CTDE method that factorises the team value as a monotone combination of per-agent values, so that each agent can recover its part of the team-optimal joint action by a local argmax (the Individual-Global-Max property), so decentralised execution is exact rather than approximate for the value-based case.
- **MAPPO** [8] is the multi-agent extension of PPO (Section 2.4): per-agent actors trained with the clipped objective of Equation (2.5), sharing a centralized critic. It is the

principal algorithm of this thesis, chosen for its stability and its strong, consistent performance on fully cooperative tasks.

Parameter sharing and variable team size. A practical device cuts across all three methods and is, for this thesis, the single most important structural choice: *parameter sharing*. Instead of training N separate networks, all agents share *one* actor and *one* critic; agents are distinguished only by their differing observations (and, optionally, an identity feature), not by separate weights. Parameter sharing has two consequences that the central hypothesis of this thesis depends on. First, the number of learnable parameters becomes *independent of the team size*, which makes training many-agent teams tractable and lets agents pool their experience. Second, and this is the property that makes a variable-N generalisation study even meaningful, a single shared network can be *instantiated for any number of agents at execution time*, because it maps one agent's observation to one agent's action and is simply called once per agent. A team of five and a team of fifteen run the *same* weights. The only remaining obstacle is that the network must accept a variable number of *other entities* (teammates, fires) in its input without its weight shapes depending on how many there are, which is exactly the problem the next subsection solves.

2.6 Permutation invariance and set encoders

A drone's observation contains its teammates and the nearby fires, but neither of these is an ordered list with a fixed length. There is no natural "first teammate" or "third fire"; relabelling them must not change the drone's decision. A representation that respects this is called *permutation invariant*: its output is unchanged when the inputs are reshuffled. Feeding such data into a network by flattening it into fixed positional slots violates this property: it forces the network to learn the same fact ("a fire two cells north is dangerous") separately for every slot a fire might occupy, and it breaks entirely when the number of fires exceeds the number of slots.

The clean solution is a *set encoder*. *Deep Sets* [17] proves that any permutation-invariant function of a set (over a countable domain, and, for real-valued elements, given a sufficiently large latent dimension) can be written in the form

$$f(\{x_1, \ldots, x_m\}) \;=\; \rho\Big(\textstyle\sum_{j=1}^{m} \phi(x_j)\Big), \tag{2.7}$$

where ϕ embeds each element independently and ρ acts on their *sum*, a pooling operation that is, by construction, indifferent to order and accepts any number m of elements. This single equation is why a set encoder both ignores ordering and handles a variable element count, the two properties parameter sharing needed.

Plain summation weights every element equally, which is wasteful when only a few fires are actually relevant to a given drone. *Attention* generalises the pooling so that the weights are *learned and content-dependent*: each element is scored for relevance and the summary is a weighted combination, emphasising the fires and teammates that matter for the current

decision. Attention was introduced as the core mechanism of the Transformer architecture [18]; here we use only its set-aggregation aspect, not the full sequence model. The result, an *entity-attention* encoder that consumes fires and teammates as unordered sets and produces a fixed-size, permutation-invariant summary, is the perception front-end of the policy of Chapter 5 (its ablation result is reported in Chapter 6).

2.7 The assignment problem and the Hungarian algorithm

Even with good perception, a cooperative team faces an allocation question: which drone should head for which fire? Sending every drone to the nearest hotspot is wasteful, since several drones may converge on one fire while another spreads unchecked. The clean abstraction is the *assignment problem*: given a cost c_{ij} for assigning agent i to target j (here, the travel distance), find the one-to-one matching of agents to targets that minimises the total cost. Brute force is factorial in the number of agents, but the *Hungarian algorithm* [19] solves it exactly in polynomial time ($\mathcal{O}(n^3)$).

This algorithm was chosen for two reasons. It is a strong and well-understood classical optimiser: it returns the cost-minimising matching exactly, with no tuning and a short, easy-to-implement procedure, which makes it a dependable reference point. It also gives this thesis a heuristic baseline of the kind the deep reinforcement learning pipeline is meant to be measured against: a purely greedy, one-step assignment that the learned policy must improve on if learning is to be worth its cost. The matching the algorithm produces is optimal only for that single step, not over the trajectory, so it is a demanding but not unbeatable yardstick. How the same assignment is also fed back into the observation as a coordination prior is described in Chapter 5.

2.8 Recurrence and LSTM memory

Section 2.2 established that under partial observability a memoryless policy is insufficient: the agent must condition on its history. A *recurrent neural network* (RNN) does this by carrying a hidden state from one timestep to the next, updating it as new observations arrive, so that the hidden state acts as a learned, compressed summary of everything relevant the agent has seen so far. Naive RNNs are difficult to train over long horizons because the gradient signal tends to vanish or explode as it is propagated back through many steps. The *long short-term memory* (LSTM) network [20] solves this with a gated memory cell: learned gates decide what to store, what to forget, and what to expose, which lets information persist over many timesteps without the gradient degrading. In our policy the LSTM sits between the entity-attention encoder and the actor/critic heads, giving each drone a short-term memory of the fire front it just saw and of its own travel–suppress–refill cycle, so it can act coherently despite seeing only a local slice of the world at any instant.

2.9 Counterfactual multi-agent credit assignment (COMA)

A team trained on a shared reward (Section 2.5) inherits a *credit-assignment* problem: when the team does well, which agent's actions deserve the credit? If every agent is updated by the same global advantage, an agent that loafed is rewarded as much as one that did the work, and the learning signal an individual receives gets noisier as the team grows: each agent's own contribution is an ever-smaller fraction of a fixed team reward. *Counterfactual multi-agent* (COMA) credit assignment [21] addresses this with a *counterfactual baseline*. Rather than asking "how well did the team do?", COMA asks the sharper, counterfactual question "how much better did the team do *because this agent took this action* rather than some default?", comparing the realised team value against the average over the agent's possible actions while holding its teammates' actions fixed. Subtracting this agent-specific baseline isolates each agent's marginal contribution, which keeps the per-agent advantage informative even for large teams. We use the COMA advantage in place of the plain per-agent advantage when training our centralized policy; like the centralized critic, it is a training-time device that leaves execution unchanged.

2.10 Curriculum learning and domain randomisation

A final pair of ideas concerns not the architecture but the *training schedule*, and they offer two competing answers to the same question: how should a policy be exposed to a range of task difficulties so that it ends up both competent and robust?

Curriculum learning [22] draws on the pedagogical intuition that hard skills are best learned by starting easy and increasing difficulty gradually. Applied to RL, the environment's difficulty is raised over the course of training (the exact schedule used in this thesis is given in Chapter 5), so the policy first masters suppression on gentle fires and only then confronts aggressive, multi-source ones. The benefit is that early learning happens where the reward signal is dense and success is achievable, avoiding the cold-start failure of facing the hardest task from scratch.

Domain randomisation [23] takes the opposite stance: rather than ordering difficulty, it randomises the task parameters on *every* episode, exposing the policy to the full range of conditions throughout training in the hope that a policy good on average across all of them transfers robustly to any particular one, including conditions outside the training range. It was introduced to bridge the gap between simulation and reality and is a standard robustness tool.

These two are genuine alternatives, and which one wins is an empirical question rather than a settled one. This thesis treats it as such and compares them head-to-head as training-time robustness strategies; the verdict, reported and defended in Chapter 6, is that on our benchmark the curriculum substantially outperforms domain randomisation, and we are careful there to state the comparison honestly rather than to assert curriculum superiority as a general law.

2.11 Summary

The principal policy of this thesis is, in the vocabulary just assembled, a *parameter-shared MAPPO* actor–critic (Sections 2.4, 2.5) trained under CTDE with a *centralized critic* (Section 2.5) and a *counterfactual* advantage (Section 2.9); its perception is an *entity-attention set encoder* (Section 2.6) feeding an *LSTM* that supplies the memory partial observability demands (Sections 2.2, 2.8); it is given an explicit coordination prior by a *Hungarian assignment* carried in the observation (Section 2.7); and it is trained on a shared team-spirit reward (Section 2.5) under a difficulty *curriculum* (Section 2.10). Every one of these choices is motivated above and tested by ablation in Chapter 6. The next chapter places these ideas in the context of prior work; Chapter 5 then specifies precisely how they are wired together into FireCoopBench and the policy trained on it.

Chapter 3 State of the Art

A word on scope before the review begins. What follows is a focused, reduced reading of the field rather than an exhaustive survey: a proper survey of either cooperative multi-agent reinforcement learning or wildfire modelling would be a year-long undertaking in its own right. The aim here is narrower and serves the thesis directly. The chapter gathers the works that proved most relevant to the problem at hand, traces how those ideas reached the wildfire domain, locates the gap they leave open, and motivates the solution this thesis attempts.

The chapter proceeds along four converging strands. Section 3.1 covers wildfire modelling and the simulators built on it. Section 3.2 surveys deep reinforcement learning across the wildfire-response pipeline (surveillance, mitigation, suppression, and detection). Section 3.3 reviews cooperative multi-agent RL as a body of methods and benchmarks rather than mechanisms. Section 3.4 does the same for the robustness and generalisation literature. Section 3.5 then positions the benchmark of this thesis against the standard cooperative-MARL environments with a compact comparison table, and Section 3.6 synthesises the gap the work fills.

3.1 Wildfire modelling and simulators

A wildfire is, at heart, a fluid-mechanics and combustion problem. The spread of the flame is governed by the coupling of reacting, turbulent flow over a fuel bed: the Navier–Stokes equations for the gas phase, the transport of heat and chemical species, and the chemistry of combustion itself, whose theoretical foundations are laid out by Liñán and Williams [24]. Solving that system at the scale of a real fire is extraordinarily expensive. Turbulence spans many orders of magnitude in length and time, the combustion chemistry is stiff, and the terrain and fuel are heterogeneous, so a direct numerical treatment is out of reach for the millions of fire realisations a learning agent must see. Every usable wildfire model is therefore an abstraction that trades physical completeness for tractability, and the level of that abstraction fixes what an agent can be trained and evaluated against.

A first step down from the full physics keeps an explicit, physically grounded spread rate while discarding the resolved flow. The semi-empirical formulation of Rothermel [25] predicts the rate of fire propagation from fuel load, moisture, slope, and wind, and it remains the kernel of operational fire-behaviour systems and of most physically grounded simulators. Further down, two rule-based abstractions dominate. Discrete cellular automata propagate a fire front through local neighbourhood rules parameterised by wind, slope, and fuel [26], trading physical exactness for cheap, repeatable, grid-native dynamics. The Drossel–Schwabl forest-fire model [27] abstracts further still, treating ignition and spread as a self-organised-critical stochastic process and thereby reproducing the heavy-tailed distribution of real fire sizes from a handful of rules. Seen from the physics downward, these traditions occupy a single fidelity-versus-tractability spectrum, and every wildfire simulator sits somewhere along it.

For reinforcement learning the binding requirement is not full physical fidelity but consis-

tent, parameterisable dynamics that permit massive episode generation, controllable difficulty, domain randomisation, and reproducible evaluation. This has driven a recent wave of RL-oriented wildfire simulators. SimFire together with its training harness SimHarness [28] exposes a configurable mitigation environment with reference dynamics and baselines, and the GPU-accelerated JaxWildfire [29] pushes throughput to the level modern on-policy training needs, vectorising fire spread so that millions of environment steps per second become feasible. On the multi-agent side, FireCommander [30] provides an interactive, probabilistic environment that couples perception with action for teams of heterogeneous agents. The benchmark of this thesis sits deliberately at the tractable, parameterised end of this spectrum: it adopts a stochastic propagation model in the cellular-automaton spirit, with difficulty exposed as a single curriculum scalar, precisely so that the generalisation and ablation study that is the point of the work can be run at scale. A more physically grounded variant, backed by Rothermel-style [25] spread-rate computations on the same grid, is a natural and explicitly scoped extension (future work), as is adversarially generated fire (Section 3.4).

3.2 Deep reinforcement learning for wildfire management

Deep RL has been applied across the full wildfire-response pipeline, and it is useful to organise the literature by the stage of that pipeline each work targets, because the coordination demands differ sharply from one stage to the next. Recent surveys of aerial robotics for wildfire confirm that the field has moved from single-vehicle automation toward multi-agent coordination [31].

Surveillance and tracking. The earliest mature use of learned controllers is persistent monitoring. Julian and Kochenderfer train decentralised deep-RL controllers for cooperative wildfire surveillance with fixed-wing autonomous aircraft [32], showing that a team can maintain coverage of an evolving fire boundary without centralized control at execution time. The task is information-gathering rather than intervention: the agents do not change the fire, only their estimate of it.

Mitigation. A second strand intervenes before or around a fire, cutting firebreaks or allocating fuel-reduction effort. Altamimi *et al.* formulate large-scale mitigation as a Markov decision process and apply deep RL for sequential decision-making under uncertainty [33], and the SimHarness line of work [28] is built precisely around this mitigation setting. Here the action space is about landscape modification, and the reward horizon is long.

Suppression. The strand closest to this thesis acts on the fire itself, with a team of agents. Haksar and Schwager train distributed deep-RL controllers for cooperative fire fighting with networks of aerial robots [34], an early demonstration that suppression can be learned rather than scripted. More recent work targets coordinated aerial suppression directly with multi-agent RL [31], and couples multi-agent deep Q-learning with heuristic guidance for swarm fire

detection and extinguishing [35]. This is the same hybrid of learned policy and algorithmic prior that the coordination design of this thesis adopts, though here the prior is the Hungarian assignment of Section 2.7 rather than a hand-tuned heuristic.

Detection and search-and-rescue. Partial observability becomes central when the fire, or a person near it, must first be found. Collignon *et al.* study multi-agent deep-RL UAV coordination for wildfire search-and-rescue explicitly under partial observability [36], the same Dec-POMDP regime (Section 2.2) this thesis operates in, applied to locating targets rather than suppressing fire.

Taken together, these works establish feasibility at every stage of the pipeline. But they share a methodological limitation that the closing synthesis (Section 3.6) makes precise: each typically reports a single task instance solved by a single method on its own bespoke environment, with the team size, the difficulty, and the evaluation protocol all fixed by that environment. There is no shared, parameterised benchmark on which coordination strategies for wildfire suppression can be compared on equal terms.

3.3 Cooperative multi-agent reinforcement learning

The methods this thesis builds on were introduced mechanistically in Chapter 2; here they are placed in their scholarly lineage. The CTDE paradigm crystallised around three landmark algorithms that between them span the design space. MADDPG [15] made CTDE concrete for continuous control by giving each agent a centralized critic, and was the first to handle mixed cooperative–competitive settings; it also contributed the multi-particle environments (MPE) that became an enduring testbed. QMIX [16] established the value-factorisation route, learning a team value as a monotone combination of per-agent values so that decentralised execution stays exact for the value-based case. MAPPO [8] carried PPO [13] into the multi-agent setting and showed, against the then-prevailing assumption that on-policy methods were too sample-inefficient for MARL, that a centralized critic atop PPO is a strong and stable cooperative learner. Counterfactual credit assignment (COMA) [21] is the fourth member of this family and the conceptual origin of the per-agent advantage used in this thesis. The mechanics of CTDE, value factorisation, and the actor–critic objective are given in Chapter 2 (Sections 2.5–2.9); the contribution here is to note that the field's consensus, reinforced by large-scale comparisons, places MAPPO among the most consistent performers on fully cooperative tasks [8], [9], [37]. That is the empirical reason it is the principal algorithm of this work.

A second, quieter line of work is what actually enables a variable-team-size study. Teammates and fires can be treated as unordered sets through a permutation-invariant aggregator, namely the Deep-Sets construction of Zaheer *et al.* [17], whose attention-weighted generalisation derives from the Transformer [18]. Such an aggregator lets a single parameter-shared network accept any number of entities and serve any team size. Chapter 2 (Sections 2.5, 2.6) details how these set encoders work; the point for the literature is that this property is widely available yet rarely exercised as an evaluation axis: most cooperative-MARL studies

fix the team size at train and test time, leaving the generalisation question this thesis asks unanswered.

The maturity of cooperative MARL is reflected less in any single algorithm than in its evaluation infrastructure. A succession of standardised benchmarks now anchors the field: SMAC and its harder successor SMACv2 [38]; Melting Pot [39]; Overcooked-AI [40]; the MPE particle environments [15]; and the level-based foraging and related cooperative tasks collected in the EPyMARL study [37], all supported by common interfaces such as PettingZoo [41]. Alongside them, a deliberate methodological literature has grown to insist that comparisons be made properly: Papoudakis *et al.* [37] re-evaluate the major algorithms under a single protocol and find that reported rankings often do not survive a controlled comparison; Gorsane *et al.* [42] propose a standardised evaluation protocol for cooperative MARL; and the extended benchmarking of [9] reaffirms which methods are robust across tasks. This thesis adopts that discipline deliberately, with a fixed evaluation protocol, the seed as the unit of analysis, component ablations, and reported uncertainty, and applies it to a domain where, as the next sections argue, it has been largely absent.

3.4 Robustness and generalisation in MARL

Two complementary routes to robustness recur in the literature, and this thesis engages both. The first is domain randomisation, introduced by Tobin *et al.* [23] to bridge the simulation-to-reality gap by randomising task parameters every episode; it has since become a default robustness tool wherever a policy must transfer to conditions it was not specifically trained on. The second is robust adversarial reinforcement learning (RARL), proposed by Pinto *et al.* [43], which trains a protagonist against a learned worst-case adversary that perturbs the environment, yielding policies hardened against the most damaging disturbances rather than the average one. Both mechanisms are explained in Chapter 2 (Section 2.10 for domain randomisation; the curriculum it is contrasted with is introduced there too).

What the broader literature rarely does, and what this thesis foregrounds, is to treat the size of the cooperating team as the axis of generalisation. The set-encoder machinery (Section 3.3) makes a policy nominally applicable to any team size, but applicability is not the same as competence: whether coordination learned with a handful of agents survives at teams larger than any seen in training is an empirical question the cooperative-MARL benchmarks above do not pose. This thesis poses it directly, and pairs it with a head-to-head comparison of a difficulty curriculum against per-episode domain randomisation as training-time robustness strategies. For the wildfire setting an adversary in the RARL sense has a natural reading: an ignition generator or an environment-parameter modulator that drives the fire toward worst-case configurations. That reading makes adversarially generated fire a well-defined next step (future work) rather than a vague aspiration.

3.5 Positioning against cooperative-MARL benchmarks

The standard cooperative-MARL benchmarks were each built to stress a particular facet of coordination, and none was designed to expose at once the combination of properties a study of team-size generalisation under a resource-logistics constraint requires. Table 1 makes this positioning explicit. The rows are the five properties that the central hypothesis of this thesis depends on; the columns are the most widely used cooperative benchmarks alongside the one introduced here.

Table 1: Positioning the benchmark of this thesis against standard cooperative MARL environments. ✓ present, ✗ absent, (∼) partial or configuration-dependent. "Variable team size at eval" means a policy is evaluated at team sizes *not seen during training*; "continuous difficulty knob" means a single scalar tunes task hardness; "resource/logistics constraint" means agents manage a consumable (here, a finite water tank with refill).

Property	SMACv2 [38]	Melting Pot [39]	Overcooked -AI [40]	MPE [15]	Level-based foraging [37]	**This thesis**
Variable team size at eval	(∼)	(∼)	✗	✗	(∼)	✓
Continuous difficulty knob	✗	✗	✗	✗	✗	✓
Partial observability	✓	✓	✗	(∼)	(∼)	✓
Resource / logistics constraint	✗	(∼)	✓	✗	(∼)	✓
Domain dynamics (propagating process)	✗	✗	✗	✗	✗	✓

A few entries need justifying, since each must be defensible. SMACv2 and Melting Pot can be configured with different population sizes, but neither is built to report performance as a function of evaluation team size relative to a fixed training size, hence (∼) rather than ✓. Overcooked-AI imposes a genuine logistics loop (ingredients must be fetched, prepared, and delivered in order), so it earns a ✓ on the resource constraint, but it is a fixed two-agent, fully observed layout and exposes no difficulty scalar. The MPE particle tasks and level-based foraging are deliberately minimal: foraging involves a cooperative pick-up constraint ((∼) on logistics) and supports varying agent and item counts ((∼) on team-size evaluation), but neither offers a continuous difficulty knob or an underlying propagating process. Only the benchmark of this thesis combines all five: a fire that spreads according to its own dynamics, a single continuous difficulty scalar, a hard water-and-refill logistics loop, partial observability by construction, and, most distinctively, an evaluation axis that pushes team size beyond the training range.

3.6 The gap addressed by this thesis

The four strands surveyed above converge on a single, specific gap. Cooperative MARL as a field is served by mature methods (Section 3.3), a robustness toolkit (Section 3.4), and an increasingly disciplined evaluation culture [9], [37], [42]. Wildfire DRL (Section 3.2) has demonstrated feasibility at every stage of the response pipeline. Yet no existing environment, as

Table 1 shows, exposes at once the three axes this thesis is built around (a continuous difficulty scalar, a variable-team-size evaluation axis that includes team sizes above the training range, and a resource-logistics constraint) set in a domain that carries its own propagation dynamics. And in the wildfire-suppression setting specifically, the deeper absence is methodological: there is no reproducible, parameterised benchmark at all on which coordination strategies can be compared with ablations, confidence intervals, and an explicit characterisation of how the cooperation requirement scales with team size and difficulty. The result is that wildfire-MARL claims do not yet compose, since each is reported on its own environment, with its own fixed team and its own protocol. The comparable cooperative-MARL sub-fields have, through benchmarks like those in Table 1, made their claims commensurable.

FireCoopBench and the study built on it are designed to close both gaps at once: the domain-specific one (a reproducible, parameterised wildfire-suppression benchmark) and the methodological one (the missing combination of evaluation axes, evaluated with the statistical discipline the cooperative-MARL literature now expects). Chapter 5 specifies the benchmark and the policy trained on it; Chapter 6 reports the variable-team-size generalisation study, the component ablation, the algorithm comparison, and the curriculum-versus-randomisation robustness analysis that together fill this gap.

Chapter 4 Motivation and Objectives

4.1 Motivation

The motivation for this thesis comes from three factors that reinforce one another.

Societal urgency. Beyond the rising hazard introduced in Chapter 1, the operational case is direct: suppression is expensive and the front-line workforce is exposed to genuinely life-threatening conditions. Tools that scale the response without increasing the human exposure are therefore of clear public value.

A scientific gap. The open problem that motivates this thesis is the large-scale, efficient coordination of many aerial suppression platforms under partial observability, a capability that remains out of reach for the methods currently used in the wildfire setting (the state of the art is reviewed in Chapter 3). The proposed solution is a learned coordination policy built on a neural architecture and a centralized-training paradigm designed for that problem. The benchmark is not the contribution but the laboratory: a controlled environment in which such control frameworks can be trained, compared, and refined together.

Industrial relevance. Multi-agent coordination of autonomous platforms under uncertainty is a core, *transferable* capability, not specific to fire, that matters for the high-stakes applications that the supervising organisation (Indra) and adjacent sectors (emergency response, civil protection, security and defence) work on. The simulation-only scope is set out in Chapter 1, and the path to higher-fidelity simulation is detailed as future work in Chapter 8.

4.2 Objectives

The main objective of the thesis is:

> *To design and study a strong cooperative policy for wildfire suppression, together with the training paradigm that produces it: a parameter-shared neural architecture trained under centralized training with decentralized execution using multi-agent PPO, whose architectural components are each independently validated, and whose generalisation across team size and across environment distributions is characterised quantitatively.*

This main objective is the demanding one: a working attention-based policy that coordinates a variable-size team is considerably harder to obtain than the environment it is trained on. It rests on three secondary objectives, which mirror the project plan in Annex B:

1. **Reproducible benchmark.** Implement a modular wildfire-simulation environment that is parameterisable and reproducible, with discrete and continuous variants and

an explicit difficulty curriculum, exposed through a Gymnasium / PettingZoo-style interface, and pair it with an evaluation protocol and a fixed set of key performance indicators so that coordination strategies can be compared on common ground.

2. **Coordination study.** Evaluate multi-agent coordination strategies under partial observability, comparing suitable DRL/MARL algorithms: the centralized-critic MAPPO policy presented here against independent learners and against hand-coded heuristic controllers.

3. **Robustness study.** Characterise robustness and generalisation through domain randomisation and out-of-distribution evaluation, with an adversarial-fire setup inspired by robust adversarial RL [43] left as future work.

These objectives shape the structure of the rest of the document: the policy, the training paradigm, and the benchmark are described in Chapter 5; the coordination study and robustness analysis are presented in Chapter 6.

4.3 SDG Alignment

The thesis aligns with several of the United Nations Sustainable Development Goals:

- **SDG 9 (Industry, Innovation and Infrastructure).** The work advances a reproducible DRL/MARL methodology with explicit industrial transfer potential; the released benchmark is intended as infrastructure on which others can build.

- **SDG 11 (Sustainable Cities and Communities).** Coordinated autonomous wildfire suppression directly supports risk reduction at the wildland-urban interface, a setting in which large fires impose the sharpest costs on communities.

- **SDG 13 (Climate Action).** The thesis contributes methodology toward improving the technological capacity for hazard response and adaptation in a climate-change-driven risk landscape.

- **SDG 15 (Life on Land).** Reducing wildfire impact directly benefits terrestrial biodiversity and ecosystem services, which are among the principal casualties of large wildfires.

Chapter 5 Methodology

Our aim is a fire-suppression agent that puts out a spreading wildfire as reliably and as cheaply as a fixed team of drones allows. Reaching it is not a single modelling step but a working loop, the same loop a practitioner follows when applying deep reinforcement learning to a control problem. We first build a simulated environment with the abstractions the task actually needs (a fire that spreads, drones that move and carry water, partial observation), designed so that it also serves the community as a reusable benchmark. On that environment we design a neural architecture, choose a training paradigm, run a training campaign, evaluate the resulting policy against well-defined metrics, and feed what we learn back into the design. The chapter is organised around that loop, and Chapter 6 reports the experiments in the same order, so that each result answers a question raised by the stage that produced it.

The two technical contributions of the thesis sit inside this loop: the benchmark itself (FireCoopBench, its state, dynamics, actions, observations, difficulty curriculum and "solved" protocol), and the cooperative architecture trained on it together with the baselines it is compared against.

Methodology summary. The chapter follows the six stages of the pipeline in order:

1. **Environment.** The simulated task and benchmark: grid, fire model, actions, water logistics, observations, and the difficulty curriculum (Sections 5.1–5.5).
2. **Neural architecture.** The policy and value networks that map an observation to an action (Section 5.2).
3. **Training paradigm.** The multi-agent learning algorithm, reward, and credit-assignment choices (Section 5.3).
4. **Training campaign.** How a policy is actually trained: the seeds, the step budget, and the schedules (Section 5.7).
5. **Evaluation.** The "solved" protocol, the metrics, and the questions each metric answers (Section 5.8).
6. **Iteration.** The baselines and ablations against which the policy is measured, which close the loop back onto the design (Section 5.9).

5.1 Stage 1: Environment: FireCoopBench

The first stage builds the world the agents act in. A good environment for this problem has to do two jobs at once: capture the abstractions that make wildfire suppression a genuine coordination task, and stay simple and reproducible enough that others can train and evaluate on it. The questions this stage has to answer are what the agents perceive, what they can do, and how the fire behaves; the following paragraphs treat each in turn.

Benchmark overview. FireCoopBench is structured as a ladder of three environments of increasing difficulty that share a common grid representation, fire model, and Gymnasium / PettingZoo-style interface. The first two are single-agent and serve as stepping-stones and sanity baselines: *escape* (the agent must avoid a spreading fire) and *suppression* (a single drone with a finite water tank must extinguish fires and protect buildings). The third, and the focus of this thesis, is the *cooperative* environment: a 22×38 grid on which a team of drones must jointly suppress a stochastically spreading wildfire. The team size N is sampled from $\{3, \ldots, 10\}$ during training; the network can physically accept up to $N_{\max} = 15$ drones, but it is trained only with 3–10, so evaluation at $N \in \{11, \ldots, 15\}$ is a genuine test of generalisation (*out-of-distribution* team sizes), and this is the construction that makes the variable-N generalisation evaluation possible.

State and fire dynamics. Each cell of the grid carries one of a small set of states: empty, building, water, burning, or burned. Fire ignites from an initial set of cells and spreads stochastically to four-connected neighbours each step with probability p_{spread}; burning buildings persist for a fixed duration before burning out, contributing an additional cost during the time they burn. Episodes run for up to $T_{\max} = 800$ steps. The benchmark ships several interchangeable fire models of increasing realism: an independent stochastic model (the default for the cooperative environment), a cellular-automaton model with neighbourhood-based propagation rules, and a multi-source variant in which fires can ignite simultaneously at several locations. A physically grounded model derived from Rothermel-style spread-rate computations is left to future work.

Actions and the water-logistics loop. Each drone has a discrete action space of six actions: the four cardinal moves, an *extinguish* action which suppresses fire in the four-adjacent cells when the drone carries water, and a no-op. Drones carry a finite tank (capacity 5) and automatically refill on or adjacent to a water cell. This logistics loop (travel, suppress, deplete, return to refill, repeat) is what makes the task cooperative rather than merely multi-agent: a single drone cannot exhaust a single fire, so a team must partition the fire front and stagger its refills instead of crowding the same hotspot.

Observations and partial observability. The benchmark is partially observable. Each drone receives a structured local observation comprising (i) its own features (position, water level, current assignment), (ii) the $k = 16$ nearest active-fire cells as a variable-length set with masking, and (iii) slots for the other drones (relative position, water level, assignment). This entity-structured observation is consumed by an attention encoder rather than flattened, so the policy reasons over fires and teammates as *sets* rather than over fixed positional slots. A separate global-state slice is exposed to the centralized critic during training only, and is never read at execution.

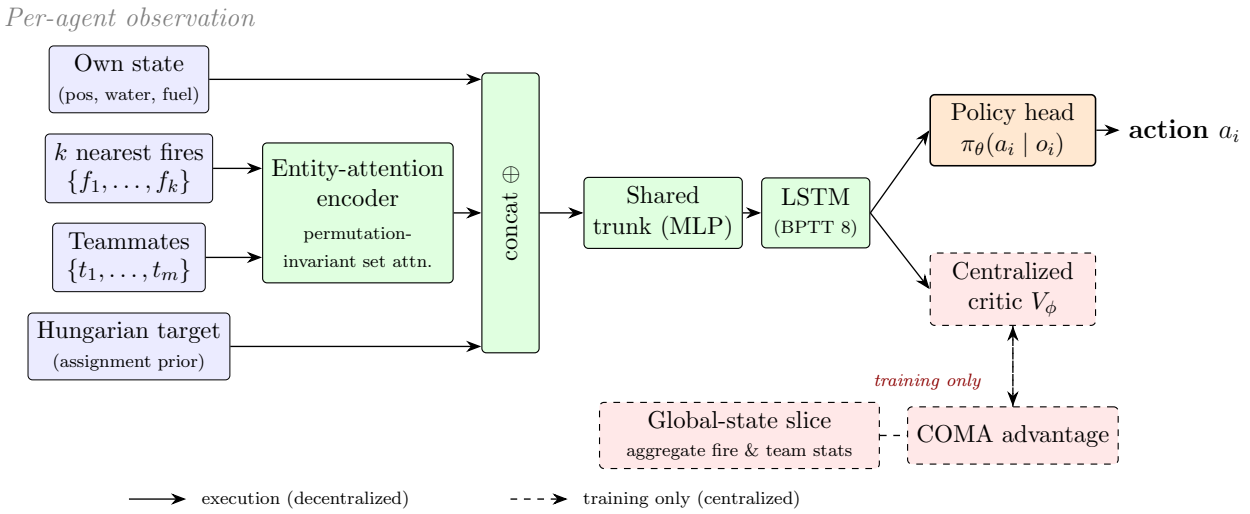

Figure 1: Cooperative policy architecture (centralized training, decentralized execution). Each agent encodes its entity-structured observation—own state, the k nearest fires and the teammates as permutation-invariant *sets*, and the Hungarian assignment prior—with a shared entity-attention encoder, an MLP trunk and an LSTM. At execution a policy head selects the action from the local observation alone (solid path). At training time only (dashed), a centralized critic additionally consumes a compact global-state slice and provides a COMA-style counterfactual advantage. Parameter sharing makes the network team-size-agnostic.

Difficulty curriculum. Task difficulty is controlled by a single scalar $d \in [0, 1]$ that interpolates the fire's aggressiveness between an easy and a hard regime: the spread probability p_{spread} rises up to 0.25 and the number of initial ignition sources up to 8 at $d = 1$. During training d is annealed from 0 to 1 on a curriculum, so the team meets the hard fire only once it can cope with it; at evaluation d is held fixed at chosen levels. The exact anneal schedule and the team-size sampling that runs alongside it are stated in Section 5.5.

5.2 Stage 2: Neural architecture

The second stage designs the network that turns an observation into an action. The question it answers is structural: what does a network need so that one set of weights can drive a team of any size against a fire of any size? Five design choices follow from that question, and each is put to the test in Chapter 6 (four by ablation, the centralised critic by the IPPO comparison).

The principal policy trained on the cooperative environment is a parameter-sharing Multi-Agent Proximal Policy Optimisation policy. Figure 1 shows the full data flow from the entity-structured observation to the action and the centralised critic; the five choices are described in turn below.

Parameter sharing with a centralized critic. All drones share a single policy network and a single value network, trained with multi-agent PPO [8], [13]. Parameter sharing keeps the parameter count independent of the team size and lets one trained network act for *any* number of drones. The value network is a *centralized critic*: during training only, it additionally consumes a compact global-state summary (aggregate fire and team statistics), following the centralized-training/decentralized-execution principle; at execution each drone acts from its local observation alone.

Entity-attention observation encoder. Rather than flattening the observation, the encoder treats the k nearest fires and the teammates as *sets* and aggregates them with permutation-invariant attention [17]. The agent's own features, the attended fire summary and the attended teammate summary are concatenated and passed through a shared trunk, which is what lets a single network handle a variable number of fires and teammates; its contribution is validated by ablation in Chapter 6.

Hungarian assignment-in-observation. To give the policy an explicit coordination prior, at each step we run the Hungarian algorithm (a classical method that computes the cheapest one-to-one matching of drones to fires) over a minimum-cost (Manhattan-distance) pairing of drones to active fires, and expose each drone's matched target as part of its observation. This per-step matching is optimal for that one-shot assignment but not over the trajectory. The assignment is only advisory, since the policy may ignore it, but it converts an implicit role-allocation problem into an explicit hint; its contribution is validated by ablation in Chapter 6.

Recurrence under partial observability. Because the environment is partially observable, each agent carries a recurrent (LSTM) network between its encoder and its policy/value heads, giving it a short-term memory of what it recently saw (the fire front and its own travel-suppress-refill cycle), so it can act sensibly despite seeing only a local slice of the map.

Counterfactual advantage. The advantage estimator uses a counterfactual credit-assignment technique (Counterfactual Multi-Agent, COMA [21]) that estimates how much this drone's action changed the team outcome, comparing against what would have happened had it acted differently while its teammates did the same. This keeps the learning signal from getting noisier as the team grows.

5.3 Stage 3: Training paradigm

With the environment fixed and the network defined, the third stage chooses how the network learns. Three coupled decisions make up the paradigm: what signal rewards the team, how that signal is shared among drones, and how credit for a shared outcome is assigned to an individual action. We treat the reward and its team-spirit mixing first, then the multi-agent learning algorithm and the centralised-critic choice that the architecture above was built for.

Team-spirit reward. A *team-spirit* reward blends each drone's own success with the whole team's, weighted by a coefficient τ ($\tau = 1$ means the drone is rewarded entirely for the team's outcome):

$$r_i \;=\; (1-\tau)\, r_i^{\text{loc}} \;+\; \tau\, r^{\text{team}}, \qquad \tau \equiv 1, \tag{5.1}$$

where each component combines a progress term (the reduction in active fire between two consecutive steps), a fire-pressure penalty, and a terminal bonus for clearing the map. We use a high constant τ: gradually shifting that weight during training (as in the OpenAI Five Dota-2 system) was tested and found to be indistinguishable from a constant τ on every metric we report, whereas removing the shared reward entirely ($\tau = 0$) degrades coordination markedly (Chapter 6). We therefore claim a constant team-spirit reward, not an annealing curriculum.

5.4 Reward decomposition and team-spirit mixing

The team-spirit blend of Section 5 (with the constant $\tau \equiv 1$) is built from three shaping terms. Written out, the team's per-step reward is their sum,

$$R_t \;=\; \underbrace{\alpha_{\text{term}}\, \mathbb{1}[\text{cleared}_t]}_{\text{terminal (sparse)}} \;+\; \underbrace{\alpha_{\text{prog}}\, \Delta F_t}_{\text{progress (dense)}} \;+\; \underbrace{\alpha_{\text{press}}\, F_t}_{\text{pressure (dense)}} \;=\; 200\,\mathbb{1}[\text{cleared}_t] \;+\; \Delta F_t \;-\; 0.15\, F_t, \tag{5.2}$$

where F_t is the number of active-fire cells at step t, $\Delta F_t = F_{t-1} - F_t \geq 0$ is the fire extinguished over the last step, and $\mathbb{1}[\text{cleared}_t]$ is paid once, when the map is cleared and the no-fire condition sustained. The three coefficients $(\alpha_{\text{term}}, \alpha_{\text{prog}}, \alpha_{\text{press}}) = (200,\ 1.0,\ -0.15)$ (Table 2) were frozen once on the principal seed (run 6) and never retuned; the whole five-seed result rests on this single, fixed triple. Each drone then receives this reward through the team-spirit blend of Equation (5.1), with R_t evaluated over the fire it itself influences for r_i^{loc} and over the whole map for r^{team}, so at the constant $\tau \equiv 1$ every drone is rewarded by the global per-step reward R_t directly. The remainder of this subsection explains why each term is shaped as it is, and why their relative magnitudes matter.

Table 2: The three frozen reward terms and their coefficients, the run 6 triple of Equation (5.2). The same coefficients are used whether a term is evaluated over a single drone's influence (r_i^{loc}) or over the whole map (r^{team}); the team-spirit weight τ then mixes the two (Equation (5.1)).

Term	Symbol	Coefficient	Shape
Terminal clear bonus	α_{term}	$+200$	sparse, episode end
Progress (fire reduction)	α_{prog}	$+1.0$	dense, per step
Fire-pressure penalty	α_{press}	-0.15	dense, per step

Terminal clear bonus ($\alpha_{\text{term}} = 200$). A large, sparse reward is paid only when the whole map is cleared and the no-fire condition is sustained (the "solved" event of Section 5). This is the true objective of the task and is deliberately made an order of magnitude larger than any single dense step reward, so that a team is never tempted to trade the final clearance for accumulated intermediate reward. On its own this signal is far too sparse to learn from: a random team almost never clears the hardest cell (Section 6.6), which is what motivates the two dense terms below.

Progress term ($\alpha_{\text{prog}} = 1.0$). We reward the *change* in remaining fire, not the level, so the policy is never paid to keep a fire alive. Formally this is potential-based reward shaping [44]: at every step the team receives the reduction in the number of active-fire cells, $\Delta F_t = F_{t-1} - F_t$, which is the change in the potential $-F_t$. This densifies the sparse terminal goal into a per-step signal that points monotonically towards it: any genuine suppression is rewarded immediately, and the cumulative progress reward over an episode that ends cleared telescopes to the total fire extinguished. Shaping with the change in a potential densifies learning without altering the set of optimal policies, because the added per-step terms telescope and cancel over any trajectory. We do not claim the strict invariance theorem holds verbatim under our finite horizon and the additional terms, but the progress term is motivated by that intuition: reward the change in how much fire remains, not the level, so the shaping cannot create a spurious optimum where the team is paid to keep fire alive.

Fire-pressure penalty ($\alpha_{\text{press}} = -0.15$). A small per-step penalty proportional to the current count of burning cells F_t encodes urgency: every step spent with fire on the map is costly, and the cost grows with the size of the front. This breaks the indifference that a pure progress term would leave between extinguishing a fire now and extinguishing it later, and it is what makes time-to-clear (Section 6.8) a quantity the policy actively minimises rather than merely a by-product. Its magnitude is kept deliberately small relative to the progress term (0.15 against 1.0) so that it shades the policy towards speed without ever rewarding the destructive shortcut of letting a fire burn out on its own, the failure mode that an over-large pressure penalty, or a penalty on the burned rather than the burning state, would invite. The sensitivity of the result to this balance is not hypothetical: the *reward-rebalance* ablation trains on the pre-rebalance reward coefficients, and its effect on hardest-cell clearance is quantified in Section 6.5.

Team-spirit mixing. Each drone i finally receives the convex blend $r_i = (1-\tau)\, r_i^{\text{loc}} + \tau\, r^{\text{team}}$, where r_i^{loc} applies the three terms of Table 2 to the fire that drone i itself influences and r^{team} applies them to the whole map. We use $\tau \equiv 1$, i.e. each drone is rewarded entirely for the team outcome. Conceptually this aligns every agent's gradient with the global objective and removes the incentive for a drone to "poach" a nearly-extinguished fire from a teammate purely to bank its local reward; empirically the choice is load-bearing (the selfish $\tau = 0$ variant loses coordination, Section 6.5) while its *schedule* is not (an OpenAI-Five-style anneal of τ is

indistinguishable from the constant, which is why we report a constant τ and not a curriculum over it).

5.5 Difficulty-curriculum schedule and team-size sampling

The "solved" protocol of Section 5 fixes how we *evaluate*; this subsection states the two stochastic schedules under which the principal policy is *trained*. Both are deliberately simple, so that each headline comparison turns on a single, clearly stated training choice: the curriculum against domain randomisation (Section 6.7), and the variable-N generalisation (Section 6.4).

Difficulty schedule. The scalar difficulty $d \in [0, 1]$ (which co-varies the spread probability up to $p_{\text{spread}} = 0.25$ and the ignition-source count up to 8 at $d = 1$) is *annealed* from 0 to 1 over the first part of training and then *held* at 1 for the remainder:

$$d(t) \;=\; \begin{cases} \min\!\big(1,\; t/t_{\text{anneal}}\big), & t \le t_{\text{anneal}}, \\ 1, & t > t_{\text{anneal}}, \end{cases} \tag{5.3}$$

where t counts environment steps and t_{anneal} marks the end of the warm-up phase. The team therefore begins on an easy fire it can clear by chance, and the difficulty rises only as fast as the policy can keep pace; once d reaches 1 the policy spends the majority of its 250 M-step budget at the hard regime, which is the regime the headline metrics report. We tested this staged exposure against the alternative: replacing it with per-episode domain randomisation (drawing d uniformly at each reset) dilutes the hard-end signal and collapses hardest-cell clearance (Section 6.7).

Team-size sampling. Orthogonally to the difficulty schedule, the team size is resampled *every episode*,

$$N \;\sim\; \text{Uniform}\{3, 4, \ldots, 10\}, \tag{5.4}$$

so that within a single training run the parameter-shared network sees the full range of in-distribution team sizes interleaved at every difficulty. The training range stops at 10 while the network can physically accept up to $N_{\max} = 15$; the gap $N \in \{11, \ldots, 15\}$ is never sampled during training and exists solely to make the out-of-distribution generalisation test of Section 6.4 a genuine one. The two schedules are independent: $d(t)$ depends only on the global step count and N only on the per-episode draw, so no team size is preferentially paired with any difficulty.

5.6 Observation tensor layout

Section 5 described, in prose, the entity-structured observation that the attention encoder of Figure 1 consumes; here we make its exact layout explicit, because the dimensions below are what the permutation-invariant pooling and the masking operate on. Each drone's observation is a tuple of three blocks rather than a single flat vector.

Self block (10-d). A fixed 10-dimensional vector summarising the drone's own state: its grid position, its water level relative to the tank capacity of 5, its current Hungarian-assigned target (Section 5), and the related bookkeeping features the policy needs to time its travel-suppress-refill cycle. Being fixed-length and always present, this block needs no masking.

Fire block ($k = 16$ entities, 6-d each). The $k = 16$ nearest active-fire cells, each described by a 6-dimensional feature (relative position, and local fire-state descriptors), presented as a *variable-length set*. When fewer than 16 fire cells are active the unused slots are zero-padded and a boolean mask marks them, so the attention pooling sums only over genuinely active fires and the count of fires never changes the shape of the tensor. Choosing the 16 *nearest* fires is what bounds the observation while keeping it locally complete; the attention-based pooling over this set is the component whose removal costs the most in the ablation (Section 6.5), because it is what lets one network reason over a fire front of any size.

Teammate block (up to $N_{\max}$ entities, 5-d each). Up to $N_{\max} = 15$ teammate slots, each a 5-dimensional feature (the teammate's relative position, water level and assignment), again as a masked set: only the $N-1$ present teammates are unmasked, so a single network operates unchanged for any team size $N \in \{3, \ldots, 15\}$. This block is what carries the coordination signal at execution time, since each drone sees where its teammates are and what they are assigned to, and, with parameter sharing, it is what makes the network team-size-agnostic.

Critic-only global slice. Finally, a compact global-state summary (aggregate fire and team statistics) is exposed *only* to the centralized critic during training and is never present in the actor's observation at execution (Figure 1, dashed path); it therefore does not appear in the three actor blocks above. That this slice turns out to be unnecessary for performance is itself a result (Section 6.6).

5.7 Stage 4: Training campaign

The fourth stage runs the paradigm of Stage 3 on the environment of Stage 1 to produce a trained policy. The campaign is deliberately plain, so that the later comparisons turn on one clearly stated choice at a time rather than on a tuned recipe.

Each seed is trained for 250 M environment steps. The difficulty scalar d is annealed from 0 to 1 over the first portion of training, after which it stays at 1; the policy therefore spends most of training at the hard regime. We train *five* seeds of the principal policy and report statistics across them. The training stack is built on Ray RLlib, chosen because it is a mature, full-featured reinforcement-learning library that manages the whole distributed-computation layer, rolling out many environment workers in parallel and gathering their experience into each policy update, so that the campaign could turn on the algorithm and the environment rather than on the orchestration of compute. Optimiser settings, network sizes, batch and mini-batch sizes, and the exact compute configuration are listed in Appendix A.

5.8 Stage 5: Evaluation: protocol and metrics

The fifth stage decides how we judge a trained policy. A single average return is not enough to answer the operational questions a fire manager would actually ask, so we fix a sweep, a success criterion, and a small set of metrics, each chosen to answer one concrete question.

The "solved" criterion. We call a scenario (d, N) *solved* when the team puts out all the fire in at least 80% of episodes. The success of one episode is the indicator

$$c^{(e)} \;=\; \mathbb{1}\Big[\, F_t^{(e)} = 0 \;\text{ and held for } H \text{ steps} \,\Big], \tag{5.5}$$

which is 1 when every fire cell is extinguished and the no-fire condition is sustained for H consecutive steps, and 0 otherwise; here $F_t^{(e)}$ is the number of active-fire cells at step t of episode e. The *clearance rate* at a cell is then the mean of that indicator over the M evaluation episodes run at (d, N),

$$\mathrm{Clr}(d, N) \;=\; \frac{1}{M} \sum_{e=1}^{M} c^{(e)}, \tag{5.6}$$

and a scenario is solved when $\mathrm{Clr}(d, N) \geq 0.80$. The headline summary statistic is

$$N_{\mathrm{solved}}(d) \;=\; \min \{\, N : \mathrm{Clr}(d, N) \geq 0.80 \,\}, \tag{5.7}$$

the smallest team that clears the bar at difficulty d. The 80% threshold is a deliberate choice: it asks for reliable suppression, not a single lucky episode, while leaving room for the residual stochasticity of the fire.

The evaluation sweep. The protocol sweeps five difficulties $d \in \{0,\ 0.25,\ 0.5,\ 0.75,\ 1.0\}$ against eleven team sizes $N \in \{1, 2, 3, 4, 5, 6, 7, 8, 10, 12, 15\}$, giving 55 cells. Per cell we run 300 deterministic episodes (three episode-sampling seeds of 100 episodes each) and report the metrics below with 95% bootstrap confidence intervals taken across the five training seeds of the principal policy, so the intervals reflect seed-to-seed variation rather than episode sampling.

Operational metrics: burned area and time-to-clear. Clearance answers *whether* a team eventually wins; two further metrics say *at what cost*, and together they let us dimension the system. For an episode e the *burned-area fraction* is

$$b^{(e)} \;=\; \frac{\big|\{\text{cells ever in the burning or burned state}\}\big|}{|\,\text{grid}\,|}, \tag{5.8}$$

the share of the 22×38 map that the fire damaged before it was contained; the *time-to-clear* is the number of steps until the cleared condition of Equation (5.5) is first met, reported as a median over the *solved* episodes alone so that it times genuine successes rather than

timeouts. We track these because they convert an abstract policy comparison into a sizing question a fleet planner can use: how many drones are needed to hold a fire of a given regime below a tolerable level of damage, and what the marginal cost of fielding one platform fewer is, measured in extra burned area and slower containment. Reading the two metrics against team size N at a fixed difficulty is what makes that trade-off explicit (Section 6.8).

Why fire size matters. A related question is how performance depends on the *size* of the fire the team is handed. The difficulty scalar d controls how aggressively a fire spreads, which in steady state sets how large a front the team must hold; sweeping d therefore probes the difference between attacking a fire early, while it is still small, and being handed one that has already spread across much of the map. The two regimes demand different amounts of cooperation, and treating performance as a function of fire regime, rather than reporting one aggregate number, lets the benchmark show how the cooperation requirement scales (Section 6.4).

5.9 Stage 6: Iteration: baselines and ablations

The last stage closes the loop. To learn which parts of the design actually carry the result, we measure the principal policy against simpler alternatives and against itself with one component removed; what these comparisons reveal feeds back into the design choices of Stages 2 and 3. The ablation study itself is reported in Chapter 6; here we state the external baselines.

The principal policy is compared against two families of baselines.

Independent PPO (IPPO). An independent-learner variant, architecturally identical on the policy side but with a *decentralized* critic (each agent's value head sees only its own observation) and a purely local reward. IPPO tests whether centralized training improves over independent learning; it varies three factors jointly (a decentralized critic, no counterfactual advantage, and a local reward), so it compares training regimes rather than isolating the critic alone.

Heuristic controllers. Two non-learned controllers sharing the same refill logic but with no learning: a *greedy nearest-fire* controller that sends each drone toward the nearest active fire cell, and a *Hungarian-assignment* controller that solves the same min-cost matching used inside the learned policy's observation. The Hungarian-heuristic controller isolates what *learning* adds over hand-coded optimal assignment.

Chapter 6 Experimental Results

This chapter reports the experiments that test the methodology of Chapter 5 stage by stage. Four questions organise the study: whether a single policy generalises across team sizes it never trained on (the evaluation protocol of Stage 5); whether each of the five architectural choices is load-bearing (the neural architecture of Stage 2, through the component ablation); how far the learned policy improves on the best-known classical controllers evaluated in this preliminary study, and whether the centralized critic earns its place against an independent learner (the training paradigm of Stage 3); and how the difficulty curriculum and per-episode domain randomisation, which we treat as a standard part of training rather than a competitor, shape the policy's robustness.

6.1 Experimental setup

All experiments use the cooperative environment and the formal "solved" evaluation protocol of Chapter 5: a fixed grid of difficulties and team sizes, with 300 deterministic episodes per cell and the same seeds across every policy compared. Reported confidence intervals are 95% *bootstrap* confidence intervals (the range that would contain the true mean 95% of the time, estimated by resampling our five training seeds), so they reflect seed-to-seed variation rather than episode sampling noise. Reaching the "solved" bar across the curriculum took on the order of 250M environment steps per seed for the principal policy; the baselines were trained at a matched compute and data budget so the comparison is fair. Training was performed on 4× NVIDIA A40 GPUs; full optimiser and hyperparameter details are in Appendix A.

6.2 The environment: throughput and fire regimes

Two properties of the benchmark itself bound what any method trained on it can be measured against, so we state them before turning to the policy.

Throughput. A single environment worker runs at about 220 environment steps per second on the host used here. Every step advances the stochastic fire, recomputes the Hungarian assignment carried in the observation, and rebuilds the entity-structured observation, so this is the cost of one fully-featured timestep rather than a bare grid update; at $N = 8$ it is roughly 1,800 agent-steps per second. Training runs up to 40 such workers in parallel, which brings a full 250M-step seed in at about a day on the four-GPU host. The environment is cheap enough to support the large, repeated evaluation the benchmark calls for, which is the practical payoff of a 2D abstraction over a higher-fidelity simulator.

Fire regimes. The difficulty scalar d is a real knob, not a cosmetic one. Figure 2 characterises the five evaluation regimes by letting the fire spread with no suppression. As d rises from 0 to 1 the peak number of simultaneously-burning cells grows from about 39 to 258 and the share of the map the fire reaches grows from about 1.4% to 17%, both monotonically, and the

active-fire trajectory steepens visibly with d. A larger d is therefore a genuinely larger fire, which is what makes $N_{\text{solved}}(d)$ a fair difficulty axis: needing more drones to clear a harder regime is expected, not an artefact of the metric.

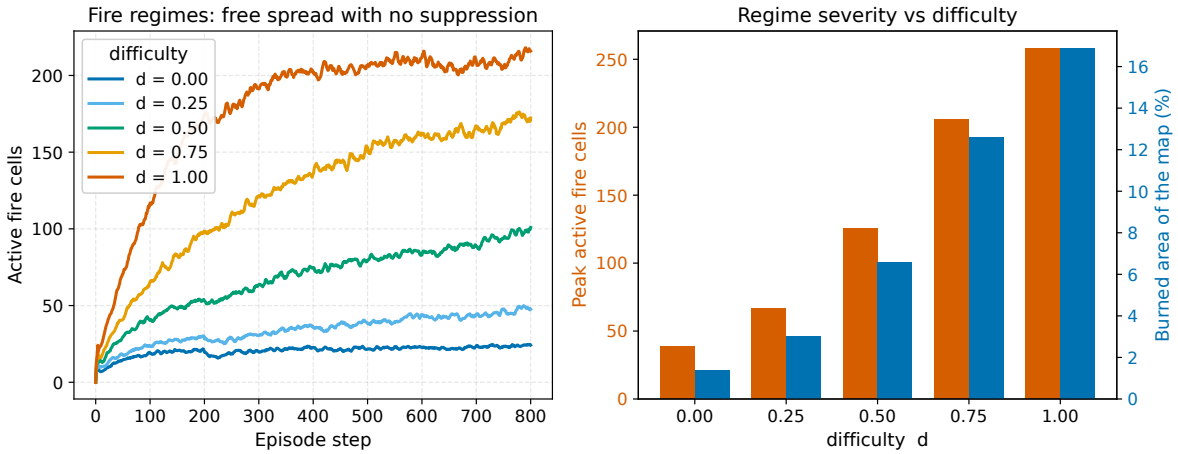

Figure 2: Fire regimes of the five evaluation difficulties under free spread (no suppression). Left: mean active-fire-cell trajectory per difficulty d, over 40 episodes each. Right: peak active-fire cells and burned area as a fraction of the map, per d. Both grow monotonically with d, so the difficulty scalar controls a genuinely harder fire.

6.3 Training campaign: the reward curve

Figure 3 shows the episode return of the five principal seeds over the first 100M steps, the window in which the difficulty curriculum ramps from $d = 0$ to $d = 1$ before holding at the hardest regime for the rest of training. The return does not climb the way it would on a fixed task: it rises quickly to roughly 20 to 30 and then holds there. That flat stretch is the part worth reading, not a stall: the team keeps the return roughly constant while the task underneath it ramps from the easiest fire to the hardest, so the policy is getting better at a steadily harder problem rather than settling on a fixed one.

The campaign was deliberately plain, so the later comparisons turn on one stated choice at a time. The reward coefficients $(\alpha_{\text{term}}, \alpha_{\text{prog}}, \alpha_{\text{press}}) = (200,\ 1.0,\ -0.15)$ were frozen once on the first seed and never retuned (Section 5.4), and the remaining hyperparameters follow the multi-agent PPO literature [8] rather than a per-task sweep, which is left as future work for compute reasons. The one genuine difficulty was reward shaping. An earlier reward let a drone "camp" a nearly-extinguished fire to keep banking progress reward, a failure mode shown qualitatively in Chapter 7, and the fix was the fixed three-term decomposition above rather than any change to the network or the learning algorithm.

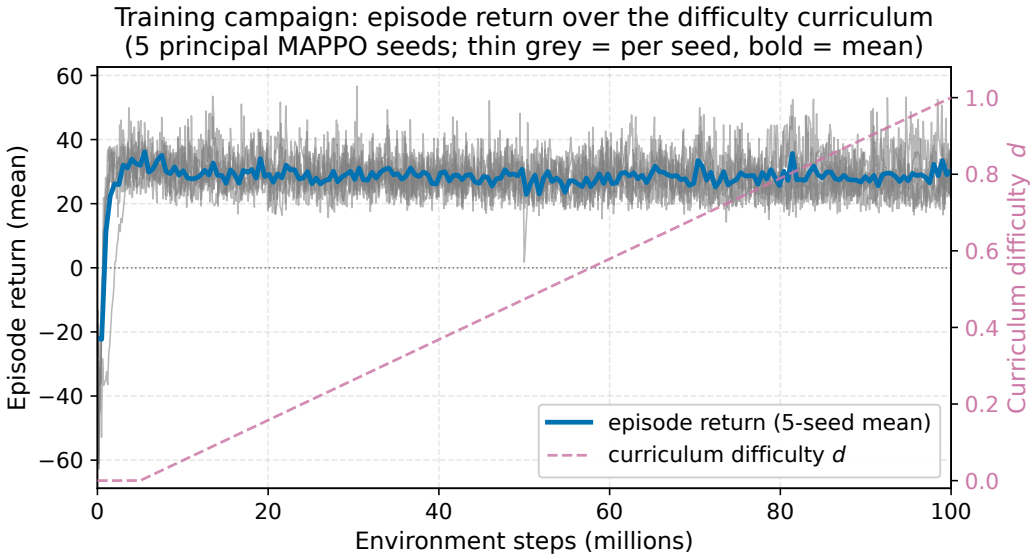

Figure 3: Training campaign of the five principal MAPPO seeds: episode return (thin grey per seed, bold mean) over the first 100M steps, during which the difficulty curriculum ramps from $d = 0$ to $d = 1$ (dashed, right axis). The return rises and then holds roughly flat across the ramp: the team keeps its performance steady while the task it faces grows from the easiest fire to the hardest.

6.4 Variable-team-size out-of-distribution generalisation

The central result is that a *single* policy, trained once on teams of $N \in \{3, \ldots, 10\}$ drones, generalises to teams it never saw during training. Figure 4 reports the clearance rate as a function of the difficulty d and the team size N, averaged over the five training seeds with 95% bootstrap confidence bands. Performance rises monotonically with team size at every difficulty and degrades gracefully with difficulty at every team size, with no collapse on the out-of-distribution sizes $N \in \{11, \ldots, 15\}$: at the hardest difficulty $d = 1$ the clearance rate climbs from 11.8% at $N = 4$ to 83.7% at $N = 15$, while at the easiest difficulty the team is essentially saturated ($\geq 99\%$ clearance for any $N \geq 7$).

We summarise this with $N_{\text{solved}}(d)$, the smallest team size whose mean clearance reaches the 80% "solved" threshold defined in Section 5. It grows monotonically with difficulty (Table 3): a team of 4 suffices on the two easiest difficulties, but the requirement rises to 6, 10 and finally 15 drones as the fire becomes harder to contain. This gives a clean, quantitative characterisation of *how much cooperation a scenario demands*: the cooperation requirement is not a single number but a monotone function of the environment difficulty.

A note on the hardest cell, for honesty. The $N_{\text{solved}}(1) = 15$ entry is a claim about the *seed-averaged* policy: the mean clearance at (d=1, N=15) is 83.7%, but the five individual seeds clear 91.3%, 76.3%, 84.7%, 86.3% and 79.7% of episodes, so two of the five seeds fall just under the 80% threshold on their own. The result is therefore best read as "the architecture

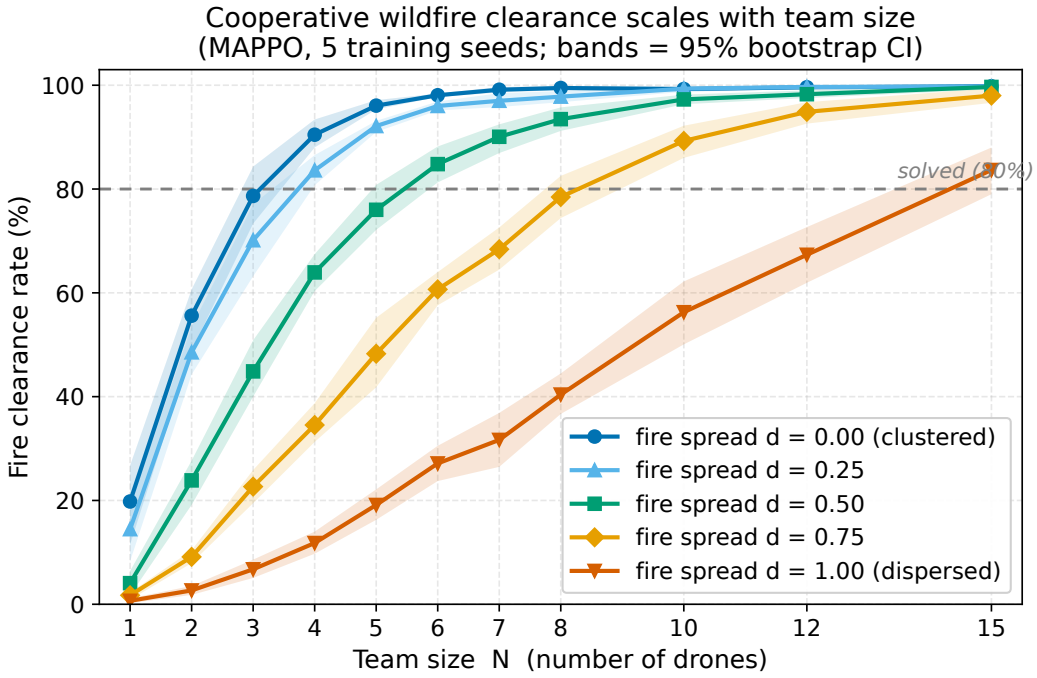

Figure 4: Clearance rate as a function of team size N and difficulty d, averaged over five training seeds; bands are 95% bootstrap confidence intervals. The minimum team size that clears the 80% "solved" threshold grows monotonically from 4 (at $d = 0$) to 15 (at $d = 1$), and performance is stable on the out-of-distribution team sizes $N > 10$.

Table 3: Minimum cooperating team size $N_{\text{solved}}(d)$ (smallest N whose mean clearance reaches 80%) for the principal five-seed policy, with the clearance attained at that team size.

Difficulty d	0.0	0.25	0.5	0.75	1.0
N_{solved}	4	4	6	10	15
Clearance at N_{solved}	90.5%	83.6%	84.8%	89.3%	83.7%

reliably solves the hardest cell, with seed-to-seed variance that straddles the threshold" rather than "every seed solves it." We report the full per-seed spread, not only the mean, throughout this chapter.

6.5 Component ablation

To establish that each design choice is load-bearing, we retrain the policy with one component removed at a time and re-evaluate on the identical sweep. Table 4 reports $N_{\text{solved}}(d)$ and the clearance at the hardest cell (d=1, N=15) for the full policy and seven ablations, ordered by severity.

Table 4: Component ablation: $N_{\text{solved}}(d)$ (smaller is better; >15 means never solved at any available team size) and clearance at the hardest cell (d=1, N=15). Each ablation removes a single component and is retrained at matched budget. Rows are ordered by hardest-cell clearance.

Variant	d=0	d=0.25	d=0.5	d=0.75	d=1	Clr. (d=1, N=15)
Full policy (principal)	4	4	6	10	15	83.7%
− τ-anneal ($\tau \equiv 1$)	4	4	6	10	15	83.3%
− team spirit ($\tau \equiv 0$)	4	5	8	12	>15	68.3%
− LSTM (memoryless)	5	6	8	15	>15	65.3%
− reward rebalance	4	5	8	12	>15	58.7%
− Hungarian assignment	5	6	10	15	>15	58.3%
− entity attention	5	6	10	>15	>15	46.3%

We read the table one component at a time, asking why the task stays solvable without each component and why removing it costs what it does.

Entity attention is the most load-bearing choice. Removing it drops the hardest-cell clearance from 83.7% to 46.3% and pushes N_{solved} beyond the largest available team from $d = 0.75$ onward. The reason is structural: without attention the network cannot pool a variable-cardinality set of fires and teammates into a fixed-size summary, so as the front grows and the team changes size the observation loses the information coordination depends on. The task stays nominally solvable at low difficulty, where a handful of fires can be handled almost reactively, but it collapses where many fires and many drones must be reasoned about together.

Setting entity attention aside, three components each cost a substantial but similar amount. The *Hungarian assignment* prior (hardest-cell 58.3%) hands the policy a near-optimal one-shot drone-to-fire matching; without it the network must learn the matching from scratch, which it partly manages, so performance falls but does not collapse at easy difficulties. The *LSTM* (hardest-cell 65.3%) supplies the short-horizon memory that lets a drone remember which fire it was heading for across the travel and refill phases; removing it makes target switching jittery, which matters most when the front is large and a drone's commitment must survive several steps. The *reward rebalance* (hardest-cell 58.7%) restores the value scale that keeps the terminal bonus, the progress term and the fire-pressure penalty in proportion; the mis-scaled reward still points roughly in the right direction, so the policy learns, but it under-weights the long-horizon clearing bonus and clears fewer of the hard cells. That all three sit in a tight 58–65% band, well above the no-attention floor, says the task is still learnable without any one of them: each removes a useful prior, not a load-bearing capability.

Team spirit matters; its annealing schedule does not. Holding the team-spirit weight constant at $\tau \equiv 1$ reproduces the annealed baseline almost exactly (83.3% vs. 83.7%, with identical N_{solved} at every difficulty), whereas removing the shared reward entirely ($\tau \equiv 0$, fully selfish agents) degrades coordination to 68.3%. The selfish variant still solves the easy cells

because at low difficulty a drone's local incentive and the team's outcome nearly coincide; they diverge only when the fire is large enough that one drone must give up a nearby fire so a better-placed teammate can take it. We therefore credit a *constant* team-spirit reward, not a schedule over τ (a distinction the dedicated constant-τ ablation makes visible, as the methodology already anticipates in Section 5).

Each ablation is a single retrained seed (versus five for the principal policy), so given the principal policy's 76–91% seed spread at the hardest cell, ablation gaps below $\sim$10 percentage points should be read as suggestive; we draw firm conclusions only from the large, consistent effects (entity attention, team spirit).

6.6 Algorithm comparison: MAPPO vs. IPPO vs. heuristics

We next ask what *learning* contributes over hand-coded control. Three non-learned controllers share the environment and the same water-refill logic but carry no learned policy: a *random* controller, a *greedy* nearest-fire controller that sends each drone toward the closest active fire, and a *Hungarian-assignment* controller that solves the very same min-cost drone-to-fire matching that the learned policy receives as an observation prior. Table 5 compares them with the principal MAPPO policy on the identical sweep.

Table 5: Learning vs. hand-coded control: $N_{\text{solved}}(d)$ and clearance at the hardest cell. The principal policy solves all five difficulties; the best heuristic solves only the two easiest, and only with $N = 10$ drones.

Controller	d=0	d=0.25	d=0.5	d=0.75	d=1	Clr. (d=1, N=15)
MAPPO (centralized critic, 5-seed)	4	4	6	10	15	83.7%
IPPO (independent, 4-seed)	4	4	6	10	15	80.3%
Heuristic: Hungarian	10	10	15	>15	>15	48.0%
Heuristic: greedy	15	15	>15	>15	>15	30.7%
Heuristic: random	>15	>15	>15	>15	>15	0.0%

The learned policy outperforms each of the scripted controllers we evaluated by a wide margin. These are the best-known classical controllers we could fit to the task within this preliminary study, not the full space of classical methods, so the claim is a comparison against strong hand-coded baselines rather than a proof of superiority over every possible controller. The strongest of them, the Hungarian-assignment controller, tops out at 48.0% clearance on the hardest cell and solves only the two easiest difficulties (and only at $N = 10$ drones), whereas the principal policy reaches 83.7% and solves all five; greedy control (30.7%) barely improves on random (0.0%). The comparison with the Hungarian controller is the sharpest: the heuristic and the learned policy are handed the *same* optimal assignment, so the 48.0% $\rightarrow$ 83.7% gap is exactly what *learning adds on top of hand-coded optimal assignment*: the timing of the travel-suppress-refill cycle, target switching as the front moves, and spatial deconfliction, none of which a one-shot matching can express. The honest separation is

between the *full* learned stack and hand-coded control: the full stack (83.7%) far exceeds any hand-coded controller, while the weakest ablation (no-attention, 46.3%) is within noise of the best heuristic (Hungarian, 48.0%).

Centralized training is not necessary here. The second comparison replaces the centralized critic with a decentralized one (IPPO): each agent's value head sees only its own observation, with a purely local reward, while the policy network is identical (Section 5). IPPO *matches* the principal policy on every solved-threshold metric (Table 5): an identical N_{solved} curve 4/4/6/10/15, 80.3% vs. 83.7% clearance at the hardest cell with fully overlapping per-seed spreads (IPPO 75.0–87.3, MAPPO 76.3–91.3; a standard significance test finds no detectable difference, Welch t=0.90, $p \approx 0.40$), and a near-identical median suppression time (≈ 77 steps for both at the hardest cell, and within a few steps of each other at every solved cell). On FireCoopBench the coordination signal is supplied by parameter-sharing and the entity-attention prior, not by the centralized critic, which is one of two pieces of standard CTDE machinery that turn out to be unnecessary here (the other being the team-spirit *anneal*, Section 6.5). This is less surprising than it may first appear. A centralized critic helps most when partial observability is severe enough that the global state carries information the agents cannot recover from their own observations; when that gap is small the benefit shrinks, and it can be cancelled by the fact that the critic encodes the global state through a different pathway than the actor encodes its local view, so the value signal the actor trains on grows noisier the more those two internal representations diverge. On FireCoopBench the entity-attention observation already surfaces the nearest fires and the teammates that bear on the next decision, leaving the global-state critic little to add, which is consistent with the two approaches landing on top of one another. Two honest caveats follow. IPPO varies three factors at once (critic, counterfactual advantage, reward), so it is a comparison of training *regimes*, not a single-factor ablation of the critic; a clean critic-only run is left to future work. And the ablation table holds the complementary clue: the τ=0 *selfish* variant (which keeps the centralized critic and COMA) scores only 68.3%, *below* IPPO's 80.3% at the same local reward. This suggests the counterfactual advantage is high-variance under a purely local reward, where there is no shared signal for the centralized critic to exploit.

6.7 Robustness analysis

Domain randomisation (DR) is a standard ingredient of deep reinforcement learning: randomising the task at every reset is the usual way to push a policy toward generalisation and away from overfitting a single configuration, and it is part of how the principal policy is trained, since the team size N is sampled afresh every episode. The difficulty curriculum is a complementary tool: it does not replace randomisation but stages it in time, so the policy meets easy fires before hard ones. The question this subsection asks is not "curriculum or DR" but how much the staging matters once the same randomisation is in place. To isolate that, we trained a two-seed variant in which the *difficulty* d is also drawn uniformly at every reset instead of being annealed, leaving everything else identical, and evaluated it on the same

sweep (Table 6, Figure 5).

The staged variant attains the higher in-distribution scores: on the hardest cell it clears 83.7% against the fully-randomised variant's 31.3%, and the randomised variant needs $N = 10$ drones to solve even the easiest difficulty and does not reach the bar for $d \geq 0.5$ at any available team size. We are careful not to over-read this as "DR is worse than a curriculum." Two more mundane explanations fit the data at least as well, and we have not ruled them out within this study. First, the two-seed variant reused the principal policy's hyperparameters, which were tuned for the staged schedule; a randomised schedule may simply need a different learning-rate or entropy setting. Second, drawing d uniformly spends a large share of episodes at the hard end, where, at small N, the scenario may be close to unsolvable at all, so part of the randomised budget is spent on episodes from which there is little to learn. The defensible reading is narrow: under our hyperparameters and randomisation ranges, staging the difficulty in time improved in-distribution performance over drawing it uniformly. It is not evidence that domain randomisation as a method is inferior.

Table 6: Difficulty curriculum vs. per-episode domain randomisation, on the same sweep, with identical hyperparameters. Under these settings the staged schedule attains higher in-distribution clearance at every difficulty; this compares two schedules of the same randomisation, not domain randomisation against a curriculum as methods.

Training schedule	d=0	d=0.25	d=0.5	d=0.75	d=1	Clr. (d=1, N=15)
Difficulty curriculum (principal)	4	4	6	10	15	83.7%
Domain randomisation (2-seed)	10	10	>15	>15	>15	31.3%

The most likely reading is about how the fixed budget is spent. Drawing d uniformly places a large fraction of episodes in regimes that, at the team sizes seen during training, are barely solvable, so part of the budget yields little learning signal; staging the difficulty concentrates the same budget on episodes the team can make progress on. On that reading the curriculum is a useful way to schedule a hard task, not a reason to prefer it over domain randomisation in general. This is also an in-distribution comparison only. A genuine robustness stress test, an adversarial fire that actively probes the team's weaknesses (RARL), is left as future work, together with a higher-fidelity fire model.

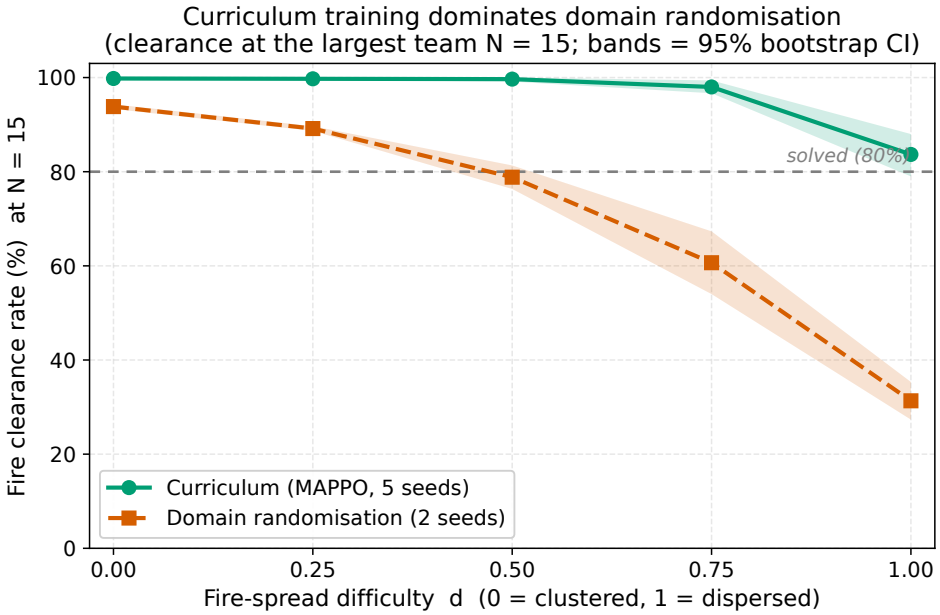

Figure 5: Clearance of the staged-difficulty (curriculum) variant versus the uniformly-randomised-difficulty variant across the difficulty range, at matched hyperparameters. Staging the difficulty, so that more of the budget is spent at the hard regime once d has annealed to 1, gives higher in-distribution clearance than spreading the same budget uniformly over difficulties under these settings.

6.8 Burned area and time-to-clear: the cost of an undersized team

The clearance rate of Section 6.4 answers whether a team eventually puts the fire out; the two operational metrics here answer at what cost, and they are the pair that matters most for sizing a real deployment. Burned-area fraction is the damage the system fails to prevent, and time-to-clear is how long it ties up the fleet; together they let an operator trade the cost of an extra drone against the area and the hours it would save, which is the question a dimensioning study has to answer. Both deteriorate sharply once N drops below $N_{\text{solved}}(d)$, which is the practical content of "undersizing" a team. Figure 6 shows the burned-area fraction: it shrinks monotonically as the team grows, and at the hardest cell (d=1, N=15) only about 2.6% of the grid burns, down from roughly 14% at small teams, a fivefold reduction in destroyed area bought purely by adding drones. Figure 7 shows the complementary speed metric. Restricting to *solved* cells (mean clearance $\geq 80\%$, the threshold of Section 5), so that we time only the episodes that actually succeed, the median time-to-clear falls steeply with N (on the easiest difficulty from about 66 steps at $N = 4$ to about 27 at $N = 15$), while the hardest difficulty d=1 contributes a single solved cell at $N = 15$ that clears in about 77 steps (5-seed mean; per-seed 59–90). The two figures together quantify the dense reward terms of Section 5.4: the fire-pressure penalty is what the policy is minimising when it drives time-to-clear down, and

the progress term is what it is maximising when it shrinks burned area. The reading we draw is that under-sizing is not a soft penalty but a regime change: below N_{solved} the team neither clears reliably (Section 6.4) nor contains the damage, and both costs climb together.

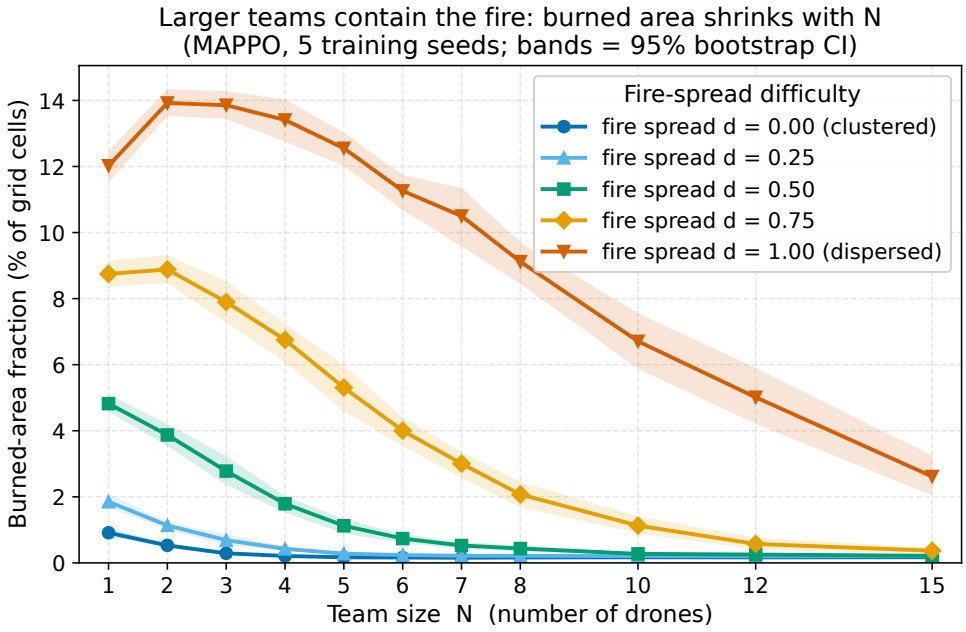

Figure 6: Burned-area fraction versus team size N, by difficulty d (five-seed mean, 95% bootstrap bands). Larger teams contain the fire earlier, so less of the 22×38 grid burns: at the hardest cell (d=1, N=15) only ∼2.6% of cells burn, against ∼14% for small teams. The curve is the damage-cost counterpart of the clearance curve in Section 6.4.

6.9 How the two solutions coincide

Section 6.6 established, on the solved-threshold metrics, that the centralized critic confers no measurable benefit, the negative result that independent learning matches centralized training on FireCoopBench. The two figures here show that coincidence rather than asserting it, which is the standard the director's rules set for a claim of no effect. Figure 8 overlays the full clearance-versus-N curves for MAPPO and IPPO at $d \in \{0.5, 0.75, 1.0\}$: the two curves track each other across the entire team-size range and their 95% confidence bands overlap everywhere, so there is no difficulty or team size at which centralized training pulls ahead. Figure 9 zooms into the single hardest cell (d=1, N=15) and plots the individual seeds: MAPPO's five seeds (91.3/76.3/84.7/86.3/79.7, mean 83.7%) and IPPO's four seeds (77.7/87.3/81.3/75.0, mean 80.3%) interleave into one overlapping band, with the seed spread of each method dwarfing the 3.4-point gap between their means. A Welch test on these seed-level scores finds no detectable difference (t=0.90, $p \approx 0.40$). We are careful about what

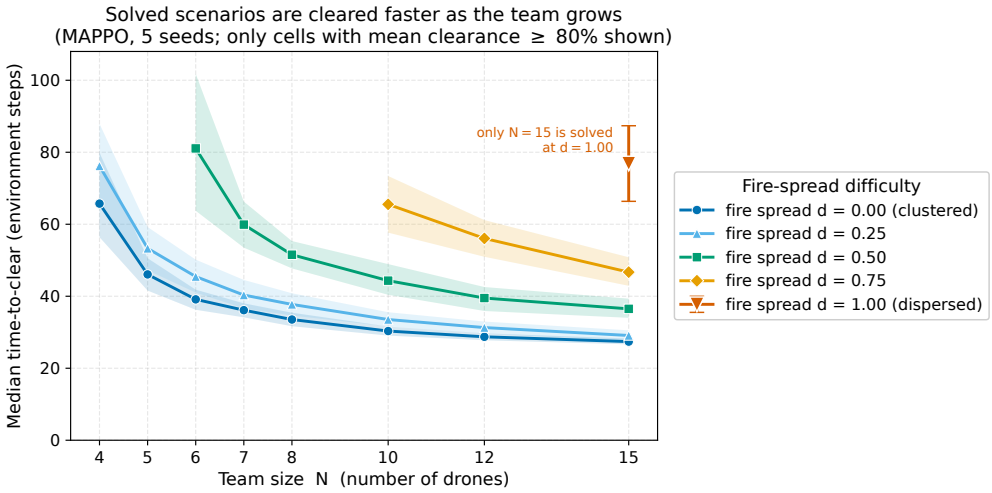

Figure 7: Median time-to-clear (in environment steps) versus team size N, computed only over *solved* cells (mean clearance $\geq 80\%$, the threshold of Section 5) so that the timing reflects successful episodes alone. Suppression gets faster with team size, for example at d=0 from $\sim$66 steps at $N = 4$ to $\sim$27 at $N = 15$; the hardest difficulty d=1 has a single solved cell ($N = 15$, $\sim$77 steps, 5-seed mean; per-seed 59–90). Faster clearance is the policy directly minimising the per-step fire-pressure penalty of Section 5.4.

this does and does not say: it is genuine evidence that the centralized critic is unnecessary in this setting, but IPPO varies three factors at once (critic, counterfactual advantage, local reward), so it is a comparison of training regimes and not a single-factor isolation of the critic. This is the caveat already drawn in Section 6.6, which a reader should carry into both figures.

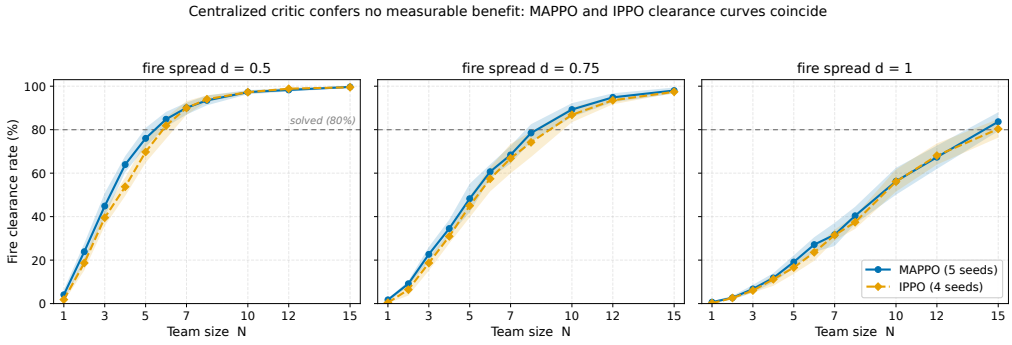

Figure 8: Clearance versus team size N for centralized MAPPO and independent IPPO, at $d \in \{0.5, 0.75, 1.0\}$ (seed mean, 95% bootstrap bands). The curves nearly coincide and their confidence bands overlap at every team size and difficulty: the centralized critic confers no measurable benefit on FireCoopBench (cf. Section 6.6).

The per-episode return distributions at the same cell tell the same story from a different

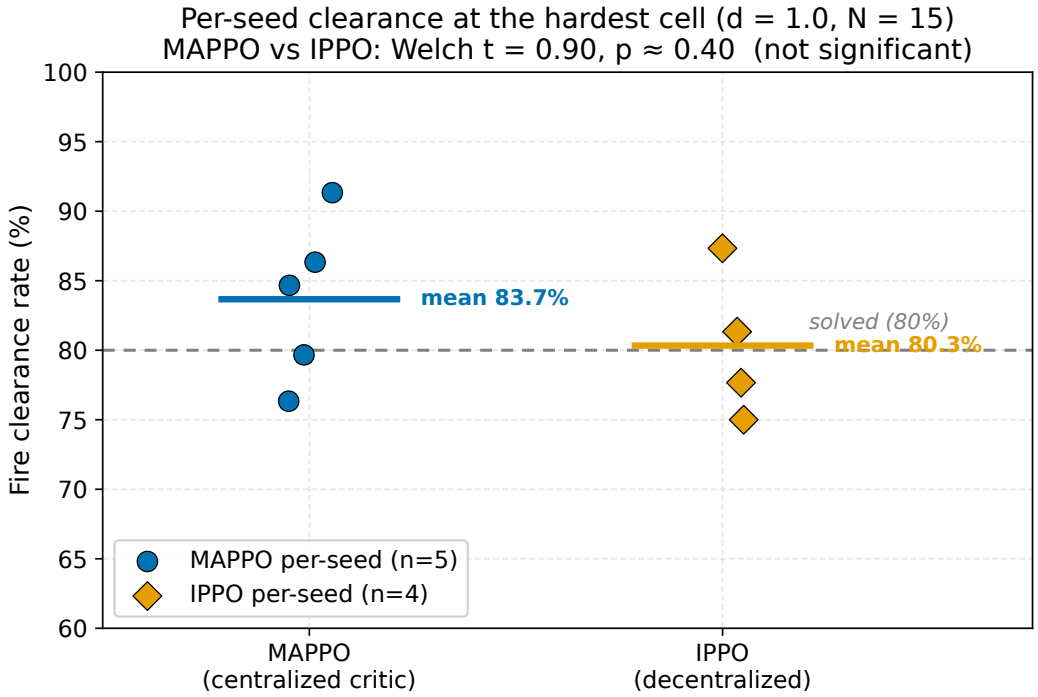

Figure 9: Per-seed clearance at the hardest cell (d=1, N=15). MAPPO's five seeds (mean 83.7%) and IPPO's four seeds (mean 80.3%) form fully overlapping spreads; the seed-to-seed variation of each method exceeds the gap between their means, and a Welch test finds the difference not significant (t=0.90, $p \approx 0.40$). The negative result on centralized training is thus shown, not merely asserted.

angle (Figure 10): both methods concentrate near the cleared-episode return and share a lower tail of occasional wipes, and their per-seed means interleave rather than separate.

6.10 Component importance at a glance

Section 6.5 reports the full ablation as a table of $N_{\text{solved}}(d)$ curves; Figure 11 renders the single most legible slice of it, hardest-cell clearance, as a ranked bar chart, so that a non-specialist reader can read off the ordering of the five design choices at a glance. The descent is steep and monotone: full policy 83.7%, τ-anneal removed (constant τ) 83.3%, selfish (τ=0) 68.3%, no-LSTM 65.3%, reward-rebalance 58.7%, no-Hungarian 58.3%, and finally no-attention 46.3%. The picture makes two points that the table states but the eye confirms faster: the τ-anneal bar is indistinguishable from the full bar (the schedule is inert), and entity attention matters most by a clear margin: removing it costs more than thirty-seven points and, as noted in Section 6.6, drops the policy to within noise of a hand-coded heuristic. We refer the reader to Section 6.5 for the per-difficulty detail; the bar chart is an entry point, not a replacement.

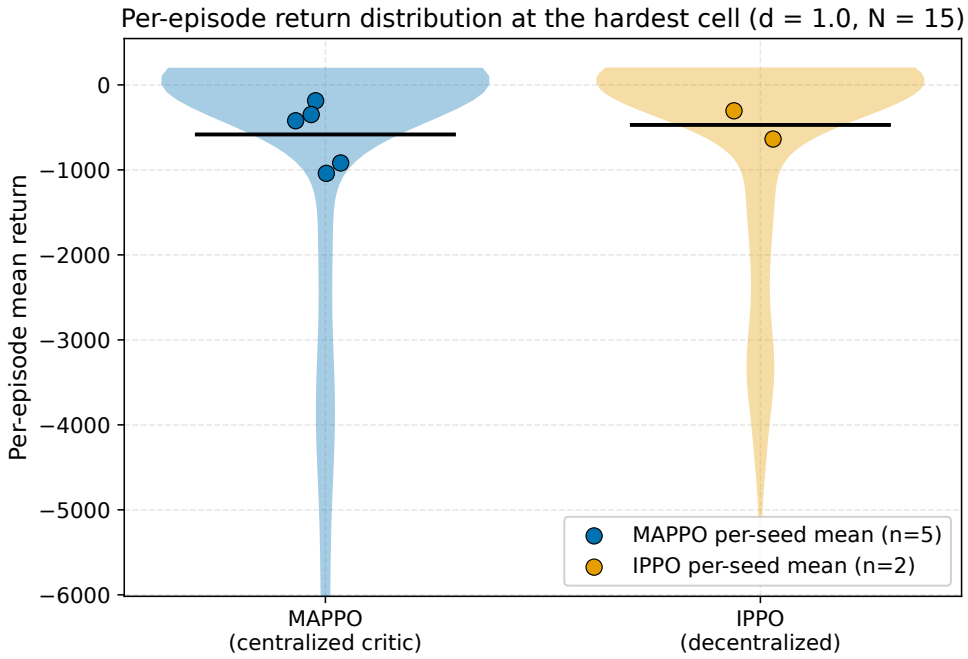

Figure 10: Per-episode mean return at the hardest cell (d=1, N=15), the distribution view complementing the per-seed clearances of Figure 9. Each violin is the distribution over individual episodes (MAPPO: five seeds; IPPO: the two training checkpoints retained on disk), with per-seed means overlaid as points. Both methods pile up near the cleared-episode return and share a lower tail of occasional catastrophic wipes (clipped here for readability); the distributions overlap, consistent with the no-significant-difference clearance result above.

6.11 Learning versus hand-coded control

Section 6.6 reports that the learned policy clears more of the fire than the best-known scripted controllers we evaluated; Figure 12 is its visual summary at the hardest difficulty, placing the two learned policies and the three scripted controllers on one axis. The separation is clear within the controllers we tested: at d=1 each learned policy (MAPPO 83.7% and IPPO 80.3% at $N = 15$) clears more than each heuristic, and the strongest heuristic, Hungarian assignment at 48.0%, sits below even the weakest learned full policy by a wide margin, with greedy (30.7%) and random (0.0%) further behind. The sharp comparison, made in Section 6.6 and visible here, is the Hungarian one: the heuristic and the learned policy receive the same optimal one-shot assignment, so the gap from 48.0% to 83.7% is exactly what learning adds on top of hand-coded optimal matching: the trajectory-level timing of refills, target switching as the front moves, and spatial deconfliction that a one-shot assignment cannot express. We keep the one honest qualifier from that section: the gap is between the full learned stack and the heuristics; the weakest ablation (no-attention, 46.3%) lands within noise of Hungarian (48.0%), so it is the complete architecture, not "learning" in the abstract, that beats hand-coded

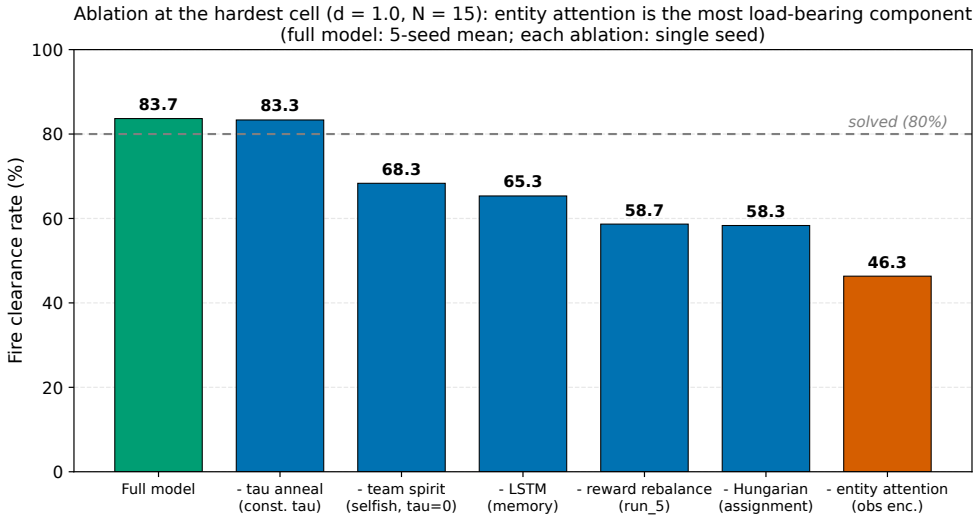

Figure 11: Hardest-cell (d=1, N=15) clearance for the full policy and each single-component ablation, ranked. Full 83.7% / τ-anneal-removed 83.3% / selfish 68.3% / no-LSTM 65.3% / reward-rebalance 58.7% / no-Hungarian 58.3% / no-attention 46.3%. Entity attention is the single most important component, and removing the team-spirit anneal (leftmost two bars) changes nothing. Full per-difficulty figures are in Section 6.5.

control.

6.12 Division of labour

A team can reach high clearance in two very different ways: by genuinely sharing the work, or by having one or two capable drones carry the team while the rest contribute little. Because clearance alone cannot distinguish these, we report two behavioural indicators at the hardest difficulty d=1 (Figure 13). The first is the median per-agent return ratio, defined as the ratio of the largest to the smallest per-agent return within an episode (so 1.0 means the best- and worst-off drone earn the same), which stays close to 1.0 across team sizes; a value near one means the reward, and by proxy the suppression work, is shared evenly rather than concentrated on one or two drones. (This ratio is defined only for episodes in which every drone finishes with a positive return, so it characterises the fully-successful episodes rather than the whole sweep.) The second is the spread of the extinguish-action share across drones, which stays in a narrow band as the team grows (about 13 to 19 percentage points, peaking near $N = 8$ and easing slightly thereafter), indicating that even at large N no small subset monopolises the extinguishing. Read together, the two indicate that, in the episodes it solves, the team keeps the work fairly evenly shared at every team size, consistent with genuine cooperation rather than a one-or-two-drone free-rider regime. We are deliberately cautious here: these are behavioural indicators, not a proof of optimal role allocation. A near-unit return ratio is consistent with balanced cooperation but does not establish that the

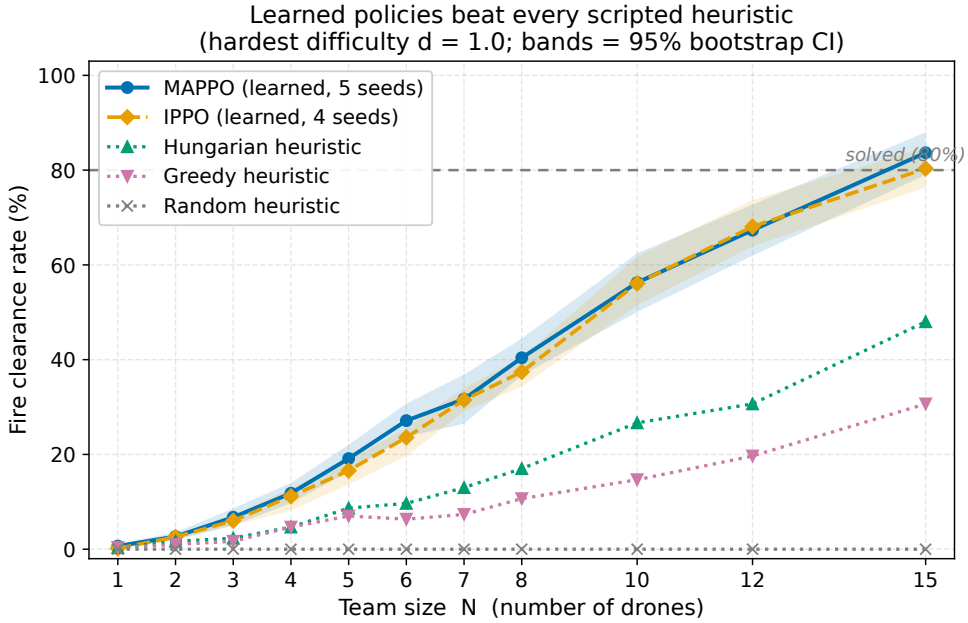

Figure 12: Hardest-difficulty d=1 clearance: learned policies versus scripted controllers. Both learned policies (MAPPO 83.7%, IPPO 80.3% at $N = 15$) clear more than every heuristic (Hungarian 48.0%, greedy 30.7%, random 0.0%). Because the Hungarian heuristic is handed the same optimal assignment the learned policy sees, the gap above it isolates what *learning* contributes beyond hand-coded matching (Section 6.6).

realised division of labour is the best one available, and the action-share spread measures dispersion, not efficiency. The behaviour gallery (Chapter 7) shows individual episodes, including out-of-distribution runs where some drones idle, that put a human-readable face on these aggregate numbers.

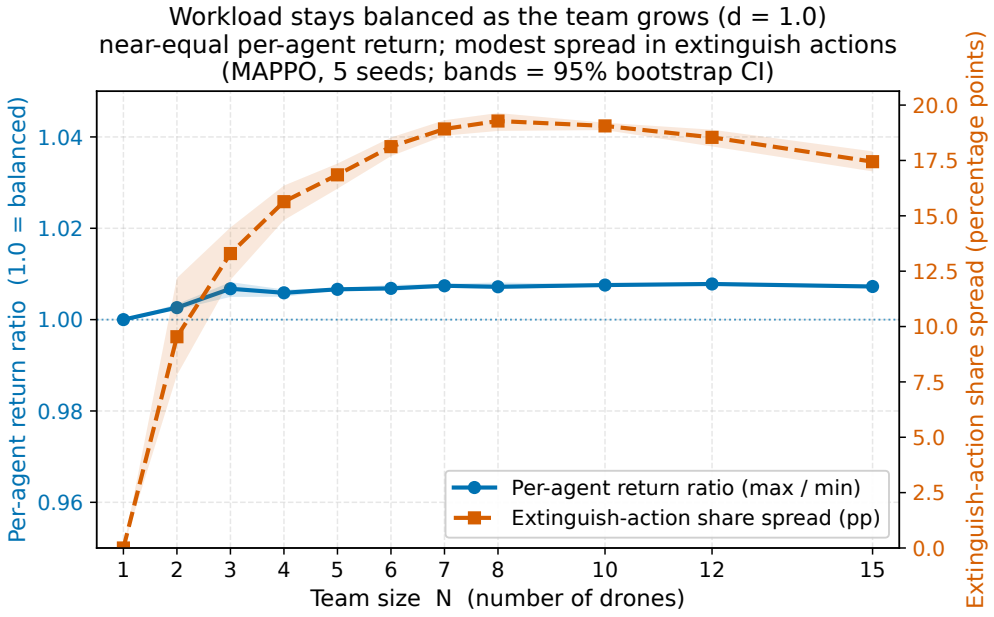

Figure 13: Two behavioural indicators of cooperative balance at d=1 versus team size N. The median per-agent return ratio (largest-to-smallest per-agent return in an episode; 1.0 = perfectly even, over episodes where all drones finish positive) stays near 1.0, and the extinguish-action-share spread across drones stays in a narrow band (about 13 to 19 percentage points, peaking near $N = 8$) as the team grows; together they suggest sustained balanced cooperation. These are behavioural indicators of load balance, not a proof of optimal role allocation.

Chapter 7 Behavioural Analysis

This chapter complements the quantitative study of Chapter 6 with a qualitative reading of how the trained policies actually behave during an episode. Every figure is a frame strip: a left-to-right sequence of frames sampled at roughly equal intervals across a single deterministic episode, read like a comic; the frames themselves carry no in-image annotation, so the caption supplies the relevant timesteps and per-agent returns. The strips are the on-paper record of behaviour; the corresponding full-motion videos are archived with the code under `videos/` (and the project OneDrive backup), regenerable by the strip-generation script, but nothing in this chapter depends on them. Each figure is fully understandable as a static image, with a caption that states the evaluation cell (d, N, policy), what to look for in the sequence, and the quantitative signature (step count and per-agent returns) that anchors the behaviour to the numbers reported earlier.

The episodes shown are deterministic (seed fixed per figure). Per-agent returns are reported throughout because their spread is usually the cleanest behavioural signature: a tightly bunched return distribution indicates an evenly shared workload, whereas a long negative tail betrays one drone absorbing the burning-building damage while the others coast.

The chapter follows the same order as the methodology and results pipeline of Chapter 6, so each strip can be read against the aggregate number it illustrates. We begin with the variable-team-size story (cooperative success at the publishable hard cell, the intended-training operating point, and out-of-distribution generalisation), then turn it on its head with the undersized-team failure mode, which is where the saturation point of Section 6.4 (the minimum number of drones a fire requires before any team can clear it) acquires a behavioural face. After a cautionary strip from before the reward redesign, we close with the two component analyses that mirror the results chapter: the most load-bearing ablation, and the independent-learner (IPPO) comparison, which reads as a difference in style rather than a gap in performance.

7.1 Cooperative success at the publishable hard cell

These two strips show the principal policy at the hardest cell it is asked to solve, (d=1.0, N=10), for two different training seeds. They are the visual counterpart of the headline generalisation result of Section 6.4. A single policy clears the hardest in-distribution cell, and independently trained seeds settle on the same cooperative pattern when they do.

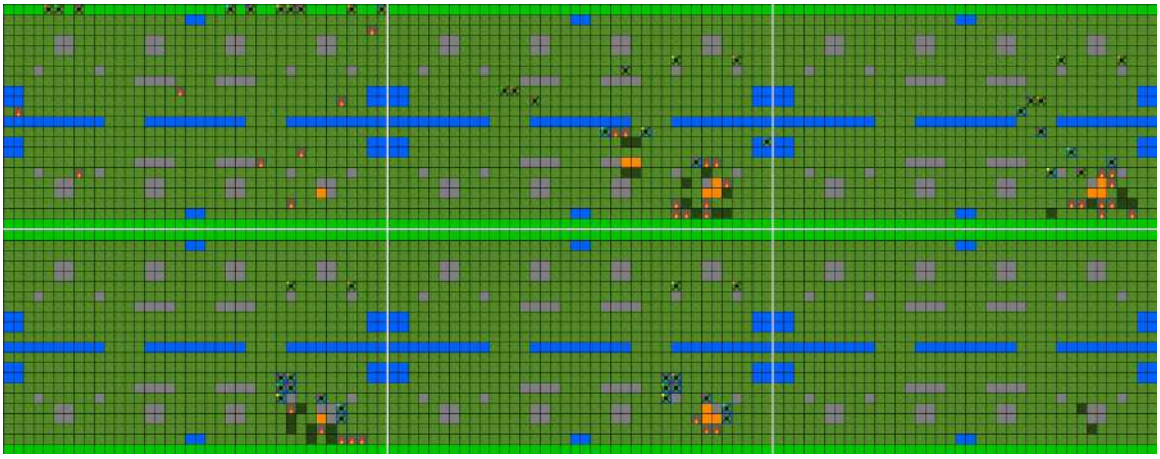

Figure 14: **MAPPO at the publishable hard cell (d=1.0, N=10, seed 42, run 7).** Read left to right: the team spreads along the fire front, refills its finite water tanks in shifts, and converges on residual hotspots without crowding a single cell. The map is cleared in 69 steps and the per-agent returns are tightly bunched (between 63 and 67). That tight bunching is the per-episode behavioural signature of the well-distributed cooperation that, in aggregate, produces the 83.7% seed-averaged clearance at this difficulty (Section 6.4).

7.2 Behaviour at the intended-training operating point

The hard cell above is the worst case; the strips below show the policy in the regime it was trained to occupy, the mid-difficulty cells around $N_{\text{solved}}(0.5)$=6 (Table 3). They illustrate

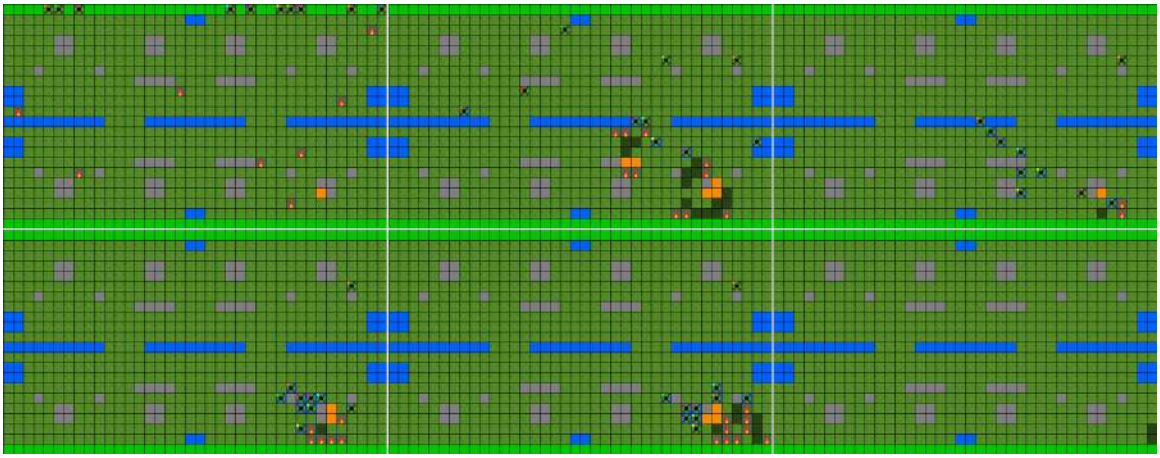

Figure 15: **Seed robustness: MAPPO seed** 9 **at the same cell (**d=1.0**,** N=10**).** A policy trained from a different seed clears the identical cell in 69 steps with the same return distribution (between 63 and 67) and the same front-spreading, shift-refilling pattern as Figure 14. Cooperation converges to qualitatively the same solution across seeds, consistent with the seed-to-seed spread reported in Section 6.4.

two distinct sub-regimes: a comfortable success with sustained suppression, and a borderline cell just below the solved threshold where the team still wins but pays for it in attrition.

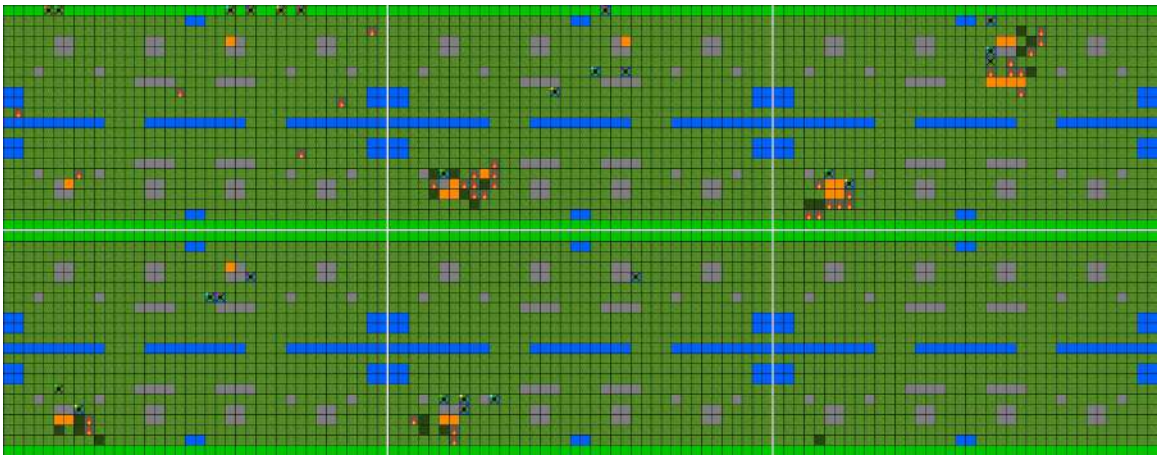

Figure 16: **Clean mid-difficulty success (**d=0.5**,** N=6**, MAPPO run** 7**/ckpt** 006241**, seed** 42**).** This is the intended operating point of the principal policy, exactly at $N_{\text{solved}}(0.5)$=6 (Table 3). The team achieves full fire clearance in 252 steps. The strip shows sustained suppression rather than a quick rush: several drones exhaust a full tank and complete a refill cycle, and two drones are lost late in the episode, yet the front never escapes containment.

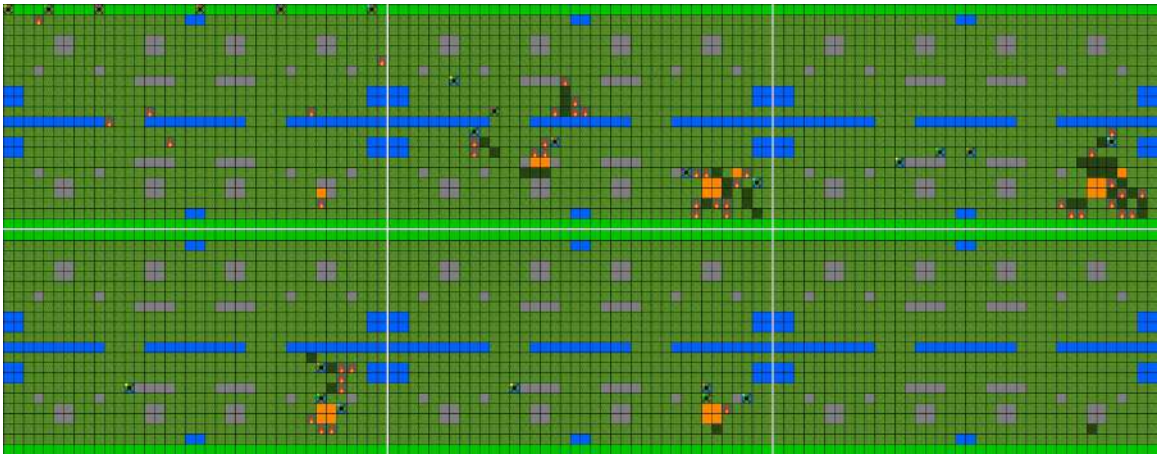

Figure 17: **Borderline harder cell (d=0.75, N=6, MAPPO run 7/ckpt 006241, seed 7).** At d=0.75 the cooperation requirement rises to N_{solved}=10 (Table 3), so a team of six is below the comfortable margin. The fire is still cleared, in 88 steps, but the outcome is mixed: two of the six drones are lost and the mean per-agent return on this episode is only +1.8. This is the *success-with-attrition* regime that sits just under the solved threshold: the team wins, but the thin return distribution shows it has no margin to spare.

7.3 Variable-team-size, out-of-distribution generalisation

The next strips probe team sizes the policy never saw during training (trained on $N \in \{3, \ldots, 10\}$). They are the qualitative evidence for the out-of-distribution stability reported in Section 6.4: the assignment mechanism keeps additional drones useful rather than letting them thrash.

7.4 Failure mode: the undersized team and the saturation point

This strip is the counter-example that gives the cooperation requirement of Section 6.4 its teeth. It shows what happens when the team is simply too small for the difficulty, no matter how good the policy is.

The strip dramatises a single number that is easy to overlook in the aggregate plots. For each difficulty d there is a minimum number of drones below which no team clears the fire reliably, and at or above which the same policy succeeds. We measured this earlier as $N_{\text{solved}}(d)$, the smallest team that reaches the 80% clearance bar (Table 3), and it behaves like a saturation point for the fire: it climbs from 4 drones at the easiest difficulty to 6, 10 and finally 15 at d=1. Read operationally, that curve answers a question a fire chief would actually ask, namely how many platforms a given fire demands before extra drones begin to help rather than to thrash. The strip below sits on the wrong side of that line: at d=1 the fire requires 15 drones, and a team of three is far under the saturation point.

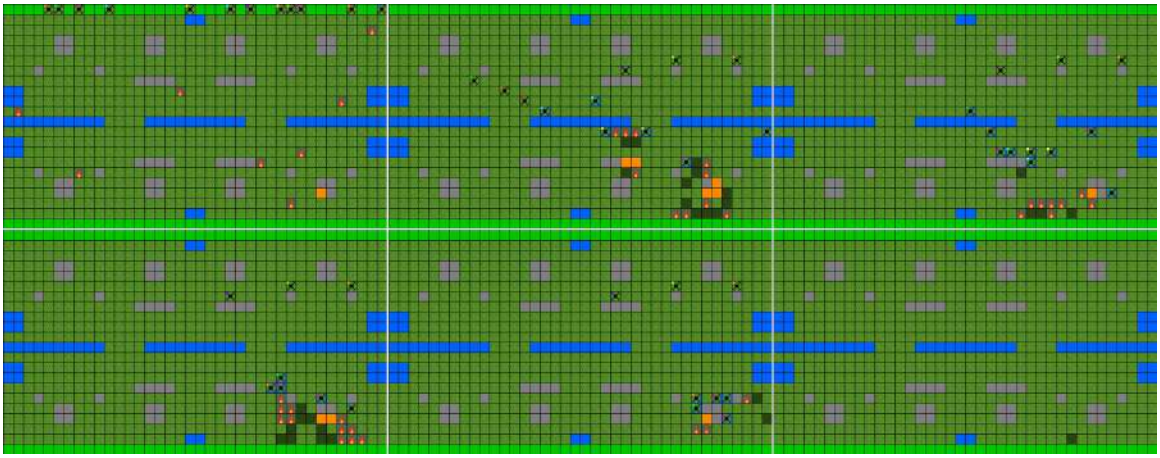

Figure 18: **Out-of-distribution success at N=12 (d=1.0, MAPPO run 7/ckpt 006241, seed 42).** With N=12 drones, two above the N=10 training cap, the policy clears the hardest difficulty in only 73 steps. The strip shows a clear division of labour: about eight drones form a working suppression core with high individual returns ($\approx +50$ each), while the remaining drones contribute little (idling on the periphery or lost) without disrupting that core. The team scales past its training drone count rather than degrading.

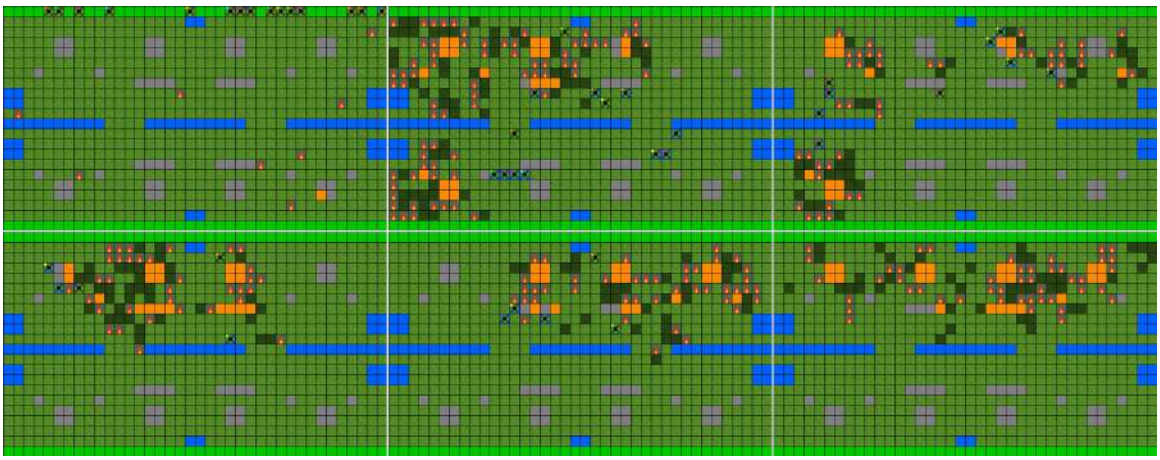

Figure 19: **Out-of-distribution success at N=15 (d=1.0, MAPPO).** At the most extreme evaluated team size (five drones beyond the training cap) the policy still clears the hardest cell, the regime that defines $N_{\text{solved}}(1)$=15 in Table 3. The front is denser and the episode longer than the in-distribution case, but the assignment prior scales: the extra drones plug residual hotspots rather than crowding the same target, which is precisely why clearance keeps rising with N in Figure 4 instead of saturating or collapsing out of distribution.

7.5 Reward hacking on the pre-rebalanced v1 policy

Before presenting the comparison baselines, we include one cautionary strip from *before* the reward redesign. It documents the failure that motivated the Phase-C reward coefficients

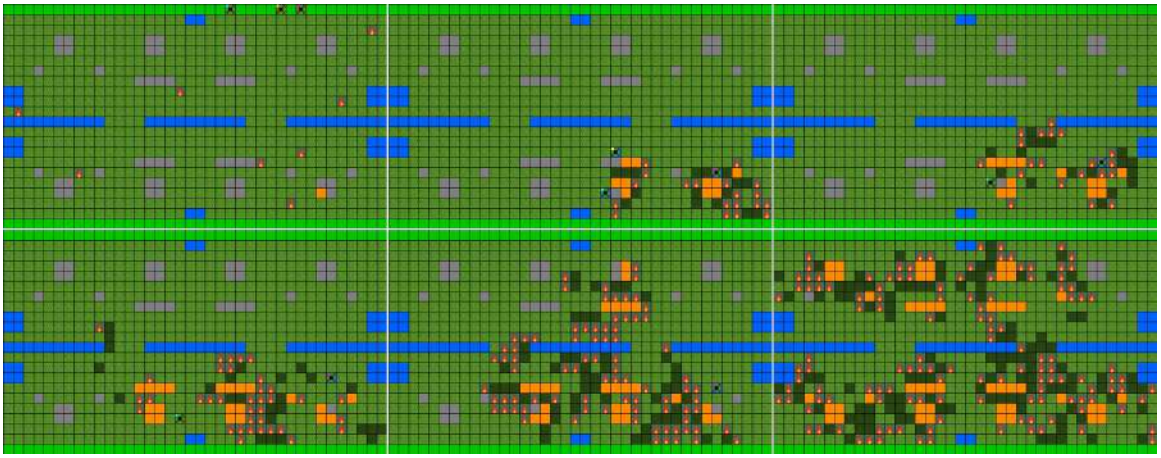

Figure 20: **Undersized team, fire wins (d=1.0, N=3, MAPPO).** Three drones are insufficient at the hardest difficulty, which requires a saturation point of N_{solved}=15 (Table 3): the team is still fighting when the episode times out at T=800, and the per-agent returns collapse asymmetrically (-489, -1447, -3669), with the most-exposed drone accumulating by far the worst losses. The strip confirms visually that the cooperation requirement is real and not an artefact of difficulty alone: an arbitrarily strong policy with too few drones still loses.

used by every other policy in this thesis, and it is a concrete instance of the reward-shaping pitfall discussed in Chapter 5.

7.6 Ablation: removing entity attention

Following the order of the results chapter, we turn from generalisation to the component analyses, starting with the ablation table (Section 6.5). This strip shows the single most damaging ablation, so that the architectural choice it isolates has a behavioural face and not only a number.

7.7 Independent-learner comparison (IPPO): a difference in style

The last pair of strips moves to the algorithm comparison of Section 6.6, where removing the centralized critic (IPPO) costs nothing measurable. Read them as illustrations of how the two solutions allocate effort, not as evidence of a performance gap. Across the full evaluation sweep (300 deterministic episodes per condition) the two are statistically indistinguishable (Welch t=0.90, $p \approx 0.40$).

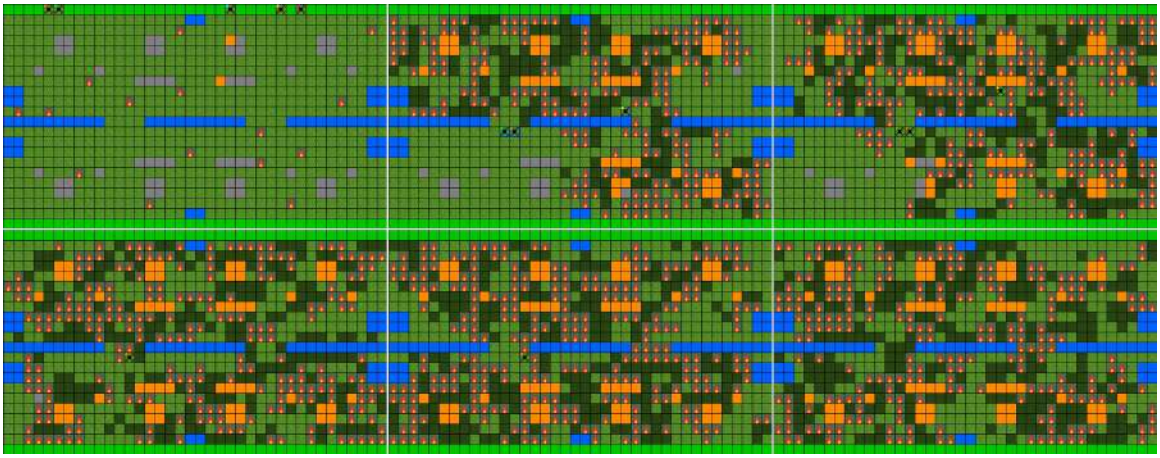

Figure 21: **The pre-rebalanced v1 policy: reward hacking by camping.** The v1 cooperative policy, trained before the reward redesign (the run `run_2_continued_1` of the pre-redesign cooperative environment), learned a degenerate strategy: *loiter beside the last burning cell to keep the per-step "progress" bonus alive instead of extinguishing it.* The episode never terminates by clearance, and the per-agent returns diverge wildly (-78, -372, -1613, -1844, -4919): the campers harvest the small positive progress credit while a single drone absorbs the bulk of the burning-building damage. This is the failure that motivated the rebalanced reward triple (`alpha_terminal`=200, `alpha_progress`=1.0, `alpha_fire_pressure`=-0.15) on which the principal v2 policy is built; the reward-rebalance ablation in Table 4 quantifies how much that redesign is worth (58.7% vs. 83.7% at the hardest cell).

Chapter 8 Conclusions and Future Work

8.1 Conclusions

The central difficulty this thesis confronts is not suppressing a fire with a single agent but getting a whole team of drones to *coordinate*: to divide the front, hand off targets, and manage refills while each agent sees only a local slice of the world. The main conclusion is that this coordination can be learned, and that two ingredients do most of the work. An entity-attention encoder turns each agent's variable-length surroundings into a compact, permutation-invariant representation of the scene, the teammates and the nearest fire cells that bear on its next decision; and a centralised-training, decentralised-execution (CTDE) scheme has every agent learn the *same* policy from a shared training signal. The second point deserves stating carefully. The cooperative behaviour itself is shaped by the team-spirit reward; what CTDE contributes is stable, homogeneous learning across the agents, so that the team converges together instead of falling into the instabilities that make multi-agent training brittle, with one agent running away with the policy while the others never catch up, or one agent so weak that the team cannot work around it. Together these produce cooperative behaviour that holds up as the team grows and as the fire gets harder: trained once on 3

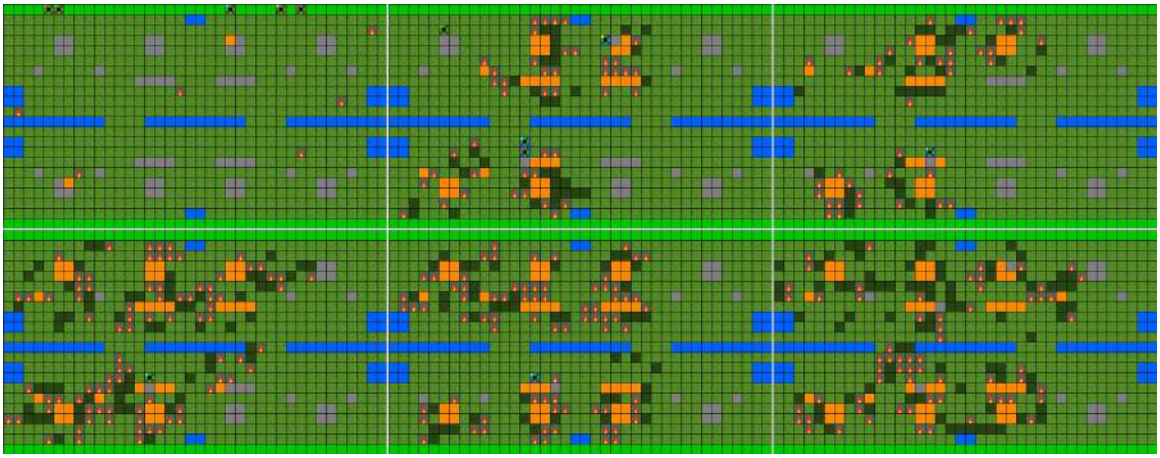

Figure 22: **No-attention ablation (d=0.5, N=5).** With the entity-attention encoder replaced by a masked mean-pool, the policy can no longer tell a nearby fire from a distant one or relate teammates to their assigned targets. The resulting behaviour is conservative and indecisive: drones cluster around a small subset of fires while the rest of the front grows unchecked, and the episode runs much longer without a clean clearance. This is the behavioural face of the most damaging ablation in Section 6.5, where removing entity attention drops hardest-cell clearance from 83.7% to 46.3%, into the noise band of the best scripted heuristic.

to 10 drones, a single network keeps solving the benchmark for substantially larger teams with no retraining (quantified in Section 6.4; the hardest-cell figure is seed-averaged, with seed-to-seed variance that straddles the 80% threshold). These are the emergent collective dynamics a real deployment would need, which is why the result bears on operational wildfire response and not only on the gridworld it was measured on.

The second contribution is the benchmark itself. Within this study every learned policy clearly outperformed the strongest hand-coded controllers we could build, including a Hungarian controller handed the same per-step assignment, which isolates what learning adds over a fixed allocation rule. The component analysis points to entity attention as the single most load-bearing piece, and one comparison was instructive precisely because it came out negative: an independent learner (IPPO) *matched* the centralised-critic policy on every solved-threshold metric, so on this benchmark the coordination is carried by perception and parameter sharing rather than by centralised credit assignment. We report that cautiously, as a comparison of training regimes rather than a clean critic ablation (n=4 vs. 5, $p \approx 0.40$), and leave a critic-only run to future work. What the work leaves behind is its point: a reproducible benchmark, an architecture whose parts have each been characterised, and an honest account of what helped and what did not, which public and private bodies working on emergency response can take as a first step and build on. Against the four secondary objectives set out in Chapter 4, the outcome is as follows:

- **Objective 1 (Benchmark).** *Met.* FireCoopBench is implemented, documented, and

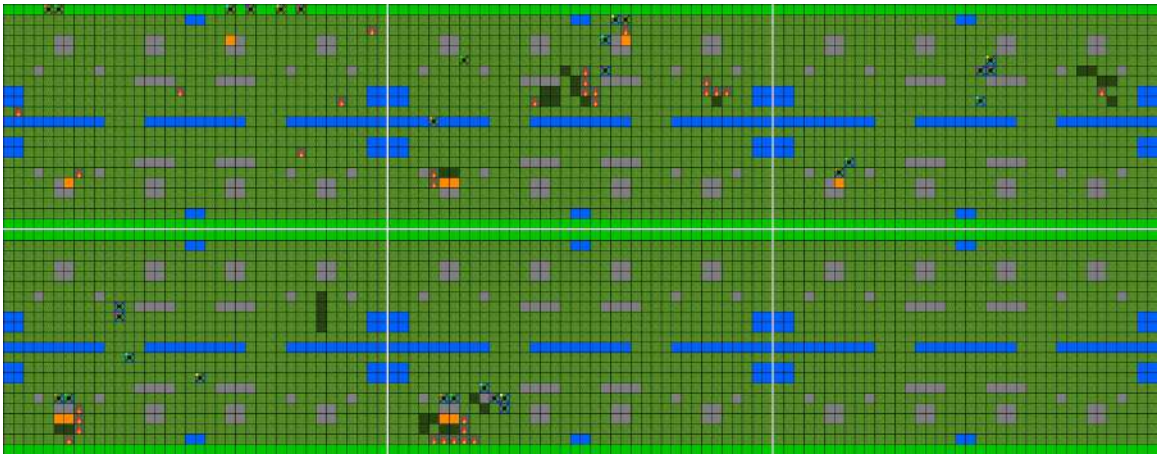

Figure 23: **IPPO mid-difficulty success (d=0.5, N=6, IPPO run 1/ckpt 006241, seed 42).** The independent learner clears the mid-difficulty cell in just 53 steps with *all six* drones surviving and uniformly high returns ($\approx +73$ each): a clean, efficient cooperative success. Compared at the same cell with the centralized policy of Figure 16, this is the visual backing for the headline negative result of Section 6.6: the centralized critic confers no measurable benefit, because parameter-sharing and the entity-attention prior already supply the coordination signal. These are single illustrative episodes; per-episode suppression time varies widely, and across the full sweep IPPO and MAPPO are indistinguishable on suppression time (Section 6.6).

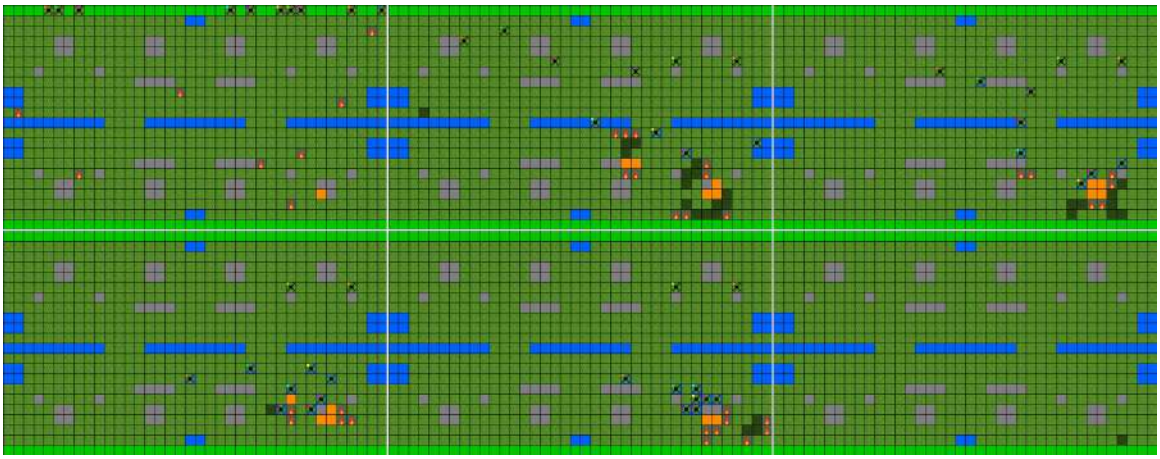

Figure 24: **IPPO at the publishable hard cell (d=1.0, N=10).** The independent learner also clears the hardest in-distribution cell, in 71 steps, essentially matching the centralized policy's 69 steps in Figure 14. On this particular episode the per-agent returns are slightly more dispersed than MAPPO's, but across the full sweep IPPO matches MAPPO on every solved-threshold metric (Section 6.6); the strip illustrates a coordination style, not a performance gap.

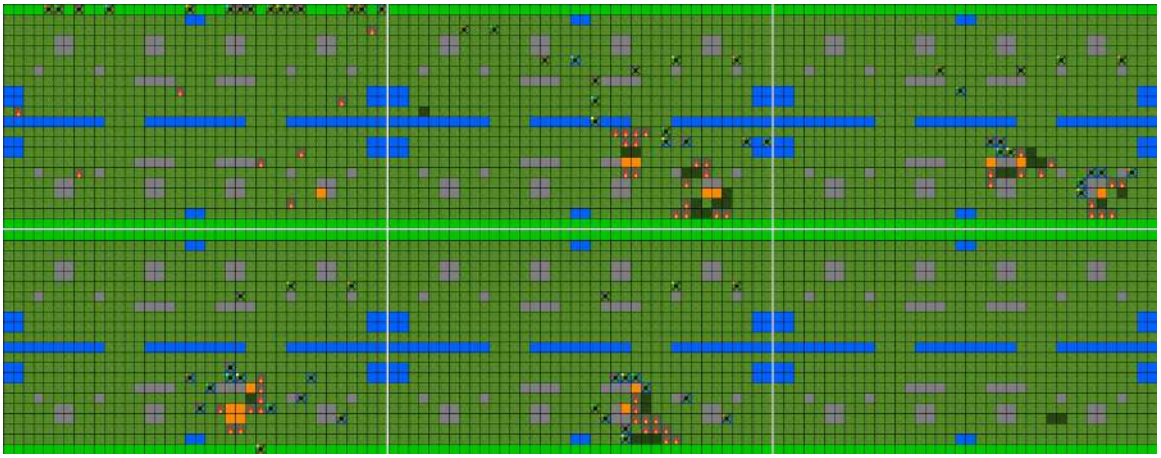

Figure 25: **IPPO out-of-distribution at N=15 (d=1.0).** IPPO generalises to the most extreme team size just as MAPPO does (Figure 19). On this episode the per-agent returns are uneven: drones 1 and 11 finish at −32 and −54 while the rest cluster around +50, a more dispersed division of labour than MAPPO's. This dispersion does *not* translate into any aggregate gap: across the sweep IPPO matches MAPPO on clearance and suppression time alike (Section 6.6). We read it as a difference in how effort is allocated at equal quality, not as a cost of dropping the centralized critic.

structured as a three-environment ladder with a difficulty curriculum and a formal "solved" protocol.

- **Objective 2 (Pipeline).** *Met.* An end-to-end training and evaluation pipeline (configuration, logging, deterministic sweep) is in place and reproduces the headline figures.
- **Objective 3 (Coordination study).** *Met.* The centralized-critic MAPPO policy is compared against IPPO and two heuristic controllers and each architectural component is ablated; the comparison yields a clear positive result over the heuristics and an informative *negative* result over IPPO (centralized training did not improve solved-threshold performance).
- **Objective 4 (Robustness).** *Partially met.* Generalisation to out-of-distribution team sizes and a comparison of difficulty-curriculum vs. domain randomisation are reported; the adversarial-fire (RARL) study is framed as future work below.

8.2 Limitations

Three limitations bound what can be claimed from this work.

Two-dimensional simulation. FireCoopBench is a 2D gridworld. It captures the cooperative-MARL structure of the task (partial observability, variable team size, the assignment problem, and the refill logistics), but real-world airframes fly in continuous time and three-dimensional space, with continuous-valued actuation, and against fire fronts whose dynamics are richer than a parameterised stochastic spread model. What the thesis contributes is a benchmark and a coordination study, not a deployable controller.

Stochastic fire model. The fire model used here is parameterisable and reproducible but not physically grounded; a Rothermel-style spread-rate model on a cellular grid would be more defensible empirically.

Single adversary axis. Robustness is evaluated under out-of-distribution team sizes and per-episode domain randomisation, but a fully adversarial fire (a learned ignition adversary) is left to future work.

8.3 Future work

Transfer to higher-fidelity simulation. The natural next stage is to port the learned coordination strategy to higher-fidelity simulation tools and training paradigms: a continuous-action, three-dimensional setting with more realistic platform dynamics, sensing, and control. The "prototype-in-2D-then-transfer" methodology used throughout this thesis was chosen with exactly this step in mind, so that the coordination logic validated here can be carried over as the realism of the platforms and the environment is raised.

Adversarial fire (RARL). Implementing a learned ignition adversary in the style of robust adversarial RL [43] would stress-test the team against worst-case ignitions and complete the robustness study along the second of Annex B's two robustness axes.

Higher-fidelity fire and perturbations. Replacing the stochastic propagation model with a Rothermel-style spread-rate model on the cellular grid would raise the realism of the environment and let the policy be tested against dynamics closer to those of operational fire-behaviour systems. A related direction is to harden the team against perturbations, from wind and ignition variability to sensor and actuation noise.

Algorithm comparison. Extending the algorithm comparison to a value-factorisation method (QMIX) and to a multi-agent advantage actor-critic (MAA2C) would complete the "four-algorithm" comparison the broader MARL benchmarking community has begun to standardise [9].

Compute profiling. The throughput numbers in Section 6.2 are aggregate. Profiling the environment step task by task, separating the cost of advancing the fire, recomputing the Hungarian assignment, and rebuilding the entity-structured observation, would show where the bottlenecks sit and which are worth optimising before scaling to larger maps or higher-fidelity dynamics.

Chapter 9 References

The works cited throughout this document are listed below.

[1] IPCC, "Climate change 2022: Impacts, adaptation and vulnerability – chapter 13: Europe", Intergovernmental Panel on Climate Change, Tech. Rep., 2022.

[2] V. Mnih *et al.*, "Human-level control through deep reinforcement learning", *Nature*, vol. 518, pp. 529–533, 2015.

[3] D. Silver, A. Huang, C. J. Maddison, *et al.*, "Mastering the game of Go with deep neural networks and tree search", *Nature*, vol. 529, no. 7587, pp. 484–489, 2016. DOI: 10.1038/nature16961.

[4] D. Silver, J. Schrittwieser, K. Simonyan, *et al.*, "Mastering the game of Go without human knowledge", *Nature*, vol. 550, no. 7676, pp. 354–359, 2017. DOI: 10.1038/nature24270.

[5] O. Vinyals, I. Babuschkin, W. M. Czarnecki, *et al.*, "Grandmaster level in StarCraft II using multi-agent reinforcement learning", *Nature*, vol. 575, no. 7782, pp. 350–354, 2019. DOI: 10.1038/s41586-019-1724-z.

[6] C. Berner *et al.*, "Dota 2 with large scale deep reinforcement learning", *arXiv preprint*, 2019. eprint: 1912.06680.

[7] J. Degrave, F. Felici, J. Buchli, *et al.*, "Magnetic control of tokamak plasmas through deep reinforcement learning", *Nature*, vol. 602, no. 7897, pp. 414–419, 2022. DOI: 10.1038/s41586-021-04301-9.

[8] C. Yu, A. Velu, E. Vinitsky, *et al.*, "The surprising effectiveness of PPO in cooperative, multi-agent games", in *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2022. eprint: 2103.01955.

[9] G. Papadopoulos, A. Kontogiannis, F. Papadopoulou, C. Poulianou, I. Koumentis, and G. Vouros, "An extended benchmarking of multi-agent reinforcement learning algorithms in complex fully cooperative tasks", in *AAMAS*, arXiv:2502.04773, 2025.

[10] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd. Cambridge, MA, USA: MIT Press, 2018.

[11] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein, "The complexity of decentralized control of markov decision processes", *Mathematics of Operations Research*, vol. 27, no. 4, pp. 819–840, 2002. DOI: 10.1287/moor.27.4.819.297.

[12] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning", *Machine Learning*, vol. 8, no. 3–4, pp. 229–256, 1992. DOI: 10.1007/BF00992696.

[13] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms", *arXiv preprint*, 2017. eprint: 1707.06347.

[14] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation", in *International Conference on Learning Representations (ICLR)*, arXiv:1506.02438, 2016.

[15] R. Lowe *et al.*, "Multi-agent actor-critic for mixed cooperative-competitive environments", in *NeurIPS*, 2017.

[16] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson, "Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning", in *ICML*, 2018.

[17] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Póczos, R. Salakhutdinov, and A. Smola, "Deep sets", in *NeurIPS*, 2017.

[18] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need", in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017, pp. 5998–6008.

[19] H. W. Kuhn, "The hungarian method for the assignment problem", *Naval Research Logistics Quarterly*, vol. 2, no. 1–2, pp. 83–97, 1955. DOI: 10.1002/nav.3800020109.

[20] S. Hochreiter and J. Schmidhuber, "Long short-term memory", *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. DOI: 10.1162/neco.1997.9.8.1735.

[21] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients", in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018. DOI: 10.1609/aaai.v32i1.11794.

[22] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning", in *Proceedings of the 26th Annual International Conference on Machine Learning (ICML)*, 2009, pp. 41–48. DOI: 10.1145/1553374.1553380.

[23] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world", in *IEEE/RSJ IROS*, 2017.

[24] A. Liñán and F. A. Williams, *Fundamental Aspects of Combustion* (Oxford Engineering Science Series). Oxford: Oxford University Press, 1993.

[25] R. C. Rothermel, "A mathematical model for predicting fire spread in wildland fuels", USDA Forest Service, Tech. Rep. INT-115, 1972.

[26] I. Karafyllidis and A. Thanailakis, "A model for predicting forest fire spreading using cellular automata", *Ecological Modelling*, vol. 99, pp. 87–97, 1997.

[27] B. Drossel and F. Schwabl, "Self-organized critical forest-fire model", *Physical Review Letters*, vol. 69, no. 11, pp. 1629–1632, 1992.

[28] A. Tapley, M. Dotter, M. Doyle, *et al.*, "Reinforcement learning for wildfire mitigation in simulated disaster environments", *arXiv preprint*, 2023. eprint: 2311.15925.

[29] U. Çakır, V.-A. Darvariu, B. Lacerda, and N. Hawes, "Jaxwildfire: A gpu-accelerated wildfire simulator for reinforcement learning", *arXiv preprint*, 2025. eprint: 2512.06102.

[30] E. Seraj *et al.*, "Firecommander: An interactive, probabilistic multi-agent environment for heterogeneous robot teams", *arXiv preprint*, 2020. eprint: 2011.00165.

[31] Z. Samadikhoshkho and M. Lipsett, "Multi-agent reinforcement learning for coordinated aerial wildfire suppression", in *RISEx*, OpenReview: ldJvKeDQ5A, 2025.

[32] K. D. Julian and M. J. Kochenderfer, "Distributed wildfire surveillance with autonomous aircraft using deep reinforcement learning", *Journal of Guidance, Control, and Dynamics*, vol. 42, no. 8, pp. 1768–1778, 2019. DOI: 10.2514/1.G004106.

[33] A. Altamimi, C. Lagoa, J. G. Borges, M. E. McDill, C. Andriotis, and K. Papakonstantinou, "Large-scale wildfire mitigation through deep reinforcement learning", *Frontiers in Forests and Global Change*, vol. 5, p. 734 330, 2022.

[34] R. N. Haksar and M. Schwager, "Distributed deep reinforcement learning for fighting forest fires with a network of aerial robots", in *IEEE/RSJ IROS*, 2018, pp. 1067–1074.

[35] A. Dutceac and C. I. Vizitiu, "An integrated madqn–heuristic framework for swarm robotic fire detection and extinguishing", *Robotics*, vol. 15, no. 1, p. 5, 2026.

[36] M. Collignon, A. Perrusquía, A. Tsourdos, and W. Guo, "Search and rescue operations in wildfires using unmanned aerial vehicles: A multi-agent deep reinforcement learning approach", *Neurocomputing*, vol. 653, p. 131 211, 2025.

[37] G. Papoudakis, F. Christianos, L. Schäfer, and S. V. Albrecht, "Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks", in *NeurIPS Datasets and Benchmarks Track*, 2021.

[38] M. Samvelyan, T. Rashid, C. S. de Witt, *et al.*, "The StarCraft multi-agent challenge", in *AAMAS*, 2019.

[39] J. Z. Leibo, E. A. Duéñez-Guzmán, A. S. Vezhnevets, *et al.*, "Scalable evaluation of multi-agent reinforcement learning with melting pot", in *ICML*, 2021.

[40] M. Carroll, R. Shah, M. K. Ho, *et al.*, "On the utility of learning about humans for human-ai coordination", in *NeurIPS*, 2019.

[41] J. K. Terry *et al.*, "Pettingzoo: Gym for multi-agent reinforcement learning", in *NeurIPS*, 2021.

[42] R. Gorsane, O. Mahjoub, R. J. de Kock, R. Dubb, S. Singh, and A. Pretorius, "Towards a standardised performance evaluation protocol for cooperative MARL", in *NeurIPS*, 2022.

[43] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta, "Robust adversarial reinforcement learning", in *ICML*, 2017.

[44] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping", in *ICML*, 1999.

Appendix A Hyperparameters and training configuration

This annex collects the optimiser, network and training settings used to produce the experimental results of Chapter 6. They are the frozen values of the published five-seed policy (the `mappo_v2` configuration in `config.py`); Table 7 lists them.

Table 7: Hyperparameters of the principal policy (`mappo_v2`).

Group	Parameter	Value
Optimisation	Algorithm	MAPPO (multi-agent PPO)
	Learning rate	3×10^{-4} (Adam)
	Discount γ	0.99
	GAE λ	0.95
	PPO clip ϵ	0.2
	Value-function clip	200
	Entropy coefficient	0.05 const. to 100M, linear $\rightarrow$ 0.02 by 200M
	SGD epochs / update	5
	Mini-batch size	2048
	Train batch size	`num_env_runners` $\times$ 1000 ($\approx$ 40,000)
	Parallel env runners	$\leq$ 40 (A40-safe)
	Total steps / seed	250M
	Training seeds	5 (`run_6` ... `run_10`)
Architecture	Module	`MARLPPOModuleV2` (param.-shared)
	Embedding dim	128
	LSTM hidden / BPTT	128 / 8
	Attention heads	4
	k nearest fires	16
	Feature dims (self/fire/other)	10 / 6 / 5
Counterfactual	COMA advantage	enabled
	Advantage mix ω	0.5 $\quad (1-\omega)\,\mathrm{GAE} + \omega\,\mathrm{COMA}$
	Q-fit loss weight	0.5
Compute	GPUs	4$\times$ NVIDIA A40 (49 GB)
	Layout	one seed per GPU, $\sim$22 h / seed
	Host CPUs	255 (intra-op threads pinned 1/actor)

Baseline variants. The baselines reuse this configuration with minimal, principled changes. **IPPO** swaps the centralized critic for a decentralized one (`MARLPPOModuleV2IPPO`), disables COMA, and sets the team-spirit weight $\tau \equiv 0$ (purely local reward). The **domain-randomisation** policy draws the difficulty $d \sim \mathcal{U}(0,1)$ per episode instead of following

the curriculum. The **heuristic** controllers (greedy, Hungarian, random) carry no learned parameters. All learned baselines are trained at the same 250M-step budget for fairness. A value-factorisation method (QMIX) and a multi-agent advantage actor-critic (MAA2C) are named as future work in Section 8.

Appendix B Per-seed training curves and sweep tables

This annex backs the headline figures and the ablation table of Chapter 6 with per-seed detail. The full 55-cell sweep tables (per-cell mean clearance, survival, time-to-clear and burned-area fraction for every policy) are released with the code as `documentation/figures/*_sweep.csv`; Table 8 reports the per-seed $N_{\text{solved}}(d)$ and hardest-cell clearance for the five seeds of the principal policy.

Table 8: Per-seed $N_{\text{solved}}(d)$ and hardest-cell clearance for the five training seeds of the principal policy. The seed-averaged values reported in Chapter 6 (Table 3) are the conservative aggregate; here two of the five seeds (`run_7`, `run_10`) fall just under the 80% threshold at the hardest cell.

Seed	d=0	d=0.25	d=0.5	d=0.75	d=1	Clr. $(1, 15)$
`run_6`	4	4	6	8	15	91.3%
`run_7`	4	5	6	10	>15	76.3%
`run_8`	3	4	6	8	15	84.7%
`run_9`	3	4	5	8	15	86.3%
`run_10`	4	4	7	10	>15	79.7%
Seed-averaged	4	4	6	10	15	83.7%

On the seed-averaged aggregate. The seed-averaged N_{solved} is computed from the mean clearance across seeds, not by averaging the per-seed N_{solved}, and is therefore the more conservative statistic: at d=0.75, for instance, three seeds individually solve at N=8 but the across-seed mean reaches 80% only at N=10. We report it as the headline precisely because it does not cherry-pick the best seed.