Symmetry Reduction Enables Model Checking of More Complex Emergent Behaviours of Swarm Navigation Algorithms

Conference Paper · May 2015

DOI: 10.1007/978-3-319-22416-9_4 · Source: arXiv

CITATIONS
12

READS
115

4 authors, including:

Dejanira Araiza-Illan
Agency for Science, Technology and Research (A*STAR)
28 PUBLICATIONS **323** CITATIONS

SEE PROFILE

Sergio Campos
Federal University of Minas Gerais
123 PUBLICATIONS **2,965** CITATIONS

SEE PROFILE

Kerstin Eder
University of Bristol
162 PUBLICATIONS **1,810** CITATIONS

SEE PROFILE

Symmetry Reduction Enables Model Checking More Complex Emergent Behaviours of Swarm Navigation Algorithms

Laura Antuña[*], Dejanira Araiza-Illan[†], Sérgio Campos[‡], and Kerstin Eder[§]

Abstract

The emergent global behaviours of robotic swarms are important to achieve their navigation task goals. These emergent behaviours can be verified to assess their correctness, through techniques like model checking. Model checking exhaustively explores all possible behaviours, based on a discrete model of the system, such as a swarm in a grid. A common problem in model checking is the state-space explosion that arises when the states of the model are numerous. We propose a novel implementation of symmetry reduction, in the form of encoding navigation algorithms relatively with respect to a reference, based on the symmetrical properties of swarms in grids. We applied the relative encoding to a swarm navigation algorithm, Alpha, modelled for the NuSMV model checker. A comparison of the state-space and verification results with an absolute (or global) and a relative encoding of the Alpha algorithm highlights the advantages of our approach, allowing model checking larger grid sizes and number of robots, and consequently, verifying more complex emergent behaviours. For example, a property was verified for a grid with 3 robots and a maximum allowed size of 8×8 cells in a global encoding, whereas this size was increased to 16×16 using a relative encoding. Also, the time to verify a property for a swarm of 3 robots in a 6×6 grid was reduced from almost 10 hours to only 7 minutes. Our approach is transferable to other swarm navigation algorithms.

1 Introduction

Robotic swarms consist of a set of robots with simple individual behaviour rules, working together in cooperation to achieve a more complex or emer-

[*]Department of Computer Science, Universidade Federal de Minas Gerais, Brazil, laura.antuna@dcc.ufmg.br

[†]Department of Computer Science, University of Bristol, UK, dejanira.araizaillan@bristol.ac.uk

[‡]Department of Computer Science, Universidade Federal de Minas Gerais, Brazil, scampos@dcc.ufmg.br

[§]Department of Computer Science, University of Bristol, UK, kerstin.eder@bristol.ac.uk

gent final behaviour. Appealing characteristics of swarms are the low cost incurred in producing the robots, which have a simple hardware design, scalability, and fault tolerance [7]. Examples of their application to real-life tasks include nanorobotics, disaster rescue missions, and mining or agricultural foraging tasks.

The emergent behaviours of a swarm of robots need to be *verified*, with respect to safety and liveness requirements [18, 20], and *validated* to determine whether it is fit for purpose in the target environment. Safety requirements are the allowed behaviours of the system, and liveness requirements specify the dynamic behaviours expected to happen during the execution of the system [20]. Verification methods include testing (over the real robotic platforms or in simulation), and formal, model checking and theorem proving. Formal verification techniques exhaustively explore all the possible behaviours of a system, to determine if the requirements are satisfied or the existence of errors. Formal verification methods, more specifically model checking, have been employed to perform the verification tasks, as in [19, 6, 13, 7, 8, 15]. In model checking, the system is modelled in a finite-state manner.

The continuous space in which the robots in the swarm move represents a challenge, since it can cause an infinite state model, which translates into a state-space explosion problem in model checking –i.e., the number of states to explore is out of pragmatic computational capabilities. The discretization of the continuous space into cells of fixed size –i.e., a grid– is a solution that has been applied in swarms, in the context of model checking [14, 8, 19]. Even with the discretization of the environment into a "small" grid (e.g., 4×4 cells), the state-space explosion problem can occur due to the presence of more variables, which lead to too many possible configurations of the robots in the grid.

Symmetry reduction techniques have been used to reduce the size of the models in model checking, mostly as an automatic process, and static –i.e., the reduction is performed before model checking– or dynamic –i.e., the reduction is performed whilst model checking, "on-the-fly" [17, 5, 2, 10, 11, 9, 1, 12]. These techniques compute a subset of representatives of all the states, after the user provides the classification criteria for the grouping, or the classification is computed by analysing similarities amongst the states. Our proposed solution to the state-space explosion problem for swarms in a grid is to exploit the symmetry of the configurations of the robots in the grid, implementing symmetry reduction in a novel manner compared to previous approaches for model checking.

In this paper, we explore the vertical and horizontal symmetry in the grid to reduce the size of the finite-state model. We implemented a *relative encoding* of the individual robots' navigation algorithm of a swarm in a grid that eliminates symmetrical equivalent states from the state space. The swarm is assumed homogeneous; i.e., all the robots are considered identical in capabilities and rank. In the relative encoding, a robot is set as the "reference", with a fixed location and direction of motion. The other robots' locations and directions are defined based on the reference. In a global or absolute encoding, if all the robots in the grid are simultaneously rotated in the same direction and shifted horizontally or vertically the same distance, the robots' new configurations change in location and direction. In a relative encoding, the locations and directions would remain

the same since they are encoded relative to the reference robot, resulting in a reduction of the state space of direction and position configurations. We applied our approach to the *Alpha* swarm algorithm [16], modelled in the NuSMV model checker [3]. Firstly, we modified the models in [7, 8] to be relative. We compared the reduction in the state space of the two encodings. Secondly, we modelled the Alpha algorithm in our own terms, employing more variables to capture its meaning more closely. This new model, despite having more variables, was of a reduced order of states compared to the global encoding in [7, 8]. This allowed us to check for larger number of robots and grid size. Furthermore, we obtained different results for the properties verified in [7, 8]. Applying the relative encoding concept to other swarm navigation algorithms in the same manner is possible, and we will be exploring this in the future.

The structure of this paper is as follows. Section 2 introduces grid-based discrete models for swarm robotics, based in [8]. Section 3 presents an overview of model checking, the state-space explosion problem and related symmetry reduction techniques. Section 4 presents the relative encoding model. Section 5 introduces the Alpha algorithm and a global encoding of it in [7, 8]. In Section 6, we present and discuss the comparison results of the relative and global encodings of the Alpha algorithm. The concluding remarks are presented in Section 7.

2 Swarms in Grids

When modelling navigating algorithms for swarms, a critical aspect is the continuous space in which the robots in the swarm act. A common approach is to discretize this environment into squared cells of the same size, forming a grid. This grid can also "wrap around", i.e., work as a sphere.

Another aspect in the modelling of a swarm is the concurrency of its elements. Four main types have been proposed [8]: *synchrony* where all robots move at the same time in each step; *strict turn taking* where only one robot moves at a time, following a strict order; *non-strict turn taking* where only one robot moves at a time in a random order, but all the robots get the chance to move after a number of steps; and *fair asynchrony*, where robots move at different time in a random order, and the only guarantee is that a robot will always eventually move in the future.

3 Model Checking and the State-space Explosion Problem

Model checking is a formal verification method. An exhaustive traversal of the reachable states of a discrete model of the system is performed to check the veracity of some desired properties, such as liveness and safety. The reachability depends on the allowed transitions from state to state. The properties to be verified are defined using a temporal logic, for example Linear Temporal Logic

(LTL), or Computational Tree Logic (CTL). Model checking is fully automatic, and counterexamples can be produced in the case of a property being false, which helps discovering the reason of the failure [4].

Model checkers can be either explicit or symbolic, the latter having an internal structure such as a Binary Decision Diagram (BDD) or Boolean functions that implicitly represent the transitions within the states of the system. The BDDs are constructed before the traversal of the model. Explicit-state model checkers traverse the model whilst verifying it, which may lead to running out of memory before finishing the traversal. Symbolic model checkers allow an initial compression of the model, at the cost of an overhead and memory usage before the checking. NuSMV is a popular open-source symbolic model checker, with its own input language [3].

The number of states and transitions to traverse in the model can cause problems for model checking (state-space explosion). Different techniques have been incorporated into model checking to alleviate this issue, such as the use of BDDs that led to the branch of symbolic model checking, symmetry reduction techniques, and abstractions of the model to reduce the number of states at the cost of meaningfulness and detail [4].

The symmetrical properties of a finite-state model [4] can be identified and the state space of the model reduced before model checking takes place. For example, a "static channel diagram" of a Promela model is computed in [9], to be used in explicit-state model checking. In [12], symmetrical components are reduced by hand, by annotating the model with the directive `TRANS` to eliminate equivalent transitions from state to state, in the NuSMV model checker. This manual approach is not trivially transferable to reducing the model of a swarm in a grid.

Symmetry reduction can be applied to BDDs, resulting from the finite-state structures [4]. "Quotient models" are proposed in [5, 11]. Automorphisms that preserve the same transition relations (or "orbit relations") in a BDD are computed, from permutations of the states, and a chosen representative state substitutes all the states in each orbit relation, forming a quotient model instead of the original BDD. When using representatives, instead of the full set of states, the number of explored states and transitions are reduced.

Symmetry reduction techniques have also been applied to explicit-state model checkers. In [17], a new data type, "scalarsets", is added to the input language of a model checker, to create automorphisms and a quotient graph –i.e., using representatives of groups of states– whilst traversing the model. Scalarsets are similar to the static channel diagram model in [9]. The automorphisms can be computed on-the-fly along with the traversal of the model, as in [2] based on heuristics, instead of given a priori. The main disadvantage of these explicit-state symmetry reduction implementations and of symmetry reduction over BDDs, is that they are applied into the algorithms of the model checker software or BDD computation, which becomes a non-trivial re-implementation task.

Our approach to avoid the state-space explosion problem in model checking is based on exploiting symmetrical properties of a swarm in a grid. The encoding of

the model in a relative manner with respect to a reference point, as opposed to a global or absolute encoding, can be interpreted as the combination of symmetry reduction and abstraction. The proposed approach is analogous to finding orbit relations or representatives in the global model, and then eliminating them through the relative encoding. The relative encoding employs representatives of the global encoding as possible states –i.e., configurations of the robots in the grid.

4 Relative Encoding of a Swarm in a Grid

In swarms, the focal point is the interaction of the elements through time. Other global information is not so relevant, such as the location of each robot respect to an absolute frame. With only the swarm's overall behaviour in mind, the swarm moving north or south, with the same distance and orientation between all the robots, represent essentially the same. Furthermore, these two behaviours correspond to the same swarm configuration if the swarm is encoded in a relative manner. A simple relative encoding is to set a robot as the reference, with fixed location and direction, and the other robots's actions based on the reference. A grid populated with robots, rotated and shifted horizontally and vertically, results in different values for the direction and position of each robot in a global or absolute encoding. However, in a relative encoding some rotations and shifting motions correspond to the same resulting grid configurations. Figure 1 shows an example of two configurations of robots in a grid that are equivalent in terms of the distance values, with robot 1 as the reference.

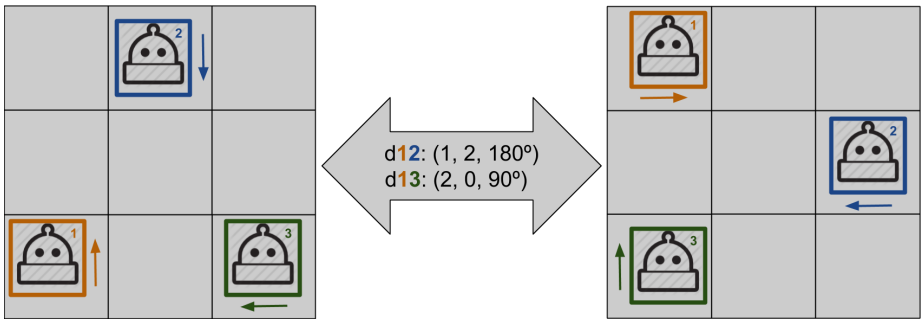

Figure 1: Equivalence of distances in two different grid configurations

If a model with r robots in a $m \times m$ size grid (locations), with d possible directions and p other variables of domain size $v_i, i = 1, ..., p$ is globally encoded, the order of the state space to be explored is of $(d \times m^2 \times v_1 \times v_2 \times ... \times v_p)^r$. In a relative encoding, the reference robot will have fixed location and direction, and the resulting state space will be of the order of $(v_1 \times v_2 \times ... \times v_p) \times (d \times m^2 \times v_1 \times v_2 \times ... \times v_p)^{r-1}$. This corresponds to a reduction of the state space of $d \times m^2$.

This decrease in the state space would improve the performance of model checking in terms of time and memory usage, when verifying emergent behaviours of the swarm. Moreover, a counterexample in the relative model is equivalent to a class in the global model.

The update of the direction and location of the robots based in the reference robot's location motion must be performed in two situations: when the reference robot makes a move to an adjacent cell, and when it combines the motion with a change in its direction (rotation). In the first case, the equivalent of the reference robot moving in a direction is the other robots moving in the opposite direction (e.g., if the reference moves north, the other robots move south instead, to keep the reference robot fixed in the grid). By following this update rule, the distance and direction relation between the swarm of robots remains the same. When the reference robot moves to another cell and also rotates, the other robots make a turn to an opposing direction and their locations need to be updated, as summarized in Table 1.

Table 1: Location and orientation update after the reference robot changed direction

Reference's change	Direction change	Location change
$n \to e$	$n \to w$, $s \to e$, $e \to n$, $w \to s$	$x' = m - y$, $y' = x$
$n \to s$	$n \to s$, $s \to n$, $e \to w$, $w \to e$	$x' = m - x$, $y' = m - y$
$n \to w$	$n \to e$, $s \to w$, $e \to s$, $w \to n$	$x' = y$, $y' = m - x$

The relative encoding in NuSMV input language has been designed to have the following structure, to facilitate modularity, employing `MODULE` constructions: *(a)* we used distances between the reference robot and other robots, instead of specific locations; *(b)* we defined and encoded the update of the distance variables in the `main` module, along with the concurrency mode logic (e.g., a `turn` variable); and *(c)* we defined and encoded the motion algorithm (next motion, step size, rotations) of each robot in `robot` modules, which update the values of the variables in the `main` module. This procedure can be partially automated through a script to create the `main` module, from selecting the number of robots and concurrency mode, which we implemented to generate the NuSMV code for the experiments in Section 6. The `robot` modules are encoded manually, as they depend on the navigation algorithm.

5 Alpha Algorithm

The Alpha algorithm has been used before as a case study of the potential of verifying swarm emergent behaviours through model checking tools, as in [6, 7, 8]. In the Alpha algorithm, the robots in the swarm navigate the environment

trying to maintain connectivity, defined as a wireless range. This is achieved by the following rules:

- The default movement of a robot is forward, maintaining its current direction.
- When a robot loses connection with another robot, if the remaining number of connected robots is smaller than a value α, the robot makes a $180\,^\circ$ turn.
- Every time a robot regains connectivity with another, it performs a random turn.

A requirement of interest for the swarm is that *all the robots shall be eventually connected*. Expressed formally in LTL, this property was proved to be not satisfied in [7, 8].

6 Results

We compared a global encoding of an abstraction of the Alpha algorithm proposed in [7, 8], against a relative encoding of it, both implemented in the input language of NuSMV. Also, we implemented a new version of the Alpha algorithm that employs more variables for a more accurate representation, encoded in a relative manner. We measured the state-space reduction of the relative models, compared to the global model. Table 2 shows the reduction in the reachable states for the Alpha algorithm, with grid size 8×8, three robots, and $\alpha = 1$. The new abstraction has fewer states than the global one. We decided not to consider the fully synchronous concurrency mode in the results, since, combined with the Alpha algorithm, it allows for behaviours that are incorrect in the real world: the robots are allowed to "swap" their cell locations.

Table 2: State space size comparison for global and relative encodings (in millions)

Concurrency	Statistic	Abstraction from [7, 8]		New abstraction
	(in millions)	Global Relative	Relative	
Fair asynchrony	Total States	134.2	1.1	31.9
	Reachable States	68.1	0.5	1.6
Strict turn taking	Total States	402.7	3.1	31.9
	Reachable States	1.1	0.2	0.4
Non-strict turn taking	Total States	2818.5	22.0	223.0
	Reachable States	48.4	0.7	1.3

Figure 2 and Figure 3 show a comparison of the state-space for global and relative encodings from the abstraction [7, 8], and the new abstraction when varying the grid size or the number of robots, when compiling the models in NuSMV. These experiments consider a strict turn taking concurrency mode. We repeated these experiments for other concurrency modes, non-strict turn

taking and fair asynchrony, with similar results in the state-space reduction. In the experiments of Figure 2, the number of robots is set to three, and the size of the grid is varied. In the experiments of Figure 3, the size of the grid is set to 6×6, and the number of robots is varied. The final points in the graph correspond to the limit in the memory for the verification to be possible.

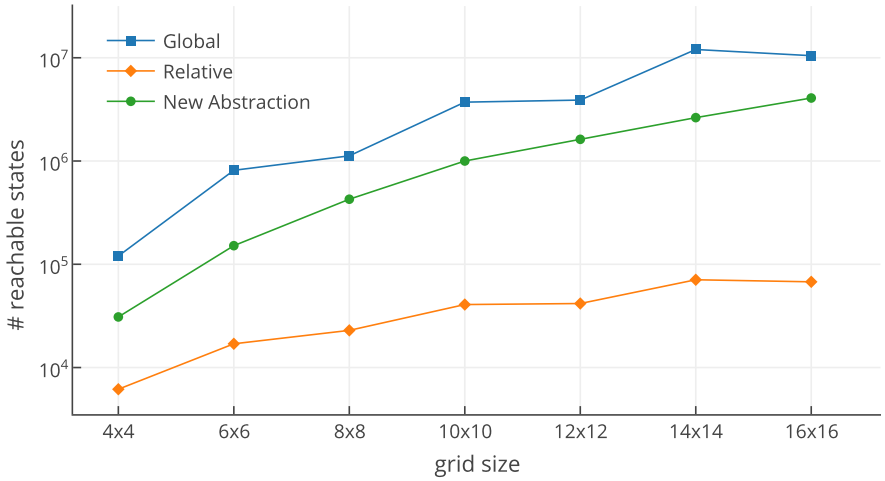

Figure 2: Reachable states as the grid size increases, for strict turn taking concurrency mode and three robots

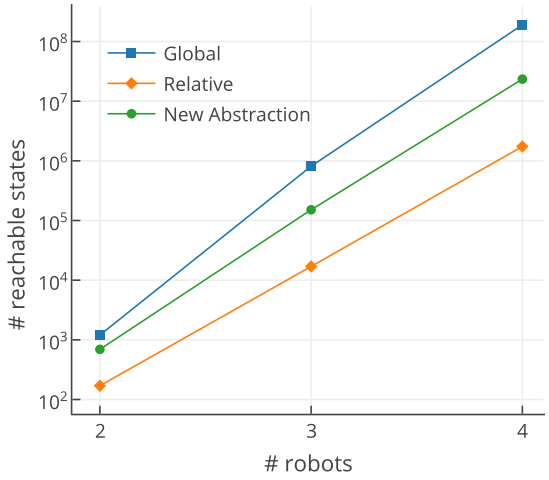

Figure 3: Reachable states as number of robots increases, for strict turn taking concurrency mode and a 6×6 grid

We verified the LTL property mentioned previously, *all the robots will be*

eventually connected, in all these models, to validate the relative encoding of the abstraction in [7, 8], with respect to their original global model. The verification results were identical, indicating our relative encoding preserved the properties of the global encoding. For the verification, we used NuSMV version 2.5.4, running in a PC with Ubuntu 14.10 with 4GB RAM.

A counterexample (failing trace) provided by the model checker, with strict turn taking concurrency mode, three robots, and a grid of size 5×5, is shown in Figure 4, for the global encoding. From state 5 onwards, all robots move south in a loop and robot C never reconnects to the swarm. In the global model, it takes 15 states until the loops starts. However, the same pattern is observed within each 3 steps from the moment when robot C changes its direction. This repetitiveness was eliminated in the relative model, as illustrated by Figure 5, achieving a reduction of 12 states. In the relative encoding, the location of the reference (the circle) is fixed to cell $(0, 0)$ and its direction to North. Subsequently, other robots update their position and orientation according to the decisions of the reference robot, or according to their individual decisions, as explained in Section 4.

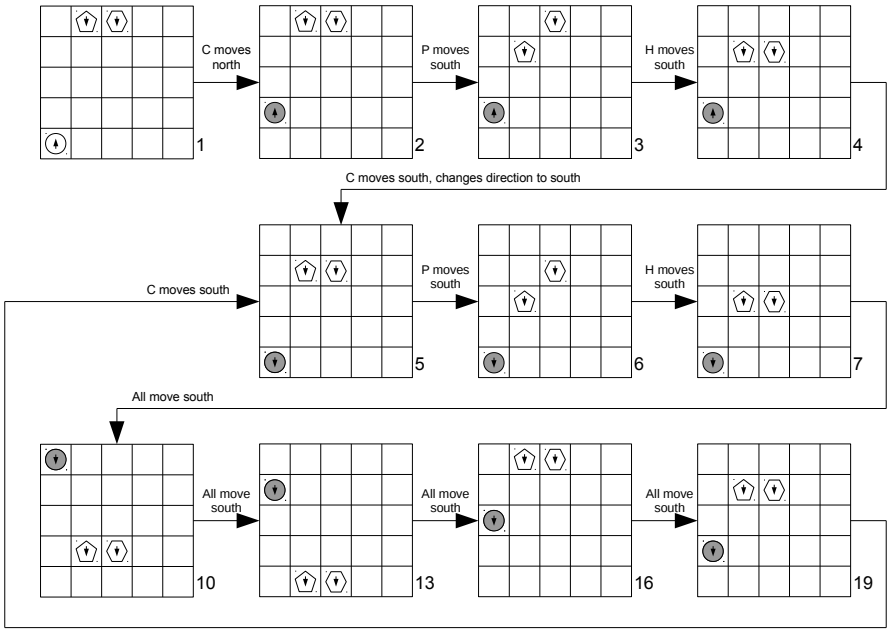

Figure 4: Failing trace of globally encoded model. C: circle, P: pentagon, H: hexagon. Disconnected robots in gray

Posteriorly, we compared the verification results using different abstractions of the Alpha algorithm, the one proposed in [7, 8] and ours, both encoded in

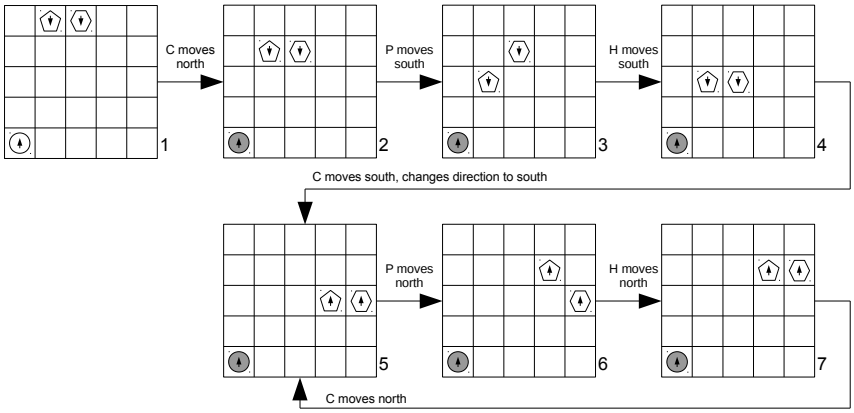

Figure 5: Failing trace of relatively encoded model. C: circle (reference), P: pentagon, H: hexagon. Reference: circle. Disconnected robots in gray

a relative manner. These experiments were conducted for $\alpha = 1$, and a strict turn taking concurrency mode. We found that the LTL property is true for some settings, such as three robots in grids of 2×2, 3×3 and 4×4, and two robots in a 5×5 grid. For other settings, such as three robots in a 5×5 grid, the property is false and the swarm will not regain connection. This is caused by the randomness of the individual robots decisions, that can be repeated infinitely –i.e., a robot can infinitely perform the same patterns of motion that do not lead to regain connectivity.

Figure 6 and Figure 7 show the verification time for the LTL property in Section 5 for global and relative encodings from the abstraction [7, 8], and the new abstraction when varying the grid size or the number of robots. These experiments, as before, consider a strict turn taking concurrency mode. In the experiments of Figure 6, the number of robots is set to three, and the size of the grid is varied. In the experiments of Figure 7, the size of the grid is set to 6×6, and the number of robots is varied. The final points in the graph correspond to a stipulated time limit of 5 days for the verification.

We observed the same time reduction patterns between the global and relative encodings of the abstraction in [7, 8], as a consequence of the state-space reduction. The number of robots is a more significant constraint to the verification time than the grid size. Although it was not possible to verify the relative encoding with 4 robots due to time restrictions, applying some constraints to the initial configuration of the swarm allowed the generation of a counterexample, a proof that the property is false for that number of robots and grid size. In contrast, the global encoding could not be verified, even when applying the same constraints. The relative encoding of the new abstraction of the Alpha algorithm takes longer to be verified than the previous abstraction, as it is more complex. Nevertheless, it allowed us to obtain different verification

results compared to [7, 8], which we believe are closer to the intention of the Alpha algorithm. The difference between the two abstractions of the Alpha algorithm need to be further investigated, to determine if any is incorrect, given the verification results.

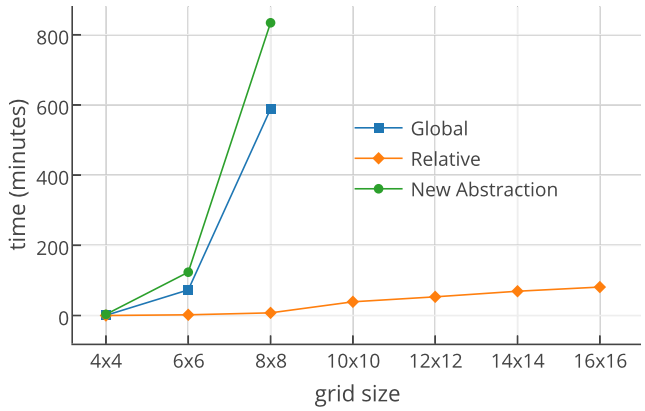

Figure 6: Verification time for the LTL property in Section 5 as the grid size increases, for strict turn taking concurrency mode and three robots

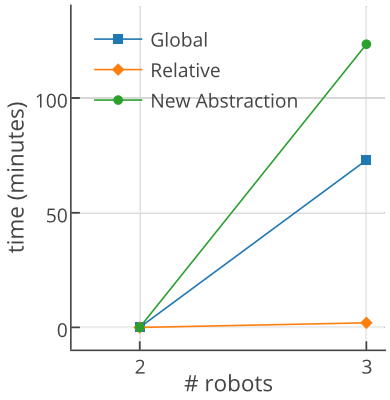

Figure 7: Verification time for the LTL property in Section 5 as number of robots increases, for strict turn taking concurrency mode and a 6×6 grid

Encoding the Alpha swarm algorithm in a relative manner allowed us to verify a LTL property for larger grid sizes, a larger number of robots, and in less time, than when using a global encoding. Throughout the experiments in this section, we demonstrated that the relative encoding is property preserving with respect to a corresponding global encoding. Furthermore, reducing the state-space through a relative model, we can implement more detailed abstractions that involve more variables, although there will always be a state-space

limitation inherent to model checking, due to our computational resources and time. Being able to implement a different abstraction, and encoding it in a relative manner, hints on the transferability of our approach to other navigation algorithms.

7 Conclusions

We presented an approach that allows model checking more complex emergent behaviours of swarms, as a consequence of the state-space reduction. We propose the use of symmetry reduction to eliminate symmetrical states (i.e., configurations of robots in the grid), implemented through a relative encoding of the swarm, where one robot is the reference, and the others' navigation is encoded relative to it. This encoding, compared to a global one, helps to reduce the state space of the model, as demonstrated by our results in Section 6. Thus, verification through model checking can be performed over larger grid sizes and number of robots, and abstractions for modelling the navigation algorithms that involve more variables.

Future work includes running more experiments with different α values, and the verification of more complex properties of robotic swarms, such as the emergence of teams or different swarms due to the connectivity properties. Also, we want to model other navigation algorithms such as the Beta [16] in a relative manner, to validate the transferability of our approach.

Aknowledgements We would like to thank Clare Dixon for providing her Alpha algorithm models, and her invaluable comments and advice. The work by D. Araiza-Illan and K. Eder was partially supported by the Engineering and Physical Sciences Research Council (EPSRC), grants EP/J01205X/1 RIVERAS: Robust Integrated Verification of Autonomous Systems and EP/K006320/1 Trustworthy Robotic Assistants.

References

[1] Christian Appold. Efficient symmetry reduction and the use of state symmetries for symbolic model checking. In *Proc. GandALF*, pages 173–187, 2010.

[2] Dragan Bosnacki, Dennis Dams, and Leszek Holenderski. Symmetric SPIN. In *SPIN Model Checking and Software Verification*, pages 1–19, Stanford,CA,USA, 2000.

[3] A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella. NuSMV version 2: an open-source tool for symbolic model checking. In *Computer-Aided Verification*, pages 359–364, Copenhagen, Denmark, 2002.

[4] Edmund M. Clarke, Jr., Orna Grumberg, and Doron A. Peled. *Model Checking*. MIT Press, Cambridge, MA, USA, 1999.

[5] E.M. Clarke, R. Enders, T. Filkorn, and S. Jha. Exploiting symmetry in temporal logic model checking. *Formal Methods in System Design*, 9(1–2):77–104, 1996.

[6] Clare Dixon, Alan F.T. Winfield, and Michael Fisher. Verification of swarm robots: the Alpha algorithm. In *Proc. ARW*, pages 10–11, Westminster, UK, 2010.

[7] Clare Dixon, Alan F.T. Winfield, and Michael Fisher. Towards temporal verification of emergent behaviours in swarm robotic systems. In *Towards Autonomous Robotic Systems*, pages 336–347, Sheffield, UK, 2011.

[8] Clare Dixon, Alan F.T. Winfield, Michael Fisher, and Chengxiu Zeng. Towards temporal verification of swarm robotic systems. In *Robotics and Autonomous Systems*, pages 1429–1441, Bristol, UK, 2012.

[9] A. F. Donaldson and A. Miller. Automatic symmetry detection for model checking using computational group theory. In *Formal Methods*, pages 481–496, Newcastle, UK, 2005.

[10] E.Allen Emerson and Thomas Wahl. On combining symmetry reduction and symbolic representation for efficient model checking. In *Correct Hardware Design and Verification Methods*, pages 216–230, LAquila, Italy, 2003.

[11] E.Allen Emerson and Thomas Wahl. Dynamic symmetry reduction. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 382–396, Edinburgh, UK, 2005.

[12] Pedro de Carvalho Gomes. Verification of symmetric models using semiautomatic abstractions. Master's thesis, Computer Science, Belo Horizonte, Brazil, 2010.

[13] Silver Juurik and Jüri Vain. Model checking of emergent behaviour properties of robot swarms. *Proceedings of the Estonian Academy of Sciences*, 60(1):48–54, 2011.

[14] Marius Kloetzer and Calin Belta. Temporal logic planning and control of robotic swarms by hierachical abstractions. *IEEE Transactions on Robotics*, 23(2):320–330, 2007.

[15] Savas Konur, Clare Dixon, and Michael Fisher. Analysing robot swarm behaviour via probabilistic model checking. *Robotics and Autonomous Systems*, 60:199–213, 2012.

[16] J. Nembrini. *Minimalist Coherent Swarming of Wireless Networked Autonomous Mobile Robots*. PhD thesis, Bristol, UK, 2005.

[17] C. Norris IP and David L Dill. Better verification through symmetry. *Formal Methods in System Design*, 9(1-2):41–75, 1996.

[18] Christopher Rouff, Mike Hinchey, Walt Truszkowski, and James Rash. Formal methods for autonomic and swarm-based systems. In *Proc. ISoLA*, pages 100–102, Paphos, Cyprus, 2004.

[19] Gopinadh Sirigineedi, Antonios Tsourdos, Brian A. White, and Rafal Zbikowski. Modelling and verification of multiple UAV mission using SMV. In *Formal Methods for Aerospace*, pages 22–33, Eindhoven, The Netherlands, 2009.

[20] Alan F.T. Winfield, Jin Sa, Mari-Carmen Fernandez-Gago, Clare Dixon, and Michael Fisher. On formal specification of emergent behaviours of swarm robotic systems. *International Journal of Advanced Robotic Systems*, 2(4):363–370, 2005.