UNIVERSITY OF PADUA

DEPARTMENT OF INFORMATION ENGINEERING
Master's Degree in ICT for Internet and Multimedia Engineering

A Goal-Oriented Communication Approach for Autonomous Mobile Robots

SUPERVISOR
Prof. Andrea Zanella
University of Padova

MASTER CANDIDATE
Mahmoud Mohamed Shaaban Mohamed Shaaban Aly
Student ID 2158012

ACADEMIC YEAR
2025–2026

Dedicated to my Professor Andrea Zanella, all my Professors, my family and friends who supported me throughout this journey, I would not have been here without you!

Abstract

Multi-agent path finding under partial observability requires agents to coordinate from limited local information. Recent learning-based planners address this by allowing agents to exchange learned messages, but they assume unconstrained communication: every agent transmits at every timestep, regardless of whether the message affects any recipient's decision. This is wasteful under realistic bandwidth constraints, and it worsens the chatter problem as team size grows.

This thesis studies goal-oriented communication for decentralized multi-agent path finding, where agents transmit only when transmission serves the goal. We build on SCRIMP, a decentralized policy that couples a small 3×3 field of view with a transformer-based communication block, trained through centralized-training/decentralized-execution using Proximal Policy Optimization and imitation learning. We extend it with a learned communication gate, realized as an additional output head that reads the same fused representation as the policy and value heads, so that each agent decides whether to transmit after having received the messages of its teammates. The decision is sampled from a categorical distribution and trained by a clipped surrogate objective sharing the advantage estimate of the navigation policy, with a penalty applied to the probability of transmitting rather than to the environment reward. A message cache held at the communication block supplies an agent's most recent transmission whenever it stays silent, so that the attention computation receives one entry per agent and no bandwidth is consumed.

Two training strategies are evaluated for the tendency of agents to suppress communication before their messages become informative: a curriculum that introduces the penalty only after navigation and coordination have been learned, and an alternation between penalized and unpenalized episodes in which the policy observes which regime applies. All experiments are conducted in a Python simulation environment and compared against unmodified SCRIMP across fifteen scenarios spanning team sizes from 8 to 128 agents and obstacle densities up to 30%. Results show that communication can be reduced by more than 97% while the success rate is preserved across the vast majority of scenarios, indicating that the great majority of the messages exchanged under an unconditional scheme carry no bearing upon the decisions of their recipients.

Contents

Abstract **vii**

List of Figures **xi**

List of Tables **xiii**

1 Introduction **1**
- 1.1 Motivation and Background . . . 1
- 1.2 Problem Statement . . . 1
- 1.3 Thesis Contributions . . . 2
- 1.4 Thesis Organization . . . 3

2 Goal-Oriented Communications **5**
- 2.1 From Data-Centric to Goal-Oriented Networking . . . 5
- 2.2 Semantic Communication . . . 6
- 2.3 Goal-Oriented versus Semantic Communication . . . 8
- 2.4 Knowledge Bases and Semantic Noise . . . 9
- 2.5 Measuring Effectiveness . . . 10
- 2.6 Goal-Oriented Communication in Multi-Agent Systems . . . 10
- 2.7 Goal-Oriented Communication Related Work . . . 12
- 2.8 Goal-Oriented Communication Research Gap . . . 13

3 Multi-Agent Path Planning **15**
- 3.1 The Path Planning Problem . . . 15
  - 3.1.1 Classical Approaches . . . 16
  - 3.1.2 Metaheuristic Techniques . . . 16
  - 3.1.3 Learning-Based Path Planning . . . 17
- 3.2 From Single-Agent to Multi-Agent Path Finding . . . 20
- 3.3 Multi-Agent Reinforcement Learning for Path Finding . . . 22
- 3.4 Multi-Agent Path Planning Related Work . . . 22
- 3.5 Open Challenges and Research Gap . . . 24
  - 3.5.1 Open Challenges . . . 24
  - 3.5.2 Research Gap in Communication for Multi-Agent Path Finding . . . 24

4 Architecture and Methodology **27**
- 4.1 The SCRIMP Baseline . . . 27
  - 4.1.1 Problem Setting and Observation Space . . . 27

4.1.2 Network Structure . . . 28
4.1.3 Learning Techniques . . . 29
4.1.4 Reported Performance . . . 33
4.1.5 Communication Mechanism Ablation . . . 35
4.2 Learned Communication Gating . . . 36
4.2.1 The Gating Module . . . 37
4.2.2 Sampling the Gate . . . 37
4.2.3 Objective . . . 39
4.2.4 Message Repetition . . . 40
4.3 Training Strategies . . . 40
4.3.1 Alternating Penalized and Unpenalized Episodes . . . 41
4.3.2 Curriculum Learning . . . 41
4.4 Summary . . . 41

5 Experiments and Results **43**
5.1 Experimental Setup . . . 43
5.1.1 Common Training Regime . . . 43
5.1.2 Experiment 1: Curriculum Over Task Difficulty . . . 44
5.1.3 Experiment 2: Alternating Penalized and Unpenalized Episodes . . . 44
5.2 Evaluation Protocol and Metrics . . . 45
5.3 Results . . . 46
5.3.1 The Alternating Model . . . 47
5.3.2 The Curriculum Model . . . 47
5.4 Discussion . . . 48
5.4.1 The Cost of Silence . . . 48
5.4.2 Where the Gate Transmits . . . 48
5.4.3 Why Gating Improves Performance in Dense Instances . . . 49
5.5 Results Analysis . . . 49
5.5.1 Measures of Communication Behaviour . . . 49
5.5.2 Communication Is Allocated Rather Than Merely Reduced . . . 50

6 Conclusion and Future Work **53**
6.1 Summary . . . 53
6.2 Future Work . . . 54
6.2.1 Event-Triggered Execution . . . 54
6.2.2 Bandwidth as an Explicit Constraint . . . 54
6.2.3 Deciding to Whom to Transmit . . . 55
6.2.4 Deciding How Precisely to Transmit . . . 55

References **57**

Disclaimer on the Use of AI **61**

List of Figures

2.1 The three levels of communication. 6
2.2 Classical, semantic-oriented and goal-oriented communication. 8
2.3 Knowledge base alignment and semantic noise. 9
2.4 Centralized and decentralized task-oriented communication. 12

3.1 Families of path planning methods. 16
3.2 General deep reinforcement learning architecture. 19

4.1 Network structure of SCRIMP. 29
4.2 Target symmetry. 32
4.3 SCRIMP extended with the communication gate. 38

5.1 Communication behaviour across team sizes and obstacle densities. 50

List of Tables

2.1 Driving 6G applications and their industry verticals. 7
2.2 6G use cases and their associated service classes. 7
2.3 Metrics for semantic and goal-oriented communication. 11

4.1 SCRIMP reward structure. 28
4.2 SCRIMP hyperparameters. 33
4.3 Reported performance of SCRIMP. 35

5.1 Training stages of the curriculum experiment. 44
5.2 Evaluation results for teams of up to 32 agents. 46
5.3 Evaluation results for teams of 64 and 128 agents. 47

List of Acronyms

6G sixth generation

AoI age of information

CTDE centralized training with decentralized execution

DRL deep reinforcement learning

FOV field of view

GoC goal-oriented communication

GRU gated recurrent unit

IoV Internet of Vehicles

ISAC integrated sensing and communications

JSC joint source-channel

KB knowledge base

LSTM long short-term memory

MAPF multi-agent path finding

MARL multi-agent reinforcement learning

MAS multi-agent system

PPO proximal policy optimization

RL reinforcement learning

SNR signal-to-noise ratio

UAV unmanned aerial vehicle

VoI value of information

1 Introduction

1.1 Motivation and Background

For seven decades, communication networks have been engineered around a single objective inherited from Shannon's framework: the reliable delivery of bits, irrespective of the meaning those bits carry or the use to which they are eventually put. This abstraction served the field well while the endpoints of communication were predominantly humans consuming content, but it holds far less well now that a growing share of network traffic is generated and consumed by machines that must act upon what they receive, under stringent constraints on bandwidth, energy, and latency [1]. Goal-oriented communication (GoC) surmounts this limitation by shifting from a data-centric paradigm to a pragmatic one, transmitting only the contextually relevant features essential for successful goal execution, where system performance is assessed by the correctness of the receiver's resulting action rather than by the fidelity of the reconstructed signal [2].

Multi-agent systems (MASs) constitute one of the clearest settings in which this shift becomes necessary. A team of autonomous robots pursuing a shared objective must coordinate on the basis of partial, local observations exchanged over links that are bandwidth-limited and frequently unreliable; communication is precisely what allows these agents to align their beliefs and actions, yet it is simultaneously a cost that scales poorly as team size grows. The governing design question is therefore not how faithfully a message can be delivered, but rather *what* is worth communicating, *when*, and *with whom* [3].

1.2 Problem Statement

Multi-agent path finding (MAPF) is a concrete instance of this question. Learning-based MAPF planners achieve scalability by restricting each agent to a local field of view (FOV), which decouples the learned policy from world size but simultaneously deprives agents of the information required to resolve deadlocks and avoid congestion. Inter-agent communication is the established remedy, and recent work has demonstrated

that a learned communication channel can recover much of the performance forfeited to partial observability. SCRIMP [4] is representative of this line of work: agents observe only a 3×3 neighbourhood and exchange learned messages through a transformer-based communication block, attaining success rates competitive with centralized planners even on densely populated instances.

However, a critical limitation remains: these planners communicate unconditionally, with every agent emitting a message at every timestep regardless of whether that message alters any recipient's decision. The consequences are twofold. Such a scheme assumes a communication budget that no realistic deployment can provide, and it worsens the chatter problem, in which agents become overwhelmed by messages that function as noise rather than as information as team size increases. There is thus a direct mismatch between how these systems are constructed and what the goal-oriented paradigm prescribes they should do.

This thesis addresses that mismatch. We adopt SCRIMP as our base architecture and augment it with a learned gate that permits each agent to decide, at every timestep, whether to send its message to the communication transformer at all. The gate is trained jointly with the path planning policy against a weighted sum of task reward and communication cost, following the formulation of [5], and is rendered differentiable through the straight-through Gumbel-Softmax reparameterization. Since agents that learn to fall silent early in training never subsequently learn to construct informative messages, we additionally adopt the message-repeat mechanism proposed in the same work: when an agent decides not to transmit, its most recently transmitted message is supplied to the communication transformer in place of a new one, so that the encoded representation distributed to the team remains complete. Two further training strategies are then evaluated against this baseline.

1.3 Thesis Contributions

The main contributions of this thesis are:

1. Goal-oriented formulation of communication in decentralized MAPF: we cast decentralized MAPF with learned communication as a GoC problem, adapting the communication mechanism of SCRIMP so that the value of a message is quantified by its contribution to goal reaching and collision avoidance rather than by its fidelity.

2. A learned communication gate: we extend the SCRIMP architecture with a per-agent, per-timestep gate based on the straight-through Gumbel-Softmax reparameterization, trained jointly with the path planning policy under an explicit communication penalty.

3. Training stabilization and strategy comparison: we stabilize training under this penalty by supplying an agent's last transmitted message to the communication transformer whenever that agent stays silent, and we evaluate two additional training strategies, namely alternating between penalized and unpenalized episodes and a curriculum over task difficulty, comparing both against the unmodified SCRIMP baseline in terms of success rate, path quality, and communication volume.

4. Substantial reduction in communication: we show that the communication rate can be reduced by more than 99% with respect to the unmodified SCRIMP baseline while the success rate remains essentially unaffected.

1.4 Thesis Organization

The remainder of this thesis is organized as follows:

Chapter 2: Introduces goal-oriented communication, tracing the progression from Shannon's technical level to the semantic and effectiveness levels, distinguishing goal-oriented from semantic communication, and reviewing the state of the art in goal-oriented communication for multi-agent systems.

Chapter 3: Addresses multi-agent path finding, spanning classical centralized planners through to learning-based decentralized approaches with communication.

Chapter 4: Presents our architecture and methodology, describing both the SCRIMP baseline and the modifications introduced in this work.

Chapter 5: Reports the experimental setup and the results obtained.

Chapter 6: Summarizes the key findings and outlines directions for future work.

2

Goal-Oriented Communications

This chapter introduces the communication paradigm on which this work rests. Section 2.1 traces the shift from the data-centric view of communication towards a pragmatic one and recalls the three levels of communication identified by Shannon and Weaver; Section 2.2 and Section 2.3 describe semantic communication and establish how goal-oriented communication differs from it; Section 2.4 covers knowledge bases and semantic noise; Section 2.5 surveys the metrics through which effectiveness is quantified; Section 2.6 addresses multi-agent systems; and Section 2.7 reviews the state of the art, closing with the gap that this work addresses.

2.1 From Data-Centric to Goal-Oriented Networking

Every generation of wireless networks from the first to the fifth has been designed around the same set of targets, namely higher data rates, lower bit error rates, and greater capacity, and this focus is a direct consequence of the way the problem was framed in 1948, when Shannon set out to determine the conditions under which a sequence of symbols can be transmitted reliably over a noisy channel and deliberately set the meaning of those symbols aside as irrelevant to the engineering problem [1]. The framing proved extremely productive, and the majority of the advances of the last seven decades, from multiple-input multiple-output transmission to modern channel coding, were built upon it.

The idea that communication admits more than one level is nevertheless as old as the theory itself. In their joint presentation of the work, Shannon and Weaver distinguished three levels at which the problem of communication can be posed, illustrated in Figure 2.1: the technical level asks how accurately the symbols of communication can be transmitted; the semantic level asks how precisely the transmitted symbols convey the intended meaning; and the effectiveness level asks how well the received meaning affects conduct in the desired way [1], [6].

Shannon confined his attention to the technical level, which was an entirely reasonable decision at the time, since it was precisely this restriction that rendered a rigorous probabilistic theory attainable; the re-

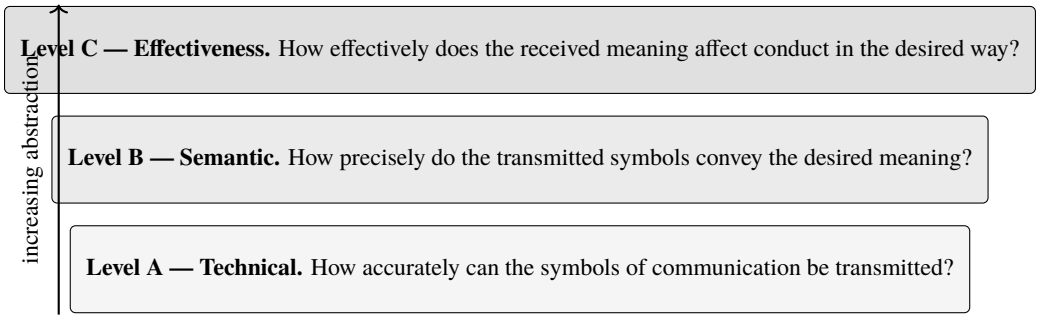

Figure 2.1: The three levels of communication identified by Shannon and Weaver. Classical communication theory addresses Level A alone; semantic communication targets Level B, and goal-oriented communication targets Level C. Adapted from [2].

maining two levels were not dismissed as unimportant but merely set aside for later treatment. What has changed in the intervening decades is that the semantic and effectiveness levels can now be addressed with tools that did not exist in 1948, and in particular with machine learning models capable of extracting meaning from raw data and of learning what a receiver requires in order to act correctly, so that semantic communication takes up Level B while goal-oriented communication takes up Level C.

The rapid development of new hardware technologies, particularly in the sensing and autonomous robot domains, together with the breakthroughs of AI are shaping new application scenarios that call for a radical shift in the way communication networks are conceived and designed [1]. New networks need to support agentic AI communications, real-time dynamic interactions between autonomous robots, secure and reliable communication for safety-critical cyber-physical applications [6]. GoC surmounts these constraints by shifting from a traditional data-centric paradigm to a pragmatic framework, transmitting only the contextually relevant features essential for successful task execution [2]. By prioritizing message effectiveness over raw, indiscriminate data delivery, GoC drastically compresses signaling overhead, minimizes end-to-end latency, and optimizes network resource efficiency.

The breadth of the application space that motivates this shift is considerable. Table 2.1 lists a representative selection of the applications expected to drive sixth generation (6G) together with the industry verticals they serve, while Table 2.2 groups emerging use cases according to the classes of service they require, where the semantic class is precisely the one introduced to support applications in which knowledge is shared between the interacting parties and in which the endpoints may be machines rather than humans.

2.2 Semantic Communication

Semantic communication aims at delivering the meaning of a message rather than its exact form, so that the transmitter applies a semantic encoder which extracts the core content of the source data and discards redundant detail, while the receiver applies a semantic decoder which reconstructs that meaning from whatever it receives [6]. Two messages are regarded as equivalent whenever they convey the same meaning, even if the symbols used to express them are entirely different, which constitutes a clear departure from the classical view in which the transmitted and the received sequences are compared symbol by symbol.

Table 2.1: Representative 6G applications and the industry verticals for which they are relevant, where • denotes relevance to a given vertical. Adapted from [2].

Application	e-Health	Automotive	Energy	Media	Factory	Transport
Haptic communication	•			•		
Massive IoT integrated smart city	•	•	•	•	•	•
Automation and manufacturing			•		•	•
Multisensory XR applications	•			•		•
Connected robotics and autonomous systems	•	•				•
Brain–computer interfaces	•			•		
The Internet of no things	•	•		•	•	•
Industrial IoT			•		•	•
Internet of robots		•			•	•
Flying vehicles		•				•
Accurate indoor positioning	•	•		•	•	•
Holographic and tactile communications	•	•		•	•	
Industry 5.0			•		•	

Table 2.2: Emerging use cases grouped by the classes of 6G service they require: massive machine-type communications supporting distributed intelligence (MMTCxDI); globally enhanced mobile broadband (GeMBB); ultra-reliable low-latency computation, communication and control (URLLCCC); and semantic services. Adapted from [1].

Use case	MMTCxDI	GeMBB	URLLCCC	Semantic
Remote areas connectivity		•		
Precision and personalized digital health	•		•	
Precision agriculture	•	•	•	
Unmanned mobility	•	•	•	
Autonomous vehicle mobility	•	•	•	
Multi-sensory immersive extended reality			•	•
Holographic communication and telepresence			•	•
Human-centric AI support			•	•
Brain-to-computer interactions			•	•
Wireless robotics	•		•	•
High-precision manufacturing and remote control	•	•	•	•
Smart digital-twin environments	•	•	•	•
Bidirectional intelligence intertwining				•

The gain arises from shared prior knowledge, since if transmitter and receiver already hold a common model of the domain, whether a natural language, a set of object categories, or a learned representation, then a substantial part of the message need not be transmitted at all because the receiver is able to recover it from what it already knows. This explains why semantic schemes perform particularly well in the low signal-to-noise ratio (SNR) regime, where errors introduced at the symbol level can frequently be corrected at the semantic level by exploiting the structure that both sides share.

Semantic and technical errors are moreover distinct phenomena, in that an error at the symbol level does not necessarily produce an error in meaning, since the receiver may still interpret the message correctly despite a number of corrupted symbols, whereas conversely a message decoded without a single bit error can still be misunderstood if the two sides do not share the same interpretation of it; this second case is treated in Section 2.4.

2.3 Goal-Oriented versus Semantic Communication

Semantic and goal-oriented communication are closely related paradigms that are frequently treated together, yet they are not equivalent. Semantic communication is concerned with the meaning of the source data and seeks to preserve it accurately, whereas goal-oriented communication is concerned with a specific task and seeks to preserve only what that task requires [3], so that goal-oriented communication may be regarded as the narrower of the two, adopting a pragmatic view in which the goal itself determines what qualifies as relevant information [2]. This characterization, which is the one adopted in the greater part of the literature, positions goal-oriented communication as a specialization of semantic communication. An alternative reading is equally defensible and in some respects more general, namely that goal-oriented communication subsumes the other two levels rather than narrowing them, since an appropriate choice of goal and knowledge base recovers each as a particular case: a goal defined as the faithful recovery of the meaning intended by the source yields semantic communication, while a goal defined as the faithful reproduction of the transmitted symbols yields the technical level of Shannon's framework.

Figure 2.2 compares the corresponding system models. In the classical model the transmitter is blind to the content it carries and the objective is to reproduce the source data at the destination; in the semantic-oriented model, semantic representation and encoding replace source encoding and the output is the recovered meaning of the message; and in the goal-oriented model the communication goal enters the encoding process itself, since a semantic filtering stage removes information irrelevant to the current goal before anything is transmitted, and the output of the system is not a reconstructed message but an action to be performed.

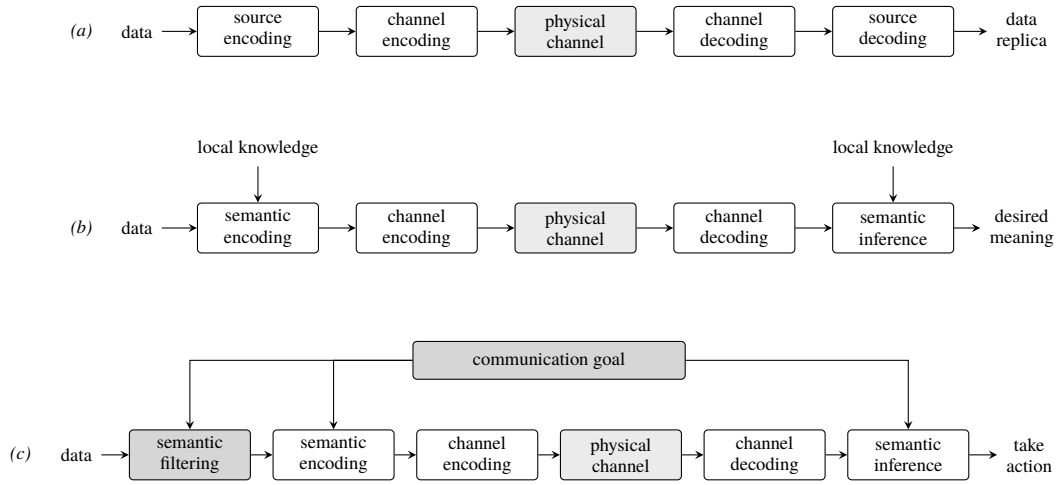

Figure 2.2: Comparison of communication system models. The classical model (a) is blind to content; the semantic-oriented model (b) recovers meaning using local knowledge at both ends; the goal-oriented model (c) additionally filters information according to the communication goal and outputs an action rather than a reconstruction. Adapted from [6].

The practical distinction is most readily appreciated through an example. Consider an image transmitted for use within a transportation system: under semantic-oriented communication the encoder is not tied to

any particular task and must therefore retain the features required by every task the receiver might subsequently perform, whether classifying objects, detecting a specific target, or simply reproducing the scene, whereas under goal-oriented communication only the features demanded by the current task are extracted and transmitted, which may amount to a small fraction of the image, and the output at the receiver is not a set of feature maps but a concrete instruction such as braking or altering the steering angle [6]. The same principle governs teams of mobile robots, where the messages exchanged exist solely to help each agent find its path to its goal without collisions.

2.4 Knowledge Bases and Semantic Noise

Both semantic and goal-oriented systems depend upon knowledge that transmitter and receiver hold in common, and this shared knowledge is customarily described as a knowledge base, which contains the representations employed, the goals of the communication task, and the reasoning rules that all participants are able to apply [6]. The knowledge base (KB) is what permits a short message to carry a large quantity of information, since the transmitter may omit anything the receiver is capable of reconstructing on its own.

The difficulty is that the two knowledge bases are not guaranteed to agree, and if the source KB and the destination KB drift apart, the receiver may interpret a correctly decoded message in a manner the transmitter never intended. This effect is known as semantic noise and is distinct from channel noise, in that it can arise even when no symbol has been corrupted in transit [2], [6]. Figure 2.3 illustrates the situation: only the region in which the two knowledge bases overlap supports reliable interpretation, and messages depending upon the non-overlapping regions remain exposed to semantic noise.

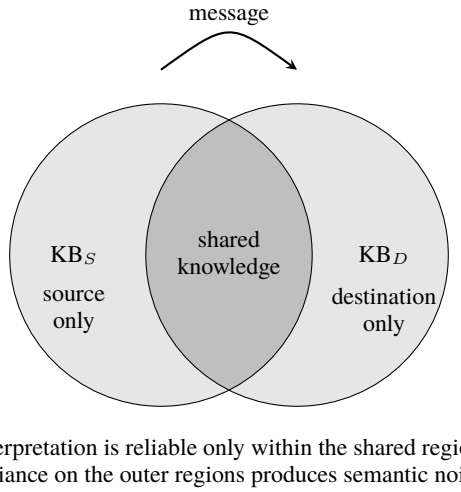

Figure 2.3: Alignment between the knowledge base at the source, KB$_S$, and at the destination, KB$_D$. Messages relying only upon shared knowledge are interpreted as intended, whereas messages depending upon knowledge held by one side alone may be misinterpreted even in the complete absence of channel errors.

Maintaining alignment between knowledge bases is straightforward in systems whose participants are trained jointly upon the same data, and considerably more demanding when agents are added, replaced, or

trained separately, which is why synchronization of the shared KB through an authoritative third party or a virtual knowledge base is a recurring element of the architectures reviewed in Section 2.7.

2.5 Measuring Effectiveness

Because goal-oriented systems are assessed by outcomes rather than by fidelity, the metrics inherited from classical communication are of limited use, since bit error rate, throughput, and delay treat every bit as equally important, which is precisely the assumption the paradigm rejects. The literature has consequently developed families of metrics defined at the semantic and effectiveness levels, and these are conventionally grouped into error-based, age-based, and value-based families [6], summarized in Table 2.3.

Error-based metrics quantify whether the meaning intended by the transmitter is equivalent to the meaning recovered at the destination, and they remain task-specific, so that no single measure serves every data type. Age-based metrics instead quantify the freshness of the information available at the receiver, and the age of information (AoI) is defined at time t as

$$\Delta(t) = t - u(t), \tag{2.1}$$

where $u(t)$ denotes the generation time of the freshest update successfully received by time t, so that $\Delta(t)$ measures the staleness of the receiver's view of the system rather than the delay experienced by any individual packet [3]. From this instantaneous quantity two long-run measures are obtained, namely the average AoI,

$$\bar{\Delta} = \lim_{T\to\infty} \frac{1}{T} \int_0^T \Delta(t)\, dt, \tag{2.2}$$

and the peak AoI, which averages the value attained immediately before each new update arrives and therefore characterizes the worst-case staleness encountered between successive receptions [3]. Value-based metrics, finally, quantify what a particular piece of information is worth to a decision that is about to be taken, and the value of information (VoI) has been employed to suppress updates that would not alter the receiver's behaviour [7], [8].

None of the three families is sufficient in isolation, since error-based measures ignore timing, age-based measures ignore content, and value-based measures are generally computable only once the task has been completed, which has motivated combined metrics such as the age of incorrect information, which penalizes the duration for which the receiver's estimate remains wrong rather than merely stale, and the query age of information, which measures freshness only at the instants at which the receiver actually requires the data [6].

2.6 Goal-Oriented Communication in Multi-Agent Systems

Recent years have witnessed a growing interest in employing networks of autonomous agents to solve complex tasks collaboratively, and such systems, known as multi-agent systems, achieve global objectives through localized interactions and decentralized decision-making among the individual agents [3]. Modern MASs are increasingly deployed in dynamic, distributed and communication-constrained environments,

Table 2.3: Families of performance metrics employed in semantic and goal-oriented communication, with the level of communication each addresses. Adapted from [3], [6].

Family	Representative metrics	Quantity captured
Error-based	BLEU, CIDEr, sentence similarity; PSNR, SSIM; SDR, PESQ	Semantic similarity between the message emitted by the source and that recovered at the destination, evaluated per data type rather than symbol by symbol. Addresses the semantic level.
Age-based	AoI $\Delta(t) = t - u(t)$; average AoI; peak AoI; age of incorrect information	Freshness of the receiver's view of the system, and in the combined variants the duration for which that view remains incorrect. Bridges the semantic and effectiveness levels.
Value-based	VoI; query AoI; task success rate	Relevance of a given piece of information to the decision the receiver is about to take, and ultimately the degree to which the goal is accomplished. Addresses the effectiveness level.

and applications such as autonomous vehicle coordination, distributed sensing and estimation, and swarm robotics demand not only local intelligence but also effective communication among the agents, since communication is what enables coordination and information sharing when the local observations available to each agent are limited or partial.

Several fundamental challenges nevertheless hinder effective communication in these systems [3]. Bandwidth limitations restrict how much information can be exchanged in real time, particularly over wireless links, and classical approaches that rely upon predefined protocols or fully connected message-passing structures assume unrestricted communication and therefore struggle in resource-constrained settings, treating all information as equally valuable and consuming bandwidth on messages that contribute nothing to the global task. The distributed architecture of MASs means that there is frequently no centralized controller possessing global knowledge, so that each agent must decide on the basis of partial, noisy or outdated information, and communication becomes a tool not merely for sharing state but for aligning beliefs and coordinating actions under uncertainty. Partial observability further complicates message design, because agents holding incomplete or private observations of the environment must infer what the others already know, anticipate their actions, and tailor their messages accordingly, which creates a form of nested reasoning that is both computationally and communicatively demanding. A further limitation lies in the separation of communication and decision-making, since in many standard architectures agents communicate first and act afterwards regardless of whether the exchanged information was critical to the decision, which forgoes the benefit of jointly optimizing messaging and action policies in service of a common task. Finally, as the heterogeneity of agents increases, fixed communication topologies or uniform message policies produce redundant information sharing, overload, or irrelevance, and thereby degrade coordination.

These constraints motivate a rethinking of communication strategies, in which the question is not simply how to convey information accurately but rather *what* is worth communicating, *when*, and *with whom*, given the task and the context [3]. Learning-based approaches address these questions by treating communication as part of the policy rather than as a fixed protocol, so that multi-agent reinforcement learning (MARL) allows agents to learn communication and control jointly, generally under a centralized training with decentralized execution (CTDE) paradigm, while attention or graph-based mechanisms permit each agent to

weigh incoming messages according to their relevance to its own decision. Sparsity is encouraged either by penalizing communication directly or by constraining the number of messages an agent may send, with the effect that transmitting becomes an action carrying a cost rather than something occurring by default.

The architecture through which agents communicate is likewise consequential, and Figure 2.4 contrasts the two principal arrangements. In the centralized case agents forward their information to a coordinating node which fuses it and returns instructions, an arrangement that affords the most complete global view but concentrates computation at a single point and introduces a synchronization bottleneck as the team grows; in the decentralized case agents exchange information directly with their neighbours, which scales considerably better and removes the single point of failure at the cost of each agent possessing only a local view.

Figure 2.4: Two arrangements for task-oriented communication in a multi-agent system. Centralized coordination routes all information through a single node, whereas decentralized coordination relies upon direct exchanges between agents. Adapted from [6].

2.7 Goal-Oriented Communication Related Work

Several recent frameworks apply goal-oriented principles to robotic and vehicular systems, and together they delimit the ground upon which the present work is built.

Cooperative semantic communications (Co-SC) [9] targets traffic analysis in the Internet of Vehicles (IoV). Its key components are: 1) *semantic encoders* extracting task-relevant features from the raw data collected by each vehicle; 2) *deep joint source-channel (JSC) encoders* mapping those features to channel symbols while implicitly protecting the vital ones by allocating them a greater number of symbols; 3) *cooperative JSC decoders* exploiting the correlation between users so as to separate shared from distinct semantics; and 4) *semantic decoders and task modules* performing the final object retrieval. Because nearby vehicles frequently observe overlapping scenes, the cooperative decoder is able to use the symbols transmitted by one vehicle to assist in decoding those of another, thereby narrowing the uncertainty space and correcting errors that a single-user decoder could not. Co-SC attains a 28.9% rank-1 accuracy gap over a deep learning baseline at -3 dB SNR, which demonstrates the magnitude of the gain obtainable by encoding for the task rather than for the data.

Cross-layer orchestration (GoAgentNet) [10] provides holistic cross-layer cooperation through agentic orchestration and closed-loop feedback, supported by a virtual knowledge layer built upon a knowledge

graph, which transforms fragmented domain actions into autonomous goal-oriented orchestration in which specialized agents collaborate and negotiate in order to realize a unified intent while optimizing their own local objectives. In a robotic fault recovery task conducted under 5 MHz of bandwidth, avoiding the transmission of raw point-cloud streams reduced energy consumption by more than 99% and improved task success by 72%.

The Aware-ISAC framework [8] addresses obstacle avoidance for unmanned aerial vehicles (UAVs), in which an integrated sensing and communications (ISAC) base station senses the trajectories of the vehicles, constructs a global view of the environment, and transmits downlink command-and-control signals. An aware deep Q-network evaluates the VoI before transmitting and thereby avoids continuous signalling, reducing the number of transmitted signals by 92.4% and the number of occupied time slots by 85.5% while maintaining a 100% success rate.

Control-centric goal-oriented communication [7] applies the same principle to remote robot control, where a deep reinforcement learning (DRL) controller transmits only task-critical commands and the receiver applies an AoI and VoI policy permitting urgent packets to override stale ones held in the queue. This reduced trajectory tracking error by 91.5% relative to periodic control and established a baseline for the temporal prioritization of semantic messages.

Pragmatic MARL control [11] addresses the observation that communication and control are conventionally designed as separate modules, with the communication subsystem optimized for throughput or reliability irrespective of the control task it serves. The authors formulate the joint problem as a cyber-physical partially observable Markov decision process, in which the state comprises both the physical process being controlled and the state of the communication system, and the action space encompasses both the control actions applied to the plant and the scheduling decisions governing which sensor transmits and when. Because the resulting problem is intractable by dynamic programming at realistic scale, it is solved through MARL, with the communication and control agents trained jointly rather than separately. The central result is that co-design produces an implicit coordination of edge actions even under severe communication constraints, in the sense that agents learn to transmit precisely when their observations would change the control decision, without any explicit scheduling rule being imposed.

Joint MARL over noisy channels [12] takes the complementary view, observing that the great majority of learned communication schemes for MASs assume an idealized channel over which messages arrive intact, an assumption that fails whenever agents communicate over a wireless medium. The authors therefore place a noisy physical channel between the transmitting and receiving agents during training itself, so that the learned message representation must be robust to the distortion the channel introduces rather than merely informative in the abstract. Message encoding and the cooperative policy are learned end to end, which allows the encoder to allocate its limited channel uses to the components of the message that matter most for the joint task, and the resulting scheme sustains coordination at channel qualities under which schemes trained on noiseless assumptions degrade sharply.

2.8 Goal-Oriented Communication Research Gap

Taken together, these frameworks demonstrate the isolated potential of pragmatic networking, yet several critical gaps remain at their intersection. The existing literature features either static, centralized semantic

architectures [9] or idealized multi-agent state sharing that presupposes error-free exchange [12], and current models lack adaptive mechanisms capable of shifting the communication pattern in real time according to changing environmental demands. Furthermore, frameworks built upon temporal and functional gating mechanisms [7], [8] optimize point-to-point links but do not quantify the trade-off between the high-level signalling overhead required for multi-agent intent negotiation [10], [11] and the semantic compression gains actually realized at the wireless edge.

The present work addresses one component of this space. A decentralized multi-agent path planning policy in which agents already exchange learned messages at every timestep is extended so that the decision to transmit becomes part of what the policy learns, with the consequence that each agent communicates only when doing so contributes to the team finding collision-free paths to their goals. The following chapter introduces the path planning problem itself, together with the methods that have been developed to solve it.

3

Multi-Agent Path Planning

This chapter reviews the path planning problem and the families of methods developed to address it. Section 3.1 defines the problem and the conditions that render it difficult; Section 3.1.1 and Section 3.1.2 examine classical and metaheuristic approaches respectively; Section 3.1.3 treats learning-based planning, covering neural approximation, the reinforcement learning formulation, and the deep policy-gradient methods on which this work builds; Section 3.2 extends the discussion from a single robot to teams of agents; Section 3.3 reviews multi-agent reinforcement learning for path finding; Section 3.5 collects the open challenges; and Section 3.4 reviews the state of the art before the gap is stated in Section 3.5.2.

3.1 The Path Planning Problem

Autonomous mobile robots are increasingly deployed across sectors such as transportation, logistics, manufacturing, agriculture, and healthcare, in all of which safe and efficient navigation is decisive for mission success. The core requirement underlying these applications is path planning, understood as the ability to determine a collision-free and efficient route from an initial position to a target destination while respecting environmental constraints, kinematic limitations, and operational objectives. The problem becomes particularly demanding in dynamic and uncertain environments, in which obstacles may move unpredictably, the environmental map may be incomplete, and the mission objectives themselves may evolve during execution [13].

It is customary to distinguish global planning, which computes a route over a known map before motion begins, from local planning, which reacts to obstacles perceived during execution, and equally to distinguish static environments, in which the obstacle configuration is fixed and known in advance, from dynamic ones in which it is not. The methods surveyed in the remainder of this chapter differ principally in which of these regimes they address, and in whether the map is assumed available a priori or must be discovered through interaction. Figure 3.1 summarizes the families considered.

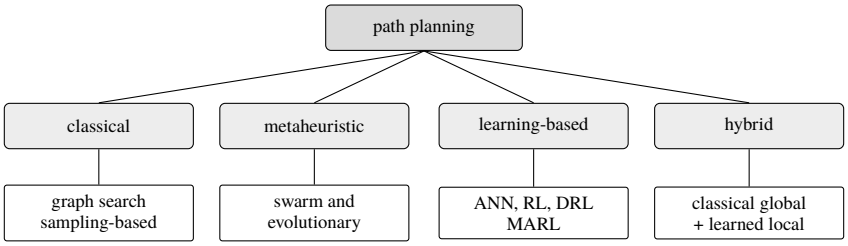

Figure 3.1: Principal families of path planning methods considered in this chapter. Hybrid architectures combine a classical global planner with a learned local policy, and are discussed in Section 3.5.

3.1.1 Classical Approaches

Traditional path planning algorithms such as A* [14] and Dijkstra's algorithm excel in static and well-structured settings but falter in applications characterized by uncertainty, unpredictability, and incomplete knowledge of the environment [15]. Search-and-rescue operations in disaster-stricken areas, for instance, generally require navigation through unknown terrain filled with debris and moving obstacles, while autonomous delivery robots operating in crowded urban environments must take real-time decisions to avoid pedestrians and other obstacles whilst still reaching their destinations efficiently. Extensions such as the focussed D* algorithm [16] confer a degree of adaptability by permitting the existing solution to be repaired rather than recomputed, yet they nonetheless incur substantial computational overhead whenever replanning is triggered, and this overhead grows with the frequency of environmental change.

A second limitation of the classical family concerns dimensionality. In environments exhibiting complex geometry or high-dimensional state spaces, search-based methods struggle on account of the exponential growth of the space that must be explored [17], and the sampling-based planners introduced to mitigate this growth trade completeness for tractability without eliminating the underlying difficulty. A third limitation concerns uncertainty: real robots contend with sensor noise, actuator variability, and unpredictable environmental dynamics, and although classical methods accommodate some of this through deterministic approximation, they lack robustness to genuinely stochastic effects [15].

The same family extends to teams of robots, where the joint configuration space grows exponentially with the number of agents and exhaustive search consequently becomes impractical. Subdimensional expansion addresses this growth by planning each agent individually and expanding the search into the joint space only in the neighbourhood of a detected conflict, and the operator-decomposition variant M*, together with its optimal extension ODrM*, applies this principle so as to retain completeness and optimality guarantees while confining the combinatorial cost to the regions in which agents genuinely interact [18]. The formulation is developed further in Section 3.2.

3.1.2 Metaheuristic Techniques

Metaheuristic and bio-inspired methods, encompassing swarm intelligence and evolutionary computation, were introduced in order to escape the scaling behaviour of exhaustive search by sampling the solution space under the guidance of a fitness function. The families most widely applied to mobile robot path planning are particle swarm optimization, the genetic algorithm, ant colony optimization, the firefly algorithm,

and grey wolf optimization, to which a broad range of further nature-inspired variants may be added [19]. These techniques offer significant advantages, including adaptability to three-dimensional navigation, practical obstacle avoidance, and robustness under dynamic conditions. Despite their potential, however, they face inherent challenges, namely sensitivity to parameter tuning, high computational demands for real-time deployment, susceptibility to premature convergence, and concerns regarding interpretability; addressing these limitations requires hybridization strategies, adaptive parameter control, and robust learning frameworks that balance global exploration against local exploitation while preserving safety and efficiency in real-world operation [20].

3.1.3 Learning-Based Path Planning

Unlike classical optimization, artificial intelligence techniques enable robots to learn from data, to adapt to dynamic and uncertain environments, and to take real-time decisions by exploiting reasoning, perception, and learning capabilities [21]. Such techniques have proven promising in handling complex, high-dimensional search spaces, moving obstacles, and multi-robot coordination, although challenges persist with respect to computational demands, the volume of training data required, generalization to unseen scenarios, and the interpretability of the learned models.

Artificial Neural Networks

Artificial neural networks are collections of interconnected processing elements, or neurons, which approximate relationships in data through weighted connections in a manner that mimics the way the human brain processes information [22]. A feedforward architecture consisting of an input layer, one or more hidden layers, and an output layer is the form most commonly employed for wheeled mobile robot path planning, and owing to their adaptability and their capacity to approximate nonlinear mappings, such networks are particularly well suited to navigation tasks in dynamic and uncertain environments [23].

Reinforcement Learning and the Markov Decision Process

Reinforcement learning (RL) is a subset of machine learning in which an agent learns a policy by interacting with its environment, maximizing cumulative rewards while receiving feedback in the form of rewards for desired actions and penalties for undesired ones [24]. This process reinforces the decisions taken at each step and thereby enables the agent to learn to act in a manner that carries it toward its goal, and RL has gained ground over traditional heuristic methods on account of its adaptability, its capacity to reach good local optima in large state spaces, and its robustness. Within autonomous mobile robot navigation the methods employed include model-free and model-based approaches, together with value-function-based, policy-based, and actor–critic formulations [25].

Formally, the problem is modelled as a Markov decision process characterized by the five-tuple

$$\mathcal{M} = (S, A, P, r, \gamma), \tag{3.1}$$

whose elements are defined as follows [15]:

- S: the state space, representing all possible states of the environment, so that a state $s_t \in S$ encodes all information relevant to the environment at a given time t.

- A: the action space, representing all possible actions available to the agent, so that an action $a_t \in A$ represents a decision or control input applied at time t.

- $P(s'|s, a)$: the state transition probability distribution, defining the probability of passing from a current state $s \in S$ to a successor state $s' \in S$ as a consequence of taking action $a \in A$, and thereby capturing the dynamics of the environment.

- r: the reward function, which assigns a scalar signal to each transition and thereby encodes the objectives of the task.

- $\gamma \in [0, 1)$: the discount factor, which determines the weight assigned to future rewards relative to immediate ones.

Instantiated for a mobile robot, each element acquires a concrete interpretation [15]:

- *State space*: the state $s \in S$ represents the robot's current status together with its environmental context, and may comprise the position (x, y), the orientation θ, the velocities, and potentially sensor measurements such as the distances to nearby obstacles and the relative position of the goal.

- *Action space*: the action $a \in A$ represents a control input applied to the robot, which may take the form of discrete motion commands such as moving forward or turning left, or alternatively of continuous control signals such as the linear velocity v and the angular velocity ω.

- *State transition*: $P(s'|s, a)$ models the robot's motion dynamics and its interactions with the environment, so that under differential-drive kinematics the passage from s to s' following the application of a follows from the equations of motion, with stochasticity arising from wheel slip and sensor error.

- *Reward function*: r encodes the objectives of the task, typically rewarding progress toward the goal and its eventual attainment while penalizing collisions with obstacles and, frequently, the elapsed time or the control effort expended.

The objective is to learn a policy $\pi_\theta(a|s)$ maximizing the expected discounted return

$$J(\pi) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right], \tag{3.2}$$

and the optimal policy is accordingly obtained as

$$\pi^*(a|s) = \arg\max_{\pi} \ \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]. \tag{3.3}$$

Why Reinforcement Learning over Classical Planning

Three properties of this formulation account for its adoption in the settings where the classical methods of Section 3.1.1 prove inadequate. The first is that RL inherently learns a policy $\pi(a|s)$ mapping states to actions, so that decisions are taken in real time without the search being repeated whenever the environment changes [15]. The second is that the policy is represented by a function approximator whose cost at execution does not grow with the size of the state space, which renders tractable at run time the high-dimensional settings described above, albeit without any guarantee that the solution obtained is more than locally optimal. The third is robustness, in that a policy trained under stochastic transitions learns to act well in expectation rather than under an assumed deterministic model, which suits the sensor and actuator noise encountered in deployment.

It should nevertheless be noted that a considerable proportion of the reinforcement learning literature in autonomous mobile robotics operates primarily as a local module for collision avoidance or for exploration in unknown environments, combined with classical or global planners within a hierarchical navigation stack, rather than serving as a standalone global planner [26], [27].

Deep Reinforcement Learning

Deep reinforcement learning extends the preceding framework by employing deep neural networks to approximate policies and value functions, which confers scalability to high-dimensional observations [28]. By combining the representational power of deep learning with the decision-making capability of reinforcement learning, it has emerged as a dominant approach for handling perception and control in autonomous navigation [15], [29], and by integrating high-dimensional sensory inputs with sequential decision-making it has demonstrated strong potential for addressing trajectory planning in uncertain and dynamic environments [30]. Figure 3.2 illustrates the general architecture, in which perception, policy learning, and reward feedback are tightly integrated so as to achieve autonomous decision-making.

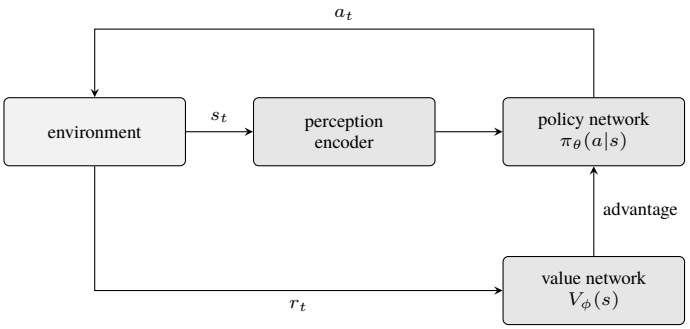

Figure 3.2: General architecture of a deep reinforcement learning navigation system, in which perception, policy learning, and reward feedback are integrated within a single loop. Adapted from [19].

Recent studies have emphasized the extent to which deep reinforcement learning enhances adaptability, trajectory efficiency, and safety relative to classical approaches, although challenges persist with respect to training efficiency, reward design, and convergence stability [31]. Taken as a whole, the field has evolved

from proof-of-concept exploration systems toward hybrid, hierarchical, and multi-agent frameworks capable of confronting uncertainty, scalability, and the demands of real-world deployment.

Value-Based, Policy-Based, and Actor–Critic Methods

Deep reinforcement learning algorithms are conventionally partitioned into three families according to what the network is trained to represent. Value-based methods approximate the action-value function and derive the policy from it by acting greedily, of which the deep Q-network is the canonical instance [32]; policy-based methods parameterize the policy directly and optimize it by gradient ascent on the expected return; and actor–critic methods combine the two, employing a critic to estimate the value function and thereby to reduce the variance of the gradient supplied to the actor [15].

Among the continuous-control members of the latter two families, the deep deterministic policy gradient [33] and soft actor–critic [34] have demonstrated strong results, the latter through the maximization of an entropy-regularized objective that promotes exploration. Proximal policy optimization [35] is the policy optimization method employed in the present work. It constrains the magnitude of each policy update by optimizing a clipped surrogate objective,

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t \left[\min \left(\rho_t(\theta)\, \hat{A}_t,\ \text{clip}(\rho_t(\theta),\, 1-\epsilon,\, 1+\epsilon)\, \hat{A}_t \right) \right], \tag{3.4}$$

in which $\rho_t(\theta) = \pi_\theta(a_t|s_t)/\pi_{\theta_{\text{old}}}(a_t|s_t)$ denotes the probability ratio between the updated and the previous policy, $\hat{A}_t$ denotes the estimated advantage, and ϵ bounds the permissible deviation. The clipping removes the incentive to move the policy beyond the region in which the sampled data remain informative, which yields the stability that has made the method a standard choice for navigation tasks. These techniques, alongside soft actor–critic and the deep deterministic policy gradient, have attained state-of-the-art results across a range of benchmark environments in problems such as autonomous navigation in cluttered indoor spaces, multi-agent coordination, and socially aware path planning [15].

3.2 From Single-Agent to Multi-Agent Path Finding

When several robots share a workspace, the planning problem acquires a coupling that the single-agent formulation does not exhibit, since the trajectory chosen by one agent alters the space available to the others, and a solution is admissible only if it is free of collisions between agents as well as between agents and static obstacles. Approaches to this problem are conventionally distinguished according to whether planning is coupled, in which case the joint configuration space of the entire team is searched and completeness and optimality guarantees are retained at the cost of a search space growing exponentially in the number of agents, or decoupled, in which case each agent plans individually and conflicts are resolved afterwards, which scales far better but forfeits the guarantees.

A parallel distinction concerns where the computation resides. Centralized planners assume a single node with access to the full state of the team, which affords the most complete view but concentrates computation at one point and requires the continuous communication of full state information, whereas decentralized planners distribute the decision among the agents themselves, each acting on the basis of lo-

cal observation, which scales with team size and removes the single point of failure at the cost of each agent possessing only partial information about the others.

Formally, a MAPF instance is defined by a graph $G = (V, E)$, a set of agents $A = \{1, \ldots, n\}$, a tuple of distinct start vertices $\mathcal{S} = (s_i \in V)_{i \in A}$, and a tuple of goal vertices $\mathcal{T} = (g_i \in V)_{i \in A}$. A configuration $Q = (v_1, v_2, \ldots, v_n) \in V^n$ is a tuple of locations for all agents, in which $Q[i] = v_i$ denotes the location of agent $i \in A$, so that a configuration constitutes a global snapshot of the team at a single timestep. Two consecutive configurations Q_t and Q_{t+1} exhibit an edge collision whenever there exists a pair of agents $i \neq j$ such that $Q_t[i] = Q_{t+1}[j] \wedge Q_{t+1}[i] = Q_t[j]$. A solution is a sequence $\Pi = (Q_0, Q_1, \ldots, Q_k)$ satisfying $Q_0 = \mathcal{S}$ and $Q_k = \mathcal{T}$, in which any two consecutive configurations are connected and collision-free [36]. Solution quality is conventionally assessed through the flowtime, defined as the sum over agents of the timesteps elapsed before each halts at its goal, or equivalently through the sum-of-loss, the number of agent actions that do not consist of remaining at a goal.

MAPF planners are broadly classified into coupled, decoupled, and dynamically-coupled approaches [37]. Coupled approaches, of which standard A* applied to the joint space is the archetype, treat the multi-agent system as a single very high-dimensional agent and consequently suffer from exponential growth in planning complexity. Decoupled approaches compute individual paths for each agent and subsequently adjust those paths so as to avoid collisions, and because both the planning and the adjustment occur in low-dimensional search spaces, they find paths rapidly for large teams at the cost of the optimality guarantees that coupled search provides.

Dynamically-coupled approaches occupy the intermediate position and are the most relevant to the present chapter. The configuration space is expanded only locally, around collisions between single-agent plans, and joint planning is performed there through limited backtracking in order to resolve the collision before the single-agent plans resume. OD-recursive-M* (ODrM*) [18] refines this principle by further reducing the set of agents for which joint planning is necessary, decomposing it into independent collision sets, and combining this decomposition with operator decomposition [38] so as to keep the branching factor small during search. ODrM* supplies the expert demonstrations from which the policy described in Chapter 4 learns by imitation.

A more recent line of coupled search has narrowed the scalability gap considerably. Exhaustive search over configurations is impractical for large teams because the number of successor configurations grows exponentially, and LaCAM mitigates this by generating successors sequentially rather than all at once, imposing constraints upon each generation so as to guarantee that all successors eventually appear; this lazy successor generation dramatically reduces the computational burden and yields a quick and scalable approach to MAPF [39]. Its refinement LaCAM* additionally guarantees the eventual discovery of optimal solutions whenever the solution cost takes the form of cumulative transition costs, and operates as an anytime algorithm that continues to improve a solution once one has been found [36]. These guarantees, however, require assumptions that a decentralized setting cannot satisfy, since LaCAM* needs complete knowledge of the map and of the state of every agent, plans centrally for the whole team, and must repeat that computation whenever the environment changes.

3.3 Multi-Agent Reinforcement Learning for Path Finding

The decentralized regime is precisely the one in which reinforcement learning has been applied to multi-robot navigation, since a policy trained to act on local observation transfers naturally to a setting in which no agent has access to global state. Globally guided reinforcement learning has been applied to navigation amidst other robots and dynamic obstacles [40], and extensions of proximal policy optimization have demonstrated strong performance in multi-robot coordination by balancing stability against efficiency, with one line of work combining the method with transfer learning [41] and another hybridizing it with deep Q-learning and convolutional perception [42].

Multi-agent reinforcement learning has meanwhile witnessed significant progress in handling complex coordination problems. A decentralized framework employing dual critics has been proposed that enables robots to optimize joint team rewards while simultaneously respecting individual constraints [43], and a hierarchical reinforcement learning method for large-scale multi-agent path finding has been introduced that enhances sample efficiency through spatiotemporal abstraction [44]. Both illustrate the general strategy by which the coupling between agents is managed without recourse to a centralized planner, namely by structuring either the critic or the temporal resolution of the problem so that the joint objective remains tractable as the number of agents grows.

The benefit conferred by inter-agent communication in such settings has been measured directly in the related domain of connected vehicle routing, where a centralized model, a decentralized model without communication, and a decentralized model with communication were compared on a grid-based course: the fully centralized model attained the best overall performance by virtue of complete action-state sharing, while among the decentralized variants the model equipped with communication outperformed the model without it [45]. Communication therefore recovers a substantial portion of the coordination that decentralization forfeits, which motivates its inclusion in decentralized planners.

The decentralized learned planners developed specifically for MAPF, together with the hybrid and curriculum-based schemes that have followed them, are examined in Section 3.4.

3.4 Multi-Agent Path Planning Related Work

The works reviewed below delimit the ground upon which the method of Chapter 4 is constructed, and are presented in order of their proximity to it.

PRIMAL [37] is the framework from which the decentralized learned approach to MAPF descends. Its authors observe that classical planning approaches are maladapted to real-world deployment, in which noise and uncertainty frequently require that paths be recomputed online, an operation that is impossible when planning times are measured in seconds to minutes. PRIMAL accordingly combines reinforcement and imitation learning in order to teach fully decentralized policies, whereby agents plan paths reactively and online in a partially observable world while exhibiting implicit coordination. The world is a partially observable discrete gridworld in which each agent observes only a limited FOV centred upon itself, a 10×10 window in the reported experiments, which the authors regard as an important step toward real-world robot deployment. Two mechanisms address the central difficulty of training a decentralized policy, namely that of encouraging agents to act selflessly even when doing so is detrimental to the immediate maximization

of their own reward: a blocking penalty, which charges an agent whose choice of position prevents another from reaching its goal, and imitation learning from the ODrM* expert, which supplies demonstrations of coordinated behaviour that reinforcement learning alone discovers only slowly. Once learned, the resulting policy may be copied onto any number of agents and scales naturally to different team sizes and world dimensions. PRIMAL deliberately dispenses with explicit communication among agents, in contrast to approaches that share observations or selected actions during training and sometimes also during execution, and it is precisely this omission that the communication-based successors discussed in Chapter 4 set out to remedy.

LF [39] pairs a centralized, discrete, full-horizon MAPF planner with a decoupled optimal trajectory controller, and thereby occupies the opposite end of the design space from PRIMAL. The coordination layer is LaCAM, which computes collision- and deadlock-free paths rapidly, and the execution layer is Freyja, a model-based optimal non-linear feedback control stack for fast and agile manoeuvres; the former replans at 5–10 Hz while the latter processes trajectories continuously at 50–100 Hz, and the intermediate refinement step converts the geometrically continuous but temporally discrete path produced by the planner into a trajectory suitable for execution. The architecture retains the long-horizon coordination that constitutes the principal attraction of coupled methods, which decoupled alternatives sacrifice, but it does so under the assumption that the system enjoys access to full state and map information, and prior attempts at online MAPF replanning during deployment have been obliged to limit the planning horizon or to subdivide the problem into small teams in order to contain the computational burden.

MAPPOHR [46] addresses the trade-off between the computational cost of continual heuristic replanning and the sample inefficiency of reinforcement learning through a two-layer hybrid architecture combining heuristic search, empirical rules, and multi-agent reinforcement learning. A heuristic planner, D*, computes a globally optimal path that serves as global guidance, while a real-time planner follows that guidance as closely as circumstances permit, detects dynamic obstacles in its vicinity, and issues actions on the basis of local observation so as to avoid collisions with other moving agents; the heuristic planner then replans in response to the instructions of the real-time planner. Training follows the centralized-training, decentralized-execution paradigm, in that observation information is shared among the robots during training in order to assist policy learning, whereas at execution each robot computes its action from its own observations alone.

CACTUS [47] targets the observation that existing neural MAPF solvers depend upon heavily engineered coordination layers which slow training and restrict generalization to unfamiliar map layouts, and proposes instead a lightweight MARL approach using less than five per cent of the parameters of conventional neural architectures. Its central mechanism is curriculum learning, a machine learning paradigm inspired by human learning in which complex tasks are mastered through the stepwise solution of easier subtasks ordered by difficulty [48], where difficulty may depend upon aspects as varied as the complexity of the data samples, the objective function, or the learned model itself. Applied to reinforcement learning, curriculum learning addresses hard exploration problems characterized by sparse rewards or dynamic constraints [49], and its key challenge lies in finding or generating a sequence of tasks that are neither too easy nor too difficult for the learner, so that progress remains steady and robust.

CACTUS adopts reverse curriculum learning, which presumes explicit goal states of the kind the MAPF formulation already provides, and constructs a curriculum as a sequence of tasks in which the expected distance between an agent and its goal increases gradually. Concretely, the goal assigned to each agent is

drawn from the set of vertices lying within a goal allocation radius R_{alloc} of its start location, with R_{alloc} initialized to unity and incremented whenever the average completion rate μ measured across an epoch exceeds a decision threshold U with sufficient confidence, formally whenever $\mu - \eta\sigma \geq U$ for standard deviation σ and confidence coefficient η. The difficulty of the instances presented to the learner is thereby governed by a signal derived from the agents' own performance together with its uncertainty, rather than by a schedule fixed in advance.

3.5 Open Challenges and Research Gap

3.5.1 Open Challenges

The approaches reviewed above leave a number of difficulties unresolved. Although learning-based path planning offers significant advantages over conventional approaches, it carries its own challenges and limitations. The quality of the solution generated depends heavily upon several factors, among them the quality and quantity of the available data, the accuracy of the models employed, the extent of the training required, and the selection of appropriate algorithmic parameters; these approaches can moreover be computationally intensive, demanding substantial processing power and storage capacity, and the transfer of solutions from simulation to real-world environments remains a significant obstacle. Notwithstanding these difficulties, learning-based path planning has achieved notable success in applications such as autonomous driving and warehouse automation [19].

Real-time decision-making constitutes a further fundamental requirement, since delays in processing sensor data and in executing control commands can significantly impair both performance and safety, and deep reinforcement learning methods that prove effective in simulation frequently struggle with real-time execution on account of computational overhead, sample inefficiency, and the dimensionality of the state and action spaces [15]. Hybrid architectures that combine a classical global planner with a learned local policy have been advanced partly in response, and their principal advantage lies in improved learning efficiency and generalization in real-world environments, although they raise in turn difficulties of computational complexity and stability.

A more recent development concerns the role of foundation models. Unlike classical learning-based planners that map observations directly to actions, models such as large language models and vision-language models are increasingly employed as high-level cognitive layers providing semantic reasoning and grounding for embodied agents, generating goals, waypoints, or constraints which classical or learning-based planners subsequently execute [19]. This arrangement is complementary to rather than competitive with the methods reviewed above, in that it addresses the specification of the planning problem rather than its solution.

3.5.2 Research Gap in Communication for Multi-Agent Path Finding

Taken together, these approaches reveal a consistent division in the manner by which coordination is obtained and in the cost at which it is purchased. Coupled planners such as LaCAM and the LF framework built upon it achieve long-horizon coordination of high quality, but presuppose access to full state and map infor-

mation together with a centralized computation that must be repeated whenever the environment changes. Decentralized learned planners relinquish that assumption and act upon local observation alone, yet in doing so they forfeit the coordination that global knowledge conferred, and they recover it by one of two means: either they dispense with inter-agent communication entirely and rely upon implicit coordination induced by reward shaping and expert imitation, as PRIMAL does, or they introduce a learned communication channel through which agents exchange messages at every timestep. The hybrid and curriculum-based schemes address orthogonal difficulties, since MAPPOHR combines global guidance with reactive local control in order to reduce replanning cost, and CACTUS reduces parameter count and improves generalization through a confidence-driven curriculum, but neither treats the volume of communication as a quantity to be optimized. The comparison reported in Section 3.3 establishes that communication materially improves decentralized coordination, yet stops short of asking how much of it is actually required.

The question of *when* an agent should communicate therefore remains open in the decentralized MAPF setting. Existing learned planners treat transmission as unconditional, so that the communication cost grows with the product of team size and episode length irrespective of whether the messages exchanged influence any recipient's decision. The following chapter addresses this question by making the decision to transmit part of what the policy learns, so that each agent communicates only when doing so contributes to the team finding collision-free paths to their goals.

4

Architecture and Methodology

This chapter describes the architecture upon which the present work is built and the modifications introduced to it. Section 4.1 presents the SCRIMP baseline, following the organization of the original paper: the problem setting and observation space, the network structure, the three learning techniques comprising transformer-based communication learning, value-based tie-breaking and the episodic buffer, the training details, and the reported results. Section 4.2 introduces the learned communication gate through which the goal-oriented principle of Chapter 2 is realized, and Section 4.3 describes the strategies employed to stabilize training once the communication penalty is imposed.

4.1 The SCRIMP Baseline

SCRIMP [4] is a decentralized MAPF planner in which agents learn individual policies from a very small field of view and compensate for the resulting partial observability through a highly scalable global communication mechanism based on a modified transformer. It extends the earlier PRIMAL [37] and PRIMAL2 [50] frameworks of the same authors, which combine RL and imitation learning so as to teach fully decentralized policies for MAPF, and it retains from them the paradigm of CTDE together with imitation learning from a centralized expert, both introduced in Section 3.4.

4.1.1 Problem Setting and Observation Space

The problem instance follows the formulation of Section 3.2, with agents navigating a discrete grid-world in which each vertex corresponds to a grid cell and edges connect cells adjacent along the four cardinal directions. A MAPF instance is a tuple $\langle G, S, T\rangle$ in which $G = (V, E)$ is an undirected graph and $S = \{s_1, \ldots, s_n\} \subset V$ and $T = \{t_1, \ldots, t_n\} \subset V$ give n unique start and goal vertices, one per agent. Time is discretized into steps, and at each step every agent executes a single action consisting either of a move to an adjacent vertex or of a wait at its current vertex; movement actions are feasible only insofar as they do not

cause a vertex, edge, or swapping collision. The objective is to find the collision-free joint path that carries every agent from S to T in the fewest time steps.

Each agent observes only a $3 \times 3 \times 8$ local field of view o_i^t centred upon itself, which keeps the dimensions of the network's inputs fixed across world sizes and remains closer to practical applications, in which robots rely upon onboard sensors whose view may be occluded by nearby obstacles or agents. The observation divides into two parts. The first consists of eight binary 3×3 matrices, each covering the same window centred upon the agent: a map marking the cells occupied by agents, including the agent itself at the centre and any other agents currently inside the window; a map marking the cell containing the agent's own goal, which is empty whenever that goal lies beyond the window; a map marking the goals of the other agents inside the window, each goal that lies outside the window being projected onto the nearest cell of its boundary; a map marking the cells occupied by static obstacles, in which all cells falling outside the boundaries of the world are likewise marked as obstacles; and four heuristic maps corresponding to the four movement actions, in which a cell is marked as unity if and only if an agent standing there would come closer to its goal by taking that action [51].

The second part is the seven-length vector $\tilde{o}_i^t$, whose components are the current normalized distances between the agent and its own goal along the x-axis, the y-axis, and in Euclidean terms, denoted dx_i^t, dy_i^t and d_i^t; the extrinsic reward re_i^{t-1} and the intrinsic reward ri_i^{t-1} received at the previous step; the minimum Euclidean distance $dmin_i^{t-1}$ between the agent's previous location and the past locations stored in its episodic memory buffer; and the agent's most recent action a_i^{t-1}.

The reward structure is reproduced in Table 4.1. In alignment with the preceding works, agents currently off-goal are penalized at each step so as to encourage faster task completion. An episode terminates either when all agents stand upon their goals at the end of a time step or when the pre-defined limit of 256 time steps is reached. Agents communicate with a delay of one time step between the sending of a message and its reception or use, and communication is not hindered by obstacles.

Table 4.1: Reward structure of SCRIMP. Reproduced from [4].

Action	Reward
Move (up / down / left / right)	-0.3
Stay (on goal, away from goal)	0.0, -0.3
Collision	-2
Block	-1

4.1.2 Network Structure

The network is composed of three modules, shown in Figure 4.1: the observation encoder, the transformer-based communication block, and the output heads.

Observation encoder. Seven convolutional layers, two max pooling layers, three fully connected layers and one long short-term memory (LSTM) unit make up the observation encoder, whose structure is nearly identical to that of PRIMAL [37]. For agent i at time step t, the local observation o_i^t is encoded into $\dot{o}_i^t$ through two stages of three convolutions and one max pooling operation, followed by a final convolutional layer. In parallel the vector $\tilde{o}_i^t$ is independently encoded by a fully connected layer and then concatenated

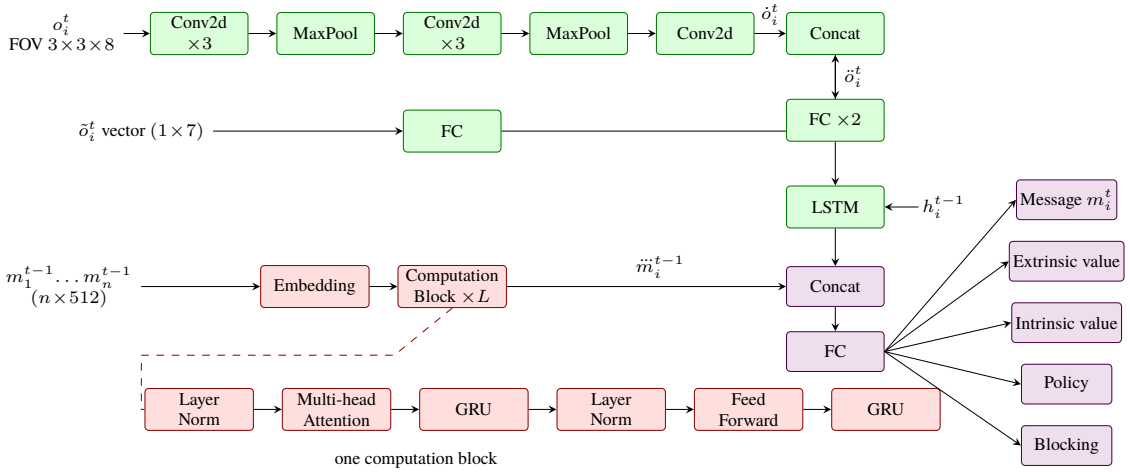

Figure 4.1: Network structure of SCRIMP. Green modules belong to the observation encoder, which is nearly identical to that of PRIMAL; red modules to the transformer-based communication block, consisting of a stack of L computation blocks whose detail is shown below; and purple modules to the output heads. Adapted from [4].

with $\dot{o}_i^t$ so as to form $\ddot{o}_i^t$. The encoded observation $\ddot{o}_i^t$ is finally passed through two fully connected layers and fed into the LSTM unit together with the hidden state h_i^{t-1} output by the unit at the previous time step, which endows each agent with the ability to integrate its own past information across time.

Transformer-based communication block. Only the encoder of the transformer is used, and it applies across agents rather than across time, in that it treats the messages output by all agents at the previous time step $\{m_1^{t-1}, \dots, m_n^{t-1}\}$ as its input. These messages are first augmented with information about the unique identity of each agent by means of a sinusoidal positional embedding, and are then passed through L iterative computation blocks so as to produce the final output of the communication block $\ddot{\dot{m}}_i^{t-1}$. Each computation block comprises a multi-head self-attention mechanism, a multi-layer perceptron, residual connections, layer normalizations and gated recurrent units (GRUs), arranged as shown at the bottom of Figure 4.1.

Output heads. The output of the communication block is concatenated with the input and the output of the LSTM cell, and the resulting representation feeds five heads producing the message m_i^t to be used at the following time step, the predicted extrinsic and intrinsic state values, the policy for the current time step, and the blocking prediction indicating whether the agent obstructs a teammate from reaching its goal.

4.1.3 Learning Techniques

SCRIMP introduces three learning techniques, described in the three subsections that follow, together with the training details inherited from the preceding works.

Transformer-Based Communication Learning

The communication mechanism allows agents to gain knowledge about the past observations of other agents, thereby mitigating the risks posed by partial observations and facilitating team-level cooperation that cannot be learned under a communication-free decentralized framework [4].

The choice of a transformer over a recurrent alternative is argued from two directions. Since transformers have delivered significant performance gains compared with the LSTM in other domains, the authors hypothesize that transformer-based communication mechanisms should likewise outperform the LSTM-based communication mechanisms previously employed in MARL [52]. In contrast to fields such as natural language processing and computer vision, however, only a small quantity of data is available for training models in online RL, and RL models must moreover adapt rapidly to changing tasks, which makes the classical transformer difficult to optimize and often leaves its performance merely comparable to that of stochastic strategies [53]. Three modifications stabilize training in consequence: the dropout layers were removed, the locations of the layer normalizations within the submodules were rearranged, and a GRU was added in place of the residual connection of the vanilla transformer, as recommended in the work on stabilizing transformers for RL [54].

Within each computation block, the multi-head self-attention is the central component, encoding both the high-level information carried by the messages and the dynamic interrelationships among them. For each agent i, at each of the h independent attention heads, its own message is linearly projected to a query by learnable weights w_q, while all messages are linearly projected to keys and values by learnable weights W_k and W_v. The relation between the self-message of agent i and the messages of all agents is computed as

$$\alpha_i = \text{softmax}\left(\frac{w_q \dot{m}_i^{t-1} \cdot \left(W_k \left[\dot{m}_1^{t-1}, \ldots, \dot{m}_n^{t-1}\right]\right)^\top}{\sqrt{d_K}}\right), \tag{4.1}$$

in which $\dot{m}_i^{t-1}$ is the representation of m_i^{t-1} processed by the preceding layers and $\sqrt{d_K}$, the dimension of the keys, serves as a scaling factor alleviating the vanishing gradient problem of the softmax function. The values are weighted-summed over α_i so as to produce the output of the attention head,

$$\text{head}_i = \alpha_i W_v \left[\dot{m}_1^{t-1}, \ldots, \dot{m}_n^{t-1}\right], \tag{4.2}$$

and the outputs of all heads are concatenated and projected by the learnable weight W_o so as to obtain the final joint output drawn from the different representation subspaces,

$$\ddot{m}_i^{t-1} = \text{Concat}\left(\text{head}_{i1}, \ldots, \text{head}_{ih}\right) W_o. \tag{4.3}$$

Where global communication is unavailable, SCRIMP admits a local variant limiting the communication range on the basis of Euclidean distance, achieved by masking the inputs of the softmax function corresponding to un-allowed communication.

State-Value-Based Tie-Breaking

Because agents act independently upon local information, two of them may select actions placing them in conflict. SCRIMP lets agents check whether they will collide at the next time step before the action is actually executed, and then samples in weighted-random fashion which agent is finally allowed to perform its preferred action, according to a learning-based priority probability; the remaining agents re-sample a new action from their policy with their former, conflicting choice masked.

The predicted difference in team state value serves as the first component of the priority, since it represents the long-term collective benefit of allowing that agent to choose its specific action and thereby prevents selfish, short-sighted behaviour. If agent i would collide with agent j at the next step, the difference between the predicted current team state value and the next team state value, obtained when the action of agent i is executed and agent j re-samples, is defined as

$$\text{diff}_i^t = \sum_{k=1}^{n} \left(\bar{v}_k^t - \bar{v}_{ki}^{t+1} \right), \tag{4.4}$$

in which $\bar{v}_k^t$ is the predicted state value of agent k at step t and $\bar{v}_{ki}^{t+1}$ that of agent k at step $t+1$. The team state value is assumed to be the sum of the individual state value functions, since this is straightforward to implement and guarantees that a global arg max over the team state value produces the same result as a collection of individual arg max operations upon each individual state value [55].

An agent closer to its goal derives a greater self-benefit from performing its preferred action, since fewer time steps, and hence smaller exponents of the discount factor, are required in order to obtain the higher rewards associated with reaching and remaining upon the goal. In the target symmetry scenario [56] depicted in Figure 4.2, however, in which one agent must move past the goal of another in order to reach its own, the agent closer to its goal ought to be assigned a lower priority probability so as to prevent deadlocks, which may contradict the result computed by (4.4). A second component is therefore introduced, relying upon the normalized Euclidean distance between the current position of each agent and its goal, and the weighted sums of the state-value differences and the normalized distances are passed through a softmax layer so as to generate the final priority probability,

$$P = \text{softmax}\left(\text{diff}_i^t + \mu \frac{d_i^t}{\sum_k d_k^t},\ \text{diff}_j^t + \mu \frac{d_j^t}{\sum_k d_k^t},\ \cdots \right), \quad k \in \{i, j, \dots\}, \tag{4.5}$$

in which μ is a pre-defined hyperparameter controlling the importance of distance, d_i^t is the distance between the current position of agent i and its own goal, and $\{i, j, \dots\}$ is the set of collided agents. The agent allowed to move is finally sampled from this probability, and it was observed experimentally that always greedily selecting the agent of highest priority probability performs worse than stochastic sampling.

Episodic Buffer for Improved Exploration

Intrinsic rewards encourage agents to explore more new area within each episode, thereby increasing the probability of reaching the goal. Each agent keeps an independent buffer of limited capacity M, which begins each episode empty. At each step every off-goal agent separately computes the Euclidean distance

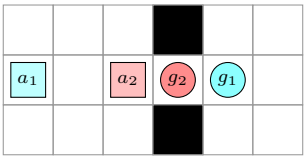

Figure 4.2: Example of target symmetry. Agent 1 (a_1) needs to traverse the goal (g_2) of a_2 in order to reach its own goal (g_1), so that the agent nearer to its goal should be granted the lower priority. Adapted from [4].

between its current (x, y) location and all locations stored in the buffer, and if the maximum distance exceeds a threshold τ the agent generates an intrinsic reward,

$$ri_i^t = \phi(\beta - \delta), \qquad \delta = \begin{cases} 1, & \text{if max distance} < \tau, \\ 0, & \text{if max distance} \geq \tau, \end{cases} \tag{4.6}$$

in which $\tau, \phi \in \mathbb{R}^+$ and $\beta \in \{0, 1\}$ are pre-defined hyperparameters. Essentially τ checks whether the agent has stepped out of the region with which it had so far become familiar during the current episode; the value of ϕ depends upon the scale of the task reward, while β determines the sign of the intrinsic reward. The values $\phi = 0.2$ and $\beta = 1$ ensure that the best total reward remains obtainable only when agents reach their goals as rapidly as possible and then remain upon them, and the final reward is $r_i^t = ri_i^t + re_i^t$. After any intrinsic reward has been computed, each agent determines whether its current maximum distance is greater than or equal to a further threshold ρ, in which case the current location is added to the episodic buffer [57]; should the buffer stand at full capacity M, a randomly chosen element is replaced by the current one.

Training Details

Beyond the three learning techniques above, SCRIMP retains several techniques introduced in the preceding works [37], [50]. At the beginning of each episode it is chosen at random, according to a predetermined ratio, whether the network will be trained by imitation learning, in the form of behaviour cloning based upon the centralized planner ODrM* [18], or by RL. A supervised binary cross-entropy loss teaches agents to avoid undesirable actions, such as those causing collision with a static obstacle including the boundaries of the world, or a direct return to the previous location, since this yields better solutions than training the same behaviour from negative rewards alone. A blocking penalty is further included in the general reward structure so as to encourage agents to move temporarily away from their goals in specific selfish blocking situations, defined as the presence of an agent extending the A*-path length of another agent by more than ten steps. The complete set of hyperparameters is reproduced in Table 4.2.

Table 4.2: Hyperparameters used to train SCRIMP. The first group applies to proximal policy optimization (PPO) and the MAPF task; the second is specific to the methods proposed in the paper. Reproduced from [4].

Hyperparameter	**Value**
Number of agents	8
FOV size	3
Maximum episode length	256
World size	$(10, 40)$
Obstacle density	$(0\%, 50\%)$
Learning rate	$1e{-5}$
Discount factor	0.95
GAE λ	0.95
Clip parameter for probability ratio ϵ	0.2
Gradient clip norm	10
Number of epochs	10
Number of processes	16
Mini-batch size	1024
Imitation learning rate	10%
Optimizer	Adam
Entropy coefficient	0.01
Policy coefficient	10
Valid action coefficient	0.5
Block coefficient	0.5
Dimension of the feed-forward network of the encoder	1024
Number of computation blocks L	1
Number of heads h	8
Dimension of q, k, v	32
Distance discount factor μ	0.1
Intrinsic value coefficient	0.08
Extrinsic value coefficient	0.08
Capacity of episodic buffer M	80
Threshold for obtaining intrinsic reward τ	random$(1, 3)$
Threshold for adding to buffer ρ	3
Number of steps before adding intrinsic rewards	1e6
ϕ	0.2
β	1

4.1.4 Reported Performance

SCRIMP is evaluated against the centralized planner ODrM* and against two learning-based communicating planners, DHC [51] and PICO [58], across team sizes, world sizes and obstacle densities, using four metrics. The episode length (EL) is the number of time steps required to complete the task, with the episode lengths of failed tasks excluded. The maximum number of goals reached (MR) is the largest number of on-goal agents at any point during an episode. The collision ratio with static obstacles including boundaries (CO) is the number of collisions divided by the product of episode length and number of agents, expressed as a percentage. The success rate (SR) is the percentage of episodes fully solved, meaning those in which all agents reached their goals within the planning time or episode length constraints. Table 4.3 reproduces a representative portion of the reported comparison.

SCRIMP is able to yield performance similar to that of the classical centralized planner ODrM*, even though ODrM* enjoys access to the entire state of the system, relies upon search-based planning that is at worst exhaustive, and can guarantee bounded-optimal solutions given sufficiently large planning times. In

the most difficult task, requiring a high degree of cooperation among 128 agents at 30% obstacle density, SCRIMP attains the higher success rate, whereas ODrM* solves only a fifth of the instances within five minutes of planning and DHC and PICO fail entirely. The collision ratio of SCRIMP, although very low in absolute terms, exceeds that of DHC and PICO in most cases, which the authors attribute to the intrinsic rewards encouraging less conservative policies that approach goals more directly and therefore encounter obstacles more often, whereas the policies of the other two approaches tend to keep agents within their starting region and consequently register markedly worse episode length, maximum goals reached and success rate.

Table 4.3: Average performance along standard MAPF metrics for instances of increasing team size, environment size and obstacle density, averaged over 100 episodes with randomly generated maps. Standard deviations are omitted for compactness. Episode lengths for failed tasks are excluded, and "–" denotes unavailable data. Reproduced in part from [4].

Method	Density	EL ↓	MR ↑	CO ↓	SR ↑
8 agents in a 10×10 *world*					
ODrM*	0%	12.64	–	0.00%	100%
	15%	13.72	–	0.00%	100%
	30%	16.16	–	0.00%	100%
DHC	0%	14.34	8.00	0.00%	100%
	15%	17.23	7.98	0.00%	98%
	30%	29.82	7.88	0.00%	92%
PICO	0%	17.35	8.00	0.00%	100%
	15%	29.23	7.51	0.00%	63%
	30%	35.04	6.68	0.00%	29%
SCRIMP	0%	12.05	8.00	0.00%	100%
	15%	14.17	8.00	0.04%	100%
	30%	18.56	7.93	0.07%	97%
32 agents in a 30×30 *world*					
ODrM*	0%	42.97	–	0.00%	100%
	15%	43.39	–	0.00%	100%
	30%	53.95	–	0.00%	97%
DHC	0%	48.64	32.00	0.00%	100%
	15%	52.48	31.97	0.00%	98%
	30%	91.29	31.61	0.00%	78%
PICO	0%	65.14	30.26	0.00%	14%
	15%	–	22.70	0.00%	0%
	30%	–	12.20	0.00%	0%
SCRIMP	0%	42.59	32.00	0.00%	100%
	15%	43.90	32.00	0.00%	100%
	30%	62.31	31.94	0.32%	96%
128 agents in a 40×40 *world*					
ODrM*	0%	64.55	–	0.00%	100%
	15%	67.77	–	0.00%	100%
	30%	76.90	–	0.00%	20%
DHC	0%	93.00	127.92	0.00%	93%
	15%	137.76	127.45	0.00%	70%
	30%	–	97.83	0.00%	0%
PICO	0%	135.00	126.59	0.00%	2%
	15%	–	95.01	0.02%	0%
	30%	–	58.11	7.80%	0%
SCRIMP	0%	66.86	128.00	0.01%	100%
	15%	73.15	128.00	0.14%	100%
	30%	135.43	121.95	3.72%	58%

4.1.5 Communication Mechanism Ablation

One ablation reported in the paper bears directly upon the present work, since it establishes what the communication mechanism contributes and therefore what stands to be lost when transmission is suppressed. Two variants were constructed, identical to the original except in the communication pathway: SCRIMP-without-com, from which the entire communication mechanism is removed so that agents never receive or use messages, and SCRIMP-without-encoder, from which only the transformer encoder is removed, the

fused message being replaced by a self-message so that agents cannot communicate with one another but can still receive their own message from the previous step.

In simple tasks involving fewer than 128 agents the three configurations perform similarly, with success rates close to 100% in most tasks, and the same holds at low obstacle density even for the largest teams, where the degradation suffered by the variant deprived of communication is very slight and the two configurations remain nearly indistinguishable. The advantage of the full mechanism emerges only in challenging tasks: SCRIMP-without-com exhibits the lowest success rate, which indicates that the content being encoded is essential to good final performance, and SCRIMP-without-encoder also performs worse than SCRIMP, which validates the effectiveness of obtaining information about other agents through dynamic fusion in harder tasks that depend heavily upon cooperation [4].

Two observations follow from this ablation, and together they motivate the remainder of the chapter. The first is that the value of communication is not uniform across instances but concentrated in the difficult ones, since agents deprived of messages perform almost as well as the full model whenever the team is small or the environment sparsely occupied, and fall behind only where obstacle density and team size combine to make coordination indispensable. The second is that the comparison is confined to the two extremes of communicating always and communicating never, leaving unexamined the intermediate regime in which agents transmit selectively. Taken together, these observations suggest that a substantial portion of the messages exchanged under the unconditional scheme is redundant, and that the redundancy is greatest precisely in the instances that the ablation shows to be easiest.

4.2 Learned Communication Gating

The architecture described above communicates unconditionally, in that every agent emits a message at every time step and the transformer fuses all n messages irrespective of whether any given message alters the decision of any recipient. This is the model that is identified in [5] as the common weakness of learned communication protocols: prior works adopt an unconstrained communication model in which an agent communicates with all other agents at every step, even when the task does not require it, which in real-world applications constrained by bandwidth, power and network capacity may prove too costly. Communicating agents in such settings face a dual objective, namely to minimize communication while maximizing a task objective, and these opposing objectives create a challenging learning problem whose typical failure mode consists in agents shutting down communication at the start of learning, before they have learnt to construct meaningful messages.

The response proposed in that work is Efficient Communication Net (ECNet), which directly optimizes the weighted sum of the task reward and a communication cost by learning communication gates, using either REINFORCE [59] in a manner similar to IC3Net [60] or the Gumbel-Softmax reparameterization trick [61]. Two techniques stabilize training: alternating between episodes that do and do not apply the communication penalty, which prevents the models from turning off their outgoing messages, and repeating previously received messages, which helps the models retain information and further improves performance. The reported outcome is a reduction of communication by 75% without any loss of performance. The extension introduced below transfers this formulation to the SCRIMP architecture of Section 4.1.

4.2.1 The Gating Module

The gate is realized as a sixth output head, placed alongside the five heads of the baseline and shown in Figure 4.3. It reads the same fused representation c_i^{fused} that feeds the policy, the two value heads, the blocking head and the message head, obtained by concatenating the output of the communication block with the LSTM state and the encoded observation and passing the result through a fully connected layer. A single linear layer maps this representation to two logits,

$$z_i^t = W_g\, c_i^{\text{fused}} + b_g, \qquad z_i^t \in \mathbb{R}^2, \tag{4.7}$$

which parameterize a categorical distribution $\pi_\theta^g(\cdot \mid c_i^{\text{fused}})$ over the binary decision $c_i^t \in \{0, 1\}$ determining whether agent i transmits at time step t.

The position of the gate within the network is consequential rather than incidental. Because the head reads a representation produced after the communication block has fused the messages of the whole team, the decision to transmit is conditioned upon what the agent has just received as well as upon its own observation, and the agent is therefore able to listen before deciding whether to speak. An agent that finds the circulating messages already sufficient to determine its next action may suppress its own transmission without loss, whereas an agent whose local situation is not accounted for by what it has received retains the incentive to broadcast. This is precisely the conditional relevance that Chapter 2 identifies as the object of goal-oriented communication, namely that the value of a message depends upon what the recipients already know, and it is available here as a direct consequence of the placement rather than requiring any explicit model of the teammates' knowledge. Had the gate instead been placed before the communication block, reading only the encoded observation and the recurrent state, the decision would necessarily have been taken in ignorance of the team's current information and could have been conditioned only upon the agent's own circumstances. The formulation is global gating, in which each agent decides whether or not to communicate at all, as distinct from pairwise gating, in which each agent decides separately with respect to every other agent [5]. Global gating is the appropriate choice here because SCRIMP's communication is itself global, in that a message once emitted is fused by the attention mechanism of (4.1)–(4.3) into a representation distributed to the entire team rather than directed at a particular recipient.

4.2.2 Sampling the Gate

The transmission decision is drawn from the categorical distribution of (4.7), and the manner of drawing it differs between training and deployment. During training the decision is sampled,

$$c_i^t \sim \text{Categorical}\big(\text{softmax}\left(z_i^t\right)\big), \tag{4.8}$$

so that the agent explores the consequences of remaining silent in situations where it would otherwise transmit, and the log-probability of the sampled decision is retained for the policy update described below. At evaluation the stochasticity is undesirable, since a random suppression at a critical bottleneck may produce a collision that a deterministic policy would have avoided, and the decision is therefore discretized by taking

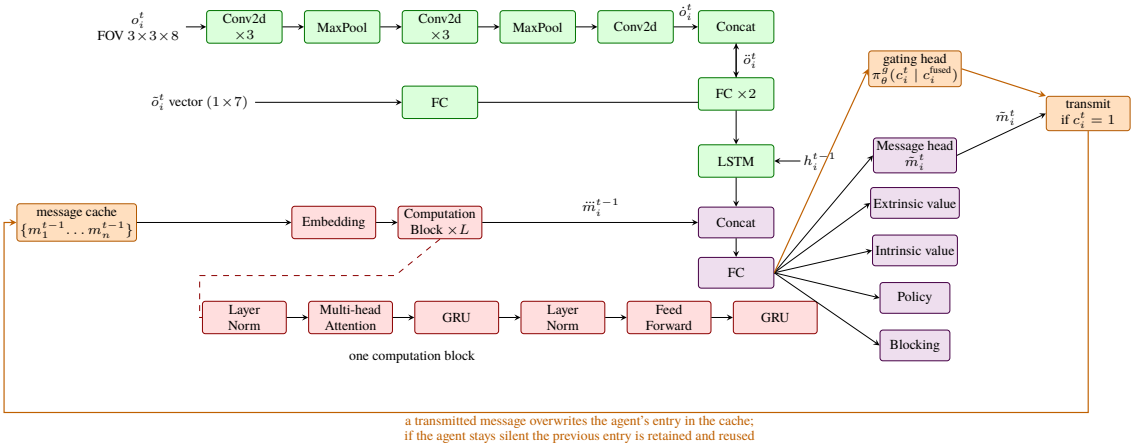

Figure 4.3: The SCRIMP architecture of Figure 4.1 extended with the learned communication gate, shown in orange. The gating head is a sixth output head reading the same fused representation as the other five, and emits the binary decision c_i^t. When $c_i^t = 1$ the freshly generated message $\tilde{m}_i^t$ is transmitted and overwrites the agent's entry in the message cache held at the communication block; when $c_i^t = 0$ nothing is transmitted and the cache retains the agent's previous entry, which the transformer reuses. The cache therefore always presents one entry per agent, so the attention computation is unaffected by how many agents chose to remain silent.

the more probable of the two outcomes,

$$c_i^t = \arg\max_{c \in \{0,1\}} z_{i,c}^t, \tag{4.9}$$

which corresponds to transmitting whenever the estimated probability of doing so exceeds one half.

Sampling from a categorical distribution is not a differentiable operation, so no gradient reaches the gating head through (4.8). The gate is consequently trained by a policy gradient rather than by backpropagation through the sample, which corresponds to the first of the two variants offered in the ECNet formulation, where gates are learned with REINFORCE [59] in a manner similar to IC3Net [60], in preference to the alternative based upon the Gumbel-Softmax reparameterization [61], a technique that renders a discrete sample differentiable by drawing it from a continuous relaxation of the categorical distribution, perturbing the logits with Gumbel noise and passing them through a temperature-controlled softmax, so that gradients may flow through the sample itself. The particular estimator employed here is the clipped surrogate of PPO described in the following section, which treats the gate as a policy in its own right.

The choice between the two estimators is dictated less by their general merits than by the structure of the architecture into which the gate is inserted. A reparameterization estimator is worth its additional complexity only where a differentiable path exists between the sampled decision and the quantity being optimized, and in SCRIMP no such path connects the gate to the task return. Communication carries a delay of one time step, so that a message emitted at step t is consumed by its recipients at step $t+1$; the consequence of transmitting is therefore mediated by the actions those recipients subsequently take and by the response of the environment to them, and during the update the incoming messages enter the network as stored inputs rather than as tensors carrying gradient. Backpropagating through the sample would consequently recover

none of the effect that the decision actually has upon the task, which is precisely the effect the gate must learn. In the setting for which ECNet was devised the messages are consumed within the same step in which they are produced, and the reparameterization estimator is correspondingly better motivated there than here.

A second consideration is one of uniformity. The agent takes two discrete decisions at every step, namely where to move and whether to transmit, and expressing the second as a categorical policy allows both to be optimized by the same clipped surrogate, sharing one advantage estimate, one set of stored behaviour probabilities and one clipping range, which removes the need to reconcile two estimators of differing variance within a single update. It should nevertheless be noted that the penalty of (4.11) is a differentiable function of the gate logits and therefore contributes an exact analytic gradient, so that the two components of the objective reach the gating head by different routes: the communication cost pathwise, and the task consequence through the policy gradient.

4.2.3 Objective

Two design decisions govern how the communication penalty enters training. The first is that the gate is optimized by its own clipped surrogate objective, so that the gating decision is treated as a second policy trained by PPO alongside the navigation policy. Denoting by π^g_θ the gating policy and by $\hat{A}_t$ the advantage estimate already computed for the navigation policy, the gate probability ratio and its surrogate objective are

$$\varrho_t(\theta) = \frac{\pi^g_\theta\big(c_i^t \mid c_i^{\text{fused}}\big)}{\pi^g_{\theta_{\text{old}}}\big(c_i^t \mid c_i^{\text{fused}}\big)}, \qquad L^g(\theta) = \mathbb{E}_t\Big[\min\Big(\varrho_t(\theta)\hat{A}_t,\ \text{clip}(\varrho_t(\theta), 1-\epsilon, 1+\epsilon)\,\hat{A}_t\Big)\Big], \tag{4.10}$$

in which ϵ is the same clipping range used by the navigation policy. Sharing the advantage function couples the two policies, in that a gating decision is judged by its consequences for the task rather than by any separate communication-specific return.

The second decision is that the communication penalty is applied to the loss rather than to the environment reward. Adding it to the reward would entangle the cost of transmitting with the value of occupying a grid cell, so that the critic would learn to evaluate perfectly safe spatial states as unfavourable merely because an agent chose to broadcast there. Confining the penalty to the actor's objective keeps the two value heads estimating purely spatial returns. The penalty itself is taken over the probability of opening the gate rather than over the sampled binary decision, which keeps it differentiable,

$$L^{\text{comm}} = \lambda_{\text{comm}}\,\mathbb{E}\big[\pi^g_\theta\big(c_i^t = 1 \mid c_i^{\text{fused}}\big)\big], \tag{4.11}$$

the expectation being taken over the training batch. The complete objective minimized is therefore

$$\begin{aligned}\mathcal{L} = \; & -L^{\text{CLIP}} - L^g - c_H\mathcal{H} + c_{\text{in}}L^V_{\text{in}} + c_{\text{ex}}L^V_{\text{ex}} \\ & + c_{\text{valid}}L^{\text{valid}} + c_{\text{block}}L^{\text{block}} + L^{\text{comm}},\end{aligned} \tag{4.12}$$

in which L^{CLIP} is the navigation surrogate of (3.4), $\mathcal{H}$ the policy entropy, L^V_{in} and L^V_{ex} the clipped intrinsic and extrinsic critic losses, and L^{valid} and L^{block} the supervised losses on valid actions and blocking inherited from the baseline.

The coefficient λ_{comm} sets the price of a single transmission relative to task performance, and may be held fixed throughout training or introduced gradually according to a schedule; both arrangements are used by the two training strategies of Section 4.3.

4.2.4 Message Repetition

A gate trained under the penalty of (4.11) faces a difficulty peculiar to the architecture into which it is inserted. The communication block expects a message from every agent, since the attention of (4.1) operates over the complete set $\{\dot{m}_1^{t-1}, \ldots, \dot{m}_n^{t-1}\}$, so that an absent entry would leave the fused representation incomplete for the entire team rather than for the silent agent alone. The remedy adapts the message-repetition technique of the ECNet formulation, in which the retention of previously received messages was shown to help models retain information and to improve performance [5]. In the present architecture the mechanism is applied at the point of transmission rather than of reception: when an agent declines to transmit, its most recently transmitted message is supplied to the communication transformer in place of a new one, so that the encoded representation distributed to the team remains complete while no fresh information is actually sent, as shown in Figure 4.3. The cache is held at the communication block rather than at the transmitting agent, so that a silent agent sends nothing at all and it is the transformer that reuses the entry already in its possession; were the agent instead to retransmit its previous message, the decision would save no bandwidth whatever. Denoting by $\tilde{m}_i^t$ the message freshly generated by the message head and by m_i^{t-1} the entry currently held in the cache for agent i, the entry presented to the communication block is

$$m_i^t = c_i^t \, \tilde{m}_i^t + \left(1 - c_i^t\right) m_i^{t-1}, \tag{4.13}$$

which expresses that the cached entry is overwritten when the agent transmits and retained otherwise, so that the gate governs the refreshing of an entry rather than its presence.

Beyond preserving temporal continuity, the arrangement is an architectural necessity. Were silent agents simply to withhold their entries, the length of the sequence presented to the transformer would vary from step to step and with it the shape of the attention matrix, which would require dynamic padding and masking, add computational overhead, and destabilize the attention computation. Eq. (4.13) instead guarantees a token matrix of fixed dimension $n \times d$ at every step, which keeps the scaled dot-product attention of (4.1) unchanged and preserves memory efficiency at large team sizes. The gate therefore governs the refreshing of an agent's contribution rather than its presence.

4.3 Training Strategies

Imposing a communication penalty upon a policy that has not yet learned to construct informative messages introduces a characteristic failure mode, in that the agents discover that silence is cheap before they discover that speech is useful, and consequently switch off their outgoing messages permanently. Two strategies for addressing this difficulty are evaluated in Chapter 5: the technique proposed alongside the gate itself, and an alternative of our own devising.

4.3.1 Alternating Penalized and Unpenalized Episodes

The first strategy is the second stabilization technique proposed alongside the gate itself, in which training alternates between episodes that apply the communication penalty and episodes that do not, thereby preventing the models from turning off their outgoing messages [5]. The unpenalized episodes preserve the incentive to produce messages that carry information, while the penalized episodes exert the pressure toward parsimony, and alternation between the two prevents either objective from dominating before the other has been satisfied.

4.3.2 Curriculum Learning

The second strategy is our own, proposed as an alternative to the alternation technique of ECNet and evaluated against it. Rather than removing the penalty on alternate episodes, it retains the penalty at every step but introduces it gradually, so that the pressure toward parsimony grows only once the navigation policy has converged and the messages have acquired content worth transmitting. In the terms of Section 3.4, the difficulty presented to the learner is ordered rather than modulated, the ordering being applied here to the economic constraint rather than to the spatial task.

Concretely, the penalty coefficient is held at zero until a step k_0 at which the navigation policy is already competent, and is then raised linearly to its target value over a warm-up window of width k_w,

$$\lambda_{\text{comm}}(k) = \lambda^{\star} \cdot \text{clip}\left(\frac{k - k_0}{k_w},\, 0,\, 1\right), \tag{4.14}$$

in which k denotes the current training step, $\lambda^{\star}$ the target coefficient, k_0 the step at which the penalty is first applied, and k_w the width of the window; the values used are stated in Chapter 5. Under the alternating strategy of Section 4.3.1, by contrast, λ_{comm} is held constant at its target value and the alternation between penalized and unpenalized episodes supplies the protection against premature silence. The staged training procedure within which both schedules operate is described in Chapter 5.

4.4 Summary

The method evaluated in the following chapter therefore consists of the SCRIMP architecture, extended with a global communication gate trained jointly with the path planning policy under a weighted communication penalty, made differentiable through the straight-through Gumbel-Softmax reparameterization, and stabilized by supplying an agent's most recently transmitted message to the communication transformer whenever that agent falls silent. Two further training strategies, namely the alternation of penalized and unpenalized episodes and a curriculum over goal allocation radius, are evaluated against this configuration and against the unmodified baseline.

5 Experiments and Results

This chapter reports the experimental evaluation of the method described in Chapter 4. Section 5.1 presents the training regime, first the settings common to both experiments and then the two training strategies in turn; Section 5.2 describes the evaluation protocol and the metrics through which performance is measured; Section 5.3 reports the results; Section 5.5 examines what the communication reduction costs and where the gate elects to transmit.

5.1 Experimental Setup

5.1.1 Common Training Regime

Both experiments were conducted in the discrete grid-world of Section 4.1.1, with worlds generated at each episode by sampling the side length from the interval $[10, 40]$ and the obstacle density from the interval $[0\%, 50\%]$, so that the policy is exposed to environments of widely varying size and congestion during training. Agents observe a 3×3 field of view and act from a five-element action space, and an episode terminates either when every agent stands upon its goal or when the limit of 256 time steps is reached.

Training follows the centralized-training, decentralized-execution paradigm inherited from the baseline, with 16 parallel worker processes each collecting 128 time steps per data-collection cycle, so that one policy update consumes 2048 environment steps. Each update performs ten epochs over minibatches of 1024 samples, using Adam at a learning rate of 10^{-5} with the clipping range fixed at 0.2 and the gradient norm clipped at 10. The discount factor and the generalized advantage estimation parameter are both 0.95. All runs use a single random seed, 1234, and were executed on one NVIDIA L40S graphics processing unit.

The choice of 128 time steps per collection cycle, rather than the larger value used in some configurations of the baseline, has a consequence worth stating explicitly. A shorter cycle yields a smaller batch, so that the advantage normalization performed at each update is computed over fewer samples and the resulting gradient estimates carry greater variance; at the same time, the number of policy updates performed for a

given quantity of environment experience rises in proportion. The setting therefore trades gradient precision for update frequency, and was retained in both experiments so that the two remain comparable in this respect.

5.1.2 Experiment 1: Curriculum Over Task Difficulty

The first experiment applies the curriculum strategy of Section 4.3.2, dividing training into three stages of increasing difficulty, summarized in Table 5.1. The stages are not merely a schedule over instance difficulty but a sequence in which each successive constraint is introduced only once the preceding one has been satisfied.

Stage 1, spanning the first five million steps with eight agents, reproduces the original SCRIMP pipeline exactly: the communication head is inactive, imitation learning from the ODrM* expert is applied with probability 0.1, and the collision penalty retains its published value of -2. The purpose of this stage is solely to learn navigation and obstacle avoidance, and no aspect of the communication mechanism is exercised.

Stage 2, from five to nine million steps, raises the team size to 32 agents while leaving communication unconstrained. Increasing the density in this manner obliges the agents to learn high-fidelity coordination representations in crowded spaces before any price is attached to bandwidth, so that by the time the penalty is introduced the messages exchanged already carry information worth transmitting. Two further changes accompany the transition: imitation learning is disabled, since the expert planner supervises only the choice of action and not the communication label, and the collision penalty is raised to -10. Placing the change of collision penalty at this boundary rather than at the introduction of the gate is deliberate, in that it allows the value functions to recalibrate to the altered reward scale during a transition that already perturbs them, rather than at the moment the gating head begins to learn.

Stage 3, from nine to fourteen million steps, activates the gating head and optimizes the joint policy against the communication penalty. The penalty coefficient is not applied at full strength immediately but follows the warm-up schedule of (4.14), rising linearly from zero to its target value of $\lambda^\star = 0.05$ over a window of two million steps, so that the pressure toward parsimony grows gradually while the policy adapts.

Table 5.1: The three training stages of Experiment 1. The gate is inactive throughout the first two stages, during which communication is unconstrained.

	Stage 1	**Stage 2**	**Stage 3**
Training steps	0–5M	5–9M	9–14M
Number of agents	8	32	32
Communication gate	inactive	inactive	active
Collision penalty	-2	-10	-10
Imitation learning rate	10%	0%	0%
λ_{comm}	—	—	0.05, warmed up over 2M

5.1.3 Experiment 2: Alternating Penalized and Unpenalized Episodes

The second experiment applies the alternating strategy of Section 4.3.1, and differs from the first in that no staged progression is imposed: the gate is active from the first step and training proceeds for thirty million steps with eight agents throughout. At the beginning of every episode the binary flag κ is drawn

with equal probability, determining whether the communication penalty applies during that episode. The flag is appended to the state vector so that the policy can distinguish the two regimes in which it is trained. Without it, identical states would yield different returns according to whether the current episode happened to be penalized, and the critic, unable to tell the two cases apart, could only learn their average. The penalty coefficient is held constant at $\lambda_{\text{comm}} = 0.01$ rather than following any schedule, since the alternation itself is intended to supply the protection against premature silence that the warm-up provides in the first experiment. Imitation learning is disabled throughout, and the collision penalty is set to -10 for the entire run. The network is trained from random initialization rather than warm-started from any earlier checkpoint.

5.2 Evaluation Protocol and Metrics

Evaluation follows the protocol established for the baseline [4], so that the results reported below may be placed alongside the published figures reproduced in Table 4.3. Fifteen scenarios are evaluated, combining team sizes of 8, 16, 32, 64 and 128 agents with correspondingly sized worlds and with obstacle densities of 0%, 15% and 30%, and each scenario is run for one hundred episodes on randomly generated maps. In contrast to training, the world size and the obstacle density are held fixed at the values stated for each scenario rather than sampled, so that every reported figure corresponds to the configuration under which it is labelled.

Two aspects of the protocol require comment because they concern the gate specifically. Actions are drawn stochastically from the policy distribution, as in the baseline evaluation, so that the comparison against published figures is not confounded by a change in action selection. The gating decision, by contrast, is taken deterministically by selecting the more probable of the two outcomes, since a policy deployed under bandwidth constraints would not sample its transmission decisions at random and a stochastic suppression at a critical bottleneck may produce a collision that a deterministic policy would avoid. In experiment 2 the penalty flag is fixed at $\kappa = 1$ throughout evaluation, so that the conditioned policy is assessed in the regime in which bandwidth is expensive.

Four metrics are inherited from the baseline and defined in Section 4.1.4: the episode length (EL), averaged over successful episodes alone; the maximum number of goals reached at any point during an episode (MR); the collision ratio with static obstacles and boundaries (CO); and the success rate (SR), being the percentage of episodes in which all agents reached their goals within the episode limit. To these is added the communication ratio, defined as

$$\text{CR} = \frac{\sum_t \sum_i c_i^t}{n \cdot T}, \tag{5.1}$$

in which c_i^t is the binary transmission decision of agent i at step t, n is the number of agents and T the length of the episode, so that the denominator counts the transmission opportunities available over the episode and the ratio expresses the fraction of them actually used. A value of unity corresponds to the unconditional communication of the baseline, in which every agent transmits at every step.

5.3 Results

The two gated models are denoted SCRIMP-Gated, produced by the curriculum of Section 5.1.2, and SCRIMP+ECNET, produced by the alternating strategy of Section 5.1.3. Their performance across the fifteen evaluation scenarios is reported in Table 5.2 for teams of up to 32 agents and in Table 5.3 for the two largest teams, alongside the published figures for unmodified SCRIMP. Standard deviations across the hundred episodes of each scenario are given in parentheses.

Table 5.2: Evaluation results for teams of up to 32 agents, averaged over 100 episodes per scenario with standard deviations in parentheses. The best success rate and the lowest communication ratio in each scenario are shown in bold.

Setting	Method	EL ↓	MR ↑	CO ↓	SR ↑	CR ↓
8 agents, 10 × 10, 0%	SCRIMP	12.05 (2.53)	8.00 (0.00)	0.00% (0.00%)	**100%**	100%
	SCRIMP-Gated	12.49 (2.77)	8.00 (0.00)	0.02% (0.15%)	**100%**	3.01% (3.12%)
	SCRIMP+ECNET	12.02 (2.54)	8.00 (0.00)	0.00% (0.00%)	**100%**	**0.00% (0.00%)**
8 agents, 10 × 10, 15%	SCRIMP	14.17 (3.52)	8.00 (0.00)	0.04% (0.18%)	**100%**	100%
	SCRIMP-Gated	13.58 (4.11)	7.97 (0.22)	0.01% (0.08%)	98%	5.25% (5.00%)
	SCRIMP+ECNET	14.70 (4.05)	8.00 (0.00)	0.01% (0.05%)	**100%**	**0.03% (0.17%)**
8 agents, 10 × 10, 30%	SCRIMP	18.56 (7.80)	7.93 (0.45)	0.07% (0.26%)	**97%**	100%
	SCRIMP-Gated	14.05 (4.41)	7.94 (0.28)	0.05% (0.24%)	95%	6.98% (6.75%)
	SCRIMP+ECNET	22.75 (16.70)	7.96 (0.24)	0.21% (0.50%)	**97%**	**0.53% (1.81%)**
16 agents, 20 × 20, 0%	SCRIMP	26.40 (3.87)	16.00 (0.00)	0.00% (0.00%)	**100%**	100%
	SCRIMP-Gated	27.57 (4.53)	16.00 (0.00)	0.00% (0.00%)	**100%**	1.91% (0.69%)
	SCRIMP+ECNET	26.37 (3.70)	16.00 (0.00)	0.00% (0.00%)	**100%**	**0.00% (0.00%)**
16 agents, 20 × 20, 15%	SCRIMP	27.75 (4.34)	16.00 (0.00)	0.02% (0.17%)	**100%**	100%
	SCRIMP-Gated	26.66 (4.69)	15.99 (0.10)	0.01% (0.04%)	99%	2.27% (1.38%)
	SCRIMP+ECNET	28.61 (7.43)	16.00 (0.00)	0.01% (0.04%)	**100%**	**0.03% (0.12%)**
16 agents, 20 × 20, 30%	SCRIMP	36.00 (8.75)	16.00 (0.00)	0.10% (0.21%)	**100%**	100%
	SCRIMP-Gated	29.66 (5.28)	15.91 (0.45)	0.02% (0.08%)	95%	2.97% (2.23%)
	SCRIMP+ECNET	41.05 (21.03)	15.99 (0.10)	0.10% (0.20%)	99%	**0.35% (0.74%)**
32 agents, 30 × 30, 0%	SCRIMP	42.59 (4.87)	32.00 (0.00)	0.00% (0.00%)	**100%**	100%
	SCRIMP-Gated	43.39 (6.99)	32.00 (0.00)	0.00% (0.00%)	**100%**	1.64% (0.47%)
	SCRIMP+ECNET	42.89 (4.86)	32.00 (0.00)	0.00% (0.02%)	**100%**	**0.00% (0.00%)**
32 agents, 30 × 30, 15%	SCRIMP	43.90 (5.11)	32.00 (0.00)	0.00% (0.02%)	**100%**	100%
	SCRIMP-Gated	45.84 (9.90)	31.95 (0.26)	0.00% (0.02%)	96%	1.65% (0.69%)
	SCRIMP+ECNET	46.31 (13.33)	32.00 (0.00)	0.01% (0.03%)	**100%**	**0.02% (0.04%)**
32 agents, 30 × 30, 30%	SCRIMP	62.31 (17.86)	31.94 (0.34)	0.32% (0.44%)	**96%**	100%
	SCRIMP-Gated	47.26 (9.49)	31.78 (0.58)	0.01% (0.04%)	84%	1.84% (1.27%)
	SCRIMP+ECNET	68.28 (23.22)	31.96 (0.20)	0.14% (0.21%)	**96%**	**0.23% (0.29%)**

Table 5.3: Evaluation results for the two largest teams, averaged over 100 episodes per scenario with standard deviations in parentheses. Both gated models were trained on teams substantially smaller than these, so the figures represent zero-shot generalization.

Setting	Method	EL ↓	MR ↑	CO ↓	SR ↑	CR ↓
64 agents, 40×40, 0%	SCRIMP	61.82 (6.13)	64.00 (0.00)	0.00% (0.00%)	**100%**	100%
	SCRIMP-Gated	66.43 (12.43)	63.92 (0.31)	0.00% (0.00%)	93%	1.15% (0.33%)
	SCRIMP+ECNET	60.75 (6.35)	64.00 (0.00)	0.00% (0.00%)	**100%**	**0.00% (0.00%)**
64 agents, 40×40, 15%	SCRIMP	65.44 (10.28)	63.99 (0.10)	0.02% (0.03%)	**99%**	100%
	SCRIMP-Gated	66.68 (11.34)	63.85 (0.43)	0.00% (0.01%)	88%	1.21% (0.49%)
	SCRIMP+ECNET	65.55 (7.01)	63.99 (0.10)	0.01% (0.02%)	**99%**	**0.06% (0.08%)**
64 agents, 40×40, 30%	SCRIMP	90.72 (29.16)	63.77 (1.23)	0.56% (1.17%)	**89%**	100%
	SCRIMP-Gated	71.84 (26.26)	62.59 (3.10)	0.02% (0.06%)	55%	1.26% (0.87%)
	SCRIMP+ECNET	109.37 (35.06)	63.73 (0.85)	0.32% (0.58%)	86%	**1.17% (0.88%)**
128 agents, 40×40, 0%	SCRIMP	66.86 (5.83)	128.00 (0.00)	0.01% (0.01%)	**100%**	100%
	SCRIMP-Gated	73.59 (12.65)	127.86 (0.42)	0.00% (0.01%)	88%	1.07% (0.32%)
	SCRIMP+ECNET	67.50 (5.63)	128.00 (0.00)	0.00% (0.01%)	**100%**	**0.00% (0.01%)**
128 agents, 40×40, 15%	SCRIMP	73.15 (8.53)	128.00 (0.00)	0.14% (0.13%)	**100%**	100%
	SCRIMP-Gated	77.88 (20.59)	126.97 (1.24)	0.01% (0.01%)	42%	0.84% (0.52%)
	SCRIMP+ECNET	82.48 (20.70)	127.98 (0.20)	0.08% (0.07%)	99%	**0.33% (0.19%)**
128 agents, 40×40, 30%	SCRIMP	135.43 (42.32)	121.95 (13.16)	3.72% (4.40%)	**58%**	100%
	SCRIMP-Gated	80.09 (17.38)	122.89 (6.57)	0.03% (0.04%)	23%	**1.35% (0.93%)**
	SCRIMP+ECNET	175.40 (37.55)	123.40 (10.98)	1.54% (1.72%)	52%	2.81% (0.86%)

5.3.1 The Alternating Model

SCRIMP+ECNET drastically reduces communication while preserving baseline performance across the majority of configurations. Its communication ratio never exceeds 2.81% and falls below 0.10% in nine of the fifteen scenarios, achieving a bandwidth reduction of more than 97% compared to the baseline's unconditional transmission. In exchange for this efficiency, its success rate matches the baseline in eleven scenarios and drops by more than two percentage points in only two cases—both occurring at the maximum 30% obstacle density for the 64- and 128-agent teams.

While the most demanding environments reveal a slight drop in overall success rate, the alternating model maintains robust collision avoidance. At 128 agents and 30% density, where the success rate falls to 52% against the baseline's 58%, the collision ratio is more than halved, dropping from 3.72% to 1.54%. Furthermore, the mean number of goals reached actually rises from 121.95 to 123.40 out of 128. A similar pattern appears at 64 agents and 30% density, where a drop in success rate (86% versus 89%) is accompanied by a reduction in collisions (0.32% versus 0.56%). This indicates that even when restricted bandwidth prevents the team from fully resolving extreme bottlenecks before the episode limit, the gating mechanism does not induce catastrophic collisions, successfully routing the vast majority of agents safely.

5.3.2 The Curriculum Model

SCRIMP-Gated reduces communication substantially, to between 0.84% and 6.98% depending upon the scenario, but does so at a cost that grows with team size. Up to 32 agents its success rate remains within a few points of the baseline, reaching 100% in four scenarios and falling to 84% only in the densest. Beyond

that point the degradation is pronounced: 55% at 64 agents and 30% density, 42% at 128 agents and 15%, and 23% at 128 agents and 30%, against baseline figures of 89%, 100% and 58% respectively.

This outcome is the more notable in that the curriculum model was trained upon teams of 32 agents while the alternating model was trained upon teams of eight, so that the configuration exposed to the greater congestion during training is the one that generalizes least well beyond it. Two features of its behaviour are nonetheless worth recording. Its collision ratio is uniformly low, never exceeding 0.05% and falling well below the baseline in every scenario, and its episode lengths in the dense instances are the shortest of the three methods, at 47.26 steps against the baseline's 62.31 for 32 agents at 30% density. The failures are therefore not collisions but timeouts, in which agents that cannot resolve a congested passage exhaust the episode limit while those that can do so complete their paths quickly.

5.4 Discussion

Three observations follow from the results, and each bears upon the question posed in Section 3.5.2, namely how much of the communication exchanged by an unconditional scheme is actually required.

5.4.1 The Cost of Silence

The reduction in communication achieved by the alternating model is very nearly free. Communication falls by more than 97% in every scenario, and across the nine scenarios involving 32 agents or fewer, the mean success rate is 99.1% against the baseline's 99.2%. This is a stronger statement than the ablation of Section 4.1.5 would suggest, since that ablation found the removal of communication to be costly in dense instances; the difference is that removing communication entirely deprives the team of information it needs at particular moments, whereas a learned gate retains the capacity to transmit precisely at those moments while remaining silent elsewhere. What the results indicate, therefore, is not that communication is unnecessary in this task, but that the moments at which it is necessary constitute a very small fraction of the episode.

5.4.2 Where the Gate Transmits

The communication ratio varies systematically with the difficulty of the instance, and in the direction the goal-oriented account predicts. For the alternating model it rises monotonically with obstacle density within each team size, from 0.00% to 0.03% to 0.53% for eight agents, from 0.00% to 0.03% to 0.35% for sixteen, and from 0.00% to 0.33% to 2.81% for the largest teams. The same monotonicity holds for the curriculum model, whose ratio rises from 3.01% to 5.25% to 6.98% across the three densities at eight agents. In sparse worlds, where agents rarely obstruct one another, both models fall almost completely silent; in the congested worlds where the ablation shows communication to be load-bearing, both transmit an order of magnitude more.

The dependence upon team size runs in the opposite direction, with the ratio falling as teams grow, from 3.01% to 1.07% for the curriculum model between the smallest and largest teams. This is a consequence of the definition in (5.1) rather than a reduction in absolute traffic: the denominator counts transmission

opportunities and therefore grows with the number of agents, so that a constant number of messages per timestep appears as a falling ratio. The absolute volume of traffic still grows with team size, though far more slowly than under the unconditional scheme.

5.4.3 Why Gating Improves Performance in Dense Instances

The most striking result is that both gated models surpass the baseline on collision ratio in every scenario. That gating should improve performance rather than merely preserve it requires explanation, and two mechanisms plausibly contribute.

The first is the raised collision penalty, which was introduced as an adaptation entailed by the gate rather than as an independent improvement, and which unavoidably confounds the comparison: a portion of the reduction in collisions is attributable to the penalty rather than to gating. The second is the chatter problem identified in Section 3.5.2. Under unconditional transmission the communication block attends over n messages at every timestep regardless of their informativeness, and as n grows the proportion of those messages that bear upon any given agent's decision falls.

5.5 Results Analysis

5.5.1 Measures of Communication Behaviour

Four quantities are reported, each averaged over the 100 evaluation episodes of every scenario. Let $c_i^t \in \{0, 1\}$ denote the transmission decision of agent i at step t, let n be the number of agents and T the length of the episode, and write

$$x_i = \sum_{t=1}^{T} c_i^t \tag{5.2}$$

for the number of transmissions made by agent i over the episode.

The communication ratio defined in (5.1) expresses the fraction of the available transmission opportunities that are actually used, and is plotted in Figure 5.1(a). The total number of transmissions per episode,

$$N_{\text{tx}} = \sum_{i=1}^{n} x_i, \tag{5.3}$$

measures instead the absolute load placed upon the network, and is plotted in Figure 5.1(b). Denoting by $x_{(1)} \leq x_{(2)} \leq \cdots \leq x_{(n)}$ the per-agent counts sorted in ascending order, the Gini coefficient

$$G = \frac{\sum_{i=1}^{n} (2i - n - 1)\, x_{(i)}}{n \sum_{i=1}^{n} x_{(i)}} \tag{5.4}$$

quantifies how unevenly the transmissions are distributed among the agents, taking the value zero when every agent transmits equally often and approaching unity as the traffic concentrates upon a single agent; it

is plotted in Figure 5.1(c). Finally the silence rate,

$$S = \frac{1}{n} \left| \{\, i \;:\; x_i = 0 \,\} \right|, \tag{5.5}$$

gives the fraction of agents that never transmit at any point during an episode, and is plotted in Figure 5.1(d).

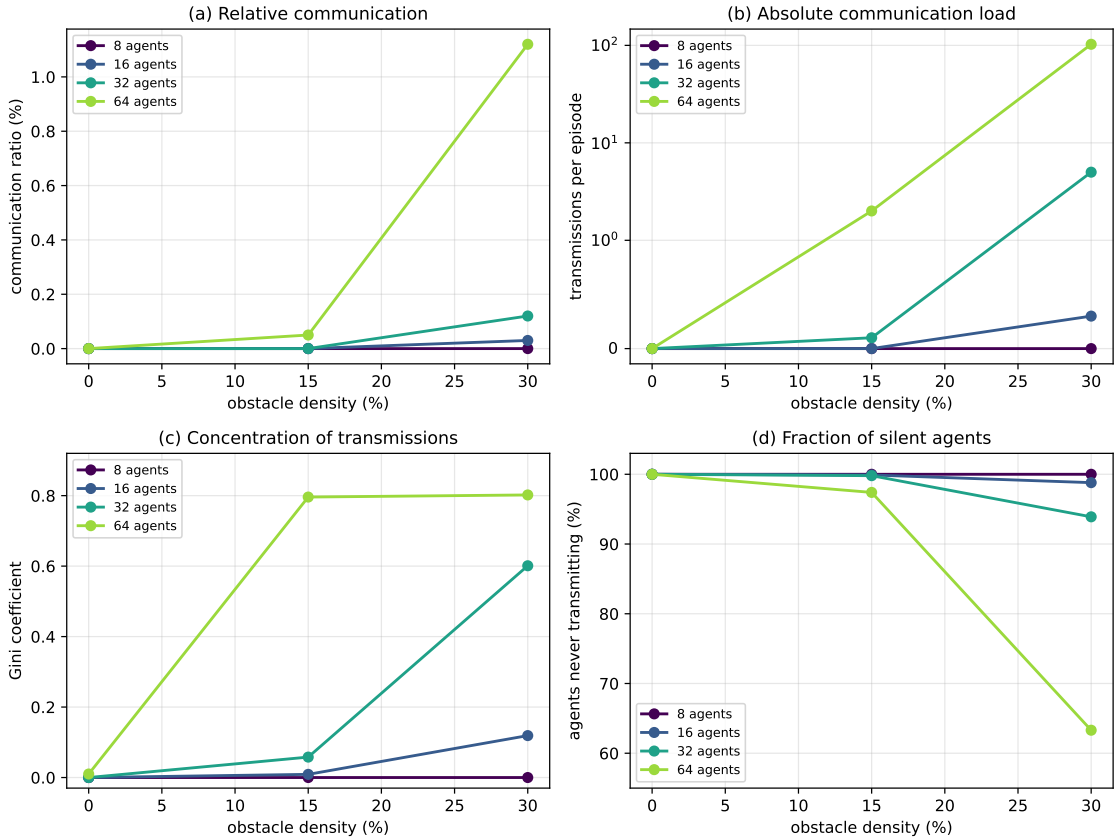

Figure 5.1: Communication behaviour of the gated policy in a 40×40 world, as a function of obstacle density and team size. Each point is averaged over 100 episodes. (a) the communication ratio of (5.1); (b) the absolute number of transmissions per episode, shown on a symmetric logarithmic scale; (c) the Gini coefficient of (5.4); (d) the silence rate of (5.5).

5.5.2 Communication Is Allocated Rather Than Merely Reduced

Fixing the environment at 40×40 and varying the team size isolates the effect of agent congestion from that of world size. Three findings follow.

The gate does not merely reduce communication uniformly; it transmits according to the difficulty of the instance. Across the whole sweep, communication remains at exactly zero for every configuration at 0% obstacle density, and for the 8-agent team it remains at zero at every density tested: in a 40×40 world eight agents rarely obstruct one another, and the policy discovers that it has nothing worth saying. Traffic appears

only where congestion makes coordination necessary, rising from 0.03% at 16 agents and 30% density to 1.12% at 64 agents at the same density.

The relative and absolute measures convey different information, and both are needed. Figure 5.1(a) shows the communication ratio never exceeding 1.12%, which understates the load a network would actually carry, whereas Figure 5.1(b) shows the same data as transmissions per episode, rising from a fraction of one to 102.6. The ratio measures efficiency and the total measures load, and the two diverge because the denominator of the ratio grows with the size of the team. Even at the busiest configuration, however, the 102.6 transmissions compare against the 6867 that unconditional communication would require over an episode of 107 steps, a reduction of 98.5%.

Finally, the traffic that remains is concentrated in a small minority of agents. At 64 agents and 30% density the Gini coefficient reaches 0.802 and only 37% of agents transmit at all; the single busiest agent accounts for 26.8 of the 102.6 transmissions, more than a quarter of all traffic and six times the mean among those agents that do transmit. The gate has therefore not learned to reduce communication uniformly, which a simple penalty might have produced, but to allocate it: a few agents, presumably those occupying contested passages, act as informants for a team that is otherwise silent. This behaviour was neither designed nor rewarded explicitly, which demonstrates that the learned policy is goal-oriented in the sense of Chapter 2 rather than merely economical.

One qualification applies to the Gini coefficient. Equation (5.4) is undefined when no agent transmits, and is reported as zero in that case, so the low values observed at 0% obstacle density indicate the absence of traffic rather than its equal distribution. Only the 32- and 64-agent configurations at 15% and 30% density carry sufficient traffic for the coefficient to be interpreted as a measure of concentration. The standard deviations are correspondingly large, the 102.6 transmissions of the busiest configuration carrying a deviation of 132.2, which indicates that the traffic is itself concentrated in a minority of the harder episodes.

6

Conclusion and Future Work

6.1 Summary

This thesis has addressed a gap between two lines of research that have developed largely independently of one another. The first, goal-oriented communication, judges a transmission by its contribution to the accomplishment of a goal, so that information leaving the behaviour of its recipients unchanged may be discarded before it is ever sent. Decentralized multi-agent path finding, meanwhile, has come to rely upon learned inter-agent communication in order to recover the coordination that decentralization forfeits, but does so unconditionally: every agent transmits at every timestep, irrespective of whether the message it sends bears upon any recipient's decision.

The work reported here has addressed that mismatch by making the decision to transmit part of what the policy learns. The SCRIMP architecture was extended with a gating head placed alongside the existing output heads, so that each agent decides at every timestep whether to refresh its contribution to the shared representation, conditioned upon what it has just received from its teammates as well as upon its own observation. Because the transmission decision is discrete and its consequences for the task are mediated by the environment and by a one-step message delay, the gate is trained not by backpropagation through the sample but by its own clipped surrogate objective sharing the advantage of the navigation policy, with a differentiable penalty upon the probability of transmitting supplying the pressure toward parsimony. A message cache held at the communication block ensures that the attention computation receives one entry per agent regardless of how many chose to remain silent.

Two strategies were evaluated for teaching the policy to satisfy both objectives at once: avoiding the transmission penalty, while still transmitting whenever communication carries information useful to achieving the goal. The first orders the task, training navigation and then coordination under congestion before the penalty is introduced and then raised gradually. The second, adapted from the ECNet formulation, alternates between penalized and unpenalized episodes while supplying the current regime to the network as an additional input, so that the policy is conditioned upon whether bandwidth is presently expensive. The

weights are unchanged between the two regimes; what differs is the input they are applied to.

The results establish that a very large proportion of the communication exchanged under the unconditional scheme is redundant. The alternating model reduces transmission by more than 97% across all fifteen evaluation scenarios while matching the success rate of the baseline in the majority of them, and in the most congested case evaluated, with 128 agents at 30% obstacle density, it solves 52% of instances against the baseline's 58% while halving the collision ratio and transmitting in just 2.81% of the available opportunities. The communication ratio of both gated models rises monotonically with obstacle density at every team size, which indicates that the gates have not simply learned to fall silent but have learned to concentrate what little they transmit upon the instances in which coordination is genuinely at stake.

6.2 Future Work

Four directions follow naturally from the recent literature on communication efficiency in multi-agent reinforcement learning.

6.2.1 Event-Triggered Execution

The gate developed here decides whether to communicate, but the agent continues to sample an action and to run a forward pass of the policy network at every timestep. Event-triggered control offers a more thorough economy, in which an agent recomputes its action only when a locally evaluated triggering condition is satisfied and otherwise holds its previous action. ET-MAPG [62] pursues exactly this, learning the control policy and the triggering policy jointly rather than treating the trigger as a fixed rule, with the next event time determined by an error signal measuring the divergence between the current state and the state at the last triggering instant. Its attention-based variant extends the principle to communication, so that an agent that does not resample its action also suppresses its message and the communication graph becomes sparse as a consequence of the triggering condition rather than through a separate mechanism. Applied to the present setting, this would reduce not only bandwidth but also the computational load at each agent, which is the binding constraint in the embedded platforms that decentralized path finding is ultimately intended for. The authors report that event-triggered agents may converge to slightly lower rewards than the time-triggered baseline while communicating far less, which is consistent with the trade-off observed in Chapter 5.

6.2.2 Bandwidth as an Explicit Constraint

The communication penalty employed here prices transmission through a coefficient chosen empirically, so that the resulting communication rate is an outcome of training rather than a specification. ETCNet [63] instead derives the penalty threshold from the available bandwidth by way of information theory, and formulates the design of the triggering strategy as a constrained Markov decision problem in which reinforcement learning is required to find the best protocol satisfying a stated bandwidth constraint. This is the more useful formulation for deployment, where the constraint is given by the physical link and the question is what performance can be achieved beneath it, rather than the reverse. Adopting it would also permit the reporting of results against a specified budget instead of against an unconstrained baseline.

6.2.3 Deciding to Whom to Transmit

The gate implemented in this work is global: an agent either contributes its message to the shared representation or it does not, and the transformer distributes the fused result to the entire team. A finer decision is possible, in which an agent selects the recipients of its message rather than merely whether to send one. The Transformer-based Email Mechanism [64] addresses this together with the scalability problem it creates, observing that targeted communication schemes which model every other agent as a potential recipient do not scale when the number of agents varies. Agents there send messages only to those they can observe, and information reaches agents beyond the observation range through forwarded message chains encoded and decoded by a transformer that selects the next recipient, in the manner of an email forwarded from one correspondent to another. The scheme maintains its performance when the team size changes without retraining, which bears directly upon the generalization gap reported in Section 5.3.2, where the curriculum model degraded sharply at team sizes beyond those seen during training.

6.2.4 Deciding How Precisely to Transmit

A third axis remains untouched by the present work. The gate decides whether a message is sent, but a message that is sent is transmitted at full precision, as a real-valued vector of the dimension fixed by the architecture. Differentiable Discrete Communication Learning [65] addresses the harder problem of learning how precisely to communicate, optimizing message precision at the bit level, where the discretization step ordinarily breaks the flow of gradients. The reported reduction in bandwidth exceeds an order of magnitude while matching or improving task performance, and the mechanism is presented as a plug-in layer applicable to any architecture. Combining it with the gating of Chapter 4 would yield a system deciding jointly whether to transmit and at what fidelity, which corresponds more closely to the goal-oriented account of Chapter 2 than either decision taken alone.

References

[1] E. Calvanese Strinati and S. Barbarossa, "6G networks: Beyond Shannon towards semantic and goal-oriented communications," *Computer Networks*, vol. 190, p. 107 930, 2021.

[2] T. M. Getu, G. Kaddoum, and M. Bennis, "A survey on goal-oriented semantic communication: Techniques, challenges, and future directions," *IEEE Access*, vol. 12, pp. 51 223–51 274, 2024.

[3] T. Charalambous, N. Pappas, N. Nomikos, and R. Wichman, "Toward goal-oriented communication in multi-agent systems: An overview," *arXiv preprint arXiv:2508.07720*, 2025.

[4] Y. Wang, B. Xiang, S. Huang, and G. Sartoretti, "SCRIMP: Scalable communication for reinforcement- and imitation-learning-based multi-agent pathfinding," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023.

[5] V. K. Vijay, H. Sheikh, S. Majumdar, and M. Phielipp, "Minimizing communication while maximizing performance in multi-agent reinforcement learning," *arXiv preprint arXiv:2106.08482*, 2021.

[6] W. Yang et al., "Semantic communications for future internet: Fundamentals, applications, and challenges," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 1, pp. 213–250, 2023.

[7] W. Wu, Y. Yang, Y. Deng, and A. H. Aghvami, "Goal-oriented semantic communications for robotic waypoint transmission: The value and age of information approach," *IEEE Transactions on Wireless Communications*, vol. 23, no. 12, pp. 18 903–18 915, 2024.

[8] W. Liu, Y. Deng, and H. Wymeersch, "Goal-oriented semantic communication for ISAC-enabled robotic obstacle avoidance," *arXiv preprint arXiv:2603.02291*, 2026.

[9] W. Xu, Y. Zhang, F. Wang, Z. Qin, C. Liu, and P. Zhang, "Semantic communication for the internet of vehicles: A multiuser cooperative approach," *IEEE Vehicular Technology Magazine*, vol. 18, no. 1, pp. 100–109, 2023.

[10] S. Chen, Q. Liao, A. Aijaz, and Y. Deng, "Goal-oriented multi-agent semantic networking: Unifying intents, semantics, and intelligence," *arXiv preprint arXiv:2512.01035*, 2025.

[11] F. Mason, F. Chiariotti, A. Zanella, and P. Popovski, "Multi-agent reinforcement learning for coordinating communication and control," *IEEE Transactions on Cognitive Communications and Networking*, vol. 10, no. 4, pp. 1566–1581, 2024.

[12] T.-Y. Tung, S. Kobus, J. P. Roig, and D. Gündüz, "Effective communications: A joint learning and communication framework for multi-agent reinforcement learning over noisy channels," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 8, pp. 2590–2603, 2021.

[13] G. K. Teja, P. K. Mohanty, and S. Das, "Review on path planning methods for mobile robot," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 239, pp. 5547–5580, 2025.

[14] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[15] R. Hoseinnezhad, "A comprehensive review of deep learning techniques in mobile robot path planning: Categorization and analysis," *Applied Sciences*, vol. 15, no. 4, p. 2179, 2025.

[16] A. Stentz, "The focussed D* algorithm for real-time replanning," in *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, vol. 2, Montreal, QC, Canada, 1995, pp. 1652–1659.

[17] S. M. LaValle, *Planning Algorithms*. Cambridge, UK: Cambridge University Press, 2006.

[18] C. Ferner, G. Wagner, and H. Choset, "ODrM*: Optimal multirobot path planning in low dimensional search spaces," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 3854–3859.

[19] M. B. Aremu, G. Ahmed, S. Elferik, and A.-W. A. Saif, "Autonomous mobile robot path planning techniques—a review: Metaheuristic and cognitive techniques," *Robotics*, vol. 15, no. 1, p. 23, 2026.

[20] T. Sheltami, G. Ahmed, and A. U. H. Yasar, "An optimization approach of IoD deployment for optimal coverage based on radio frequency model," *Computer Modeling in Engineering & Sciences*, vol. 139, pp. 2627–2647, 2024.

[21] S. Deshpande, A. K. Kashyap, and B. K. Patle, "A review on path planning AI techniques for mobile robots," *Robotic Systems and Applications*, vol. 3, pp. 27–46, 2023.

[22] K. Katona, H. A. Neamah, and P. Korondi, "Obstacle avoidance and path planning methods for autonomous navigation of mobile robot," *Sensors*, vol. 24, p. 3573, 2024.

[23] G. Peng, C. Yang, W. He, and C. P. Chen, "Force sensorless admittance control with neural learning for robots with actuator saturation," *IEEE Transactions on Industrial Electronics*, vol. 67, pp. 3138–3148, 2019.

[24] S. V. Albrecht, F. Christianos, and L. Schäfer, *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*. Cambridge, MA, USA: MIT Press, 2024.

[25] R. Singh, J. Ren, and X. Lin, "A review of deep reinforcement learning algorithms for mobile robot path planning," *Vehicles*, vol. 5, pp. 1423–1451, 2023.

[26] S. Thrun, "Probabilistic algorithms in robotics," *AI Magazine*, vol. 21, no. 4, p. 93, 2000.

[27] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *International Journal of Robotics Research*, vol. 32, pp. 1238–1274, 2013.

[28] Y. Tang et al., "Perception and navigation in autonomous systems in the era of learning: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, pp. 9604–9624, 2022.

[29] Y. Zhao, Y. Zhang, and S. Wang, "A review of mobile robot path planning based on deep reinforcement learning algorithm," *Journal of Physics: Conference Series*, vol. 2138, p. 012 011, 2021.

[30] H. Sun, W. Zhang, R. Yu, and Y. Zhang, "Motion planning for mobile robots—focusing on deep reinforcement learning: A systematic review," *IEEE Access*, vol. 9, pp. 69 061–69 081, 2021.

[31] M. Nguyen and R. Dubay, "Application of deep learning in autonomous mobile robot control: An overview," in *Proceedings of the IEEE International Systems Conference (SysCon)*, Montreal, QC, Canada, 2025.

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

[33] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[34] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proceedings of the 35th International Conference on Machine Learning*, Stockholm, Sweden, 2018.

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

[36] K. Okumura, "Engineering LaCAM*: Towards real-time, large-scale, and near-optimal multi-agent pathfinding," in *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2024.

[37] G. Sartoretti et al., "PRIMAL: Pathfinding via reinforcement and imitation multi-agent learning," 3, vol. 4, 2019, pp. 2378–2385.

[38] T. Standley, "Finding optimal solutions to cooperative pathfinding problems," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2010, pp. 173–178.

[39] A. Shankar, K. Okumura, and A. Prorok, "LF: Online multi-robot path planning meets optimal trajectory control," *arXiv preprint arXiv:2507.11464*, 2025.

[40] B. Wang, Z. Liu, Q. Li, and A. Prorok, "Mobile robot path planning in dynamic environments through globally guided reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6932–6939, 2020.

[41] S. Wen, Z. Wen, D. Zhang, H. Zhang, and T. Wang, "A multi-robot path-planning algorithm for autonomous navigation using meta-reinforcement learning based on transfer learning," *Applied Soft Computing*, vol. 110, p. 107 605, 2021.

[42] H. Bae, G. Kim, J. Kim, D. Qian, and S. Lee, "Multi-robot path planning method using reinforcement learning," *Applied Sciences*, vol. 9, p. 3057, 2019.

[43] V. Gabler and D. Wollherr, "Decentralized multi-agent reinforcement learning based on best-response policies," *Frontiers in Robotics and AI*, vol. 11, p. 1 229 026, 2024.

[44] Z. Song, R. Zhang, and X. Cheng, "HELSA: Hierarchical reinforcement learning with spatiotemporal abstraction for large-scale multi-agent path finding," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023.

[45] J. Jung, M. Shun, and B. Krishnamachari, "Coordinating autonomous vehicles using reinforcement learning," in *Proceedings of SoutheastCon*, 2025, pp. 964–970.

[46] S. Peng, "Multi-robot path planning combining heuristics and multi-agent reinforcement learning," *arXiv preprint arXiv:2306.01270*, 2023.

[47] T. Phan, J. Driscoll, J. Romberg, and S. Koenig, "Confidence-based curriculum learning for multi-agent path finding," *arXiv preprint arXiv:2401.05860*, 2024.

[48] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th International Conference on Machine Learning (ICML)*, 2009.

[49] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone, "Curriculum learning for reinforcement learning domains: A framework and survey," *Journal of Machine Learning Research*, vol. 21, no. 1, 2020.

[50] M. Damani, Z. Luo, E. Wenzel, and G. Sartoretti, "PRIMAL$_2$: Pathfinding via reinforcement and imitation multi-agent learning — lifelong," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2666–2673, 2021.

[51] Z. Ma, Y. Luo, and H. Ma, "Distributed heuristic multi-agent path finding with communication," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 8699–8705.

[52] Y. Wang and G. Sartoretti, "FCMNet: Full communication memory net for team-level cooperation in multi-agent systems," in *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2022, pp. 1355–1363.

[53] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, "A simple neural attentive meta-learner," *arXiv preprint arXiv:1707.03141*, 2017.

[54] E. Parisotto et al., "Stabilizing transformers for reinforcement learning," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2020.

[55] P. Sunehag et al., "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2018.

[56] J. Li, G. Gange, D. Harabor, P. J. Stuckey, H. Ma, and S. Koenig, "New techniques for pairwise symmetry breaking in multi-agent path finding," in *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, vol. 30, 2020, pp. 193–201.

[57] N. Savinov et al., "Episodic curiosity through reachability," *arXiv preprint arXiv:1810.02274*, 2019.

[58] Z. Ma, Y. Luo, and J. Pan, "Learning selective communication for multi-agent path finding," 2022.

[59] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, pp. 229–256, 1992.

[60] A. Singh, T. Jain, and S. Sukhbaatar, "Learning when to communicate at scale in multiagent cooperative and competitive tasks," in *International Conference on Learning Representations (ICLR)*, 2019.

[61] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," in *International Conference on Learning Representations (ICLR)*, 2017.

[62] U. Siddique, A. Sinha, and Y. Cao, "Adaptive event-triggered policy gradient for multi-agent reinforcement learning," *arXiv preprint arXiv:2509.20338*, 2025.

[63] G. Hu, Y. Zhu, D. Zhao, M. Zhao, and J. Hao, "Event-triggered communication network with limited-bandwidth constraint for multi-agent reinforcement learning," *arXiv preprint arXiv:2010.04978*, 2020.

[64] X. Guo, D. Shi, and W. Fan, "Scalable communication for multi-agent reinforcement learning via transformer-based email mechanism," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2022.

[65] A. Kapoor, Y. Bhisikar, B. Freed, J. Peters, and M. Sun, "Learning what to say and how precisely: Efficient communication via differentiable discrete communication learning," *arXiv preprint arXiv:2511.01554*, 2025.

Disclaimer on the Use of AI

This manuscript includes portions written with the support of a generative artificial intelligence tool (Claude). This tool was used exclusively for linguistic review, text formatting, and drafting assistance; all analyses, results, and scientific content remain the sole responsibility of the author.