A Multi-Agent Systems approach to Test Generation for Simulation-based Autonomous Vehicle Verification

Greg Chance[1], Abanoub Ghobrial[1], Severin Lemaignan[2], Tony Pipe[2], Kerstin Eder[1]

***Abstract*—Simulation-based verification is beneficial for assessing otherwise dangerous or costly on-road testing of autonomous vehicles (AV). This paper addresses the challenge of efficiently generating effective tests for simulation-based AV verification using software testing agents. The multi-agent system (MAS) programming paradigm offers rational agency, causality and strategic planning between multiple agents. We exploit these aspects for test generation, focusing in particular on the generation of tests that trigger preconditions of assertions. On a small example we show that, by encoding a variety of different behaviours respondent to the agent's perceptions of the test environment, the agent-based approach generates twice as many effective tests than pseudo-random test generation. It is both scalable and efficient. Moreover, agents can be encoded to behave more naturally without compromising the effectiveness of test generation. Our results suggest that generating tests using testing agents allows engineers to reach edge cases and rare events more easily. Our results suggest that generating tests using testing agents significantly improves upon random and simultaneously provides more realistic driving scenarios.**

***Index Terms*—Test Generation, Simulation, Autonomous Driving, Verification, Multi-Agent System, Testing Agent**

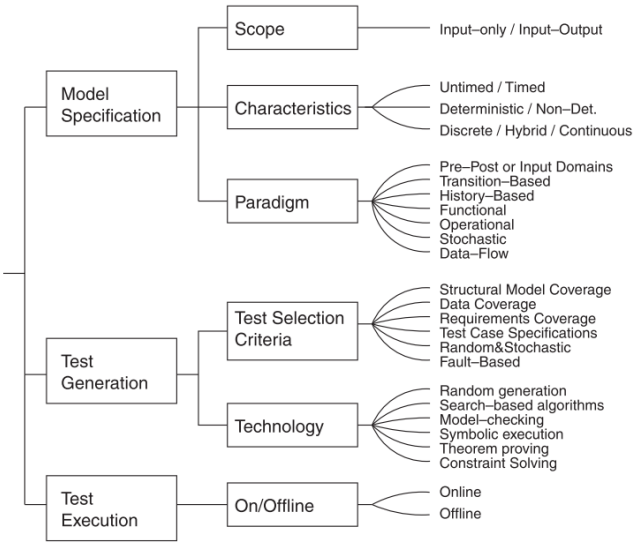

Fig. 1. Taxonomy of model-based test generation, from [12]

I. Introduction

VERIFICATION is the process used to gain confidence in the correctness of a system wrt. its requirements [**bergeron???**]. Testing is a technique that can be used to achieve this by showing that the intended and actual behaviours of a system do not differ and detecting failures against the requirements in the process [12].

Testing autonomous driving functions and safety critical scenarios in simulation can benefit from a fully controlled environment where road layouts, weather conditions and other traffic participants (GC: driving scenario parameters???) can be directed to achieve specific test targets. These tests may look to convince auditors of the functional safety of the vehicle or whether it complies with commonly agreed upon road conduct, such as the Vienna convention [13], which is usually interpreted locally by each country, e.g. the UK Highway Code [2]. In addition, there are also road traffic laws and penalties, e.g. [10], and unwritten rules or social conventions to follow. These may be culturally or geographically different, e.g. flashing headlights to override junction priorities to give way to another driver.

Semiconductor hardware verification can take up to 70% of the design effort [International Technology Roadmap for Semiconductors, Design Chapter, 2005 Edition. Avail- able at http://public.itrs.net] and similar to AV verification exhaustive testing is intractabe due to vast parameter space.

How do you generate interesting tests that cover the same level of testing as 100M miles of motorway driving? For the public to gain trust in autonomous vehicles (AV), the manufacturers must demonstrate that their AVs comply with, amongst many things, road safety requirements. While on-road testing will contribute significantly towards AV verification, testing in simulation has complimentary benefits as it offers a safe and effective environment.

Automation:

Testing in simulation enables many processes to be automated which may be convenient for ensuring compliance during version change, e.g. software updates or patches to the vehicle control system. But automation can also apply to not just the process but the method of test generation which is the focus of this paper.

Considering the AV as a device under test (DUT) the verification engineer is presented with a familiar set of tasks entail-

[1]Abanoub Ghobrial, Greg Chance and Kerstin Eder are with the University of Bristol, Bristol, UK {greg.chance, abanoub.ghobrial, kerstin.eder}@bristol.ac.uk

[2]Severin Lemaignan and Tony Pipe are with the Bristol Robotics Laboratory, University of the West of England, Bristol, UK {severin.lemaignan, tony.pipe}@brl.ac.uk

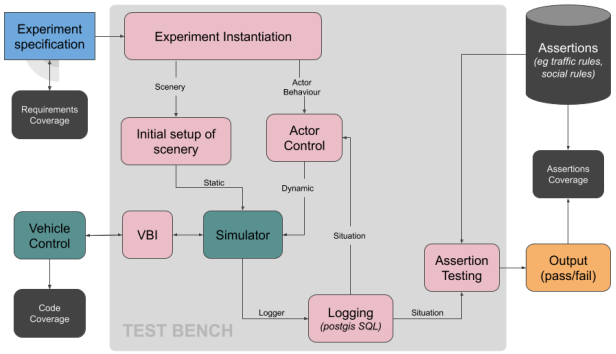

Fig. 2. Test bench design.

ing the discovery of bugs, corner cases, unanticipated events and to reach complete [**KE?**] coverage within reasonable time and cost. A suitable anaolgy is that of a responder which interacts with an abstracted level of the testing environment and as such only interacts passively (responds) to a recieved stimulus and whilst also fulfilling it's own objectives is also expected to adhere to legal protocol, which are the expected driving behaviours to that stimulus for the AV. Investigating the opportunity of agency to address the challenges of verifying the responder is the principle hypothesis of this research.

Phrase as a question

Our key contribution to the field is to propose a new test generation technique: agent based test generation. This technique will sit alongside and complement the techniques in the existing verification taxonomy [12]. The new method will form a new leaf node at the Test Generation-Technology branch of the existing taxonomy of model-based test generation, Fig 1.

This work proposes using an agent based model for test generation. This fits in within the existing taxonomy of model-based testing [12] under the test generation - technology branch.

more/stronger motivation - why agents? what is the challenge/problem?

mention how there are formal and informal rules of the road, both of which are valid testsRegulators would use the former to assess the safety case for the vehicle, whereas public opinion may be persuaded more by compliance of the latter. Trustworthiness will certainly be composed of both aspects of differing proportions depending on the audience.

The proposed test bench, see Fig. 2, is driven by a specification for the experiment which defines the scene and scenario [11] including all dynamic actors. The Vehicle Behaviour Interface (VBI) connects the AV controller to the simulator. It provides the simulator with the driving decisions of the AV and forwards updates on the scene to the AV controller. A geospacial database logs the AV and all other actors to enable post-simulation assertion checking as described in [**.**]

The experiment or test specification [**ref to the paper with all the terminology**] to the test bench may be generated in a number of ways familiar to the verification engineer. Tests may be manually generated by the verification engineer which will be both valid and accurate in producing the desired test conditions but at the highest cost. Random methods are usually employed at an early stage of testing to get coverage quickly but potentially suffer from generating invalid tests or tests that are not interesting, where interesting in this case refers to exercising the decision making processes of the AV. Model-based test generation sits between these methods in terms of validity of tests and the cost to produce them. Model-based testing requires a model that encodes the behaviour of the test environment then used to generate tests for execution.

A. *Multi-Agent System*

Georgeff and Lansky were key in the development of the belief-desire intention agent programming approach [7] and also in the early work of multi-agent systems [6]. Waters et al. describe improvements to BDI agents through modifying the intention selection method using priority-based schedulers. They propose *enablement checking* where an agent should not attempt to pursue an intention if there is no way to accomplish it and claim substantial benefits can be gained when 'running vulnerable programs in dynamic environments' [14]. This is an interesting approach and highly relevant to the AV verification domain potentially preventing unwanted emergent agent behaviour such as 'zombie attack'. Schut et al. argue that the intentions of a rational agent are not static and should reconsider their intentions if they cannot be acted upon [9]. Considering the degree of inaccessibility of the goal and cleverly considering intention reconsideration as an action the authors use POMDP to find the optimal intention reconsideration policy. Faccin et al. propose higher level intention selection using a meta-model to learn and predict plan outcomes [5] demonstrating positive experimental results.

Automatically generating stimulus that will increase functional coverage is a key challenge in simulation based verification and is a key vitue of constrained pseudo-random techniques. But such techniques are also far from completely automated, as the verification engineer must supply the suitbale constriants which become ever more complex in real driving scenarios. Feedback based coverage-driven generation (CDG) is a technique employing machine learning techniques to automatically generate tests based on rewards from a neural network or Markov Decision Process (MDP). Inductive Logic Programming based Coverage Driven Generation (ILP-CDG) has been shown to improve coverage closure by training on data generated during psuedo-random test generation and learning rules to constrain subsequent simulations [3]. Combining the BDI framework with an automated test generation approach leads to the idea of a software agent capable of generating tests. Intelligent agents have been used in the human robot interaction (HRI) domain with a coverage feedback driven generation approach using reinforcement learning to explore collaborative manufacturing [1]. The idea of a test agent has also been proposed by Enoiu et. al., [4] for regression testing althouhg more for test selection than generation, where agents decide what tests to execute and with what prioritisation and include agent messaging.

II. Related Work

Describe conventional methods for test generation for verification, e.g. formal methods, random methods, hand-written tests.

Include koopman, and stanford ISL

Include the latent logic approach

This work proposes using an agent based model for test generation. This fits in within the existing taxonomy of model-based testing [12] under the test generation - technology branch.

more/stronger motivation - why agents? what is the challenge/problem?

How will we achieve this? Set out the scene with the simulator and the test-bench structure – lead into how to the focus here which is to generate the tests - define scene, scenario and any other sim terms Utting et al. [12] present a taxonomy of existing model-based test generation approaches Discuss the ways that you can generate tests 1)use random 2) use engineers to write tests 3)AI/ML methods ...

There are many factors that may influence the choice of test generation for your application, here we choose to focus on accuracy and cost, see Fig**??**. Accuracy in this context is defined as the ratio of tests that exercise the targeted assertion compared to the total number of tests generated. In another way this is the efficiency of the test generation method in 'finding' coverage.

Cost in this context is defined as either computational costs or time put in to write the test by a verification engineer. As shown in Fig**??**, some methods may be relatively low cost with low accuracy such as a random method. Random based methods are very popular for AV verification (refs) with the idea being that enough low probability events over millions of miles of testing will give auditors the confidence that enough corner cases have been experienced.

The other end of the spectrum takes us to hand written tests by verification engineers or those with vested interests in specific scenarios, e.g. a local authority assessing safety of a new road layout. The test case written by the engineer will almost certainly achieve the desired coverage point but at a much higher cost than a random method.

In this paper we explore the use of a multi-agent system approach to test generation. In theory the MAS technique should achieve better accuracy than a random method and with a relatively small increase in cost.The former we can quantitatively measure in controlled tests where the latter is slightly more difficult to assess (e.g. development time) but here we elect to use computational time and number of lines of code. A full description of the assessment is given in Section **??**.

say why

III. Problem and Hypothesis

We want to verify the AV's suitability for deployment on the road by testing the autonomous driving functions against a set of assertions that an auditor or regulator may take as part of a body of evidence to assess the vehicle's safety case.

To gain trust the AV must demonstrate its ability to make the right decision and, as discussed in [**koopman**], for the right reason, although we will not go to that level of assessment in this paper. So how do we gain trust in this context? On-road testing is an obvious start and an essential part of testing albeit the most costly and time consuming. Testing in simulation is an appealing route many are now considering [**refs**] due to the level of control and alleviates many safety related restrictions to the type of tests that can be undertaken. There are the brute-force approaches to simulation [**ref**] where verification is sought simply through a high number of driven miles which has been estimated should be in the millions [**ref**] if not hundreds of millions [**ref**] and requiring substantial computing resources to complete within a 1-2 month time-frame [**ref**].

Do we discuss model-based test gen? Or formal methods?

both, so we can compare

Generating targeted test cases can be a more efficient way to get complete coverage for the system. By finding tests that exercise the complete decision making domain we can do away with 'unnecessary miles' of testing.

Finish saying that this is a conceptual proof and not a complete exploration of the concept, choosing only a single assertion may invite criticism but the goal here is to prove that with a small cost/outlay of engineering time we can generate agents that accurately generate tests. The next level would be agents that can act on multiple assertions and their intention selection is a feedback controlled from the coverage checker.

This is done to assess that the test generation conception can generate valid tests and how this compares to random based techniques.

A. Natural Behaviour vs. Edge Case

One could argue that the simulation environment presented to the AV should be as close to the real world it seeks to emulate as the optimum proving ground [ref]. In this case such efforts seek to reduce the *reality gap* [8] providing the most likely scenarios and agent behaviour. One approach is to monitor real traffic scenarios from traffic monitoring cameras, tracking individual vehicles, cyclists and pedestrians and from these build behavioural models for each road scene that you have traffic data for [ref latent logic]. This type of natural behaviour model uses data from a distribution of users that will include various styles of driving and road behaviour(?). An interesting way to further differentiate this distribution would be to assign personality traits to certain behaviours. Assigning personality traits to agents [15] in a simulation environment would be a way to bias the normal behavioural distribution towards a certain scenario of interest. For example, an egoist personality type may accept a lower time-to-collision (TTC) than more risk-adverse personalities resulting in scenarios that promote the AV to exercise decisions related to braking and collision avoidance. Over-representing certain personality traits in the simulation may result in scenarios where *the unlikely happens more often*.

It is also easy to see this approach could be expanded with other traits such clumsiness (pedestrian bumping into each other), social engagement level (inter-agent engagement) or propensity for distraction, e.g. engagement with mobile device, to further enrich the simulation environment.

This research proposes that agent based test generation (ABTG) can be used as an efficient and scaleable method for AV verification. Within existing multi-agent frameworks [ref JASON] agents are given specific goals and actions which are updated based on perceptions of the environment. The proposed implementation would work similarly, where agents (cars, pedestrians, cyclists, traffic lights etc.) are given goals that relate to the coverage task required. As such, simulated driving scenarios are not prescriptive at the outset but evolve with time based on the actions of the agents and the test vehicle.

For example, an agent could be tasked with assessing the emergency braking function of the autonomous vehicle. In which case the agent would promote behaviours that may result in the AV taking this action.

IV. Case Study

This section describes a case study to explore the proposed test generation method. The aim is to generate tests that exercise the precondition for a single assertion: collision avoidance. The precondition in this case states that for the associated decision making process in the AV to be exercised an entity in the path of the vehicle should be avoided either by braking or manouvering. A single assertion is chosen to isolate assessment of the test generation concept and not a demonstrate a complete verification process. The assertion chosen in this example is collision avoidance with a pedestrian and a successful test would be when the pedestrian enters the emergency braking zone of the DUT, i.e. within a 1s time-to-collision (TTC) window of the leading edge of the vehicle. The outcome of the test and the behaviour of the AV are not the focus here, but that tests are generated using the agent framework that satisfy the precondition for the assertion.

A. Test Agents

To assess if the agent based model of test generation is effective it is compared to random methods and repeated sufficiently for statistical power. A simple scene is chosen containing a straight road with a pavement either side represented in a grid world with a finite (1.0s) time interval. For this example pedestrian and test agent are synonymous. At the start of a test pedestrians are randomly located onto the pavement area and the AV is represented as a vehicle that drives at constant speed in a straight line and will not brake or turn. The test agents have differing levels of behaviour from simple random to more complex and directed, which is the focus of this section.

The different pedestrian actions can be split into two main classes: random and directed, see Table **??**. For the random class, the *random action* is where the test agent can make any random movement at each tick where the available actions are: do nothing (stand still), move up, down, left or right. For the *random behaviour* the pedestrian is instantiated walking along the pavement before the simulation starts and has only one action; to randomly choose when to cross the road. The random behaviour is included so that a comparison between crossing the road at a targeted or random time can be made.

The second class of actions are based on perceptions of the pedestrian and these agents are initialised walking along the pavement as in random behaviour. The *proximity* behaviour instructs the agent to cross the road when the AV is within a certain radius. Using the *intersect* behaviour, the agent calculates an intersection time and chooses to cross the road if the pedestrian can reach the intersection point before the AV has passed. The *election* behaviour ranks the probability of each agent to enact the intersect behaviour and elects the agent with the highest likelihood of success.

The *proximity* behaviour describes an agent, for this assertion, with suicidal tendencies that will step into the road whenever a car is nearby but is a highly improbably model of pedestrian actions. The *proximity* behaviour also suffers from the 'zombie effect' where all nearby pedestrians are drawn to the passing vehicle which may lead to the coverage point required but through an improbable scenario. The *intersect* behaviour is a more refined version of the proximity behaviour but may still result in multiple agents attempting to intersect the AV simultaneously whereas the *election* behaviour should limit this to a single agent.

The number of agents is also a factor to consider for this demonstration, too few and the likelihood of an agent triggering the assertion will be low and entirely dependent on the initial starting position in the test environment due to differences in speed. The maximum number of agents could be considered all the available grid locations (1.5m spacing) on the pavement without overlap. As the number of agents, nA, tends to this maximum the probability of hitting the coverage point using a random class increases significantly as more agents appear in the road with an increase in computation expense. As nA increases using more intelligent behaviour methods, suitable agents to complete the assertion should be found more readily resulting in shorter and hence more efficient tests. The range of nA explored was 1-10.

As with any model refinements could be made to improve coverage success, such as comparing direction of travel between the AV and the agent or modifying pedestrian speed, but this example is kept simple to illustrate the concept.

B. Test Environment

The testing environment, see Fig. 3 is a straight two-lane road 99m long and each lane is 6m wide. Pavements are on both sides of the road which are three meters in width giving a total size is 18m x 99m. The assumed AV velocity is 20 mph which is equivalent to 9.1 m/s that is rounded down to 9 m/s. The pedestrian velocity is 3 mph equivalent 1.4 m/s rounded up to 1.5 m/s. The map is discretised with 1.5 meter resolution for simple division into the AV and pedestrians velocities. Thus, in the discretised world the AV velocity is 6 unit/sec and the pedestrians velocity is 1 unit/sec. The total number of map cells is 12 x 66 = 792 cells.

The AV travels along the left hand lane of the road starting at cell $y = 0$ and travelling to the end, cell $y = 66$. If the assertion is triggered or the AV reaches the end of the road then the test is restarted. The AV occupies an area 4.5x3m equivalent to 4x2 cells. There are no other vehicles and the right hand side of the road is unoccupied.

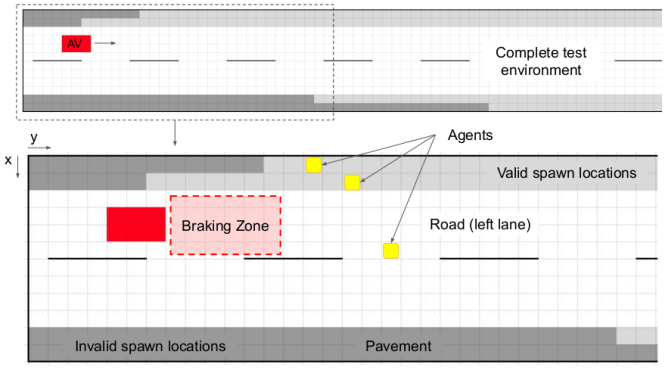

Fig. 3. Test environment at full scale (left) and in detail (right) including valid and invalid spawn locations for the pedestrians and the position and direction of the AV.

The environment is initialised with the agents spawned randomly on any valid pavement area (see note below). When the environment is reset new random locations are set for the agents. Control over random locations are set using a fixed seed based on the experiment number (1-1000) and so tests are repeatable between agent types, i.e. An experiment with 5 agents using a random method with have the same starting locations as using any other method which ensures valid comparison between agent types.

Valid test if agent found in 'braking zone' of the ave which extends 6 units forwards of the leading edge of the vehicle. Any agent found in this area is defined as a valid test and the environment is reset.

Some areas of the environment are impossible for the agents to intersect the AV and are therefore invalid spawn locations. On the left hand-side of the left pavement a pedestrian needs to move three cells to the right to reach the left part of the centre of the left lane. This is equivalent to three seconds. In this time the AV crosses 18 units. Therefore, the first 18 cells of the left-hand side of the left pavement are dead-zone where pedestrians cannot intersect the car if spawn inside this zone. Similarly, the first 12 cells of the right-hand side of the left pavement, the first 36 cells of the left-hand side of the right pavement and the first 54 cells of the right-hand side of the right pavement are all invalid spawn locations.

the following sectiong has changes missing

C. Scoring

Each agent behavioural method will be repeated 1'000 times and the number of valid tests counted. To reiterate, a valid test is when a pedestrian intersects the AV and for simplicity we assume this is when they have the same coordinates. In an attempt to impose more realistic behaviour on the agents, a scoring system is used that penalises certain actions. A living cost is imposed on the agents to promote shorter tests and a penalty is given for agents that are in the road but not intersecting the AV.

The scoring system is:

- A reward of 20 is given if the agent intersects the AV.
- A penalty of -1 for each time step.
- A penalty of -7 for each time step spent in the road.

D. Simulation and Logging

Each of the agent behaviours were written into the Jason platform along with the test environment as described above. For each test a basic log of the agent actions, score and time to completion are recorded and then averaged for each type. A list of random start locations was created for the pedestrians and this was done for each value of nA explored, i.e. 1-10. The list was then used to spawn the pedestrians for each behavioural type to ensure the initial conditions were identical.

Ultimately the scoring should reflect the benefits of the methods against the needs of test generation: accuracy, cost and robustness. We want tests that are accurate (valid tests, collects coverage) and sufficiently low cost (CPU hours, engineering time). If tests are robust they are adaptable to real-time events, e.g. AV slows down around bend requiring agents to recalculate intersection.

V. Results

The results section is split into 4 parts; Accuracy is the ratio of sucessfull tests the agents generated as a ratio of all tests, Score is a measure of how natural the agents behaved, Combined Score combines accuracy with score, and Time is how long the agents took to generate the test in both simulation ticks and wall clock time. Both Score and Time have distributions associated with them and as such confidence intervals are provided.

A. Test Accuracy

Test generation accuracy, defined as the number of tests that have activated the pre-condition for the asserion as a ratio of all tests, are shown for each agent type in Fig. 4. The *random action* (RA) and *random behaviour* (RB) have lower accuracy compared to directed agents when $nA < 10$. For high nA the accuracies converge mostly due to a saturation of agents (explained above). Fig. 4 also shows that for $nA = 1$ a directed agent outperforms a random agent by more than 2:1. The *election* agent generates a slightly higher accuracy than the *proximity* agent but only by 2.2%. However, for $nA = 2$ and above the *proximity* agent has a 10% increase over the election agent which is because this agent has multiple attempts to trigger the pre-condition whereas the election agent only has one.

Agent density vs. accuracy - can be used to det how many agents required for map size

B. Test Score

For tests that activate the pre-condition the average agent score is compiled including 95% confidence interval range, see Fig. 5. The maximum theoretical score of any agent is 94 which includes 100 points for a successful test sub tracting a living cost of 1 and road penalty of 5. For a single agent scores between agent types are similar as accuracy is being controlled for by including only the successful tests. As the number of agents increases the random class diverge from the directed class and the variance in the random agent score increases.

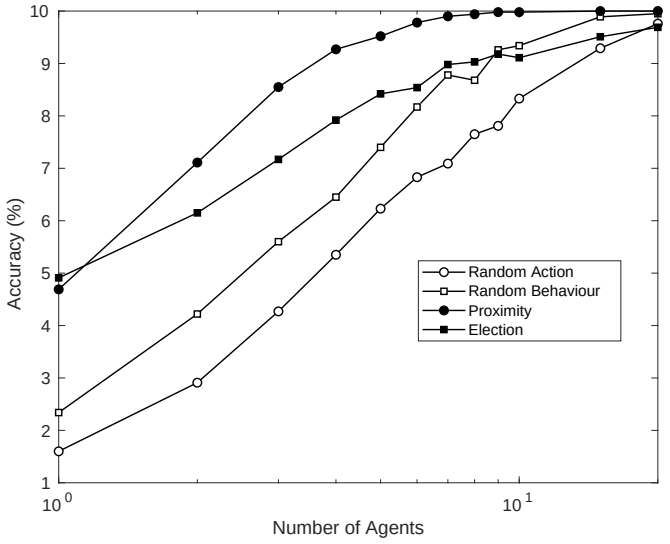

Fig. 4. Accuracy.

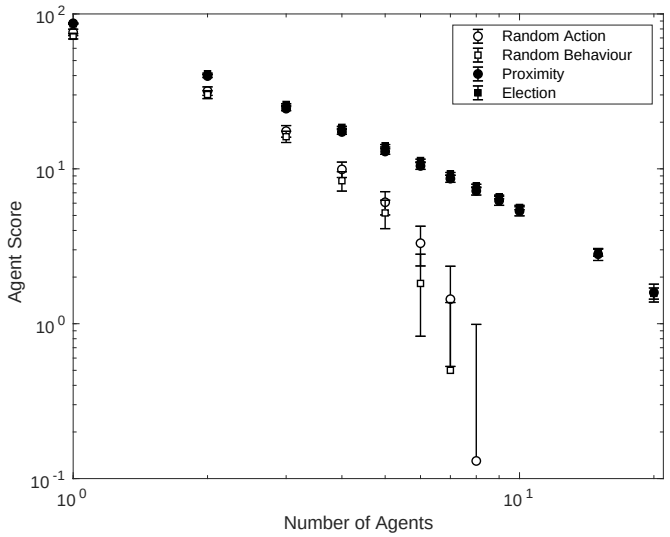

Fig. 5. The average agent score for successful tests with 95% confidence intervals for each agent type. Random class are hollow markers and directed class are filled. The maximum theoretical score is 94 points but this is only applicable to a single agent hence the declining average with increasing agent numbers.

why no variance for nA=1 in random? this is counter intuitive, it suggests random agents that succeed score only the highest value score!

As nA increases for the random agents, the number of agents found on the road also increases and hence average score drops rapidly, whereas the directed class are only found crossing the road when they deem fit and are on the pavement at all other times keeping the score much higher.

include the theoretical max score line

Note that the directed agent class are close to the theoretical maximum score, Fig. 5(dashed line). The high score for the $nA = 1$ random class is surprising especially as the error bars are small indicating that, although the accuracy is low, tests generated are relatively efficient, i.e. the pedestrian does not spend a lot of time in the road. No significant dfference between the directed class is seen in the score even with a high number of agents indicating both represent a similar level of natural behaviour, at least within the limited scope of this example.

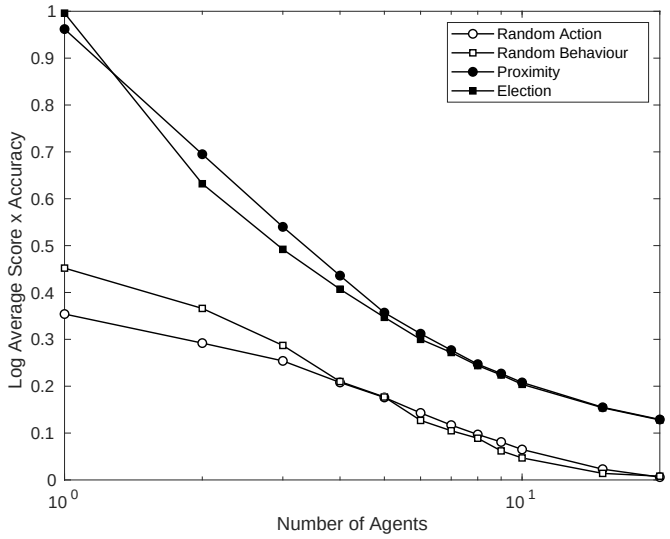

Fig. 6. The combined accuracy and score for each agent type.

C. Combined Score

As the score (above section xx.xx) controls for accuracy it can be misleading to interpret these results so a combined score is provided which is given by 1x10-3 * score * accuracy. This combined measure promotes scores that are attached to high accuracies, describing agents that can generate useful tests with natural pedestrian behaviour. Time could have been included as a numerator here but is similar in scope to the living penalty so not included.The normalised results, Fig 6, show that the directed agents are over twice as effective within this new combined definition than random agents for $nA = 1$ although this advanted drops rapidly with increasing agent numbers to reach a steady gap of around 12%. The *random behaviour* agent outperforms the *random action* for $nA < 4$ beyound which there is little difference. So if random is your choice, you might as well just use completely random unless you want to use low agent numbers. The *election* agent has the highest combined score for $nA = 1$ but higher agent numbers are favor the *proximity* agent up to $nA = 4$ beyond which there is no notable difference between the directed agent types.

Describe relation to theoretical maximal?

D. Time

For each successful test geneated the reported time was averaged over k-tests and compared across different agent numbers, Fig. 7. The directed agents consistently improve over the time taken by the random class for $nA = 1$ by around a single simulation tick and this trend continues as agent numbers increase. By $nA = 20$ the directed agents are 1.85 simulation ticks faster than the random agents. Overall the *proximity* agents find tests in the shortest time.

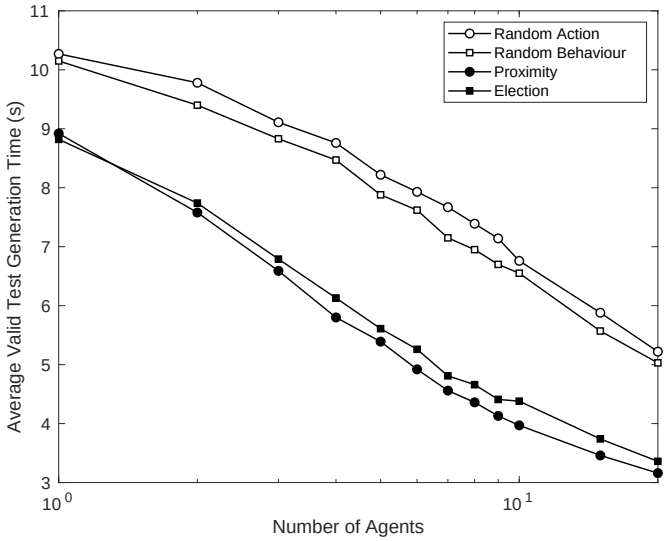

Fig. 7. The average time taken for an agent to find a successful tests.

change time y-axis to sim ticks not seconds, remove valid, add wall clock time to second y-axis, add error bars to time

E. Results Analysis

The results show that by nearly all the metrics and parameter combinations discussed above, (accuracy, natural behaviour score, test generation time) the directed agents outperform the random agents. This shows that even a small amount of intelligence can be a distinct advantage over random techniques. But what was the cost of developing these agent behaviours? And is there a limited payoff to gain significantly superior intelligence for really complex agent behaviour? Comparing the lines of code for the agent action definition can help to weigh the investment in agent complexity. Lines of code for each are below and the combined score for $nA = 1$ is in brackets:

- *random action* = 23 (0.35),
- *random behaviour* = 82 (0.45),
- *proximity* = 86 (0.96),
- *election* = 235 (1.0)

indicating that for a small investment the *random action* agent can be improved significantly with 4x the initia code to improve it's combined score to near the theoretical maximal. To improve further with the *election* agent took 10x the initial code with minmal score improvement suggesting this was not worth the investment and also considering for $nA > 1$ *proximity* proved more effective.

This also raises the question of how comlex the agents sould be? The analysis above would suggest that the level of agent complexity should be considered carefully as a simpler level of intelligence could be more beneficial.

VI. Conclusion

The MAS programming paradigm offers rational agency and strategic planning to software agents that have been exploited in this research for test generation. On a small example we show that, by encoding a variety of different behaviours respondent to the agent's perceptions of the test environment, the agent-based approach generates twice as many effective tests than pseudo-random test generation and agents can be encoded to behave more realistically without compromising the effectiveness of test generation. Our results suggest that generating tests using testing agents significantly improves upon random and simultaneously provides more realistic driving scenarios.

A. Future Work

In this small case study there is only a single assertion considered. But how agents behave when multiple assertions or *desires* exist will be key to the success of this method. This could be achieved with a number of methods such as MDP and ANN. As discussed in [3] agents that have their goal selection modified based on coverage feedback would be an important step in this direction. This also fits in with the feedback reward of many such architectures, e.g. rewards for MDP and backpropagation for ANN.

Composing agents with a feature based representation of their environment is also a direction to ensure agents can scale to large physical maps and also adaptable to new features as needed. This approach also fits in with most ML techniques. Including personality to agents is also another avenue that could provide insightful as a tuning parameter. Conventionally agressive (or egoist) behaviour may seem the most dangerous and therefore require oversampmling. But there could also be merit in exploring more risk-advers behaviours. There is also the possibility to have have combination of random and directed agents to ensure the initial coverage is satisfied, then add more intelligent agents for hole coverage.

Acknowledgment

This work was performed under the framework of the ROBOPILOT and CAPRI project (Innovate UK, Grant no.s xxxxx and xxxxx).

code is available at github:.

References

[1] D. Araiza-Illan, A. G. Pipe, and K. Eder. "Intelligent agent-based stimulation for testing robotic software in human-robot interactions". In: *ACM International Conference Proceeding Series* (2016), pp. 9–16.

[2] A. O. CODES. "HIGHWAY CODE". In: (2011).

[3] K. Eder, P. Flach, and H. W. Hsueh. "Towards automating simulation-based design verification using ILP". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 4455 LNAI (2007), pp. 154–168.

[4] E. P. Enoiu and M. Frasheri. "Test Agents: The Next Generation of Test Cases". In: *2nd IEEE Workshop on NEXt level of Test Automation* (2019).

[5] J. Faccin and I. Nunes. “Bdi-agent plan selection based on prediction of plan outcomes”. In: *2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*. Vol. 2. IEEE. 2015, pp. 166–173.

[6] M. Georgeff. “Communication and interaction in multi-agent planning”. In: *Readings in distributed artificial intelligence*. Elsevier, 1988, pp. 200–204.

[7] M. P. Georgeff and A. L. Lansky. “Reactive reasoning and planning.” In: *AAAI*. Vol. 87. 1987, pp. 677–682.

[8] N. Jakobi, P. Husbands, and I. Harvey. “Noise and the reality gap: The use of simulation in evolutionary robotics”. In: Springer, Berlin, Heidelberg, 1995, pp. 704–720.

[9] M. Schut, M. Wooldridge, and S. Parsons. *The theory and practice of intention reconsideration*. 2004.

[10] *UK Road Traffic Act 1988*. http://www.legislation.gov.uk/ukpga/1988/52/contents. Accessed: 2019-10-03.

[11] S. Ulbrich, T. Menzel, A. Reschka, F. Schuldt, and M. Maurer. “Defining and Substantiating the Terms Scene, Situation, and Scenario for Automated Driving”. In: *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC* 2015-Oct (2015), pp. 982–988.

[12] M. Utting, A. Pretschner, and B. Legeard. “A taxonomy of model-based testing approaches”. In: *Software Testing, Verification and Reliability* 22.5 (2012), pp. 297–312.

[13] *Vienna Convention on Road Traffic*. https://treaties.un.org/Pages/ViewDetailsIII.aspx?src=TREATY&mtdsg_no=XI-B-19&chapter=11&Temp=mtdsg3&lang=en. Accessed: 2019-10-03.

[14] M. Waters, L. Padgham, and S. Sardina. “Improving domain-independent intention selection in BDI systems Article in Autonomous Agents and Multi-Agent Systems · Improving Domain-Independent Intention Selection in BDI Systems”. In: (2015).

[15] A. Zoumpoulaki, N. Avradinis, and S. Vosinakis. “A Multi-Agent Simulation Framework for Emergency Evacuations Incorporating Personality and Emotions”. In: *Hellenic Conference on Artificial Intelligence. Springer, Berlin, Heidelberg, 2010.* 2010, pp. 423–428.