Policy synthesis for collective dynamics

Policy synthesis for collective dynamics

Paul Piho and Jane Hillston

University of Edinburgh, UK

Abstract. In this paper we consider the problem of policy synthesis for systems of large numbers of simple interacting agents where dynamics of the system change through information spread via broadcast communication. By modifying the existing modelling language CARMA and giving it a semantics in terms of continuous time Markov decision processes we introduce a natural way of formulating policy synthesis problems for such systems. However, solving policy synthesis problems is difficult since all non-trivial models result in very large state spaces. To combat this we propose an approach exploiting the results on fluid approximations of continuous time Markov chains to obtain estimates of optimal policies.

1 Introduction

The study of collective dynamics has a wealth of interesting applications in collective adaptive systems (CAS) where examples range from swarming behaviour of insects to patterns of epidemic spread in humans. Such systems are highly distributed and robust in nature and for that reason are an interesting paradigm for the design of highly-distributed computer-based systems.

The study of such systems concentrates on the emergent behaviour arising from simple behaviour and communication rules at the level of individuals. However, due to CAS exhibiting non-linear dynamics it is difficult to verify or predict the emergent behaviour. It is harder still to know a priori how to design the behaviour and capabilities of individual agents in order to achieve a system level goal. Moreover, the individual agents in such systems are often unreliable and prone to failure making it natural to express objectives at the collective level in terms of a proportion of the population achieving a goal within a time bound.

We seek to develop a framework in which the flexibility afforded by having a homogeneous population of agents can be leveraged in planning decisions and explore how communication can alter the likelihood of satisfying the collective goals. The basis of the framework is a process algebra supporting formal expression of basic behaviour of population members, including patterns of communication in the collective. Based on the formal specification the policy synthesis problems are framed in terms of continuous time Markov decision processes (CT-MDP) which give a versatile framework for a variety of stochastic control and planning problems.

Policy synthesis for CT-MDP models is a computationally complex problem in general. Consequently, there have been a number of recent works on policy synthesis and closely related model checking of CT-MDPs e.g. [1,2]. Our problem

domain, the collective dynamics of homogeneous agents, offers a way to approximate system dynamics through fluid approximation describing the state of the system in terms of continuous variables. In many cases this alleviates the problem of state space explosion. We study leveraging fluid approximation results, inspired by similar results for discrete time Markov decision processes [3], in the context of policy synthesis for collective dynamics. We note that broadcast communication is a natural way to model information spread in collective systems. However, this makes it impossible to directly apply the existing results like [4]. Thus, we propose an approximations for a class of systems involving broadcast communication.

The paper makes the following contributions. Firstly, we propose a class of systems arising from collective systems involving broadcast communication. Secondly, we present a process algebra based on the existing language Carma [5] which together with specification of goals from [6] provides a high level framework for formulating policy synthesis problems. Next, we propose an efficient policy synthesis approach, exploiting the structure suggested by the language level description, to find configurations that can satisfy the defined collective goals. We concentrate on applications of fluid approximation results in cases involving broadcast communication where standard results do not apply. Finally, we frame a simplified foraging example, inspired by the design of a robot swarm, in the presented framework and consider the policy synthesis.

The paper is structured as follows: in Section 2 we detail the formal specification of policy synthesis for collective dynamics. Particularly, in Section 2.1 we introduces a class of systems arising from broadcast communication in collectives being treated as a switch in the dynamics. The Section 2.2 outlines a process algebraic language with its semantics for specifying models in the CTMDP framework. In Section 3 we discuss ideas arising from fluid approximations and adapting them for policy synthesis for collective dynamics. In particular, we suggest an approximation, based on, fluid results for dealing with broadcast communication. In Section 4 we present a simplified case-study inspired by swarm robotics to motivate the developed framework. Finally, we end with concluding remarks and discussion of further work in Section 5.

2 System specification

Carma process algebra Carma [5] is a stochastic process algebra designed to support specification and analysis of CAS. A Carma system consists of a *collective* operating in an *environment*. The collective describes a set of interacting *agents* and models the behaviour of a system. The description of an agent consist of a *process*, that describes the agent's behaviour, and of a *store*, that models its *knowledge*. Here *knowledge* is limited to the values of key attributes.

The processes described within components interact via a rich set of communication primitives. In particular, the language supports both broadcast and unicast communication. Both are attribute-based so that a component can only

receive a message when its store satisfies the sender's target predicate and receivers use a predicate to identify acceptable message sources.

The environment is responsible for setting the rates and probabilities governing action execution and message exchange. Thus the environment models all aspects intrinsic to the context the system operates in, e.g., the rate at which a component moves may depend on the terrain at the given location.

The formal semantics give rise to a continuous time Markov chain (CTMC) – the state space of the system is represented as a finite, discrete set of states and the times of transitions are governed by the rates given in the model description where each rate is taken to be the parameter of an exponential distribution.

Continuous-time Markov decision processes Our target mathematical object for policy synthesis is a continuous-time Markov decision process (CT-MDPs) rather than a CTMC. CT-MDPs are a common framework in stochastic control theory and operations research providing a natural formalisation of policy optimisation problems. Here we give relevant standard definitions for CT-MDPs.

Definition 1. *Continuous-time Markov decision process is defined by the tuple* $\{\mathcal{S}, \mathcal{A}, \{\mathcal{A}(i), i \in \mathcal{S}\}, q(j \mid i, a)\}$ *where* $\mathcal{S}$ *is the countable set of states,* $\mathcal{A}$ *is the Borel measurable set of actions,* $\{\mathcal{A}(i), i \in \mathcal{S}\}$ *is the set of feasible actions in state* i *and* $q(j \mid i, a)$ *gives the transition rates* $i \to j$ *given the control action* a*.*

The evolution of CT-MDPs is described by the following: after the process reaches some state and an action is chosen the process performs a transition to the next state depending only on the current state and the chosen action. The time it takes for state transitions to happen is governed by exponential distributions with rates given by the function q in Definition 1. The actions at every such step are chosen according to some policy as defined below.

Definition 2. *A policy is a measurable function* $\pi : \mathbb{R}_{\geq 0} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$ *which for every time* $t \in \mathbb{R}_{\geq 0}$*, state* $s \in \mathcal{S}$ *and action* $a \in \mathcal{A}(s)$ *assigns a probability* $\pi(t, s, a)$ *that the action* a *is chosen in* s *at time* t*. We call a policy where for every* $t \in \mathbb{R}_{\geq 0}$ *and* $s \in S$ *we have that* $\pi(t, s, a) \in \{0, 1\}$ *a deterministic policy. A policy* π *independent of* t *is a stationary policy.*

For the purpose of leveraging fluid approximations we introduce the concept of population CT-MDPs which result from models where components are only distinguished through their state. Thus the state of the system is represented as a vector of counting variables detailing the number of components in each state.

Definition 3. *A population CT-MDP is a tuple* $(\mathbf{X}, \mathcal{T}, \mathcal{A}, \beta)$ *defined by:* $\mathbf{X} = (X_1, \cdots, X_n) \in \mathcal{S} = \mathbb{Z}^n_{\geq 0}$ *where each* X_i *takes values in a finite domain* $\mathcal{D}_i \subset \mathbb{Z}_{\geq 0}$*;* β *is a function such that* $\beta(a, \mathbf{X})$ *returns a boolean value indicating whether action* $a \in \mathcal{A}$ *is available from state* $\mathbf{X}$*;* $\mathcal{T}$ *is a set of transitions of the form* $\tau = (a, \mathbf{v}_\tau, r_\tau(\mathbf{X}))$ *such that* $\beta(a, \mathbf{X}) = 1$*,* $\mathbf{v}_\tau$ *is an update vector specifying that the state after execution of transition* τ *is* $\mathbf{s} + \mathbf{v}_\tau$ *and* $r_\tau(\mathbf{X})$ *is a rate function.*

A population CT-MDP is associated with a CT-MDP in the following way: the state and action space of the corresponding CT-MDP is the same as for the population CT-MDP; the set of feasible actions for state $\mathbf{i} \in \mathcal{S}$, denoted $\mathcal{A}(\mathbf{i})$, is defined by $\mathcal{A}(\mathbf{i}) = \{a \in \mathcal{A} \mid \beta(a, \mathbf{i}) = 1\}$; the rate function q is defined as

$$q(\mathbf{i} \mid \mathbf{j}, a) = \sum \{r_\tau(\mathbf{j}) \in \mathcal{T} \mid \tau = (a, \mathbf{v}_\tau, r_\tau(\mathbf{j})) \wedge \mathbf{i} = \mathbf{j} + \mathbf{v}_\tau\}$$

2.1 Broadcast communication

Broadcast is a natural communication pattern to consider in CAS. However, in general it makes it impossible to apply the fluid approximation results which informally rely on the effect of actions being bounded as more components are introduced to the system. This limits the usefulness of fluid approximation methods when analysing collective systems.

We propose a class of population systems for which the problems arising from broadcast communication can be mitigated. In particular, we consider cases where broadcast can be thought of as a switch between dynamic modes of the population. More generally, we consider population processes in the CT-MDP framework with the mode switching dynamics as described in the following. Let $\mathcal{M} = (\mathbf{X}, \mathcal{T}, \mathcal{A}, \beta)$ be a population CT-MDP with $\mathbf{X} = (X_1, \cdots, X_{2n})$. The model $\mathcal{M}$ exhibits mode switching dynamics if, up to reordering of variables, there exist $\mathbf{X}_1 = (X_1, \cdots, X_n)$ and $\mathbf{X}_2 = (X_{n+1}, \cdots, X_{2n})$ ($\mathbf{X}$ is the concatenation of $\mathbf{X}_1$ and $\mathbf{X}_2$) such that

- $\mathbf{X}_1 \neq \mathbf{0}$ if and only if $\mathbf{X}_2 = \mathbf{0}$ – the system can be in one mode or the other.
- there is a transition $(a, \mathbf{v}_\tau, r_\tau(\mathbf{X})) \in \mathcal{T}$ with an update vector $(-s_1, \ldots, -s_n, s_1, \ldots, s_n)$ that happens with a non-zero rate from state $(s_1, \ldots, s_n, 0, \cdots, 0)$. This ensures there exists at least one transition between the modes.
- There is no state $(0, \cdots, 0, s_{n+1}, \cdots, s_n)$ for which there exists a transition with an update vector $(s_{n+1}, \cdots, s_n, -s_{n+1}, \cdots, -s_n)$ happening at a non-zero rate – we consider models where the mode switching is unidirectional.
- $(\mathbf{X}_1, \mathcal{T}, \mathcal{A}, \beta)$ and $(\mathbf{X}_2, \mathcal{T}, \mathcal{A}, \beta)$ define a population CT-MDP.

Although the definition is given for two dynamic modes we can easily extend it. As mentioned, such population models can arise from considering broadcast communication. In particular, we study situations where broadcast is used to propagate knowledge acquired by a component in the system to the rest of the components leading to a change in component behaviour. Such knowledge, once acquired, is not lost or forgotten leading to unidirectional mode changes. In the following section we introduce a process algebra based language for constructing such models and set up our running example of mode switching.

2.2 Language and configuration

We propose a process algebra based language, CARMA-C, which uses a subset of syntactic constructs of CARMA to simplify the creation of the described

population models. We focus on models of collectives involving broadcast communication and thus retain the constructs of CARMA for broadcast communication, knowledge stores and the environment. For specification of policy synthesis problems, we introduce non-determinism in transitions interpreted as possible control actions. The control actions are encoded in terms of attributes in knowledge stores where the values of such attributes are left partially specified — instead of particular values we define the value domains for the attributes.

As unicast and attribute based communication are not the focus of this paper we do not carry over the syntactic constructs for these from CARMA. However, note that the material in this section can easily be extended to specify systems making use of unicast and attribute based communication.

2.3 Syntax

As in CARMA, we say a system consists of a collective N operating in an environment $\mathcal{E}$. We let SYS be the set of systems S, and COL be the set of collectives N where a collective is either a component C in the set of components COMP or the parallel composition of collectives. A component C can either be the inactive component, denoted $\mathbf{0}$, or a term of the form (P, γ) where P is a process and γ is a store. In particular, systems, collectives and components are generated by the following grammar:

$$S ::= N \,\mathbf{in}\, \mathcal{E} \qquad N ::= C \mid N_1 \parallel N_2 \qquad C ::= \mathbf{0} \mid (P, \gamma)$$

The grammar for the processes is given by the following:

$$\begin{aligned} P, Q &::= \mathbf{nil} \mid act.P \mid P + Q \mid P \mid Q \mid [\pi]P \mid A \; (A \triangleq P) \\ act &::= \alpha^*\langle \vec{e} \rangle \sigma \mid \alpha^* \, (\vec{e}) \, \sigma \end{aligned}$$

The processes are defined using standard constructs — action prefix, choice, and parallel composition — from process algebras literature. In addition we allow the definition of the inactive process **nil** and guards on processes. The action primitives are defined for broadcast output in the form $\alpha^*\langle \vec{e} \rangle \sigma$ and broadcast input in the form $\alpha^* \, (\vec{e}) \, \sigma$. The broadcast output action is defined as non-blocking and an output action with no corresponding input is interpreted as a spontaneous action of a component.

The following notation is used: α is an action type used to distinguish between different actions; $\vec{e}$ is an expression specifying the message sent over broadcast communication; π is a boolean expression such that a process guarded by π is only activated when π evaluates to true; σ is an update specifying a probability distribution over the possible store configurations following the given action.

Example 1. For the running example we consider a scenario where robots need to locate a source by sensing their local environment and move to the location of the source. Components in the system correspond to the robots with the following behaviour: robots can move on a grid, take measurements from their

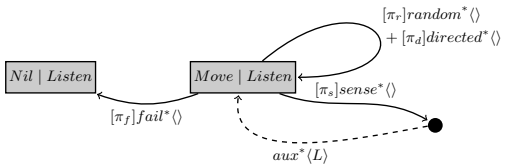


Fig. 1: Behaviour of individual *Robot* components.

location and broadcast this information to the rest of the swarm. The model we use to describe the behaviour of the robots is illustrated in Figure 1. The action $random^*$ corresponds to the robot exploring the environment through a random walk while $directed^*$ corresponds to moving towards a found source location. The action $sense^*$ models the robot taking measurements of its locale. If a source is detected then the auxiliary action aux^* immediately broadcasts the set of source locations L. The action $fail^*$, resulting in the robot not performing further actions, models failure. Finally, the process $Listen$ receives the corresponding broadcast input action $aux^*(L)$.

2.4 Semantics

The novelty of relating a CARMA-C model to a CT-MDP lies in defining the set of admissible controls via the stores. In particular, instead of fully specifying store variables, as done in CARMA, we allow them to take values in a general Borel measurable set defined in the model. The set of feasible actions (as in Defn 1) then corresponds to possible refinements of stores to particular values.

Store In CARMA a store is a function that maps attribute names to particular values which are then used in the semantics for the transition rate calculations. In CARMA-C, we instead define the store as a function that maps attribute names to permitted value domains. That is, a store γ maps a set of attribute names $a_0, \cdots, a_n$ in its domain to the value domains of the attributes. This introduces non-determinism in the choice of particular store values. Such non-determinism for system specification has previously been considered in the case of interval Markov chains (IMC) [7], constraint Markov chains (CMC) [8] and probabilistic constraint Markov chains (PCMC) [9]. The non-determinism in these cases is treated as arising from a transition probability or a rate for which the true value is not known or that the probability can take any value in the given region. In our approach the non-determinism will be resolved by a policy maker.

Example 2. For the running example we define the local store of each robot consisting of attributes location giving the location of the robot, and source holding the set of locations identified as source. Figure 2 illustrates the effects of actions on the local stores. In particular, $random^*$ and $directed^*$ change the location of the robot. The former picks with uniform probability a target location reachable from location (x, y) – the corresponding update is denoted by $R(x, y)$. The latter picks a location that takes the robot closer to the source L – update denoted by

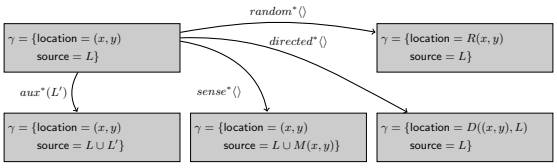


Fig. 2: Local component store changes induced by actions.

$D((x, y), L)$. If the robot's location corresponds to a source location the action *sense** includes the location in the set of sources with some probability thus modelling the possibility of false negatives resulting from noisy measurements. In particular, we define M such that if the location (x, y) is a source then M returns (x, y) with probability p and the empty set with probability $1 - p$. The input action $aux^*(L')$ adds the locations L' in the message to the set of sources.

We use guards to achieve the desired behaviours of components. Specifically, the guard π_r for *random** is true when the attribute source corresponds to the empty set – source location has not yet been found. For simplicity, we also allow the action *sense** only if the source has not been discovered. Conversely, the guard for *directed** is true when the attribute source defines a non-empty set of locations. The guard π_f for *fail** evaluates to true everywhere except when the robot is at the source location.

Environment Like in CARMA, the environment in CARMA-C models aspects intrinsic to the context where the agents under consideration are operating – it sets the rates of actions and mediates the interactions between components.

For a system $S \in$ SYS we say that the environment $\mathcal{E}$ is defined by the global store γ_g and an evolution rule ρ. The evolution rule ρ is a function which given a global store γ_g and the current state of the collective $N \in$ COL returns a tuple of functions $\varepsilon = \langle \mu_p, \mu_r, \mu_u \rangle$ called the evaluation context. The functions given by the evaluation context are interpreted as follows: μ_p expresses the probability of receiving a broadcast; μ_r specifies the execution rates of actions; μ_u determines the updates on the environment store.

Example 3. For the running example we define the following environment: the global store γ_g defines a store attribute failr $\in [0, 1)$; the probability of receiving the broadcast for the aux^* action is set to 1; the constant rate of spontaneous actions *random** and *directed** is given by r_m and the rate of *sense** is given by r_s; the action aux^* emulates an instantaneous action with its rate set very high; the rate r_f of *fail** is set equal to the store attribute failr introducing non-determinism in the behaviour of the system; we set μ_u such that the global store remains unchanged through the evolution.

Resolving non-determinism The semantics for the construction of a CT-MDP model from a syntactic description of a CARMA-C model is done in two stages. The first part of the semantics resolves the non-determinism in the model. In particular, we consider a system S defined by $(P_1, \gamma_1) \mid \cdots \mid (P_n, \gamma_n)$ **in** (γ_g, ρ).

The set of control actions $\mathcal{A}(S)$ available from S is defined by the following: let $\mathcal{A}(S)$ be a set of functions such that for all $\gamma \in \{\gamma_1, \cdots, \gamma_n, \gamma_g\}$, $f \in \mathcal{A}(S)$ maps all attributes a in the domain of γ to particular value in $\gamma(a)$. That is, a set of feasible control actions from a system S corresponds to the set of possible functions that fix the store values. In the formal semantics we introduce a refinement step labelled by a chosen control action f that transforms S into

$$S_f \stackrel{\text{def}}{=} (P_1, f(\gamma_1)) \mid \cdots \mid (P_n, f(\gamma_n)) \textbf{ in } (f(\gamma_g), \rho)$$

Let us define the sets $\textsc{Sys}^f$, $\textsc{Col}^f$ and $\textsc{Comp}^f$ as sets of systems, collectives and components after application of f. We assume that elements of $\textsc{Sys}^f$, $\textsc{Col}^f$ and $\textsc{Comp}^f$ are derived only from elements in Sys, Col and Comp for which f is sufficient to fully resolve the non-determinism in the behaviour. We call such sets *resolved systems, collectives* and *components*, respectively.

Example 4. For the running example the set of control actions corresponds to the possible assignments of failr. Lower values of failr correspond to lower failure rate and thus more robust components.

Interleaving semantics The second stage of the semantics determines the rates at which a system changes state given a control action. This is achieved through construction of functions $\mathbb{C}_f$, $\mathbb{N}_{\varepsilon,f}$ and $\mathbb{S}_f$ parametrised by a chosen control action f. In particular, the function $\mathbb{C}_f$ takes a resolved component in $\textsc{Comp}^f$ and an action label and returns a probability distribution over components in Comp. Components assigned a non-zero probability are reachable from the resolved component. The function, $\mathbb{N}_{\varepsilon,f}$ builds on $\mathbb{C}_f$ to describe the behaviour of collectives. Based on a resolved collective in $\textsc{Col}^f$ and an action label and it returns a probability distribution over Col. As before non-zero probabilities are assigned to reachable collectives. Finally, function $\mathbb{S}_f$ takes a resolved system in $\textsc{Sys}^f$ and an action label and returns a function over systems Sys that specifies the rate at which the transitions happen. As for Carma, these function are constructed via FuTS-style [10] operational semantics. The semantic rules for the second step resulting in the transition rates between systems closely match the semantics given for Carma in [5] and are not detailed here.

Population model The population CT-MDP model $\mathcal{M} = (\mathbf{X}, \mathcal{T}, \mathcal{A}, \beta)$ for a system $S \in \textsc{Sys}$ can be derived iteratively based on the assumption that components with the same configuration (same process state and store) are indistinguishable. We start with S consisting of a collective $C_1 \parallel \cdots \parallel C_N$ operating in an environment $\mathcal{E}$. The function $\mathbb{C}_f$ can be used to determine all possible future configurations of each of the components C_i. If the union of all possible component configurations is finite we can define the finite state space $\mathcal{S}$ of $\mathcal{M}$ as the space of counting vectors specifying all possible future configurations of S.

For each state $\mathbf{s} \in \mathcal{S}$ we have a set $Sys_{\mathbf{s}} \subset \textsc{Sys}$ of corresponding Carma-C systems. For each system $S \in Sys_{\mathbf{s}}$ the set of feasible actions will be the same

by construction. For the derivation of the population model we add a restriction that any control action acts in the same way on the set of indistinguishable components. The rates corresponding to chosen actions and the reachable states are found using the function $\mathbb{S}_f$. In particular, given a control action f denote the system S resolved by f by S_f. The rate of transition from $\mathbf{s} \in \mathcal{S}$ to $\mathbf{s}' \in \mathcal{S}$ given control action f is then given by

$$\sum_{S \in Sys_{\mathbf{s}}} \sum_{S' \in Sys_{\mathbf{s}'}} \sum_{\ell \in \text{LAB}_S} \mathbb{S}_f[S_f, \ell](S')$$

3 Policy synthesis

The main contribution in this paper is a method leveraging fluid approximation results for CTMCs to policy synthesis in the context of collective dynamics involving broadcast communication. Here we state the relevant optimisation problem and discuss how fluid approximation results can be exploited.

Fluid approximations of CTMCs The aim of fluid approximation of CTMCs, as introduced by Kurtz in [4], is to derive a set of ordinary differential equations (ODEs) for which the sample paths of the CTMC lie, with high probability, close to the solution of the chosen set of ODEs.

Consider a system of N components each evolving in a finite state space $SS = \{1, \cdots, K\}$ and where components are only distinguishable through their state. Let the state of the object n at time t be denoted by $Y_n^{(N)}(t)$. Let the variable $\mathbf{X}^{(N)}(t) \in \mathbb{R}^K$ be a counting vector giving the state of the system at time t. In particular, the i-th entry of $\mathbf{X}^{(N)}(t)$ is given by $X_i^{(N)}(t) = \sum_n \mathbb{1}\{Y_n^{(N)}(t) = i\}$.

Next consider the set of transitions, denoted $\mathcal{T}^{(N)}$, consisting of elements $\tau = (R_\tau, r_\tau^{(N)})$ where R_τ is a multi-set of update rules of the form $i \to j$ specifying that an agent in state i goes to state j if the transition τ fires. The $r_\tau^{(N)}$ denotes a rate function $r_\tau^{(N)} : \mathbb{R}^K \to \mathbb{R}_{\geq 0}$ depending on the state of the system. We assume that R_τ is independent of the population size N — all transitions involve a finite and fixed number of individuals. The update vector $\mathbf{v}_\tau$ is constructed from R_τ so that the transition τ changes the state of $\mathbf{X}^{(N)}$ to $\mathbf{X}^{(N)} + \mathbf{v}_\tau$. We define the Markov population model by a tuple $\mathcal{X}^{(N)} = (\mathbf{X}^{(N)}, \mathcal{T}^{(N)}, \mathbf{X}_0^{(N)})$ where $\mathbf{X}_0^{(N)}$ denotes the initial state of the system. Given $\mathcal{X}^{(N)}$ it is trivial to construct the underlying CTMC $\mathbf{X}^{(N)}(t)$ describing the time-evolution of the model.

The fluid approximation is achieved by first considering the normalised population counts obtained by dividing each variable by the total population N — $\hat{\mathbf{X}}^{(N)} = \frac{\mathbf{X}^{(N)}}{N}$. The initial conditions are scaled similarly — $\hat{\mathbf{X}}_0^{(N)} = \frac{\mathbf{X}_0^{(N)}}{N}$. The transitions are scaled as follows: for each $(R_\tau, r_\tau^{(N)}) \in \mathcal{T}^{(N)}$, let $\hat{\mathbf{r}}_\tau^{(N)}(\hat{\mathbf{X}})$ be the rate function expressed in terms of normalised variables. The corresponding transition in the normalised model is $(R_\tau, \hat{\mathbf{r}}_\tau^{(N)}(\hat{\mathbf{X}}))$ with update vector $\frac{1}{N}\mathbf{v}_\tau$. Suppose that for all transitions $\tau \in \mathcal{T}^{(N)}$ there exists a bounded and Lipschitz continuous

function $f_\tau : \mathbb{R}^K \to \mathbb{R}_{\geq 0}$ such that $\frac{1}{N}\hat{\mathbf{r}}_\tau^{(N)}(\hat{\mathbf{X}}) \to f_\tau(\hat{\mathbf{X}})$ uniformly as $N \longrightarrow \infty$. To define the limit ODEs, we introduce the drift $\mathbf{F}(\hat{\mathbf{X}}) = \sum_{\tau \in \hat{\mathcal{T}}^{(N)}} \mathbf{v}_\tau f_\tau(\hat{\mathbf{X}})$.

Theorem 1 (Deterministic approximation theorem [4]). *With $\hat{\mathbf{X}}^{(N)}(t)$ we assume there exists a point $\mathbf{x}_0$ such that $\hat{\mathbf{X}}^{(N)}(0) \to \mathbf{x}_0$ in probability. Let $\mathbf{x}(t)$ be a solution to $\frac{d\mathbf{x}}{dt} = \mathbf{F}(\mathbf{x})$ with $\mathbf{x}(0) = \mathbf{x}_0$. Then, for any finite time horizon $0 \leq t \leq T$, $\epsilon \in \mathbb{R}_{\geq 0}$ we have*

$$\mathbb{P}\left(\sup_{0 \leq t \leq T} \|\hat{\mathbf{X}}^N(t) - \mathbf{x}(t)\| \geq \epsilon\right) \xrightarrow{N \to \infty} 0$$

Policy optimisation Consider a population model $\mathcal{M}^N = (\mathbf{X}^{(N)}, \mathcal{T}, \mathcal{A}, \beta)$ derived from a CARMA-C model with N components. We can easily extend the normalisation of CTMC described previously to consider the corresponding normalised population CT-MDP denoted $\hat{\mathcal{M}}^N$. We deal with the following problem: *find a policy π in the space of stationary deterministic policies of $\hat{\mathcal{M}}^N$ that maximises some reward function over a finite time horizon.* In particular, consider the functional $Q^N : \mathit{\Pi} \to \mathbb{R}$, where $\mathit{\Pi}$ is the set of stationary deterministic policies. The optimisation problem is thus defined as maximising some defined functional Q^N, i.e., finding a policy π^* that satisfies

$$Q^N[\pi^*] = \sup_{\pi \in \mathit{\Pi}} Q^N[\pi]$$

Suppose we fix a policy π and consider the resulting normalised population CTMC model denoted $\hat{\mathcal{X}}_\pi^{(N)}$. As before, let $\hat{\mathbf{X}}_\pi^{(N)}(t)$ denote the stochastic process describing the time-evolution of the population CTMC. Let $V : \mathscr{D}_{\mathcal{S}} \to \mathbb{R}$ be a reward function on the space of trajectories of the stochastic process $\hat{\mathbf{X}}_\pi^{(N)}(t)$. Corresponding reward functional on the space of policies is then given by

$$Q^N[\pi] \stackrel{\text{def}}{=} V(\hat{\mathbf{X}}_\pi^{(N)}(t))$$

Example 5. Take a state reward function $r : \mathcal{S} \to \mathbb{R}_{\geq 0}$ mapping the states of the CT-MDP to positive real values. Define a value function corresponding to reward function r and policy π as the expected finite time-horizon ($0 \leq t \leq T$) cumulated reward:

$$Q^N[\pi] \stackrel{\text{def}}{=} V(\hat{\mathbf{X}}_\pi^{(N)}(t)) \stackrel{\text{def}}{=} \mathbb{E}\int_0^T r(\hat{\mathbf{X}}_\pi^{(N)}(t))dt$$

3.1 Policy synthesis via fluid approximation

Evaluating a given functional Q usually reduces to considering the transient evolution or steady state of $\hat{\mathcal{X}}_\pi^{(N)}$ which, especially for large population sizes N, is computationally expensive. One way to alleviate this problem is to consider Monte Carlo estimates of the functionals based on simulated trajectories of $\hat{\mathcal{X}}_\pi^{(N)}$,

e.g. , using the Gillespie algorithm. This is done in the context of statistical model checking [11] and has recently been applied to learning effective time-dependent policies for CT-MDPs [12]. Here, we argue that in the case of some reward functionals a good estimate can be achieved via fluid approximation. Indeed, suppose that $\hat{\mathbf{X}}_\pi^{(N)}$ converges to $\mathbf{x}_\pi$ in the sense of Theorem 1. Then as a simple consequence of the Portmanteau lemma [13] we can say that for any bounded and continuous reward function V we get

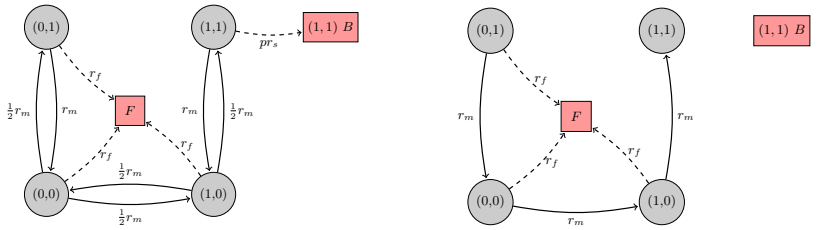
$$Q^N[\pi] = \mathbb{E}(V(\hat{\mathbf{X}}_\pi^N(t))) \xrightarrow{N\to\infty} \mathbb{E}(V(\mathbf{x}_\pi(t))) = q[\pi]$$

Example 6. For the running example consider the system of N robots and the global goal: *at least* 80% *of the robots reach the source location in time* T. We translate this goal into a value function by considering a logistic function $I(x) = 1/(1+e^{-2k(x-0.8)})$ which for large k approximates a step function. We define a reward function corresponding to the goal by $V(\hat{\mathbf{X}}_\pi^{(N)}(t)) = I(X_{\pi,s}^{(N)}(T))$ where $X_{\pi,s}^{(N)}(t)$ denotes the evolution of the population at the source location. Thus if $\mathbf{x}_\pi$ approximates $\hat{\mathbf{X}}_\pi^{(N)}$, in the sense of Theorem 1, then as I is both continuous and bounded then $\mathbb{E}(V(\mathbf{x}_\pi(t)))$ approximates $\mathbb{E}(V(\hat{\mathbf{X}}_\pi^{(N)}(t)))$.

3.2 Approximation for mode switching

In this section we present a method for approximating the behaviour of population systems exhibiting switching behaviour described in Section 2.1. Again the discussion here concentrates on systems with two such dynamic modes where mode changes are unidirectional but the idea can be extended to more modes. The method we propose is based on the observation that the behaviour of the system within a single dynamic mode can be given a fluid approximation as described in Section 3. Note that this has similarities to hybrid limit behaviour of Markov population processes considered in [14]. However, for mode switches arising form broadcast communication the rate of switching can depend on the population size which restricts the applicability of results provided in [14].

In detail, consider a normalised population model $\hat{\mathcal{M}}^N$ with two modes described by $(\mathbf{X}_1, \mathcal{T}, \mathcal{A}, \beta)$ and $(\mathbf{X}_2, \mathcal{T}, \mathcal{A}, \beta)$. Fix a policy π of $\hat{\mathcal{M}}^N$ and consider the resulting stochastic processes $\hat{\mathbf{X}}_{\pi,1}^{(N)}$ and $\hat{\mathbf{X}}_{\pi,2}^{(N)}$ corresponding to the two modes for which we can give a fluid approximation. Denote the resulting approximations by $\mathbf{x}_{\pi,1}$ and $\mathbf{x}_{\pi,2}$. The difficulty now is related to combining the two approximations into an approximation for the mean behaviour of the full process. We propose the following method: identity a variable and a threshold in the fluid approximation of the first mode which serves as an indicator for the mode switch – when the variable reaches the given threshold we expect the mode switch to take place; use this to estimate the switching time t^*; approximate the behaviour of the full process by $\mathbf{x}_\pi(t) = \mathbb{1}\{t \le t^*\}\mathbf{x}_{\pi,1} + \mathbb{1}\{t > t^*\}\mathbf{x}_{\pi,2}$. In the following section we evaluate the accuracy of such approximation for the running example and use it to obtain estimates for policy synthesis tasks.



(a) Before broadcast – random walk (b) After broadcast – directed walk

Fig. 3: Behaviour of individual robots.

4 Analysis: running example

In this section we make use of the presented ideas to analyse the running example of a foraging robot swarm. In particular, we consider the system consisting of N robots and restrict the robots to a 2×2 grid with paths $(0,0) \leftrightarrow (1,0)$, $(0,0) \leftrightarrow (0,1)$ and $(1,0) \leftrightarrow (1,1)$ to keep the constructions manageable by hand. All robots start by following the behaviour illustrated in Figure 3a where the location $(1,1)$ is designated as the source location. At location $(1,1)$, with a probability $p = 0.1$, the sense action results in a broadcast. A robot sending out such broadcast at location $(1,1)$ causes the rest of the collective to follow the behaviour given in Figure 3b giving rise to two dynamic modes for the system.

We construct an approximation for the system dynamics by considering variables x_{00}, x_{10}, x_{01}, x_{11} giving the proportion of robots in locations $(0,0)$, $(1,0)$, $(0,1)$ and $(1,1)$ respectively. Additionally, let s_{11} be the proportion of robots that have sensed the source resulting in a broadcast being sent out and let f denote the proportion of robots that have broken down. We construct $\mathbf{x}_1$ and $\mathbf{x}_2$ as below and claim that these give a fluid approximation for the system dynamics before and after the broadcast respectively.

$$
\begin{aligned}
\mathbf{x}_1(t) = \mathbf{x}_2(t) \; &= [x_{00} \quad x_{01} \quad x_{10} \quad x_{11} \quad s_{11} \quad f] \\
\frac{d\mathbf{x}_1}{dt} \; &= [-x_{00}(r_m + r_f) + r_m(x_{10} + \tfrac{1}{2}x_{01}) \quad \tfrac{1}{2}r_m x_{00} - x_{01}(r_m + r_f) \\
&\qquad \tfrac{1}{2}r_m x_{00} + r_m x_{11} - x_{10}(r_m + r_f) \quad \tfrac{1}{2}r_m x_{10} - x_{11}(p_s r_s - r_m) \\
&\qquad p r_s x_{11} \quad r_f(x_{00} + x_{10} + x_{01})] \\
\frac{d\mathbf{x}_2}{dt} \; &= [-x_{00}(r_m + r_f) + r_m x_{01} \quad -x_{01}(r_m + r_f) \quad r_m x_{00} - x_{10}(r_m + r_f) \\
&\qquad x_{01} r_m \quad 0 \quad r_f(x_{00} + x_{10} + x_{01})]
\end{aligned}
$$

Suppose all robots start from the location $(0,0)$, i.e. the initial condition for the approximation is $\mathbf{x}_1(0) = [\,1\ 0\ 0\ 0\ 0\ 0\,]$. To combine the two modes we use the estimate proposed in Section 3.2 by considering the time evolution of variable s_{11} and taking the expected time till the first broadcast to be the time t^* such that $s_{11}(t^*) = 1$. Thus, we approximate the mean behaviour of the system by

$$\hat{\mathbf{x}}(t) = \mathbb{1}\{t \leq t^*\}\mathbf{x}_1(t) + \mathbb{1}\{t > t^*\}\mathbf{x}_2(t)$$

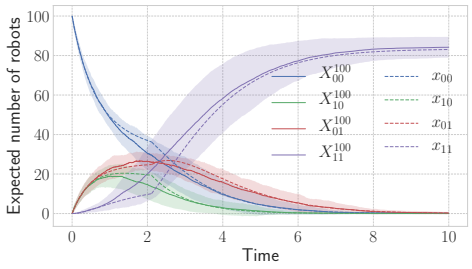


Fig. 4: Comparison of expected trajectories for failr $= 0.05 = r_f$ with rate of move and sense actions set to 1 (mean and variance of 100 simulated trajectories: solid lines and fluid approximations: dashed).

where $\mathbf{x}_2$ is such that $\mathbf{x}_2(t^*) = \mathbf{x}_1(t^*)$. The trajectories are compared with the stochastic simulation in Figure 4 and Table 1 for different parameters giving an empirical justification for the approximation. Figure 4 suggests that a good approximation is achieved for times away from the mode switching. For Table 1, we consider the mean of relative errors between the stochastic and approximate trajectories for location $(1, 1)$ at time points $t = 2.0, 4.0, 6.0, 8.0, 10.0$. For each parametrisation, the table gives a mean figure over 10 comparisons and shows that as expected the approximation is better for larger population sizes.

4.1 Policy synthesis

In the context of the running example we consider the synthesis of failr parameter as a special case of policy synthesis. In particular, how robust should the behaviour of the robots be for the collective to satisfy its goal. We provide a simple application for the presented fluid approximation ideas by studying the action (or parameter) space of the running example through logistic regression and via direct optimisation of a more complex reward functional.

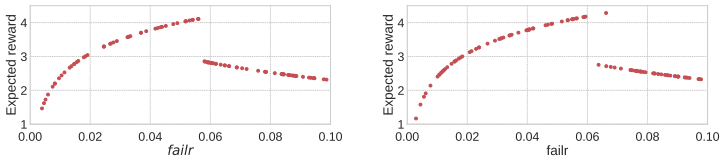
Logistic regression We consider the logistic reward function defined in Example 6 and note that the reward function is defined so that the specified goal (at least 80% of the collective reaches $(1, 1)$) is satisfied for rewards greater or equal to 0.5. As a first example we consider the following question: *what is the region of* failr *values for which we are expecting the policy to be satisfied.* We treat failr as an indication of robustness of individual components and classify the different possible values based on goal satisfaction. Throughout the rest of this section we set $r_m = r_s = 1$.

Table 1: Mean approximation error.

$(r_m, r_s,$ failr$)$	pop. size	mean error
$(1, 1, 0.05)$	100	10.8%
$(0.8, 1, 0.01)$	100	11%
$(2.0, 1, 0.1)$	100	4.3%
$(1, 1, 0.05)$	500	1.6%
$(1, 1, 0.02)$	500	2.0%
$(2.0, 1, 0.1)$	100	0.6%

Table 2: Logistic regression. Fitting based on 100 trajetories.

	decision boundaries
fluid	$0.0566, 0.0553$
	$0.0579, 0.0560$
stochastic	$0.0594, 0.0624$
	$0.0616, 0.0610$



(a) Fluid approximation. (b) Stochastic simulation.

Fig. 5: Sampled reward functional.

The set-up for this is standard: consider a linear function $y = w_0 + w_1 r$ of single explanatory variable (in this case value of failr, denoted r) and a logistic function $\sigma(r) = 1/(1 + e^{-w_0 - w_1 r})$ where $\sigma(r)$ is interpreted as the probability of success given failr value r. We are going to expect the goal to be satisfied if $\sigma(r) > 0.5$. The weights for the regression model are going to be fitted based on trajectories sampled using stochastic simulation and the constructed approximation for 100 random failr values. Table 2 gives the comparison of decision boundaries obtained by the two methods. Results suggest that for the considered parameters we slightly under-approximate the proportion of robots at $t = 10.0$.

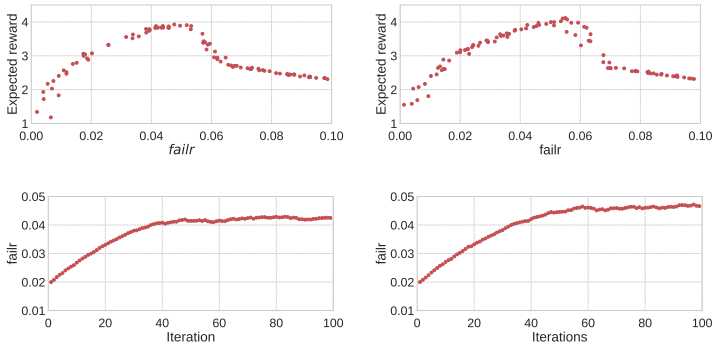
Direct optimisation of a reward functional Alternatively, we can consider direct optimisation of a reward functional. For example, consider the following functional to find maximal value for failr that results in the goal being satisfied

$$Q^N[\pi] = \begin{cases} I(X^{(N)}_{\pi,s}(T)) + \log(\pi) + c_0 & \text{for } I(X^{(N)}_{\pi,s}(T)) \geq 0.5 \\ I(X^{(N)}_{\pi,s}(T)) - \log(\pi) & \text{otherwise} \end{cases}$$

where c_0 is some chosen constant and $\pi \in [0, \infty)$ corresponds to the chosen policy. The first part of the reward functional corresponds to the satisfaction of the goal as defined previously. The constraint of failr being non-negative is taken into account by adding a logarithmic barrier function. The term $-\log(\pi)$ is used to penalise failr values that are further away from satisfying the goal. Figures 5a and 5b show the reward functional sampled uniformly from $\pi \in [0.0, 0.1]$. Note that due to stochastic variance we are going to have values of π for which we are uncertain about whether the goal is going to be satisfied or not.

To optimise for this, currently discontinuous, reward functional we consider the method of policy gradient presented, for example, in [15] with parametrised policies $\pi \sim \mathcal{N}(\mu, \sigma^2)$ — in a control scenario this would correspond to looking at a stochastic controller. Figure 6 shows the expected reward functional for $\pi \sim \mathcal{N}(\mu, 0.05)$ with 100 samples of μ taken uniformly from $(0.0, 0.1]$.

Considering such Gaussian policies together with the approximation to mean behaviour allows us to implement a fast policy gradient algorithm for synthesising approximations to optimal policies for the system. In particular, the gradient of the functional at π is going to be estimated based on the ideas in Section 3 by $\nabla Q^N[\pi] \sim (q[\pi + \epsilon] - q[\pi])/\epsilon$ where q is the functional corresponding to the expected reward of the fluid approximation. The evolution of policy parameter values for a simple gradient ascent algorithm is given in Figure 6.



(a) Fluid approximation. (b) Stochastic simulation.

Fig. 6: Sampled reward functional for Gaussian policies and the gradient ascent initialised at $\pi \sim \mathcal{N}(0.02, 0.005)$. Reward estimates based on 10 samples of π.

Performance Multiple stochastic trajectories are needed to get a good estimate of the mean behaviour — for the model of 100 robots we used 100 trajectories, which took about 3 sec of computing on a single thread. The approximation took about 0.04 seconds translating into a non-trivial speed-up in cases where trajectories for lots of parametrisations have to be considered. In particular, when generating 100 samples for considered reward functional this translates into roughly 5 minutes via stochastic simulation and 4 seconds via the approximation.

5 Conclusion

In this paper we presented a framework for exploiting fluid approximation results in the context of policy synthesis for collective dynamics involving broadcast communication. To that end we proposed a class of population CT-MDPs arising from systems with broadcast communication where the communication can be thought to separate the dynamics of the system into modes. To aid the construction of such models we introduced a language CARMA-C, based on CARMA, and outlined the semantics which gives a natural way for specifying policy synthesis problems in a high-level language. We discussed the application of fluid approximation in policy synthesis and suggested an approximation for the population models with mode switching for which the classic results cannot be applied directly. We used the proposed approximation to analyse the running example of a robot swarm.

For further work we plan to give a more formal treatment for the approximations for mode switching. In particular, the current method gives a good approximation to mean behaviour for times sufficiently far from where the mode change is expected to happen. However, this presents a limitation when we are interested in the behaviour of the system around the time of mode change or in the case of multiple modes where two mode changes can happen close to each other. To address this we aim to devise a more sophisticated approximation for switching time. For that we plan to consider the linear noise approximation [16],

as done for example in [17], to help recover information about stochasticity and estimate the distribution of switching times.

Acknowledgement. This work was supported by EPSRC grant EP/L01503X/1 (CDT in Pervasive Parallelism).

References

1. Buchholz, P., Dohndorf, I., Scheftelowitsch, D.: Optimal decisions for continuous time markov decision processes over finite planning horizons. Computers & OR **77** (2017) 267–278
2. Butkova, Y., Hatefi, H., Hermanns, H., Krcál, J.: Optimal continuous time markov decisions. In: Automated Technology for Verification and Analysis, 2015, Proceedings. (2015) 166–182
3. Gast, N., Gaujal, B.: A mean field approach for optimization in discrete time. Discrete Event Dynamic Systems **21**(1) (2011) 63–101
4. Kurtz, T.G.: Solutions of ordinary differential equations as limits of pure jump markov processes. Journal of Applied Probability **7**(1) (1970) 49–58
5. Loreti, M., Hillston, J.: Modelling and analysis of collective adaptive systems with CARMA and its tools. In: Formal Methods for the Quantitative Evaluation of Collective Adaptive Systems, Advanced Lectures. (2016) 83–119
6. Piho, P., Georgoulas, A., Hillston, J.: Goals and resource constraints in carma. In: Proceedings of the Ninth International Workshop on the Practical Application of Stochastic Modelling (PASM). (2018) 155 – 172
7. Jonsson, B., Larsen, K.G.: Specification and refinement of probabilistic processes. In: Proceedings of the Sixth Annual Symposium on Logic in Computer Science (LICS '91). (1991) 266–277
8. Caillaud, B., Delahaye, B., Larsen, K.G., Legay, A., Pedersen, M.L., Wasowski, A.: Constraint markov chains. Theor. Comput. Sci. **412**(34) (2011) 4373–4404
9. Georgoulas, A., Hillston, J., Milios, D., Sanguinetti, G.: Probabilistic programming process algebra. In: Quantitative Evaluation of Systems - 11th International Conference. (2014) 249–264
10. De Nicola, R., Latella, D., Loreti, M., Massink, M.: A uniform definition of stochastic process calculi. ACM Comput. Surv. **46**(1) (2013) 5:1–5:35
11. Legay, A., Delahaye, B., Bensalem, S.: Statistical model checking: An overview. In: Runtime Verification - First International Conference, Malta. (2010) 122–135
12. Bartocci, E., Bortolussi, L., Brázdil, T., Milios, D., Sanguinetti, G.: Policy learning in continuous-time markov decision processes using gaussian processes. Perform. Eval. **116** (2017) 84–100
13. Billingsley, P.: Convergence of probability measures. Second edn. John Wiley & Sons Inc., New York (1999)
14. Bortolussi, L.: Hybrid behaviour of markov population models. Inf. Comput. **247** (2016) 37–86
15. Peters, J., Schaal, S.: Reinforcement learning of motor skills with policy gradients. Neural Networks **21**(4) (2008) 682–697
16. Van Kampen, N.: Stochastic Processes in Physics and Chemistry. North-Holland Personal Library. Elsevier Science (2011)
17. Bortolussi, L., Lanciani, R.: Model checking markov population models by central limit approximation. In: Quantitative Evaluation of Systems, Proceedings. (2013) 123–138